DISSERTATION

On the Practicability of Information-Theoretic Cryptography

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines Doktors der technischen Wissenschaften unter der Leitung von

O.Univ.Prof. Dipl.-Ing. Dr.techn. Harmen R. van As
E389 - Institute of Telecommunications
Technische Universität Wien

und

Dr. Stephan Krenn
Center for Digital Safety & Security
AIT Austrian Institute of Technology

eingereicht an der Technischen Universität Wien
Fakultät für Elektrotechnik und Informationstechnik

von

Dipl.-Ing. Thomas Lorünser
Matr.-Nr. 9326233

Wien, im Jänner 2023

Abstract

Cryptography is considered the strongest technical control to protect data, but state-of-the-art methods suffer from shortcomings when applied in cloud computing or the Internet of Things. New approaches are needed enabling more agile data handling, supporting computations on encrypted data, and providing long-term security, even against quantum computer attacks.

In this thesis we present research results in building long-term secure but practically efficient protocols and systems based on cryptography with information-theoretic security (ITS) for modern cloud-based applications. It brings together old and new technologies from the world of information-theoretic cryptography to overcome limitations of standard cryptographic approaches and enable end-to-end security in modern application scenarios. The focus is on secret sharing, multiparty computation and quantum key distribution, which are well known to the cryptographic community but not broadly applied in practice.

In essence, we explored the possibilities to build ITS solutions for data storage, data processing and communication. Nevertheless, pure ITS is not always necessary nor possible, thus we also study combinations with computational (but also quantum-safe) symmetric primitives where appropriate for better efficiency. This thesis comprises three main parts each containing individual contributions.

Firstly, the problem of secure cloud storage and data sharing is addressed. A novel architecture for a secure distributed multi-cloud storage is presented based on the combination of secret sharing with a Byzantine fault-tolerant (BFT) protocol. To cope with performance problems encountered in the first proof-of-concept, a performance model was developed and results from extensive simulations of the networking layer are presented. We also explored and optimized encoding performance for secret sharing in software and show the potential for hardware acceleration. Additionally, to also support means for data integrity monitoring we present an easy to realize and low-cost auditing approach for the developed storage system. The technique is based on batching which has also been extended further to generic batch verifiable secret sharing.

Secondly, we present efficient solutions for privacy preserving data processing based on ITS flavors of secure multiparty computation (MPC). Fortunately, ITS-MPC relies on secret sharing for encoding and thus nicely extends the previous work on secure storage. We compared most relevant software frameworks and did intensive performance testing, revealing only limited scalability of the technology for more advanced computations, especially with respect to the number of MPC nodes. Therefore, we propose the use of verifiable MPC to build privacy preserving data markets. By combining MPC with compatible zero-knowledge protocols (ZKP) we were able to demonstrate an end-to-end verifiable but privacy preserving market platform for smart manufacturing which can efficiently perform auctions with a large number of participants. We also explored the possibility to run more elaborated market mechanisms based on optimization and achieved very favorable results for a use case in air traffic management.

Thirdly, regarding secure communication this thesis presents results achieved in researching some particular aspects of quantum key distributions (QKD). A very efficient algorithmic approach for timing synchronization between QKD peers is presented which helped to free an optical channel in a QKD system developed at AIT. To overcome the problem of expensive compute hardware to run QKD post-processing on device, we introduce the novel idea of offloading post-processing from the device in a secure way and prove that it is possible to securely outsource information reconciliation to a single server for the case of direct reconciliation. Additionally, we also show a negative result for an efficient authentication protocol in QKD already proposed in 2004, which we were able to fully break with the method presented in this thesis. Finally, we discuss possibilities to integrate QKD with communication systems and report a real-world demonstration of the combination of secure storage with QKD to achieve information-theoretic security from end-to-end in a medical use case.

Acknowledgement

I am deeply grateful to Professor Harmen R. Van As for giving me the opportunity to conduct my PhD and his continuous support over the many years. He hosted me during my sabbaticals and offered an open environment to develop and to do research with the academic freedom I wanted, while giving valuable advice and guidance.

Simultaneously, I would also like to give special thanks to Stephan Krenn for the outstanding support as my local supervisor at AIT Austrian Institute of Technology, where I was mainly working during my PhD studies. His support enabled me to work at the interface of communication technology and applied cryptography, and he motivated me to follow my interest and curiosity.

During the course of my studies my research was embedded into various research projects of different scale. Some of them were particularly designed by me to finance my PhD research and allowed me to follow my passion, but also nibbled away some time for administration and management. However, during these projects I met so many interesting and inspiring people that I can say with confidence it was worth the effort.

I would also like to thank my former colleagues at the TU Wien as well as my collaborators at AIT Austrian Institute of Technology for the many fruitful discussions and support, particularly Florian Wohner for his helping hand and invaluable knowledge in software engineering, and Christoph Striecks for his support on batching for verifiable secret sharing, as well as Benjamin Rainer for the collaboration on BFT performance.

I am also very grateful to my parents, who have supported me throughout my university studies in the first place, and taught me to persistently follow my curiosity, even if it takes longer or requires detours.

Last but not least, I would like to thank my wife Ulrike and my children Mia-Sophie and Matheo for their endless support, understanding and patience during the last years. Without their love and dedication to our family I would have not been able to undertake this exceptional journey.

Contents

1 Introduction **1**
1.1 Motivation and Challenges . . . 1
1.2 Background and Research Context . . . 3
1.3 Relevance and Research Question . . . 4
1.4 Research Approach and Methodology . . . 5
1.5 Outline and Contributions . . . 7
1.6 Background and Preliminaries . . . 9

2 Securing Data at Rest **19**
2.1 Introduction . . . 19
2.2 New Architecture for Secure Distributed Storage . . . 20
2.3 BFT Performance Modelling and Optimization . . . 35
2.4 Simpler Configuration and Hardware Acceleration . . . 58
2.5 Efficient Privacy Preserving Remote Data Checking . . . 70
2.6 Batch Verifiability Against Malicious Clients . . . 83
2.7 Discussion . . . 97

3 Privacy Preserving Data Processing **101**
3.1 Introduction . . . 101
3.2 Performance Comparison of two MPC Frameworks . . . 102
3.3 A Novel MPC Based Platform for Data Markets . . . 118
3.4 Verifiable MPC Solver for the Assignment Problem . . . 134
3.5 Discussion . . . 153

4 Securing Data in Transit **157**
4.1 Introduction . . . 157
4.2 Timing Synchronization for QKD . . . 161
4.3 Offload Post-Processing from QKD Nodes . . . 171
4.4 QKD Authentication with Non-ITS MAC . . . 190
4.5 QKD Integration and Discussion . . . 201

5 Summary, Conclusion and Outlook **207**

6 Bibliography **211**

Chapter 1

Introduction

1.1 Motivation and Challenges

Information and communication technologies (ICT) are driving the next big revolution and digitalization will be at the heart of the transformation to the information society. A report by the World Economic Forum [67] identified six megatrends in the field of ICT which will shape the future:

- Ubiquitous and cloud computing,
- Internet of Things,
- Artificial intelligence and big data,
- Sharing economy and distributed trust,
- The digitization of matter and cyber physical systems, and
- Peoples' interaction with the Internet.

The huge expectations in the transformation to the information society is due to the rapid progress made in the development of computer hardware, software and communication technologies which enable a more connected world. However, the new approaches also come with many new challenges and risks. Especially security and privacy topics are considered extremely important for emerging data driven ecosystems but are unsolved in many aspects.

To illustrate the importance of security and privacy we show two examples. Firstly, cloud computing changes the way we are accessing ICT resources. It enables us to consume remote resources on demand and to scale in real-time via a self-service paradigm. However, from a security standpoint, cloud computing raises many interesting questions and challenges the status quo in data sovereignty. In essence, data is stored and processed in an external infrastructure which makes protection of data harder from a data owners' point of view. Furthermore, the intrinsic multi-tenancy of cloud computing together with the broad connectivity found in the cloud introduce many new threats. This combination ultimately leads to increased security risks for data managed in the cloud, e.g, as pointed out in [68, 69].

Solving security challenges in connected and dynamic environments requires novel approaches, especially if cryptography is considered, because end-to-end security cannot be easily achieved anymore with existing methods [41]. Nevertheless, end-to-end security would be very appealing in the cloud context, because if data is stored outside the own infrastructure, it is much harder to protect it over the whole lifecycle. For example, if it cannot be guaranteed that data is deleted after usage, it has to be protected with additional cryptographic means to be long-term secure, which is challenging with the state-of-the-art methods from cryptography. Many of current cryptographic schemes are susceptible to quantum attacks and may even be broken retrospectively, due to the progress in hardware development. While post-quantum (PQ) secure schemes exist for basic primitives, they are not yet fully mature for complex primitives, and PQ schemes still need to pass a test-of-time. In summary, a multitude of new problems arise with cloud usage, especially with respect to information security and privacy [70, 71].

Secondly, the Internet of Things (IoT) paradigm is also pushing towards increased data pooling and sharing in larger ecosystems. A characteristic of IoT applications is that sensors and actors are connected over cloud infrastructure to enable interaction with the physical world, i.e., they form cyber-physical systems. Due to the heterogeneity and distributed nature of IoT application data protection gets even harder compared to pure cloud outsourcing and novel cryptographic solutions are very desirable to cope with the large attack surface IoT systems have.

Additionally, it is nowadays a fact that the market for cloud computing and IoT backend systems is dominated by the few big hyperscalers which are forming an oligopoly. In this situation protection of personal data as well as business assets becomes essential to regain sovereignty for local industries and economies. Therefore, we believe that future data spaces for seamless and collaborative exploitation of data also require new cryptographic solutions, supporting richer functionality and long-term security while protecting the data from end-to-end.

In essence, a common trend in IT is the shift to systems with increased complexity, connectivity and dynamicity. However, if these three characteristics (sometimes called the trinity of trouble) come together, security becomes particularly challenging. The comprehensive use of cryptography can help in this situation to protect data from cyberattacks or even unintentional data breaches, e.g., by misconfiguration. The research in this thesis was motivated by this fact. Thus, novel practical and efficient cryptographic solutions were researched, to be used in emerging use cases like data markets for improved security and data sovereignty. Ideally, the proposed solutions are able to re-establish end-to-end security in heterogeneous, large, interconnected infrastructures, also in the long run. In particular, we focus on application of primitives from information-theoretic cryptography and analyze the practical aspects and performance supporting real industry driven use cases.

1.2 Background and Research Context

Cryptography is considered the strongest technical control to protect data. However, cryptography is not easy to use and deploy. Even worse, the state-of-the-art suffers from shortcomings with respect to cloud and IoT usage. The novel ICT paradigms require completely new approaches enabling more agile data handling, ideally by allowing operations and computations on ciphertext. Additionally, scalable quantum computers will be able to compromise current cryptographic methods which base their security on the hardness of number theoretic problems, thus long-term security is in question for these methods. Even if it is currently not possible to build scalable quantum computer and it is unclear when they will be available, the threat to data security must be already addressed today. As an example, consider ciphertext stored in the cloud which is leaked to an adversary who stores them for future quantum attacks. In the following we describe some background on the three major challenges driving the research in this thesis.

Limited functionality. The main cryptographic methods used in practice typically provide secure encryption/decryption functions for confidentiality protection and authenticity by digital signatures or message authentication codes. However, the methods work in an all-or-nothing fashion, i.e., decryption reveals the full plaintext and signatures only verify complete data. This was sufficient for many decades, but modern cloud-based data sharing and collaboration patterns for IoT applications require richer functionalities, e.g., only selectively sharing views on data or statistics about joint data sets. Therefore, data has to be decrypted in intermediary systems—the cloud—to be processed in plaintext, if used as input for more complex operations. This undermines the basic principle of end-to-end security, which was considered a major milestone in the development of cryptographic security.

Novel cryptographic methods like fully homomorphic encryption or multiparty computation would allow for processing of encrypted data and could therefore help to solve the problem, however, because they are considered inefficient and not practical they are rarely used in practice. Additionally, secret sharing, a rather old cryptographic scheme which can even provide information-theoretic security for data, could be used to achieve long-term security against all kind of future attacks, but is also not considered practical.

Moreover, secret sharing is also compatible with certain multiparty computation protocols. Thus, it can be directly used to store MPC input or output data. This is also a reason we selected this two primitives for our research. In this thesis we address both, encrypted but agile solutions for data storage and sharing among multiple stakeholders based on secret sharing and means for data processing based on MPC. Both can be instantiated with information-theoretic security, and both complement each other well when applied in decentralized data spaces.

Quantum computer threat. Quantum computers impose a tremendous threat to asymmetric cryptography used today and triggered a whole new line of research on quantum-safe cryptography. To cope with the situation two research strands are developing quantum-safe technologies. On the one hand, in post-quantum cryptography replacement algorithms for digital signatures and asymmetric encryption schemes are developed which are expected to be intractable for quantum computers. On the one hand, quantum key distribution (QKD) emerged as a novel method offering ITS key exchange, which is not possible from public key cryptography.

In this work we focus on the latter, because it is information-theoretic and therefore ideally complements the other technologies, i.e., secret sharing and multiparty computation. The author also spent significant time of his early career in the development of practical QKD systems and contributed to the topic on many levels. However, in the thesis a focus is given on some particular functionalities which have been researched recently and were considered important for practical application of the technology.

Furthermore, besides looking for direct quantum-safe replacements of affected cryptography, the we investigating the application of ITS cryptography more broadly and show how it can be used beyond communication, thus leading to quantum-safe systems and platforms. In particular we study secret sharing methods to build long-term secure multi-cloud storage and also research practical aspects on privacy preserving computation on the data for advanced cloud-based collaboration.

Deployment difficulties. Interestingly, the direct application of ITS cryptography in the context of modern applications worked very well for the problems and use cases identified. The proposed solutions achieved practical performance, were rich in functionality and provided long-term security. However, because secret sharing and MPC rely their security on the non-collusion assumption the deployment is very important for the technology and operators have to understand the principles and basic approaches for both, to achieve high security [22] for reasonable cost [23]. Furthermore, estimating the real performance which can be achieved in concrete scenarios and exploring the limits on scalability turned out to be more difficult and makes the application of the technologies still very challenging. Our research was therefore accompanied with detailed scalability and performance analysis as well as evaluation of adequate configurations and the design of new add-ons to achieve good performance, e.g., by transport layer optimization for storage or extensions for public verifiability in MPC.

1.3 Relevance and Research Question

The thesis present research results in building long-term secure but practically efficient protocols and systems for modern cloud-based data sharing and collaborative IoT applications. It brings together different old and new technologies from the world of information-theoretic cryptography to overcome problems with existing cryptographic approaches suffering from functionality and long-term security issues. Furthermore, the pure ITS technologies are enriched with additional methods and features relevant to support emerging data spaces. The goal was to research and overcome inhibitors for the use of information-theoretic cryptography and to show how it can help to design modern end-to-end secure cryptographic solutions.

Thesis statement: Methods and tools with built-in information-theoretic security to achieve long-term secure systems will be extremely valuable for the handling of personal and sensitive digital information in the future. Future ICT solutions must be designed to support outsourcing to less trusted infrastructure and collaboration between mutually distrusted parties, and data must be protected over the full lifecycle and ideally from end-to-end. Especially, primitives which naturally support decentralized approaches, like secret sharing, multiparty computation and quantum key distribution can be realized in practice and used to build novel cloud-based applications with high security, if applied and deployed in the right way. However, they need to be adapted, extended and integrated into higher-level protocols and systems to make best use of them and to be practically efficient.

Research question: How can information-theoretic cryptography be exploited to protect data in cloud-based data spaces to provide long-term security and improve data sovereignty?

Sub-question: How scalable and performant are the concepts and which methods can be used to overcome bottlenecks or shortcomings encountered for selected real-world applications?

Sub-question: How can they be combined with existing methods and tools to increase efficiency without sacrificing the good long-term security aspects, i.e., maintaining their quantum-safe property?

1.4 Research Approach and Methodology

In this thesis we focus on the use, extension and integration of information-theoretically secure cryptographic methods with practical efficiency in modern use cases. We show that the use of such technologies can not only lead to better long-term security but also improve functionality in modern interconnected ICT systems. However, we also researched ways to extend our ITS systems where necessary, e.g., to improve efficiency or allow for coexistence with less restricted data, but without giving up on the main properties at the core and only using quantum-safe tools. In essence, we explored the possibilities of information-theoretic cryptography for data storage, processing and communication in modern application scenarios and with practical performance.

In the information-theoretic setting, the adversary is modeled as being computationally unbounded, i.e., no restrictions on the computational capabilities are assumed. Thus, security does not rely on any unproven complexity assumptions; rather, it is "information-theoretic". Two types of unconditional security which are distinguished:

(a) *Perfect security:* Very informally, here the result of a real execution of the protocol with a real adversary must be exactly the same as the result of an ideal execution with a trusted party and an ideal-world adversary/simulator.

(b) *Statistical security:* Here, the result of the real protocol execution needs "only" be statistically close to the result of an ideal execution, but with negligible probability.

In cryptography only a few ITS primitives are known, with one-time pad (OTP) being the most prominent one. OTP is used for perfectly secure communication and per se not useful, because it requires a secure channel to start with. However, it is frequently used as sub-protocol and its introduction was groundbreaking.

Secret sharing is another important primitive, which also introduced the concept of threshold cryptography. It can be used to distribute trust among many parties and is getting more and more important in modern ICT systems, where a single trusted entity who centrally manages private keys cannot be relied on anymore. Interestingly, secret sharing is also an important building block to achieve secure computation on encrypted data. It has been shown that secret shared data held by different parties can be used to securely compute any function [72, 73] on them, assuming that more than $2/3$ of the parties are honest (or a simple majority if the parties are additionally given a broadcast channel). This method is called multiparty computation and another important method used in the thesis.

Finally, because secret sharing and multiparty computation rely on perfectly secure communication channels to exchange messages, an additional technology to protect the communication channels is needed to achieve fully information-theoretically secure systems. Fortunately, quantum key distribution provides ITS key exchange and can be used to realize secure communication. Thus, if all technologies are combined, decentralized collaboration platforms with information-theoretic security can be built, however, the technologies can also be used in their own right. In this thesis we research the practicability of the different technologies and how they can be potentially leveraged for higher security in cloud-based solutions. Nevertheless, pure ITS is not always needed or possible, thus we also study combinations with computational (but also quantum-safe) secure primitives where appropriate for better efficiency.

The research presented in this thesis was conducted during participation in three publicly funded collaborative research projects. The line of research comprises cryptographic solutions for communication, data storage and processing of data. The research was funded by the European Union Horizon 2020 research activity No. 644962 *PRISMACLOUD*, from the Austrian Research Promotion Agency *FlexProd*, and the SESAR

Joint Undertaking under Horizon 2020 grant agreement No 890456 *SlotMachine*. The author was contributing as key researcher to the respective projects and also coordinated two of them. The knowledge gained during these collaborations was an invaluable experience and also influenced research direction of this thesis on the way.

Especially input from industry partners was helpful in defining realistic use cases and for evaluation of the solutions. Additionally, we were able to experience the difficulties encountered in researching and designing real-world cryptographic solutions in an interdisciplinary team at first hand. Integrating cryptography into applications is often hampered by a missing common language between different people involved (cryptographer, software architect, software developer, security experts, etc.). Therefore, we also developed a new development methodology [28] along the way to improve communication between different stakeholder which is not part of this thesis. The author also contributed to research projects in quantum key distribution and gained a lot of experience on the technology [49, 53] which is also not directly part of the thesis. Nevertheless, this experience helped to identify the topics included in this work and to assess the potential for combination of the technologies.

1.5 Outline and Contributions

This thesis comprises three main parts (Section 2-4) each containing individual contributions. Besides that, in the current section the problem and research challenges are discussed in this section, and conclusions are given in Chapter 5. The structure was naturally given by different stages in the data lifecycle, i.e., data storage, computation and communication. In the following we quickly motivate the structure and summarize results achieved in each chapter to guide the reader.

In Chapter 2 the problem of **secure (cloud) storage and data sharing** is addressed. In the early phase of studying the topic, a novel concept for a secure distributed storage was developed and demonstrated [1]. However, the first approach showed serious performance problems and was missing essential functions. This spawned follow up research to overcome identified shortcomings. The research results achieved are threefold.

Firstly, the performance problems of the initial solutions were analyzed in detail by modelling and simulation of the networking layer [2]. Secondly, simpler configurations were studied to improve in performance but also to reduce the cost and flexibility by only requiring passive storage nodes [3]. Therefore, we also studied optimized implementations for encoding and decoding performance in secure distributed storage and also report the results achieved by hardware acceleration of the core components [4]. The performance improvement by migrating to hardware implementations were substantial and pave the way for intra-data center usage with high throughput and short response times. Thirdly, to also support means for system auditing and integrity monitoring we investigated efficient protocols for remote consistency checking during operation. We present an easy to realize and low cost (computational and networking) auditing approach for our storage system [6], which was also patented [5]. The technique is based on batching and has been extended further to generic verifiable secret sharing [7].

Chapter 3 deals with cryptographic methods for **secure data processing on encrypted data**. In particular we investigate practical aspects of multiparty computation (MPC), which can also offer information-theoretic security guarantees. The usage of MPC is somewhat an extension of our approach on secure distributed storage and can be integrated with it, because ITS protocols for MPC also rely on secret sharing. A huge body of work exists on MPC and progress in the last decade led to practical protocols and implementations. However, achievable performance for concrete applications is not easy to predict and depends on many factors as shown in [8], e.g., the impact of network imperfections. We also found the security guarantees provided by standard MPC not satisfactory, if many stakeholders are involved in the computation, because of the limited scalability of MPC for advanced computational problems. In researched applications for novel data markets, we still required clients to delegate the computation to an external MPC system, because not all users could run their own MPC node. Thus, additional assurance was required from a users' perspective, which motivated the development of a new platform with built-in public verifiability and end-to-end authenticity [9]. We also explored possibilities to use more elaborated market mechanisms based on optimization and achieved very favorable results in settings with three to four nodes [10].

When it comes to **secure communication**, we looked into particular aspects of quantum key distributions and present the achievements in Chapter 4. Firstly, we developed a very efficient algorithmic approach for timing synchronization on the quantum channel. It was essential to enable free-space applications as demonstrated in [11] and helped to free an optical channel in fiber-based systems. The developed algorithms are still part of the original AIT QKD software stack [44] as pre-sifting module. Additionally, to overcome the problem of expensive investments in compute hardware to run QKD post-processing on the device, we developed a new method to offload it in a secure manner, enabling new use cases and more flexible deployment options for QKD. The work also resulted in a patent [12]. Finally, we worked on certain aspects of QKD protocols to make them more efficient in terms of key usage on the classical communication layer or in being more flexible in deployment. In fact, we introduced the novel idea of offloading post-processing from the device in a secure way [13] and prove that it is possible to securely outsource information reconciliation to a single server for the case of direct reconciliation. Additionally, we study MPC implementations for cases where single server offloading is impossible. Another line of research was already started early in [64] to improve the efficiency of QKD channel authentication by leveraging non-ITS message authentication codes (MAC). However, the developed protocol had some issues and researchers already broke certain aspects of our proposal [55]. Nevertheless, it was during this thesis that we were finally able to fully break our own approach and therefore showed that non-ITS authentication should not be used in such systems after all [14], an interesting research result in its own right.

1.6 Background and Preliminaries

Before presenting the results in the following chapters we are quickly introducing most relevant concepts used in the work. Our intention is to analyze and research practical aspects of information-theoretically secure cryptographic schemes. Therefore, it is important to understand the basic notions in the field and to introduce the most important primitives used in the thesis.

1.6.1 Relevant Cryptographic Primitives and Protocols

The main idea of this thesis is to study practical aspects of secure systems leveraging information-theoretic cryptography at the core. The main primitives and protocols used in this work are secret sharing, multiparty computation and quantum key distribution. Additionally, one-time-pad and ITS message authentication codes are also used as sub-protocol. These are basically the main known ITS primitives and we used them to secure data at rest, in processing, and on transit. Additionally, we use symmetric cryptography to extend ITS methods where appropriate to achieve additional functionality or improved efficiency, but we are not directly using any asymmetric protocol. We are aware of recently standardized post-quantum cryptography (PQC), but did not rely on them in our work, i.e., we are following a rather conservative approach. However, we assume PQC is used to establish secure channels in case no alternative can be used, e.g., QKD or manual key management.

One-Time-Pad. The so-called Vernam cipher was the first primitive shown to be information-theoretically secure in the famous work of Turing [74]. By combining a binary message with a random key of the same length using the exclusive-or operation, a random ciphertext is generated which do not leak any information about the plaintext. Although perfectly secure, the concept of OTP is not practical because it requires a secure channel for key exchange in the first place, it is mainly used as sub-protocol.

Secret sharing. A secret sharing scheme allows a dealer to distribute a secret s between n parties P_i for $i = 1, \ldots, n$ such that the secret can only be reconstructed if a qualified set of these parties collaborate, while no other (non-qualified) set can not learn any information about the secret. The family of all qualified sets form the monotone access structure of the system. The most important access structure is the threshold access structure, which only requires a pre-defined threshold of parties to join their shares. Now, a secret sharing scheme for a monotone access structure is defined as a pair of algorithms, *share* and *reconstruct* where the first takes a s and generates n shares and the latter recovers the message from k shares of a qualified set. The main properties of a basic secret sharing algorithm are completeness and privacy. Completeness assures that all qualified set can reconstruct the secret, and privacy guarantees that no unqualified set of parties can reconstruct it. Many variants of secret sharing exist with different properties for different adversary models. Of importance for this work are standard schemes, robust secret sharing and verifiable secret sharing which are used in different scenarios.

Multiparty computation. In secure multiparty computation (MPC) a set of parties can jointly evaluate a function without leaking information to any of the participating parties, beyond what they can derive from their own inputs and the result of the computation.

Interestingly, information-theoretic multiparty computation (MPC) exists, and ITS variants are also based on secret sharing —as introduced by Ben-Or et al. [72] and Chaum et al. [73]. Thus, it is possible to obliviously compute arbitrary functions on secret shared data providing long-term security. The respective class of MPC protocols with information-theoretic security operate in the honest majority setting, i.e., under the assumption that an adversary corrupts less than half of the MPC computing nodes. Besides the non-collusion assumption, the protocols also rely on secure channels, which can be assured by different means. More concretely, MPC provides *input secrecy* (or *input privacy*), i.e., no party learns the input values of any other party, and *correctness*, i.e., the receiver of the result is ensured that the result is correct. Fortunately, information-theoretically secure MPC protocols are among the most performant approaches for computing on encrypted data and achieve practical performance in many application scenarios.

Quantum key distribution. Quantum key distribution (QKD) is the only known information-theoretically secure method to exchange keys among peers. As its name suggest, it leverages quantum mechanic properties to achieve unconditionally secure key exchange. In a QKD system, quantum bits are exchanged between two peers which cannot be intercepted or copied without being noticed, contrary to classical signals. Because QKD is universally composable it can be safely combined with secret sharing based systems to protect the communication channels between the parties, as proposed for secure storage and multiparty computation. However, because of many limitations of the technology in practice, we studied certain aspects in our work to make it more flexible and efficient.

Information-theoretic MAC. A message authentication code (MAC) appends a secret key-dependent short-length tag to a message which can be safely sent along an insecure channel. At the receiving side a verification algorithm is run with the same key as input to verify the message-tag pair received and therefore confirm its authenticity. Wegman-Carter type MAC (UMAC) follow a special construction template which is already four decades old. UMAC are known to be very fast and efficient when compared to other MAC algorithms that are based on block ciphers or hash functions but were not very popular on the Internet until recently. However, for our work they are relevant because they can be instantiated with information-theoretic security to be used for QKD classical channel authentication or to realize batching in our storage auditing protocol.

Computational symmetric primitives. Besides the ITS schemes and protocols presented we are also using most prominent symmetric primitives in our work. In particular, symmetric message encryption, message authentication codes and hash functions are used on various occasions. In general, mentioned symmetric primitives rely on basic operations and structures which are well analyzed and understood, and they are designed in a way to be resistant against all know attacks. It is currently also well established that this approach is not susceptible to quantum attacks, thus, modern symmetric ciphers are considered quantum-safe. This is why we are also relying on them in certain protocols.

Zero-knowledge proofs of knowledge. A zero-knowledge proof of knowledge is a two-party protocol between a prover and a verifier, which allows the prover to convince the verifier that it knows some secret piece of information without revealing it. Zero-knowledge proofs of knowledge must satisfy three main security properties: correctness, soundness and zero-knowledge. The latter two seem contradictory in the first place, however, they also make ZKP an invaluable tool in cryptography, especially for designing privacy enhancing technologies. Correctness means that honest verifiers can always be convinced by honest provers. Soundness, on the other hand, assures that the prover really knows the secret claimed, if the protocol succeeds and the verifier is convinced, with only negligibly small error probability. Finally, zero-knowledge guarantees that the verifier cannot infer any information from the proof, except from what is known from the claim anyway. In our work we leverage the concept to make our storage system auditable and we use them to make our multiparty computation system publicly verifiable, both being important functionalities for practical systems.

1.6.2 On the Security of Cryptographic Primitives and Protocols

In our work we heavily rely on the usage of cryptographic schemes and protocols. It is important to note, that cryptographic schemes and protocols are always analyzed and proven secure in a certain model. The cryptographic methods have to provide certain functionality and can typically be parametrized to a certain extent. However, the particular security properties have to be well understood to use the respective schemes accordingly. Especially the adversary model a protocol can resist is essential to characterize the security. This can be particularly challenging for distributed protocols, because many different flavors of adversaries exist.

To explain the concept, we give a quick example by using simplified multiparty computation, based on emerging ISO/IEC standard for MPC [75]. In a first step the generic model of MPC is defined as well as fundamental requirements. As a bare minimum to qualify for an MPC protocol, the solution must enable two or more parties to compute an intended function while keeping the input private. The computation can be parametrized by encodings, roles and deployment configurations, but as a minimum they have to provide correctness and input privacy. Optionally, additional properties can be given regarding the robustness of the protocol. However, from a security perspective the adversary model is of major interest. One of the properties is, if the respective MPC protocol can resist active or passive corruptions and how many nodes can be maximally corrupted. The computational limitations of the adversary are other important restrictions. In our design, we target information-theoretic security for our solutions wherever possible and reasonable, and do not impose any assumptions there. However, adversaries can be characterized by many more characteristics as explained below.

1.6.3 Adversary Models

An adversary's capabilities can be categorized in three mostly independent dimensions: regarding the *computational power*, the *network model*, and the adversary's power on a *protocol level*. In the following we will briefly describe the most important aspects in each of these dimensions.

Computational Power. Regarding the computational power, we categorize adversaries in two dimensions. On the one hand, we specify if we impose limitations on the computational power available to the adversary. Restricting the capabilities of an adversary is often essential to allow for efficient schemes and is the foundation for modern cryptography. However, as shown in this thesis, the few known ITS methods are also relevant and can be helpful in different applications. On the other hand, we additionally distinguish if the adversary has access to quantum resources or not. Although, this is also some restriction on the capabilities of the adversary, we want to highlight this difference to emphasize the current trend towards quantum-safe cryptography. In the following we quickly introduce the different types.

Bounded vs. unbounded. We distinguish computationally bounded and unbounded adversaries. In the former, we assume that the number of computations the adversary can perform is bounded above by some polynomial in the security parameter. Security is then proved under some computational assumption; that is, breaking the scheme would require the adversary to solve some computational problem which is believed to become super-polynomially more complex when increasing the security parameter. This approach is taken for most cryptographic primitives, such as encryption or signing. Protocols secure against bounded adversaries are also said to provide *conditional* security. For the latter, we do not pose any restrictions whatsoever on the computational power; schemes secure against unbounded adversaries are also called *information-theoretically secure* or *unconditionally secure* and are of importance when designing long-term secure cryptographic systems. This is because security is guaranteed independent of potential improvements on the computational capabilities of the adversary in the future.

Classical vs. quantum. Besides the standard definitions, for our work it is also important to distinguish between classical adversaries which have no access to quantum resources and quantum adversaries which can use quantum computers or quantum measurement equipment. In that sense, quantum adversaries with access to scalable quantum computers are assumed to break asymmetric cryptography based on number theoretical problems, which are underlying a broad spectrum of schemes used in practice today. If a protocol can resist quantum adversaries it is called quantum-safe. Currently, most symmetric schemes, including most hash functions, MACs, or ciphers, are considered to be quantum-safe, if appropriate parameters are used, together with information-theoretic cryptography which does not rely on any assumption at all. Additionally, a whole new branch in cryptography called post-quantum cryptography (PQC) emerged in the last decades, which aims at development of novel quantum resistant asymmetric cryptography. PQC relates its' hardness on problems also assumed to be intractable for quantum computers. However, post-quantum cryptography is not in the focus of this work and the protocols and system presented in this work mainly rely on ITS primitives and partially on symmetric cryptography to achieve quantum-safe properties.

Network Model. Regarding the network model, we mainly distinguish between *Synchronous vs. asynchronous.* In synchronous systems, it is assumed that protocols proceed in rounds: all messages sent in one round are available to their receivers at the beginning of the next round. This assumption often allows for elegant protocols and relatively simple proofs. However, such a network model is only realistic if all nodes run at roughly the same speed, the network is very stable, etc. In contrast, in an asynchronous model, also network latency or simply different computation speeds on different nodes where one does not want to always wait for the slowest node are modeled. Assuming an asynchronous network is often far closer to reality, in particular for protocols that are to be carried out over the Internet and not only, e.g., in a company-internal data center. However, allowing asynchronicity often causes a considerable overhead in the computational costs of a protocol, and also results in much more intricate security proofs.

Protocol Level. On a protocol level, there are various properties one has to distinguish when classifying adversaries. Here we present some relevant categorizations which were considered when designing our systems.

Passive vs active. This is the most basic categorization of adversaries on a protocol level. A passive adversary is only allowed to corrupt a party in the sense that he may see the party's internal state and all messages it receives or sends, but it is not allowed to change the behavior of the machine. Such adversaries are also called *honest-but-curious* and are, e.g., useful in a cloud-based setting where the provider is trusted to run the right payload but must be prevented from learning about user data. In contrast, an active or *Byzantine* adversary may in general force corrupted nodes to deviate arbitrarily from the protocol specification. This could be the right model if data is stored or processed on untrusted servers.

Adaptive vs non-adaptive. Generally speaking, a non-adaptive adversary has to decide which parties to corrupt right away when the protocol starts. On the contrary, an adaptive adversary is allowed to corrupt parties depending on what he has already seen in the protocol so far. That is, an adaptive adversary is allowed to corrupt the “most promising” parties at a given stage of the protocol. One can think of a non-adaptive adversary as a malicious party which already compromises a system at manufacturing time, but which cannot compromise additional devices once they have been shipped, which would still be able for an adaptive adversary. From a practical point of view, adaptive security clearly becomes more important with an increasing lifetime of the system.

Static vs mobile (distributed system). Assuming a static adversary in a distributed system means that once a party got corrupted, it will stay corrupted until the end of the system. This may be reasonable for short living systems. However, in the case of long-term storage scenarios, a single node might be reset to an honest state once the corruption is detected. We call such an adversary mobile, as he is able to corrupt different nodes at different points in time. Sometimes the respective corruptions are referred to as *transient*. They are relevant in our storage scenario, where we rely on the non-collusion assumption for long-term security.

Honest vs corrupted dealer (secret sharing). An honest dealer in secret sharing is always assumed to send consistent shares to all servers, whereas a corrupted dealer might send arbitrary and inconsistent shares to the servers. In particular, for a corrupted dealer it cannot be guaranteed that every qualified set of shares reconstructs to the same secret, or whether it reconstructs to a valid secret at all. While assuming a dishonest dealer might seem artificial at first glance, this might, e.g., happen due to corrupted devices sending altered messages on the network interface without the user noticing. Also, transmission errors on a network level could be detected if a scheme is resistant against dishonest dealers. Secret sharing protocols that can withstand malicious dealers are called *verifiable*, whereas all others are said to be *non-verifiable*. In our system for secure data storage and computation, e.g., the client delivering the input data is one example where the model has substantial impact. If we can assume that the client is under our full control it is safe to assume a honest dealer, however, if the client device can be easily compromised a corrupted dealer has to be assumed.

1.6.4 Security Assurance and Proofs

In addition to the different models and assumptions involved in secure protocol design, there are also different ways to prove the security of protocols. To understand the difference and implications we quickly present the main approaches.

Game-based security. In this approach the security of the algorithm is phrased as a game played between the adversary and a hypothetical challenger. The two entities are probabilistic processes which interact and the security is bound to an event which should occur with a certain target probability (0 or $1/2$), or in the case of sequences of games, close to the probability of a game with a different challenger. Security in this context means that every efficient adversary cannot do better in output guessing of the security event than with non-negligible probability, i.e., its distribution is close to the target distribution. Moreover, because in cryptography typically the security relies on assumptions which cannot be proven ultimately, the game-based technique is used to prove reductions to problems assumed to be hard. For example, an asymmetric encryption protocol is shown secure by proving that an adversary able to break the protocol can also break the underlying hard problem, e.g., the Decisional Diffie-Hellman problem.

Simulation-based security. The simulation paradigm follows a different approach and defines the security properties to achieve in an ideal model which also make use of ideal components such as a incorruptible trusted party. The security of a protocol is then shown by proving that what an adversary can do in a real protocol (real model) execution is almost the same as in an ideal scenario (simulation), which is secure by definition. Thus, this approach is also called the real/ideal paradigm, and a protocol is said to securely compute the ideal functionality, if for every adversary in the real model there exists a so called simulator in the ideal model, such that the view of the adversary in the real protocol execution is indistinguishable from the view of the adversary when executing the simulator in the ideal world.

Universal composability. Universal composability is a very rigorous approach used to overcome the problem of protocols being analyzed as standalone applications. It enables the construction of security models where security is retained under protocol composition which is the most generic result to achieve. Multiple instances of the protocols can run concurrently or even interact with each other without violating the security model. Various frameworks can be found in the literature and the methodology somehow extend the simulation paradigm. The UC framework introduces an adversarial entity called the environment, who generates the inputs to all parties (including inputs for honest parties), reads all outputs, and interacts with the adversary in an arbitrary way throughout the computation. A protocol is said to UC-securely compute an ideal functionality if for any adversary that interacts with the protocol there exists a simulator such that no environment can tell whether it is interacting with a run of the protocol with the adversary, or with a run of the ideal model and an according simulator. However, it comes with a number of drawbacks, e.g. sometimes impractically high computational overheads. Thus, designing UC-secure tools requires effort on the primitive level in order to get efficient building blocks that can be composed in a modular way.

Direct proof. In a direct proof we can argue about the security of an algorithm or protocol in a rather ad-hoc way. This approach is rarely used but is sometimes useful in the information-theoretic setting. In cryptography the more versatile techniques mentioned above have been developed. They are able to capture in detail the adversarial capabilities and define security properties in a fine-grained way.

How we used the different approaches. In our work we rely on the different approaches above as follows. When we discuss the problem of offloading QKD post-processing we directly show the properties based on information-theoretic arguments, i.e., we directly prove the desired property. However, this is a very particular problem with a very simple nature and therefore the only scheme in this category.

For the case of batch verifiable auditing and secret sharing in the storage section we leverage game-based proving, because it is well suited to capture the desired security properties. Simulation-based security proving is heavily used in multiparty computation. In the part on MPC we build on existing results and rely on secure protocols which have been proven in this framework.

Besides that, we extend MPC in a very natural way to achieve better security, e.g., when we combine MPC with zero-knowledge proof methods to achieve publicly verifiable MPC. Finally, for secure communication we leverage QKD, which has shown to be universally composable and can therefore be safely combined with other primitives and protocols. This is what we also exploit in our last part where we integrate it with secure storage and discuss possibilities for computation.

1.6.5 Performance of Protocols

A particular focus of our work is to research, design and evaluate the practical performance of system in advanced use cases. We therefore analyze and optimize protocols regarding performance which can be categorized in various dimensions.

Round and communication complexity. On the network layer round and communication complexity are essential factors for a protocol. Round complexity measures the number of sequential steps in the algorithm, i.e., the number of interactions required by parties in a distributed protocol. Communication complexity measures the accumulated number of data (bits) communicated between parties during protocol execution.

Computational complexity. Computational complexity, also known as time complexity, measures the number of operations required to run a particular algorithm or protocol, i.e., the time needed for execution. However, because it is hard to give absolute numbers, as the complexity generally increases with the size of the input and the runtime is also hardware (compute model) dependent, the complexity is typically expressed as a function of the size of the input in the generic form of asymptotic behavior (big O notation).

Empirical analysis. Deriving practical performance figures for different deployments and input is not straight forward in distributed systems. Especially, if the performance depends on a combination of computational and communication complexity as well as network layer behavior, e.g., as it is the case for multiparty computation. Additionally, as we have seen in the case of secure storage, the peculiarities of a concrete transport layer protocol leads to unpredictable behavior if connections are susceptible to channel imperfections, e.g., packet loss.

Thus, because concrete performance of a real instantiation depends on so many parameters, settings and side effects, we make heavy use of empirical testing and benchmarking in our work. Building proof-of-concept implementations and simulations were a valuable tool to measure or estimate achievable performance for the various protocols and systems developed.

In our work on distributed protocols the networking layer often turned out to be the limiting factor in performance, therefore, we were carefully researching this aspect for different scenarios. The systems for secure storage and processing of data mainly suffered from the interactivity in the underlying protocols which we tried to understand and optimize in our work. For example, during development of our batch auditing and batch verifiable secret sharing protocol the goal was to minimize the round and communication complexity for efficient operation on bulk data. However, for certain aspects where computation was the bottleneck, we researched efficient algorithms and implementations, e.g., efficient software and hardware encoding for secret sharing. Additionally, when we discuss offloading of QKD post-processing in secure communication, we propose a different approach to increase efficiency for a decoding problem which cannot be improved upon.

Chapter 2

Securing Data at Rest

2.1 Introduction

Cloud-based data storage and sharing is a very appealing technology and brings many advantages over conventional local storage systems. In this chapter we present a novel approach for long-term secure multi-cloud storage developed over recent years. It is based on the idea, that adopting suitable cryptographic mechanisms from the realm of information-theoretic cryptography—namely secret sharing—is a reasonable way to achieve adequate security for outsourced data storage and processing. For the design of our new type of secure and privacy preserving distributed storage we identified the following requirements as being important:

- Resist passive and active attacks for confidentiality, integrity, and consistency
- Provide long-term security and specifically resist attacks by quantum computers
- Enable concurrent access for multiple users
- Support increased resiliency and availability
- Support dynamic access rules, e.g., dynamic groups or write only
- Support efficient means for remote consistency checking, i.e., third party auditing

In this chapter we present the research results achieved in addressing these topics. Results in this chapter were also published in [1, 2, 3, 4, 5, 6, 7]. It is structured in the following way.

First, a new distributed multi-cloud data storage architecture called ARCHISTAR is outlined in Section 2.2. It secret shares data among several cloud providers and thus improves data security and availability compared to existing solutions using a single provider. The system developed comprises an architecture and protocols together with a proof-of-concept implementation of selected features. Parts of the implementation have also been released as open-source software on GitHub[1]. However, the implementation

[1]`https://github.com/Archistar`, accessed 2023-01-10.

only served as a tool to study the practicability of the proposed architecture and helped to identify shortcomings for subsequent research.

The main identified gaps of the proposed approach were performance issues on the network layer and data integrity for the secret shared data. To cope with the performance problems of the consensus mechanism and to optimize performance for imperfect networks, a model was developed and evaluated by simulations. The results are shown in Section 2.3. Furthermore, for the case of passive storage servers, a simplified proxy approach is introduced in Section 2.4. It has the potential for high-speed encoding via hardware acceleration, as shown in our performance study, and can therefore also be used within data centers. Additionally, to overcome the limitations on data integrity in ARCHISTAR, a very efficient batch auditing solution for secret shared data was developed in Section 2.5, which also builds the basis for our batch verifiable secret sharing introduced in Section 2.6.

2.2 New Architecture for Secure Distributed Storage

Based on the discussed requirements, we developed a new framework for secure and agile data storage and sharing in the cloud without any single point of trust or failure. It is based on a distributed architecture and provides full multi-user support without complex key management and revocation mechanisms and guarantees long-term security. It is intended to be used with the most appealing cloud delivery model, which is also the most vulnerable, i.e., public clouds. It addresses major user concerns in cloud usage as identified by Cloud Security Alliance [76] which are data breach and data loss.

Additionally, to the high-level security requirements mentioned in Section 2.1 we derived more concrete implementation features for ARCHISTAR, which are 1) no unauthorized user (including the cloud provider) must be able to access the data, 2) that a data owner must be able to dynamically authorize user access to their data with fine granularity, 3) authorized user access should be possible at any time without requiring the data owner's intervention, 4) data owners must be able to revoke access to data for any authorized user, and 5) no authorized users must be able to grant or revoke access rights (unless explicitly allowed to do so by the data owner).

Building such a cloud-based solution requires the combination of various techniques ranging from authentication mechanisms to access control systems up to strong isolation between different tenants. Strong privacy for user data can only be achieved through strong encryption techniques which ideally protect confidentiality from end-to-end. ARCHISTAR is based on secret sharing at the core, to achieve long-term security in the first place, and extended with additional consensus techniques to achieve integrity, consistency and the additional usage requirements.

2.2.1 Problems of Encrypted Cloud Storage

Secret sharing was not only chosen because of its long-term security, but also because standard cryptographic methods cannot be easily used for secure collaboration on encrypted data in dynamic groups. The usage of existing primitives from symmetric and public key cryptography would lead to very complex key management and thus scalability issues as well as serious problems when it comes to access right revocation. In particular, if an action is executed, e.g., granting or revoking access rights, the user performing this action must be online. In order to illustrate the problem with such a solution, we sketch the procedure of adding and revoking a member of a group:[2] to add a member, the user has to retrieve the random symmetric key and encrypt it again under the new user's public key. To revoke access for a particular user (or only his key in case of loss or exposure), the data owner must perform a re-encryption. Thus, she has to retrieve and remove all cloud-stored data, decrypt and subsequently encrypt it with a new random key before uploading the new data and keys to the cloud. The computation and communication overhead is significant as all computation must be performed client-side. This limits its usability for mobile or low-power clients. We note that although typically lazy revocation [77] is applied, i.e., re-encryption is delayed to the next write operation, the effort still can be far too high. Furthermore, this approach requires a public key infrastructure (PKI) to be available to all users.

Related approaches, e.g,. [78], solve some of the problems associated with this use case, but introduce other difficulties. Most prominently discussed are currently *attribute-based encryption* [79, 80] and *proxy re-cryptography*[81, 82], which we briefly discuss below.

Attribute-based encryption. Attribute-based encryption (ABE) [79, 80] is a recent public key encryption paradigm that allows for end-to-end secure data storage and sharing systems [83, 84]. In ciphertext policy ABE (CP-ABE) approaches [85], a user's private key is associated with a set of attributes and a ciphertext specifies an access policy over a universe of attributes. A private key holder is able to decrypt a ciphertext if and only if his attributes satisfy the policy of a ciphertext. CP-ABE thus allows to realize implicit authorization, i.e., authorization is included into the encrypted data and only people who satisfy the associated policy can decrypt data. People who may decrypt may not be known when producing ciphertexts. In a data sharing scenario, keys can be issued by the data owner or some other authority. Recent works, e.g., [86], consider ABE in data-sharing scenarios and investigate schemes where ciphertexts can be re-encrypted to stricter policies and also efficient constructions for revocable ABE.

Proxy re-cryptography. The use of proxy re-cryptography [81, 82] can also overcome the limitations of content encryption in data-sharing applications. The basic idea

[2]We assume the use hybrid encryption, i.e., to use an efficient symmetric encryption scheme to encrypt the content data with a key chosen uniformly at random and then store the encrypted content data along with the random key encrypted under the public keys of the members of the group the data is shared with.

behind proxy re-encryption is that one can give a re-encryption key to a semi-trusted proxy that allows the proxy to re-encrypt data encrypted under one key into data encrypted under another key without seeing the plaintext. Consequently, re-encryption can be outsourced, although meaningful (and efficient) proxy re-encryption schemes only exist for the public key setting. Thus, the application of such schemes does not eliminate all problems within revocation as discussed above.

However, ARCHISTAR is a new approach based on information-theoretic secret sharing extended to a *multi-user distributed storage system*, hence, combining flexibility with long-term security.

2.2.2 The Cloud-of-Clouds Approach

The use of secret sharing to protect data privacy can also provide increased integrity and availability. In a distributed storage system, secret sharing can be used to securely encode data into n fragments (secret shares) for subsequent distribution on multiple storage providers or servers (parties). Furthermore, the share combinations allowed to reconstruct the original message can be defined at encoding time as monotone subsets of the distributed original shares, e.g., a configurable threshold $k \leq n$ in the simplest case. Any adversary, i.e., an external attacker or a storage provider, holding less shares than required for reconstruction, cannot learn the original data. However, plain secret sharing is just a set of two algorithms (share and reconstruct) and a lot more it needed to build a secure and robust storage system out of it. ARCHISTAR integrates secret sharing with protocols known from fault-tolerant computing to resist fail-stop events or even active attacks without service interruption or data loss and to provide strong consistency guarantees for multiple transactions processed in parallel.

Advantages. Through their inherently distributed nature, secret sharing based storage schemes provide means for increased data confidentiality, integrity and availability. This so-called cloud-of-clouds approach is very appealing, as it prevents data breaches and data loss at the same time. If built without any single point of failure or trust it additionally mitigates vendor lock-in.

Today's service-level agreements (SLAs) for cloud storage capacity typically provide guarantees in terms of uptime but specify little in terms of data availability or data protection. A typical value for service availability is 99.9%, e.g., as provided by Amazon web services (AWS), one of the leading public cloud platforms. It defines this under the term “Service commitment" and makes Amazon S3 available with a “Monthly Uptime Percentage" of at least 99.9% during any monthly billing cycle [3]. However, not much is said about guarantees for data loss or confidentiality and only very limited compensation is given for service commitment violations. Public cloud offerings still delegate the ultimate responsibility of protecting against data loss or data breaches to

[3]`https://aws.amazon.com/s3/sla/`, accessed 2022-11-17.

users. To be on the safe side, users should encrypt data and replicate it to different locations, which means that dutiful users need to use different providers anyway. A secret sharing based architecture has this property by design and due to the flexibility of the reconstruction threshold it can be more efficient than simple replication as also shown in the availability analysis in Section 2.4.

Another advantage of secret sharing based storage systems is that there is no need for explicit encryption keys and therefore key management issues do not emerge. The keyless property makes them a very appealing candidate to build cloud-based information sharing systems with increased dynamicity and capabilities for advanced sharing scenarios. Because secret sharing is information-theoretically secure, the strong confidentiality properties can be used to build storage systems which are long-term secure [87]. Its security is provable against any type of attacker as long as he is prevented from accessing enough shares to reconstruct the message.

Disadvantages. There are two major disadvantages of secure distributed cloud systems: while well understood and provably secure, the use of ITS secret sharing introduces a different notion of security than known from encryption schemes. The security model requires an explicit non-collusion assumption, and the guarantees are only true for a limited number of colluding nodes (providers). Without additional methods, it is not possible to protect or detect breaches of any kind if enough parties collude. Thus, additional mechanisms must be integrated to relax this assumption and add detection mechanisms for confidentiality or integrity breaches.

In addition, distributed systems are more complex to implement and add additional administrative overhead. In the case of cloud storage, this disadvantage is less dramatic as it can relieve the user from the complexity by offering the solution as a service. Even conventional systems must include backup storage locations to protect against data loss, if availability is a concern, and single points of failure must be prevented. Hot-backup sites with full state-replication for immediate fault-recovery lead to systems that are comparable in complexity and costs to secret sharing systems. Nevertheless, our research in [22] showed that the usability issues could be a substantial barrier, which is why we designed the system to be simple in deployment, operation and use.

Related work. Various secret sharing based storage systems have been proposed and implemented in the last years (cf. [88]). We will briefly summarize the most relevant ones and discuss their shortcomings when it comes to their applicability as cloud-based information sharing systems.

TahoeLAFS [89] encrypts data, disperses the encrypted data through erasure-coding and stores the fragments on multiple active backend storage servers. The storage servers themselves can utilize locally attached cloud storage for their storage backend. It employs a proxy-pattern where only a single proxy can access the stored data at a time.

Nubisave [90] allows for flexible combination of operations upon user data. Incoming user data can be deduplicated, erasure-coded and then distributed upon multiple cloud storage providers. Cloud providers can be dynamically selected due to run-time criteria. Operations are mostly implemented though external libraries, the `splitter-ng` component can perform limited secret sharing (Shamir) through the `JSharing` libary. Parallel access to backend data by multiple Nubisave instances is not supported.

RACS [91] utilizes erasure-coding to distribute data upon multiple cloud storage providers. Its use-case is prevention of vendor-lock-in, security and privacy is of no concern, i.e., data is not encrypted. Access to clients as well as to backend storage is provided through an Amazon S3-compatible protocol. Parallel access to stored data through multiple RACS instances is achieved through usage of Apache Zookeeper, which implements the Zab primary-backup protocol for synchronization.

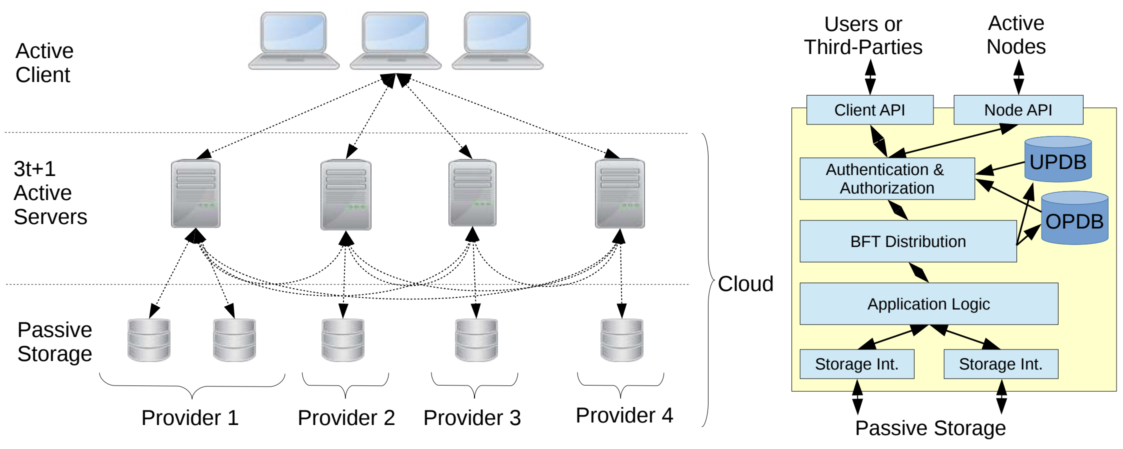
DEPSKY-CA [92] is a Byzantine fault-tolerant storage system. It provides availability and confidentiality through computational secret sharing [93]. In DEPSKY's system model, servers cannot communicate with each other, clients must synchronize access to files through a low-contention lock mechanism that uses cloud-backed lock files for synchronization. This mechanism is only obstruction-free and depends upon synchronized clocks. We would prefer a synchronization directive that is suitable for high-contention situations (multiple parallel writers) and is robust in face of live-locks and malicious clients.

In summary, many proposals for storage solutions with information dispersal mechanisms have been proposed. They are using secret sharing or similar techniques like all-or-nothing transforms to protect confidentiality. Some of them even deal with Byzantine robustness. However, to the best of our knowledge, at the time of writing none of them supports full concurrency for a multi-user environment, nor does any of them provide a solution to implement information sharing[4]. Additionally, none of the proposals discuss the interplay with access control mechanisms.

2.2.3 The ARCHISTAR Architecture

The ARCHISTAR architecture is intended as distributed storage systems for secure and resilient cloud based-data sharing. It focuses on data availability and integrity as well as confidentiality guarantees, even when considering the storage providers as adversaries. At the core it integrates secret sharing with protocols for Byzantine robustness in a multi-cloud setup. To enable secure agile collaboration between stakeholders, it integrates an authentication and authorization layer. The combination of these three components leads to a new type of system not yet considered in this context. It provides a fully decentralized multi-user system without any single point of failure or trust and increases attacker's effort significantly compared to other systems.

[4]We note, however, that there are solutions for the non-distributed case, e.g., [94].



(a) System (b) Node

Figure 2.1: Architectural Overview. On the left (a) the layered architecture is shown which consists of passive storage servers holding the data, active server components accepting client request and running the consensus protocol, and clients secret sharing the data and generating requests. On the right (b) the software components of the active server nodes are depicted.

ARCHISTAR's high-level physical structure is shown in Figure 2.1a. Active clients represent user devices which encode/decode user-data with a configurable secret sharing scheme. This prevents servers from ever touching plain-text data and therefore increases the level of security. The use of multiple sharing schemes allows for a configurable trade-off between security and required communication bandwidth for different client scenarios ranging from enterprise network access to mobile clients.

On the cloud provider side, the architecture foresees the use of two elements—*passive* storage and *active* server components with the latter running in virtual machines. Active servers are mandatory for the execution of the Byzantine fault-tolerant protocols and are necessary to validate user credentials and enforce access control to stored data fragments. They do not rely on persistent data and should be able to execute from a read-only certified program image while volatile data is only persisted to the passive storage layer. Through this separation of active and passive components we achieve flexibility with regard to storage-provider requirements.

The node architecture is shown in Figure 2.1b. Users access the node through the Client Application Programming Interface (Client API) which is highly influenced by the Amazon S3 access protocol. Node-to-node communication is performed through a Node API but both node and client control flow is unified within the node. All submitted commands are subject to the Authentication and Authorization component that verifies user's identity and their corresponding access rights. Required user and object information can be gathered from the User Policy Database (UPDB), object access information is stored within the Object Policy Database (OPDB). All operations, including accessing UPDB and OPDB, are synchronized through the BFT consensus component before they are finally executed within the Application Logic Component. To allow for modularity all data storage access is managed by minimal storage system interfaces ("Storage Int.").

In general, the ARCHISTAR architecture does not depend on special modifications of cloud offerings but is rather intended as an overlay or virtual cloud service on top of them. This approach is preferable to approaches which require modifications in the used cloud stack.

2.2.4 Byzantine Robustness and Concurrent Access

Within Byzantine fault-tolerant (BFT) systems a defined number of components may fail arbitrarily without impacting overall system stability. Initially thought to be too inefficient for general use, *practical fault tolerance* [95] (PBFT) sparked a renaissance of BFT algorithms in 2001 which further culminated in a tremendous research push by blockchain. PBFT turned out to be well suited for our use case, because it is based on a practical network model (weak-synchrony), provides optimal robustness (lower than a third malicious servers), and only relies on symmetric cryptography; theoretically it can even be ITS if perfectly authentic channels are assumed. Until today, it is still the most efficient protocol not relying on any asymmetric cryptography. However, subsequent PBFT protocol improvements reduced the minimal replica count and thus improve performance by either limiting the attacker's possibilities [96, 97] or introducing the concept of speculative BFT [98, 99]. Nevertheless, during faults those protocols commonly fall back to PBFT, therefore, we rely on pure PBFT in our first design.

An overview of PBFT is given in Section 2.3.1 and an overview of the core 3-phase protocol used in shown in Figure 2.2. ARCHISTAR employs a Byzantine fault-tolerant distribution algorithm akin to PBFT. PBFT assumes that each server has full access to the stored plain-text data. As this assumption does not hold for a secret shared storage system, the protocol had to be adopted. Clients perform secret sharing locally and pass fragments to each node along with metadata. This can be considered a relatively simple extension to standard PBFT, where all nodes receive identical data. In that sense, the received metadata from the client is treated as the original PBFT payload and the fragments are handled as associated data. Because the associated data is not identical it cannot be used for transaction matching, but if integrated in this way it does not need any change to the basic PBFT message flow and does therefore not produce overhead compared to plaintext transaction.

The use of fragments as associated data in PBFT also influences the result phase where the client performs a majority voting upon the operation's result. Also, here only the metadata can be treated according the original protocol and secret shared data has to be treated differently. Looping through secret shared fragments is very beneficial for the system performance, which is not affected, but lead to significant security implications. However, the priority was on system performance and the introduction of additional protocol phases or computational effort had to be avoided.

Concrete impact on the security and system configuration are discussed in the next section. In a nutshell, because all access to the backend storage is synchronized through the BFT layer, ARCHISTAR can inherently deal with parallel concurrent write requests, even from multiple users. However, only the consistency of the metadata can be guaranteed in case of a corrupt client. For secret shared data a malicious client could send an inconsistent sharing in the beginning of the request, which cannot be efficiently detected by the replicas without adding additional interaction and substantial computational load. To cope with inconsistent sharings and malicious clients, we subsequently developed dedicated solutions in Section 2.5 and Section 2.6 which are particularly efficient for storage applications. However, the first ARCHISTAR proof-of-concept was based on basic threshold secret sharing techniques which could not prevent from a cheating dealer (client).

2.2.5 Integration of Secret Sharing

For integration of secret sharing the following relevant protocols have been identified to support a wide range of use cases. In general, only threshold sharing schemes were selected, because they are the most efficient ones in terms of storage overhead (share size) and can also be efficiently implemented in software. The four modes are summarized in Table 2.1 and explained in the following.

Selected schemes. The main ITS scheme used in ARCHISTAR is Shamir secret sharing [100]. It is a perfectly secure secret sharing (PSS), provides a n-out-of-k threshold access structure and is optimal in terms of share size, i.e., each share is as long as the message. The scheme is based on the observation that in a field a polynomial of degree $k-1$ is uniquely determined by k values, while knowing the function values on at most $k-1$ positions does not reveal any information about the slope on any position different from the known ones. Thus, to share a secret s, the dealer now chooses a random polynomial $f(x)$ of degree $k-1$ such that $f(0) = s$, and gives $f(i)$ to P_i for $i = 1,\dots,n$. To reconstruct the secret from sufficiently many shares, the polynomial can be reconstructed, and evaluated at 0.

For improved storage efficiency and to support storage of less sensitive data along with ITS protected data, computational secret sharing (CSS) as introduced by [93] was selected, because it has a rather simple structure and can be implemented as extension to PSS. For ideal schemes the storage overhead is the same as when storing n replicas of the original data. This is significantly worse than in non-private information dispersal systems [101] (IDS) or in common RAID systems, where the overhead is only about n/k if the system should be able to compensate for up to $n-k$ losses. Krawczyk [93] proposed a system to combine the space efficiency of information dispersal with the privacy of secret sharing schemes, if one is willing to accept a computationally bounded adversary. On a high level, the idea is to first encrypt the data m using a symmetric encryption scheme, and then apply the PSS only to the used key and an IDS to the resulting ciphertext.

Interestingly, the IDS scheme is similar to Shamir's scheme. In fact, a simple and efficient scheme to disperse k elements $a_0, \ldots, a_{k-1} \in \mathbb{Z}_q$ is to define the polynomial $f(x) = a_{k-1}x^{k-1} + \cdots + a_1 x + a_0$, and to output shares $\sigma_i = f(i)$ for $i = 1, \ldots, n$. Now, if the encryption scheme maps into $\mathbb{Z}_q$, this information dispersal scheme—and as a result also Krawzcyk's computationally secure secret sharing scheme—asymptotically achieves the optimal storage overhead of n/k. ChaCha20 [102] is used as instantiation for the symmetric cipher in CSS.

To cope with active corruptions more efficiently we also implemented robust versions of the respective schemes. A robust secret sharing (RSS) also comprises two algorithms (share and reconstruct) but reconstruction can deal with active corruption. A scheme is called t robust if it can reconstruct the original message even if up to t shares were maliciously changed by an adversary. It can be shown that Shamir's secret sharing scheme is robust for every $t < n/3$, if $k = t + 1$ is assumed. In general, for a meaningful definition in a threshold setting, only $t < k$ needs to be considered for confidentiality reasons. It is thus easy to see that robust secret sharing for $t \geq n/2$ cannot be achieved and maximally robust schemes are given by $n = 2t + 1$ for $k = t + 1$.

To support robustness in the PSS case we rely on the basic scheme from [103] (R-PSS) often referenced as information checking (IC). In the scheme the dealer extends the share sent to a party by message authentication codes. That is, for every party P_j, the share of P_i is authenticated with a tag $\tau_{i,j}$, and the corresponding key is given to P_j. At reconstruction phase, the shares can now be checked for correctness by checking whether they are accepted by sufficiently many tags. To have a failure probability negligible in λ, MACs with length at least λ bits need to be used, and thus each player needs to additionally store $\Omega(\lambda n)$ bits. If an ITS secure MAC is used for IC in combination with PSS, it also becomes an ITS secure robust secret sharing (R-PSS). The ITS-MAC is instantiated with Poly1305 [104], an efficiently computable universal hash family for message authentication. It is also important to note, that IC supports immediate reconstruction and can recover the secret as soon as k correct shares are available, whereby PSS can only reconstruct if $\lceil \frac{k+n}{2} \rceil$ correct shares are available.

Finally, for the CSS case we leverage the fingerprinting technique (FP) to make the CSS scheme robust (R-CSS), which was already proposed in [93]. FP significantly reduces the overhead compared to R-PSS. Namely, instead of using a keyed MAC, one can simply compute a hash of each share, which is then distributed among the parties P_i. This distribution is done such that only $t + 1$ honestly returned parts are necessary to reconstruct the hash value, i.e., it also supports immediate decoding. Doing so, the reconstruction algorithm can then check which shares of the secret were returned honestly, and the secret can be recomputed. Reconstructing the hash value from partially altered parts can be achieved by using an error correcting code, where we denote the coding function by C and the decoding function by D. On a high level, such a code maps strings of length j into a sequence of n strings in a way that allows to reconstruct a message if at least $\frac{d-1}{2}$ indices have not been modified. Here, d is the distance of the code. As we assume up to d malicious shares, we require the code to have $d \geq 2t+1$. Secure hash SHA256 is used as collision resistant hash in R-CSS together with a simple replication code with allows for majority voting in D.

Basic Modes	Description	Example Suite
M1	Null confidentiality (BFT)	None
M2	Perfect security (PSS)	ShamirPSS
M3	Robust PSS (R-PSS)	ShamirPSS-Poly1305
M4	Computational security (CSS)	ShamirPSS-ChaCha20
M5	Robust CSS (R-CSS)	ShamirPSS-ChaCha20-SHA256

Table 2.1: Supported modes of encryption also mentioning concrete instantiations used. Shamir secret sharing is used as PSS. Krawczyk's scheme as CSS with PSS for key sharing, ChaCha20 as cipher, and IDS for erasure coding. For R-PSS Rabins IC is added to PSS with Poly1305 as ITS-MAC. For R-CSS Krawczyk's FP method is used with SHA256 for hashing and a repetition code (replication) for encoding.

Impact on PBFT configuration. Now, it is interesting to investigate the implications on the PBFT configuration for the various types of secret sharing. Given a PBFT system which is capable of coping with f malicious nodes[5], one requires at least $3f+1$ nodes in the system. A successful transaction is characterized by $2f+1$ nodes being in the COMMIT state and $f+1$ identical REPLY messages received by the client, with additional f malicious REPLY messages in the worst case of f malicious nodes present in the final quorum.

When secret shared data is added as associated data various problems arise, because we basically separated transaction consistency from data consistency. Data consistency cannot be checked within the system easily and majority voting cannot be applied to recover secret shared data at the client. In fact, a malicious client would be able to send corrupt data shares unnoticed by the nodes. However, he would not be able to influence the overall system state based on plaintext metadata managed in the PBFT system. This approach was taken to prevent from slowdown through cryptographic protocols required to check consistency of payload. Thus, for the time being we assume a client which

[5]We assume each replica is holding a fragment of the sharing and use the common terminology of f for corrupt nodes from now on to also refer to the number of corrupt shares for RSS reconstruction.

always generates a honest sharing of the message and discuss remediation solutions later. Now, to protect from malicious replicas adequate reconstruction procedures have to be used and thresholds have to be respected. Furthermore, the threshold for the sharing cannot be selected freely and has to satisfy certain conditions, i.e., $k > f$ for confidentiality.

Because PSS can correct up to $f = \lfloor \frac{n-1}{3} \rfloor$ errors, for an optimal value of $k = f + 1$ the reconstruction procedure requires at least $2f + 1$ correct shares to successfully complete. Thus, f additional answers are required in the last phase of the BFT protocol to reliably reconstruct in the presence of f malicious replies. Given, that $3f + 1$ has to be a valid quorum in PBFT we end up with an overall system configuration of $n = 4f + 1$ which is not optimal from a PBFT perspective. However, the amount of system nodes can be reduced back to optimal by application of robust secret sharing (R-PSS), which can reconstruct from $2t + 1$ shares correctly by adding limited overhead in storage and reconstruction. Therefore, the use of R-PSS is important if most efficient PBFT operation at $n = 3f + 1$ is needed or the number of nodes should be minimal. Additionally, if the RSS schemes support immediate decoding, i.e., reconstruction terminates as soon as $t + 1$ consistent shares are available, the system performance is not affected by adding the secret sharing layer to PBFT. However, linear properties of PSS are lost with R-PSS which could prevent from advanced operations on the data, e.g., computation on fragmented data as analyzed in Chapter 3. Finally, the very same arguments hold for the CCS and R-CSS variant. Technically, the handling of the R-CSS variant gets even simpler if a repetition code (aka replication) is used for encoding, because the hash values of all shares can be maintained within the plaintext metadata and produce only linear overhead in the number of nodes n.

In summary, ARCHISTAR provides a carefully selected set of secret sharing schemes as presented to minimally impact PBFT performance and give modest storage overhead. However, it requires stronger assumptions on the client, which has to provide consistent sharings for the system to work. If this assumption is not valid a dedicated protocol for efficient offline consistency checking is presented in Section 2.5, which could be runs regularly on the stored data. Moreover, if immediate verification is required, the batch verifiable secret sharing (BVSS) from Section 2.6 and be used, which requires one additional interaction with the client.

2.2.6 Authentication and Authorization

To protect the information from unauthorized external access, strong authentication methods must be put in place at each node which is not different to conventional systems. However, to protect data from unauthorized access, we propose to maintain a user and policy database via the already available BFT functionality. If all changes to user and policy databases are only made via BFT transactions, globally consistent user and access control information is guaranteed. and because the access policies are correctly enforced by all honest nodes, malicious nodes or clients will never get enough shares to compromise the system, even if the adversary controls all malicious nodes and clients in a coordinated way.

Thus, we achieve a fully decentralized data collaboration system without single point of trust or failure. The components of the node are illustrated in Figure 2.1b, every node integrates a dedicated stand-alone authentication and authorization module on top of the BFT layer. Conceptually, it has only read access to the globally maintained user policy database (UPDB) containing the basic user credentials and the object policy database (OPDB) containing the access control information for data objects.

In essence, the keyless nature of secret sharing enables enormous flexibility and agility in data sharing, but it must be combined with dedicated AA system to prevent from unauthorized access to fragments. By maintaining access right in a consistent way we achieve the required guarantees and the users can be assured that their data is secured.

2.2.7 Proof-of-Concept Evaluation

In the following we briefly overview the proof-of-concept implementation, especially the performance achieved for encoding and decoding of secret sharing and BFT throughput. A first version of the proof-of-concept (PoC) was developed early during the thesis to evaluate the basic concept and identify gaps which could be closed by additional research, e.g., performance optimization for PBFT or efficient consistency checking for secret shared data.

Some components of the ARCHISTAR framework are available on GitHub[6] as open-source software. Effort was spent to create generic components, which can be used in different use cases and future projects. In general, the major security properties could be achieved and verified, which shows that that the approach is valid and secure by design. However, the achievable performance was of major interest for the practicability of the system which could only be evaluated with a real implementation. We briefly summarize the measured performance of two major system components, i.e., a secret sharing library and a PBFT implementation.

Secret-sharing engine performance. All mentioned ARCHISTAR secret sharing modes have been implemented in a Java secret sharing library called `archistar-smc`. The software abstracts implementation details behind two important interfaces (*Secret-Sharing* and *InformationChecking*) and thus allows for easy reconfiguration of implemented algorithms. The BouncyCastle library[7] is used as algorithm-provider because the Standard Java CryptoAPI does not provide sufficient error information during de-

[6] `http://github.com/archistar`, accessed 2023-01-10.
[7] `https://www.bouncycastle.org`, accessed 2023-01-10.

cryption. The initial version was rather slow but continuous optimization led to a well performing widely used library. The implementation works on byte level, i.e., in a finite field of GF(2^8), which enables fast finite field operations via lookup tables. Larger word sizes would cause a significant slowdown, which is not desirable for storage applications, except there are particular reasons, e.g., for auditing or multiparty computation support. Thus, the current library cannot be used for secure computation purposes, because this would require to use larger prime fields. Nevertheless, the software can be easily extended if structured data must be secret shared for use in multiparty computation.

Achieved performance figures with some level of parallelization are shown in Table 2.2. Please note, currently parallelization only up to n (maximum number of shares) threads is used—in our test-case, the embedded Intel Atom processor would offer more cores than our maximum number of shares, thus no full utilization of the processor cores is achieved. Sharmir refers to the ITS secure scheme (PSS), and Rabin refers to information dispersal (IDS) [101] which is used as sub-protocol for erasure coding in Krawczyk CSS, i.e. to fragment the encrypted data. In that sense, IDS is a non-systematic erasure code and works similar to PSS, but instead of evaluating an almost random polynomial, all coefficients are data words.

In summary, with optimizations we achieved good software performance and were able to use all available cores. We reached over 400 Mbit/s for PSS and are even able to saturate a Gbit/s network connection for CSS with a midrange desktop processor. Even on small embedded processors, we reached more than 16 Mbit/s for PSS and 30 Mbit/s for CSS in a $4/3$ sharing configuration. We also found sharing and reconstruction providing the same speed for PSS and reconstruction being roughly 2 times slower than sharing in CSS. Although some improvements would be possible by using other programming languages, no systematic improvements are feasible, especially for use of IDS which resembles a non-systematic error correcting code where vectorized instructions cannot help. The current implementation is reasonably fast and were not a performance bottleneck in our evaluations. However, it may be too slow for use within data center. If substantial speed-ups are needed, hardware based solutions should be studied (c.f. Section 2.4).

Finally, we also researched ways to get reasonable performance for JavaScript, which enables web based clients. Surprisingly, after a lot of manual optimization we were able to reach comparable performance to the Java implementation by making use of *asm.js*[8] The resulting software module `archistar-smc-js` is also open source and the detailed results are shown in Table 2.3.

PBFT protocol implementation. For PBFT performance measurements a Java implementation with support for secret shared data has been developed in `archistar-bft`. It provides a BFT state machine which is used at each active server/replica and inter-

[8]`http://asmjs.org`, accessed 2022-12-12.

CPU	Speed	Algorithm	Split-Up	Combine
i5-4690K	3.5GHz	Shamir (PSS)	53753.3	52378.0
C2758	2.4Ghz	Shamir (PSS)	8192.0	7881.5
Exynos A9	1.7GHz	Shamir (PSS)	6917.8	3932.0
ARM A53	1.2Ghz	Shamir (PSS)	3502.4	2056.9
i5-4690K	3.5GHz	Rabin (IDS)	694237.3	124498.5
C2758	2.4Ghz	Rabin (IDS)	99417.5	26072.6
Exynos A9	1.7GHz	Rabin (IDS)	67590.8	10529.6
ARM A53	1.2Ghz	Rabin (IDS)	40634.9	4395.2
i5-4690K	3.5GHz	Krawczyk (CSS)	167868.9	79844.1
C2758	2.4GHz	Krawczyk (CSS)	32507.9	16862.9
Exynos A9	1.7GHz	Krawczyk (CSS)	16430.0	6949.4
ARM A53	1.2Ghz	Krawczyk (CSS)	9669.5	3765.1

Table 2.2: Performance (Version *a*8*a*344*b*) with multi-core enabled Rabin IDS. Currently a maximum number of *n* threads is used. The tested Atom C2758 board offers 8 cores, so it is not utilized to the maximum of its abilities. All values are in kByte/s.

	PSS		IDS		CSS	
	Enc	Dec	Enc	Dec	Enc	Dec
Java	22.2	21.9	169.2	58.2	67.1	38.9
JS naive	0.1	2.3	5.5	5.3	2.8	2.6
JS optimized	6.5	28.1	67.1	61.0	38.9	36.9

Table 2.3: Performance comparison of `archistar-smc-js` JavaScript (JS) with *archistar-smc* Java implementation taken on a Core2 Duo E8400 @ 3.00GHz running Ubuntu16.4, Java8 and Node.js 4.2.6. All values in MB/s.

actions with other servers are implemented through callbacks (inversion-of-control pattern). TCP based communication is used between nodes with optional support for TLS. Although, the performance seemed reasonable during local testing on a single computer, after cloud deployment the performance degraded to unacceptable levels.

The same was true for `archistar-bft-js`, an alternative implementation based on JavaScript. We investigated a scenario that deals with networks that are less reliable than pure LAN implementations, but still have reasonable connectivity, especially in the optimistic case without node failures. Ideally, we would like a protocol whose throughput closely tracks the network's performance especially for the optimal case of no faults, but under the assumption of unreliable transport. However, in the concrete practical multi-cloud storage deployment where we compared the performance of the same implementation, once in a LAN-based setup and once in a cloud-based setup with Amazon Web Services (AWS) we found the same unexpected behavior as with our first implementation.

For reproducibility we configured a Docker container setup which consists of four nodes, where the RTT between each node is distributed following a truncated normal distribution with mean 20ms and a variance of 5ms and with a link bandwidth of 1 Gbit/s (which depends on the host machine, actually it is a net bandwidth of 500 Mbit/s). The host machine for the Docker based experiments is a Dell Latitude featuring an Intel Core i7-2640M CPU accompanied by 8 GB of RAM. We use different file sizes for testing our deployed system. In order to test the upload and download performance (time until the request/transaction is finished) we used different file sizes: 1 kiB, 500 kiB, 1 MiB. Files with the aforementioned sizes are uploaded or downloaded (obviously bigger files are fragmented the latest by IP, due to MTU). We further deployed the system on Amazon Web Services and measured its performance. The upload performance for both setups are depicted by Table 2.4. In the table, "Total" accounts for the overall processing time of a request (upload/write), which includes encoding the data (using secret sharing) and uploading them to the replicas (triggering a transaction for the upload). "Transaction" denotes the time between sending the request to the underlying BFT protocol and receiving all responses (we waited for all responses in our test to measure overall worst case timing, normally only a majority is used). The ex-

		AWS (ms)		Docker (ms)	
		Total	Transaction	Total	Transaction
1 kiB	Upper CI	466	457	322	314
	Average	435	426	289	281
	Lower CI	404	396	249	262
500 kiB	Upper CI	3338	3078	576	300
	Average	2749	2488	553	281
	Lower CI	2159	1899	531	262
1 MiB	Upper CI	3612	3043	922	369
	Average	3390	2821	905	359
	Lower CI	3167	2598	889	348

Table 2.4: Upload performance results (95% confidence intervals) for different file sizes using `archistar-smc-js` for our Docker setup and Amazon Web Service (AWS).

periments clearly showed upload and download latencies for the AWS case to be worse than could be expected. We suspected the reason of this discrepancy to be somewhere in the interplay of TCP, TLS, congestions, and higher latency. Although this was only within a single cloud, the transaction times we experienced in our system were already unacceptable, and we decided to have a closer look at the dependencies of transaction times on network parameters like packet loss and latency, as multi-cloud settings will experience even rougher networking conditions (c.f. Section 2.3).

2.3 BFT Performance Modelling and Optimization

In this section we take a deep dive into the network layer and protocols for PBFT implementations for lossy and medium to high latency channels. To the best of our knowledge, we present the first approach for a performance model of PBFT. We analyze the core 3-phase view-consensus protocol in PBFT without additional features like leader change and checkpointing and develop an analytical performance model for success probability and latency of transactions. Then we present simulation results and analyze system performance using TCP and UDP as transport protocols. We further explore the parameters available for tuning such systems and evaluate the model by benchmarking different configurations. We also propose a hybrid transport mode that is able to increase performance by making use of both TCP and UDP. The results are then compared to a real implementation in a comparable environment.

2.3.1 PBFT Protocol

We briefly review the PBFT protocol which is used to develop a performance model that specifically considers unreliable communication channels, which is always the case in real systems. This analysis was motivated by the poor performance measured when deploying ARCHISTAR which used TCP for communication in large scale multi-cloud environments. Although PBFT is known to perform well in local LAN settings with high-bandwidth connectivity, low latency, and low loss, we found the performance achieved in typical multi-cloud settings disappointing.

PBFT is a leader-based state replication protocol with three phases and able to cope with arbitrary failures. It utilizes a quorum based voting mechanism between all involved servers to provide strong consistency and message ordering in the face of Byzantine faults. Despite the tremendous progress in the field, it is still the most efficient protocol which does not use any asymmetric cryptography and would be even ITS if the servers would be connected over perfectly authentic channels. Still, over inauthentic and unreliable channels as found in communication networks, it only requires methods from symmetric cryptography (e.g. MAC) to compensate for the channel insecurity. In contrast, newer consensus protocols make heavy use of asymmetric cryptography to achieve better efficiency.

PBFT is a state replication mechanism that can work over unreliable channels and guarantee safety and liveness even in asynchronous environments such as the Internet. For this, it needs at minimum $3f+1$ nodes, tolerating up to f of them being arbitrarily faulty in the Byzantine model. The core view-consensus protocol is depicted in Figure 2.2. It comprises three phases which on a high level work as follows.

Being leader-based, one node takes over leadership in linearizing transactions for a given period of time, the so-called view, which can also be changed if enough nodes are not satisfied with the current leader (view-change). During a view, the leader is getting

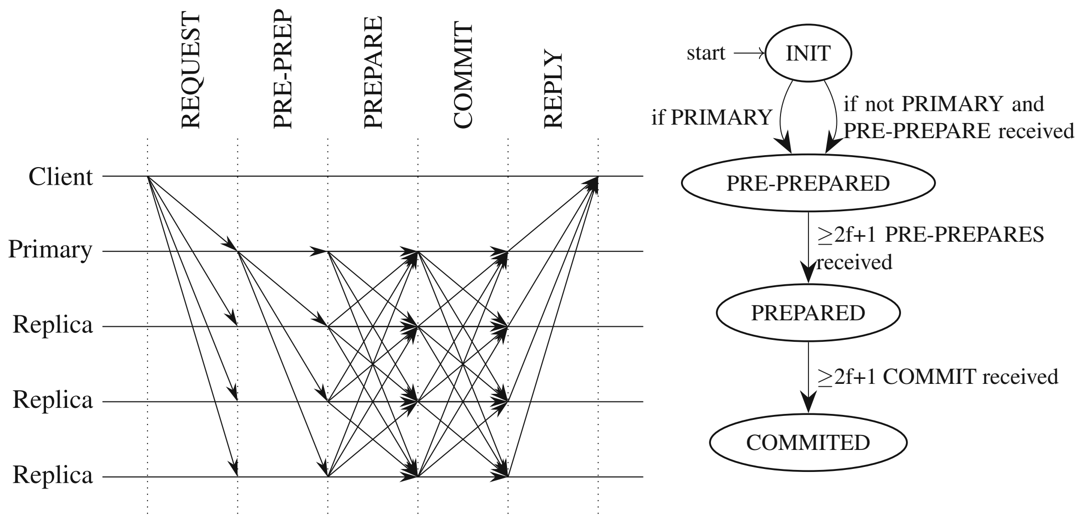


Figure 2.2: Message flow diagram of PBFT phases is shown on the left. Corresponding state diagram on the right shows conditions for protocol progress.

transaction requests from clients and orders them by assigning a transaction identifier. However, because the client does not know the current leader, it sends the request to all nodes. For our analysis, which is only looking at the performance of the leader consensus, this part can be omitted. Having received the request, the leader broadcasts a PRE-PREPARE message. If nodes receive a PRE-PREPARE they check transaction data and send a PREPARE message to all other nodes if it is consistent with their state. If nodes receive enough PREPARE messages from other nodes they enter the prepared state and send a COMMIT message to all other nodes. A node transitions into the committed state if it has received at least $2f+1$ (also including its own) COMMIT messages, and finally send a REPLY message to the client. The client considers the transaction to be committed when it has received $f+1$ identical REPLY messages. In fact, if f malicious node are still present in the committed state a total of $2f+1$ REPLY messages can be required for the majority voting at the client.

The protocol provides safety by only progressing if an honest majority is assured (at least $2f+1$ nodes are in the same state). In fact, a quorum of $q = \lceil \frac{n+f+1}{2} \rceil$ nodes is required to make progress, but this corresponds to the $q = 2f+1$ majority in the optimal case of $n = 3f+1$. Furthermore, the commit phase is used to guarantee this property within views and the commit phase is needed to assure it over view changes. Finally, liveness is guaranteed if the network satisfies weak synchrony conditions, which is often a reasonable assumption but could lead to large timeouts in software implementations and bad performance when the right timeout has to be found. Weak synchrony means that eventually after a bounded time Δt the network becomes synchronous.

The PBFT protocol also applies cryptography to implement authentic channels and transaction certificates. As an adversary cannot break cryptography this reduces its capacity on the channel to delaying and deleting messages, i.e., he cannot introduce new messages from nodes he is not controlling. From a practical perspective, however, an attacker will not always have control over all communication channels between all nodes and this model seems unnecessarily restrictive when it comes to performance evaluation. Nevertheless, with our model we want to optimize performance for normal operation and therefore assume the case of no adversary present but communication channels which suffer from imperfections. An adversary who would also have access to the communication channels and could arbitrarily delay messages would render performance modeling inadequate.

Related work. Two types of deployments for PBFT based consensus mechanisms can typically be found in the literature, LAN and blockchain. If deployed in a closed network within a single administrative domain, e.g. as a LAN based distributed lock manager like the ”5 Chubby nodes within Google” environment, best performance is achieved with the usage of UDP for message transmission. However, as the experiments of [105] showed, due to congestion, packet loss can occur even in the ideal LAN setting, and the triggered view-changes severely degrade performance.

If PBFT type of consensus is used in (permissioned) blockchain protocols, different assumptions and requirements hold [106, 107, 108], and results cannot easily be ported from one world to the other. Many transactions are typically batched, and consensus is organized in epochs comprising all currently pending transactions. The models also assume that a reliable channel can always be established with little overhead over unreliable channels and that the network buffers at nodes are infinite. In practice, they typically apply TCP or its secure variant TLS, if authenticity is required.

When it comes to performance analysis of BFT protocols, benchmarking is typically used to compare and estimate the performance of protocols [109]. The only known more systematic approach was presented in [110], which use Stochastic Reward Nets (SRN) to model “mean time to complete consensus”. However, they model the network as a reliable channel where the rate of message transmission between all pairs of peers is the same and fit individual distributions from measurements.

In summary, a large body of research exists in BFT and many protocols have been proposed and benchmarked, but only little is known when it comes to performance modeling of such protocols.

2.3.2 Modeling Packet Loss

In this section we focus on modeling transaction success and compare the usage of UDP and TCP transport protocols, and their impact on the performance of basic PBFT transactions. The optimistic case is analyzed with no malicious nodes present but possibly unreliable network conditions, which it the most interesting case for normal operation. The goal of this study is to fully leverage the redundancy inherent in PBFT to achieve short transaction times for high throughput respectively. Note that in the case of errors we can always fall back to a standard implementation for non-optimistic case with known performance degradation. Moreover, the model itself is generic and can easily be adapted to various scenarios by changing parameters accordingly.

2.3.2.1 Modeling Transaction Success

As mentioned before, if requests time-out they have to be re-requested and an optional view change can be triggered. View changes inflict high resource costs (especially on the network level); in addition, new requests can only be executed after the view change has been completed. Thus, it would be beneficial to know (or at least estimate) the probability that the system is able to successfully process a request a priori. This knowledge could significantly improve the overall system performance because if an unreliable transport mechanism, i.e., UDP, is used the system may switch over to reliable network communication, i.e., TCP, if the chance of a view change increases.

The employed PBFT protocol heavily relies on network communication between the replicas. Thus, delay and packet loss can have a tremendous impact on the overall system performance. There are basically two transport protocols: UDP (connectionless) and TCP (connection oriented). Both protocols are suited for our system (both provide disadvantages and advantages), however, UDP employs the least overhead and delay while TCP requires maintaining a connection and provides a reliable transport service. In order to minimize communication overhead and delay, UDP is favored. However, with increasing packet loss, we may run into the problem that nodes do not receive at least $2f+1$ messages from other nodes in a phase (cf. Figure 2.2). If this applies to more than $2f+1$ nodes, phases cannot be accepted anymore because of missing (distinct) messages and, therefore, requests will time-out. This leads to re-requesting timed-out requests and finally ends in even more requests timing-out. Thus, if the packet loss increases, TCP intuitively becomes superior to UDP, while trading performance for reliability. Thus, the question "when should TCP be used instead of UDP?" arises, as well as help and guidance on UDP usage to stay in the efficient regime as long as possible. For the following considerations $f \in [0, \lfloor \frac{n-1}{3} \rfloor]$, in order to have more than f correct working replicas we need $n - 2f > f \Rightarrow n > 3f$ replicas, thus the smallest number of needed replicas is $3f+1$ assuming f faulty ones. In the following we will provide a criterion which answers the aforementioned question based on probability theory.

Intuition tells us, that we would switch over to TCP if the expected number of nodes that receives more than $2f+1$ message is less than $2f+1$ in order to have enough replicas transitioning between the declared PBFT phases. Our goal is it to investigate how errors in the actual transmission between the BFT protocol phases propagate and how these errors influence the successful completion of a given transaction under the assumption of f faulty nodes. Without loss of generality, we assume that multicast is not in place and, therefore, nodes have to rely on unicasts. If messages are attacked by man-in-the-middle attacks and are altered (thus altering the recalculated digest) we assume that the message is lost.

Taking a look at Figure 2.2 and having in mind that messages may get lost we have the following phases if a request is received by the primary:

1. **PRE-PREPARE**: The primary sends a PRE-PREPARE message to all nodes (including itself). Nodes can only successfully commit a transaction if they successfully accept all phases, this also includes the reception of a PRE-PREPARE message which actually fires off the consensus protocol. Assuming that there is packet loss, $m-1$ out of $n-1$ ($m,n \in \mathbb{N}, m \leq n$) nodes may receive a PRE-PREPARE message. The primary itself sends $n-1$ PRE-PREPARE messages to the other nodes, our of which $2f$ have to be received in order to have enough nodes ($2f+1$) in the next state.

2. **PREPARE**: m (including the primary) nodes broadcast a PREPARE message to all n nodes. Each node has to receive at least $2f+1$ PREPARE messages to successfully accept the PREPARE phase and in order to transition into the next phase. We start with m nodes and may end up with only k out of m nodes ($k,m,n \in \mathbb{N}, k \leq m \leq n$) receiving at least $2f+1$ PREPARE messages. A node in this phase will only need to receive $2f$ distinct PREPARE messages from $m-1$ nodes because one message is sent to itself.

3. **COMMIT**: k nodes transition into this phase and broadcast a COMMIT message to all n nodes. Since only k nodes successfully accepted the previous phase we again have at most k nodes which can successfully accept the last phase. Thus, we have j out of k nodes ($j,k,m,n \in \mathbb{N}, j \leq k \leq m \leq n$) which again need $2f$ messages from $k-1$ nodes.

4. **REPLY**: j nodes arrive in this phase and will send a REPLY to the client. The client sees its request as fulfilled if it receives $f+1$ identical REPLY messages, i.e., $f+1$ REPLY messages in total (best case), or $2f+1$ messages if malicious nodes are also considered (worst case), out of j possible ones.

We denote the random variables for the number of nodes reaching certain phases as follows: M (PRE-PREPARE), K (PREPARE), J (COMMIT), and S (REPLY). We do not take into account the reception of a request. If a request is not received, no transaction will be triggered. The final number of nodes, thus, relies on the number of nodes that are able to successfully accept each phase. We assume that the probability of successfully transmitting a packet is independent and identically distributed.

The expected value $\mathbb{E}[S, J \geq 2f+1, K \geq 2f+1, M \geq 2f+1]$ as a function of successful transmitting a message/packet, should suffice the following properties:

1. Let $f \in [0, \lfloor \frac{n-1}{3} \rfloor], \forall p_{l,i}, p_{l,j} \in]0,1[, p_{l,i} \leq p_{l,j}$:

$$\mathbb{E}[S, J \geq 2f+1, K \geq 2f+1, M \geq 2f+1](p_{l,i}) \leq \\ \mathbb{E}[S, J \geq 2f+1, K \geq 2f+1, M \geq 2f+1](p_{l,j}).$$

2. Let $p_l \in]0,1[$:

$$\mathbb{E}[S, J \geq 2f_i+1, K \geq 2f_i+1, M \geq 2f_i+1] \geq \\ \mathbb{E}[S, J \geq 2f_j+1, K \geq 2f_j+1, M \geq 2f_j+1],$$

$\forall f_i, f_j \in [0, \lfloor \frac{n-1}{3} \rfloor], f_i \leq f_j$.

The probability that a client receives s replies from j nodes, where j out of k nodes accepted the COMMIT phase, k out of m nodes accepted the PREPARE phase, and m out of n nodes successfully accepted the PRE-PREPARE phase is given by Equation (2.1). There, we define p_l as the probability for successfully transmitting a packet with length l (we will later derive this probability or provide means to measure it). The actual probability p_l does depend on the underlying transport protocol T. Furthermore, $\mathbb{P}_T(X = k|y, p_l)$ denotes the probability that k out of y packets/messages are successfully transmitted given p_l using transport protocol T.

Finally, in Equation (2.1) and for all following treatment we assume that we have an optimal node configuration of $n = 3f+1$, which require each node to wait for $q = 2f+1$ messages (including self-message) to transition to the next phase during any-to-any intermediate phases.

$$\begin{aligned} \mathbb{P}(S = s, J = j, K = k, M = m) = \\ & \mathbb{P}_T(X = s|j) \\ & \cdot \binom{k}{j} \mathbb{P}_T(X \geq 2f|k-1)^j (1 - \mathbb{P}_T(X \geq 2f|k-1))^{k-j} \\ & \cdot \binom{m}{k} \mathbb{P}_T(X \geq 2f|m-1)^k (1 - \mathbb{P}_T(X \geq 2f|m-1))^{m-k} \\ & \cdot \mathbb{P}_T(X = m-1|n-1) \end{aligned} \tag{2.1}$$

Based on Equation (2.1) we can compute the success probability for a given transaction as follows. If we sum over all successful cases we get the overall success probability which is a key indicator of the overall performance. Transactions have to go through with high probability to achieve good performance. This will avoid additional retransmissions or view changes which substantially slow down the system.

The success probability is given in Equation (2.2) also for the optimal node configuration and all honest nodes ($S \geq f+1$).

$$\begin{aligned} P_{succ} :=\ & \mathbb{P}(S \geq f+1, J \geq 2f+1, K \geq 2f+1, M \geq 2f+1) = \\ & = \sum_{m=2f+1}^{n} \sum_{k=2f+1}^{m} \sum_{j=2f+1}^{k} \sum_{s=f+1}^{j} \mathbb{P}(S=s, J=j, K=k, M=m) \end{aligned} \tag{2.2}$$

Alternatively to the success probability P_{succ} it can be interesting to look at the probability distribution of S or their expectation, which can also be computed from Equation (2.1). The model presented so far was based on optimal node configurations with least overhead. However, a further generalization of the model can be made for non-optimal node settings. PBFT is based on the idea of a quorum of nodes which is guaranteed to eventually be in the same state thus defining the actual status of the system. The size of the required quorum to make progress in PBFT is given by $q = \lceil \frac{n+f+1}{2} \rceil$, which results in $q = 2f+1$ for the optimal case of $n = 3f+1$. However, if $\mathbb{P}_T(X \geq q-1|m-1)$ and $\mathbb{P}_T(X \geq q-1|k-1)$ are used for the respective terms in Equation (2.1) and respective Equation (2.2) is formulated as $\mathbb{P}(S \geq f+1, J \geq q, K \geq q, M \geq q)$ arbitrary configuration can be evaluated.

2.3.2.2 TCP vs. UDP

In the following we derive the probability $\mathbb{P}_T$ for UDP. Assume that the probability $p(l)$ of not encountering a packet loss when a message (with length l) is transmitted using UDP is given. Then $p_{l,UDP} := p(l)$ because UDP does not bother whether a message has been successfully sent. The probability of receiving j out of n messages using UDP reads as

$$\mathbb{P}_{UDP}(X = j|n) = \binom{n}{j} p(l)^j (1-p(l))^{n-j}. \tag{2.3}$$

Based on $\mathbb{P}_{UDP}$ we can also compute $\mathbb{P}_{UDP}(X \geq j|n)$ by summing up the good cases as shown below in Equation (2.4). Thus we can directly use both versions of $\mathbb{P}_{UDP}$ as an instantiation of $\mathbb{P}_T$ in Equation (2.1) and derive the concrete probabilities for each possible system state as well as the success probability and expected number of replies.

$$\mathbb{P}_{UDP}(X \geq j|n) = \sum_{i=j}^{n} \mathbb{P}_{UDP}(X = i|n) \tag{2.4}$$

To get a first idea of the basic behavior of the system when using UDP we computed the success probability P_{succ} over increasing transmission probability p_l. The result for different failure modes f and according quorums ($q = 2f+1$) and nodes ($n = 3f+1$) in a standard configuration is shown in Figure 2.3. From the graph, it can be seen that higher f are favorable in tolerating small errors but perform worse if the transmission

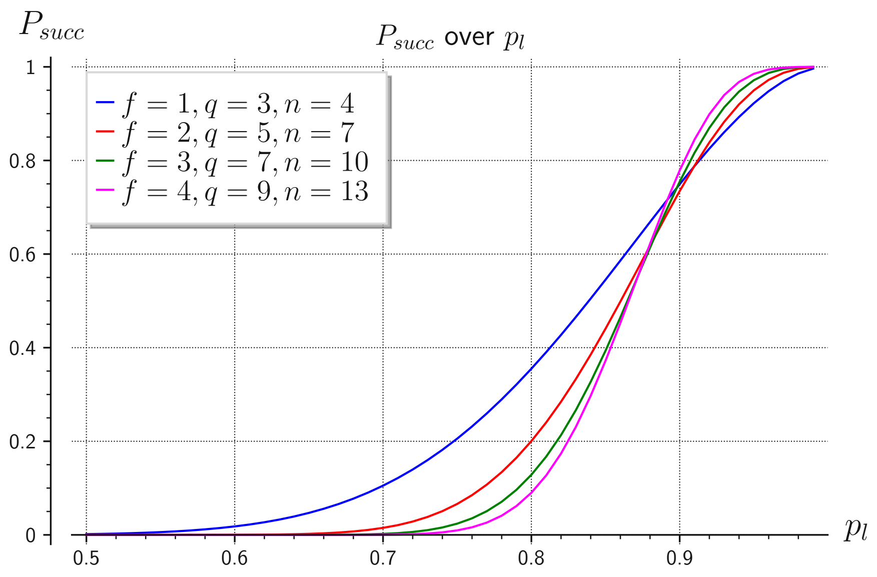


Figure 2.3: Basic UDP model for different standard configurations.

errors ($p_{err} = 1 - p_l$) gets bigger. However, from a system operator point of view it is clear, that only high success probabilities will lead to god overall performance, because after every failure the transaction has to be rescheduled or even worse, a view change has to be triggered. Both cases induce additional overhead which should be reduced to reasonable values. In practice we would expect values of 99% or 99.9% and higher, depending on the use case. Thus, in typical working regimes with low error rates we experience better behavior in systems with larger f comprising more redundancy as can be also seen in Figure 2.4. Although P_{succ} is a very good measure to tune system performance, a service provider may also use the expected value $\mathbb{E}[S, J \geq 2f+1, K \geq 2f+1, M \geq 2f+1]$ to decide whether it should switch from a UDP based transmission to TCP. A criteria for switching the transport protocol could be $\mathbb{E}[S, J \geq 2f+1, K \geq 2f+1, M \geq 2f+1] < 2f+1$ because at least $f+1$ (in the best case) or $2f+1$ (in the worst case) replies are needed by the client accepting the transaction.

The major advantage of TCP usage is that we gain reliable connections at the expense of (even more) delay for transactions to complete. In fact, using TCP in theory we can assume a constant transaction success probability of one if we allow for an infinite number of retransmissions. Nevertheless, even if transactions are guaranteed to complete eventually, the resulting performance can be devastating and render the service unusable, because of the many retransmissions needed. Even in the good case with no retransmissions does TCP perform slower, because of the intrinsic acknowledgment mechanism which always requires two time the network latency to complete. Thus, it is important to also understand behavior of TCP in detail in order to minimize the impact of retransmissions, and not only rely on its guaranteed delivery property.

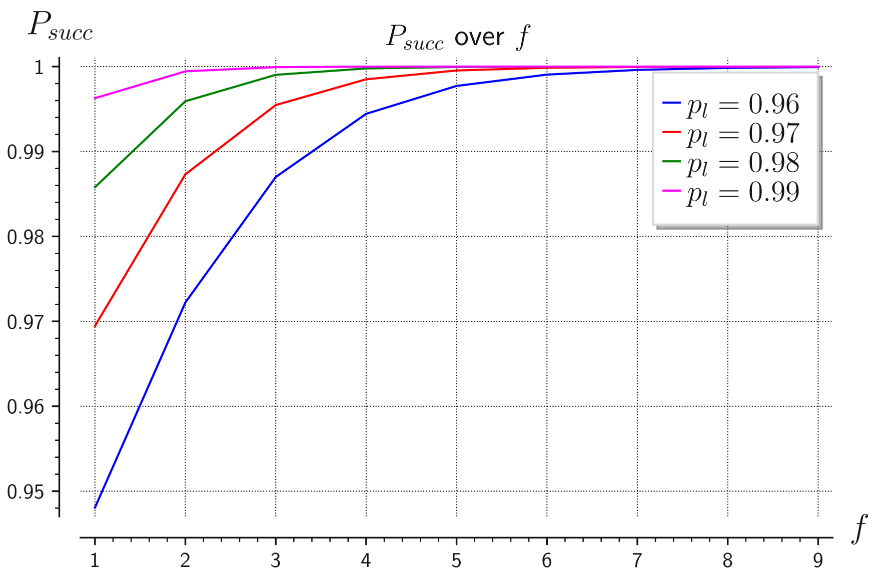


Figure 2.4: UDP success probability with increasing f for small transaction errors.

2.3.2.3 Number of Retransmissions

In order to shed some light on the probability and expected value of TCP retransmissions (and UDP retransmissions in the following), we assume that TCP connections are already set up and we only account for the transmission of data segments/messages. Again, we assume that the probability of successfully transmitting a packet of length l over the wire/channel is given by $p(l)$. However, a segment is only successfully transmitted using TCP if we receive an acknowledgment (ACK) otherwise a time-out will trigger, and a retransmission of the segment will be initiated. Therefore, both (segment + ACK) have to be transmitted successfully. We do not consider any extensions of TCP. A message may be divided into several segments which all have to be successfully transmitted. The probability of successfully transmitting a segment reads as

$$\mathbb{P}(\neg M|p) = p(l)(1-p(ACK)) + (1-p(l))$$
$$\mathbb{P}(M|p) = p(l)p(ACK).$$

If $p(l) \approx p(ACK)$ then we have $\mathbb{P}(M|p) = p(l)^2$, in general we have $p(l) \leq p(ACK)$ and we obtain $\mathbb{P}(M|p) \geq p(l)^2$. We derive the probability of successfully transmitting a segment with a certain number of allowed retransmissions $m \in \mathbb{N}_0$ by Equation (2.5). In order to derive the probability of successful transmitting a TCP segment, we model this process $(X_n)_{n\in\mathbb{N}}$ by a Markov chain with the state space $\Omega = \{1,2\}$ with the following transition matrix

$$P = \begin{pmatrix} 1 & 0 \\ \mathbb{P}(M|p) & 1-\mathbb{P}(M|p) \end{pmatrix}.$$

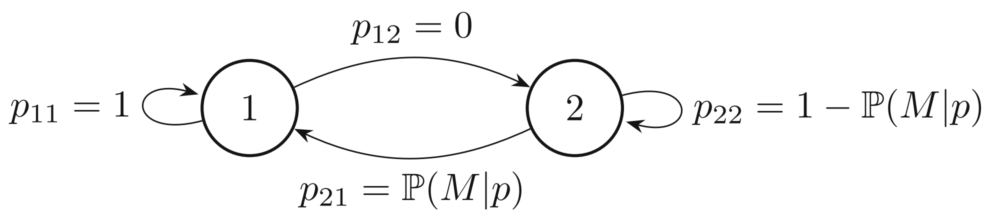


Figure 2.5: Markov chain for TCP retransmission modelling. p_{ij} are the transition probabilities of P. The system starts in state 2 and after a successful transmission of message an acknowledgment it resumes in state 1. However, for every unsuccessful attempt it stays in state 2 and initiates a retransmission.

The structure of the Markov chain is also shown in Figure 2.5. In our model we assume that we start in state 2 and transition with ever try to transmit a message. Eventually we will arrive in state 1 after a successful transmission, i.e., message and acknowledgment are received. However, for our analysis we are interested in the probability of a successful transmission after a maximal number of retransmissions r, which we call $\mathbb{P}_{RETCP}(M|r,p)$. According to the Kolmogorov–Chapman equation we obtain this quantity from the probability of being in state 1 after $r+1$ transitions when starting from state 2 as read

$$\mathbb{P}_{RETCP}(M|r,p) := \mathbb{P}(X_r = 1|X_0 = 2) = P^r_{2,1} = \mathbb{P}(M|p)\sum_{k=0}^{r}(1-\mathbb{P}(M|p))^k. \tag{2.5}$$

Equation (2.5) can also be easily verified by applying induction. However, we still have to prove the informally introduced property of guaranteed delivery for the case of infinite retransmissions. Intuitively this property can also be seen from the Markov chain, because state 1 is the only absorbing state, but we are going to show it formally in the following.

Proposition 2.3.1. Let $(\Omega, \mathcal{A}, \mathbb{P}_{RETCP})$ be a probability space, where $\Omega = \{M, \neg M\}$, with the states accounting for a successful and not successful transmission of a TCP segment. Where, $(\mathbb{P}_{RETCP})$ is conditional probability measure given a certain number of retransmissions $r \in \mathbb{N}_0$. Then the following holds

$$\lim_{r\to\infty} \mathbb{P}_{RETCP}(M|r,p) = 1.$$

Proof. Using Equation (2.5) and the fact that the probability of successfully transmitting a message $(\mathbb{P}(M|p))$ is Bernoulli distributed, we easily obtain the intuitively result:

$$\lim_{r\to\infty} \mathbb{P}_{RETCP}(M|r,p) = \mathbb{P}(M|p) \lim_{r\to\infty} \sum_{k=0}^{r} (1-\mathbb{P}(M|p))^k$$
$$= \mathbb{P}(M|p) \lim_{r\to\infty} \frac{1-(1-\mathbb{P}(M|p))^{r+1}}{\mathbb{P}(M|p)} = 1$$

$\square$

Corollary 2.3.1. Equation (2.5) can also be written as

$$\mathbb{P}_{RETCP}(M|r,p) = 1-(1-\mathbb{P}(M|p))^{r+1}.$$

A message sent by our BFT solution may be split up into several TCP segments. Assuming an *i.i.d.* packet loss, the success probability of a message which is divided into u different segments finally reads as

$$p_{l,TCP} := \mathbb{P}\left(\bigcap_{j=1}^{u} M_j|r,p\right) = \prod_{j=1}^{u} \mathbb{P}_{RETCP}(M_j|r,p) = \tag{2.6}$$
$$(1-(1-\mathbb{P}(M_1|p))^{r+1})^{u-1}(1-(1-\mathbb{P}(M_u|p))^{r+1}),$$

there are u segments where $u-1$ are of the same size and the k-th segment may have a smaller length than its predecessors. However, because the messages sent between replicas in PBFT are rather short, TCP segmentation is very unlikely to happen and we can safely consider $u = 1$ for the rest of this analysis. Now, based on $p_{l,TCP}$ the probability that a replica receives k messages from n messages sent using TCP reads as

$$\mathbb{P}_{TCP}(X = k|n, p_l) = \binom{n}{k} p_{l,TCP}^k (1-p_{l,TCP})^{n-k}. \tag{2.7}$$

The probability $\mathbb{P}_{TCP}$ can then be used in Equation (2.1) to calculate overall node probabilities and subsequently the success probability for TCP for a given number of retransmissions. A comparison of different levels of retransmission is shown in Figure 2.6. P_{succ} can be significantly increased for higher number of r. Interestingly without any retransmission, TCP behaves worse than UDP. This is due to the fact that the channel is used two times for a successful transmission compared to one time with UDP.

Finally, we may also use the insights gained from TCP for UDP. If we set $\mathbb{P}(M|p) = p(l)$ in Equation (2.5) we have the case of UDP with retransmission (repetition code). In this case we can conduct the very same analysis as done previously in UDP, but with nodes sending multiple copies of the messages to reduce the transmission error. In this approach we can try to optimize how often each BFT node should duplicate (incl. sending) a message in order to arrive at high success probability. This strategy could

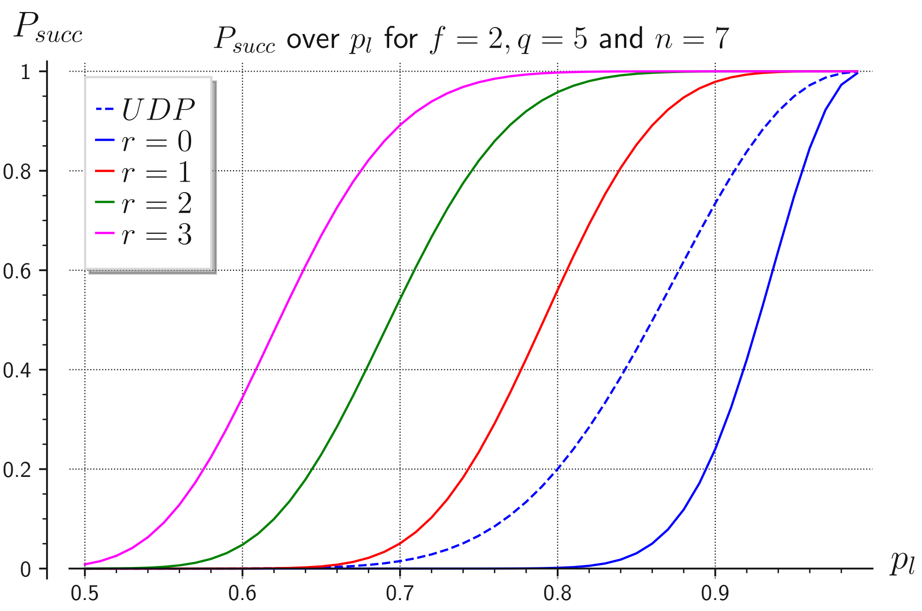


Figure 2.6: Influence of different allowed TCP retransmissions *r* on the success probability of transactions (together with UDP as baseline).

be very efficient, because in UDP the duplicates are sent subsequently along with the original data, which prevents from large delays in acknowledgment-based protocols. However, because the redundant data is sent proactively it increases traffic and could therefore cause other negative effects. Nevertheless, before switching to TCP, the BFT nodes may try to send each message *r* times.

2.3.2.4 Exploring the Design Space

In the following we discuss the most important parameters and improvements to tune system deployment to optimize the performance.

Forward Error Correction (Repetition Code). To improve the probability for a packet being transmitted successfully in UDP without the introduction of a handshake mechanism like in TCP, we could apply forward error correction (FEC). The simplest way would be to apply repetition codes, which send the data multiple times. This concept was already mentioned in the previous chapter, where the prerequisites were discussed along with the theory. In the case of an immediate retransmission in UDP over a lossy channel with transmission probability $p_l = p(l)$, we get a new transmission probability p_{l_2} for sending an additional duplicate. From a channel user point of view, the redundancy decreases the effective packet loss in the channel substantially by doubling the traffic. For our channel model with i.i.d. loss, a message is only received if not both transmitted messages are lost, which leads to $p_{l_2} = 1-(1-p_l)^2$. This result is the same as if Equation (2.5) was used with $\mathbb{P}(M|p) = p_l$ and $r = 1$.

In Figure 2.7 we show the simulation results of the influence of different amounts of retransmissions (replications) r on the success probability of transactions. The impact is huge already for small number of retransmissions and P_{succ} close to one can be achieved up to relatively high error rates. However, given that about $2(n+n^2)$ messages are exchanged in total during a transaction without retransmission, the amount of messages increases quadratically which could lead to problems on the channel (congestion, latency, additional loss). Therefore, the system should always operate in the optimal regime where a minimal number of duplicates is produced.

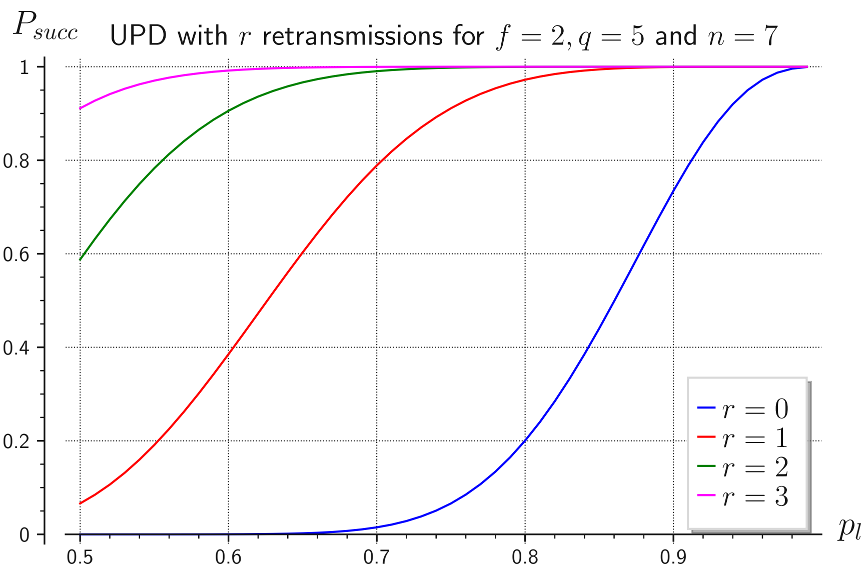


Figure 2.7: Influence of retransmissions for repetition codes in UDP.

Additional Redundancy in Nodes. An alternative solution would be the use of additional nodes beyond the optimal $3f+1$ robustness bound. For the standard case with reliable channels it does not make sense to go beyond the optimal number of nodes, because no robustness is gained. However, from a performance perspective, increasing the amount of nodes $3f+1+x$ leads to higher success probabilities in the UDP case and could improve system performance if switching to TCP could be pushed to higher error rates or even avoided for the expected communication channels. Nevertheless, increasing the number of nodes also requires an increase in the quorum size for the protocol to $\lceil \frac{n+f+1}{2} \rceil$, which is not considered in the formulas above but will be used in the simulations.

The beneficial effect of additional nodes is shown in Figure 2.8. Although not as significant as repetition codes, we see a substantial improvement for smaller transmission errors. This could be relevant for many use cases, because the overhead on messages produced is small compared to message retransmission.

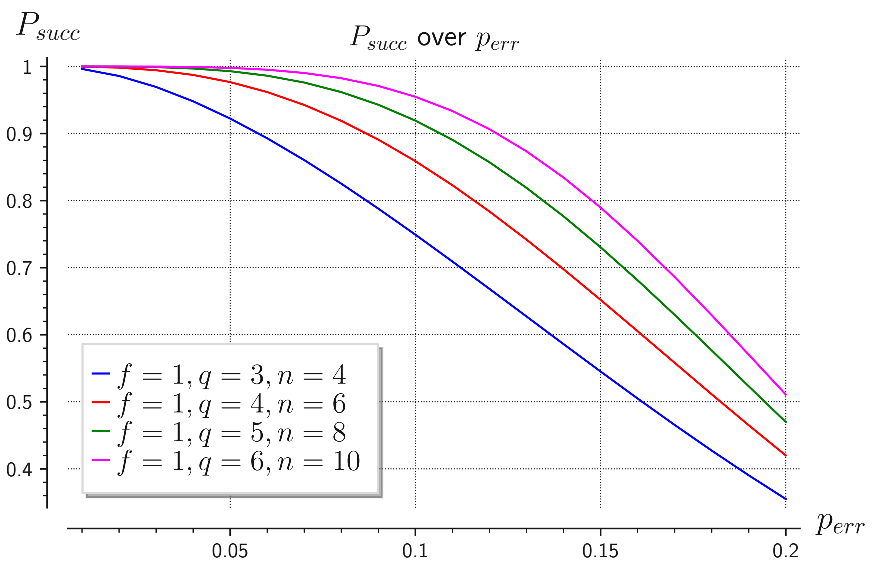


Figure 2.8: Influence of additional nodes on the transaction success probability for UDP.

2.3.3 Simulations and Measurements

In order to investigate the performance of the proposed approach further and to validate the theoretical results we also simulated the BFT protocol as described in Section 2.3.2. We selected OMNet++ 5.6[9] as the underlying simulation environment and use INET 3[10] as the network simulator on top of which we implemented the altered PBFT protocol using TCP and/or UDP as transport protocol for exchanging messages on the application layer. We use a simplified topology where *n* replicas are connected through a router. Furthermore, in our simulation we did not consider the computation times of nodes. Especially the overhead of the cryptographic mechanisms also needed in a full implementation are assumed to be negligible for this analysis. Finally, we also benchmarked a real PBFT implementation developed in a project for multi-cloud storage to verify the results from the event simulation and test improvements.

2.3.3.1 Model Validation

In a first step we validated the accuracy of the theoretical model with the OMNet++ simulation. We set the bandwidth of each link (between node and router) to 100 Mbps, and the delay is truncated normal distributed (always ≥ 0) with mean 20ms and a variance of 5ms. We varied the bit error rate of the channel from 0 to $13 \cdot 10^{-5}$ in 10^{-5} steps and measured the actual packet loss seen at the transport layer. We used 20 replicas, a message size of 128 bytes, and we assumed the maximum number of faulty nodes (6 in the case of 20 nodes). For each simulation run we did 100 requests and for each

[9]https://github.com/inet-framework/inet/issues/75
[10]https://inet.omnetpp.org/

simulation parameter configuration we did 20 repetitions. Figure 2.9 depicts the probability using the model provided in Equation (2.1) ($P_{succ} := \mathbb{P}(S \geq 2f+1, J \geq 2f+1, K \geq 2f+1, M \geq 2f+1)$) and the data obtained by the experiment. It is evident that the theoretical model fits the observed experimental data. Thus, the presented theoretical model can be used to explore basic behavior and the presented analysis is a solid foundation for further performance optimization.

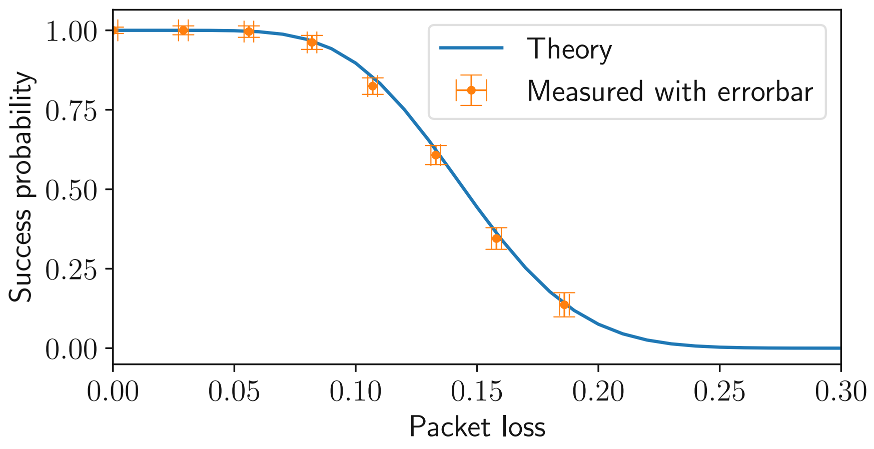


Figure 2.9: Transaction success probability as a function of packet loss obtained by experiments vs. Equation (2.1) using UDP.

2.3.3.2 Simulation Results

To better understand and improve the UDP behavior we also explore the design space available to improve success rates and analyze their impact on the latency in our simulation environment. As discussed, two immediate and easy to realize options exist for the improvement of the success probability of individual transactions P_{succ}. One is to increase the number of nodes and the other to better cope for channel losses by means of forward error correction (FEC).

To prevent transactions from failing by losing synchronization at certain nodes, increasing the number of nodes seems a good way to increase resilience. However, the main configuration parameters of a BFT system (n, f) cannot be freely chosen and have to fulfill certain requirements. In general, a setting with $n = 3f+1$ is believed to be optimal and typically used, as the quorum size is also minimal with $2f+1$. We therefore compared settings with different robustness f from a performance point of view and for the suitability of UDP. The results are shown in Figure 2.10, and it can be seen that with increasing number of nodes n, the success probability P_{succ} also increases. For settings with a higher number of nodes (e.g. $n >= 19$) we see higher transaction success even for substantial packet loss, which indicates that application of UDP can be practical. However, beyond about 12% packet loss the effect reverses and configurations with more

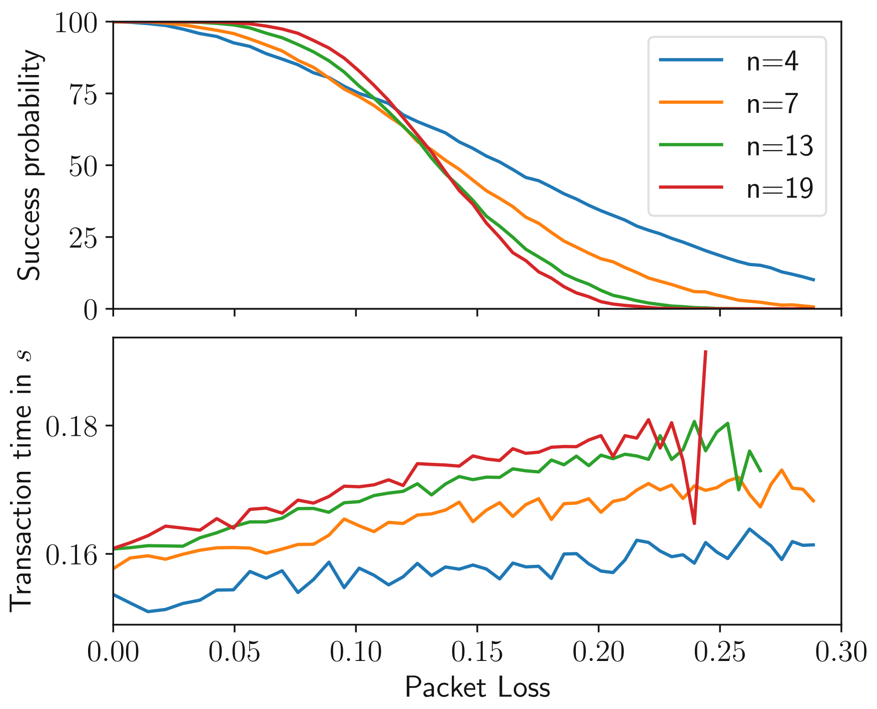


Figure 2.10: Success probability over increasing packet loss for UDP with different f and minimum node configuration $n = 3f + 1$.

nodes behave worse. Furthermore, as expected the transaction times are much better with UDP compared to protocols using acknowledgements and only slightly increases with higher packet loss and number of nodes.

If FEC is used, repetition codes are the most efficient solution in our case, as the amount of packets should be kept low and only short messages are exchanged in multiple rounds. The effect of repetition codes is shown in Figure 2.11. As expected it raises P_{succ} substantially by reducing the effective packet loss on the channels through proactive retransmission of packages. This comes at the cost of an (unnecessary) increase of messages transmitted. Interestingly, the overall transaction time is not affected if enough bandwidth is available and the good timing behavior is maintained in all situations.

Given an accurate channel model and some bandwidth left on the network, this method turned out to be the most effective. However, if the channel changes behavior or is not known at all, this approach could lead to completely different results, e.g., for burst failures this FEC strategy would fail. Additionally, overhead on the network is produced and it should only be used if enough bandwidth is available and no additional congestion is induced.

Besides the evident options presented above, it is natural to ask if going beyond optimal configurations of $n = 3f + 1$ could make sense from a performance point of view, although not necessary from a robustness perspective. We suspected that adding

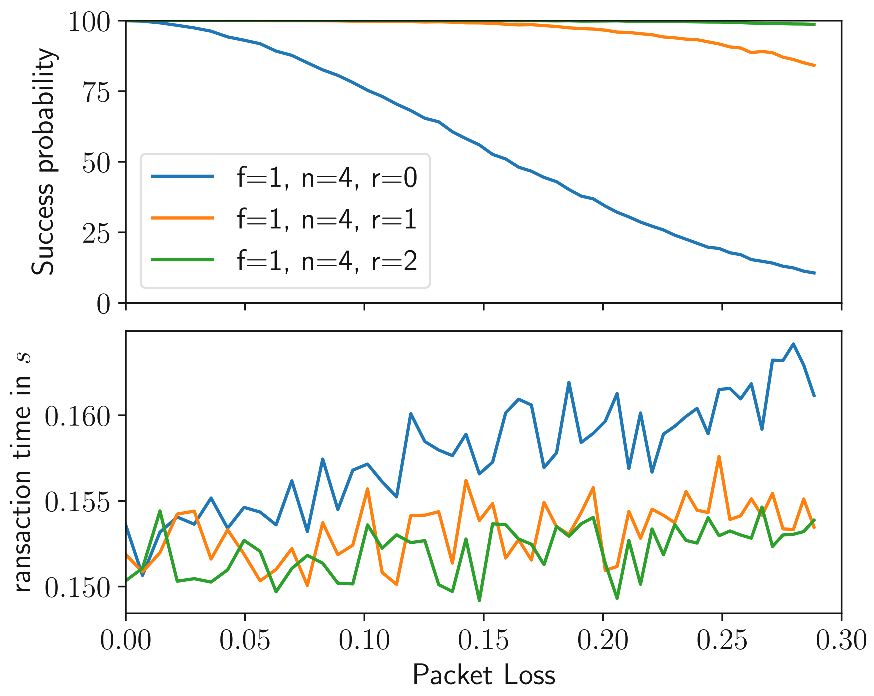


Figure 2.11: Success probability over increasing packet loss for UDP with $f = 1$ and increasing repetitions r.

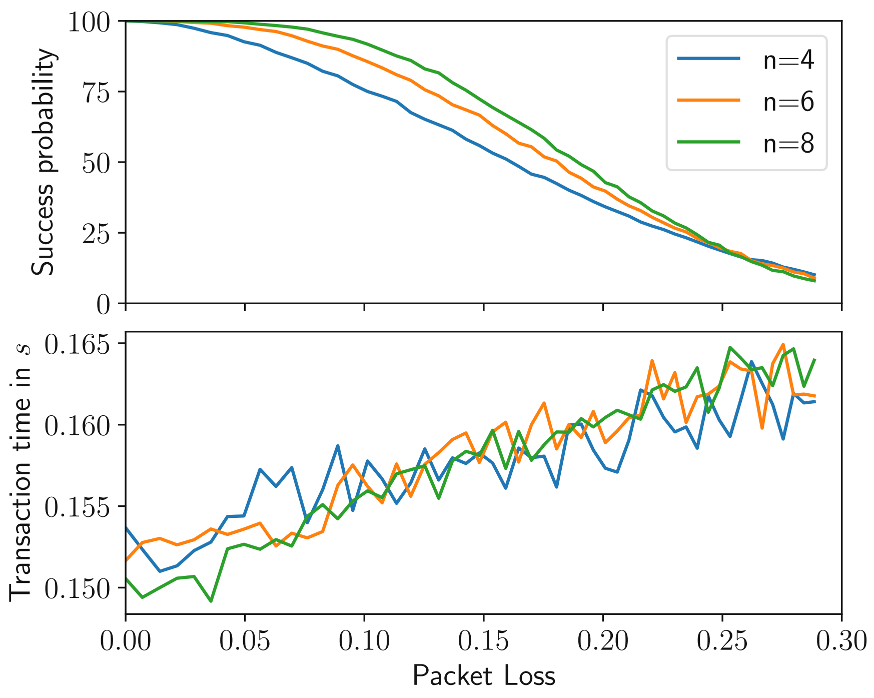


Figure 2.12: Success probability over increasing packet loss for UDP with $f = 1$ and increasing node redundancy

additional nodes could help to improve UDP usage even with certain packet loss, but is was not clear how it would impact the overall latency and how big the improvement in success probability would be.

In Figure 2.12 we show the results of this analysis. With additional nodes the success probability with lossy links can be increased and at the same time we get even shorter transaction times. The effect is best seen for small configurations which can benefit from this idea. Nevertheless, because PBFT is a quorum-based protocol, nodes have to be added pairwise. Adding a single node to an optimal configuration degrades performance, because the required quorum also increases, i.e., if more than $(n+f)/2$ servers have to be in the same phase, the servers have to wait for more PREPARE and COMMIT messages.

Finally, in our simulations we also verified that TCP behaves worse for increasing packet loss as is shown in Figure 2.13. Even for no losses the transaction time was already almost twice as high as with UDP. This can be easily explained by the basic nature of TCP using acknowledgements. Even worse, with increasing packet loss the transaction time started to rise to unexpectedly high values in the seconds range and due to timeout behavior we even saw some transactions not finishing. This result confirmed our findings from the first experiments mentioned in the introduction.

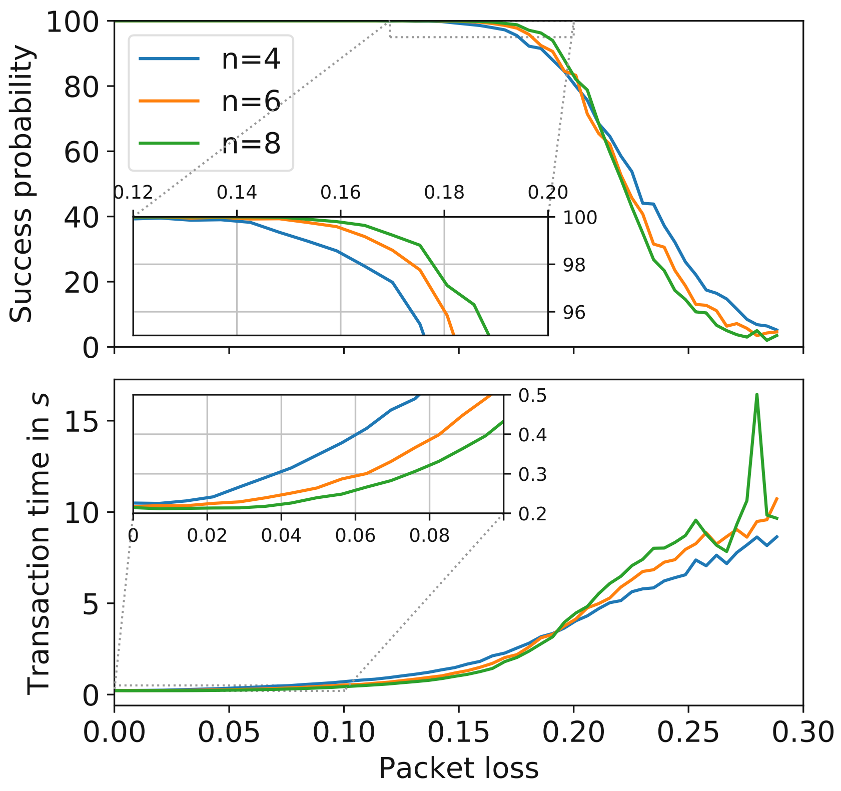


Figure 2.13: Success probability over increasing packet loss for TCP with $f = 1$ and increasing node redundancy

Although TCP is an extremely versatile and attractive protocol for many situations to build reliable channels over unreliable ones, for the BFT type of interactive protocols with many short messages sent among nodes it turned out to be not a good fit. This is also aligned with our intuition of TCP being throughput optimized for channels with high bandwidth-delay product. Nevertheless, in situations with a lot of uncertainty about the channel and high losses it can be a valuable tool to increase the transaction rate in such rough conditions. Surprisingly we also found that the success probability was not 1 in all situations, and even with long timeouts some of the transactions did not complete in scenarios with higher packet loss. This is because of the limit of 12 retransmissions in the TCP implementation of INET.

Finally, we also tried to compare different TCP types to show their behavior, but we could find no significant differences between the algorithms implemented in INET (Tahoe, Reno, New Reno). This may be due to a known problem of this framework [111].

2.3.3.3 System Measurements

In addition to the simulation, we also performed measurements on a real implementation done in Python [1]. To establish similar conditions for our comparison we opted for an emulated network on a single Linux PC deployment where each node was run as a separate instance and the local network stack was used for communication. To evaluate different networking conditions the Linux *netem* kernel module [112] was used to provoke packet delay and network loss. This setup provided the stable and controllable environment we needed to verify the results of the simulation and the analytical model. For the measurements the same channel settings were used as in the simulation, i.e. normally distributed network latency with 40ms mean and 10ms variance (equals 20ms mean and 5ms variance in the star topology used in the simulation) with an additional packet loss varying from 0 to 30%.

The comparison of the measurements and the simulation is shown in Figure 2.14. Overall, the measurements taken from the PBFT implementation show a very good match to the simulated results and show that model and simulation are correct and can be used to estimate performance. The success probability in particular resembles the simulated values well. The measured latency shows a smoother behavior over increasing packet loss corresponding to smaller variances in the measurements which can be attributed to buffering effects in the software and OS stack used. We also found a slightly higher transaction time in the real implementation for increased packet loss, however, even for very high packet loss it was within 10% margins.

Additionally, in our protocol analysis we found that especially the PRE-PREPARE phase is susceptible to packet loss and could greatly impact the overall performance in terms of successful transaction termination. This is due to the leader-based structure of the core view-consensus protocol in PBFT. In such a protocol one node initializes the transactions by distributing relevant data to all other nodes, the backups. In this

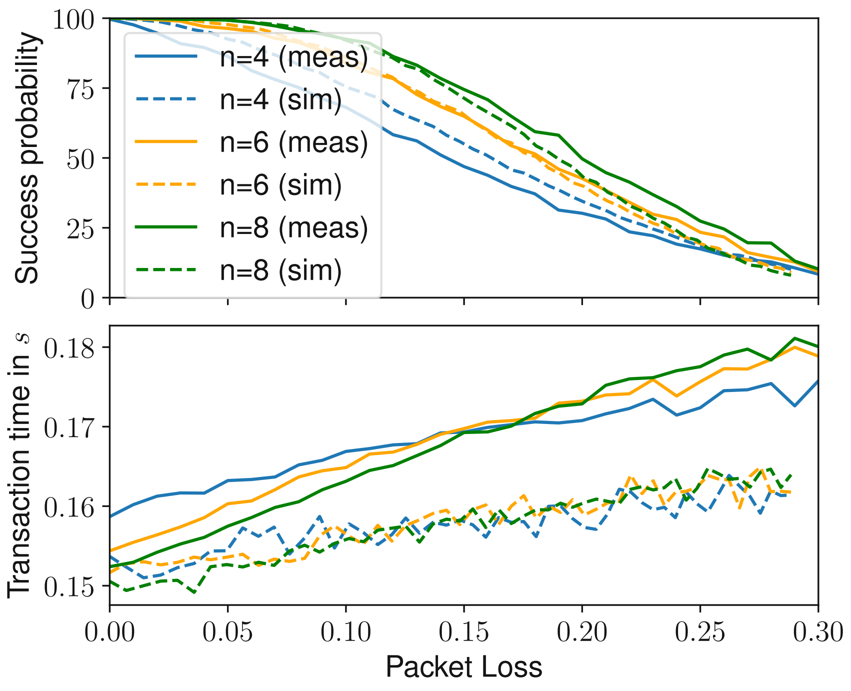


Figure 2.14: Comparison of measured value from implementation to simulated vales for UDP. Measured values are drawn with continuous lines and simulated values dashed.

phase the protocol has less redundancy compared to later phases. Interestingly, adding redundancy by message repetition only in this phase gives a high increase in success probability with relatively low additional communication cost. With one retransmission in the PRE-PREPARE phase only $n-1$ packets are added, compared to n^2 packets per retransmission in the other phases, but the success probability can be substantially increased. To verify this behavior we measured the increase in success probability for one and two retransmissions in the PRE-PREPARE phase.

The results are presented in Figure 2.15, and the data show that adding one retransmission in the PRE-PREPARE phase leads to the same or even higher $P_{success}$ as adding two additional nodes, but saves a lot of communication overhead. Given a total of $(r_{pp}+2)n+2n^2$ messages sent in the view-consensus protocol with its three phases, with r_{pp} being the number of retransmission in the pre-prepare phase, the overhead introduced with one additional retransmission is low. For systems which tolerate one faulty node out of 4 nodes we get about 11% of message overhead, with 5 nodes we see 9% overhead and about 7.7% overhead are required for 6 nodes. This leads to a significant improvement compared to the communication overhead introduced by adding an additional node without retransmission to increase P_{succ}, i.e., a total of 53% more messages must be sent if n is increased from 4 to 5. Nevertheless, both measures can be combined to get UDP performance up to 5% packet loss and more if two additional nodes are combined with retransmission in the pre-prepare phase as an example.

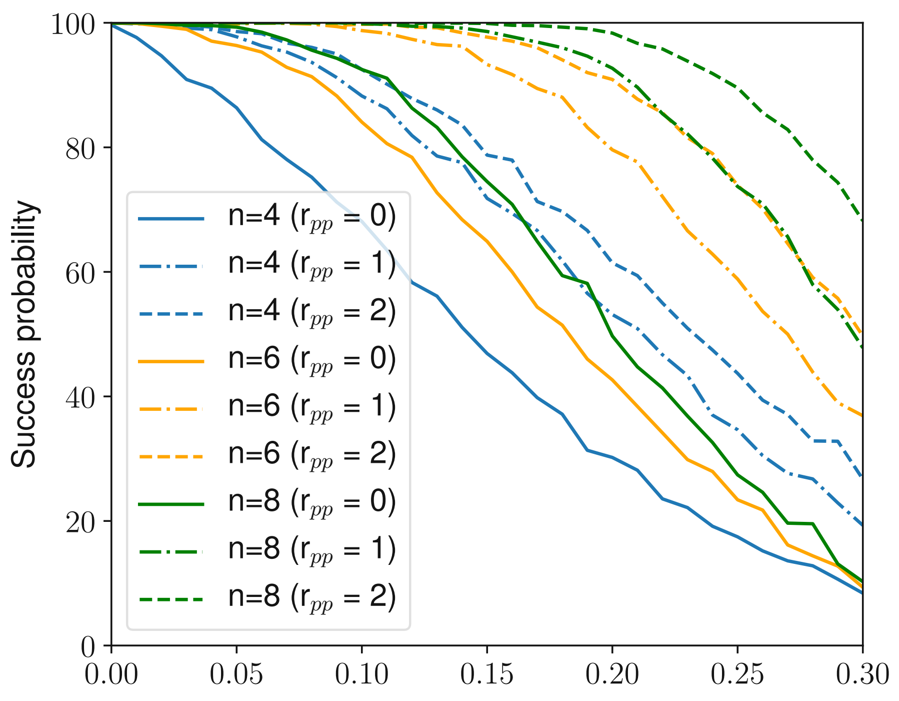


Figure 2.15: Measured success probability with retransmission only in the pre-prepare phase. Lines without retransmission are depicted as continuous lines and results with 1 (2) retransmission of pre-prepare messages are drawn with dash-dot (dashed) style.

2.3.4 Interpretation

From the results we see that careful design on the network layer is essential for PBFT and protocols with similar communication patterns to achieve best performance in challenging network settings. Especially multi-cloud configurations fall in this category, but single cloud deployments with a certain level of geo-separation could also introduce substantial latencies. As can be seen from the measurements taken at CloudPing [113], latencies between continents are crucial, for example between Europe and North America, where they range from $100-150$ms (50th percentile). Even within a single continent they are the dominating factor for BFT performance, e.g., they go up to $40ms$ (50th percentile) for servers within Europe. Thus even intra-region BFT will face substantial latencies and has to rely on UDP for performance reasons. However, if UDP is used, its performance should not degrade if higher packet loss is encountered and switching to TCP should be avoided if high transaction rates are required.

In general, it is desirable to use UDP and to avoid TCP wherever possible, because it leads to unacceptable performance degradation for higher error rates on the transmission channel. Although from a robustness point of view there is no reason to use more than $3f+1$ nodes to run a PBFT system, when it comes to unreliable communication it turns out that adding nodes is a means to improve the redundancy on the network layer.

Additionally, the use of repetition codes can also lead to significant performance improvements as UDP can be used over TCP even in situations with increased packet loss. If the channel behavior is known in advance we recommend to configure the deployment adequately to stay in the UDP regime. In the end, for our type of application a dedicated network protocol would be desirable which adaptively optimizes retransmissions and other parameters without increasing latency.

Adaptive and selective network layer. From the structure of the communication pattern it turned out that unreliable channels have different impact in different phases. A node missing a single PRE-PREPARE message could already be out of sync for the current transaction, contrary if f PREPARE messages do not arrive, it will still have enough information to proceed, the same is true for REPLY messages. This shows that especially the first broadcast from the primary is relatively more important than the rest of the messages and measures taken to increase its probability of success will have a disproportionate impact on the success of the whole transaction. It could therefore make sense to use TCP only for this phase, or, as we have done, to pro-actively repeat this message once or twice.

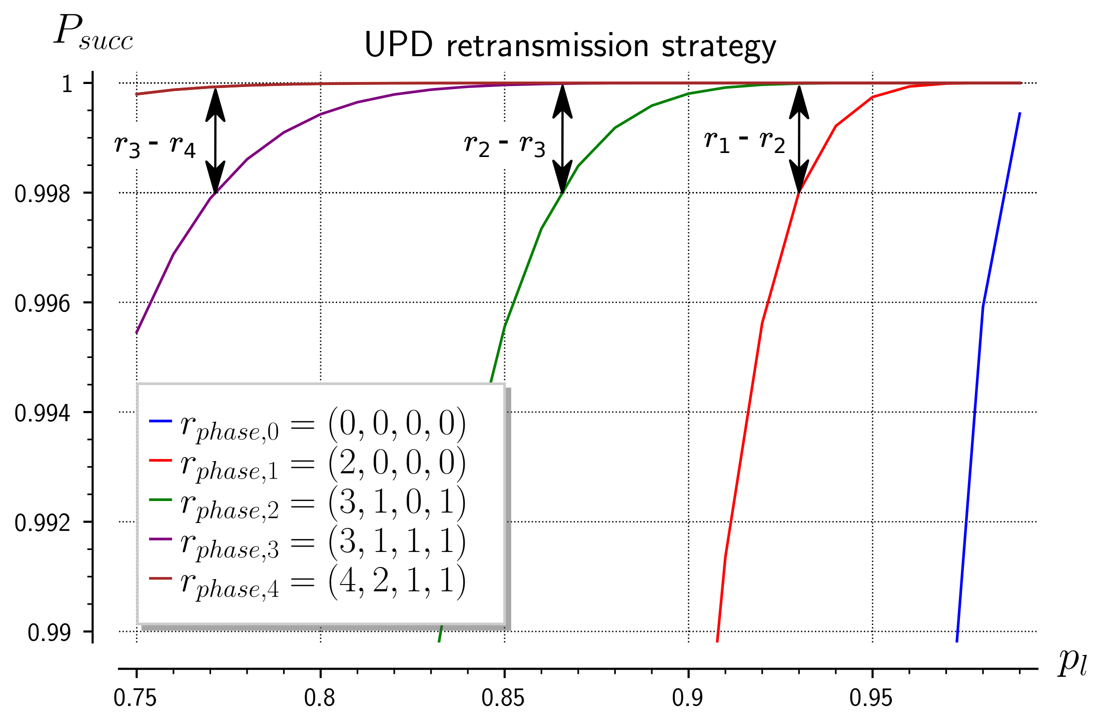


Figure 2.16: Success probabilities for different retransmission modes which are named $r_{phase,1},\ldots,r_{phase,4}$. A retransmission mode is defined by an array of four r values, one for each of the four phases of the protocol (PRE-PREPARE, PREPARE, COMMIT, REPLY). Additionally, an optimal strategy is sketched by defining switching point between modes to stay in the $P_{succ} > 0.998$ band.

However, based on our model we were able to develop good retransmission strategies which take the specific aspects of the different phases into account and achieve high transaction success probabilities while reducing the overall number of messages sent. The result of our empirical optimization is shown in Figure 2.16 and we included a switching strategy for staying above 99.8% success probability up to 25% transmission errors as an example. This approach shows how adaptive retransmission strategies can be designed to achieve low-latency systems with high transaction rates.

The presented results show the significant dependency on packet loss probability, which should ideally be continuously measured and adjusted to. Since, single BFT nodes have only a local view on the network state/connection(s) we have to signal packet loss information between the BFT nodes. Especially, in the case of UDP, where the transport layer won't track any lost packets. Therefore, we briefly mention a possible approach where BFT nodes attach their packet loss statistics to the meta-data of each message. This packet loss statistics are derived as follows if UDP is used. Each node collects in every phase of a transaction information on how many of the nodes transmitted a matching message for the current phase x of the pending request. Since each BFT nodes transitions into the next phase if $q = 2f + 1$ messages have been received but still keeps listening for further messages, we can add loss statistics in the $x+2$ phase of a pending request to account for network delay, or the BFT nodes report back to the primary if it did not receive enough messages. The primary aggregates all information and calculates the packet loss statistics and sends the updated statistics with each request to all BFT nodes as an estimate for the new transaction. Based on the new loss estimates, the nodes set their retransmission mode according a defined strategy, e.g., as depicted in Figure 2.16.

Byzantine case. If f nodes behave fully malicious, their messages are ignored by the honest nodes if they do not follow the protocol. Therefore, the best they can do to slow down transactions—and therefore slow down service time—is to delay their transmissions or remain silent. For the network layer this would mean that no redundancy is left to cope with packet loss as all $2f + 1$ honest nodes have to reach the final state for the transaction to complete and in this case packet loss would be fatal. However, by increasing the redundancy beyond $3f + 1$ nodes we reach the same regimes as presented above. In fact if $5f + 1$ nodes are used we reach in the worst case similar success probabilities, because such a system would require a $3f + 1$ quorum and leave $2f$ overall redundancy in the system, i.e. f Byzantine nodes and f honest nodes whose message do not need to arrive. However, this is only true if the adversary does not have access to the channels between honest nodes, which was the assumption we started from. Alternatively, the implementation can always fall back to TCP and therefore emulate reliable channels over unreliable ones, if the packet loss or the number of node failures is too big for UDP usage. In essence, the safety property of the system is never compromised, only performance is improved in rather optimistic scenarios.

2.4 Simpler Configuration and Hardware Acceleration

The ARCHISTAR architecture introduced provides very strong security and consistency properties through the combination of secret sharing with an efficient consensus mechanism. However, it requires active nodes which are not always available and introduce substantial cost which are not always justified, e.g., in case of low transaction rates. Therefore, we also developed a rather straight forward alternative architecture which do not need any specific code to be executed by the server, the *Archistar-Proxy*. With this approach we embrace and enhance existing storage technologies and protocols to allow for seamless integration.

2.4.1 Proxy Architecture

The basic concept of the Archistar-Proxy was to rely only on passive backend interfaces, e.g., like the Amazon Simple Storage System (S3). Therefore, all logic had to be put into the proxy, which was designed to serve as intermediary, ultimately serving end-user clients. The proxy itself acts as a S3 server and can therefore be transparently plugged into existing applications. This enables seamless integration into existing infrastructure and simple centralized management. The S3 interface is accessed as key-value store which provide functionality akin to non-hierarchical dictionaries [114]. Similar to block-based storage systems they utilize unique keys for identifying data, in contrast to block-storage they allow storage of arbitrarily sized values. They focus upon availability and horizontal scale-out, i.e., sharding, have become prominent for cloud applications. The foremost known key-value network protocol is the industry-standard *Amazon S3* protocol which is served over HTTPS.

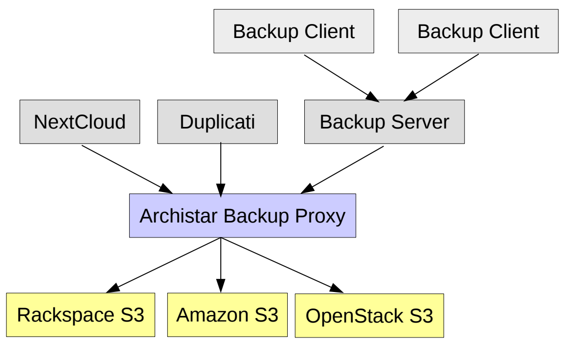
Figure 2.17a shows the proxy-based cloud storage scenario: multiple clients and servers communicate with the proxy which, in turn, communicates with multiple cloud storage providers at the bottom. A simplified version of the data-flow can be seen in Figure 2.17b. The Archistar-Proxy assumes that, the backend storage can fail arbitrarily and also suffer from weaker consistency models. Cloud outages and breaches[11,12] have shown that this a reasonable assumption and AWS was providing only eventual consistency for S3 until 2020[13].

The design of the proxy is rather straight forward if only a single proxy is writing to the cloud. Different secret-sharing methods are available to split up data—provided by an actual client—into shares and then distributed upon multiple potentially untrusted cloud storage providers. However, different research challenges emerged in the devel-

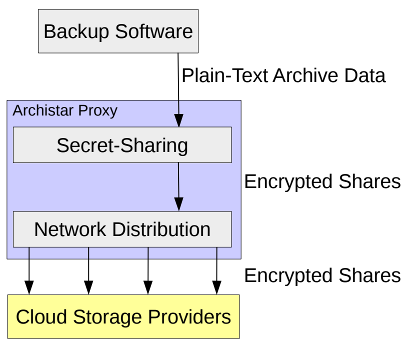
[11]Gitlab data loss: https://about.gitlab.com/2017/02/01/gitlab-dot-com-database-incident, accessed 06/12/2017.

[12]Amazon S3 outage: https://aws.amazon.com/message/41926/

[13]https://aws.amazon.com/about-aws/whats-new/2020/12/amazon-s3-now-delivers-strong-read-after-write-consistency-automatically-for-all-applications/, accessed 17/11/2022.



(a) High-level network system overview.



(b) Example data-flow from backup client.

Figure 2.17: High-Level network system overview and example data-flow from backup clients to the cloud storage servers.

opment which had to be addressed. Firstly, metadata was an important topic to deal with. The use of a user side proxy provided the opportunity to increase security for metadata compared the fully cloud-based BFT system. Secondly, S3 introduces a lot of overhead and good performance is not easy to achieve especially for storage of a large number of small objects. Thirdly, for large objects as encountered in our demonstration of health data [115], the encoding and decoding speed based on our software implementation where a limiting factor. Finally, also multi-user write support turned out to be of paramount importance and cannot be easily achieved in our setting without any direct synchronization between the proxies.

Metadata security. To limit the amount of information leaked the proxy server keeps its own metadata within its so-called index. The index is a container for file data, i.e., a file's original name, size, SHA-256 based HMAC and MD5[14] hashes, modification date, content type, used secret-sharing algorithm and other metadata. In addition, it includes information on where secret-shared parts are stored. This is needed, as we store each part under a separate identifier to hamper data analysis. Apart from that, each index can—depending on the versioning configuration—contain a reference to its predecessor. As each index includes information about all available files and directories, this allows for implicit versioning of all data. To reduce storage overhead, the number of stored versions, reaching from zero to infinity, can be configured through the backup server's configuration.

Using an index also reduces storage overhead from a robustness point of view. Because the index is encoded with robust secret-sharing and contains MAC keys and tags of all data object shares, the data shares itself can be encoded by more efficient standard secret sharing methods. An attacker gaining access to a storage location and learning all secret shares cannot corrupt data shares, because he would have to modify the share in a

[14]The MD5 hash is needed for compatibility with the Amazon S3 protocol.

way that the MAC stored in the index verifies, however, without knowing the randomly selected key to the MAC he can only guess it with negligible probability. Furthermore, the index itself is encoded with robust secret sharing and can also not be tampered with up to RSS robustness guarantees. Thus, the whole system is robust although bulk data is only encoded with standard secret sharing.

Performance for small objects. The increased security by managing the metadata in a secure index also turned out to have a performance downside, if the index is also maintained in the very same multi-cloud configuration. The index grows with the number of files it contains and if it has to be written often it caused a significant slowdown. To cope with this effect we introduced smart caching techniques and made the index updatable. The idea was inspired by modern copy-on-write file systems which use similar approaches. This approach led to a very performant system and the index caching had even more positive effects on speed and privacy. By making status queries about objects a local operation they perform significantly faster and prevent the cloud providers from learning the user queries also increasing metadata privacy. This was paramount for performance, as some clients issue a surprisingly high number of such requests before and after each actual file operation. In addition, this allows the proxy to better cope with the potential high-latency during accessing networked Amazon S3-compatible storage servers.

Encoding and decoding speed. Contrary to the effects seen with small files, in the case of large file transferred to the cloud the encoding engine turned out to be a bottleneck. For backing up images of virtual machines or when dealing with whole slide images in a medical use case we were studying [115], the software implementation reached its limit.

During our research we were already able to significantly improve performance of the *archistar-smc* cryptographic library already presented in [1] (cf. Section 2.2.7), but for high-performance applications more throughput would be desirable. Therefore, we studied possibilities to accelerate the secret sharing engine with a hardware-based approach. The results of our study are shown in Section 2.4.3.

Mulit-proxy support. To support multiple proxies operating on the same data we had to follow a very different path compared to the BFT solution. The research was targeted towards a mechanism which allowed proxies to work on the same cloud storage and data in a loosely synced way. Proxies must be able to join and leave the system dynamically and are not allowed to communicate to each other or to use an external synchronization mechanism, e.g., a locking service. In essence, the proxies have to synchronize via the available clouds only assuming very weak consistency guarantees. It is clear that total ordering cannot be achieved in such a setting but other protocols are needed maintain a consistent state. To cope with this situation, we used a conflict-free replicated data type (CRDT) [116, 117] to maintain the index of the storage system. If the index is a CRDT, each proxy can maintain its own cached version and update it independently. Inconsistencies can be merged without conflict by each proxy individually through reloading the server side changes after updates from other proxies. Thus, during operation not all proxy are guaranteed to have the same state, but it is guaranteed the they eventually converge to a common global state.

Note on secure channels. Besides the research challenges solved, we wanted to note that the system is only ITS secure if the channels between proxy and cloud are also ITS. Ideally, the channel would be secured by ITS mechanisms like quantum communication as demonstrated in [115], or some physically isolation. However, this approach is very expensive and only relevant if large amounts of very sensitive data is handled. Nevertheless, if QKD is available it can be added by an additional layer of security or by integration with state-of-the-art security mechanisms used for secure HTTP connections, i.e., extending TLS with Quantum-TLS [118] or Quantum-Quick [119] (cf. Section 4.5).

Alternatively, the use of post-quantum cryptography is recommended for the standard case, ideally in a hybrid version [120]. Currently PQ-TLS is not available from large cloud providers for S3 connectivity, but work on the transition is ongoing[15] and we expect the post-quantum secure versions of TLS to be available in the near future.

2.4.2 Availability Model

To also highlight the advantages in availability if threshold secret sharing is applied, we quickly analyze the possibility to compose arbitrary availability for the overall system. The same argument can be used to calculate the durability (protection against data loss) and optimized faster response times of the system (cf. Section 2.3).

In reliability theory an n-component system that works if and only if at least k of the n components work is called a k-out-of-n:G system. The presented storage system is exactly implementing such a structure in a multi-cloud setting which lets us directly apply some results from reliability theory in our availability analysis. In particular, the reliability $R(k,n)$ of a k-out-of-n:G system with i.i.d. components, i.e., components which are independent of each other, is equal to the probability that the number of working components is greater than or equal to k. In particular the reliability is calculated as follows.

$$R(k,n) = \sum_{i=k}^{n} \binom{n}{i} p^i q^{n-1} \tag{2.8}$$

The k-out-of-n is a generic model for adding fault tolerance to systems by increasing redundancy, which is exactly what we are doing with secret sharing in the proxy , if we leave the security aspects aside for now.

[15] AWS PQ-TLS for KMS: `https://aws.amazon.com/blogs/security/post-quantum-tls-now-supported-in-aws-kms/`, accessed November 2022.

If we compare the different settings presented in the previous section with the reliability model, we get the following results. For the case of data replication we have $k = 1$, which leads to the analogous of a parallel system in the reliability model and $R(1,n) = 1 - \prod_{i=1}^{n}(1 - p_i) = 1 - (1 - p)^n$. For the cases of perfectly secure (ITS) secret sharing and computational secret sharing the reliability parameters can flexibly be adjusted through encoding between $1 \leq k \leq n$, which leads to a non-trivial k-out-of-n system if k is selected accordingly ($k > 1$ and $k < n$). However, if the redundancy is fully removed for security reasons ($k = n$), the systems becomes a simple series system with $R(n,n) = \prod_{i=1}^{n} p_i = p^n$. Thus, from a reliability standpoint, both secret sharing variants provide the same level of reliability, although providing different levels of security and storage overhead.

Now, based on the parameters of k and n we can directly calculate the system availability for a given server availability. Alternatively, we can also use this approach to derive system parameters from a target availability to achieve. Enabling this SLA tailoring via multi-cloud configurations is very attractive, because the standard cloud storage market provides only limited flexibility in the configuration of service level agreements (SLA). In fact, serving standardized SLAs to customers is a major feature of cloud computing which helps to enable the elasticity and self-service capabilities the customers want to have.

We are solving this problem by letting the user design a storage system according to their needs and requirements as a fault-tolerant composition of different cloud offerings. In particular, the k-out-of-n paradigm is used to design systems which can theoretically provide arbitrary high levels of availability. Availability classes are typically given as number of leading nines of the availability value, i.e., a “three nines” availability means 99.9% which corresponds to a downtime of 8.76h per year or 43.8min per month. We used this type of availability classes to demonstrate the theoretical values we can reach in our system with reasonable number of storage nodes.

In Table 2.5 we show the calculated availability classes for different configurations of n and k, whereby an overall availability of 98% ($p = 0.98$) is assumed for the individual cloud storage offerings used to store the data fragments. Composing a system out of individual services gives the user much more flexibility and enables him to design his own SLA for a virtual storage service with respect to availability, confidentiality and integrity on top of existing cloud offerings. This can also help to speed up and improve the cloud migration process in general [121]. Furthermore, if the configurations would be matched against cloud services databases, the best provider offerings can be selected to also get a price optimal solution [23].

n	k											
	1	2	3	4	5	6	7	8	9	10	11	12
3	5	3	1	0	-	-	-	-	-	-	-	-
4	7	5	3	1	0	-	-	-	-	-	-	-
5	8	6	4	2	1	0	-	-	-	-	-	-
6	10	8	6	4	2	1	0	-	-	-	-	-
7	12	9	7	5	4	2	1	0	-	-	-	-
8	14	11	9	7	5	3	2	1	0	-	-	-
9	15	13	10	8	6	5	3	2	1	0	-	-
10	17	14	12	10	8	6	5	3	2	1	0	-
11	19	16	14	11	9	8	6	4	3	2	1	0
12	20	18	15	13	11	9	7	6	4	3	2	1
13	22	19	17	15	12	11	9	7	5	4	3	2
14	24	21	18	16	14	12	10	8	7	5	4	3
15	25	23	20	18	16	14	12	10	8	7	5	4

Table 2.5: Number of leading nines for system reliability $R(k,n)$ for given n and k and a individual storage node reliability of $p = 0.98$, which is a typical value taken from cloud storage provider SLA.

2.4.3 Hardware Acceleration

While various software solutions for secret sharing exist, there are only few hardware implementations known. However, a dedicated hardware implementation can substantially increase the performance and has the potential to expand its applicability to high-bandwidth low-latency settings as found in data centers or cloud environments. Therefore, we investigated the problem of hardware-based encoding and decoding for relevant secret sharing techniques in storage applications.

When a data object is secret shared, it is first split into words on which the secret sharing algorithm is applied. Although the security of secret sharing is not influenced by the word size, there is a trade-off in terms of efficiency and capabilities. Software solutions for storage applications are usually working on byte level, which lower the mathematical complexity for encoding and decoding to achieve best performance. However, if additional verifiability is required, e.g., for auditing procedures as in [6, 7], small word sizes do not provide adequate security and efficiency. Therefore, the feasibility and efficiency of hardware implementations for different bit widths were investigated and optimized. All investigations were done on a Field Programmable Gate Array (FPGA) for word-widths of 8, 16, 32, 64 and 128 bits.

A proof-of-concept of a dispersed secure storage application was developed to evaluate and benchmark the developed IP core. A Zedboard [16], containing a Xilinx Zynq-7000 AP SoC XC7Z020-CLG484, was selected as FPGA platform which is partitioned into a processing system (PS) and a programmable logic (PL). Synthesis and implementations were performed with Vivado® 2015.3.

Related work. The developed cores include both, a perfectly secure secret sharing scheme (PSS) and a computational secure scheme (CSS). For PSS we used Shamir secret sharing [100] as for our software implementation. and CSS according to [93].

There are only few hardware implementations of secret sharing algorithms known from the literature and no extensive treatment of such has been done so far. The focus in these works is different from ours and—to the best of our knowledge—there exists no speed optimized FPGA implementation for full CSS schemes. Moreover, no work deals with the analysis and evaluation of different word sizes and their trade-offs.

In [122] secret sharing is used in a network monitoring application. Only the front-end sharing part, which handles the data at Gigabit rates, was realized in hardware. A significant performance increase was gained by restricting the bit width of the x-value. However, the multiplier in the computational core was not optimized for this application and gives potential for further performance increase. The work was implemented on a network FPGA card using a Virtex-II Pro 50 FPGA. The isolated examination of the share generation reveals a throughput of 2359 Mbit/s with the usage of 1633 slices in this design. These are roughly 3266 4-input look-up-tables. The full key share units reach a throughput of 343 Mbit/s with 3687 slices and 18 instances of BRAMs.

Another implementation from [123] focuses on secure secret sharing. The target architectures are application-specific integrated circuits (ASICs), synthesized with Cadence Encounter RTL Compiler with the Nangate 45 nm Opencell library. They apply robust codes and algebraic manipulation detection to resist strong cheating attacks. Besides the size of the implementation the results are focusing on the efficiency of cheating detection and correction and the causing area-overhead. There are no performance results given in terms of throughput or any bit-width dependencies.

There are various software implementations on secret sharing. The crucial parts of software implementation are the time-consuming polynomial multiplications limiting the performance. Multiplications of small word-widths can be processed efficiently with look-up-tables, but the sizes grow exponentially with the used Galois field and respectively with the bit widths. Therefore, software implementations are only efficient for a width of up to 16 bits.

In [124] a collection of libraries supporting secret sharing was gathered. The GF-Share library[17] operates in a GF(2^8) field and was analysed in detail on an Intel i5-2500K, 3.3 GHz, 8 Gbit RAM, computer system. The achieved throughput for sharing

[16]http://zedboard.org/ (Accessed: 03.07.2017)

[17]`http://web.eecs.utk.edu/~plank/papers/CS-07-593/` (Accessed: 03.07.2017)

large files for an applied $5/3$ threshold scheme was 7.4 Mbit/s. This value is relatively small compared to Gigabit networks and small extension fields.

In [125] secret sharing schemes were analysed in terms of their performance. The focus of their work is the comparison of different schemes according to the threshold parameters n and k. There was no information about the computer system. However, with Shamir secret sharing a throughput of about 9 Mbit/s could be achieved for generating 10 shares. The Computational secret sharing achieved a throughput of about 18 Mbit/s for the same amount of shares.

To increase the throughput, efforts were made to use the Graphics Processing Unit (GPU) for better performance. In [126] it was shown that the benefit of a GPU increases with higher thresholds. A threshold of 4 achieves a throughput of 48 Mbit/s for the calculation of one share. Another implementation of secret sharing based on cellular automata on a GPU [127] reveals a speed of 40-160 Mbit/s in a $5/5$ threshold scheme.

The most comprehensive secret sharing library for storage applications was presented in [1] as part of a full multi-cloud storage application. However, the implementation is in Java and the reported performance figures are in the order of about 500 Mbit/s in a $4/3$ threshold scheme.

Architecture. The main functional block is structured into a Share Generation Unit (SGU), a Secret Reconstruction Unit (SRU) and an Advanced Encryption Standard Core (AES), which performs the Enc and Enc^{-1} function of the CSS algorithm. Moreover, a True Random Number Generator (TRNG) based on the design of Wold and Tan [128] was developed, but excluded from the core to enable different TRNGs in order to reach the individual space security trade-off. The SGU and SRU are designed to work in PSS mode as well as in the CSS mode. They are capable to share the key and the payload, selectable via a single signal. Due to the high resource occupation of the AES, only one AES is shared in between the SGU and SRU.

While the AES, SGU and SRU operate on streams, independently processing words of a certain bit width, the overall architecture is packet based for better communication on the external interfaces and a better combination of payload shares with their according key share. Figure 2.18 represents this architecture. Header signals are added to each packet to enable fragmenting and identifying packets, which are additionally passed through the CSS core.

While the AES statically works with 128 bits, the SGU and SRU are designed to work generically at a bit width of 8, 16, 32, 64 or 128 bits. The bit width of the SGU and SRU is selected before synthesis via generic parameters. The design is then adapted to certain implementation strategies for the set bit width. All operations are performed within a Galois field GF(2^n), where n corresponds to the bit width.

Share Generation Unit (SGU). The share generation unit supports two modes, PSS and information dispersal (IDS). Both require the definition of a unique polynomial and the shares are generated by evaluation at an arbitrary point x with the condition $x \neq 0$,

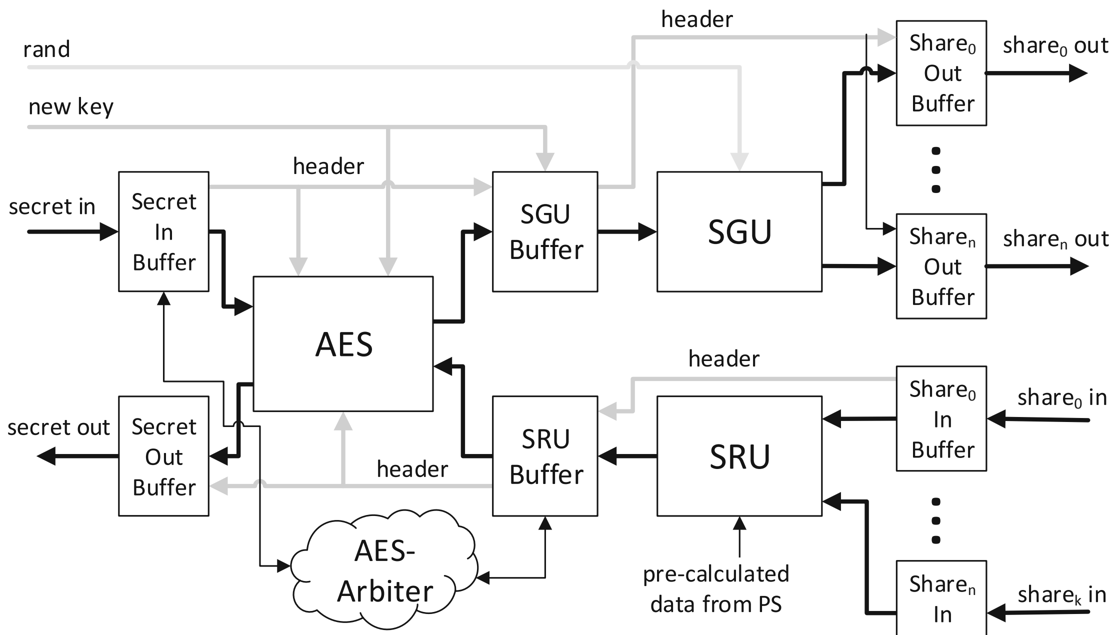


Figure 2.18: Architecture of the CSS core.

i.e. $f(x) = \sum_{i=0}^{n} c_i x^i$. In PSS mode the lowest coefficient c_0 is a secret word and the other coefficients are filled with random words, while in the IDS mode all coefficients $c_0 \ldots c_n$ are filled with secret-words. Evaluating the polynomial directly requires $\sum_{i=0}^{n}(i)$ multiplications and n additions. In the Horner scheme the coefficients are processed in an opposite way from c_n to c_0 and allow a significant reduction to n multiplications and n additions, as shown in Equation (2.9).

$$f(x) = (\ldots(c_n x + c_{n-1})x + \ldots + c_2)x + c_1)x + c_0 \tag{2.9}$$

The resulting architecture of the SGU consists of multiple Polynomial Evaluation Units (PEUs) which perform the evaluation of the Shamir polynomial in parallel. Each PEU needs $k-1$ cycles to construct the share, where the coefficients are loaded sequentially. Each PEU consists of an adder, realizable with XOR operations, a multiplier and a reduction circuit, regarding the applied Galois field. Constructing the Galois field by an irreducible polynomial of low weight, the reduction circuit can be realized by a minimum number of static-XOR connections. The polynomial multiplier shows the strongest size increase with ascending bit width. As previously mentioned in [122], a significant reduction can be realized by restricting the x-value. While this value is generic in the design, it was set to 8 bit for all results in this paper. By holding the x-value at a certain bit width, the size grow is linear, which results in a similar share generation performance for all the investigated bit widths. In Figure 2.19 implementation results are shown for a parallel generation of 10 shares.

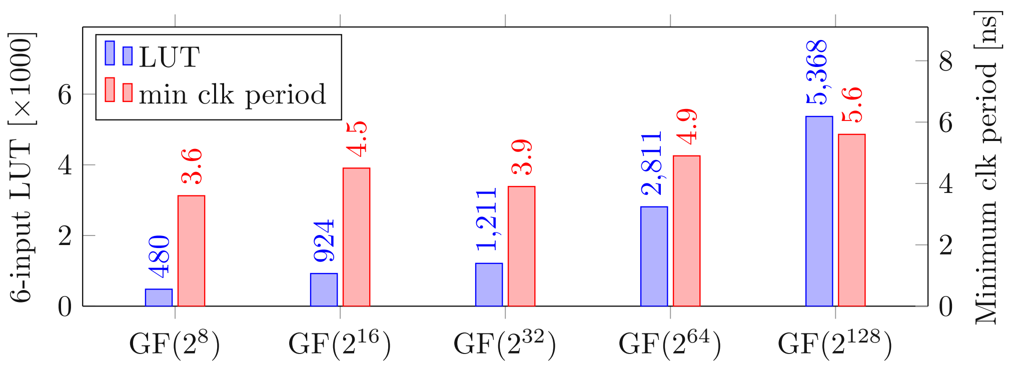


Figure 2.19: Implementation results of the SGU, where 10 shares can be generated in parallel.

Share Reconstruction Unit (SRU). The evaluation process of the Shamir polynomial is a linear equation system. Written in matrix notation it leads to Equation (2.10), with a matrix **A** containing the coefficients of the polynomial, a matrix **S** containing a set of shares for reconstruction and a matrix **X** with x-values and their powers.

$$\underbrace{\begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{pmatrix}}_{\mathbf{S}} = \underbrace{\begin{pmatrix} 1 & x_0 & x_0^2 & x_0^3 \\ 1 & x_1 & x_1^2 & x_1^3 \\ 1 & x_2 & x_2^2 & x_2^3 \\ 1 & x_3 & x_3^2 & x_3^3 \end{pmatrix}}_{\mathbf{X}} \underbrace{\begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix}}_{\mathbf{A}} \tag{2.10}$$

By solving the equation after **S**, the matrix **X** has to be inverted, which leads to Equation (2.11) for the reconstruction of a set of secrets.

$$\mathbf{A} = \mathbf{X}^{-1}\mathbf{S} \tag{2.11}$$

Because the calculation of the matrix $\mathbf{X}^{-1}$ is of high computational effort and is only required if new x-values are applied, it has high potential for HW/SW partitioning. Therefore the PS of the Zynq-7000 SoC calculates all matrix coefficients and loads these values into the SRU. In the SRU every row of the matrix X is calculated in parallel and the multiplication results are added and reduced in order to obtain one secret of the set. Since there is no feedback loop, the whole process can be pipelined. To meet a custom design trade-off of resources and timing requirements, the core allows to set the amount of pipelining stages at synthesis time by generics. While the polynomial multiplier becomes the bottleneck of the SRU, a bit-width limitation of one input, as it was done for the SGU, is not applicable here.

However, because these multipliers essentially limit the applicability of higher bit widths, further optimizations were necessary. One approach is applying Karatsuba's algorithm [129] to decrease the mathematical complexity. A multiplication of n bits is broken down to 3 multiplications of $n/2$ bits and 4 additions, which reduces the complexity from $O(n^2)$ to $O(2^{\log_2(3)})$. This method is applied recursively. In the case of 6-input LUTs a resource reduction is observable upon a width of 16 bit. Therefore, in the proposed design, Karatsuba's algorithm is applied recursively until a 16×16 bit sub-multiplication is reached.

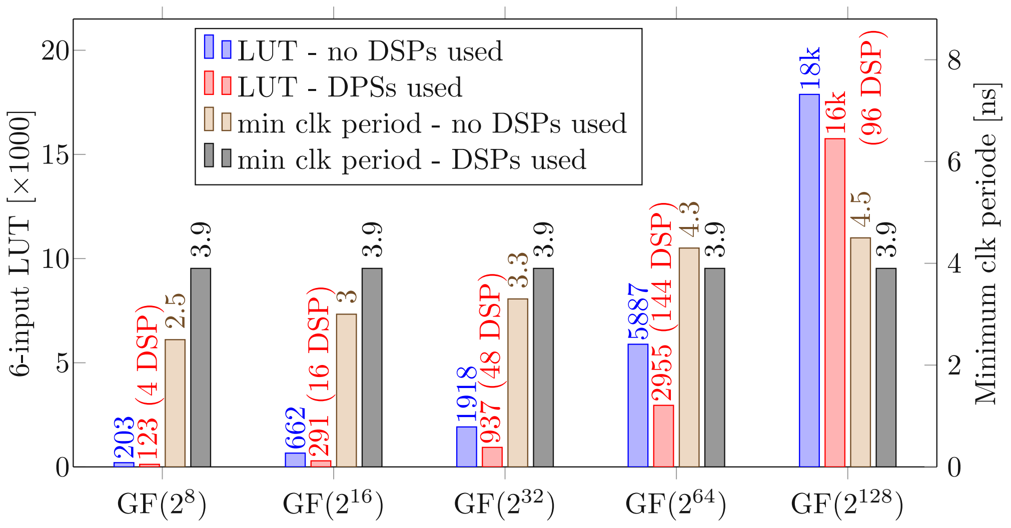


Figure 2.20: Implementation results of the CSS SRU with and without the inclusion of DSPs.

As polynomial multiplications are performed, arithmetic multipliers of FPGA-DSPs are not directly applicable. The main difference is the absence of the carry bit. Designs to include DSP multipliers were investigated, even if it is in a less efficient manner. The implementation results for the SGU, assuming a threshold of 4, are presented in Figure 2.20, comparing the involvement of DSPs in the polynomial multipliers against pure LUT utilization.

Results. In order to evaluate the complete architecture in a real setup, a complete system was developed, embedded in a network environment to manage, share and reconstruct complete files. As target platform the Zedboard was extended by an Ethernet FMC, where 3 Gigabit Ethernet connections are used. The CSS core was encapsulated in a wrapper, to correctly distribute each packet of the 3 physical connections to its according internal buffer within the CSS core. A specially developed protocol works on the top of UDP to fragment and identify all packages.

The result is a resource optimized architecture, widely parametrizable for parameters such as the bit width, x-value range, buffer size, packet size, n/k threshold scheme, inclusion of DSPs, BRAM/FF usage, pipelining stages and others.

In order to obtain the theoretical throughput, the developed cores were evaluated independently, and the results were extrapolated for a 50% utilization of the target FPGA, listed in Figure 2.21. In this theoretical estimation also the data flows and buffers were

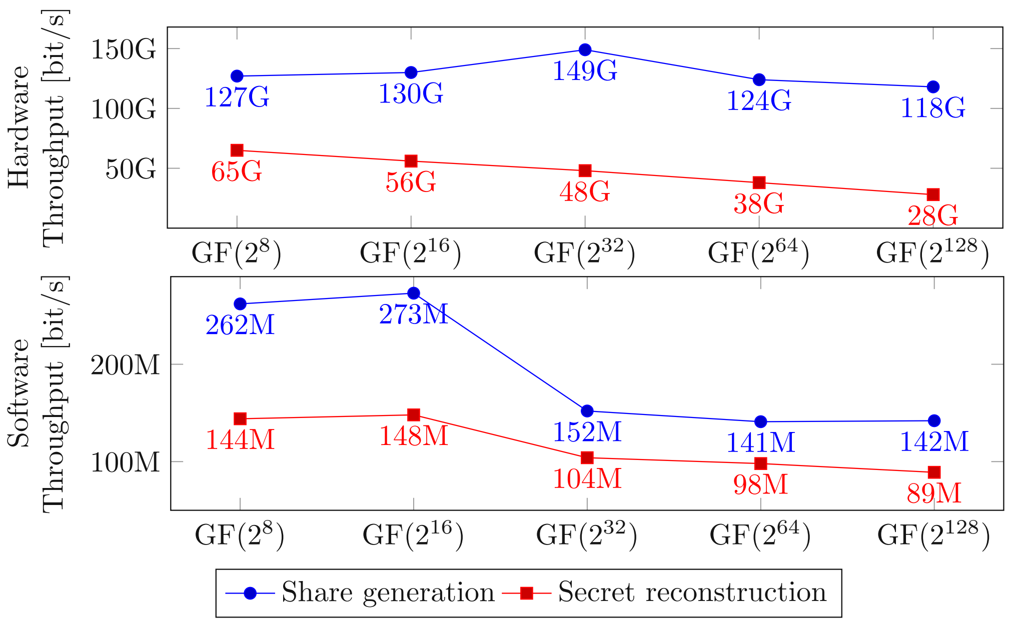


Figure 2.21: The theoretical maximal throughput of a SW and a FPGA implementation for an 8/4 threshold scheme. The throughput is measured in share-bit/s for the share generation and secret-bit/s for the secret reconstruction.

Functional Unit	LUT	(%)	FF	(%)	BRAM	DSP	P(W)
Complete Design	27745	(100)	35405	(100)	90	144	2.99
PS	—	(–)	—	(–)	—	—	1.53
3 Ethernet IP	7317	(26)	3984	(11)	4	0	0.88
TRNG	146	(0.5)	52	(0.1)	0	0	0.001
Share Switch	119	(0.7)	151	(0.7)	0	0	0.014
CSS Core	17135	(62)	20448	(58)	77.5	144	1.02
AES	2588	(9.3)	8851	(43)	36	0	0.46
SRU	2942	(11)	7467	(36)	0	144	0.3
SGU	1638	(6)	1095	(5.3)	0	0	0.033
CSS Buffers	3154	(11)	9728	(48)	39	0	0.15

Table 2.6: Hierarchical FPGA utilization results for the functional units of a complete CSS system in a 4/8 threshold scheme.

neglected. DSPs were not included in this design to allow a more universal comparison. The counterpart was a software-implementation, which also neglected all data flows and simply performs all of the required calculations, performed on an Intel i5-4590 Quad-core running at 3.3 GHz and full utilization of all cores. The share generation rate in the FPGA design could be held almost constant due to the fixed bit width of the x-point. The reconstruction shows a significant performance decrease with ascending bit width, for both the software and hardware implementations. Overall, the FPGA design shows the capability of throughputs 100 to 1000 times faster than its software counterpart. This was observed for the software applied in Figure 2.21 as well as the previously described existing software libraries summarized in Section 2.2.7.

The complete core in the prototype was built with a 64 bit architecture, as it is a good trade-off in terms of resource utilization and capabilities. It is capable of processing 6.4 Gbit/s, working with 64 bit words and an $8/4$ threshold scheme. The bottleneck is the external Ethernet connection, which is limiting the speed to 1 Gbit/s. The resource utilization, structured in functional components, is summarized in Table 2.6 as well as their power consumption.

2.5 Efficient Privacy Preserving Remote Data Checking

One problem that all outsourced storage systems have in common is that one needs to trust the storage nodes that they actually stored the data. To relieve users from this trust requirements, proofs of data possession and proofs of retrievability have been introduced [130, 131]. Such schemes allow one to efficiently verify that all data is still available without having to download everything from the system. In the best case, such an auditing mechanism is publicly verifiable, i.e., can be executed by an external auditor and not necessarily the data owner himself. However, in the case of shared data it becomes necessary that the auditing is also private, i.e., an auditor must not be able to learn anything about the stored data, even when collaborating with a subset of storage servers.

In this section we propose a highly efficient, flexible, and information-theoretically private auditing mechanism for secret sharing based storage system as ARCHISTAR. In more detail, we first formally define the syntax and security requirements of privacy preserving auditable distributed storage systems. We then present an instantiation based on additively homomorphic threshold secret sharing schemes, which we concretely instantiate with Shamir's scheme [100]. In contrast to previous auditing solutions, our scheme is completely keyless. This is achieved by exploiting non-collusion assumptions intrinsic to secret sharing based storage systems. We use the assumption that for n storage servers, at most $t \leq \frac{n-1}{2}$ are corrupted, i.e., that there is an honest majority of storage nodes. By introducing a batch version of our auditing scheme, we achieve a constant communication complexity even when auditing an arbitrarily large set of shares simultaneously. The basic structure of our auditing protocol is depicted in Figure 2.22.

Furthermore, the computational complexity on the servers' side is essentially given by computing a sum over the audited data shares, plus a minor constant overhead which is independent of the number of shares to be batch-audited. Leveraging results of Ateniese et al. [130], we further show how this computational complexity can be made sublinear in the number of shares. On the auditor's side, the computation consists of one degree-t interpolation polynomial of n values, and is therefore fully independent of the batch size. Besides its efficiency, our construction has multiple benefits compared to

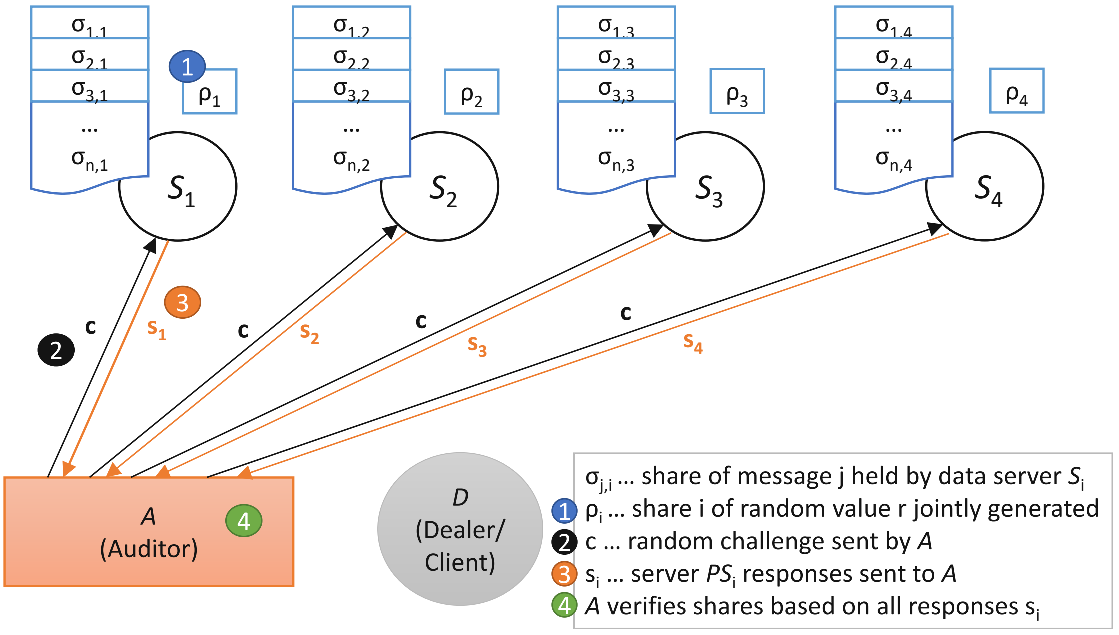

$\sigma_{1,1}$ $\sigma_{2,1}$ $\sigma_{3,1}$ … $\sigma_{n,1}$
ρ_1
S_1
$\sigma_{1,2}$ $\sigma_{2,2}$ $\sigma_{3,2}$ … $\sigma_{n,2}$
ρ_2
S_2
$\sigma_{1,3}$ $\sigma_{2,3}$ $\sigma_{3,3}$ … $\sigma_{n,3}$
ρ_3
S_3
$\sigma_{1,4}$ $\sigma_{2,4}$ $\sigma_{3,4}$ … $\sigma_{n,4}$
ρ_4
S_4
c
s_1
s_2
s_3
s_4
A (Auditor)
D (Dealer/ Client)
$\sigma_{j,i}$ … share of message j held by data server S_i
1 ρ_i … share i of random value r jointly generated
2 c … random challenge sent by A
3 s_i … server PS_i responses sent to A
4 A verifies shares based on all responses s_i

Figure 2.22: Auditing protocol overview with four servers. The protocol enables a potentially untrusted auditor to verify the consistency of secret shared messages held by different servers in one round and with fixed length message size. On a high level the auditor *PA* broadcasts a random challenge to the servers holding the shares of the data (step 2). The servers PS_i then compute a polynomial hash over their shares $\sigma_{j,i}$ with challenge c and send the result back to the auditor (step 3). However, to prevent *PA* form learning any information about the data hold by the system, the result value sent by the sever is also masked with a jointly generated random value not known to any party, i.e., $s_i = \sum_{j=1}^{\ell} c^j \sigma_{j,i} + \rho_i$. The auditor can then simply check if the results returned comprise a consistent sharing (step 4). The protocol is intended to work with linear threshold secret sharing schemes as used in our storage applications.

previous solutions. First, because of not requiring keys or preprocessing, our solution is very flexible, and offers direct support for modifications (such as additions or deletions) of data. Second, again because no pre-computation or encoding is necessary before the data is secretly shared and stored, our auditing mechanism is fully compatible with existing Shamir based storage solutions such as [1], and can easily be integrated without having to even touch the stored data. Third, our approach is also compatible with proactive security steps in storage systems. For highly sensitive data, it might be necessary to renew the shares on the storage nodes at certain discrete points in time.

2.5.1 Related Work

The notions of proofs of data possession and proofs of retrievability have been introduced simultaneously by Ateniese et al. [130] and Juels and Kaliski [131]. Since then, various publicly and privately verifiable schemes have been proposed, making different underlying assumptions, and having very different efficiencies, e.g., [132, 133, 134, 135, 136, 137].

All these protocols focus on single server storage solutions. In the following we will give a more detailed discussion about related work for multi server storage solutions, which achieve far higher performance than simply running multiple instances of the single server solutions in parallel.

Schwarz and Miller [138] propose a scheme that allows a client to verify the storage of (n,m) erasure-coded data across multiple sites even if sites collude. The scheme can also be used to verify storage on a single server and relies on a special construct, called *algebraic signatures*. The solution was intended for application in peer to peer networks and allows the creation of very large-scale verifiable distributed storage systems. The authors also propose performance optimizations to achieve checking throughputs of hundreds of Mbytes/sec. Their approach is the first to show that the distributed setting allows for very efficient solutions for remote data checking with low computational and low storage overhead and minimal communication requirements. However, the scheme only receives an informal security analysis and requires secret keys. Moreover, privacy of data is not considered and therefore third party auditing is not possible.

Other extensions to remote data checking include extending the data possession guarantee to multiple servers based on replication without encoding each replica separately. For example, MR-PDP [139] allows a client that stores t replicas of a file in a storage system to verify through a challenge-response protocol that (1) each unique replica can be produced at the time of the challenge and that (2) the storage system uses t times the storage required to store a single replica. MR-PDP extends previous work on data possession proofs for a single copy of a file in a client/server storage system. Using MR-PDP to store t replicas is computationally much more efficient than using a single-replica PDP scheme to store t separate, unrelated files (e.g., by encrypting each file separately prior to storing it). Another advantage of MR-PDP is that it can generate further replicas on demand, at little expense, when some of the existing replicas fail.

In [140] and [141] solutions based on erasure coding are introduced which also include a sound security treatment. The solution of Wang et al. [140] applies a special encoding scheme to construct a systematic Reed-Solomon (RS) code together with a challenge-response protocol which not only detects the retrievability state as a binary value, but also provides the localization of data error in an efficient way. Additionally, the scheme supports secure and efficient dynamic operations on data blocks, including: update, delete, and append. Furthermore, an extensive security and performance analysis shows that the proposed scheme is highly efficient and resilient against Byzantine failures, malicious data modification attacks, and even server colluding attacks.

In [141] integrity-protected error correcting codes (IP-ECC) were introduced together with a data checking framework called HAIL. IP-ECC are cryptographic primitives that act both as a message authentication (MAC) as well as an error-correcting code. IP-ECC achieves cross server redundancy through ECC and represents a corruption resilient MAC on the underlying data. The construction of an IP-ECC in [141] is based on a $(n,l,n-l+1)$ RS code and can be adapted for systematic RS codes (l is the number of elements in the file vector $\vec{f}$). To tag a file f, it is encoded under the RS code, and then a suitable pseudo random function (PRF) is added to the last s code symbols (for $1 \leq s \leq n$ being a system parameter), obtaining a MAC on each of those s code symbols. When reconstructing a file, a codeword is considered valid if at least one of its last s symbols are valid MACs under UMAC on the decoded file f.

In summary, only few approaches have been proposed for remote data checking in a distributed setting comprising multiple servers, and what all these schemes have in common is that they do not give formal privacy guarantees. The situation is similar for most single-server instantiations, except for a few protocols, e.g., Gritti et al. [142]. Note that simply encrypting the data before storing it into the cloud would mitigate this problem, but could not offer information-theoretic privacy, which arguably is desirable for long-term archiving of confidential data.

2.5.2 Preliminaries

Algorithms and parties are denoted by sans-serif fonts, e.g., A, B. For potentially probabilistic algorithms we write $a \xleftarrow{\$} \mathsf{A}(in)$, if a is the output of A on inputs in. An interactive protocol is denoted by $\langle \mathsf{A}(in_{\mathsf{A}}); \mathsf{B}(in_{\mathsf{B}}) \rangle$, where A and B take in_{A} and in_{B} as inputs, respectively. For a set $\mathcal{S}$, $s \xleftarrow{\$} \mathcal{S}$ denotes that s is drawn uniformly at random from $\mathcal{S}$. Throughout the paper, λ denotes the main security parameter, and $|s|$ denotes the bitlength of a string or integer s.

Shamir Secret Sharing. We next want to recap the perfectly private threshold secret sharing scheme proposed by Shamir [100]. Let therefore be n the number of participants, $t+1 \leq n$ be the threshold required for reconstruction, and $\mathbb{F}_q$ be a field with $q > n$ elements. The scheme is now based on the observation that in a field a polynomial of degree t is uniquely determined by at least $t+1$ points, while knowing the function values on at most t positions does not reveal any information about the slope on any position different from the known ones. To share a secret $s \in \mathbb{F}_q$, the dealer chooses a random polynomial $f(x)$ of degree t such that $f(0) = s$, and gives $f(i)$ to S_i for $i = 1, \ldots, n$. To reconstruct the secret from $t+1$ shares, first the polynomial $f(x)$ is reconstructed using Lagrange interpolation, and then it is evaluated at $x = 0$. In the following, the sharing and reconstruction algorithms of Shamir secret sharing are denoted by $\mathsf{ShamirShare}$ and $\mathsf{ShamirRecon}$, respectively.

2.5.3 Auditable Distributed Storage Systems

An auditable distributed storage system consists of five algorithms Setup, KGen, Store, $\mathsf{Reconstruct}$, and Verify, and an interactive protocol $\langle \mathsf{A.Audit}; \mathsf{S}_i.\mathsf{Audit} \rangle$. In such systems participants are grouped into dealers (D), auditors (A), and storage servers (S).

The publicly available system parameters are generated using Setup. Depending on the concrete instantiation this might have to be done by a trusted third party, which in practice could be realized through a joint computation of multiple parties with varied interests. Afterwards, a dealer D who wants to securely store its data on servers $\mathsf{S}_1, \ldots, \mathsf{S}_n$ computes its key pair using KGen. A message can then be stored on the servers using Store, and later be retrieved using $\mathsf{Reconstruct}$. The received parts from the different storage servers can be verified using Verify. This algorithm is typically not used in the normal mode of operation, but is only required for security definitions and proofs. Finally, the interactive auditing protocol $\langle \mathsf{A.Audit}; \{\mathsf{S}_i.\mathsf{Audit}\}_{i=1}^n \rangle$ is executed between a potentially external auditor A and the storage servers.

Besides the canonical requirements like completeness, an auditable distributed storage system should guarantee that even if a subset of the storage systems colludes, no information about the stored messages is leaked to an adversary. Furthermore, the auditor should accept an execution of the audit protocol if and only if all servers have access to valid and consistent shares of the message that was originally stored by the dealer. For practical purposes we do not require here that the auditor and the dealer are the same entity. Therefore, it is of prime importance that even a malicious auditor has no chance to learn any information about the messages stored by a dealer. This should hold even if the malicious auditor colludes with a subset of storage servers, and even in case the servers jointly deviate from the original protocol specification.

2.5.3.1 Syntax

$\mathsf{Setup}(1^\lambda, n)$**:** On input the security parameter λ in unary representation and the number of servers n, this algorithm outputs system parameters *spar*.

$\mathsf{KGen}(spar)$**:** On input the system parameters *spar*, this algorithm outputs a key pair $(sk_\mathsf{D}, pk_\mathsf{D})$ for the dealer.

$\mathsf{Store}(M, spar, sk_\mathsf{D})$**:** On input a message M, the system parameters *spar*, and the dealer's secret key sk_D, this algorithm outputs a share σ_i to be sent to S_i for $i = 1, \ldots, n$.

$\mathsf{Reconstruct}(spar, sk_\mathsf{D}, \{(i, \sigma_i)\}_{i \in I})$**:** On input the system parameters *spar*, the dealer's secret key sk_D, and a set of shares $\{(i, \sigma_i)\}_{i \in I}$, this algorithm either outputs a reconstructed message M' or $\perp$.

$\mathsf{Verify}(spar, M, sk_\mathsf{D}, \{\sigma_i\}_{i=1}^n)$**:** On input the system parameters *spar*, a message M, the dealer's secret key sk_D, and a full set of shares, this algorithm outputs `accept` or `reject`, depending on whether or not the shares are consistent with M.

```
**Experiment** $\mathsf{Privacy}_{\mathsf{Adv}}(\lambda, n, t)$
  $b \xleftarrow{\$} \{0,1\}$
  $(spar) \xleftarrow{\$} \mathsf{Setup}(1^\lambda, n)$
  $(sk_{\mathsf{D}}, pk_{\mathsf{D}}) \xleftarrow{\$} \mathsf{KGen}(spar)$
  $(M_0, M_1, state) \xleftarrow{\$} \mathsf{Adv}(spar, pk_{\mathsf{D}})$
  $(\sigma_1, \ldots, \sigma_n) \xleftarrow{\$} \mathsf{Store}(M_b, spar, sk_{\mathsf{D}})$
  $b' \xleftarrow{\$} \mathsf{Adv}^{O(\cdot)}(state)$
    where $O(\cdot) = O(spar, pk_{\mathsf{D}}, \{\sigma_i\}_{i=1}^n, \cdot)$ behaves as follows:
      On input $(\texttt{corrupt}, \mathsf{S}_i)$, it marks $\mathsf{S}_i$ as corrupt, and returns $\sigma_i$ to the adversary.
      On input $(\texttt{corrupt}, \mathsf{A})$, it marks $\mathsf{A}$ as corrupt, and returns $\varepsilon$ to the adversary.
      On input $(\texttt{audit})$, it simulates all honest parties in an execution of the auditing protocol.
    Let $t'$ be the total number of corruption requests.
  return $(b \overset{?}{=} b') \wedge (t \overset{?}{\leq} t')$
```

Figure 2.23: Privacy game

$\langle \mathsf{A}.\mathsf{Audit}(spar, pk_{\mathsf{D}}); \{\mathsf{S}_i.\mathsf{Audit}(spar, pk_{\mathsf{D}}, \sigma_i)\}_{i=1}^n \rangle$**:** This is an interactive protocol between the auditor and n storage servers. The auditor takes as input the system parameters $spar$ and the dealer's public key pk_{D}. On the other hand, each server takes as input the system parameters $spar$, the dealer's public key pk_{D}, and its share σ_i. At the end of the protocol, the auditor either outputs `accept` or `reject`, indicting whether or not the servers passed the audit.

2.5.3.2 Security Definitions

Besides the standard properties of completeness and privacy, we also introduce extractability as a way to prove data possession.

Completeness. This property captures the intuitive requirement that if all parties are honest and follow the protocol specifications, the auditor should always output `accept`. Also, Reconstruct should always return the correct original message if a sufficient amount of shares is received in input. Finally, Verify should always output `accept` if all shares are consistent with the message. Because of its limited insights, we omit here a formal definition.

Privacy. Informally, an auditing system is said to be private if no adversary can infer any information about the stored message M from a bounded number of shares and arbitrary many runs of the auditing protocol. More formally, we let the adversary choose two messages, one of which is distributed among the servers. This message is then audited upon the adversary's request. Furthermore, the adversary can corrupt at most t out of the servers and the auditor. At the end of the game, the adversary should not be able to tell which of the two messages was distributed in the first phase.

```
Experiment $\mathsf{Extractability}_{\mathsf{Adv}}(\lambda, n, t)$
    $(spar) \xleftarrow{\$} \mathsf{Setup}(1^\lambda, n)$
    $(sk_{\mathsf{D}}, pk_{\mathsf{D}}) \xleftarrow{\$} \mathsf{KGen}(spar)$
    $(M, I) \xleftarrow{\$} \mathsf{Adv}(spar, pk_{\mathsf{D}})$
        where $|I| \leq t$
    $(\sigma_1, \ldots, \sigma_n) \xleftarrow{\$} \mathsf{Store}(M, spar, sk_{\mathsf{D}})$
    $\{tr_i\}_{i=1}^m \xleftarrow{\$} \mathsf{E}_1^{\mathsf{S}_1, \ldots, \mathsf{S}_n}$
        where $\{\mathsf{S}_i\}_{i \in I}$ are controlled by $\mathsf{Adv}(\{\sigma_i\}_{i \in I})$,
        and $\mathsf{E}_1$ has rewindable black-box access to the system
    $(\sigma'_1, \ldots, \sigma'_n) \xleftarrow{\$} (\mathsf{E}_2^1(\{tr_{i\downarrow 1}\}_{i=1}^m, \ldots, \mathsf{E}_2^n(\{tr_{i\downarrow n}\}_{i=1}^m)$
        where $tr_{i\downarrow j}$ is the $i^{\text{th}}$ transcript reduced to only the messages that were exchanged with $\mathsf{S}_j$
    return $\mathsf{Verify}(spar, M, sk_{\mathsf{D}}, \{\sigma'_i\}_{i=1}^n)$
```

Figure 2.24: Extractability game

Definition 2.5.1. An n-server distributed storage system with audit is *t-private*, if for every PPT adversary Adv there exists a negligible function negl such that the following holds:

$$\left| \Pr\left[\mathsf{Privacy}_{\mathsf{Adv}}(\lambda, n, t) = 1\right] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda),$$

where $\mathsf{Privacy}_{\mathsf{Adv}}(\lambda, n, t)$ is defined in Fig. 2.23.

Note that this definition implies privacy of the storage system itself. For instance, the adversary can get up to t shares and must not be able to tell which message it was.

Extractability. Informally, this property guarantees that a distributed storage system can only pass a run of the auditing procedure if every server actually stored the share received from the dealer.

Definition 2.5.2. An n-server distr. storage system with audit is *t-extractable* with retrievability error $\rho : \mathbb{N} \to [0, 1]$, if there exists a stateless alg. $\mathsf{E} = (\mathsf{E}_1, (\mathsf{E}_2^1, \ldots, \mathsf{E}_2^n))$ such that for every adversary Adv that makes the auditor accept with probability $\varepsilon > \rho$ it holds that:

$$\Pr[\mathsf{Extractability}_{\mathsf{Adv}}(\lambda, n, t)] = 1,$$

where the expected running time of E is upper-bounded by $\mathsf{poly}(\lambda)/(\varepsilon(\lambda) - \rho(\lambda))$.

The above definition was inspired by that of distributed proofs of knowledge [143], where multiple provers need to convince a verifier that they jointly know a secret piece of information. However, our definition differs in several subtle aspects from this definition:

- First and foremost, the definition of distributed proofs of knowledge would only have guaranteed that all valid witnesses exist *in the system*. That is, it is only guaranteed that the witness (i.e., all shares) can be extracted from the set of provers (i.e., servers). In this setting, it would be allowed that malicious nodes delete their shares, as long as these shares can be recomputed from the remaining ones. However, this contradicts the intuition of our auditing procedure, as it needs to be guaranteed that every single node correctly stored its share. For this reason it is necessary to split the extractor into two phases, where the actual "extraction" may only use the interaction of the extractor with the respective server.

- Second, in (distributed) proofs of knowledge, the knowledge extractor takes as input also the value for which knowledge the witness needs to be proven. This is meaningful, as the extractor should in particular be able to play the role of the verifier. However, in our case the verifier (i.e., the auditor) does not know this value (i.e., M), and thus it would be unnatural to give M to E.
- Finally, the standard definition of (distributed) proofs of knowledge is fully independent of valid witnesses, i.e., the σ_i in our definition. However, in our definition it is crucial to start from a valid set of shares. In fact, depending on the instantiation the adversary may only be allowed to corrupt a limited number of servers (parametrized by the allowed size of I). For the other servers to be able to honestly play their parts in the protocol it is crucial for them to receive their valid shares as inputs.

2.5.4 Efficient Instantiation Based on Shamir Secret Sharing

In the following, we present an efficient auditable distributed storage system based on an arbitrary additively homomorphic threshold secret sharing scheme, which we will instantiate with Shamir's scheme [100] for clarity of presentation. By exploiting the assumptions required for a meaningful secret sharing based storage system, a completely keyless system is achieved. Furthermore, the system does not require any pre-processing of messages. Therefore, our auditing procedure can be applied directly to any data that has already been stored using Shamir's scheme, as no precomputation is required.

In Sec. 2.5.4.1, we will first sketch two trivial auditable distributed storage systems to further motivate our solution. Then, in Sec. 2.5.4.2 the basic protocol of our instantiation is presented, which is formally proven secure in Sec. 2.5.5. In Sec. 2.5.6 we show how to significantly increase the efficiency of our basic protocol using a batch technique. For intance, the use of a batching technique allows to audit at once multiple messages, potentially across multiple dealers. In Sec. 2.5.7 it is discussed how single storage servers can be blamed in case of failure.

2.5.4.1 Canonical Instantiations

In the following we briefly discuss two seemingly natural instantiations for auditable distributed storage systems, and explain their shortcomings.

In a first instantiation, one might try to first encrypt the message using an arbitrary encryption scheme, and then store replicas on n servers. To audit, each server sends a hash of the ciphertext to the auditor, who accepts if and only if all servers return the same hash value. This scheme would be very efficient, and directly support a batch version (also across different dealers) by simply hashing an entire batch of ciphertexts. Furthermore, the privacy of the scheme does not depend on a non-collusion assumption between the servers. However, extractability requires a non-collusion assumption between any 2 (not t) storage nodes: otherwise, it cannot be guaranteed that each server indeed stores a full replica of the data, as one server could just send the correct hash value to another server which stores nothing at all but simply forwards the hash value to the auditor. Moreover, the scheme is not keyless and, more importantly, is only computationally secure, and is therefore not suited for long-term archiving of confidential data.

Alternatively, to achieve information-theoretic privacy, one might first share the data using a secret sharing scheme, apply standard PORs/PDPs techniques to the single shares, and then audit every server independently. On the positive side, the extractability property in this case does not depend on a non-collusion assumption, while the privacy property clearly does. However, the scheme is not keyless, and furthermore none of the existing publicly verifiable private data checking techniques supports batch verification across multiple dealers. Furthermore, this approach is not compatible with existing storage solutions, does not (directly) support proactive steps, and also introduces a storage overhead compared to plain secret sharing.

In the following we therefore present a protocol which is information-theoretically hiding, allows for efficient batch audits across different dealers, does not require any storage overhead, supports proactivity, and is backward compatible with existing solutions. This is achieved by leveraging the non-collusion assumption of the underlying distributed storage system to also prove privacy and extractability of the resuling scheme.

2.5.4.2 An Audit Protocol for Single Messages

In this section we describe our basic protocol for auditing a distributed storage system. Storing and reconstructing messages is a direct application of Shamir's secret sharing scheme (cf. Sec. 2.5.2). The idea of the auditing mechanism is the following. The parties jointly compute a distributed random value, i.e. each party obtains a share of the randomness used to blind the shares of the message by simply adding it. If all those sums are consistent, the auditor accepts the audit, otherwise it rejects.

We again stress that the above construction is fully key-less. In particular, the dealer does not need to store any information whatsoever, which relieves him from any complex key management issues when accessing the data from different devices.

$\mathsf{Setup}(1^\lambda, n)$**:** On input the security parameter λ and the number of servers n, this algorithm outputs the system parameters:

$$spar = (q, t, n, (\mathsf{S}_1, \ldots, \mathsf{S}_n)) .$$

More precisely, these are: a prime number $q > n$ defining the field $\mathbb{F}_q$, an integer $t \leq \frac{n-1}{2}$ defining the minimum amount of shares required to reconstruct the secret, and unique identifiers of n storage servers $\mathsf{S}_1, \ldots, \mathsf{S}_n$.

$\mathsf{KGen}(spar)$**:** On input the system parameters $spar$, this algorithm outputs a key pair $(sk_\mathsf{D}, pk_\mathsf{D})$ for the dealer. The key pair is computed as follows:

$$(sk_\mathsf{D}, pk_\mathsf{D}) = (\varepsilon, \varepsilon) .$$

$\mathsf{Store}(M, spar, sk_\mathsf{D})$**:** On input a message $M \in \mathbb{F}_q$, the system parameters $spar$, and the dealer's secret key sk_D, this algorithm outputs a share σ_i to be sent to server S_i for $i = 1, \ldots, n$. The shares $\sigma_1, \ldots, \sigma_n$ are computed as follows:

$$(\sigma_1, \ldots, \sigma_n) \overset{\$}{\leftarrow} \mathsf{ShamirShare}(M, q, t, n) ,$$

where $\mathsf{ShamirShare}$ is Shamir's secret sharing algorithm (cf. Sec. 2.5.2). Note that each S_i receives the corresponding σ_i over a secure channel.

$\mathsf{Reconstruct}(spar, sk_\mathsf{D}, \{(i, \sigma_i)\}_{i \in I})$**:** On input the system parameters $spar$, the dealer's secret key sk_D, and a set of shares $\{(i, \sigma_i)\}_{i \in I})$, this algorithm outputs $\perp$ if $|I| < t+1$. Otherwise, it checks whether there exists a unique interpolation polynomial $f(x)$ of degree t such that $f(i) = \sigma_i$ for all $i \in I$. If this is the case, it outputs $M' = f(0)$; otherwise it outputs $\perp$.

$\mathsf{Verify}(spar, M, sk_\mathsf{D}, \{\sigma_i\}_{i=1}^n)$**:** On input the system parameters $spar$, a message $M \in \mathbb{F}_q$, the dealer's secret key sk_D, and a full set of shares $\{\sigma_i\}_{i=1}^n$, this algorithm outputs `accept`, if and only if:

$$M \overset{?}{=} \mathsf{Reconstruct}(spar, sk_\mathsf{D}, \{(i, \sigma_i)\}_{i=1}^n) .$$

$\langle \mathsf{A}.\mathsf{Audit}(spar, pk_\mathsf{D}); \{\mathsf{S}_i.\mathsf{Audit}(spar, pk_\mathsf{D}, \sigma_i)\}_{i=1}^n \rangle$**:** This interactive protocol consists of the following steps.

1. The servers jointly compute a distributed uniformly random value in $\mathbb{F}_q$. That is, at the end of this interaction, every server has a share ρ_i of a Shamir-shared random value r with threshold t.
2. Next, the auditor broadcasts a challenge value $c \overset{\$}{\leftarrow} \mathbb{F}_q$ to the servers.
3. Each server computes $s_i = c\sigma_i + \rho_i$, which it returns to the auditor.

4. Finally, the auditor outputs `accept` if and only if the provided s_i are all consistent, i.e., if:

$$\mathsf{Reconstruct}(spar, sk_{\mathsf{D}}, \{(i, s_i)\}_{i=1}^{n}) \neq \perp .$$

2.5.4.3 Distributed Generation of a Random Value.

In Step 1, we use a subroutine for jointly computing a Shamir shared, uniformly random value in $\mathbb{F}_q$. That is, by the end of the protocol, every S_i should have a share of a uniformly random value. Besides completeness, we require the protocol to be private. That is, even if up to t servers behave maliciously, they must not be able to learn anything about the shared randomness. In particular, they must not be able to bias the resulting value in any sense.

This building block could trivially be instantiated by letting all servers draw a fresh random secret r_i which is shared using $\mathsf{ShamirShare}(r_i, q, t, n)$. The resulting shares are sent to the corresponding servers. At the end, each server adds all its shares. The resulting shared secret is clearly uniformly random as long as there is at least one honest server in the system. More advanced protocols have been proposed in the literature, and can be found, e.g., in [144].

2.5.5 Security Proofs

In the following we provide detailed proofs that the system described above satisfies the security properties defined in Sec. 2.5.3.2.

Theorem 2.5.1. The above scheme is perfectly complete.

Proof. This property follows trivially from the completeness of Shamir's secret sharing scheme, the protocol used for computing a shared randomness, and the additivity of shares for the same threshold. □

Theorem 2.5.2. The above scheme is t-private according to Definition 2.5.1.

Proof. Clearly, if the auditor does not get corrupted, privacy follows immediately from the fact that Shamir's secret sharing scheme does not reveal any information if at most t shares are known.

If, on the other hand, the auditor gets corrupted, note that responses received from honest servers do not contain any information about their original shares (and thus about the message). This is, because by assumption the subroutine used for computing the shared randomness guaranteed that all ρ_i obtained by honest servers are uniformly random. □

Theorem 2.5.3. The above scheme is t-extractable with retrievability error $\rho = 1/q$ according to Definition 2.5.2.

Proof. Assume an adversary that can make the auditor accepting with probability more than ρ.

In the first phase, the extractor E_1 uses its rewindable black-box access to obtain two different sets of transcripts with the same shared randomness but different challenges c_1 and c_2. This can be done by rewinding the system of servers to the end of Step 1 in the auditing protocol, or to the beginning of the protocol. The precise mode of operation of E_1 is identical to the knowledge extractor in the proof that every Σ-protocol is a proof of knowledge, and can be found, e.g., in Damgård [145]. From there it in particular follows that the running time of E_1 is bounded above by $\mathsf{poly}(\lambda)/(\varepsilon(\lambda)-\rho(\lambda))$.

Let the transcripts be given by $tr_1 = (c_1,(s_{1,1},\dots,s_{1,n}))$ and $tr_2 = (c_2,(s_{2,1},\dots,s_{2,n}))$. We then have that $tr_{1\downarrow j} = (c_1, s_{1,j})$, and similarly $tr_{2\downarrow j} = (c_2, s_{2,j})$.

In the second phase of the extractor, E_2^j now outputs

$$\sigma'_j = (s_{1,j}-s_{2,j})\cdot(c_1-c_2)^{-1} \in \mathbb{F}_q\,.$$

Clearly, this computations can be done in polynomial time.

What now remains to show is that these shares are indeed consistent shares for the original message M. To see this, first note that for $i = 1,2$, it holds that $s_{i,1},\dots,s_{i,n}$ are consistent shares for R_i by Step 4. Thus, by the linearity of Shamir's secret sharing scheme, $\sigma'_1,\dots,\sigma'_n$ are consistent shares for $M' = (R_1-R_2)\cdot(c_1-c_2)^{-1}$. Now, as by construction we have that $n \geq 2t+1$, it follows that there are at least $t+1$ servers that replied honestly in both transcripts. Clearly, the corresponding σ'_j are consistent also with the original message M. This is because $s_{i,j} = c_i\sigma_{i,j} + \rho_j$ and thus $\sigma'_j = \sigma_j$. As those (at least) $t+1$ shares uniquely determine the shared message, we have that $M' = M$. □

2.5.6 Batch-Auditing of Stored Messages

The main drawback of the basic protocol specified in Sec. 2.5.4.2 is that it requires a fresh distributed random number to be computed for each message to be audited. We therefore show how to reduce these costs to a practically negligible amount if a large number of data blocks are audited at the same time. This is particularly interesting in our setting, where no pre-processing is necessary. Namely, it is even possible to simultaneously audit a large batch of message *across different dealers*. That is, the auditor can audit the storage solution of a company as a whole, and does not need to run this audit individually per employee, as different employees do not need to use different private keys in the Store phase. The only requirement is that all messages to be batch-audited have been shared for the same system parameters *spar*, i.e., using the same threshold t and the same n servers.

Assume now that each message $M_1,\dots,M_\ell$ has been distributed and stored according to Sec. 2.5.4.2, resulting in shares $\sigma_{1,i},\dots,\sigma_{\ell,i}$ on each server S_i, $i=1,\dots,n$. We then have:

$\langle \mathsf{A}.\mathsf{Audit}(spar, pk_{\mathsf{D}}); \{\mathsf{S}_i.\mathsf{Audit}(spar, pk_{\mathsf{D}}, \{\sigma_{j,i}\}_{j=1}^{\ell})\}_{i=1}^{n}\rangle$**:** The protocol works just as in the basic case, except that in Step 3 each server computes its response as:

$$s_i = \sum_{j=1}^{\ell} c^j \sigma_{j,i} + \rho_i \,.$$

That is, the response is computed as a polynomial hash of the shares stored by the server. It can now be shown that for every constant ℓ, the resulting protocol is a t-extractable system with retrievability error ℓ/q. The proof is similar to Theorem 2.5.3: first, rewinding allows one to extract sufficiently many transcripts as in the case of Σ_m-protocols [146]. The second step then essentially corresponds to solving a linear system of equations.

The communication complexity is exactly the same as for the basic protocol, and in particular is independent of the batch size ℓ. The computational complexity in the batch setting consists of the joint computation of a single distributed random value, computing a sum for each server, and calling Verify once on the auditor's side.

Increased Efficiency through Spot-Checking. The computational complexity on the server side can further be reduced by using spot checking techniques. For instance, the auditor could define a random sequence of shares that he wants to audit, and only these shares are used for computing the responses s_i. For instance, if for a $10'000$ blocks file the auditor wants to ensure that with 99% probability no server deleted more than 1% of its shares, spot-checking 460 blocks would be sufficient, cf. Ateniese et al. [130]. Applying an error-correcting code to the data before storing it on the servers could then be used to cope with this potential 1%-loss of data.

2.5.7 Identifying Malicious Servers

Both, the basic and the batch version of our auditing protocol, so far required that $n \geq 2t+1$, i.e., that the majority of nodes behaves honestly. In this setting it is possible for the auditor to detect inconsistencies among the servers. However, it is not possible to identify which shares caused the inconsistency, and thus it is also not possible to blame malicious storage servers.

This feature can be introduced whenever $n \geq 3t+1$. Using standard results from coding theory, it is then possible to compute the correct reconstructed value in Step 4, and to identify the inconsistent shares, e.g., using the Berlekamp-Welch algorithm [147].

Reverting to a Consistent State. Depending on the frequency how often a specific server fails in the audit mechanism, one might choose to either replace it, or to revert it to a consistent state again. In the following we recap a protocol based on the enrollment protocol by Nojoumian et al. in [148], which allows one to recompute lost shares for a specific server.

Assume therefore that storage server S_j has been identified to be inconsistent with the other servers, and the auditor has sent a message requesting a recomputation step for this server to the system. Receiving this message S_j returns to the last state that has been accepted by the auditor and requests the missing shares from a set of storage servers that were verified positively.

The basic idea is to recompute a missing share σ_j for storage server S_j by re-computing $f(j)$ in distributed fashion. Since the Lagrange interpolation is used for this computation only k storage servers are needed. The auditor can be in charge of selecting them. Assume it selected subset $\{\mathsf{S}_i\}_{i\in I}$. Then, the following steps are performed:

1. Each storage server S_i, for $i \in I$, computes its Lagrange interpolation constant $\gamma_i = \prod_{1\leq l\leq k, i\neq l} \frac{j-l}{i-l}$. Then, it multiplies γ_i by its share σ_i and, randomly, splits the result into k portions, such that $\gamma_i \cdot \sigma_i = \sigma_{1,i} + \ldots + \sigma_{k,i}$. Finally, it sends value $\sigma_{l,i}$ to storage server S_l using a private channel.

2. Each storage server S_i, for $i \in I$, collects all values $\sigma_{i,l}$ received from the storage servers S_l, where $l \in I$, and computes $\sigma_i = \sum_{l\in I} \sigma_{i,l}$. Then, it sends σ_i to storage server S_j through a private channel.

3. Storage server S_j computes its share σ_j by adding all received values, i.e $\sigma_j = \sum_{i\in I} \sigma_i$.

2.6 Batch Verifiability Against Malicious Clients

In the constructions for auditing so far we showed how to assure an honest dealer that his data is securely and reliably stored in a system. However, in practice a malicious dealer might distribute inconsistent shares of data to the storage servers, and then try to harm the reputation and trustworthiness of storage servers by announcing an unjustified complaint. This would be possible because in the construction so far the storage servers have no means to verify the honest behavior of the dealer.

Unfortunately, existing verifiable secret sharing schemes are too inefficient for real world usage. On a very high level, this is because the dealer has to prove to the share holders that all their shares resulted from an honest execution of the sharing algorithm. This is essentially done by letting the dealer commit to his random coins and broadcast these commitments such that the share holders can then (potentially interactively) verify that their shares are consistent with the commitments. This might be acceptable for single data blocks but not the bulk data scenarios we are aiming at. For efficiency reasons, secret sharing is usually implemented on a several-byte level, while files are often megabytes to gigabytes in size such that a file often has to be split into 2^{20} blocks or more. As a result, the computational overhead of computing, transferring, and verifying the high number of commitments renders existing VSS schemes impracticable for real-world usage.

Related work. Verifiable secret sharing has first been introduced by Chor et al. [149]. Since then, a large body of work has been performed in this field. For instance, solutions that are unconditionally private and committing have been proposed for $n \geq 2t+1$ [150, 151, 152]. If one is additionally aiming for perfect completeness one has to use $n \geq 3t+1$ [153, 154].

For efficiency reasons, many publications consider a setting which is perfectly private but only computationally committing. The probably most prominent such VSS scheme is that of Pedersen [155]. Its computationally private but perfectly committing counterpart has been proposed by Feldman [156]. An optimized generalization of Pedersen's scheme has been presented by Backes et al. [157].

All schemes mentioned consider private verifiability only. That is, only the share holders are able to verify that the distributed shares are actually correct and consistent. If computations on the share holder's side are very expensive, publicly verifiable secret sharing schemes can be used, where also external parties can verify the consistency of the distributed shares without learning anything about the shared message [158, 159, 160]. Finally, another line of research also considers VSS in an asynchronous communication model where the adversary is allowed to partially control the network, e.g., [157, 161, 162, 163].

In all works mentioned so far, verification is done on a per-message level. The only papers considering a batch version of VSS have been done by Bellare et al. [164] and Demirel et al. [6]. Starting from different motivations, both papers show related protocols that allow storage servers to efficiently batch-verify multiple Shamir-shared messages. However, both these works do not achieve the standard completeness guarantees for VSS. Namely, one typically requires that the protocol succeeds for all honest servers, if the dealer is honest and up to t servers are corrupt. However, in [164, 6] completeness is only guaranteed if the dealer and *all* servers follow the protocol specification. As a result, single servers could easily carry out denial-of-service attacks.

Our Contribution In a nutshell, we propose the first practically efficient, batch verifiable secret sharing scheme that satisfies the standard definitions for VSS.

More precisely, we first formally define what we understand by a batch VSS scheme. While this is intuitively clear, we believe that a detailed formalization is necessary, in particular in light of existing definitions of VSS which are often quite informal [157, 165, 155, 160], where it is not formally defined, e.g., at which points in time, the adversary is allowed to corrupt which parties and which information is then revealed.

In the following, we present two instantiations of our definitions based on Pedersen's VSS scheme [155] in a fully adaptive adversary model. Both our instantiations are perfectly private and complete, and computationally binding. One instantiation is perfect (i.e., the shares stored by each server have the same size as the original message) but requires that $n \geq 3t+1$. In the other instantiation, we only require the optimal bound $n \geq 2t+1$ while still achieving asymptotical perfectness in the sense that when batch-verifying m messages, the storage overhead is only $1+O(\frac{1}{m})$ which converges to 1 for large batch sizes.

Our construction. On a technical level, our protocols are closely related to Pedersen's scheme. First, the dealer shares each message M_j using a degree-t polynomial $f_j(x)$ according to Shamir's scheme. It then draws a random value b_0 and shares it using another polynomial $r(x)$. After having received the shares, the servers first interactively agree on a random challenge w which they send to the dealer. The dealer then computes commitments to a linear combination of the used polynomials, namely to $F(x) := \sum_{j=1}^{m} w^j f_j(x)$, using the coefficients of $r(x)$ as randomness. Similar to Pedersen's scheme [155], using these commitments, each server now computes a committed version of $F(i)$ and checks this against a locally computed commitment of the respective linear combination of its shares. A high level overview of the protocol is shown in Figure 2.25.

Efficiency analysis. The savings resulting from our modifications of Pedersen's scheme are significant and for the first time make verifiable secret sharing also applicable for large files and messages. Namely, using standard Pedersen, the extension factor compared to simply transferring the single Shamir shares would be about $2+c(t+1)$, where c is the extension factor of the deployed commitment scheme. Furthermore, at least $2m$ full-length exponentiation have to be computed at each node.

Straightforward approaches to reduce the overhead are the following. First, one could use batch verification techniques for modular exponentiation (such as, e.g., the small exponent test) [166]. However, this would only reduce the computational overhead, but the communication complexity would not be improved. Second, one could use a vectorized version of Pedersen's commitment scheme where one commits to many messages in a single commitment using different bases. This approach would reduce the communication complexity considerably; however, the number of modular exponentiations at each server would still be linear in m.

In this work, we propose a solution which makes both, the communication *and* the computational overhead independent of m. More precisely, our protocols have a constant communication overhead, and thus the extension factor is only $1+O(\frac{1}{m})$ (with a small hidden constant). Furthermore, the number of full-length exponentiations is only about t^2, which again is independent of the batch size. As a result, the amortized costs per message become negligible for large batch sizes, resulting in an (almost) free verifiability feature for Shamir's secret sharing scheme.

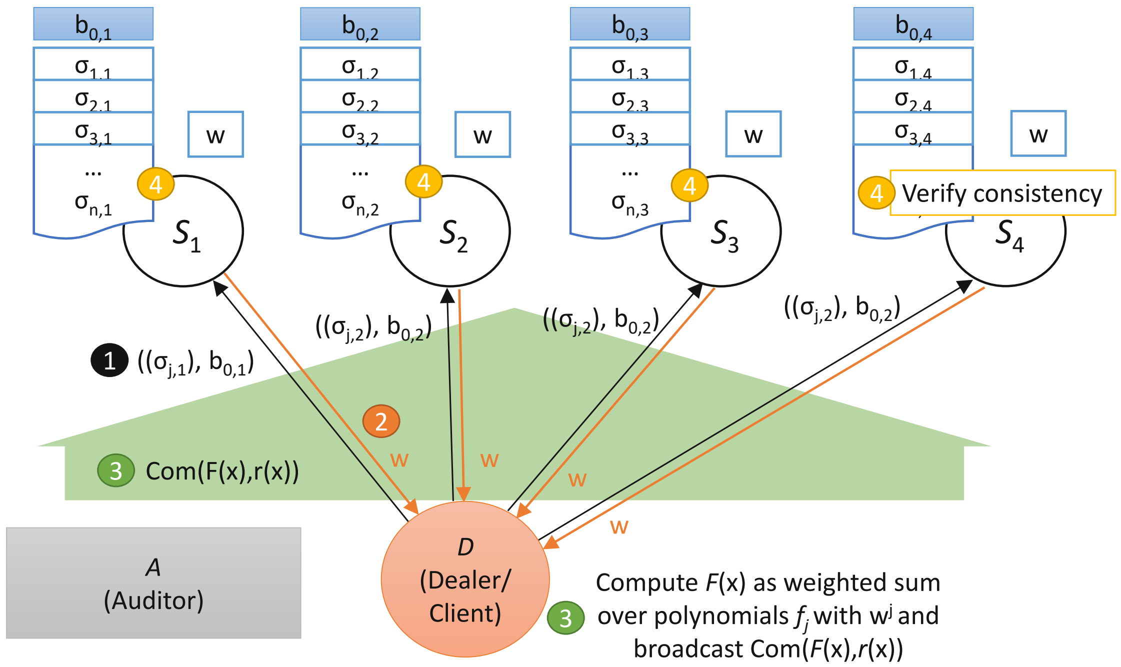


Figure 2.25: BVSS protocol overview with four servers. The protocol enables the server to efficiently check if the received share belongs to a valid sharing. In step1 the client *D* (dealer) generates the sharing of the messages along with a random sharing and distributes the shares. In step 2 a random challenge is generated jointly and sent to *D*. Based on the random challenge, *D* computes a weighted sum of the sharing polynomials and commits to it as in [155] by using the random polynomial for ITS hiding. The commitments are then broadcast to the servers (step 3). With the commitments the servers are individually able to check their respective shares by evaluation of the share generation in the exponent via the available commitments and randomness *w*.

2.6.1 Preliminaries

We first introduce some notation and then recapitulate the basic cryptographic primitives that will be used in this section.

For $n \in \mathbb{N}$, let $[n] := \{1, \ldots, n\}$ and let $\lambda \in \mathbb{N}$ be the security parameter. We denote algorithms by sans-serif letters $(\mathsf{A}, \mathsf{B}, \ldots)$ and sets by calligraphic letters $(\mathcal{R}, \mathcal{S}, \ldots)$. For a finite set $\mathcal{S}$, we denote by $s \leftarrow \mathcal{S}$ the process of sampling s uniformly from $\mathcal{S}$. For an algorithm A, let $y \leftarrow A(1^\lambda, x)$ be the process of running A, on input 1^λ and x, with access to uniformly random coins and assigning the result to y. We assume that all algorithms take 1^λ as input and we will sometimes not make this explicit in the following. To make the random coins r explicit, we write $A(1^\lambda, x; r)$. An algorithm A is probabilistic polynomial time (PPT) if its running time is polynomially bounded in λ. Furthermore, we write $\Pr[\mathcal{E} : \Omega]$ to denote the probability of event $\mathcal{E}$ over the probability space Ω. A function $f : \mathbb{N} \rightarrow \mathbb{R}$ is negligible if it vanishes faster than the inverse of any polynomial, i.e., if $\forall c \exists \lambda_0 \forall \lambda \geq \lambda_0 : |f(\lambda)| \leq 1/\lambda^c$. We write $(o_A; o_B; o_C) \leftarrow \langle A(in_A); B(in_B); C(in_C) \rangle$ for an interactive protocol between participants A, B, C. (Here, in_A, in_B, in_C and o_A, o_B, o_C denote the inputs and outputs of the different parties.)

2.6.1.1 Homomorphic Commitments Schemes

A commitment scheme with message space G consists of the following three PPT algorithms $(\mathsf{CPar}, \mathsf{Com}, \mathsf{Open})$.

Parameter sampling. $\mathsf{CPar}(1^\lambda)$, on input the security parameter 1^λ in unary, outputs a public parameter pp. (For the sake of readability we will sometimes omit making pp explicit in the inputs whenever there is no danger of confusion.)

Commit. $\mathsf{Com}(pp, M)$, on input message $M \in G$, outputs a commitment-witness pair (C, d).

Open. $\mathsf{Open}(pp, C, M, d)$, on input commitment C, message M, and witness d, outputs a verdict $b \in \{0, 1\}$.

In the following, we describe the required security properties.

First, honestly computed commitments can always be opened by an honest party:

Definition 2.6.1 (Correctness)**.** A commitment scheme is *correct*, if for all $pp \leftarrow \mathsf{CPar}(1^\lambda)$, for all $M \in G$, for all $(C, d) \leftarrow \mathsf{Com}(pp, M)$, the equation $\mathsf{Open}(pp, C, m, d) = 1$ holds.

Next, a commitment must not leak any information about the committed message:

Definition 2.6.2 (Hiding)**.** A commitment scheme is *hiding*, if for every PPT adversary A there exists a negligible function negl such that the following holds true:

$$\Pr\Big[b = b' \wedge M_0, M_1 \in G : b \leftarrow \{0,1\}, pp \leftarrow \mathsf{CPar}(1^\lambda), (M_0, M_1, \mathsf{st}) \leftarrow \mathsf{A}(pp),$$
$$(C, d) = \mathsf{Com}(pp, M_b), b' \leftarrow \mathsf{A}(\mathsf{st}, C)\Big] \le \frac{1}{2} + \mathsf{negl}(\lambda).$$

The scheme is said to be *unconditionally* or *perfectly* hiding if $\mathsf{negl} = 0$ and *computationally* hiding otherwise.

Finally, no adversary must be able to open a commitment to two different messages:

Definition 2.6.3 (Binding)**.** A commitment scheme is *binding*, if for every PPT adversary A there exists a negligible function negl such that the following holds true:

$$\Pr\Big[\mathsf{Open}(pp, C, M_1, d) = \mathsf{Open}(pp, C, M_2, d') = 1 \wedge M_1 \neq M_2 \wedge$$
$$M_1, M_2 \in G : pp \leftarrow \mathsf{CPar}(1^\lambda), (C, M_1, M_2, d, d') \leftarrow \mathsf{A}(pp)\Big] \le \mathsf{negl}(\lambda).$$

The scheme is said to be *unconditionally* or *perfectly* binding if $\mathsf{negl} = 0$, and *computationally* binding otherwise.

We will consider a special type of commitment schemes, namely *homomorphic* ones. In such schemes, the message space G as well as the co-domain are groups and computations on commitments have a direct correspondence on the contained messages:

Definition 2.6.4 (Homomorphism)**.** A commitment scheme is *homomorphic*, if for all messages $M_1, M_2 \in G$ and random coins r_1, r_2, for $M' := M_1 \cdot M_2$, for all $(C_1, d_1) := \mathsf{Com}(pp, M_1; r_1)$, $(C_2, d_2) := \mathsf{Com}(pp, M_2; r_2)$, for $C' := C_1 \cdot C_2$ and $d' := d_1 \cdot d_2$, it holds that $\mathsf{Open}(C', M', d') = 1$. (Here, $\cdot$ denotes the respective group operations.)

In the following, we will denote homomorphic commitment schemes by HC.

For concreteness, a reader might simply think of Pedersen commitments [155] as a concrete instantiation of commitment schemes. In such an instantiation, CPar would define G as a cyclic group of prime order q such that computing discrete logarithms in G is hard; further, two generators g, h of G are fixed. To commit to a message $M \in G$, Com draws $r \leftarrow \mathbb{Z}_q$ and outputs $(C, d) := (g^M h^r, r)$. Finally, Open outputs 1 if and only if $C = g^M h^d$. It is easy to see that this scheme is homomorphic, computationally binding, and perfectly hiding.

As an example of a homomorphic, computationally hiding, and perfectly binding commitment scheme, the reader may think of any homomorphic, perfectly complete, and IND-CPA secure encryption scheme.

2.6.1.2 Non-Interactive Verifiable Secret Sharing

The most prominent threshold scheme is due to Shamir [100] and is based on the observation that in a field a polynomial of degree t is uniquely determined by $t+1$ function values. It was also introduced in section 2.5.2. Based on this scheme Pedersen [155] proposed an elegant verifiable secret sharing scheme. In his construction, for a message M, the dealer chooses a random value b and Shamir shares M and b_0 using degree-t polynomials $f(x) := a_t x^t + \cdots + a_1 + M$ and $r(x) = b_t x^t + \cdots + b_0$, respectively. It then sends $(f(i), r(i))$ to server S_i. Next, the dealer computes commitments $C_0, \ldots, C_t$ to the coefficients of $f(x)$, using the coefficients of $r(x)$ as randomness, i.e., $C_j := g^{a_j} h^{b_j}$ where $a_0 = M$. It then broadcasts all $(C_j)_{j=0}^t$ to the servers. Using these commitments, every S_i then essentially compute a commitment to $f(i)$ with randomness $r(i)$ and check it against its private shares by verifying that $C_0 \cdot C_1^i \cdots C_m^{i^m} = g^{f(i)} h^{r(i)}$. If this is the case, S_i accepts, otherwise it rejects.

2.6.2 Security Model of Batch Verifiable Secret Sharing

In this section, we introduce the syntax and security requirements of batch verifiable secret sharing schemes.

2.6.2.1 Syntax

An (n,m,t)-batch-verifiable secret sharing ((n,m,t)-BVSS) scheme BVSS is a system consisting of a *dealer* D and n servers/players $\mathsf{S}_1,\ldots,\mathsf{S}_n$. Besides the number of servers, it is parametrized by the reconstruction threshold $t+1 \in \mathbb{N}, 1 \le t < n$, the number m of messages that can be batch-verified, the message space $\mathcal{M}$, and the share space $\mathcal{S}$.

A BVSS consists of a *parameter generation* algorithm SPar that generates public parameters which are accessible to all participants. Depending on the concrete instantiation, it might be necessary that the public parameters are generated by a trusted third party, or through a joint computation of the different system participants. Furthermore, the system consists of two interactive protocols (or *phases*). During an invocation of the *sharing phase*, the dealer can share m messages $M_j \in \mathcal{M}$ among the severs $\mathsf{S}_1,\ldots,\mathsf{S}_n$ in a confidential yet provably consistent way. Then, in the *reconstruction phase*, the S_i jointly reconstruct the shared messages by broadcasting their shares and calling an algorithm Rec.

It is required that all algorithms and protocols are PPT. Furthermore, we assume that each server has access to fresh and uniformly random random coins whenever necessary.

Parameter generation. $\mathsf{SPar}(1^\lambda)$, on input unary 1^λ, outputs public parameters pp. (We assume that each server has implicitly access to pp.)

Sharing phase. In the beginning of this phase, the dealer D, on input pp and messages $(M_1,\ldots,M_m) \in \mathcal{M}^m$, obtains shares $(s_{1,j},\ldots,s_{n,j}) \in \mathcal{S}^n$ of M_j, for all $j \in [m]$. Further, D distributes $(s_{i,1},\ldots,s_{i,m})$ to S_i, for all $i \in [n]$. In each round, each server can communicate with every other server privately and is eligible to broadcast data. After the last round, each server S_i outputs its shares and an auxiliary parameter $s_i = (s_{i,1},\ldots,s_{i,m},ax_i)$, for $i \in [n]$, while the dealer D outputs ε.

Reconstruction phase. In the beginning of this phase, each server S_i broadcasts s_i, for $i \in [n]$. Further, the deterministic reconstruction algorithm $\mathsf{Rec}(pp,s_1,\ldots,s_n)$ outputs messages $(M_1,\ldots,M_m) \in (\mathcal{M} \cup \{\perp\})^m$ which is also defined to be the protocol's output.

2.6.2.2 Security Requirements

In the following, we define the formal security guarantees that need to be fulfilled by a BVSS scheme. Even though these requirements are very similar to the standard security definitions for VSS, we will discuss them in detail. This is because in the literature the security of VSS schemes are often formulated in a rather informal way [157, 165, 160], which might potentially cause subtle differences in the way they are interpreted.

Corruption. From now on, we will say that an adversary is *t-valid* if and only if it corrupts at most t servers $\mathsf{S}_{i_1},\ldots,\mathsf{S}_{i_t}$. We therefore grant the adversary access to a corruption oracle $\mathsf{O}_{\mathsf{cor}}$ in all security experiments. At any point in these experiments, the adversary may ask $\mathsf{O}_{\mathsf{cor}}$ to corrupt a server of its choice, receiving its entire internal view (including all messages received, its share, etc.) as a response. A corrupted server is then under full control of A, who can, e.g., send or broadcast data on behalf of the server.

Informally, *correctness* says that for an honest dealer, it is always guaranteed that the original messages $M_1, \ldots, M_m$ can be recovered from the shares accepted by the servers — even if up to t servers behaved arbitrarily maliciously during the sharing phase.

Definition 2.6.5 (Correctness)**.** An (n,m,t)-BVSS scheme is *correct* if for every t-valid PPT adversary A there exists a negligible function negl such that the following holds true for all $M_1, \ldots, M_m \in \mathcal{M}$:

$$\Pr\Big[\mathsf{Rec}(pp, s_1, \ldots, s_n) = (M_1, \ldots, M_n) : pp \leftarrow \mathsf{SPar}(1^\lambda),$$
$$(\varepsilon; s_1; \ldots; s_n) \leftarrow \big\langle \mathsf{D}(pp, (M_j)_{j\in[m]}); \mathsf{S}_1(pp); \ldots; \mathsf{S}_n(pp)\big\rangle\Big] \geq 1 - \mathsf{negl}(\lambda)\,.$$

Note here that the output of corrupted servers can be arbitrarily chosen by the adversary. In the literature, correctness is sometimes only required if the dealer and all servers are honest, e.g., [164, 6, 167]. However, we believe this is a too weak requirement for practical purposes as it would enable any single malicious server to launch denial-of-service attacks. It is thus important that correctness also gives robustness guarantees against malicious servers.

A batch verifiable secret sharing scheme is called *private* (i.e., satisfies *privacy*) if no adversary controlling up to t servers can learn any information about the message distributed by the dealer.

Definition 2.6.6 (Privacy)**.** The privacy experiment, dubbed $\mathsf{Exp}^{\mathsf{privacy}}_{\mathsf{BVSS},\mathsf{A}}(1^\lambda)$, in which the adversary has access to $\mathsf{O}_{\mathsf{cor}}$, is defined as follows: First, fresh public parameters $pp \leftarrow \mathsf{SPar}(1^\lambda)$ are honestly generated and passed to A. Then, A chooses messages tuples $(M_{0,1}, \ldots, M_{0,m}), (M_{1,1}, \ldots, M_{1,m}) \in \mathcal{M}^m \times \mathcal{M}^m$ and sends these to the experiment. Next, the experiment tosses a coin $b \leftarrow \{0,1\}$ and executes the sharing phase on one of the message tuples, i.e., it launches the following protocol, taking the roles of the dealers and all honest servers:

$$(\varepsilon; s_1; \ldots; s_n) \leftarrow \big\langle \mathsf{D}(pp, (M_{b,j})_{j\in[m]}); \mathsf{S}_1(pp); \ldots; \mathsf{S}_n(pp)\big\rangle$$

Eventually, A outputs a guess b'; if $b = b'$ holds, then we say that A wins and the experiment outputs 1, otherwise the experiment outputs 0.

An (n,m,t)-BVSS protocol satisfies *privacy* if and only if for any t-valid PPT adversary A there exists a negligible function negl such that the following holds true:

$$\left| \Pr\left[\mathsf{Exp}^{\text{privacy}}_{\mathsf{BVSS},\mathsf{A}}(1^\lambda) = 1\right] - 1/2 \right| \leq \mathsf{negl}(\lambda)\,.$$

Finally, the *commitment* property is the distinguishing feature of verifiable secret sharing schemes. It guarantees that the dealer cannot change its mind about the distributed value after the sharing phase — even if it adaptively corrupts up to t servers. That is, if one has access to all shares, then one will always reconstruct the very same secret message M', even if up to t shares are arbitrarily and adaptively modified by the adversary.

Definition 2.6.7 (Commitment)**.** Consider the following commitment experiment, which is dubbed $\mathsf{Exp}^{\text{commit}}_{\mathsf{BVSS},\mathsf{A}}(1^\lambda)$, in which the adversary has access to $\mathsf{O}_{\mathsf{cor}}$: First, fresh public parameters $pp \leftarrow \mathsf{SPar}(1^\lambda)$ are honestly generated and passed to A. The experiment runs the sharing phase $\langle \mathsf{D}(pp,(M_j)_{j\in[m]}); \mathsf{S}_1(pp);\ldots;\mathsf{S}_n(pp)\rangle$ for $M_1,\ldots,M_m \in \mathcal{M}$ chosen by A. Let $(s_i)_{i\notin \mathcal{C}}$ be the shares of the non-corrupted parties. In the next step, the adversary is given $(s_i)_{i\notin \mathcal{C}}$ as input and outputs two full sets of shares $(s'_i)_{i\in[n]}, (s''_i)_{i\in[n]}$, such that $\#\{i \in \mathcal{C} : s_i = s'_i\} \geq n-t$ and similar for s''_i. The adversary now wins (meaning that the experiment outputs 1), if and only if $\mathsf{Rec}(s'_1,\ldots,s'_n) \neq \mathsf{Rec}(s''_1,\ldots,s''_n)$.

An (n,m,t)-BVSS protocol is now said to be *committing* if and only if for every t-valid adversary A there exists a negligible function negl such that the following holds true:

$$\Pr\left[\mathsf{Exp}^{\text{commit}}_{\mathsf{BVSS},\mathsf{A}}(1^\lambda) = 1\right] \leq \mathsf{negl}(\lambda)\,.$$

Let us explain the rationale behind this definition. Namely, the adversary can already compromise a number of servers before and during the sharing phase. Then, in the reconstruction phase, all servers broadcast their shares. The adversary is thus allowed to see all shares held by honest servers, and may adaptively corrupt further ones. (Alternatively, one could have required that the adversary must decide which servers to corrupt before the shares were revealed; however, we believe that our stronger modeling is reasonable as it guarantees security against highly adaptive adversaries that might be able, e.g., to block broadcast messages.) The requirement now is that no matter how the adversary modifies the shares of corrupted parties, the reconstruction phase will always yield the same result and, thus, the dealer is committed to this value after the sharing phase.

Another flavor of verifiable secret sharing sometimes requires that *all qualified subsets of shares held by servers that followed the sharing protocol reconstruct to the same message*, e.g., [155, 6]. This definition is weaker than ours in the sense that it only quantifies over shares of honest parties in the reconstruction algorithm and, thus, does not offer any robustness guarantees against adaptively modified shares. Formally, this means that in Def. 2.6.7, the adversary would only be allowed to delete shares in the second phase of the experiment which is just a special case of our definition. Consequently, in the optimal case of VSS schemes, where $n = 2t+1$ holds, our definition directly implies the alternative definition mentioned above.

Finally, we also want to point out the relation of our definition to the standard binding property of commitment schemes. Namely, the adversary first outputs a commitment. (In our definition, this can be thought of as the shares of honest servers.) Then, the adversary has to come up with two different messages and openings that are consistent with this commitment (in our case, the messages would be the results of Rec and the openings would be the entire sets of shares).

2.6.3 Efficient Instantiations of BVSS

In this section, we present an instantiations of a (n,m,t)-BVSS scheme and prove that it indeed satisfies the requirements from Section 2.6.2. Furthermore, we suggest an variant of our construction for the situation where minimum storage overhead is of prime importance.

2.6.3.1 Instantiations from Homomorphic Commitments

In the following, we let p be a prime or a prime power, and we define the message space of our BVSS scheme as $\mathcal{M} := \mathbb{Z}_p$. Our instantiations are based on any additively homomorphic commitment scheme $(\mathsf{CPar}, \mathsf{Com}, \mathsf{Open})$ with message space $\mathbb{Z}_p$; for concreteness, the reader may simple think of Pedersen commitments [155] in the following. Furthermore, we use an interactive randomness generation protocol $(\mathsf{RPar}, \mathsf{Rnd}_{\mathsf{BVSS}})$ as an additional building block. Informally, this protocol, executed among the n servers, has to guarantee that all honest servers receive the same uniformly random output from $\mathbb{Z}_p$, even if up to t of the servers behaved maliciously. For a formal definition and a concrete instantiation we refer to Section 2.6.3.2.

Asymptotically perfect (n,m,t)-BVSS for $n \geq 2t+1$. We now present an instantiation which works for all $n \geq 2t+1$, and is asymptotically perfect in the sense that it only has a constant additive storage overhead compared to plain Shamir secret sharing.

The algorithms and protocols of our (n,m,t)-BVSS scheme are defined as follows:

Parameter generation. $\mathsf{SPar}(1^\lambda)$ obtains $pp' \leftarrow \mathsf{CPar}(1^\lambda)$ as well as $pp'' \leftarrow \mathsf{RPar}(1^\lambda, n, t, p)$, and outputs public parameters $pp := (pp', pp'')$.

Sharing phase. The sharing phase consists of the following steps:

– The dealer D, on input pp and messages $(M_1, \ldots, M_m) \in \mathbb{Z}_p^m$, chooses $m+1$ degree-t polynomials

$$f_j(x) = a_{j,t}x^t + \cdots + a_{j,1}x + M_j \quad \text{and} \quad r(x) = b_t x^t + \cdots + b_0$$

with (uniform) coefficents $a_{j,v}, b_v, b_0 \leftarrow \mathbb{Z}_p$, for all $v \in [t]$ and all $j \in [m]$. Further, the dealer D sends shares

$$((f_j(i))_{j\in[m]}, r(i))$$

to each server S_i, for all $i \in [n]$.

- Once every server S_i received $((\hat{f}_{j,i})_{j\in[m]}, \hat{r}_i)$ from the dealer, the servers engage in an execution of the joint randomness generation protocol $\mathsf{Rnd}_{\mathsf{BVSS}}(1^\lambda, n, t, p)$. Let w_i be the output of S_i which is send to the dealer D. The dealer defines w as the majority of the received w_i.[18]
- Next, D computes

$$(C_v, d_v) := \mathsf{Com}(pp', \sum_{j=1}^{m} a_{j,v} w^j; b_i),$$

for all $v \in [t] \cup \{0\}$, where $a_{j,0} := M_j$, and broadcasts $(C_v)_{v\in[t]\cup\{0\}}$. (After the broadcast, D outputs ε.)
- Upon having received $\hat{C}_0, \dots, \hat{C}_t$, each server S_i verifies the consistency of its shares by validating that

$$\hat{C}_0 \cdot \hat{C}_1^i \cdot \hat{C}_2^{i^2} \cdots \hat{C}_t^{i^t} \stackrel{?}{=} \mathsf{Com}(pp', \sum_{j=1}^{m} \hat{f}_{j,i} w^j; \hat{r}_i) \tag{2.12}$$

holds. If the equation does not hold, then S_i terminates the protocol; otherwise, it outputs

$$s_i := ((\hat{f}_{j,i})_{j\in[m]}, \hat{r}_i), (w, \hat{C}_0, \dots, \hat{C}_t)). \tag{2.13}$$

Reconstruction phase. The reconstruction procedure is only one-round. Concretely, each server first broadcasts its entire share to all other servers. Now, on input $(s_1, \dots, s_n)$, Rec first verifies each s_i by verifying whether the contained values satisfy (2.12). If less than $t+1$ s_i are valid, Rec outputs $\perp$. Otherwise, for each $j \in [m]$, it takes the respective parts of the valid shares and computes the degree-t interpolation polynomial f'_j, and outputs $M'_j := f'_j(0)$.

Computational complexity. Looking at our protocol specification, it can be seen that the computational overhead on the server side is independent of the batch size, except for the computation of the sum in (2.12). However, in practice, these costs are negligible, in particular compared to the evaluation of m equations of the same form in the direct generalization of existing VSS schemes.

[18]Note that after this step, by definition of the building block, it is guaranteed that all honest S_i and the dealer D hold the same uniformly random value which we denote by w in the following.

Perfect (n,m,t)-BVSS for $n \geq 3t+1$. Looking at (2.13), one can see that each server has to store its shares for all M_j plus two elements in $\mathbb{Z}_p$ (i.e., $\hat{r}_i$ and w) as well as $t+1$ commitments. This storage overhead is fully independent of the batch size and is practically negligible already for moderately large values of m. In particular, our scheme is asymptotically perfect in the batch size m.

However, if one is aiming for a perfect BVSS scheme, i.e., a scheme where the share per server has exactly the same size as the original messages, the following modification of our protocol could be applied: instead of storing the s_i as in (2.13), each server only stores $(\hat{f}_{j,i})_{j\in[m]}$. Then, in the reconstruction phase, each server would not verify the shares of other servers by re-evaluating (2.12), but would find the correct interpolation polynomials, e.g., by applying the error-decoding technique of Welch and Berlekamp [147]. Theorem 2.6.2 proving the security of our scheme would still literally apply to this modified version, with the only drawback that we would have to set $n \geq 3t+1$ instead of $n \geq 2t+1$. We believe that for most practical applications asymptotic perfectness is sufficient and the higher robustness guarantees of the original scheme are preferable.

2.6.3.2 Joint Randomness Generation

In our BVSS protocol, we require that n servers and the dealer jointly agree on a uniformly random value $w \in \mathbb{Z}_p$ in the sharing phase.

Such a joint randomness generation scheme for $\mathbb{Z}_p$ consists of the following two algorithms and protocols:

Parameter generation. $\mathsf{RPar}(1^\lambda, n, t, p)$ outputs public parameters pp.

Randomness generation. This is an interactive protocol executed between n servers, at the end of which each server receives an output w_i:

$$(w_1; \dots; w_n) \leftarrow \langle \mathsf{S}_1(pp); \dots; \mathsf{S}_n(pp) \rangle .$$

A joint randomness generation protocol has to satisfy the following property:

Definition 2.6.8 (Uniform output)**.** A joint randomness generation scheme has the *uniform output* property if for all $\lambda, p \in \mathbb{N}$, all $n \geq t+1$, all $pp \leftarrow \mathsf{RPar}(1^\lambda, n, t, p)$, all t-valid PPT adversaries controlling at most t servers, and all

$$(w_1; \dots; w_n) \leftarrow \langle \mathsf{S}_1(pp); \dots; \mathsf{S}_n(pp) \rangle ,$$

it holds that $w_i = w_j$ for all honest servers $\mathsf{S}_i, \mathsf{S}_j$, and that this value is uniformly distributed in $\mathbb{Z}_p$.

For our instantiation, let $\mathsf{C} = (\mathsf{CPar}, \mathsf{Com}, \mathsf{Open})$ be an arbitrary (potentially non-homomorphic) commitment scheme with message space $\mathbb{Z}_p$. $\mathsf{Rnd}_{\mathsf{BVSS}}$ together with the parameter generation RPar is defined as an interactive protocol between the servers $\mathsf{S}_1, \dots, \mathsf{S}_n$ as follows:

Parameter generation. $\mathsf{RPar}(1^\lambda, n, t, p)$, on input 1^λ and integers $n, t, p \in \mathbb{N}$, with $n \geq t+1$, outputs $pp := (pp', n, t, p)$, for $pp' \leftarrow \mathsf{CPar}(1^\lambda)$.

Randomness generation. Each server S_i, on input pp, samples $w_i \leftarrow \mathbb{Z}_p$ and broadcasts a commitment C_i in the first round, for $(C_i, d_i) \leftarrow \mathsf{Com}(pp', w_i)$, for all $i \in [n]$. In the second round, each server S_i broadcasts the corresponding opening (w_i, d_i), for all $i \in [n]$. Let $w'_1, \dots, w'_l$ be the received consistent openings, for $l \in [n]$. Each server S_i outputs $w_i := w'_1 + \dots + w'_l \bmod p$.

Lemma 2.6.1. The protocol above has the uniform output property according to Definition 2.6.8.

Proof. As we have $n \geq t+1$, after the second round, each server received at least 1 valid commitment and opening from an honest server, guaranteeing the uniform distribution of the output. This is because any t-valid adversary could only suppress valid messages of at most t servers. To see the consistency for honest servers, simply note that by assumption to the broadcast channel, every server received the same commitments and openings from malicious servers. □

2.6.3.3 Theorem and Security Proofs

Theorem 2.6.2. If $\mathsf{Rnd}_{\mathsf{BVSS}}$ is a joint randomness generation protocol in the sense of Section 2.6.3.2 and $\mathsf{HC} = (\mathsf{CPar}, \mathsf{Com}, \mathsf{Open})$ is a homomorphic commitment scheme, then the protocol BVSS as described in Section 2.6.3.1 is an (n, m, t)-BVSS protocol, for $n \geq 2t+1$ and for all $m \in \mathbb{N}$ (where all values are polynomial in λ).

Proof. We need to proof the correctness, privacy, and commitment properties of BVSS.

Correctness. Intuitively, we have to show that even when any t-valid PPT adversary controls up to t servers during the execution of the protocol, Rec outputs exactly the messages in the reconstruction phase which were given to the (non-corrupted) dealer D as input in the sharing phase.

More formally, for all $\lambda \in \mathbb{N}$, for all $pp' \leftarrow \mathsf{CPar}(1^\lambda)$, for all non-corrupted D and for all t-valid PPT adversaries, for all messages $M_1, \dots, M_m \in \mathcal{M}$, for all D-chosen functions $((f_j)_{j\in[m]}, r)$, for all $(w_1, \dots, w_n) \leftarrow \mathsf{Rnd}_{\mathsf{BVSS}}(1^\lambda, n, t, p)$, for the uniform majority value w of $(w_i)_{i\in[n]}$, for all commitments $((C_i, d_i) := \mathsf{Com}(pp', a_{1,i}w + \dots + a_{m,i}w^m; b_i))_{i\in[t]\cup\{0\}}$, we have that Equation (2.12) holds (i.e., all commitments are valid), thus, at least $t+1$ servers from $(\mathsf{S}_i)_{i\in[n]}$ output consistent shares of the correct form

$$s_i := (((f_j(i))_{j\in[m]}, r(i)), (w, C_0, \dots, C_t))$$

at the end of the sharing phase, and Rec outputs the messages $(M_1,\dots,M_m)$ in the reconstruction phase. (Note that since we have $n \geq 2t+1$ and we only deal with t-valid PPT adversaries, we have a honest majority of non-corrupted servers and the dealer will obtain the same value w from the majority vote as all honest servers.)

The correctness of Equation (2.12) follows directly from the homomorphism property of HC. (Note that D is honest.) Hence, at least $t+1$ (honest) servers broadcast their consistent shares in the beginning of the reconstruction phase (which potentially would suffice to reconstruct the messages). However, we need to show that no t-valid PPT adversary can provide at least one different opening for the D-provided commitments from the sharing phase (i.e., a consistent opening that that does not, however, correspond to the D-chosen functions f_j, r and value w, for some $j \in [m]$, and which will be valid in the sense of Equation (2.12). Concretely, we show that no such t-valid PPT adversary can exist assuming the binding property of HC. Concretely, assume that there exist a t-valid PPT adversary A that is able to provide at least one $\mathbb{Z}_p$-value

$$m' \neq f_{v,i'},$$

for some $v \in [m]$ and for some corrupted server $\mathsf{S}_{i'}$, that is consistent with a D-provided commitment from the sharing phase; i.e., concretely, for

$$(C',d') = \mathsf{Com}(pp', \sum_{j=1, j\neq v}^{m} f_{j,i'}w^j + f_{v,i'}w^v; \hat{r}'_i),$$

$$(C_{i'},d_{i'}) = \mathsf{Com}(pp', \sum_{j=1}^{m} f_{j,i'}; \hat{r}'_i)$$

we have that $C' = C_{i'}$. Then, A can be used to break the binding property of HC by providing

$$(C'_{i'}, \sum_{j=1, j\neq v}^{m} f_{j,i'}w^j + f_{v,i'}w^v, \sum_{j=1}^{m} f_{j,i'}, d', d_{i'})$$

in a binding experiment with HC. It follows, that no t-valid PPT adversary can provide inconsistent shares (in the sense of D-chosen functions $(f_j)_{j\in[m]}$ and r) for a given sharing-phase commitment and, hence, no malicious server $\mathsf{S}_{i'}$ is able to provide different, but consistent shares that reconstruct to different messages using Rec.

Thus, since at least $t+1$ honest servers contribute in the beginning of the reconstruction phase and no malicious server is able to provide different consistent shares in the sense of above, Rec outputs the messages $M_1,\dots M_m$ (since at least $t+1$ shares are valid in the sense of Equation (2.12) and the binding property of HC holds).

Privacy. Privacy against any t-valid PPT adversary (that does not corrupt D) follows directly from the (computational) hiding property of HC and from the degree-t polynomial properties. Note that the D-chosen polynomial $\mathbb{Z}_p$-exponents $(b_i)_{i\in[t]\cup\{0\}}$ and the bit b are uniformly and independently distributed. Since we deal only with t-valid PPT adversaries, the adversary only sees at most t function evaluations $r(i)$, for some $i \in [n]$; hence, it cannot reconstruct r. This follows from the polynomial properties. Note that the adversary chooses the message tuples in the t-privacy experiment and, thus, the constant term of the polynomials $(f_j)_{j\in[m]}$. However, the adversary has no information about b and, hence, does not know which messages were selected as the constant terms in $(f_j)_{j\in[m]}$. Further, the adversary is able to corrupt up to t servers and, thus, can reconstruct all possible $(f_j)_{j\in[m]}$. However, it cannot compute the corresponding commitments on these polynomials. This follows from the (computational) hiding property of HC, i.e., all t broadcasted commitments at least computationally hide the values $(f_j(0))_{j\in[m]}$ (in particular, a linear combination thereof) and also the coefficients from r. Hence, any t-valid PPT adversary is only able to win the privacy experiment with at most negligible probability larger than $1/2$.

Commitment. The commitment property holds due to the binding property of HC. Note that honest servers stop the sharing phase with D if the equation (2.12) does not hold. Hence, the adversary (which controls the dealer D) must send consistent commitments in the sharing phase. Further, since at least $t+1$ non-corrupted servers output its shares at the end of the sharing phase, the output of Rec is uniquely determined except with negligible probability. (This follows from the correctness property of BVSS.) Further, since no t-valid PPT adversary is able to break the binding property of HC with non-negligible probability, it cannot provide altered shares in the beginning of the reconstruction phase that reconstructs to a different output of Rec compared to the (implicit) Rec-value in the end of the sharing phase. (If such an adversary would exist, then we can use it to break the binding property of HC; see the proof of the correctness property above for a more detailed discussion.) Hence, any t-valid PPT adversary is only able to win the commitment experiment with at most negligible probability.

$\square$

2.7 Discussion

In this chapter we presented the research results achieved during development of a distributed storage system with long-term security. We presented a new architecture called ARCHISTAR for secure multi-cloud storage by integrating secret sharing with a protocol for Byzantine fault tolerance. The architecture enables modern cloud-based data sharing in dynamic groups but still provides information-theoretic security for the data held by providers. To evaluate the approach we developed a proof-of-concept implementation and did intensive benchmarking. However, the evaluation revealed various shortcomings of the existing approach and spawned consecutive research which was intended to close the identified gaps for practical application.

Especially packet loss and latency had a significant impact on the performance of the PBFT system for used transport protocols. Therefore, we provided an analytical framework and validated the obtained analytical model by simulations. We further explored the design space available in PBFT deployments to optimize performance. The results have also been compared to a real implementation. The gained insights are highly valuable and help to tune performance of concrete deployments.

For the case where server cannot run computing tasks, i.e., in the case of pure S3 cloud storage, we also developed a simplified proxy configuration which relies on loosely synchronized proxies only using passive cloud storage with weak consistency guarantees (e.g. eventual consistency). However, the proxy allows for high transaction performance and can also be used in local deployments, e.g., within a cloud data center as shown in [168]. To explore the limits of high-throughput encoding and decoding for secret sharing we researched the first hardware-based computational secret sharing core for FPGA, which integrates information-theoretically secure secret sharing and information dispersal as dedicated components along with AES encryption. The proposed core is widely parametrizable and demonstrated a throughput orders of magnitude higher than software solutions.

Besides performant operation of a secure storage system, it is also important to have efficient means to check the integrity of secret shared data held in the system. For multi-cloud storage efficient ways to continuously check the consistency turned out extremely important. However, because secret shared data differs in each storage location a new way for efficient auditing was needed. We therefore defined and instantiated (privacy preserving) auditable distributed storage systems. Our instantiation is based on any homomorphic secret sharing scheme, is information-theoretically private, and supports arbitrary modifications on the stored data, including proactive re-sharing steps. To the best of our knowledge, our system is the first dedicated *distributed* storage system with formal privacy guarantees in case of an external audit, even if the auditor colludes with a subset of the storage nodes. We further showed various extensions, e.g., for identifying malicious servers or to support batch audits. In the latter case, the communication complexity as well as the computation complexity on the auditor side are independent of the batch size, while the single servers simply have to compute sums over their data shares. In particular, our system does not rely on computationally expensive cryptographic primitives. Finally, we presented an efficient protocol which can be used as a replacement for verifiable secret sharing schemes under certain rationality assumptions.

Finally, we extended the basic idea of the auditing mechanism to achieve batch verifiable secret sharing in general. Sharing large batches of data in a practically efficient way can be a valuable tool in many applications and is of paramount importance when building secure distributed storage systems. If typical block sizes in the order of bytes are considered, even the most efficient VSS schemes would not lead to practical storage solutions when applied in a naïve way. This is due to the overhead they introduce on the communication channel as well as computation-wise. The computations necessary on both sides, the dealer and the different storage nodes, immediately render the naïve approach impractical. In fact, this is the major reason why verifiability is not supported by existing dispersed storage solutions. The proposed solutions are extremely efficient and introduce little overhead in communication and storage compared to plain secret sharing when larger batches are considered, i.e., it is possible to add verifiability almost for free. We presented two instantiations which are closely related to Pedersen's VSS scheme but introduce a new interaction round between dealer and servers to make the batch operation possible. Our main instantiation is designed for maximal robustness and can cope with up to $t \leq \lfloor \frac{n-1}{2} \rfloor$ malicious players. It has constant storage overhead independently of the batch size, therefore it is asymptotically perfect when amortized over all blocks. Alternatively, we proposed a variant that does not introduce any storage overhead at all compared to plain secret sharing, i.e., it is ideal in that sense, but is only robust up to $t \leq \lfloor \frac{n-1}{3} \rfloor$ corrupt system nodes. For both protocols the computational requirements are also very limited. In fact, the computational overhead for verifying a batch is the same as verifying a single block in Pedersen's VSS protocol plus a joint randomness generation.

In summary, many interesting aspects for multi-cloud storage systems with long-term security have been researched and new results were achieved which make the technology practically relevant.

Chapter 3

Privacy Preserving Data Processing

3.1 Introduction

Multiparty computation (MPC) is a technology for computing on encrypted data in a distributed setting, i.e., with multiple nodes holding only secure fragments of input data not learning anything from them. The concept has been invented more than 30 years ago and a huge amount of research went into the topic over the last 3 decades. For a long time, it was considered only theoretical, but progress in recent years led to many interesting applications which can be realized with practical efficiency given a suitable deployment. In principle, MPC can be used to decentralize systems where typically a central trusted authority is needed to execute a function on behalf of the users. With MPC the function is evaluated jointly between multiple parties such that the correctness of the output is guaranteed and the privacy of the inputs of the individual parties is preserved, only the output of the computation is learned. Furthermore, information-theoretically secure MPC exist which makes it the ideal method for our ITS toolbox on long-term secure data protection.

We quickly present the generic model of MPC as introduced in [75], which was also edited by the author of this thesis. In a generic MPC system there exist different roles which have to be present in order to qualify as such. Input parties are holding inputs for the secure computation which must be encoded and are then sent to the compute parties. The compute parties run the actual multiparty protocol, which is executed among them as they jointly compute the intended function on the encoded inputs. The function to be computed is not kept secret and is defined according to the use case. The function is composed of basic operations available to the MPC protocol and typically composed of simple gates from a Boolean or arithmetic circuit, depending on the encoding and protocols used. After the computation the result is held by the compute parties in encoded form and then sent to the result parties, which can reconstruct the result of the computation. The main security properties as already mentioned are correctness and input privacy and it is the latter which guarantees the confidentiality of the data. Depending on the protocol, the security parameters could hold against different kind of adversaries as described in section 1.6. Additionally, some optional security guarantees are possible, e.g., fairness, guaranteed output delivery or covert security. Fairness means, that malicious parties only receive their output if also the honest parties do so. With guaranteed output delivery the honest parties always receive their output. Contrary, in a covert security model, the protocol aborts in case of error and allows for cheater detection.

In our work—which was also published in [8, 9, 10]—we focus on MPC with information-theoretic security. Protocols from this category work on the basis of secret sharing, which is why they optimally complement our work on secure storage. Therefore, we consider MPC as a processing extension for the multi-cloud storage system if only views of data should be shared between different stakeholders, e.g., as demonstrated in [169], or collaboration on joint data sets is required. Additionally, as with secure storage, for the developed MPC extensions we also assure the same stringent ITS privacy requirements as for the core ITS-MPC protocols, i.e., support for public verifiability is based on protocols with perfect zero-knowledge leveraging perfectly hiding commitments.

Our first research activity was to study interfacing with data stored in secure cloud storage systems, especially data encoded with computational secure secret sharing. We thus explored and compared the throughput achievable by obliviously executing symmetric data encryption and decryption algorithms in Section 3.2. This study was also important to get a good understanding of the different behavior of software frameworks available. In a second step, we increased the scope of our research and extended the concept to the idea of collaborative data spaces and privacy preserving market platforms. Our first research results showed that MPC does not scale in practice and setups beyond three to four nodes are not feasible for more complex functions to compute. Therefore, we studied the concept of publicly verifiable MPC in Section 3.3. Finally, we tried to push the boundaries for the developed data markets and studied the feasibility of doing a full-fledged optimization. The results are show in Section 3.4.

3.2 Performance Comparison of two MPC Frameworks

To better understand the potential of MPC in real life application scenarios we analyze the problem of interfacing with encrypted storage as a generic application. Being able to execute symmetric ciphers within MPC systems has many appealing use cases. However, the performance has to be reasonably good to support privacy preserving data mining on a larger data set. This study should also give indication about the problems arising when porting algorithms to MPC in general and on the portability of results.

The research is motivated by recent progress in MPC research reaching practical performance. However, many of the published prototyping results in research papers have been done under ideal settings typically only considering a very limited scope. Additionally, many of the open-source frameworks used to generate the results are not very mature and their behavior differ significantly from the algorithm in question. First tests also revealed the difficulty in the implementation of common algorithms in a predictable and portable way. We found it challenging to achieve good performance for algorithms without a deeper understanding for specific MPC protocols and manual optimization was in almost all cases necessary.

We are not aware of any such empirical study which tries to identify important parameters in the usage of MPC technology in a systematic and platform independent way. We compare similar MPC settings and protocols as close as we could get and therefore selected two of the most actively developed frameworks which run the same protocols under the hood and support the same adversary model. For our study we are working the in the honest-majority setting with information-theoretical security and only semi-honest adversaries. This is the most basic setting which have all major ingredients to build more complex scenarios. Computational secure protocols for dishonest majorities are out of scope for this work. It should also be noted, that we also do not intend to compare cipher of equal security strength. We are aware, that stream cipher like Trivium do not provide the same security level as AES, however, our goal is to compare how the two selected MPC frameworks can deal with the different structures of the various algorithms and what has to be considered in the porting of algorithms to MPC systems.

This study is important for practical application of MPC technologies. The question about portability of MPC applications is essential to get independence from specific frameworks, especially if provisioning of MPC-as-a-service is envisioned. Furthermore, it is important to understand the implications of the usage of certain implementations to later combine them into hybrid systems which leverage the best of the different worlds.

Contribution. In particular, this section has the following contributions. We implemented, analyzed and benchmarked four different ciphers in two MPC frameworks. We found that estimating the performance of algorithms for MPC systems in advance is hard and real performance deviated from the expected in many cases. Our work also shows a performance comparison for networks with higher latency or loss, where we identified some unexpected behavior which suggests that there is room for improvement. In our study Trivium turned out to be most versatile cipher, i.e., it supports the widest range of MPC frameworks, but it has also the least security level.

3.2.1 Related Work

The most recent and comprehensive work on comparing MPC frameworks was presented in [170]. It tries to give an overview on the state of the art from a user's point of view. It shows the huge amount of work done in this field but also the complexity when it comes to realization of certain problems and use cases. Furthermore, because [170] does not focus on a certain technique or scheme, it can only do a high-level comparison. In particular, no performance benchmark was possible because of the very different approaches considered. Additionally, in the given work they only analyzed basic operations in the spirit of "hello world" applications as was shown by the code snippets in the appendix. They did basic multiplications (Multiply Three), an inner product between two vectors and crosstabulation.

In our work, we follow a different path and focus on benchmarking of more complex algorithms, i.e., symmetric ciphers in our case. We decided to focus on two popular frameworks using very similar protocols for a fair comparison of the different approaches. We choose MPyC and MP-SPDZ, two representative frameworks for secret sharing based MPC, one being very easy to use and leveraging an asynchronous programming concept, the other implementing a circuit optimizing compiler together with an application virtualization concept. Both systems work on arithmetic circuits and provide the same correctness and privacy properties, i.e., ITS secret sharing by Shamir, if configured accordingly. To make results comparable, we only use Shamir secret sharing in both frameworks which in linear and allows for local addition of shares and multiplication with publicly known constants. However, because multiplication of secret values is a non-linear operation after each multiplication step communication is required between the nodes to generate a fresh sharing of the product [171], [72], [73], [172]. This means that the runtime of the protocols is governed by the number of rounds needed by the protocols and the number of rounds is determined by the multiplicative depth of the circuit to be executed. Because this is the limiting factor it seems legitimate to estimate the multiplicative depth of an algorithm to reason about the expected runtime for a given function to be computed. However, as we show in our work, it is not straight forward to estimate the runtime from the theoretical number of multiplications necessary, especially for our intended application of encrypting and decrypting bulk data with symmetric ciphers.

Besides comparing MPC frameworks, this work was also motivated by the progress made in developing optimized ciphers for application in multiparty computation (MPC), fully homomorphic encryption (FHE) and zero-knowledge proof systems (SNARKs) [173], [174], [175], [176], [177] and [178], which are typically only benchmarked in ideal settings (almost perfect connectivity and fast servers). This somehow contradicts the idea of generic platform independent secure computation, which should be the goal for widespread use of the technology. In our work we are using this new cipher designs and compare them with the baseline being AES. Additionally, we also include well known stream ciphers in our comparison because they are also considered interesting candidates for MPC. In fact, this work was motivated by the difficulties encountered in straight forward implementation of MPC optimized ciphers which turned out to not behave as expected in all circumstances and are not platform independent in regard to their performance, i.e., they depend on used data types, necessary data conversions and communication performance in different networking environments. We are analyzing important dependencies we encountered during our implementations and show that further work is necessary to generate portable code and good interoperability between MPC solutions. It is not the goal of our work to absolutely compare the many proposed ciphers, but to see how good their structure is suited in general for framework independent application and this is only a first step towards broader benchmarking activities and this is only a first step towards broader benchmarking activities.

3.2.2 MPC Frameworks

Various frameworks have been developed over the last years, specifically in the open-source domain. As presented in [170], even for experts it is hard to understand the current status of the given solutions, and even more so for general users. And although they invested a lot of work, they only tested some basic operations and not their stability and efficiency for more complex algorithms.

We selected two candidate frameworks, i.e., MP-SPDZ and MPyC, which both have the capability to work in the semi-honest setting with honest majority and they use secret sharing based protocols. Nevertheless, on the higher layer and the programming paradigm they follow a completely different approach. In the following we are discussing the architecture and basic features of the two frameworks used for cipher evaluation in this work. In Table 3.1 and Table 3.2 we extend the results of [170] for MP-SPDZ and MPyC which were not considered in their work.

	MP-SPDZ	MPyC
Protocol family	Hybrid	MC
Parties supported	●	●
Mixed-mode	●	●
Semi-honest	●	●
Malicious	●	○
Language docs	●	●
Online support	●	◐
Example code	●	●
Example docs	◐	●
Open source	●	●
Last major update	●	●

Table 3.1: A summary of defining features and documentation types, which extends the results of [170] with MP-SPDZ and MPyC.

MPyC. MPyC [179] is a fork of the discontinued VIFF framework [180]. Interestingly, although MPyC is the successor to VIFF and actively maintained, it is only rarely used for MPC benchmarks. In our opinion, it follows an interesting concept which could lead to both easy access for programmers and reasonable performance. It

		MP-SPDZ	MPyC
Data Types	Boolean	○	○
	Fixed int	●	●
	Arbitrary int	◐	●
	Float	●	●
	Array	●	●
	Dynamic array	○	●
	Struct	◐	◐
Operators	Logical	○	○
	Comparisons	●	●
	Addition	●	●
	Multiplication	●	●
	Division	●	●
	Bit-shifts	●	●
	Bitwise	●	●
Grammar	Conditional	●	●
	Array acccess	●	●
	Private index	ORAM	naive

Table 3.2: A summary of functionality and expressibility of each framework which extends the results of [170] for MP-SPDZ and MPyC.

is a Python framework that implicitly represents multi-party computations as graphs of regular Python values, secret-shared values, and operations on them. These operations are implemented by overloading of the corresponding plain-text operators, so the whole process is mostly invisible to the user of the framework. The resulting graph is built at runtime and evaluated asynchronously, so no static analysis, and therefore no optimization, is performed. However, heavily optimized primitives (mostly vector and matrix operations) are available.

The framework is passively secure in an honest-majority setting. Only Shamir's secret sharing and pseudorandom secret sharing are supported. As we are interested in ITS secure protocols we selected the first one for our comparison. On top of these protocols, both integer and fixed-point types have been implemented. For integer computations, there is both a type based on prime fields, and one based on bit fields; the fixed point type is based on prime fields. Additionally, MPyC provides an overloaded version of Python lists that can be obliviously indexed by a secret value.

Code written in the framework can be hand-optimized by making use of the built-in `gather()` and `_reshare()` methods. Applying the first method on a shared value will run all outstanding asynchronous computations and then unpack the share to return an element of the underlying field. Any further computations on it will then be performed locally only. Applying the second method to this value will return a shared value again. Everything that happens in between is therefore part of a single round of communication. Obviously, this can destroy correctness, and it is left to the programmer to ensure that it does not.

Internally, MPyC makes ample use of this facility to improve the performance of its built-in methods and operators, and to provide efficient vectorized versions of some common operations. We looked at four of those operations and compare them to their unoptimized counterparts in Table 3.4.

MP-SPDZ. MP-SPDZ [181] is a fork of the SPDZ2 software which was originally developed at the University of Bristol and is based on the SPDZ type of protocols [182], [183], [184]. The name SPDZ is derived from the authors of the original protocol in [182]. Since forking, MP-SPDZ has integrated more and more protocols and is now the most prominent framework used for benchmarking MPC protocols in general. It supports very flexible use of different protocols and also separation of online and off-line phases for performance measurements.

Its approach is to let users write their programs in a Python-like language that is then heavily optimized and compiled to bytecode for a fast virtual machine implemented in C++. Because the framework is geared mainly towards benchmarking, there is no way to interface with the compiled programs. Inputs have to be passed via specially formatted files and it is sometimes tricky to do so and the documentation on this has some gaps.

The framework implements a wide variety of protocols for several different security models, based on arithmetic as well as Boolean circuits. Both integer and fixed-point numbers are supported and security models of honest majority as well as dishonest majority are supported, even for both semi-honest and malicious adversaries. In this work we were only using the `shamir-party.x` program as this resembles the same protocol used by MPyC which was required for a fair comparison of the different programming models and also fulfill the ITS requirements providing long-term security.

3.2.3 Algorithms

In the following we are quickly reviewing the basic principles and properties of the algorithms we selected for implementation. The algorithms have been selected because they are considered to be lightweight or because they were specifically proposed as ciphers optimized for application in MPC settings based on secret-sharing with a low circuit depth for multiplication gates. In general, the algorithms should help to understand how the two approaches of MP-SPDZ and MPyC are able to optimize the processing of the various ciphers and how suitable the structures of the ciphers are for the respective frameworks.

AES. The advanced encryption standard is the de-facto benchmark when it comes to MPC for the evaluation symmetric ciphers since [185]. Although not considered as practical in the beginning, a lot of progress was made in the evaluation of AES in secret-sharing based systems with [186] claiming the best performance. They report on a cluster of three 20-core servers with a 10Gbps connection, which carries out over 1.3 million AES computations per second, which involves processing over 7 billion gates per second. However, these results are achieved with a dedicated protocol for Boolean circuits supporting only three parties, and a lot of parallelization that can always be achieved in encryption tasks with symmetric ciphers.

However, these numbers heavily depend on the environment they are measured in, specifically the network setup (throughput and latency), the level of batching and parallelism applied, as well as the overall latency of the system from reading input data to delivering the result.

AES implementations were available in both frameworks, and we took them as baselines for our benchmarks. This is particularly interesting for systems operating on arithmetic circuits, because AES does not lend itself well to secure computation over prime fields [187], [188].

ChaCha20. ChaCha20 [102] is a stream cipher and the successor to Salsa20, and is one of several novel ciphers recommended for new implementations by the eSTREAM [189]. This cipher is also used in practice, together with the Poly1305 authenticator [190] it has been adopted in secure protocol implementations such as OpenSSH and OpenSSL, as well as for Google Chrome on smartphones.

ChaCha20 is a typical ARX-cipher, consisting only of unsigned 32-bit integer additions, fixed-width bit rotations, and XORs. In the MPC setting, this mixing of integer and logical operations is a problem and suggests two different implementation strategies: one would be to represent one's data as integers and convert to and from a bit-level representation as needed. The other, likely more efficient, strategy would be to use a bit-level representation throughout, implementing the integer operations via Boolean circuits, though this seems viable only in the case of addition.

Trivium Trivium, based on a nonlinear feedback shift register (NFSR), is another one of the ciphers in the eSTREAM portfolio [189] and has been accepted as an ISO standard [191]. It has a simple structure with only bit operations, so that it can be applied to resource-constrained environments such as wireless sensors in IoT.

By iteratively using NFSR, the degree increases rapidly and the output is a complex Boolean function over key and IV bits. The internal state of Trivium consists of 288 bits. It is initialized by a key and IV of 80 bits each, with all other bits except for the last three set to 0. To complete the initialization, 1152 keystream bits are generated and discarded. The generation of a keystream bit is the same for initialization phase and regular operation: the state is shifted by inserting three new bits. One such bit is generated by two XORs and one AND. The XOR of the new bits is the output bit.

We chose Trivium because of this very simple construction and the low multiplicative complexity it promised. It has also been considered in [192] for usage in homomorphic encryption. As detailed, it only takes three multiplications to produce one output bit, but by construction, this operation can be parallelized for up to 64 bits. This means that a fully optimized implementation of Trivium should be able to generate 8 bytes in a single communication round.

LowMC LowMC is a block cipher based on a substitution-permutation network which can be parametrized in a very flexible way. The number of rounds needed to achieve the desired level of security is determined as a function of several parameters. The way this is done is to consider and try to bound all known attacks and choose the number of rounds so that the most effective attack for a particular set of parameters is just not able to violate the security expectation.

The first version of this round 'formula' was introduced at Eurocrypt 2015 [173]. Soon after, optimized attacks were demonstrated and as a result, an updated round formula for LowMC was proposed by the designers [174] to take these new insights into account. LowMC is instantiated with these parameters plus, for each round, three matrices and a scalar. One matrix is for the linear layer, its inverse is used for decryption, and the third is the key matrix that is used to compute a round key.

One round of LowMC takes a block, applies the determined number of 3-bit S-boxes to it, multiplies it with the linear matrix as detailed below, and then XORs this new block with the round key and round constant. Decryption is the exact reverse, using the inverse of the linear matrix. The output of the multiplications involving the three matrices is such that each bit of the new block (or round key) is the XOR-sum of the bitwise AND-product of the old block (or secret key) and a row of the respective matrix.

MiMC / HADES Introduced by [193], MiMC is a radically simplified construction based on the idea to explore the typical field size used in MPC. The numbers of rounds is $\lceil \frac{n}{\log_2 3} \rceil$ (with $n \bmod 2 = 1$ as the chosen block size). The round function just adds the key and a round constant, then takes the result to the power of three. To decrypt, the process is simply performed in reverse, but with the exponent $\frac{2^{n+1}-1}{3}$ instead of 3. As decryption is therefore massively more expensive than encryption, the authors recommend using modes where it is not needed. Encryption, however, should be very efficient. Since the design does not contain any S-boxes and only uses addition and multiplication, the cipher can be evaluated in a binary field without any conversion, and needing only two multiplications per round.

Later, [177] developed what they call the HADES design. The core idea of this approach is to apply reduced versions of the non-linear layer in some rounds. Instantiated for MiMC, this means that the cipher now operates not on a single, but any number of blocks, and in certain rounds, the exponentiation is only applied to some (in this instantiation: the first) of them. Additionally, a generalized MiMC was also presented [178] which can cope with prime fields and work on many field elements once and therefore gives good amortized cost if it can be parallelized. Our work is based on the initial design in [193] but can be naturally extended to the new versions.

3.2.4 Implementation Findings

During implementation of the different algorithms we were faced with some design decisions that we will report together with our findings and some problems we encountered. The implementations are released as open source modules on Github[1]

For AES no specific implementation has been done and the existing solutions were used. In the case of MP-SPDZ two AES implementaions are available and for fair comparison we used the one programmed in pseudo language and not the circuit provided as netlist.

ChaCha20. When implementing ChaCha20, the problem was indeed the mismatch between logic and arithmetic operations in the MPC setting, suggesting two possible implementations:

In the first variant, we use the equivalent of unsigned 32bit integers for the additions, switch to a bit-vector representation for the XOR and bit rotations, then again go back to the integer representation for the additions, and so on and so forth. Given the structure of ChaCha20, this necessitates 640 decomposition and 320 recomposition operations per block of 64 bytes.

In the other variant, we convert the input values only at the beginning and end, and therefore have to perform addition on the bit-level. This can be done in one multiplication per bit plus one vectorized multiplication of all bits of the addends.

As expected, the second variant performs better, but we could only test this in MPyC, as we were not able to correctly implement the first variant in MP-SPDZ. Of note here is also the wide disparity in performance between the frameworks.

Trivium. Since Trivium operates on bits, there was no question of different approaches, and the implementation is a straightforward translation of the specification. MP-SPDZ was able to generate optimal code without any additional work on our part, but for the MPyC implementation, we had to do some optimizations by hand to achieve the expected performance.

In the description of Trivium, we said that it allows for (at least) 64 output bits to be computed in one round of communication.[2] We therefore implemented the cipher so that it always generates blocks of 64 bits, by simply running the output function 64 times in a loop. The MP-SPDZ compiler rolls out this loop, and by analyzing the data dependencies within, finds that all operations can be performed in one round, and generates the respective code.

MPyC, however, had to be helped along by way of the `gather()` operation described above. We placed it right before the loop, and the corresponding `_reshare()` immediately after, therefore achieving by manual optimization the same result that the MP-SPDZ had produced automatically.

[1]`https://github.com/Archistar/archistar4mpc-cipher`, accessed 2023-01-10.

[2]It turned out that even 72 bits can be computed per round. As Trivium is initialized by turning over the internal state (288 bits) four times, initialization can be done 16 rounds of communication.

LowMC. Implementing LowMC was a surprise in many ways. It has already been observed by [174] that the enormous number of XORs entailed by the matrix operations of LowMC means that they can no longer be considered free. We found this to be true to an even larger extent in our setting.

Even though it was clear that the MPyC implementation could only be done on a bit-level representation with all the overhead it entails - using one of the recommended parameter sets (256 bits blocksize, 80 bits keysize, 12 rounds), and keeping in mind that even a small Python integer weighs in at 26 bytes, the multiplication matrices take up almost 50 mega-bytes - we were still negatively surprised by the resulting performance.

The MP-SPDZ immediately performed closer to our expectations, achieving 6 KB per second.

MiMC / Hades. MiMC is specified for GF(2n), but can also be used in GF(p). We chose the first variant, for which no conversion is necessary, as only XORs and multiplications are performed. Our findings indicate that encryption is indeed fast, as expected. However, the high number of rounds that are necessary to achieve adequate security, has a noticeable impact on performance. Only a few measurements were done on decryption as it was immediately obvious that the high cost of reversing the exponentiation made it as slow as predicted.

From MP-SPDZ to MPyC. As already mentioned above, the two frameworks follow different design paradigms from an architectural view. MP-SPDZ uses a compiler to generate intermediate byte code which is then interpreted by a virtual machine and MPyC makes heavy use of asynchronous IO in Python. Therefore, it is interesting to see what improvements could be achieved by optimizing the circuit in regard to its multiplicative depth and how this compares to the asynchronous software architecture. Furthermore, we tested if the best of both worlds could be achieved by a straightforward combination.

Algorithm	r	init	enc	dec	s (kB)
AES-128	10	-	53	-	3416
ChaCha20	20	-	1732	1732	1025
LowMC	12	1	12	12	13669
MiMC	82	-	164	15498	1403
Trivium	-	13	1	-	241

Table 3.3: Number of rounds, multiplicative complexity, byte code size of the selected algorithms

In Table 3.3 we give an overview of what the MP-SPDZ compiler was able to do with our implementations. For each cipher, it first lists the number of rounds of the cipher itself. This is followed by the multiplicative complexity of the initialization phase, the encoding, and the decoding operation. Finally, the table lists the size of the resulting bytecode. As can be seen, the multiplicative complexity of LowMC seems especially favorable, taking only one communication round for initialization, and one per round of encryption/decryption, which means encryption and decryption take a total of twelve rounds each. Trivium, however, takes 13 rounds for initialization, but then only one round total for encryption. As expected, the high number of conversions necessary for ChaCha20 is reflected in its very high multiplicative complexity.

In order to benefit from the optimizations achieved by an optimizing compiler we developed a transpiler to translate the intermediary bytecode of MP-SPDZ to MPyC programs. The goal of this effort was to compare the performance of the same byte code on the two platforms in order to develop a better understanding of the differences in the evaluation of the same circuit in the two frameworks. Additionally, this solution gives MPyC users access to an optimizing compiler which can execute circuits with optimized multiplicative depth.

We tested this approach on the Trivium implementation because the compiled code is very simple and basically consists of long chains of additions, punctuated by blocks of batched multiplications. This structure also explains why we did not get any good performance out of this approach. MP-SPDZ considers additions to be "free" and therefore does not vectorize them. In MPyC however, we observed that just adding a number of scalars is substantially slower than batching them up as vectors and using the built-in `vector_add()` operation. The significant difference can be seen in Table 3.4 between the performance of "naive" operations ("MPyC") and their vectorized built-in counterparts ("MPyC Vec").

3.2.5 Performance Evaluation

In this section we report our performance comparison results. We started from an ideal setting with tree nodes and an almost ideal network setting, i.e., all nodes running on the same physical host. All tests were done on a rather standard hardware, namely a Dell Latitude E7440 notebook with a Core i7-4600U CPU running on 2.1 Ghz with 4 cores.

In all our tests we did not try to optimize the overall throughput by exhausting all hardware resources through massive block level parallelism, we only intended to measure the time for one block (or a fixed number of blocks) in sequential mode. Compared to other works which optimize overall throughput by parallelization, we think it is essential to understand the behavior of a single block. This defines the expected latency from input to output of the basic operation, parallelization is then always possible, if the higher-level functionality allows it, e.g. for block cipher in counter mode, and enough hardware resources are available. In our work we focused on other parameters we were interested in. In particular, we wanted to also understand how the frameworks behave practical network setting reaching from LAN configurations to worldwide deployments as well as how many nodes can reasonably supported for the given task.

3.2.5.1 Performance of Basic Operations

As a basis for the understanding of the expected performance, and as a reference for the actual platform we use, we provide the actual measured latency for basic non-linear operations which require network communication. Linear operations can be done local and are considered to be negligible compared to the ones presented in Table 3.4. In particular we looked into the timing of multiplication and vectorized version thereof.

n	MP-SPDZ	MPyC	MPyC Vec
	map(operator.add, a, b)		vector_add(a, b)
1	<1	7	14
10	<1	46	19
100	<1	353	71
	map(operator.mul, a, b)		schur_prod(a, b)
1	12	59	60
10	13	258	82
100	17	2309	336
	reduce(operator.add, a)		mpc.sum(a)
1	<1	<1	9
10	<1	33	12
100	<1	507	18
	reduce(operator.mul, a)		mpc.prod(a)
1	<1	1	12
10	97	381	163
100	1017	3834	416

Table 3.4: Time (ms) for 100 basic operations over vectors a, b of length n

The main observations from this experiment are the following. In MP-SPDZ, latency for addition is vanishingly small; for multiplication, it grows slightly (but clearly sublinearly) with increasing vector size, except for the multiplicative reduction, which exhibits an almost exactly linear slowdown with increasing input size. This may seem surprising at first glance, but is actually to be expected. Contrary to the other three operations under test, the multiplicative reduction cannot be performed in one communication round. As it is written, the multiplication operations form a linear list, so one should expect it to scale linearly with input size. The only optimization possible would be to organize the multiplications as a tree, so that multiplications on the same level could be performed in parallel, and the whole operation would scale logarithmically. Such an optimization is, however, not trivial to find and prove correct. It is therefore not surprising that the MP-SPDZ compiler did not perform it.

In MPyC, the unvectorized operations scale about linearly with vector size. The vectorized operations are faster, but only the sum is clearly sublinear in its behavior, scaling somewhat similar to MP-SPDZ's vector multiplication; the other operations exhibit a noticeable slowdown as their input vectors grow in size.

3.2.5.2 Performance of Cipher Implementations

The results from our benchmarking of the cipher implementation with almost ideal networking (localhost) is shown in Table 3.5. There the results for AES are the baseline and show the basic difference and potential of the basic frameworks. In MP-SPDZ AES latency is about 20 times lower than in MPyC with optimized code. Chacha20, the lightweight stream cipher, did not perform better. In MP-SPDZ it took almost the same time and in MPyC it took even twice the time of AES. Most surprisingly LowMC performed worse than expected. Leaving aside the initialization time it took twice the time of AES on MP-SPDZ and was 6 times slower on MPyC. MiMC the second specifically for MPC designed cipher performed very well. It could not achieve a speedup on MP-SPDZ but performed 10 times faster on MPyC for encryption. Unfortunately, because of its design, decryption takes substantially more time (between a factor of 30 and 50), which makes it less attractive for some applications. Trivium, another stream cipher under test, performed exceptionally well on both frameworks. On MP-SPDZ it was more than 5 times faster and in MPyC with certain manual optimization we achieved roughly a 20-time speedup compared to AES. However, it has to be noted that Trivium in the standard configuration has only a security level of 80 bits compared to the other ciphers which provide at least 128 bits. Nevertheless, the structure Trivium is based on seems well suited for MPC implementations and should be used as a basis for future designs with stronger security levels.

Cipher	MPyC unopt		MPyC opt		MP-SPDZ	
	init	enc	init	enc	init	enc
AES-128	-	-	-	110	-	5
ChaCha20	-	-	-	216	-	6
LowMC	764	679	772	637	28	9
MiMC	-	-	-	11	-	5
Trivium	3300	23	780	6	18	<1

Table 3.5: ms/Byte for encryption in 3-party MPC, no latency

3.2.5.3 Network Latency and Loss

Besides the basic performance in ideal settings is it also important to investigate other aspects that are relevant for the real-world performance of MPC, but that sometimes gets sidelined: network latency and network loss. One stated reason for not caring (too much) about it is that it is assumed that performance degrades linearly with increasing latency. This is indeed what we found in our experiments, however, what was surprising though was the size of the constant factor, and that it was significantly different between the frameworks.

We measured the behavior of the implementations under increasing network delay and loss. The measurements were done on the very same hardware with all processes running on the same host and by using kernel level `netem` features to emulate network delay and loss. To make the results easier to compare, we performed computations that used about 1000 rounds of communication, then normalized the obtained time to one communication round.

The results of our experiments with variable latency are shown in Figure 3.1 and Figure 3.2 for a one-way delay from 0 to 50 ms. In the ideal zero-latency setting, a multiplication round takes less than a milli-second in both MPyC and MP-SPDZ; every 5 ms of one-way delay (which can be estimated to equal about 10 ms of round-trip delay) adds about 5 ms to MPyC, but almost 10 ms to MP-SPDZ. This means that the seemingly superior performance of MP-SPDZ versus MPyC disappears rapidly with increasing latency. The situation is slightly, but not fundamentally, different when we look at batched multiplications. Repeating the same test with vectors of length 100, with zero latency we get about 3 ms and again less than one ms for MPyC and MP-SPDZ respectively, and once again an increase of about 5 ms vs. almost 10 ms for 5 ms of one-way delay. This means that multiplications (looked at in isolation) are faster in MPyC than MP-SPDZ even in low-latency contexts. The breakeven point for Trivium, on the other hand, is around 35 ms in our measurements, which is in a realistic range for reasonable distributed deployments which go beyond a single data center implementation.

All this indicates that although the Python based implementation of MPyC may be slower and seemingly less capable than MP-SPDZ with its impressive optimizing compiler, MPC operations are ultimately bound by the network, and MPyC profits from Python's highly-optimized asynchronous network stack.

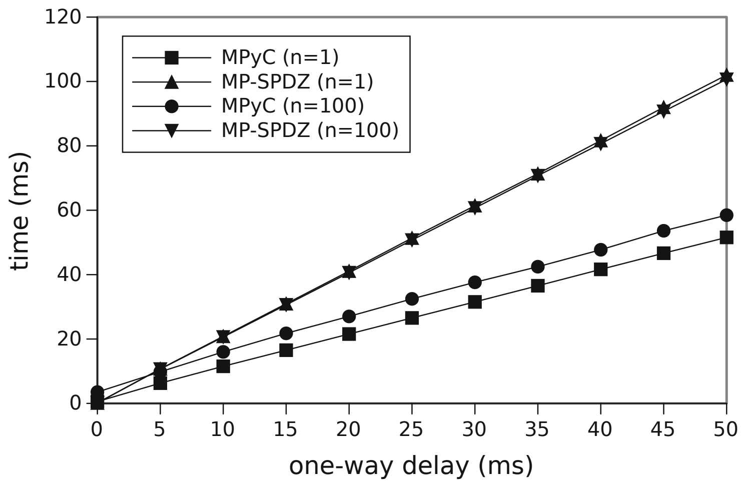


Figure 3.1: Time for one multiplication with increasing network delay.

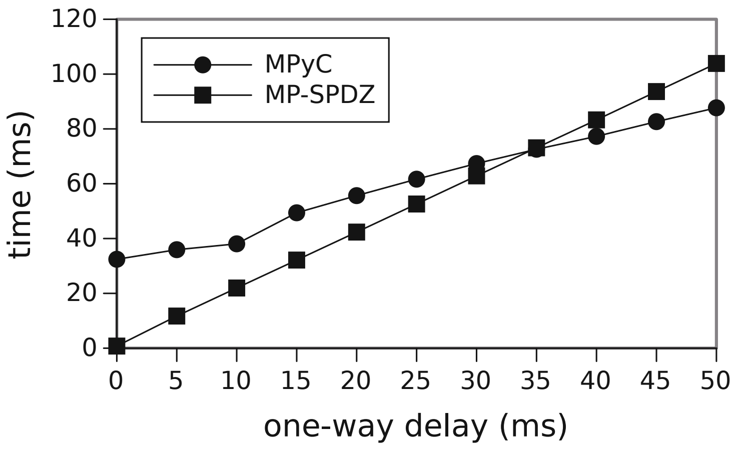


Figure 3.2: Time per communication round of Trivium with increasing network delay.

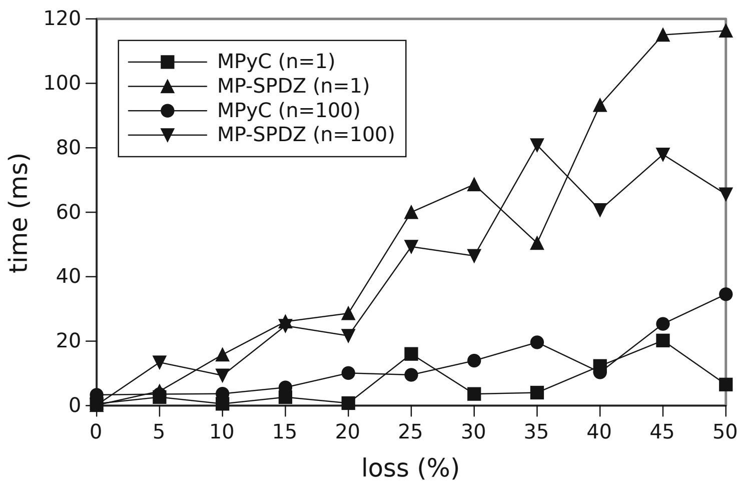


Figure 3.3: Time for one multiplication with increasing network loss.

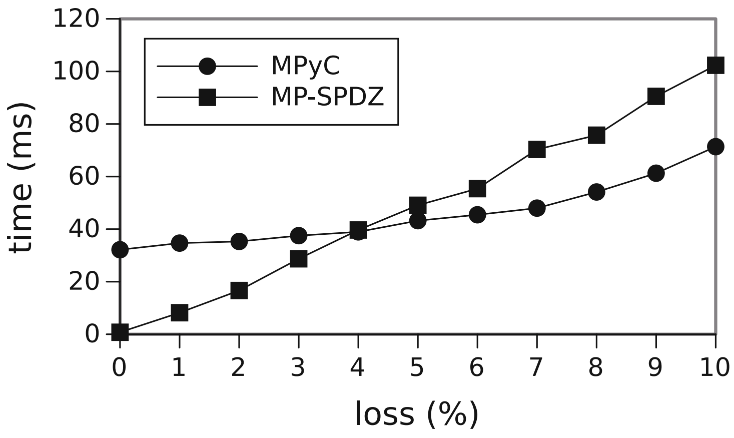


Figure 3.4: Time per communication round of Trivium with increasing network loss.

In a second series of experiments we tried to simulate packet loss. The results are shown in Figure 3.3 and Figure 3.4. The slowdown is somewhat erratic, but otherwise comparable to the one we observed with one-way delay.

3.2.5.4 Scalability

Although MPC is very appealing, in reality most systems are only supporting two or three nodes. From an application point of view this is often disappointing, as in many typical use cases more parties than that want to collaborate. For example, in a secure auction many more users are submitting bids and if they are not MPC nodes in their own right they have to trust the MPC nodes not to collude. From the protocol it is clear that the communication overhead limits the number of nodes for particular computations, however, in our study we wanted to see what is possible with current frameworks in settings with more than 3 parties, as this could greatly improve the security (reconstruction threshold) but also the resilience and robustness of the system.

In Figure 3.5 we show the results for more parties in the typical honest majority setting. Here, MP-SPDZ performs better with increasing number of parties than MPyC, which experience a significant slowdown with increasing number of parties.

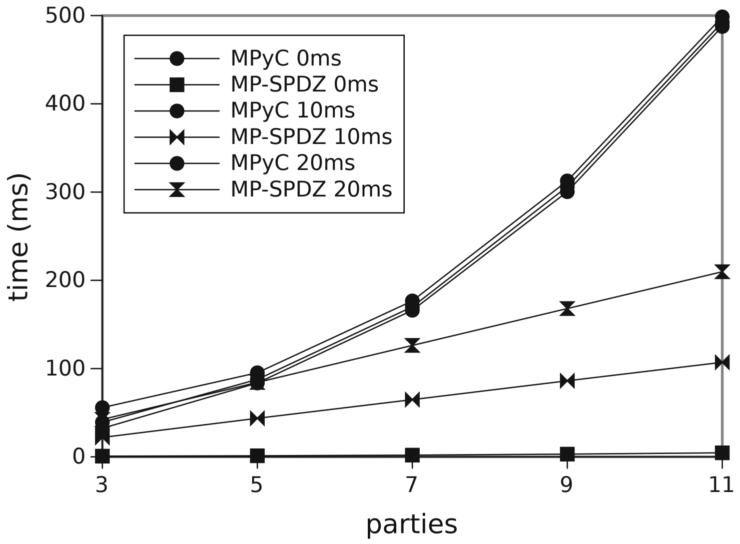


Figure 3.5: Time per communication round of Trivium with increasing number of parties.

3.3 A Novel MPC Based Platform for Data Markets

After essential evaluation and benchmarking of MPC frameworks providing information-theoretic security, we are studying ways to build secure cloud-based systems and platforms out of it. Emerging needs for data spaces in the business domain are driving the research of new approaches to leverage cloud computing and the Internet of Things in a more secure and privacy friendly way. One such domain is manufacturing, where the sharing economy is expected to have a significant impact in multiple dimensions, ranging from reduced costs, over increased innovativeness and competitiveness to considerable environmental benefits. However, mutual distrust and data sovereignty is of special concern in the manufacturing industry, e.g., related to corporate secrets and customer data.

Control over data is still hampered in large infrastructures and the trend to centralization is alarming for a prosper economy. By connecting production facilities to the cloud, many security and compliance approaches relying on a pure contractual basis (service level agreements, SLA) must be reconsidered, especially with respect to the existing oligopoly in the cloud market. A clear understanding of emerging security and privacy issues is needed, and security paradigms based on cryptography rather than SLAs have to be considered in order to guarantee confidentiality and data sovereignty in the cloud.

Contributions. In this section, we propose a networked, decentralized architecture for end-to-end verifiable yet privacy-preserving auctions, in order to support future manufacturing clouds and marketplaces. Our platform combines different technologies in a new way to provide adequate protection of business secrets as well as transparency and end-to-end verifiability.

On the technical side, this is achieved by a combination of secure multi-party computation and zero-knowledge proofs of knowledge that also incorporates the edge of the network to achieve the main properties envisaged by the different stakeholders. The platform not only allows for determining the lowest bid, but also advanced price-finding mechanisms based on price-ratio methods. Our solution minimizes the necessary mutual trust, not only between different bidders but also towards the auctioneer. We implemented a proof of concept of our approach, integrating and extending several existing tools and frameworks, e.g., for MPC and zkSNARKs. The performed benchmarking demonstrates the practicality for realistic sets of parameters for use cases encountered in our use case.

3.3.1 Related Work

MPC can be considered the most practical approach for generic computation on encrypted data. This means that virtually any function can be computed in an MPC system in principle, however, due to the overhead introduced by MPC protocols they are slower than a classical computer by orders of magnitude. Nevertheless, the first realization of a real application was demonstrated in an auction held for the Danish sugar beets market in 2009 [194]. This was the first large-scale and practical application of multiparty computation and enabled farmers to get a fair market clearing price. They used a local setup with three computers in the same room and ran a semi-honest MPC protocol to calculate the clearing price. About 4000 values for prices were supported at most and 1229 bidders participated in the auction. The inputs from the individual bidders were encoded by verifiable secret sharing and the computation lasted for half an hour.

This setting already shows the basic problem in the application of MPC. Because MPC lacks scalability only 3 nodes were chosen to execute the computation. This prevented the bidders from directly participating in the MPC protocol but required them to encode their inputs for the external MPC nodes, which made the computation on behalf of the bidders, i.e., it is still an outsourcing problem from a users' perspective. The improved security in this setting is evident, but to further increase the trustworthiness of the system, and to decrease the trust assumptions in the MPC nodes, some form of public verifiability would be desirable to assure the bidder about the correctness of the computation.

To cope with this issue new research combined the mechanisms with blockchain and (non-interactive) zero-knowledge proof (NIZK) techniques. The blockchain is an ideal candidate to be used for storage of relevant audit data in an accessible manner, however, because all data written to the blockchain is visible to every party additional machinery is required to maintain confidentiality and privacy. NIZKs enable parties to publish proofs about statements without revealing secrets per se (witnesses) and are therefore an ideal tool to integrate blockchain with the confidentiality preserving MPC functionality.

Sánchez [195] proposed Raziel, a system that combines MPC and NIZK to guarantee the privacy, correctness and verifiability of smart contracts. The idea underlying Raziel is a smart contract which in addition to the standard properties also guarantees correctness of auctions. The validity of the generated NIZKs can also be verified by third parties, thereby achieving publicly verifiability.

Another approach to verifiable auctions has been presented in [196] and a software prototype can be found on GitHub[3]. The solution combined homomorphic commitments and NIZK together with a verifiable comparison protocol to achieve a secure FPSBA. The system is verifiable and privacy preserving against outsiders, however, a trusted auctioneer is still required because he learns all bids.

Furthermore, Blass and Kerschbaum [197] presented Strain, a protocol to implement sealed-bid auctions on top of blockchains that protects the bid privacy against fully malicious parties. In a nutshell, the protocol works as follows: bids are encrypted bitwise and are stored on the blockchain. Bidders then to run interactive zero-knowledge

[3]`https://github.com/HSG88/AuctionContract`, accessed 2022-02-23

protocols generating proofs of relations between bids, thereby realizing auctions in a peer-to-peer fashion. Albeit being scalable by the peer-to-peer nature, the protocol still needs a semi-trusted auctioneer as arbiter and requires all parties to be online during all auction phases. It also leaks the full order of all bids compared to the winning bid as required by auctions.

Kosba et al. [198] presented Hawk, a framework to establish privacy preserving smart contracts on the Ethereum blockchain. Hawk is intended to protect transaction data on chain by leveraging zero-knowledge proof techniques. The goal was an easy-to-use framework providing a compiler managing the cryptographic tasks. Up to our knowledge, the Hawk framework has still not been released yet. However, it is considered that the approach cannot be efficiently integrated with MPC.

In another work [199], Galal and Youssef utilized Zero-Knowledge Succinct Noninteractive Argument of Knowledge (zk-SNARK) to realize privacy friendly auctions on a blockchain. However, their solution makes use of a trusted auctioneer who learns the bids. This is contrary to our goals; however, the approach contains interesting aspects and by realizing the auctioneer in a distributed fashion using MPC, the concept resembles the core ideas of our approach. Additionally, cryptocurrencies have been used to incentivize fairness and correctness, and avoid deviations from the MPC or NIZK protocol. In these systems money has to be escrowed in deposits which are only returned if the behave honestly. This in effect encourages parties to strictly follow the protocols to avoid the financial penalty. Protocols in this direction have been proposed in [200, 201, 202, 203].

Baum [204] proposed publicly auditable MPC; however, this work is mainly of theoretical interest and never really implemented. It also integrates with SPDZ but is likely to be too expensive for practical applications, as the idea is basically to make each computational step verifiable by adding a zero-knowledge proof.

3.3.2 Market Platform

A key requirement in a cloud manufacturing is to optimally match supply and demand, i.e., available manufacturing resources and customers' needs. In the following we thus present a verifiable and privacy-preserving marketplace for manufacturing resources. In our scenario, we consider a marketplace provider, at which owners of manufacturing sites can sign up as *producers* and register their machines as well as meta information such as configurations, quality levels, etc., cf. also Figure 3.6. *Customers* (or *buyers*) can now put orders, and producers can provide bids to win the order. Leveraging multi-party computation to ensure confidentiality, blockchain to immutably store encrypted bids and results, and zero-knowledge proofs to ensure integrity and verifiability, the marketplace will then match the bids against the order, and announce the winning bid. The precise data flow will be described in Section 3.3.3.

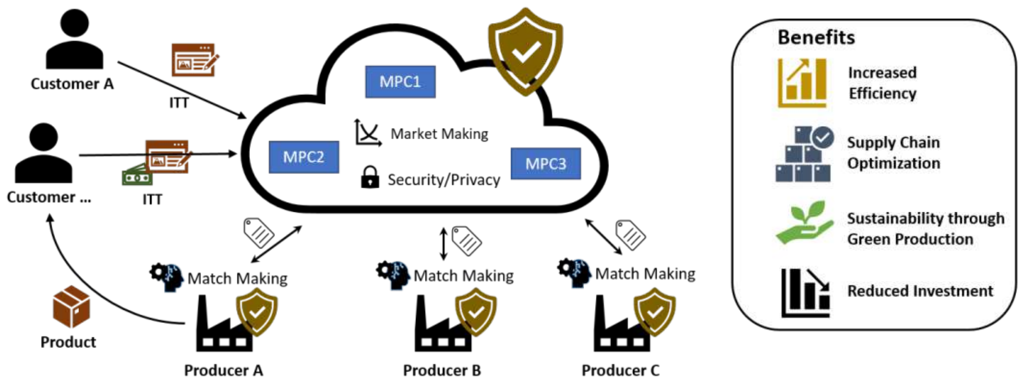


Figure 3.6: Use case of cloud manufacturing and marketplace.

3.3.2.1 Security Requirements

In the following, we introduce the security and privacy requirements to a marketplace for cloud manufacturing applications, which were derived in collaboration with industry partners. On a high level, we found an environment which is aware of their assets but is currently not sure how they could leverage the data they own because of concerns regarding the insufficient protection of business-critical information, and the risk of a potentially colluding harmful environment during auctions. More precisely, the requirements are as follows.

Confidentiality. Confidentiality of the producers' bids is of utmost importance through all phases of the auction. In particular, the bids do not only need to be protected from unauthorized access through competitors, but also from the platform provider. This is because of the risk of this central entity colluding with certain producers, thereby fully undermining the price finding mechanisms.

Furthermore, our interviews with industry also showed, that production and shop floor data can be business critical. That is, competitors should not gain any information about a producer's current capacity utilization, machinery status, or process information on production lines.

Integrity. Besides the requirement of correctness in the case of exclusively honest entities, it is necessary that the integrity of an auction's result can also be guaranteed in the case of a malicious operator of the marketplace. This even needs to hold in the case that the provider is colluding with other entities in the system, including producers and buyers, in order to ensure that no party can manipulate the outcome of the auction in their own interest.

Availability. While this is often not considered in the design of cryptographic protocols, it turned out to be of high importance to our partners. On the one hand, producers demand assurance that they will not miss opportunities. On the other hand, related to integrity, producers also need to be guaranteed that they cannot be excluded from an auction; that is, whenever a producer places an offer, it shall also be guaranteed that this offer was indeed considered.

Anonymity and Pseudonymity. In addition to confidentiality of bids, producers may wish to even hide the information whether or not they placed an offer for a given auction, as this might already reveal sensitive information about the current utilization or condition of a production line. Depending on the specific business model of the marketplace, this requirement, however, needs to be balanced against the marketplace provider's needs.

Fairness. Another important aspect for the clients was fairness in terms of fair conditions. This can also be interpreted as an open and transparent way of computation and selection of the winning offer. However, the evaluation criteria for matching and ranking need to be clear and traceable. The goal is to establish a fair comparison of offers on a comprehensible algorithm to establish fair market prices.

Transparency. Finally, transparency requires that all participants in the system are able to trace progress and activities on a high level. Our partners were also interested in historical data in case they were offline for certain times and could not participate in auctions. However, all this functionality needs to be achieved without compromising any of the previous goals, especially those related to confidentiality and privacy.

3.3.2.2 Auction Mechanism

Auctions are considered a good mechanism to achieve fair market prices for goods. The underlying idea is that every bidder in an auction bids the real value which leads to a real market price. This market mechanism can be undermined in various ways by malicious parties, especially in centralized online platforms. As summarized in [196] there are four main types of auctions which are of practical interest and two of which work on hidden bids.

First-price sealed-bid auctions (FPSBA), where bidders submit their bids in sealed envelopes to the auctioneer, which opens them to determine the bidder with the highest bid. Second-price sealed-bid auctions (Vickrey auctions) are similar to FPSBA with the exception that the winner pays the second highest bid instead.

In addition to the traditional approaches, various other price finding mechanisms are relevant in the industry. In the requirements assessment for a manufacturing marketplace, it became clear that it is not only the price, which is relevant for selection of the best offer, but also other parameters. Currently delivery options, logistic costs, quality requirements and other parameters are also part of the decision process to select the best offer.

In that sense, it is essential to have a very flexible mechanism for ranking offers. At the heart of our matching mechanism is thus the idea of a *matching score*, which allows to address additional targets and be more versatile in configuration. In this mechanism, the buyer can define different criteria and priorities among them, which should be taken into consideration. This starts from basic capabilities of the production facilities and can be arbitrarily extended for additional topics as mentioned above. These parameters are defined as part of the tender and distributed along with it. The matching is then based on capabilities of a producer, which are immutably registered on a blockchain when signing up as producers or registering equipment. In essence, this leads to a case where the price has to be combined with a matching score to rank results.

Specifically interesting are price finding methods based on linear interpolation. The so-called *price ratio methods* are often used to compare offers which are not comparable otherwise. In this approach, a score R for a given bid is computed as:

$$R = \omega_L \cdot \frac{L}{L_{max}} + \omega_P \cdot \frac{P_{min}}{P}, \tag{3.1}$$

where for simplicity we only consider a single criterion here. In this equation, L means the level of fulfillment of this criterion by an offer, which corresponds to the matching score, and L_{max} is the best level possible. The actual price for a certain offer is then given in P, with P_{min} being the lowest in the auction. The ω_l and ω_P are weights for the respective ratios on fulfillment grade and pricing, which are defined by the buyer and can be either public and known to the producers as part of the tender or also kept secret, depending on the preferences of the buyer. By inspecting the method it gets clear that not all values can be computed at the edge, i.e., on the client side, as in particular the ratio $\frac{P_{min}}{P}$ of the prices requires the minimum over all offers.

From this analysis it becomes evident that simple price ranking is not enough for winner estimation and the many possible options require a flexible computing system which goes beyond oblivious sorting of bids. In fact, each buyer would like to define his specific matching criteria and also decide on a ranking mechanism to fulfill his needs. This turned out to make the overall system architecture more challenging and rendered many smart contract based approaches from the literature inadequate. On the other hand, the MPC solutions available can cope with the degree of flexibility required but do not provide means for public verifiability as needed for our architecture.

3.3.3 Framework

The proposed framework is designed to address the security objectives defined in Section 3.3.2.1, especially bringing together typically contradicting goals of privacy and verifiability in a single solution. It has a decentralized architecture and follows data minimization principles.

By the use of secure multiparty computation, the platform itself is operated in a way that the provider does not learn sensitive data and by making every step of the of the tendering process verifiable the trustworthiness is achieved. For end-to-end verifiability, publicly verifiable zero-knowledge proofs of knowledge are generated for all computations, even for the MPC steps. Finally, to trace all interactions and proofs we use a distributed ledgers which serves as a trust anchor and immutable append-only data base.

3.3.3.1 Data Flow

In the following we detail the data flow in our platform. To ease understanding, Figure 3.7 provides a high-level overview, where we omit setup steps for the sake of clarity.

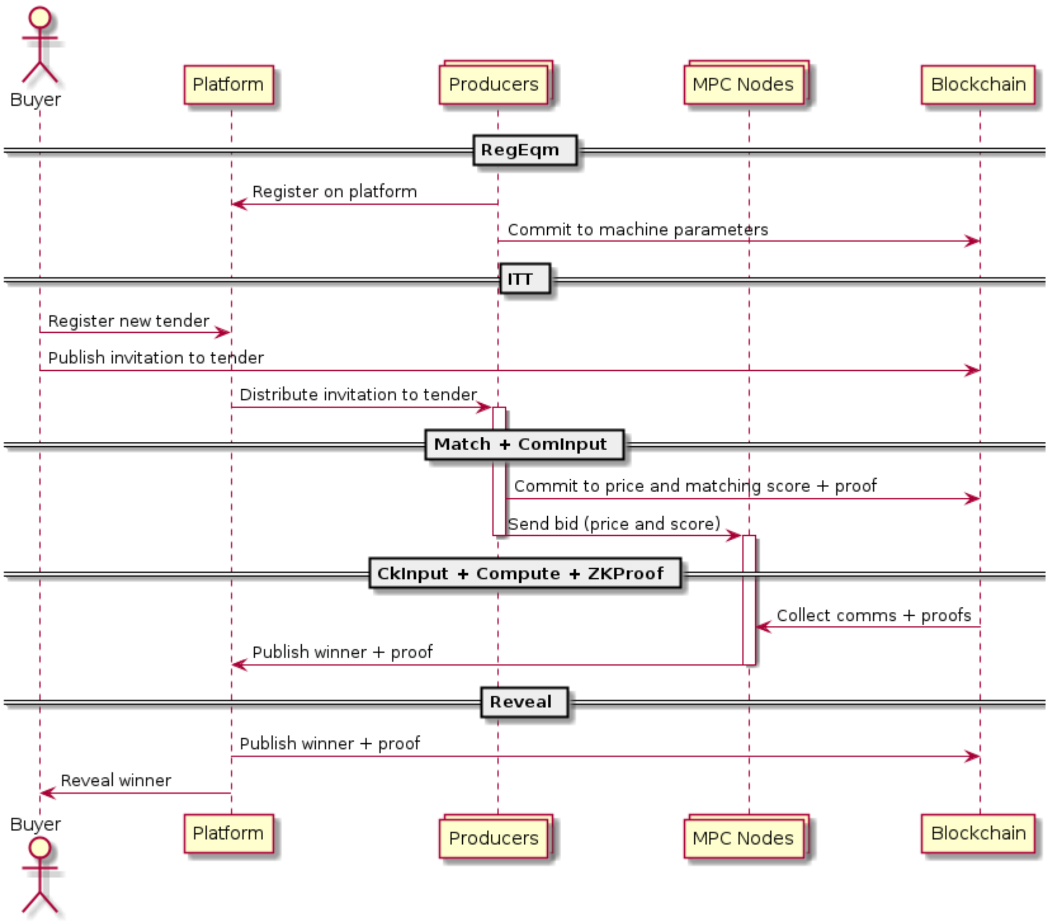


Figure 3.7: Sequence diagram for an auction.

Setup phase. The following setup steps are necessary to operate the system.

1. On the one hand, ZKSetup is used to generate the common reference string (CRS) needed for the NIZKs. On input the security parameter and a circuit, this algorithm outputs the CRS which is assumed to be implicit input to all further algorithms and parties. It is important to note that the system is specifically designed in a way that this step is only needed once and does not need to be invoked again

if different ranking mechanisms are used, as they are all supported by the specifically designed circuit with built-in flexibility. In practice this setup algorithm can be run in dedicated setup ceremony, including, e.g., secure hardware elements or dedicated MPC-based ceremonies.

2. On the other hand, RegUser is a protocol which is run by the user and the platform to register with the platform. It is used to generate necessary identities and credentials to authenticate the user and set up the necessary permissions on the ledger.

Auction phase. After the setup is complete the following steps are conducted in the protocol for a particular auction.

1. RegEqm. A client registers equipment (machine) for usage in the system. On input machine parameters, this algorithm outputs a commitment to the machine parameters which is then stored on the ledger. The registration of new equipment has to be done before a producer can participate in an auction.
2. ITT. A buyer sends a request for quotation with relevant parameters to all parties by storing them to the blockchain.
3. Match. Based on the tender information received, the producers compute a matching score for their machines. Based on the score a local matching decision is done to decide whether or not to participate in the auction. If the producer does not participate, the local process is aborted.
4. ComInput. If the producer is participating in the bidding, it computes a NIZK for the matching score and a commitment to the bid. The bid commitment and the proof are stored in the blockchain.
5. Input. In this step the producers (i.e., bidders) send their bids together and matching scores to the MPC system in a secret-shared fashion.
6. CkInput. The MPC system retrieves the corresponding commitments and proofs from the blockchain and verifies them in the encrypted domain. This is done by recomputing the commitments on the shares (for bids and matching scores) at each node and comparing the reconstructed commitments with the plaintext ones. Additionally, each node verifies the proof for the local matching score individually. If either of the checks fails, the system complains about the producer.
7. Compute. The MPC system calculates a ranking based on scores and bids according to the ranking function defined by the buyer.

8. ZKProof. The MPC system generates a NIZK for the winning bid, proving that it is the best ranked result according to the predefined ranking function. It does so by each node computing the proof on its share of the result.

9. Reveal. To reveal the result in a verifiable form, the winning bid is reconstructed from the encodings as well as the final proof enabling the verification of the winner calculation. The data is recorded in the blockchain and finalizes a particular auction.

There are many variations possible in practice, but this will only result in subtle changes, e.g., if the winning producer and/or bid has to be kept secret.

3.3.3.2 Protocols

Different protocols have been used, extended and integrated to achieve all desired properties for our framework. At the core we combine multiparty computation with non-interactive zero-knowledge proofs of knowledge (NIZK) to achieve confidentiality and public verifiability that the same time. Regarding MPC we do not rely on any specific protocol but only require a method which is based on secret sharing. However, because we aim at public verifiability the correctness of the computation is going to hold even if all nodes are corrupt. Therefore, depending on the individual assumptions made for the MPC deployment, it can be sufficient to rely on passively secure protocols.

To achieve verifiability, the system is based on adaptive zk-SNARKs as introduced in [205]. Working with commitments to track different steps in the process is essential to guarantee privacy of sensitive data. However, the protocol is not guaranteeing any authenticity which is essential to track the flow from end to end. Therefore, we leverage ideas from ADSNARK [206] and use signatures on the commitments to assure the authenticity of the data right from the source. In our use case both can be used, standard signatures but also group signatures or delegated signatures [17], if a certain degree of anonymity is still required, e.g., if it should not be visible which subsidiary of a larger organization the resources belong to.

An important requirement was to reduce the number of times the setup procedures of the zk-SNARKs have to be executed. Ideally, it has only to be done once when initializing the platform and can then be reused for all subsequent auctions. Therefore, we use a hybrid approach that turned out to be very efficient. On the one hand we use the idea of subroutines (sub-qaps), which are basically predefined subroutines at setup time but can be connected during proof generation by means of intermediate commitments, to establish the required circuit. This concept is very flexible with only little overhead, i.e., the additional commitments increase the proof size and verification time for each subroutine defined. To enable even more freedom in the configuration of ranking algorithms we integrate the ideas of universal circuits as presented with MIRAGE [207].

Altogether, at setup time we generate a proof circuitry which comprises both, static elements and freely re-configurable components to get the best of both worlds. Starting from a common pattern of auction markets, we can now instantiate the right building blocks to support a wide range of auction systems with extensive configuration options. To that end, this approach is somehow similar to partially reconfigurable hardware.

3.3.3.3 Security

In the following we will informally discuss the achieved security goals; a more formal treatment is planned as future work. Moreover, we also give some design rational and explain several architectural decisions.

Confidentiality. The privacy of sensitive information is protected in our framework by two main primitives. On the one hand, MPC is used to compute the winner of the auction in a privacy preserving way and sensitive data is protected by the input privacy provided by MPC. We rely on information-theoretically secure MPC protocols to provide long-term security.

On the other hand, to enable transparency we are recording inputs at different stages in the blockchain. To achieve confidentiality there we use commitments in combination with NIZK to hide input while making the system verifiable. Given that sensitive inputs are never handled in cleartext in the system we achieve strong cryptographic protection, which also results in the discussed properties of bid privacy and posterior privacy. Furthermore, the selected adaptive zk-SNARK also provide ITS confidentiality. In particular, the commitments are perfectly hiding and the resulting NIZK system is perfectly zero-knowledge. Thus, privacy of data is secure in the long term and will never be compromised, even if quantum computers will be available. However, the binding and correctness properties of the protocols are not quantum-safe, and alternative protocols have to be researched with the advent of quantum computers. Nevertheless, the system can be considered used as long as scalable quantum attacks are not feasible and transition to new protocols are not immediately necessary.

In our marketplace, we even apply a decentralized matching which allows for local computation of a matching score and therefore follows data minimization principles. Alternatives would be a producer-based matching or a platform-side matching, which both would lead to problems: For the former, it would be required to share sensitive information about available capacities and process information with potential competitors, which directly violates the requirements. For the latter, either the platform would learn sensitive information, or the matching would have to be done in the encrypted domain.

Integrity and correctness. In essence, the basic idea of the framework is to preserve the integrity and authenticity of data in the system and to prove the correctness of each computation in between. However, instead of directly signing the input to the system, only commitments are signed and also put in a blockchain, which guarantees that producers are bound to their bids and bids.

In our case we use the extractable commitments presented in [205]. In the original work these particular commitments were used with a dedicated key to distinguish between input from different parties, but this key is produced in the initial setup phase and has to be distributed to parties, which opens up many attack vectors in practical implementations. We rely on locally generated private keys used to sign the commitments, which never leave the local area and are registered with the platform or the blockchain. This would also allow for the use of group signatures for even more flexibility in the management of edge components without sacrificing the security.

After the tender is initiated and all bidders recorded the required data to participate in the auction, the MPC computation is started. The input is comprised of the private bid, the private matching score as well as the data also recorded on the blockchain, i.e., the commitments on initial machine parameters and the matching score, thereby guaranteeing confidentiality while still binding bidders to all input values. Finally, the MPC network not only outputs the winning bid, but also a NIZK proving its optimality, thereby guaranteeing the correctness of the final result.

It is worth noting that the performed local matching introduces another problem with the integrity of data: As the matching score is relevant to calculate the ranking on the platform, it has to be assured that the score was computed correctly. Therefore, the bidder is required to generate a proof on the score before sending it to the MPC system. We do so by forcing the bidder to commit not only to the bid but also to the matching score and additionally to generate a NIZK which is then stored in the blockchain, letting everybody to also verify the local pre-processing.

Finally, it is important to note that with this approach we are basically tweaking the security model of MPC for the overall system. Because the correctness of the computation is publicly verifiable by means of NIZKs, the integrity of the computation can even be assured if all MPC nodes maliciously deviate from the protocol specification. Even more, in our setting malicious behavior can be attributed to the right stakeholder, i.e., it is not possible to blame the platform for malicious input from bidders or vice versa. This is achieved by letting the MPC system check all inputs for consistency with the information in the blockchain before it computes a result. Only if all inputs are consistent with the stored commitments and the matching score is computed correctly, the MPC system will incorporate the bid in the auction, and only then it will be able to compute a proof for the winning bid.

As a result, the full auction flow is accompanied with NIZKs and every participant can verify the correctness of the auction from end-to-end. Even if privacy is compromised by an adversary which compromises enough MPC nodes to recover the bids, he will not be able to influence the winning bid or market mechanism.

Availability. The availability of the system is provided by a blockchain component which provides the properties to serve as robust and immutable public append only log. Depending on the deployment of the MPC system, also robustness properties such as fairness or guaranteed output delivery can be achieved. Additionally, as the system is non-interactive, client-side computations cannot be interrupted or blocked by individual participants, resulting in a highly available decentralized architecture. Although the platform server is currently needed to run auctions it would also be possible to remove this single point of failure.

Anonymity and Pseudonymity. For the given use case it is not desired to build a completely open and permissionless infrastructure. The clients in this system are part of an ecosystem which requires some level of assurance for producers and buyers. Therefore, we only provide pseudonymity for certain steps in the auction. The pseudonyms are maintained at the platform which mainly prevents the buyers from bypassing the business model of the brokerage role of the platform. The only relevant issue for producers might be that one can determine whether or not a specific producer has submitted a bit for a given auction. This leakage can be easily abolished by always participating in the auction protocol but with a ∞ bid.

Fairness. In our prototype, the tender information was identified as public and could thus be put into the blockchain, which also serves as a broadcast channel in this step, so that all participants are reliably informed about new opportunities as well as the detailed evaluation and ranking criteria of an auction.

Although for certain buyers it would be preferred to not reveal the details of the ranking mechanism (e.g., the weights in Eq. (3.1)), in all auctions will the matching mechanism be transparent and consistent for all producers.

Transparency. By logging every step into the blockchain in a privacy-preserving way and also proving that all computations are correct, we achieve public verifiability. Every user of the system will thus be able to verify all auctions based on the public data stored in the blockchain without compromising the privacy of individual inputs, thereby achieving the requirement of transparency.

3.3.4 Evaluation

In order to evaluate the efficiency and practicability of our framework, a proof-of-concept prototype has been implemented in Python, which has been used to study and benchmark different use cases. The basis for the implementation was existing work on *PySNARK*[4], *qaptools*[5] and *MPyC*[6], which have been integrated and extended with novel

[4]`https://github.com/meilof/pysnark`, accessed 2023-01-10.
[5]`https://github.com/Charterhouse/qaptools`, accessed 2023-01-10.
[6]`https://github.com/lschoe/mpyc`, accessed 2023-01-10.

functionality, notably Universal Circuits. The new framework seamlessly integrates the steps along the data flow.

All measurement results presented in this work have been measured in a local setup on a single Intel NUC computer equipped with an Intel(R) Core(TM) i5-8259U CPU running at 2.30GHz maximum frequency.

Winning bid computation. The basic operation in winner determination is finding the maximum of the ranked bids. The computation depends on inputs of all bidders as well as the buyer and is therefore done by the MPC system. We present measured computing times for computing the basic result for different numbers of parties without any additional proofs in Table 3.6. Note that the system we used to take the measurements only had four CPU cores, so not all of the increase in time between the settings with four and five parties is attributable to the inherent computational overhead.

#bids	3 parties	4 parties	5 parties
10	0.9	1.3	3.4
100	10.9	16	35
1000	115.3	176.8	-

Table 3.6: Computation time (s) with increasing number of nodes under 3 minutes.

The given measurements hold under the unrealistic assumption of no network latency. We therefore also performed measurements assuming different network delays, as shown in Table 3.7, which suggests that for our use cases the latency of the network has a higher effect on the overall efficiency than the selected MPC framework, see also [8].

#bids	0 ms	2 ms	20 ms
10	0.9	1.1	4
100	10.9	11.4	36.6
1000	115.3	124.1	386.1

Table 3.7: Computation time (s) with increasing network latency between 3 parties.

Winning bid proof generation. For our use cases, the generic approach of proving the correctness of every computational step turned out to be unnecessarily inefficient. Alternatively, the size of the equation system used in the zkSNARK, i.e., the quadratic arithmetic program (QAP), can be significantly reduced by generating a single proof at the end of the computation, showing the optimality of the announced winning bid with respect to the defined ranking function.

The proof can be defined as simple as shown in the Listing 3.1, where `matching` holds the matching score of all bidders and `prices_rec` are a list of all reciprocals of the prices submitted. The weights defined by the buyer are `w_l` and `w_p`, cf. also Eq. (3.1). `res` is the output truth value which is 1 if and only if no bid is ranked better than the winner.

```
res = 1
for l, p_rec in zip(matchings,
                    prices_rec):
  score = w_l*l + w_p*p_min*p_rec
  res *= (score >= s_min)
res = auction.get_public_output(res)
```

Listing 3.1: Proof computation for winning bid generation.

This type of proof turned out to be very efficient and the measurements in Table 3.8 show that all steps in proof generation are extremely practical and are even far below the times needed by the MPC system to compute the winner.

#bids	t_{Setup}	QAP size	t_{Prove}	t_{verif}
10	0.4	226	0.3	0.02
100	3.6	2206	2.1	0.6
1000	22.3	22006	18.7	6.6
10000	186	220006	160	65

Table 3.8: Performance comparison. Times are in seconds.

The same performance can be achieved if the winning bid is not the very best but among a predefined threshold or at a particular place, i.e., like in the Vickrey auction. This variation can be achieved by counting the number of bids above a threshold and optionally also below, which results in the same running time as above.

Fixpoint operations. To work with the quotient method as shown above, division and fixpoint operations are needed. Implementing this within the MPC component would have had a significant impact on the performance and drastically limited the number of bids which can be handled in reasonable time. We therefore compute the reciprocal in the pre-processing step at the producer and send it (together with the bid) as input to the MPC system. To ensure end-to-end verification, we then also have to generate a proof that the reciprocal was computed correctly, but this introduces only minimal overhead. In our implementation clients submit the price in integer and fixpoint representation together with the reciprocal and the resulting error term. This results in a very simple and efficient proof computed on the producer side and even less work for the MPC phase, because computation of reciprocals would consume substantial resources. The additional check necessary to prove consistency of fixpoint representation with price value is shown in the Listing 3.2, where the `res` is 1 if and only if the multiplication of the price `p_fxp` with the reciprocal `p_rec` corrected by the error term `p_err` equals 1 and the error term given by the representation is within the allowed bound.

```
res = ((p_fxp * p_rec + p_err) == 1)
res *= (s_err * fxpmul_ <= s_fxp)
```

Listing 3.2: Proof generation for correct fixpoint conversion.

3.3.5 Verifiability with zkSNARKs

As discussed in Section 3.3.2 a marketplace for outsourcing manufacturing tasks requires a very flexible way to compare offers and match producers. zkSNARKs lack in this respect, because they require the circuits to be proved to be fixed during the setup phase. We circumvent this downside by integrating two mechanisms: subroutines and reconfigurable circuits.

Subroutines. On the one hand, we use the properties of adaptive zkSNARKs to also enable subroutines, as already introduced in [205]. Subroutines are built by committing to their input and output and therefore interconnect these modules to a larger circuit. This approach also provides the first level of reconfiguration and reuse. Existing circuits can be reused without any additional setup call and rewired at runtime by using the according commitments as wiring mechanism to enforce correctness of signals along the computation chain. Figure 3.8 shows the main structure used in our framework which the blocks being subroutines.

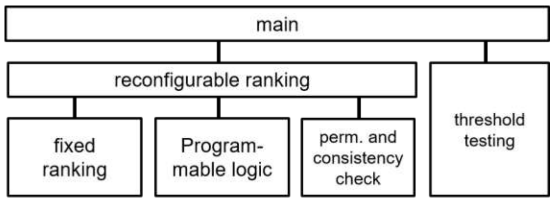


Figure 3.8: Internal structure of the proof generation.

Reconfigurable circuits. In addition to the described subroutines, we also incorporated universal circuits. They are general purpose computing blocks which can be reconfigured at runtime without calling setup routines. The concept has been inspired by *MIRAGE* [207] but integrated with the subroutine mechanism, which is well suited for this. In particular, to assure consistent use of signals within the universal circuit the necessary additional permutation and consistency check was realized as subroutine. A dedicated subroutine was designed which does all checks and was compared to the integrated approach of *MIRAGE*. For our implementation we required only a simplified version of the first opcode presented in the paper, because we did not need any binary operations. The most interesting part four our work was to understand the impact of the size of the universal circuit on the proof generation time and to compare the performance of the original approach in *mirage* to our new way of integrating the permutation and consistency checks.

```
p1 = 1
p2 = 1
for i in range(len(zi)):
  p1 *= r2 - (zi[i]+r1*l1[i] )
  p2 *= r2 - (zi[si[i]]+r1*l1[si[0]])
res = (p1 == p2)
s = 0
for i in range(1, len(z1)):
  s +=
    (1 - (l1[si[i]] - l1[si[i-1]]))*
    (z1[si[i]] - z1[si[i-1]])
res *= (s == 0)
```

Listing 3.3: Explicit proof for permutation and consistency check of signals.

Permutation and consistency check are performed on public input `li` and private input `zi` which are part of the interface commitment for the universal circuit as shown in Listing 3.3. After calculation of sorted indices `si` the permutation check scores `p1` and `p2` are calculated. Additionally, the consistency checking score `s` is also calculated and both necessary conditions for a successful check are combined into the proof, i.e., assert `p1 = p2` and `s = 0`, which assures intermediary signals connecting gates are consistent, which assures intermediary signals connecting gates are consistent.

Generally, the use of commitments to intermediary results in circuits increases the size of the proof as well as proofing time. The overhead in size is not an issue for our use case and verification runtime is typically not a problem at all for marketplaces. The advantage of not requiring to call setup phase and generation of additional CRS components is by far more important in real world applications than the increased size or running time experienced. In Table 3.9 we show the measured times universal circuit composed of different amount of gates, ranging from 10 gates up to 10000 gates. The implemented opcode for our configuration consumes 35 equations in the QAP (20 equations for the logic and 15 equations for the consistency check) if directly combined in one subroutine. The overhead of 21 equations is required for a minimal benchmarking setup.

#gates	t_{setup}	Qap size	t_{prove}	t_{verif}
10	0.5	371	0.7	0.02
100	2.5	3521	2.2	0.02
1000	32	35021	38	0.1
10000	307	350021	378	0.9

Table 3.9: Performance for universal circuit with integrated consistency checks. Times are in seconds.

To further reduce the time for proof generation, we introduced a second approach for the integration of the consistency checking mechanism. In particular we realized the permutation and consistency check as separate subroutines. This means that during configuration of the universal circuit a commitment to the secret values is generated. This commitment is then used together with the public configuration values l_i to perform the permutation and consistency check in a standalone sub-QAP. The measurement results for this variant is shown in Table 3.10. There can be seen that the size for the logical part can be significantly reduced by about 40% which directly results in reduced proving times. Interestingly, the size of the QAP needed to do permutation and consistency checking is in the same order as the pure uc for our opcode and leads to similar proving times. This is very convenient because it naturally supports parallelization and leads to a reduced overall time although in total about 10% more equations have to be processed.

#gates	uc-Qap	t_{prove}	check-Qap	t_{prove}
100	2004	0.8	1804	1.2
1000	20004	18	18004	19
10000	200004	160	180004	180

Table 3.10: Performance for uc usage with external consistency check. Times are in seconds.

Overall complexity. When considering static scoring functions, the overall complexity of an auction is not dominated by the runtimes given in Table 3.7 and Table 3.8. When aiming for a general marketplace supporting flexible scoring functions, also the runtimes presented in Table 3.10 are to be considered, where the number of gates per bid strongly depends on the complexity of the scoring function and auctioning logic. For the case of a FPSBA with the scoring function defined in Eq. (3.1), two gates per bid are required.

3.4 Verifiable MPC Solver for the Assignment Problem

Air traffic management (ATFM) can also benefit from a privacy preserving data market similar to the manufacturing domain discussed in the previous section. Efficient ATFM and thus use of infrastructure and resources is of utmost importance for high competitiveness and low prices for consumers. With the increasing degree of digitization and the ongoing trend towards cloudification, it becomes easier than ever before to achieve the goal of efficient resource usage also beyond company boundaries, e.g., in a sharing economy, where an optimal match between supply and demand has to be found.

One important task in such a scenario is described by the linear assignment problem, which deals with the question how to assign n tasks to n machines while minimizing the total costs, knowing the costs of assigning each task to each machine. Assignment problems have been studied for multiple decades, and a variety of efficient algorithms solving such problems can be found in the literature.

However, when assigning resources among competitors, e.g, by the means of an auction, companies - and in particular potential competitors - might often have confidentiality concerns, as the individual costs per task might be sensitive and contain company secrets. This might lead to hiding the true valuations [208], or let companies not participate in an auction at all [209], e.g., because the leaked information might be used against them by competitors [210].

The classic approach in such a case would be to agree on a trusted party that collects all inputs from all participants, and locally computes the optimal assignment. However, finding such a trusted authority might be difficult in many situations, e.g., in the case of competitors from different countries, or in case of a high frequency of such assignments close to real-time. In this work we thus study solvers for the linear assignment problem, guided by the following main requirements:

i) No central authority shall be required in the entire process, i.e., all computations need to be carried out in a distributed fashion. In particular, all sensitive input data needs to be protected from unauthorized access by any involved entity.
ii) The output of the distributed computation shall be publicly verifiable (or at least by all participants), without the need to trust any other entity in the system.
iii) Computations need to be sufficiently efficient and scalable to support a high frequency of executions close to real-time.

Motivating Application Scenario. For specificity, we explain the motivating use case for our research, which was developed in close collaboration with relevant stakeholders from the aviation industry as an important step towards practical deployment [19, 18]. Deviations from original flight plans due to variance and external events such as changing weather conditions or congestions are part of the day-to-day business at airports. Therefore, an optimal reorganization of starting and landing sequences across competing airlines could contribute to minimizing costs or delays on a large scale. Additionally, other factors and dependencies on airline conditions have to also be considered. To do so, the current situation at an airport would ideally be continuously monitored and optimized, thereby considering airline priorities–determined, e.g., by passenger flows, crew constraints, or maintenance schedules–as well as airport constraints for most efficient operations.

The need for such a system has been underpinned by the European Organisation for the Safety of Air Navigation (Eurocontrol), which has developed a first system called User-Driven Prioritization Process (UDPP) [211] to enable airlines to react on varying conditions and swap flights within their own fleet for given flight sequence. It is assumed that UDPP "has the potential to safe hundreds of millions of Euros over the next 20 years" [212]. However, because airlines only optimize locally within their own fleet, the achieved improvement is not optimal on a airport level. Especially low-frequency carriers such as business aviation suffer from limited degrees of freedom in flight swapping.

One main reason why such a global optimization is currently not possible is the airlines' reluctance to share their preferences with other airlines, thus leaving significant room for improvements regarding the efficient use of limited resources at an airport. This is due to the fact, that their preferences directly derive from internal cost structures and processes which they consider sensitive.

In our approach we aim at tackling this problem by leveraging multi-party computation to develop a decentralized cloud-based platform that enables collaboration for optimal flight sequencing in challenging conditions. Based on dedicated market mechanisms, set up to incentivize airlines to participate in the system, a model for an optimization process was developed [19].

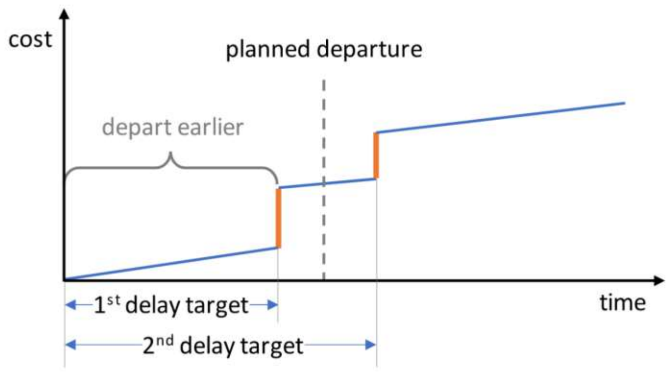
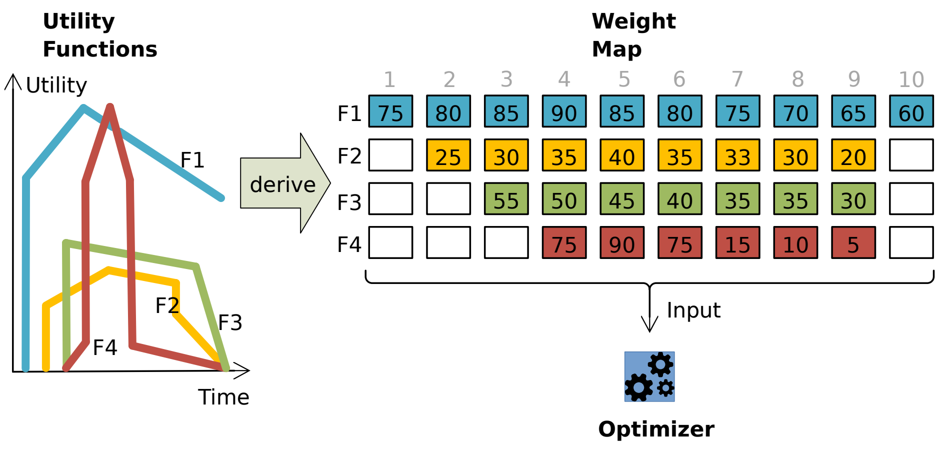


Figure 3.9: Non-linear cost function of departure or arrival delay.

For an airline, delay of a flight means additional costs which is often described by a non-linear step function with significant excess cost if certain delay targets are not met, e.g., connecting flights are missed. Hence, each flight has one or more delay targets as shown in Figure 3.9. From a modeling point of view, a weight map is used by airlines to define flight priorities for particular slots in the flight sequence as shown in Figure 3.10. Looking at the modeling of the optimization problem, it turns out that it basically resembles a so-called linear sum assignment problem (LSAP). The overall size of the problem which has to be handled is in the order of hundreds of slots to manage flight sequences in the upcoming hours at an airport. Thus, the problem would be easily solvable in the clear but gets already challenging if it must be done in the encrypted domain to protect the inputs of the parties.

Furthermore, related to (ii), given the financial and economic impact of slot assignments, airlines have a strong requirement regarding the authenticity of any slot assignment in order to overcome the risk of unjustified prioritization of a single airline. This is especially true for integration into a distributed platform [9], where strong authenticity



	1	2	3	4	5	6	7	8	9	10
F1	75	80	85	90	85	80	75	70	65	60
F2		25	30	35	40	35	33	30	20	
F3			55	50	45	40	35	35	30	
F4				75	90	75	15	10	5	

Figure 3.10: Flight prioritization is modeled by a weight map where priorities are assigned to each slot for each flight.

guarantees from end-to-end are key to increase user trust and willingness to participate. Even worse, because MPC does not scale well in the number of compute nodes, only a very limited number of nodes (3-5) seem feasible and not every airline can run its own node, i.e., also from this standpoint are means to verify optimization results essential. Nevertheless, being able to develop novel community-based collaborative applications as shown in Figure 3.11, without strong involvement of regulatory bodies can significantly speed up innovation in the air traffic flow management (ATFM) domain.

Finally, slots need to be assigned multiple times per hour due to the high traffic volume at major airports, and the frequency of delays, changing weather conditions, etc., thus requiring computations to be carried out in seconds to minutes at most, cf. (iii).

Related work. In the following we provide a brief overview over related work. Numerous privacy-preserving algorithms for different types of matching algorithms have been proposed in the literature. Considering general two-party linear programming (LP), [213] provide an efficient protocol for semi-honest parties, as well as extensions to prevent certain malicious behavior. Linear programming using MPC was considered by [214, 215, 216, 217].

Regarding specific assignment tasks, [218, 219] provide a privacy-preserving version of the famous matching algorithm by [220], based on mix networks and homomorphic encryption, however only considering a weak (passive) adversary model. A first MPC-based implementation was presented by [221], scaling to multiple thousand input values. A first provably secure and scalable implementation was later presented by [222] based on garbled circuits [223]. Specifically for LSAP, a privacy-preserving version of the Hungarian algorithm (as in Section 3.4.2.2) based on homomorphic encryption was presented by [224]. However, only the theoretical complexity of the protocol is analyzed, and no performance data is available.

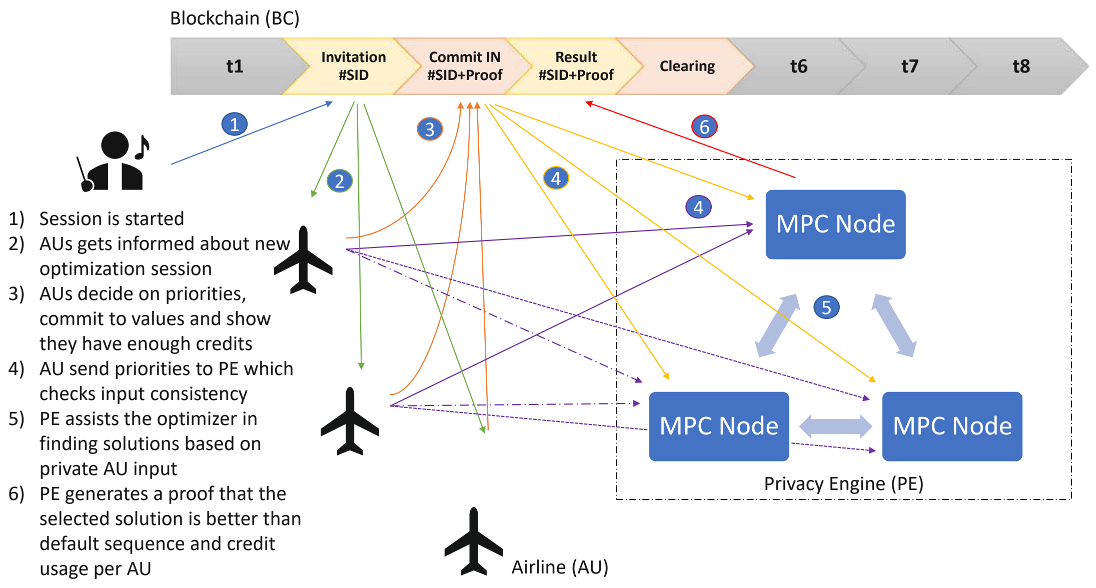

Blockchain (BC)
t1
Invitation #SID
Commit IN #SID+Proof
Result #SID+Proof
Clearing
t6
t7
t8
1
2
3
4
4
5
6
1) Session is started
2) AUs gets informed about new optimization session
3) AUs decide on priorities, commit to values and show they have enough credits
4) AU send priorities to PE which checks input consistency
5) PE assists the optimizer in finding solutions based on private AU input
6) PE generates a proof that the selected solution is better than default sequence and credit usage per AU
MPC Node
MPC Node
MPC Node
Privacy Engine (PE)
Airline (AU)

Figure 3.11: Overview of message flow and cryptographic approach for a fully decentralized marketplace with data privacy and authenticity from end-to-end. The participating airlines (AU) can collaborate and optimize their flight sequences without involvement of a trusted party.

None of these protocols offers means to publicly verify the correctness of the computation result, which however is a key requirement for our use case. A notable exception is the work by [225], who present verifiable MPC-based solutions for general linear programming. However, due to the generality of LP as well as the choice of primitives for the correctness proofs, our efficiency requirements cannot be achieved by this work. An efficient publicly verifiable auctioning platform for traditional sealed-bid auctions was recently proposed by [9].

For the sake of completeness, we also mention [204, 226] who give a generic framework for publicly verifiable multi-party computation, which however is mainly of theoretical interest in our setting due to the computational overhead.

Contributions. Following the above guiding principles, the main contributions of this section can be summarized as follows. In a first step, we perform a comprehensive analysis and comparison of secure multi-party computation (MPC) based approaches to solve the assignment problem in a privacy-preserving way. Secondly, we provide optimized implementations and benchmarks to compare the performance of different approaches, achieving an improvement over existing implementations by a factor of 50. Finally, we extend our implementation by public verifiability mechanisms based on zkSNARKs and Bulletproofs, thereby significantly outperforming related work and demonstrating the practical efficiency of decentralized, privacy-preserving, verifiable solvers for the assignment problem.

3.4.1 MPC Approaches to LSAP

In this section we briefly review the linear sum assignment problem (LSAP) and discuss important aspects when it comes to the realization of a privacy-preserving version based on MPC.

Assignment Problem. An instance of LSAP is described by a weight matrix W, where each $w_{i,j}$ represents the cost associated with matching task i of the first set (a flight in our case) and resource j of the second set (a slot in our case). The goal of the optimization is then to find a complete assignment of flights to slots which is of minimal cost according to a defined objective function, which is essentially the sum of weights.

Formally, let X be a Boolean matrix where $x_{ij} = 1$ if and only if row i is assigned to column j. Then the cost of the optimal assignment is computed as

$$\min \sum_i \sum_j w_{ij} x_{ij}$$

where the minimum is taken over all X, where each row is assignment to at most one column, and each column to at most one row. In our analysis the matrix W was assumed to be quadratic, however, it can be easily generalized to a rectangular problem by techniques discussed below.

A large number of algorithms has been developed for the LSAP, cf., e.g., [227, 228, 229, 230]. They range from primal-dual combinatorial algorithms, to simplex-like methods, cost operation algorithms, forest algorithms, and relaxation approaches. The worst-case complexity of the best sequential algorithms for the LSAP is $O(n^3)$, where n is the size of the problem.

For this work we selected at least one representative for each important class of algorithms and analyzed/implemented a MPC version of it to measure the practical performance which can be achieved. The selected algorithms are the

- simplex based solution strategy, where we leveraged linear programming based on the max-flow formulation approach.
- Hungarian algorithm (aka Munkres), one of the most important candidates for the primal-dual strategy,
- auction algorithm, a algorithm working in the dual domain of "shadow prices", and
- variants of shortest augmenting path (SAP) algorithms.

If the weight matrix is quadratic in size n, i.e., there is the same number of tasks and resources, the LSAP is called *balanced*. It means that both parts of the bipartite graph have the same number of vertices, when treating the problem as matching in bipartite graphs.

In the *unbalanced* case, the number of vertices is different for each side in the corresponding bipartite graph, resulting in a rectangular cost matrix $n \times m$. In that case, either not every machine can be matched to a task or not every task is occupied. Fortunately, most of the algorithms tested can be directly generalized to unbalanced problem solving. However, even if the solver only works for balanced problems, there are methods to convert an unbalanced solution to a balanced one. The straight forward technique is to augment the smaller set of vertices with $|n-m|$ additional entries and to connect them to the existing vertices with edges of cost 0. There also exist even more efficient technique [231] requiring even less additional edges. As a result, all these techniques are also compatible with MPC and only result in an additional pre-processing step.

MPC Aspects. Multi-party computation (MPC) allows parties to jointly perform computations in a way that only designated receivers obtain a result at the end of the computation, while no further information is revealed to any other participant in the system. In particular, the inputs are kept confidential from all other participants in the system. MPC can be considered the most practical approach for generic computation on sensitive data. It allows to perform arbitrary computations in principle, however, depending on the concrete computation to be performed, MPC protocols are often slower than a local computation by orders of magnitude.

Generally speaking, the algorithms used to solve the LSAP are not MPC-friendly. By their nature, they are mostly sequential with very little potential for vectorized operations. One such vectorizable operation is testing for zero. Even though this is a costly procedure in MPC that involves random number generation and comparisons, it can easily be done for a whole array in parallel, because testing one element does not involve any other elements of the same array. Also, the result can be cached, is only invalidated if the value itself changes, and can easily be recomputed on demand.

With most other operations, however, this is not possible. Take for example the minimum of a collection of elements. Finding it involves in the order of log n comparisons that have to be performed in sequence. Any change of the collection over which the minimum was computed could possibly change the minimum, so caching it is not viable. (When an element is added or changed, a single comparison is sufficient to recompute the minimum, but when an element is removed, the minimum has to be recomputed from scratch.)

To get tolerable performance we must trade-off between privacy and speed and inevitably leak some indirect information, e.g., branches been taken. However, the final assignment will be public and is known to be optimal, which also means some leakage. If that is not enough, [232] have shown how to efficiently implement graph algorithms that, like ours, reveal branching information, yet do not leak information by just obliviously permuting the original data.

Another problem is that every algorithm that uses some form of ε-scaling needs to use floating-point numbers. This is not just a question of numerical stability. If the underlying numerical representation is not precise enough, ε-scaling may terminate with a solution that is not optimal, or may not even terminate at all. In [233] the authors propose to multiply every element of the $n*n$ matrix by $(n+1)$ and use only integer values (down to 1) for ε but notes that this may in practice lead to integer overflow because prices can then be somewhere in the order of $n^2 \max_{(i,j)\in A} |a_{ij}|$.

3.4.2 Algorithm Evaluation

In the following we compare MPC performance of different solution strategies used to solve the assignment problem. The different algorithms have been implemented and benchmarked in *MPyC*[7] with default settings and a 3 party configuration. *MPyC* is based on secret sharing and is targeted towards semi-honest adversaries, however, the results can also be transferred to other frameworks with reasonable effort. The performance of the simplex solver from [225] served as a baseline for our comparison and was included in our analysis as shown below. A single Intel NUC computer equipped with an Intel(R) Core(TM) i5-8259U CPU running at 2.30GHz maximum frequency and with 32GB of memory was used as hardware, to make the results comparable. All parties were run in a local setup without any additional network latency and other restricting settings, if not explicitly stated otherwise. If not explicitly stated otherwise, all presented runtime are in seconds.

3.4.2.1 Simplex for Linear Programming

The assignment can be viewed in different forms. In essence, it is a special case of the transportation problem, which itself is a special case of the minimum cost flow problem, which belongs to category of linear programs. Therefore, the most generic solving approach would be to leverage existing simplex implementations in MPC and model the problem accordingly.

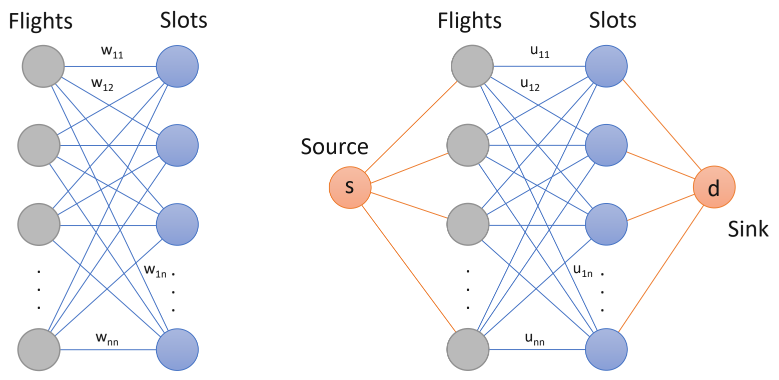


Figure 3.12: Bipartite graph structure for matching and minimum cost flow modelling.

The two major representations in LP form are shown in Fig. 3.12. They are either modeled as minimum cost matching in a bipartite graph or min cost flow problem.

[7]`https://github.com/lschoe/mpyc`, accessed 2023-01-10.

The latter may be rather counter intuitive because one would more likely expect the formulation as an integer program because of the binary nature of a match.

The corresponding LP is defined as follows. In a bipartite graph each edge (i, j), where i is in A and j is in T, is assigned a weight w_{ij}. Additionally, for each edge (i, j) we have a binary variable x_{ij} indicating if a certain edge is in the solution or not. Therefore, the resulting LP is given by:

$$\begin{array}{ll} \text{minimize} & \sum_{(i,j)\in A\times T} w_{ij}x_{ij} \\[1em] \text{subject to} & \sum_{j\in T} x_{ij} = 1 \text{ for } i \in A, \\ & \sum_{i\in A} x_{ij} = 1 \text{ for } j \in T \\[1em] \text{with} & 0 \le x_{ij} \le 1 \text{ for } i, j \in A, T, \\ & x_{ij} \in \mathbb{Z} \text{ for } i, j \in A, T. \end{array} \tag{3.2}$$

Because of the binary variables the model resembles an integer linear program. Fortunately, the problem can still be solved with standard methods known from continuous LP, albeit the integrality constraints, by simply dropping the integrality constraint. This is due to the fact, that for optimal solutions variables always take integer values, despite fractional values being allowed.

Furthermore, converting the problem to a maximization solution by inverting the weights leads to a further simplified formulation with less slack variables. In the presented use case we where anyhow maximizing the utility which is represented by the cost. Reducing the number of slack variables and problem size is essential for MPC performance and by converting the equality constraints to $\sum_{j\in T} x_{ij} \le 1$ and $\sum_{i\in A} x_{ij} \le 1$ the most compact formulation is achieved.

The implementation used is based on the simplex version presented in [225] and the results of our performance measurements are shown in Table 3.11. The upper part in the table are benchmarks for randomly generated weight matrices and the lower part is for typical sample data from our use case. Because the use case data is more structured slightly better runtimes can be expected, but the improvement is not significant. Even worse, the implementation was not able to generate the dual certificate also incorporated in the implementation because of the high memory usage required for problem sized bigger than 50. Furthermore, also the implementation itself also stopped working because of networking problems for problem sizes beyond 70. We did not further investigate this behavior as we were interested in alternative solution approaches anyway, however, this measurement served as a reference for our other implementations.

s	n	m	$iter$	$t_{simplex}$	t_{dual}
10	100	20	23	7.8	1.1
20	400	40	41	61.8	12.8
30	900	60	71	275	57
40	1600	80	104	806	157
50	2500	100	145	1920	410
60	3600	120	167	3468	-
70	4900	140	224	7333	-
10	100	20	21	7.1	1.1
20	400	40	41	59.7	11.9
30	900	60	66	253	57
40	1600	80	86	690	180
50	2500	100	117	1643	378
60	3600	120	140	2869	-

Table 3.11: Running time in seconds of LP solver with random weight vectors (above) and sample data from slot management problem (below). s is the size of the quadratic weight matrix and $n \times m$ is the dimension of the respective dimension of the A matrix in a LP of the form $Ax \leq b$.

3.4.2.2 Hungarian Method

Our second implementation is based on the Hungarian algorithm, also known as the Munkres or Kuhn-Munkres algorithm [234, 235, 236]. It was one of the first polynomial-time algorithms for solving the assignment problem. The basic idea of Munkres algorithm is to iteratively improve the matching in a bipartite graph along augmenting path between unmatched vertices. It has the fastest strongly polynomial run-time complexity with $O(mn + n^2 \log n)$, where n is the number of vertices and m is a number of edges, when implemented with Fibonacci heaps.

Our MPC version is based on a standard implementation as presented in [214]. Contrary to the original algorithm for manual evaluation with 4 phases, it comprises 6 steps but follows the main paradigm of finding minimum coverings of zeros in the weight matrix manipulated by reducing rows and columns.

However, a fully oblivious implementation would be rather slow and would require further measures to prevent from leaking information. Therefore, we opted to reveal certain aspects during computation, which will be discussed in Sec. 3.4.2.6. With this approach we achieved a substantial performance speedup compared to the simplex variant and after some manual optimization we achieved a speedup of almost a factor of 30 compared to the simplex.

The detailed performance results are shown in Table 3.12. The table shows the duration of an optimization run in seconds depending on the problem size. It also con-

size	#steps	$t_{munkres}$	#iszero	#min
10	38	0.8	400	821
20	86	4.4	1981	4016
30	184	17.8	10309	20734
40	308	33.7	26368	52931
50	441	67.0	52714	105712
60	601	125	98444	197280
70	775	170	154046	308603
80	962	220	230023	460682
90	1193	345	336901	674583
100	1401	412	460890	922709
10	36	0.9	399	818
20	92	5.0	2633	5323
30	185	16.1	10219	20556
40	341	53.0	34020	68253
50	488	77.8	67730	135772
60	725	215	149425	299310
70	959	293	264503	529618
80	1183	350	396112	792990
90	1450	546	591673	1184293
100	1770	869	884127	1769411

Table 3.12: Running time in seconds of Munkres with random weight vectors (above) and sample data from slot management problem (below).

tains information about the amount of costly MPC operations needed (minimum finding and zero testing) in the processing. Interestingly, for Munkres the average performance measured for the random case is twice as fast as for the particular use case data which is more structured. This is in contrast to the simplex solver where the algorithm could benefit from the structure in the use case data.

3.4.2.3 ε-scaling Auction Algorithm

The auction algorithm is an intuitive method for solving the classical assignment problem. It was first introduced in 1979 by [237], and has since then evolved as a valuable tool in network optimization [233]. Auction algorithms were selected for implementation because they have good practical average performance, although worst case performance is the same as for the Hungarian algorithm, i.e., $O(n^3)$.

In this paragraph we quickly recap the description from [233], which is based on the idea of economic equilibrium problem that turns out to be equivalent to the assignment problem; for a detailed presentation, we refer to [228, 238, 239].

The auction algorithm works in the dual of the problem acting on the so called *shadow prices*. In a first step, it determines ε to be the highest absolute cost, then repeats the auction with progressively smaller ε until it is smaller than n (the number of participants/objects). The rate α of decrease can be freely chosen.

This seems to indicate that floating point numbers have to be used in order to guarantee correctness and termination of the algorithm. However, as mentioned before, it is possible to scale all costs by $n+1$ and remain in the integer domain, with the risk of integer overflows.

In practice, the auction algorithm is very MPC-unfriendly. To find the initial ε involves taking the maximum of the whole cost matrix, and afterwards repeatedly finding the two indices at which the current price vector is minimal. As the price vector is highly variable, there is no possibility of caching. Given the high overhead of floating-point arithmetic in our development environment, initial benchmarks showed a slowdown of a factor of more than 10 compared to all other solutions already for small problem sizes, such that this type of algorithms was not further considered in our analysis.

3.4.2.4 Shortest Augmenting Path Algorithms

Another important category of algorithms are shortest augmenting path algorithms such as the Jonker-Volgenant-Castanon (JVC) [240]. These algorithms are somewhat similar to Hungarian method but apply a better way to update solutions together with a number of pre-processing techniques, including column reduction, reduction transfer, and reduction of unassigned rows. While the Hungarian algorithm finds any feasible augmenting path, JVC and a number of other algorithms find the shortest augmenting paths in a minimum cost network flow, where each node in S transmits one unit and each unit in T must receive one unit of a single commodity. Indeed, an optimal solution can be found by considering one source in S at a time and finding the shortest path emanating from it to an unassigned node in T.

We compared two implementations of this class. The first implementation is based on a solution used in the optimization module of SciPy module[8]. The second implementation is based on the py-lapsolver project[9], which itself is based on the Stanford ACM-ICPC teams site[10].

[8]`https://github.com/scipy/scipy/blob/v1.7.0/scipy/optimize/rectangular_lsap/rectangular_lsap.cpp`, accessed 2023-01-10.

[9]`https://github.com/cheind/py-lapsolver`, accessed 2023-01-10.

[10]`https://github.com/jaehyunp/stanfordacm/blob/master/code/MinCostMatching.cc`, accessed 2023-01-10.

n	random t_{scipy}	use case t_{scipy}	#iszero	#min	random $t_{lapsolve}$	use case $t_{lapsolve}$	#iszero	#min
10	1.2	1.7	641	199	0.7	0.9	492	380
20	7.1	11.9	5155	998	3.4	4.5	2860	1940
30	20.4	35.3	17535	2797	8.2	10.4	8322	5480
40	35.9	64.3	42641	5996	17.0	22.9	18429	11800
50	74.0	97.5	78803	10995	34.9	38.2	34280	21700
60	128	135	126972	18194	56.5	57.4	58762	35980
70	176	174	186655	27993	86.7	73.5	88404	55440
80	262	224	267247	40792	116	103	129582	80880
90	393	255	381406	56991	149	138	182114	113100
100	438	289	526712	76990	185	188	247934	152900

Table 3.13: Running time in seconds of scipy_lsa and lap_solver with random weight vectors and sample data from slot management problem. Measurement of number of zero tests and minimum search were done on use case data.

The performance results of the MPC implementation are summarized in Table 3.13. For random data the SciPy version performs similar to Munkres, but the algorithm also benefits from the structure in typical use case data. However, the MPC version of the ACM-ICPC solver turned out to be the fastest in class and also in general. It is more than two times faster than the SciPy version and more than four times faster than Munkres for the use case specific data. Nevertheless, in this version the smart updating mechanism based on augmenting paths did not make a real difference for different data types. Overall, the MPC-ACM-ICPC solver performs about 55 times better than the state of the art based on simplex solver which pushes practical applications to bigger problem sizes as in the case of the markets for air traffic management slot exchange.

3.4.2.5 Impact of Network Latency

To make the results comparable all benchmarks were done on the same single PC with the very same software framework and the same configuration. However, no delays between network nodes have been introduced. In principle network delays increase the time during sub-protocols for non-linear operations, i.e., multiplication steps in our protocols (cf. see [2]). Therefore, the experienced slowdown depends linearly on the multiplicative depth of the arithmetic circuit defining the function to be computed. To show the impact in practical terms, in Table 3.14 we show the impact of latency between MPC nodes. Real world solutions have to take this effect into account. It could lead to substantial performance penalties for distributed setups with larger network latency [8].

n → latency ↓	10	20	30	40	50
0	0.7	3.5	9.1	16.4	34.8
5	5.3	30.4	72.6	134.3	305
10	10.1	56.4	132	239	536
15	14.4	80.2	184	335	751
20	18.4	102	238	435	978

Table 3.14: Performance (running time in seconds) of MPC version of ACM-ICPC solver for different network delays and problem sizes. Latency is in milliseconds.

3.4.2.6 Leakage and Countermeasures

To achieve a good performance, trade-off between privacy and efficiency had to be accepted, which we will discuss in the following.

The Hungarian method is a completely sequential algorithm, and branching encodes information. Therefore, a fully oblivious version would run in constant time and not even reveal the number of iterations needed. However, this is not practical and render the technology obsolete for the aspired goal and problem sizes. Certain trade-offs were already considered in [225], where also minimal information is leaked by the algorithm to achieve better performance, e.g., the number of iterations.

In our implementation we never reveal costs at any time. Yet, we carry out certain tasks, such as the row and column covering, in the plaintext domain. This reveals information about the position of minimum elements in rows and column by doing public zero testing after the minimum of certain rows and columns have been subtracted obliviously. This leakage could enable an observer (e.g., a semi-honest MPC node) to learn certain aspects about the structure of the cost matrix by following the covering results for rows and columns over the iterations.

Similarly, for the class of shortest augmenting path algorithms, the MPC-ACM-ICPC implementation is completely sequential and therefore leaks in case of non-oblivious branching, while never revealing costs themselves.

To cope with certain leakages, dedicated countermeasures can be put in place in form of pre- and post-processing. For instance, for the Hungarian method, the leakage of the index of the minimum cost in a row or column can be defeated by obliviously permuting rows and columns before running the algorithm and reversing the operation after a solution is found. This is due to the fact, that an optimal solution is given by a full covering will all columns marked, i.e., reversing the permutation fully removes all information about the intermediate steps for an adversary. Computing permutations can be done highly efficiently with only a minimal overhead compared to the actual optimization process. This approach still enables reasonable performance but prevents from attributing row and column properties to the real cost matrix.

To be more concrete and estimate the overall runtime of the pre- and post-processing phase we implemented an oblivious shuffle2d and unshuffle2d algorithm permuting rows and columns of a $n \times n$ cost matrix through appropriate matrix multiplications. The runtimes are shown in Table 3.15. However, oblivious generation of the permutation matrix is included in the shuffling time, which can be clearly done offline to further speed up the process.

n	shuffle2d (s)	unshuffle2d (s)
10	0.1	0.02
50	1.1	0.2
100	5.5	0.9

Table 3.15: Performance (running time in seconds) of permutation based pre- and post-processing. The values for shuffling and unshuffling are different because the time for distributed permutation matrix generation is included in the first one.

3.4.3 Public Verifiability

As discussed earlier, a solver for the assignment problem should not only protect the privacy of the inputs, but also give formal guarantees about the correctness of the result. That is, the computation result should also come with a cryptographic certificate (or proof) that allows any party to check whether all computations have been carried out correctly, without leaking any information about the inputs. Such an approach, called *publicly verifiable MPC*, minimizes the trust that needs to be put into the MPC network, as soundness can even be guaranteed in case that all MPC nodes get corrupted. To achieve this, we deploy non-interactive zero-knowledge proofs of knowledge (NIZK) [241].

The idea for this type of optimization was introduced in [225]. The approach is based on the duality theorem which defines for every linear program an equivalent problem in the dual with dual variables and a dual objective function. In fact, the efficient algorithms presented before already make use of the duality internally, which make them also good candidates for extension with verifiability. By directly proving the optimality of the dual solution, it is no longer necessary to prove the correctness of every single computational step, which would not be feasible in a reasonable amount of time.

The dual of the linear assignment problem in (3.2) is as follows:

$$\begin{aligned} &\text{maximize} && \sum u_i + \sum v_j \\ &\text{subject to} && u_i + v_j \leq c_{ij} \text{ for } i,j \in A,T, \\ &\text{with} && u_i, v_j \in \mathbb{Z} \text{ for } i,j \in A,T. \end{aligned} \tag{3.3}$$

Here u_i and v_i are the dual variables which an not restricted to positive values contrary to the primal variables. The dual variables are also interpreted as shadow prices which is why operations in the dual are often called auctions.

Thus, in order to efficiently prove the correctness of a optimization result we first need to compute the corresponding dual, which has to be kept private. This can be achieved by augmenting the MPC algorithms to also compute the dual in the oblivious domain and only reveal the primal solution in the clear. Secondly, in order to prove optimality, it is necessary to show that:

1. the optimum is indeed the sum of the costs (NIZK or directly),
2. the constraints of the primal are fulfilled (this can be done in clear),
3. there exist dual variables such that the primal optimum is equal to the dual, i.e. $\sum u + \sum v = f_{opt}$ (NIZK), and
4. the dual variables fulfil the constraints (NIZK).

The most challenging task are the last two steps which have to be done without revealing the dual variables or the costs which requires the usage of NIZK and the generation of them within the MPC system, without revealing any witness to any entity in the clear. In the following we explain how this can be achieved efficiently using MPC and NIZK.

3.4.3.1 Augmented LSAP

In a first step, the optimization algorithms have to be adapted to also provide the dual solution. We use the idea of augmented algorithms which provide both, an optimal solution of the original problem as well as the corresponding dual variables.

Verifiability for general LP. For the case of LP the approach has already been demonstrated in [225], and we use this implementation as a baseline for our improvements. The respective solution also comes with verifiability, and consists of four main steps:

1. the basic simplex operation,
2. computation of the solution,
3. computation of the dual, and
4. verification of the dual and optimality.

Unfortunately, especially the verification step turned out to be very resource intensive. Besides adding another 20% of overhead to the computation time, especially the memory usage was extensive. As shown in Table 3.11, we were not able to conduct tests beyond problem sizes of 40 slots on our test machine, rendering the implementation impractical for our requirements.

Overcoming limitations for LSAP. In our work we thus developed and tested an augmented version of the Hungarian algorithm, as it also follows a primal-dual approach.

The core extensions to the Hungarian algorithm are shown in Listing 3.4 and 3.5.

```
for i in range(n):
    # Find min value for each row
    minval = min(cost_matrix[i])
    # Subtract minval from every element in the row.
    for j in range(n):
        self.C[i][j] -= minval
    # Update dual u_i
    self.u[i] = minval
```

Listing 3.4: Augmented Munkres Step 3

```
for i in range(self.n):
    for j in range(self.n):
        if self.row_covered[i]:
            self.C[i][j] += minval
            if j == 0: self.u[i] -= minval
            events += 1
        if not self.col_covered[j]:
            self.C[i][j] -= minval
            if i == 0: self.v[j] += minval
            events += 1
        if self.row_covered[i] and not self.col_covered[j]:
            events -= 2 # change reversed, no real difference
```

Listing 3.5: Augmented Munkres Step 6

In step 3 of the algorithm the u_i have to be updated with each row modification and in step 6 u_i and v_j are updated according row and column modifications. All other steps are not affected and the necessary modifications in steps 3 and 6 are also very MPC friendly, i.e. only addition and subtraction on secure values.

Because shortest augmenting path (SAP) algorithms are very similar in nature to Hungarian, we expect similar results for them although we did not implement them.

3.4.3.2 Adaptive zkSNARKS

We now explain how the necessary NIZKs to prove the optimality of the dual solution are computed in our system, thereby significantly improving over the efficiency achieved by [225].

In our first approach we use the adaptive zkSNARKs by [205], which are well suited and optimal in terms of proof size. The idea is to have commitments on all relevant witnesses for the proof, which could, e.g., be stored on a blockchain to make them publicly available.

As described above, the proof is composed of four components where three have to be shown in zero-knowledge and the evident plaintext constraints are omitted. Because the commitment used in the zkSNARKs system are homomorphic, the correctness of the optimum can be shown directly by combining the respective commitments on the input weights. However, if the more efficient vector commitments are used a dedicated proof has to be computed explicitely in the MPC system, which is also straight forward. The same has to be done for the dual solution which also has to sum up to the optimum with no slack space left to the primal.

The most challenging task is proving the dual constraints. Basically, we have to prove n^2 inequalities on the dual variables. If we consider the fact that the slack for cost incorporated in the final solution is zero, we can convert n inequalities to equalities, however, because of the use of vector commitments they also have to be integrated into the same proof and cannot be done at the verifier.

To measure performance we implemented a version based on PySnark[11] and QapTools[12] which is shown in Listing 3.6. It is a fully privacy-preserving version with index and optimum also hidden (inside a commitment), and proves the optimum of the primal and the dual as well as all dual constraints in a single proof.

```
@pysnark.runtime.snark
def is_optimum(C, ind, opt, u, v):
    n = len(C)
    res = 1

    # verify primal optimum is correct
    cost = 0
    for r, c in ind:
        cost += C[r.value][c.value]
    res *= opt == cost

    # verify optimum primal-dual
    res *= opt == (sum(u) + sum(v))

    # verify dual constraints
    for i in range(n):
        for j in range(n):
            z_ij = u[i] + v[j]
            res *= z_ij <= C[i][j]
    return res
```

Listing 3.6: zkSNARK example for optimality proof.

The achieved performance is shown in Table 3.16. From the figures it can be seen that the proof computation is the only relevant factor but is still faster than the optimization process. It should be noted that the proof computation is independent of the

[11] `https://github.com/meilof/pysnark`, accessed 2023-01-10.
[12] `https://github.com/Charterhouse/qaptools`, accessed 2023-01-10.

network latency and does not need any communication between MPC nodes except for a final reconstruction step.

n	genprog (s)	prove (s)	verify (s)
10	0.03	1.9	0.07
20	0.10	3.5	0.07
30	0.22	5.7	0.07
40	0.39	9.4	0.11
50	0.61	18	0.21
60	0.89	21	0.21
70	1.23	37	0.40
80	1.71	42	0.42
90	2.15	65	0.40
100	2.45	72	0.75

Table 3.16: Runtime in seconds for public verifiability of MPC based LSAP solving with adaptive zkSNARKs. The table shows results for different problem sizes n.

3.4.3.3 NIZK Without CRS

Despite their practicality, zkSNARKs require a common reference string (CRS), which needs to be computed in a setup phase. Although this CRS can also be generated in distributed way (e.g., in an MPC ceremony) in order to ensure that no entity knows, e.g., any trapdoor information of the CRS, it is sometimes undesirable to require a setup phase, in particular as the purpose of the NIZK is to protect against malicious MPC nodes, and thus an independent MPC network would be required for the MPC ceremony.

To also support a method without a CRS we leverage Bulletproofs [242], which were designed to support efficient range proofs, the most demanding step when proving (3.3). When batching u interval proofs for intervals of bitlength v, the resulting proof size is only $2(\log_2(u)+\log_2(v))+4$ group elements plus 5 $\mathbb{Z}_p$ elements.

Table 3.17 shows proof generation and verification times for different batch sizes and problem sizes. The major issue at the moment are the size of the input commitments, because vector commitments are not supported and for each witness a dedicated Pedersen commitment is needed, which results in n^2+2n commitments for the weight matrix and the dual variables.

The benchmark results only consider the computationally intensive part of range proof processing, however, because the additional comparisons can be directly done at the verifier in parallel, they are good estimates for overall performance. A fully integrated implementation also supporting MPC is currently under development.

The times for MPC based optimization can be summed with the corresponding prove times for the given problem size to get an overall time. Additionally, the verification

n	prove (s)	verify (s)	fast verify (s)	proof size (elem.)
4	0.26	0.14	0.1	20
5	0.25	0.27	0.2	22
8	1.04	0.54	0.4	24
11	2.07	1.09	0.81	26
16	4.14	2.17	1.61	28
22	8.28	4.33	3.22	30
32	16.5	8.65	6.45	32
45	33.0	17.2	12.8	34
64	66.2	34.5	25.7	36
90	132	68.8	51.5	38
128	267	138	102	40

Table 3.17: Runtime and size for optimality proofs based on Bulletproofs. Results for different problem sizes n are shown.

times is only done by the results parties. Moreover, in some use cases the verifiability part could be done offline to further speed up overall application performance.

3.5 Discussion

In this section we explored ways to use MPC in decentralized data markets for two relevant use cases. To do so, we first compared and benchmarked two generic MPC frameworks in Section 3.2, both using the same linear ITS secret-sharing protocols, but with a fundamentally different software approach. As expected, MP-SPDZ performed best in a local setup but was harder to use and manage. MPyC on the other hand, is extremely flexible and easy to use. We implemented various symmetric ciphers from the literature, some of them particularly optimized for MPC, and did extensive testing and benchmarking with both frameworks. The goal was to understand how universal and generic the available software is and to which extend they can be used without special knowledge about MPC core protocols.

From our tests we learned that even for the most versatile software frameworks available it is hard to get things right and MPC is still far from being usable for software developers not familiar with the field. Furthermore, we showed that the practical performance cannot be trivially estimated from the algorithms to be implemented by estimating the multiplicative depth in advance. It is often additional work introduced by data format conversion which significantly penalizes the speed of the algorithms and a lot of manual optimization is needed to get optimal performance.

It was also unexpected, that the specifically developed algorithms optimized for lower number of multiplications did not perform as expected, i.e., substantially better than AES, and also some stream ciphers turned our to be not well suited for ITS-MPC. We found Trivium being the best cipher for platform independent application in MPC and we recommend it for applications where the lower security parameter is not an issue. A very interesting open question is if the Trivium approach can be scaled up to generate more than 64 (and maybe even arbitrarily many) keystream bits per round of communication without weakening security or exponentially increasing the size of the internal state. Recently, [192] proposed Kreyvium as an extension of Trivium, but focused on improving security without increasing multiplicative complexity.

Additionally, we did intensive benchmarking for various scenarios. Contrary to most existing literature we were also addressing non-optimal network settings. In our tests we found, that although MP-SPD performs by far better in high throughput low latency settings, it gets surprisingly outperformed by MPyC in scenarios with higher network latency. The asynchronous architecture of MPyC achieves more efficient use of the network layer in those scenarios and could even compensate for the optimizing compiler used by MP-SPDZ. However, for the scalability in the number of parties we found the opposite, here MP-SPDZ behaved as expected and MPyC seem to experience significant slowdowns. 8Generally—independent of the framework—ITS-MPC is not very scalable in the number of nodes and only small number of nodes are realistic if advanced functions must be computed.

In Section 3.3 we explored ways to leverage MPC in more complex use cases, specifically building a market platform for smart manufacturing. We showed that simple cryptographically secured sealed bid auctions are not suitable for complex application scenarios like markets places in the manufacturing domain and MPC can deal with more advanced mechanisms. From the requirements established together with relevant stakeholders, we identified many challenging and partially contradicting objectives which motivated the design of a new architecture and framework. The framework enables secure and privacy-preserving price finding for outsourcing tasks in the production industry, but also beyond [19]. For increased trustworthiness it further enables every participant to publicly verify all steps in the auction in a privacy-friendly way. This was achieved by extending secure multiparty computation with zero-knowledge proofs of knowledge in a seamless way for the designer of the application. To assess the practical performance, we implemented a proof-of-concept and tested various scenarios. As our main result we were able to show that many requirements given could be achieved with our approach in a single framework and with practical performance. Furthermore, we also showed that the proposed framework is suitable to realize complex use cases in a proof-of-concept implementation.

Finally, in Section 3.4 we researched the possibility to go beyond auctions and do full optimization. From our previous experience we knew that it is difficult to estimate MPC performance for more complex functions without empirical analysis and manual optimization. This was especially true for solving the linear assignment problem in a privacy preserving but verifiable manner, as encountered in a use case from air traffic management. Therefore, we did a comprehensive analysis of solving the LSAP with MPC. We were able to improve by a factor of 50 compared to the existing simplex-based approach and also showed that shortest augmenting path algorithms are the fastest solvers in the MPC setting. On top of privacy, we also demonstrated that efficient zero-knowledge proof generation is possible by selecting most appropriate NIZK frameworks. Compared to the existing approach based on Schnorr proofs the usage of modern NIZK techniques showed major speedups and practical relevance. Furthermore, the platform still provides information-theoretic privacy, because the integrated NIZK rely on perfectly hiding commitments and protocols with perfect zero-knowledge. However, their soundness property is not quantum-safe. Thus, future work will be on protocols with soundness properties based on intractable problems for quantum computers. Nevertheless, the systems can be safely used until scalable quantum computers are available, because privacy is guaranteed in the long term and cannot be broken retrospectively.

Chapter 4

Securing Data in Transit

4.1 Introduction

To achieve information-theoretic security for message transmission, i.e., to establish secure channels, adequate key agreement protocols are required. It is known that secure key exchange can only be achieved with asymmetric cryptography, which is susceptible to quantum computer attacks and can never be realized with information-theoretic security. However, with quantum key distribution (QKD) a new technology is on the horizon to break the limitations of asymmetric cryptography and provide information-theoretically secure key exchange, the ingredient necessary to achieve ITS secure communication.

QKD is fundamentally different in many ways to existing approaches. In this chapter we present our research results achieved by addressing some aspects in the use of the novel technology. Although, QKD seems very appealing at first sight to replace existing key exchange protocols, the technology suffers from many shortcomings and limitations compared to an algorithmic approach. In this thesis we discuss aspects researched to make QKD more practical and efficient. We also discuss how we integrated QKD with technologies from previous chapters.

Quantum key distribution is a new technology for key exchange which is unconditionally secure and thus is also quantum-safe. It was first proposed by Bennett and Brassard in [243] and its' security is governed by and can be reduced to the laws of quantum mechanics, a well-established theory. Thus, it relies on a very strong security assumption. QKD is used to agree on a key between two peers—typically called Alice and Bob—but not to directly transmit information, however, when combined with perfectly secure one-time-pad encryption it results in a perfectly confidential and authentic communication channel.

Since its' first proposal in 1984, QKD has attracted a lot of research from different domains ranging from theoretical physics, experimental physics, optics, quantum optics, electrical engineering to computer science and cryptography. QKD is a hardware-based technology and fundamentally different to solutions we use in cryptography today, which are based on algorithmic concepts. That is one factor, why its' introduction and commercialization was hampered in the past.

However, research in QKD made a lot of progress and culminated in the realization of first QKD networks around 2009 [53]. First commercial systems for link encryption have been presented already around 2004 and were first deployed in 2007 to protect Swiss election[1], but commercial success was hampered by the many limitations of QKD [244, 245]. Nevertheless, research continued and after a period of less activity in the field, the technology is now considered ready for large scale deployment [246, 247, 248, 115]. The topics researched in this work were encountered during discussions with industry partners in our collaborative projects and contribute to the challenges in integrating QKD.

In the following we will quickly introduce the basic QKD system at AIT used for our work and then present our improvements and findings, which were also published in [11, 12, 13, 14]. First, we show how we reduced the number of optical channels required by researching an algorithmic synchronization mechanism called pre-sifting in Section 4.2. By independently registering photon detection times at both peers with high precision, we were able to remove the dedicated synchronization channel used by the AIT-QKD system. After that we investigate another practical aspect for post-processing in Section 4.3. We study the possibility to offload computationally intensive tasks from the device to external and potentially untrusted hardware to improve flexibility and cost efficiency of QKD devices. During our research on pre-sifting algorithms, we also encountered a novel attack on a previously developed channel authentication scheme, which we were finally able to fully break, as presented in Section 4.4. Finally, in Section 4.5 we analyze how QKD can be reasonably combined with the other technologies presented in the previous chapters.

4.1.1 Quantum Key Distribution

Quantum key distribution (QKD) was invented almost 40 years ago and is currently a more vital field of research than ever. With commercial impact on the horizon, application of QKD is gaining substantial momentum and the technology is expected to be deployed at large scale in the upcoming years. This is true for both, terrestrial applications as well as space communication.

QKD is the only known information-theoretic primitive for key exchange and can be considered a part of the quantum-safe toolbox to build long-term secure ICT systems which resist quantum computer attacks. However, its' wide adoption is still hampered by various challenges which must be overcome to make QKD practically relevant and

[1]`https://www.newscientist.com/article/dn12786-quantum-cryptography-to-protect-swiss-election/`, accessed 2022-12-05.

facilitate commercial adoption. On the one hand, research is thus continuously improving protocols and optics/electronics to achieve better bandwidth and distance, as well as co-existence with existing infrastructure. On the other hand, miniaturization and electro-optical integration are important topics to make the technology more reliable and cost-effective.

Contrary to most other cryptographic primitives, quantum key distribution (QKD) is a key-agreement protocol which derives its' security from properties of the physical layer. It uses a quantum channel to exchange quantum information which cannot be perfectly copied or eavesdropped according to the laws of quantum mechanics. In a prepare and measure QKD protocol, so called quantum bits (qubits) are encoded and transmitted over a quantum channel. Typically, the qubits are encoded on photons and the transmission channels are either fiber optics or free space. Finally, the qubits are measured at the receiver and decoded. From the measurement of quantum bits classical information is derived and all following steps are done in the classical domain. However, due to their interaction with the environment and/or eavesdropper, photons are subject to distortion and absorption. To detect and cope with these modifications in the transmission channel, post-processing steps have to be applied in order to get the full key agreement primitive with practical correctness and secrecy.

The outstanding property of QKD is that it is an information-theoretic secure (ITS) and universally composable (UC) key agreement protocol [249] given that its classical communication is performed over an authentic channel (note that all key-agreement protocols are insecure over non-authentic channels). ITS message authentication codes based on universal hashing [250] which use pre-shared keys in the first round and QKD keys from previous rounds in later rounds, are a means to generate an ITS authentic [251] channel. Thus, QKD is a very powerful cryptographic primitive which cannot be realized with non-quantum protocols.

A basic prepare and measure scheme is shown in Figure 4.1. Alice is the sender and encodes information into the polarization of single photons. For each photon (qubit) it generates two random classical bits. The first bit is used to select one out of two bases to encode the photon, and the second bit to decide the concrete polarization between the two orthogonal choices. In essence, the two bits are used to randomly select one of four possible polarization encodings. The photon is then sent over the quantum channel to the receiving peer (Bob), which is measuring the photon in one of the two bases at random. The measurement is done with single photon detectors registering photons as classical binary information (clicks). After the qubit transmission on the quantum channel all information at both peer is fully classical (non-quantum), and the post-processing is started as explained below. This is also evident from the described behavior, Alice only remaining with the random bit string used to describing the polarization encoding of the qubits sent (2 bit per qubit), and Bob only storing the received polarization encoding within the randomly selected basis (2 bit per qubit).

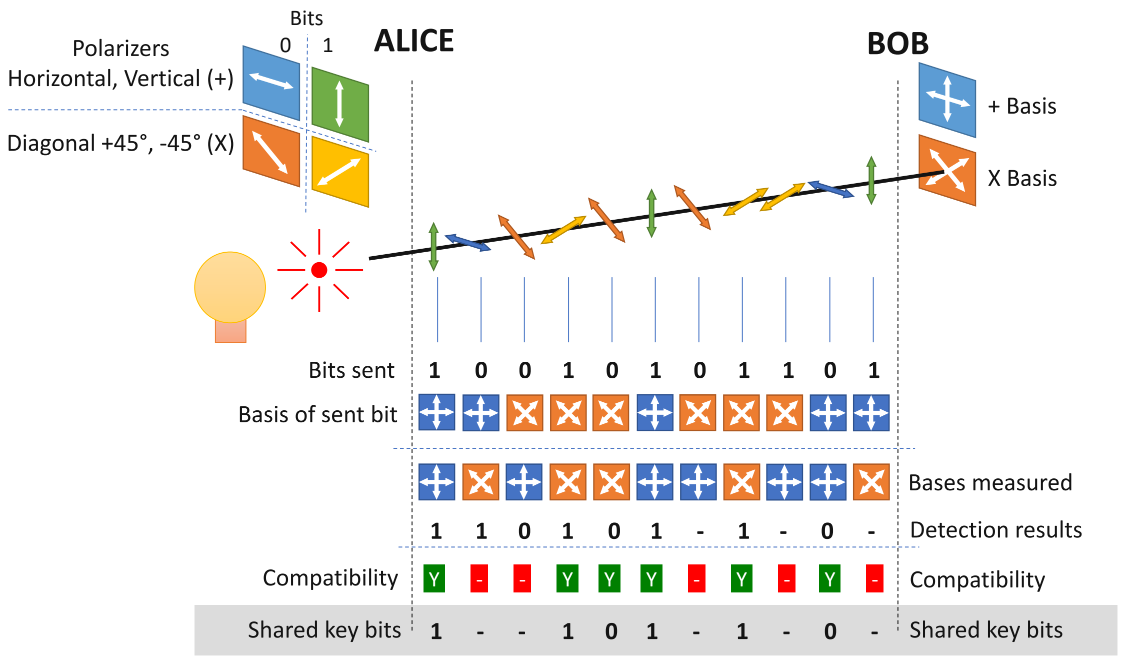


Figure 4.1: Basic functionality of BB84 protocol (adapted from [252]). Qubits are randomly encoded by Alice in one of four polarizations, corresponding to two bases, and sent over the quantum channel. Bob randomly chooses bases and measures polarization of incoming qubits. Because measurements are only correlated for Qubits measured and encoded the same basis, all other events are eliminated to extract the shared key among the peers.

4.1.2 QKD Post-Processing

In the following we quickly introduce QKD post-processing to better understand the results in this work. QKD comprises two main phases to arrive at a shared key with strong correctness and security guarantees. First, qubits are randomly generated on one side, transmitted over the optical quantum channel, and measured on the other side to generate the so-called raw key. In the second phase, a non-quantum (classical) post-processing is executed to agree on identical keys (correctness) on both ends of the transmission line, and to render useless any information a potential attacker could have learned by attacking the transmission phase (privacy/secrecy).

In detail the steps necessary to extract a secure key from the raw data of the transmitted quantum bits are as follows:

(i) In **sifting**, non-relevant information is removed from the raw key; e.g., for conjugate coding protocols, events prepared and measured in different bases would be deleted. Also, events not received by Bob are discarded in discrete-variable protocols (cf. Section 4.3.2). Discarding of empty slots can be optionally done in a dedicated pre-sifting step. (cf. Section 4.2).

(ii) After that, **error estimation** determines an upper bound on the information leakage on the quantum channel to an adversary and provides information to optimize the information reconciliation. Although more advanced methods have been proposed in the literature, this is typically done by cut-and-choose methods. Additionally, the idea of using a confirmation phase to replace error estimation was proposed by Lütkenhaus [253].

(iii) In **information reconciliation**—which often uses methods from forward error correction—errors in the remaining raw key are corrected so that sender and receiver should obtain identical keys. The classical (non-quantum) messages exchanged in this process must not leak information on the final key. Typically, the leakage is tracked and treated during the privacy amplification step.

(iv) In the **confirmation** step, the peers check that their keys are identical after the error correction step. If it fails, the parties go back to the error correction step or abort the QKD protocol.

(v) Finally, in the **privacy amplification** step, the information leaked during all protocol steps (quantum and classical) is eliminated from the final key by running a (strong) randomness extraction protocol between the peers.

All processing steps together enable Alice and Bob to agree on a final key which is ε-close to an ideal key. The post-processing steps are also visualized in Figure 4.2.

Various optimizations of the above key agreement process have been proposed in the past, either for efficiency reasons or implementation aspects, but the basic steps are typically contained in one form or another.

4.2 Timing Synchronization for QKD

One of the main problems encountered in the realization of the first QKD demonstrators at AIT, was the effort and resources needed to synchronously measure photon detection events in entanglement-based systems. The first entangled QKD prototypes developed used a dedicated time-stabilized channel to perform synchronization of the distant peers [65, 49]. Especially for fiber-based systems working at telecom wavelength the synchronization was needed to operate indium gallium arsenide (InGaAs)-detectors in gated mode, i.e., only enabling them if a photon was expected. However, with the newly developed free-running InGaAs-detectors [254] at AIT such additional synchronization was no longer necessary for detector gating and enabled the development of a free-running mode for our third generation QKD-system EPR-S405. In this section, we present our results on replacing a dedicated synchronization channel with a network based algorithmic synchronization mechanism called pre-sifting. This section also serves as an

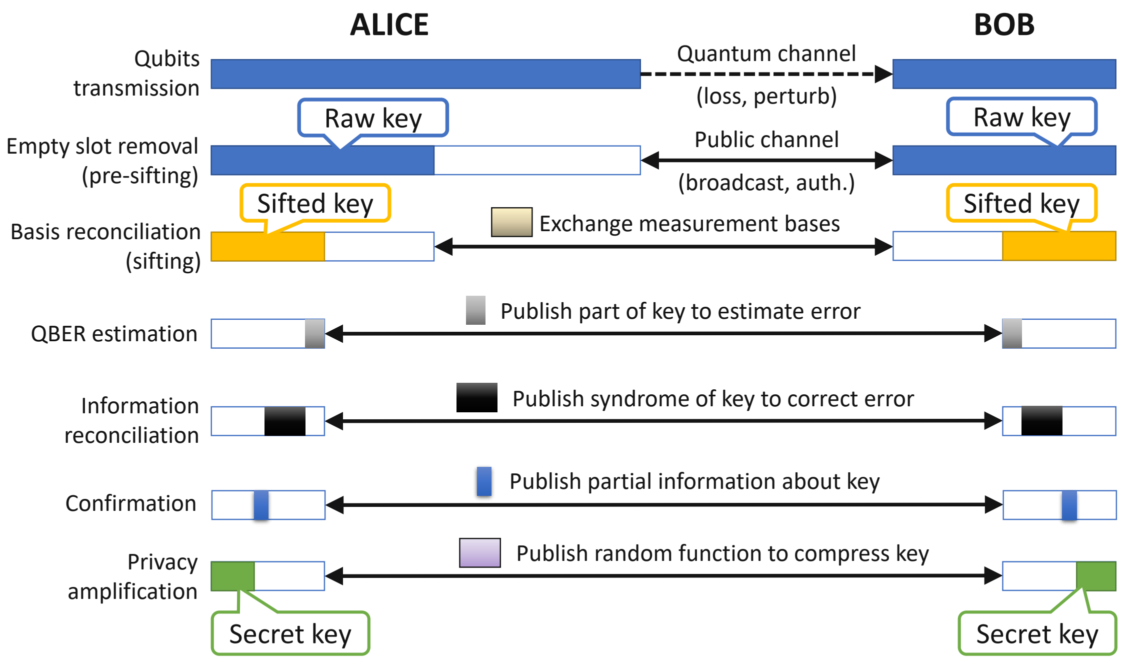

ALICE
BOB
Qubits transmission
Quantum channel
(loss, perturb)
Raw key
Raw key
Empty slot removal (pre-sifting)
Public channel
(broadcast, auth.)
Sifted key
Sifted key
Basis reconciliation (sifting)
Exchange measurement bases
QBER estimation
Publish part of key to estimate error
Information reconciliation
Publish syndrome of key to correct error
Confirmation
Publish partial information about key
Privacy amplification
Publish random function to compress key
Secret key
Secret key

Figure 4.2: Overview post-processing steps with a dedicated pre-sifting phase where lost signal is removed first. The compression rate in the last privacy amplification step depends on all information about the key in previous steps plus an addition factor dependent on the actual error rate.

introduction on QKD technology to make the reader familiar with the topic, which is substantially different to the algorithmic approaches used in the previous chapters.

In Figure 4.3 the EPR-S405 is shown. It is a fully functional QKD-system capable to use both, free-space or fiber channels to distribute photons to the communication partners. It integrates all the optics and embedded electronics to generate and distribute entangled photon pairs, measure and process them to extract a secure shared keys between the two devices.

The goal of our research was to develop a self-synchronizing acquisition system for entanglement-based protocols which do not need a dedicated synchronization channel or precise and stable atomic reference clocks. This requirement also prohibited the use of external reference systems like the global positioning system (GPS).

To this end, we successfully exchanged the synchronization channel by a pure software solution and which in fact leverages the photon pair distribution to eliminate offset and drift between free running clocks in a extremely efficient way. The novel software module introduces a pre-sifting step in the post-processing stack, which is handling the synchronization. Contrary to the intuition and previous proposals, we show that time bases algorithms perform better than frequency-based solutions and are able to provide real-time synchronization for high rates even on embedded systems with limited computing capabilities.

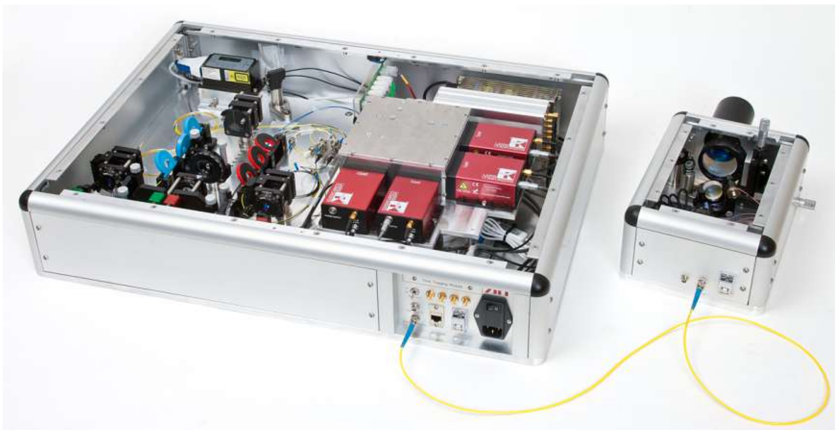


Figure 4.3: AIT-EPR-S405 QKD lab prototype used for research and teaching.

Related work. The possibility of synchronizing two local clocks by exploiting correlation information had already been demonstrated, however, the proposed solutions did not yet meet our needs. AIT already provided special correlation finding algorithms for preliminary satellite communication scenarios [255], which are optimized for low signal rates, high offset search ranges and large block sizes. Another approach from [256] uses Fast Fourier Transform (FFT) based convolution algorithms for offset finding, which are rate independent. Nevertheless, to find the offset with high accuracy, the limitation in the block size requires high rates for signal locking. Our current approach overcomes these limitations.

4.2.1 System Overview

As shown in overview Figure 4.4, the entangled photon pairs are generated in a spontaneous parametric down conversion (SPDC) source, which is basically a nonlinear barium borate (BBO) crystal pumped with a strong continuous laser. The entangled photons generated in the BBO crystal are then distributed to the receivers at Alice and Bob. To this end it is different to the standard BB84 protocol for prepare and measure, however, if the detection of one entangled photon is done locally at Alice as in our implementation, it resembles a BB84 protocol where the random encoding of the photon sent is done optically.

The photons are analyzed in polarization modules (BB84-module) in the receiver which has one output for each polarization state to be detected by single photon detectors (Si-SPADs). All detection events are finally registered with a dedicated timestamp in a time-tagging module (TTM8) which uses its own local clock. Thus, the two devices are free running with respect to their local clocks and can only communicate over a network connection used to exchange information in the post-processing phase and for management purposes. We used the network connection to also synchronize the embedded computers which are part of the devices to achieve a first rough common reference

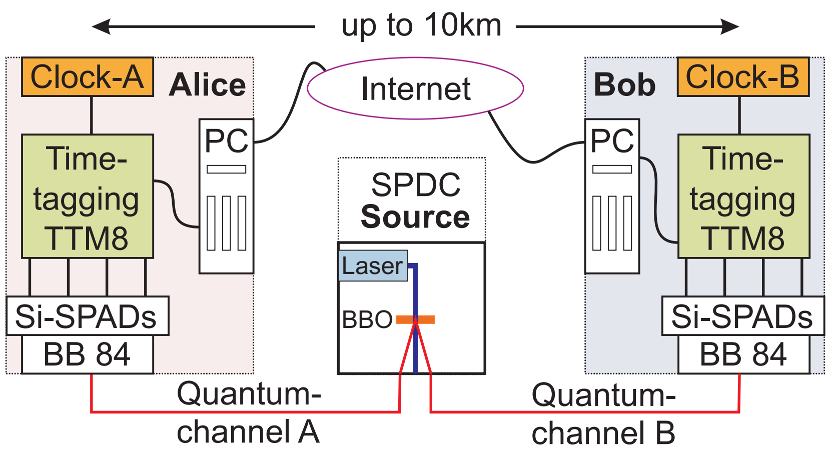


Figure 4.4: Schematic overview of AIT QKD system. SPDC is source for entangled photons. BB84 modules are passive polarization separators and Si-SPADs are the single photon detectors. Events are then registered at TTM8 with a resolution of about 100ps.

time. By using the precision time protocol (PTP) over a direct Ethernet connection we were able to synchronize the PC-clocks within 1ms. This served as the starting position for our search algorithm which was required to further synchronize the clock down to sub-nanosecond level continuously.

The acquisition system was integrated into the AIT-EPR-S405 demonstration system [257] as part of the post-processing software stack [44]. The system implements the BBM92 [258] protocol for entanglement based QKD, a variant of the original BB84 protocol [243]. In this protocol, the correlated measurement results of a shared entangled state are used to generate a secure key, as follows: Alice keeps track of all her detection events together with the measured polarization states; Bob communicates to Alice which of the synchronization triggers resulted in a detection event on his side and also communicates his measurement basis; Alice deletes all entries in which Bob did not detect a photon and where the bases did not match. Both parties share then a sifted key, which is further processed through the QKD protocol stack to yield a secure key.

However, to align events on both sides without a common clock the newly developed pre-sifting algorithm is run at Alice to calculate the time and frequency offset between the two local clocks. Bob sends the timetags of all received events to bob but without any additional information to not compromise the security. With all timetags available Alice is able to calculate the exact time and frequency offset of the local clocks in the so-called pre-sifting step which is then used to correlate the events on both peers. After this filtering the standard QKD-protocol stack is executed to generate the secure key shared between Alice and Bob.

In Figure 4.5, the times and distribution of the photon arrival at the device of Alice are shown, whereby the pair-photons are indicated by a flag. The counterparts are detected at Bob's site, but a difference in clock frequency causes a relative drift of all measured events.

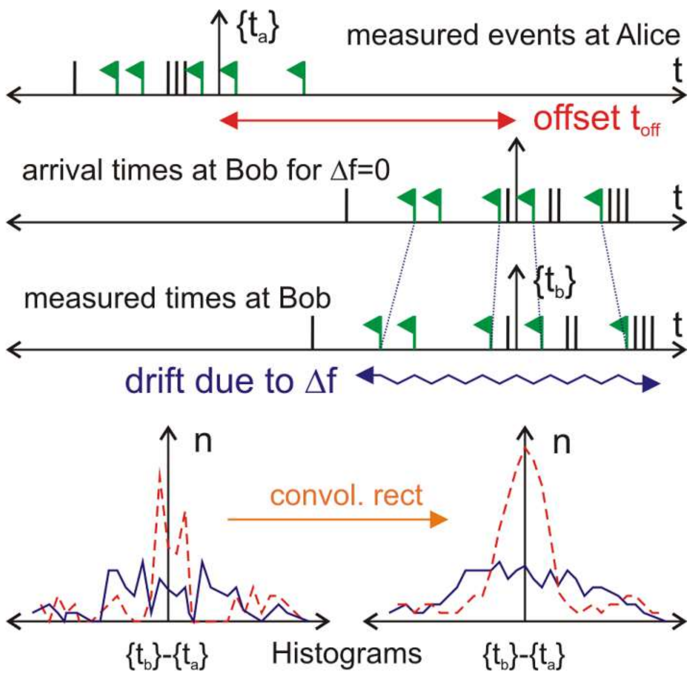


Figure 4.5: Measurement timing at both peers with offset and jitter.

The QKD software stack is located on two embedded computers, one connected to Alice and the other to Bob. All classical communication between them is routed over a standard network connection and the AIT QKD stack [44] war used to generate the keys. Alternatively, the QKD stack could run directly on the embedded CPU for a single-chip quantum cryptography solution [58]. QKD post-processing is known to be computationally intensive, and the concrete implementation was limited to key rates of 1 kbit/s due to the limitations of the embedded CPU. To impede a man-in-the-middle attack, all communication between Alice and Bob is authenticated by applying secure message authentication in the form of an ITS MAC [250].

4.2.2 A Novel Pre-Sifting Method

From a data acquisition point of view the scheme is very similar to BB84, because the entanglement source is integrated with the Alice device and the random measurement of Alice' photon determines the state of the other photon which is transmitted to Bob. The main difference is that the random polarization encoding is selected passively via optical elements and the photons are generated spontaneously, i.e., they follow an exponential distribution, and are not synchronously generated with a fixed clock. The goal of the pre-sifting module is timing synchronization and alignment, i.e., to enable a stable synchronization to identify photons from the same entangled pair. To do so, some preconditions have to be fulfilled on the hardware level. The photon source has its own source stabilization which guarantees stable production and coupling of photons into the quantum channel. The state alignment does automated alignment of the entangled state of photons and additional polarization alignment compensates polarization drifts in the quantum channel.

One of the main challenges in our approach was to cope with the relatively large clock drift of cheap commercial-of-the-shelf crystal oscillators as clocks. No expensive atomic clocks or any GPS based synchronization mechanisms as used in previous works were available. Thus, the pre-sifting algorithm has to cope with short term drift due to frequency offset between the clocks and also long-term drift due to other imperfections, e.g., power stability or aging. In fact, the drift (difference in frequency) in crystal oscillators is continuously varying within substantial bounds also based on environmental conditions and susceptible to fluctuations. The only stability requirements we required for the local clocks were, that frequency differences could be approximated as constant within intervals up to a second, as discussed later. The measured time drift for the given clocks over time is shown in Figure 4.6 and shows significant variation.

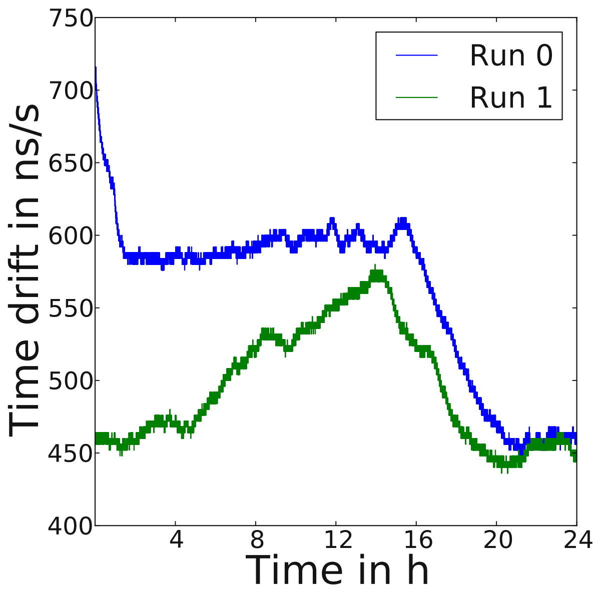


Figure 4.6: Measurement of clock drift between free running clocks at Alice and Bob. The same signal is fed into the two modules and the change in time offset is measured each second. A measured difference in time after one second of 1 ns/s corresponds to a frequency offset $\Delta f = 10^{-9}$ 1/s.

The time domain algorithm developed is based on a multi-step process to compensate for the different effects in a structured way. It assumes two coarsely synchronized peers within 1ms and implements a stateful signal tracking mechanism to optimize resource usage. The corresponding state diagram is shown in Figure 4.7 and it operates on block level, where each block contains all measurement events within $t_b = 250$ms in our case. In a first step, a potential signal is searched based on a dedicated signal-to-noise

(SNR) measure as introduced in Equation (4.1), which is set to a relatively low threshold. The used coincidence window is broad and frequency offset between clocks is not considered, thus, it works in a *coarse mode* corresponding to states 0 and 1. Looking at the transition between the states, it is clear that the states also represent the number of consecutive blocks with a tracked signal. If the weak signal is confirmed by a second block, a more detailed search can be used in a so called *finesync mode* (state 2 and 3) including parameter estimation for time and frequency offset as well as optimal coincidence window settings. The system then remains in state 3 as long a signal is locked, which is defined by a good SNR together with an accurate prediction of current offset, indicating a valid time offset and drift estimation in previous blocks. The algorithm significantly reduces search space in finesync mode and continuously update parameter estimations on consecutive blocks.

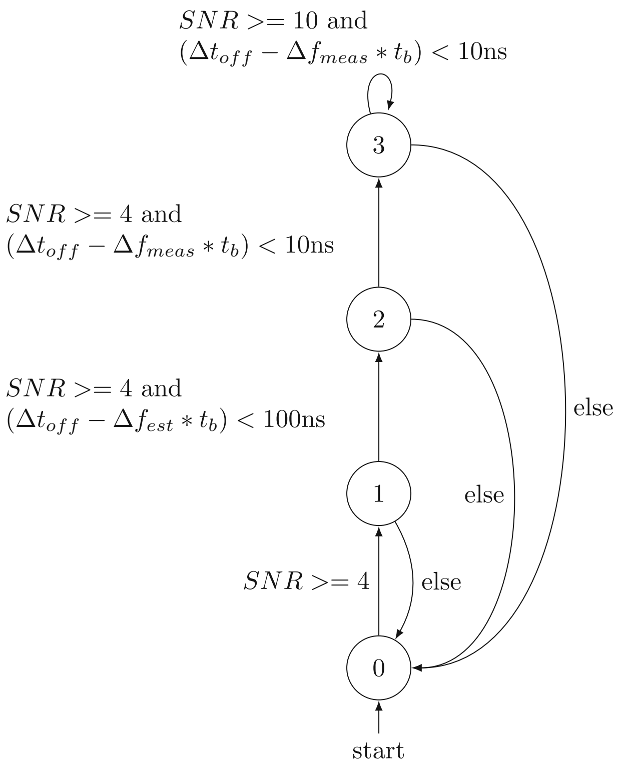

$SNR >= 10$ and
$(\Delta t_{off} - \Delta f_{meas} * t_b) < 10$ns
3
$SNR >= 4$ and
$(\Delta t_{off} - \Delta f_{meas} * t_b) < 10$ns
2
$SNR >= 4$ and
$(\Delta t_{off} - \Delta f_{est} * t_b) < 100$ns
else
1
else
$SNR >= 4$
else
0
start

Figure 4.7: State machine for drift correction algorithm developed. After each block a state transition occurs. Δt_{off} is the change in offset measured between two blocks. Δf_{est} and Δf_{meas} are the estimated and measured drift (frequency offset). t_b is the length of a block in seconds, i.e., 0.25s in our case.

The algorithm itself is working in the time domain and capable to process data in real-time. It is executed at one peer after both measured events of a corresponding block and the remote peer sent the extracted timetags. For the alignment of events at this level the local PC time is used which is synchronized via PTP over the network. I is also used to compute a first estimate of coarse offset t_{off} after cold start (state 0). The first estimate for the frequency drift Δf is taken from previous calibration measurements.

4.2.3 Time Domain Algorithm

After all input data (events) are available the main time domain search is started which works as follows. The registered events at Alice and Bob are represented as ordered sets $\{t_a\}$ and $\{t_b\}$ and their sizes n_a, n_b can be estimated by assuming a Poisson distribution with rates r_a and r_b being the mean number of arrivals per time interval. The number of valid coincidences per second is given by r_c. The correlation algorithm works with blocks of 250ms and searches for the real offset as correction Δt_{off} of an estimated offset t_{off} in the vicinity of $\pm t_{search}$ as follows:

- all events in $\{t_b\}$ are offset by t_{off} and scaled according to $(t_{b,i} + (t_{b,n_b} - t_{b,1})/2)(1 + \Delta f)$ to bring them to the same time base as Alice' events, the corrected timetags are $\{t'_b\}$;
- for all timetags in $\{t_a\}$ the differences to corrected timetags in $\{t'_b\}$ within $t_{a,i} \pm t_{search}$ are computed and stored as histogram with full resolution bin size, which is about 100 ps in our setup with TTM8, and the better t_{off} is known, the smaller t_{search} can be, leading to very efficient search in higher states, e.g., a factor of 1000 in our case with a switch from 1ms to 1μs;
- the histogram data are then convoluted with different *rect* functions representing different window sizes and ranging from 20 ns down to ns;
- the window which delivers the best *SNR* is selected and the corresponding events are communicated to Bob.

The algorithm turned out to be very efficient for our system configuration. However, the design of a meaningful *SNR* to adaptively find optimal settings for time windows based on histogram data was challenging but essential to achieve high QKD key rates. After a thorough empirical analysis we choose the following definition in our work which delivered very favorable results.

$$SNR = \frac{\frac{n_{corr}}{t_{win}}}{\frac{n_{hist} - n_{corr}}{2t_{search} - t_{win}}} - 1 \tag{4.1}$$

Where n_{hist} is the number of events contained in in the histogram, n_{corr} is the number of correlations found for a given coincidence window t_{win}.

Additionally, through the use of a stateful algorithm we are able to improve the estimations for time and frequency offset t_{off} and Δf after a signal is found which makes the processing very efficient. In state 3 the time and drift are not only corrected but the frequency offset Δf itself is updated based on the time offset found to the last block. Thus, by averaging over the last few blocks and updating Δf, the uncertainty in time offset and drift estimation can be dramatically reduced which helps to reduce search space. Additionally, with access to precise frequency offset it is also possible to narrow down coincidence windows leading to higher SNR and ultimately higher QKD key rates.

4.2.4 Results

With the proposed approach we were able to achieve very favorable results, especially in continuous operation with a tracked signal, it required only negligible computing resources compared to QKD post-processing. Also cold start operation achieved good performance with the given compute resources. Interestingly, because key generation was only triggered in state 3, the computing capacities otherwise used for QKD post-processing could be used for faster signal search in parallel during course mode, i.e., when signal lock was lost. Thus, the tasks of system synchronization and key generation ideally complement each other and make efficient use of the compute hardware in a time-sharing approach possible. In Table 4.1 we compare our approach with the different algorithms proposed in the literature.

	Algorithms		
	TD1 [255]	TD2 [this work]	FFT [256]
Throuphput	low	high	high
Offset search	high	medium	low / high *
Resolution	max	max	high / low *
Real-Time	no	yes	yes
Δf compensation	no	yes	no
Time complexity	$O(N^2 t_s t_b + N^2 t_s t_b \log(N^2 t_s t_b))$	$O(N^2 t_s t_b + \frac{t_s}{t_{res}})$	$O(\frac{t_b}{t_{res}} \log \frac{t_b}{t_{res}})$
Space complexity	$O(N^2 t_s t_b)$	$O(\frac{t_s}{t_{res}})$	$O(\frac{t_b}{t_{res}})$

Table 4.1: Comparison of algorithm properties and their complexity. The input size N is the number of events per second assumed to be equal at both peers for this comparison, t_b is the block size and t_s the search window (short for t_{search}) in seconds. t_{res} is the resolution to achieved. For complexity we look at the relevant part of timetag processing. * trade-off due to limited size of the vector for the FFT-algorithm.

Additionally to the qualitative analysis of our algorithm, we also give some concrete figures measured in the laboratory setup. For cold start in *course mode* searching for time differences of ± 0.5ms for block size of 250ms at 65 kEvents/s takes about 50 ms/block and at 500 kEvents/s it takes 200 ms/block. These values show, that real-time operation is possible even with weak signals that could be identified in our experiments.

After a signal is found in course mode the *finesync mode* is entered for the next block searching only $\pm 0.5\mu s$ around a locked signal. This adaptive approach is 1000 times faster and requires only negligible CPU time in the continuous regime, therefore freeing up CPU resources for key generation tasks.

Most interestingly, the time-based approach significantly outperformed an FFT based approach, which we tried alternatively. This is due to the fact, that we only calculate a small fraction of the convolution of the signal compared to a full convolution which have to be computed with FFT. Therefore, the benefits of reducing the search space more than compensates for the $O(n \log n)$ speedup achieved with FFT based processing. To show this behavior, in Figure 4.8 we depict the expected runtime of a FFT based convolution, for the given parameters of 2.5 ns resolution and $250ms$ block size we would need the signals encoded in vectors of size 10^8, no matter how many detection events are actually occurring.

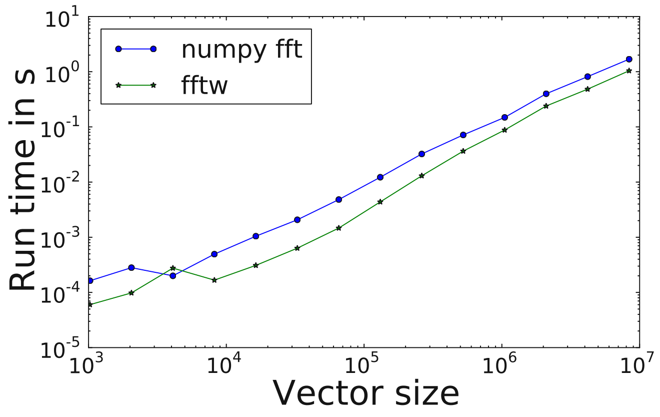


Figure 4.8: Limitations of FFT usage for increasing vector size. For our application vector sizes of $> 10^7$ are required independent on the signal strength and event rate, which makes real-time processing not possible and also means fully CPU usage for synchronization only for the given platform, i.e., an additional processing power would be needed for key generation.

Furthermore, in Figure 4.9 we show that the state based drift compensation works as expected and continuously updates drifts between blocks (left) and optimizes the coincidence pair rate of the QKD-system (right). The first measurement plot on the left is over one hour of offset correction from one block to the next (called fine-offset) clearly indicates the capability of our algorithm to continuously lock to a signal and correct for offset t_{off} and drift Δf in real-time. The adaptive selection of a fine-offset around 0 together with small coincidence window show the optimal operation of the system. The histogram on the right over the measured coincidence rates shows a symmetric Gaussian function with no pronounced tail towards lower rates, which would be present if the algorithm fails to synchronize the clocks accordingly.

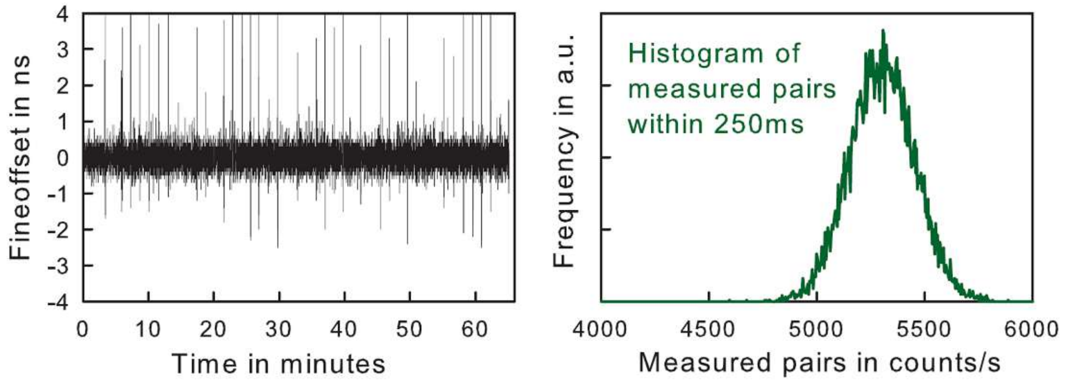


Figure 4.9: Evaluation of drift compensation mechanism. The correction of the estimated offset per block is shown on the left and the number of correlated pair rate per block is shown on the right. Both graph indicate proper operation of the new algorithm.

4.3 Offload Post-Processing from QKD Nodes

After introducing a pre-sifting phase to save a dedicated synchronization channel in QKD and to enable free running clocks in entanglement based QKD, we looked at other possible means to make QKD more practical and flexible. One particular aspect of QKD, which seemed to prevent from deployment of the technology in access networks, was the complexity and size of the compute hardware needed to process the key material. The high-performance hardware needed for post-processing is also a major factor in energy consumption and system cost for QKD devices. Therefore, we looked at ways to offload post-processing from devices or hardware sections where sensitive key material is handled to simplify the development and operational aspects of QKD.

Related work. A huge body of research exists on QKD post-processing and improving the efficiency or throughput of the post-processing phase is still an interesting challenge. Research is focused on algorithmic improvements to reduce computational effort (c.f. [259, 260, 261, 14]), on extending the local computational resources with special hardware for high-performance computing and or graphics processing units [44, 262, 263], and on developing dedicated hardware designs in field programmable hardware designs as co-processing units and local deployments [264, 265].

In contrast to current efforts in making the equipment more cost-effective, our work focuses on the possibility to offload (outsource) computationally intensive tasks in the QKD post-processing phase to external infrastructure without compromising the overall security. Being able to outsource these tasks to external data centers allows for simpler and less power-hungry devices in the field and ultimately results in more versatile QKD devices.

Contributions. In this section, we present new methods to securely offload and therefore outsource computational intense tasks of QKD post-processing. To the best of our knowledge, offloading or outsourcing QKD post-processing has not been considered yet as a means to improve efficiency and performance. To do so we combine our expertise from QKD and cryptography in order to motivate the problem and show the benefits as well as present protocols and barriers. We present and analyze protocols for outsourcing information reconciliation to external and untrusted environments therefore facilitating new application scenarios, e.g., usage in low power access networks. We furthermore discuss possibilities to outsource the privacy amplification step, which could further help to reduce the required processing power in QKD nodes. Additionally, we present use cases in order to undermine the practical relevance of the novel developed methods.

4.3.1 Motivation to Offload Post-Processing

From a computational perspective, information reconciliation is by far the most computational intense task in the stack and is typically limiting the throughput, especially in high-speed systems which set out to be used in long distance communication [266, 267]. The second most computationally demanding task is privacy amplification [262]. The rest of the protocol steps are rather simple tasks and can be executed in real-time even on embedded platform.

Therefore, we introduce and study the idea of offloading these tasks from devices by outsourcing computation to untrusted or less trusted hardware in an ITS secure way. The ability to outsource information reconciliation (IR) and potentially privacy amplification (PA) would enable new applications scenarios for both, the access network and the transmission systems. Also satellite-based QKD can become more practical if only lightweight processing resources are required on the devices itself.

The two main advantages gained by offloading processing to external hardware are increased efficiency and flexibility in the use of compute resources—also resulting in energy efficiency—and a reduced attack surface by limiting components dealing with secure key material.

The pooling/hotelling of computational resources allow for more efficient processing use of hardware. Moreover, QKD systems are deployed for long-term security and produce large CAPEX spending, i.e., they are used over a long period of time. Putting all the processing power into the devices at build time hinders later updates and prevent the operator to benefit from Moors' law. If the hardware is outsourced, it could be updated during the lifetime of the system with new technologies resulting in further optimized energy usage. Furthermore, if the hardware has not to be trustworthy and certified, cheap off-the-shelf hardware could be used. It would even be possible to completely outsource it to public cloud infrastructures in the extreme case.

Additionally, time sharing allows for further improvements. QKD is typically used in hybrid encryption protocols to establish session keys, therefore, high key rates of QKD systems are rarely needed and sharing the computational resources between links can further reduce CAPEX and OPEX cost.

Hence, putting the computational expensive tasks into efficient data centers which do not even need to be trusted is very desirable. It allows for hardware updates and joint management of all QKD workloads in the field with continuous upgrading probabilities, which is especially favorable for operators of QKD networks. Because the communication overhead is minimal compared to the computational one, a clear advantage in terms of energy efficiency arises and the gained flexibility in managing tasks is very advantageous.

Furthermore, the technology could also be used within a system. Treating parts of the system as untrusted could eventually provide the possibility for system updates without compromising system certification and help to reduce OPEX cost during system lifetime.

4.3.2 Outsourcing Information Reconciliation

As mentioned in Section 4.1.2, information reconciliation (IR) is the most demanding task in post-processing of QKD, independent of the protocols being used on the quantum level. Error correction is computationally intense, because of high error rates encountered in combination with constraints on the amount of information disclosed during error correction. The information revealed during the public discussion must be kept as short as possible to maximize overall system performance, ideally IR works close to the Shannon limit. If keys have to be processed in real-time, error correction is the bottleneck of post-processing, and can introduce substantial problems in resource constraint environments.

On a quantum level, QKD protocols can be divided into two classes–*discrete-variable* (DV) and *continuous-variable* (CV) QKD–which also result in different requirements on information reconciliation. In DV-QKD protocols, e.g., BB84 [243], qubits are measured by single photon detectors. Due to channel attenuation and non-perfect detectors the rate of detected photons is typically orders of magnitude lower than the rate of prepared photons. Consequently, in DV-QKD, IR schemes must typically provide the possibility to operate on raw key rates in the order of Kilobit [268] up to Megabit per second [266]. In CV-QKD systems signals are only perturbed but not lost through channel effects, resulting in very high raw-key rates but also high error rates compared to discrete variable system.

Additionally, two basic types of IR protocols are distinguished in QKD systems. On one hand, interactive protocols have been developed for highly efficient correction capabilities near the Shannon limit, with CASCADE [269, 259, 270] being its most prominent representative. On the other hand, forward error correcting schemes have been adopted and developed further to be used in operational regimes encountered in QKD [271]. IR based on low-density parity-check codes (LDPC) is currently the most efficient representative in this category and used in many prototype systems [272]. Although interactive two-way information reconciliation can come with smaller leakage than any one-way protocol, due to the interactive nature their practical performance is limited by the latency of the classical channel. One-way schemes have many desirable properties when it comes to realization and can easily be parallelized to increase performance. Moreover, one-way schemes are divided into two usage scenarios: direct reconciliation (DR) used with DV-QKD, and reverse reconciliation (RR) typically used in CV-QKD as discussed later.

4.3.3 Linear One-way Information Reconciliation

Before presenting our scheme for offloading protocol, we first explain one-way IR in the context of DV-QKD in more detail and informally define the concept of secure outsourcing for IR. Traditional error correcting codes consist of sets of codewords that contain redundant information. Before sending data over a noisy channel, the data is encoded into codewords. The contained redundancy can then be used by the receiver to correct the introduced errors.

One-way reconciliation–aka source coding (or compression) with side information–has been studied since the 1970s [273, 274]. While related to error correcting codes, the idea here is that the data is transmitted over a noisy channel without adding any redundancy. Rather, the source additionally sends additional compressed data over a *noiseless* channel, which can then be used by the receiver, together with the side information he has about the message, to decode the original data.

Let $\mathbf{H}$ denote the parity check matrix of a linear block code without the typically included identity sub-matrix known from traditional channel coding, i.e., only composed of the parity matrix. For IR, Alice then encodes/compresses her sifted key $\mathbf{k}_A$ by computing the corresponding syndrome as:

$$\mathbf{s}_A := \mathbf{k}_A \mathbf{H}^\top,$$

Bob has obtained a noisy version $\mathbf{k}_B = \mathbf{k}_A + \mathbf{e}$, where $\mathbf{e}$ denotes the error vector. Furthermore, Alice sends $\mathbf{s}_A$ over the noiseless classical channel. Bob decodes the key by (approximately) solving the problem of finding the vector $\hat{\mathbf{k}}_A$ which among all vectors with syndrome $\mathbf{s}_A$has the smallest Hamming distance to $\mathbf{k}_B$.

Searching for this vector is computationally hard, and can easily be seen to be equivalent to solving the standard channel decoding problem, i.e., which for a given code requires to find a codeword $\mathbf{x}$ given $\mathbf{y} = \mathbf{x} + \mathbf{e}$ for an error $\mathbf{e}$ with small weight . To see this, note that for a codeword $\mathbf{x}\mathbf{H}^\top = \mathbf{0}$ holds. Obviously, an information reconciliation decoder that works for arbitrary syndromes will find a solution for the all-zero syndrome, i.e. it will solve the channel coding problem. Vice versa, consider a channel decoder that corrects a noisy version $\mathbf{y} = \mathbf{x} + \mathbf{e}$ to $\mathbf{x}$. Effectively, due to the linearity, channel decoders make only use of the syndrome $\mathbf{y}\mathbf{H}^\top$ and not directly of $\mathbf{y}$ to find $\mathbf{e}$. Thus they solve the problem to find $\mathbf{e}$ given the syndrome $\mathbf{y}\mathbf{H}^\top = \mathbf{e}\mathbf{H}^\top$. Consequently, they can find $\mathbf{e}$ given $\mathbf{k}_B$ and $\mathbf{s}_A$, because in that case $\mathbf{e}\mathbf{H}^\top = \mathbf{k}_B\mathbf{H}^\top - \mathbf{k}_A\mathbf{H}^\top = \mathbf{k}_B\mathbf{H}^\top - \mathbf{s}_A$. Thus linear channel decoders solve the linear information reconciliation problem.

In the context of QKD, if Alice computes the syndrome and Bob corrects erroneous bits to obtain the key of Alice, the protocol is called *direct reconciliation* (DR). Because IR in DV-QKD is considered symmetric [275], the roles of Alice and Bob can also be interchanged. However, the same is not true for CV-QKD where the direction of the protocol does matter for higher transmission rates as discussed later.

4.3.4 Protocol for offloading direct reconciliation

We present a simple scheme for remote (outsourced) information reconciliation called `REM-IR` (remote IR), which allows the computationally intense step of syndrome decoding to be outsourced to an untrusted party in a secure way. The idea is to give the error syndrome, i.e., $\mathbf{s}_\mathbf{e} = \mathbf{s}_B - \mathbf{s}_A = \mathbf{k}_B\mathbf{H}^\top - \mathbf{k}_A\mathbf{H}^\top$ to an external party, which returns the error vector $\mathbf{e}$ with minimal weight satisfying $\mathbf{s}_\mathbf{e} = \mathbf{e}\mathbf{H}^\top$.

We want to emphasize, that searching for $\mathbf{e}\mathbf{H}^\top = \mathbf{s}_\mathbf{e}$ can be done with the very same algorithms and techniques typically used locally at Bob, e.g., efficient variants as in [276] or [277]. The problem is of finding $\mathbf{e}$ with a minimum hamming weight is fully equivalent to the problem of finding a $\hat{\mathbf{k}}_B$ with minimum hamming distance to $\mathbf{k}_B$ which fulfills $\hat{\mathbf{k}}_B\mathbf{H} = \mathbf{s}_B$, which is due to the linearity of the code. As an example, if LDPC codes are used which are characterized by a sparse $\mathbf{H}$, all types of decoders can be used in the very same way to find $\mathbf{e}$ close to the zero code word as for finding $\hat{\mathbf{k}}_B$ close to $\mathbf{k}_B$. The only difference in the two ways error decoding is applied, in the latter $\hat{\mathbf{k}}_B$ is directly computed and in the fist case it is computed by $\hat{\mathbf{k}}_B = \mathbf{k}_B + \mathbf{e}$.

Informally, the protocol is secure because the information leaked by publishing $\mathbf{s}_\mathbf{e}$ and $\mathbf{e}$ additionally to $\mathbf{s}_A$, and thus also $\mathbf{s}_B = \mathbf{s}_A - \mathbf{e}$ does not increase the information of Eve about the agreed key string $\mathbf{k}_A$. The intuition behind is that just learning a bit flip vector of a largely unknown key does not increase the information about the key. Or put differently, the information leaked about the final key $\mathbf{k}_A$ by additionally learning $\mathbf{e}$ is 0, because $\mathbf{e}$ is independent of $\mathbf{k}_A$ and anyway removed from $\mathbf{k}_B$ in the IR step. The described protocol is also equivalent to interactive error decoding as introduced in CASCADE [269], which also leaks parity information and error bit locations during the public discussion. A detailed description of the protocol is shown in Figure 4.10 and the security of the protocols is proved in the following.

Theorem 4.3.1 (Security of `REM-IR`). `REM-IR` is a secure scheme for offloading direct reconciliation for DV-QKD and does not leak any additional information about the agreed key by public discussion compared to a local IR, i.e., the mutual information between Eve's key and the agreed key is the same as with local IR.

Protocol `REM-IR`:

1. Alice generates her syndrome as $\mathbf{s}_A = \mathbf{k}_A\mathbf{H}^\top$ and sends it to Bob.
2. Bob generates his syndrome as $\mathbf{s}_B = \mathbf{k}_B\mathbf{H}^\top$ and calculates the error syndrome as $\mathbf{s}_e = \mathbf{s}_B - \mathbf{s}_A$
3. Bob sends the error syndrome $\mathbf{s_e}$ to the third party
4. The third party calculates the error vector $\mathbf{e}$ corresponding to $\mathbf{s_e}$, i.e., it searches for the $\mathbf{e}$ with the minimum weight fulfilling $\mathbf{s_e} = \mathbf{eH}^\top$ and returns $\mathbf{e}$ to Bob (generic formulation of the standard error decoding problem)
5. *Optional:* Bob verifies that $\mathbf{s_e} = \mathbf{eH}^\top$ and that $\mathbf{e}$ has low weight, cf. Section 4.3.6
6. Bob calculates $\hat{\mathbf{k}}_A = \mathbf{k_B} + \mathbf{e}$

Figure 4.10: `REM-IR` Protocol

Proof. Let $\mathbf{K}_A, \mathbf{K}_B$ be n bit random variables representing correlated sifted keys at Alice and Bob, which are used as input to information reconciliation. $\mathbf{S}_A$ is the random variable representing the syndrome computed by Alice and $\mathbf{E}$ the random variable for the error introduced on n channel usages. The quantum channel between Alice and Bob is then modeled as a binary symmetric channel $BSC(e)$ with quantum bit error probability e. Let further $L_E^{IR}(\mathbf{K}|Q)$ be the additional information leaked to Eve about the agreed key $\mathbf{K}$ during the information reconciliation phase, beyond what Eve already gained during the previous steps of the key exchange.

Moreover, $H(\mathbf{K}_A) = H(\mathbf{K}_B) = n$ for uniformly random input encoding, and mutual information $I_{AB} := I(\mathbf{K}_A; \mathbf{K}_B) = n(1 - H_b(e))$ is defined by the error probability on the channel. Thus, the amount of information required to be exchanged during public discussion is $|Q| \geq H(\mathbf{K}_A|\mathbf{K}_B) = nH_b(e)$, where we assume an ideal reconciliation algorithm which works at the Shannon limit, i.e, equality holds.

Without loss of generality, we assume that Bob will correct his errors and the agreed key will be $\mathbf{k} = \mathbf{k}_A$. Note here that in DV-QKD type protocols we have $I_{AE} = I_{BE}$ [275], which makes them suitable for direct reconciliation. Now, we show that the information Eve gains about the final key during a protocol run of `REM-IR` is equal to the information leaked in local IR where it only sees $\mathbf{S}_A$. Concretely, by revealing $\mathbf{S}_e$ and therefore $\mathbf{E}$, $\mathbf{S}_B$ additionally to $\mathbf{S}_A$, the information Eve learns about the final key ($\mathbf{k}$) is described as:

$$L_E^{\texttt{REM-IR}}(\mathbf{K}|\mathbf{S}_A, \mathbf{S}_e) = L_E^{\texttt{REM-IR}}(\mathbf{K}|\mathbf{KH}^\top, \mathbf{EH}^\top) = L_E^{\texttt{REM-IR}}(\mathbf{K}|\mathbf{KH}^\top) = L_E^{\texttt{REM-IR}}(\mathbf{K}|\mathbf{S}_A)\,.$$

This is due to the fact, that the additional information Eve gains by learning $\mathbf{S}_e, \mathbf{E}$ and therefore $\mathbf{S}_B$ is only about $\mathbf{EH}^\top$ which is not correlated to the final key and also removed from $\mathbf{k}_B$ during the IR step. It can also be seen by looking at Bob's key $\mathbf{K}_B = \mathbf{K}_A + \mathbf{E}$, which is the sum of two independent random variables whereby the error term is removed by IR and do not contribute in any form to the final key string $\mathbf{K}$. Therefore, the leakage during the protocol run is $L_E^{\texttt{REM-IR}}(\mathbf{K}|\mathbf{S}_A) = |\mathbf{S}_A| = nH_b(e)$. □

Variants of `REM-IR` could be, e.g., to let Alice and Bob directly send $\mathbf{s}_A$ and $\mathbf{s}_B$, respectively, to the third party, who then computes $\mathbf{s_e} = \mathbf{s}_B - \mathbf{s}_A$. This version is equivalent, as also in `REM-IR` the third party knows all syndromes, i.e., it can compute $\mathbf{s}_B = \mathbf{s_e} + \mathbf{s}_A$ from the publicly known $\mathbf{s_e}$, $\mathbf{s}_A$. Furthermore, to increase the reliability and availability of the results, the computation can be delegated and distributed to an arbitrary number of third parties. The security is not jeopardized by any extended protocol involving more external untrusted parties and serves as a general baseline for such scenarios.

4.3.5 On Outsourcing Reverse Reconciliation

For continuous-variable QKD (CV-QKD) we have different requirements than for DV-QKD which not only impact the modulation schemes but also the information reconciliation. On the qbit level CV-QKD uses homodyne detection which allows for soft or hard decoding. For simplicity we will look only at discrete modulated CV-QKD, in particular binary modulation. Therefore, in the following we treat the CV-QKD system as hard-input–hard-output channel which operates on classical bit strings.

The idea of reverse reconciliation was introduced by Maurer [278] for classical communication and later applied to CV-QKD to overcome the 3dB loss limit [279]. In essence, reverse reconciliation is based on one-way error correction in reverse configuration with Bob sending the syndrome $\mathbf{s}_B$ to Alice, and Alice correcting her bits.

The underlying model is based on two channels, one connecting Alice and Bob and the other connecting Alice and Eve. Interestingly, if reverse reconciliation is applied in this scenario, a key can still be distilled even if the channel from Alice to Eve is superior to the one from Alice to Bob. The secret capacity of the channel for reverse reconciliation in [278] was derived as $C_s = H_b(e+d-2ed) - H_b(e)$, when Alice and Bob have access to a broadcast channel for public discussion. The bit error probabilities are e, d for the channel from Alice to Bob, and Alice to Eve respectively, and $e+d-2ed$ for the conceptual channel from Bob to Eve. H_b is the binary Entropy function.

In the classical model of [278] Shannon Entropy is used in the analysis. For the case of CV-QKD, the mutual information between Bob and Eve has to be replaced by the Holevo information and finite key effects have to be considered [280]. However, both refinements do not affect our treatment based on generic BSC channels.

For the secret channel capacity argument to be valid, Alice' key have to be kept private, thus, preventing Eve to correct error bits with her key. With this additional requirement, outsourcing information reconciliation directly as done in `REM-IR` is not possible. If both, $\mathbf{s}_e$ and $\mathbf{s}_B$ are leaked, $\mathbf{s}_A = \mathbf{s}_e + \mathbf{s}_B$ can be easily computed and the advantage over the conceptual channel is lost, because Eve can correct all errors in the string with Alice and remove uncertainty $H(\mathbf{K}_E|\mathbf{K}_B)$. More formally, the following result shows that *fully offloading* error corrections—i.e., letting a third party perform the entire error correction and simply return $\mathbf{e}$—to an untrusted party cannot be achieved for both, classical reverse reconciliation and in the quantum setting.

Theorem 4.3.2 (Impossibility of external syndrome decoding for classical RR)**.** For reverse reconciliation in the classical (non-quantum) setting fully offloading syndrome decoding is not possible with positive key rate.

Proof. Alice is connected to Bob and Eve over binary symmetric channels (BSC) with error rates e and d, respectively. She sends out the very same signal $\mathbf{k}_A$, which is received as $\mathbf{k}_B$ and $\mathbf{k}_E$. In the case of RR we further have that Alice sends the signal $\mathbf{k}_A$, but corrects her key for the error received by Bob, i.e., $\mathbf{k}_B$ is final key $\mathbf{k}$.

For binary random input encoding it holds that $H(K_A) = H(K_B) = 1$, and the mutual information $I_{AB} := I(K_A; K_B) = 1 - H_b(e)$ is defined by the error probability on the channel. K_A, K_B and K_E are the binary correlated random variables at Alice, Bob and Eve. The amount of information required to be exchanged during public discussion for reverse reconciliation per channel use is $|Q| \geq H(K_B|K_A) = H_b(e)$. For the proof we assume that optimal codes reaching the Shannon limit are used, i.e., equality holds for syndromes communicated.

Thus for offloading, any external party taking over the syndrome decoding for n bit keys based on a public $\mathbf{H}$ needs $|\mathbf{s}_e| = nH_b(e)$ amount of information to correct for the errors on the AB channel. Note here that $\mathbf{s}_e$ itself does not carry any information about the key, yet still fully defines the error $\mathbf{e}$.

We now prove the impossibility in two steps. In a first step (i) we calculate the change in mutual information by offloading the computation of $\mathbf{e}$ by Alice, and therefore publishing the error syndrome $\mathbf{s}_e$. In (ii) we then discuss the influence of discussion needed between Alice and Bob to compute the error syndrome $\mathbf{s}_e = \mathbf{s}_A - \mathbf{s}_B = \mathbf{k}_A\mathbf{H}^\top - \mathbf{k}_B\mathbf{H}^\top$ in the first place, which clearly needs contributions from both peers.

Furthermore, we know that Eve is not allowed to learn enough information about $\mathbf{k}$ to correct all errors through its conceptual channel, i.e., $I_{AB} - I_{EB}$ have to be preserved or at least be larger than 0 to leave Alice and Bob with a secure key.

In the beginning of the protocol we have $I_{AB} = 1 - H_b(e)$ and $I_{EB} = 1 - H_b(e + d - 2ed)$. After publishing $\mathbf{s}_e$ and computing $\mathbf{e}$ in step (i), the mutual information per bit changes to $I_{AB}^{(i)} = 1$ and $I_{EB}^{(i)} = 1 - H_b(e + d - 2ed) + H_b(e) = I_{EA}^{(i)}$. This means that with knowledge of $\mathbf{s}_e$ and implicitly $\mathbf{e}$, Alice can correct for all errors with Bob but Eve is left with some remaining uncertainty.

Now, to compute $\mathbf{s}_e = \mathbf{s}_A - \mathbf{s}_B$, another $nH_b(e)$ bits have to be communicated in advance between Alice and Bob (ii), which further impacts the knowledge of Eve about the keys. However, after exchanging another $nH_b(e)$ bits about $\mathbf{k}_B$ in public to compute the error syndrome we still have $I_{AB}^{(ii)} = 1$ but $I_{EA}^{(ii)}$ is also increased to 1 because $I_{EB}^{(i)} + H_b(e) = 1 - H_b(e + d - 2ed) + 2H_b(e) > 1$. Alice already corrected all errors in step (i), the additional information does not further increase their knowledge. Contrary, for Eve the information in step (ii) is useful and further increases the mutual information with Bob up to the maximum of 1, which means Eve has full knowledge about the agreed key. This is due to the fact, that the published information is about independent random variables K_E and K_B both contributing to the key agreement individually $\mathbf{k} = \mathbf{k}_B = \mathbf{k}_A + \mathbf{e}$. In summary, Eve either learns the key or if step (ii) is encrypted it leads to a negative key balance for QKD in the region of interest with $d \leq e$. □

Corollary 4.3.3 (Impossibility for quantum RR)**.** For reverse reconciliation in the quantum setting fully offloading syndrome decoding is not possible with positive key rate.

Proof. The quantum case is based on the same assumptions as the classical case and derives by looking at the entropies. Bob and Eve are connected to Alice over a quantum channel respectively, whereby $I(\mathbf{K}_A;\mathbf{K}_E) \geq I(\mathbf{K}_A;\mathbf{K}_B)$ holds. We also assume a symmetric system with $H(K_A) = H(K_B) = 1$, and consequently $H(\mathbf{K}_A|\mathbf{K}_B) = H(\mathbf{K}_B|\mathbf{K}_A)$ as well as $H(\mathbf{K}_A|\mathbf{K}_E) = H(\mathbf{K}_E|\mathbf{K}_A)$, due to Bayes' theorem. We also know from the definition of mutual information, that Eve has less uncertainty about the final key, i.e. Bob's key, than Alice $H(\mathbf{K}_A|\mathbf{K}_E) \leq H(\mathbf{K}_A|\mathbf{K}_B)$. Additionally, because the entropy function is concave we also know that $H(\mathbf{K}_A|\mathbf{K}_E) \leq H(\mathbf{K}_B|\mathbf{K}_E) \leq H(\mathbf{K}_A|\mathbf{K}_E) + H(\mathbf{K}_A|\mathbf{K}_B)$. Due to Slepian-Wolf's theorem [273] we require Bob to communicate $H(\mathbf{K}_B|\mathbf{K}_A)$ bits (e.g. $\mathbf{s}_B$) to enable Alice to compute the error syndrome. Furthermore, we require Alice to eventually publish $H(\mathbf{K}_A|\mathbf{K}_B)$ bits (e.g. $\mathbf{s}_e$) in order to fully outsource error correction also assuming an optimal code. Contrary to forward reconciliation, both strings published are useful for Eve, because the information about the error is independent from the bits revealed about Bob's key.

Thus, with access to this public information, Eve is now able to reduce its uncertainty $H(\mathbf{K}_B|\mathbf{K}_E)$ about the key, because

$$H(\mathbf{K}_B|\mathbf{K}_E) \leq H(\mathbf{K}_A|\mathbf{K}_E) + H(\mathbf{K}_A|\mathbf{K}_B) < 2H(\mathbf{K}_A|\mathbf{K}_B)\,,$$

leading to $I(\mathbf{K}_A;\mathbf{K}_E) = 1$. In essence, after seeing $\mathbf{s}_e$ and $\mathbf{s}_B$, Eve can calculate $\mathbf{s}_A = \mathbf{s}_B - \mathbf{s}_e$ and remove all uncertainty $H(\mathbf{K}_E|\mathbf{K}_A) < H(\mathbf{K}_A|\mathbf{K}_B)$ about $\mathbf{k}_A$ and subsequently the final key $\mathbf{k} = \mathbf{k}_B$. □

However, even with this results in mind, it is unclear if weaker notions of offloading would enable certain levels of partial or assisted secure outsourcing with positive key rates. An impossibility result for *partial offloading* is hard to formalize, as in an edge case no meaningful computation would be delegated to the untrusted server and the entire error reconciliation would be performed as local operations. In the following we argue that no obvious or natural approaches for reasonable (partial) delegation of computations do exist.

We have seen, that to be left with a secure key after all steps, the outsourced error reconciliation has to hide either $\mathbf{s}_e$ or $\mathbf{s}_B$ with ITS properties. However, $\mathbf{s}_e$ cannot be encrypted by masking, because the nature of the outsourced computation is to find a minimum weight vector which fulfills $\mathbf{e}\mathbf{H}^\top = \mathbf{s}_e$ for a given $\mathbf{s}_e$ and a public $\mathbf{H}$, which always requires to also publish a target vector $\mathbf{e}$ which is the reference for distance minimization. Therefore, a simple solution is to encrypt $\mathbf{s}_B$ during transmission with previously acquired secure key material. This requires $nH_b(e)$ additional key bits leading to a reduced capacity of $C_{s-enc} = H_b(e+d-2ed) - 2H_b(e)$. Although the protocol is secure and enables offloading of error correction, it does not lead to positive key rate for the regions of interest where $d < e$, which is also shown in Figure 4.11.

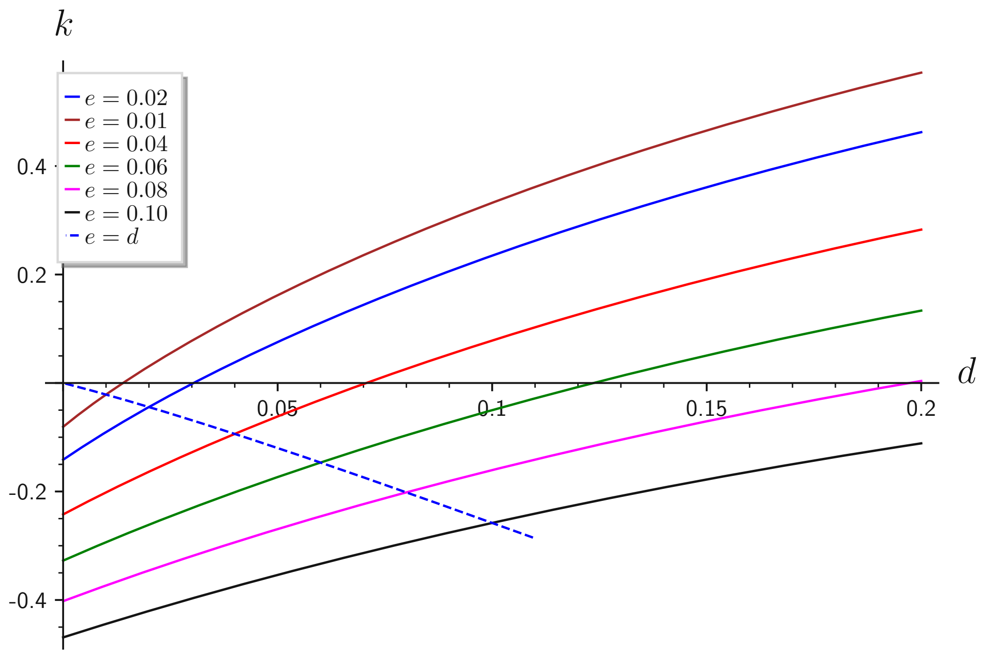


Figure 4.11: Secret key length balance (secret channel capacity) for reconciliation with encrypted s_B enabling error correction offloading to untrusted parties. Values are shown for different e. Left of the dashed line is the interesting region $d < e$, which has negative key balance and is therefore unfeasible.

To give more evidence that an encrypted IR protocol with positive key balance is not achievable, we review most relevant and evident techniques to protect the key of Alice or even $\mathbf{s}_A$ in an ITS sense, to prevent Eve from learning Alice' key or increase $I(\mathbf{K}_A;\mathbf{K}_E)$. In order to build an encrypted RR protocol, different techniques could be used, however, the parity check matrix $\mathbf{H}$ is considered to be publicly known, which limits the application of hiding techniques to the raw-key vector. Furthermore, the discussed solutions should not increase the computational effort to correct errors. We start from the syndrome decoding equation $\mathbf{s}_e = \mathbf{s}_A - \mathbf{s}_B = \mathbf{e}\mathbf{H}^\top$ and discuss options to hide $\mathbf{s}_A$ from Eve, or to prevent from any increase in I_{AE} by public discussion, e.g., by letting Eve learn about the bit flip positions.

Encryption. If the goal is to hide $\mathbf{s}_A$ in an ITS way given $\mathbf{H}$ is public, either $\mathbf{s}_e$ or $\mathbf{s}_B$ must be one OTP encrypted. $\mathbf{s}_B$ can be encrypted when transmitted to Bob or already at the key level, therefore ultimately hiding the key $\mathbf{s}'_B = (\mathbf{k}+\mathbf{m})\mathbf{H}^\top$, where $\mathbf{m}$ is a random masking value, which must also be securely transmitted from Bob to Alice. However, in the first case $|\mathbf{s}_B| = nH_b(e)$ bits are optimally required and in the second case number of raw-key bits $|\mathbf{k}|$ are required, which is extremely inefficient. Above, we have already shown that even the first case leads to negative key rates.

Unfortunately, encryption of $\mathbf{s}_e$ also cannot be used to hide bit error positions, because decoding requires a start vector to explore the vicinity to. Finding a vector close to a random vector with public $\mathbf{H}$ leaks the bit flip positions $\mathbf{e}$ and therefore also $\mathbf{s}_e$.

Permutation. An alternative method to hide $\mathbf{e}$, $\mathbf{s}_e$ and thus $\mathbf{s}_A$ would be by permuting the raw-key bits before running RR with an unencrypted $\mathbf{s}_B$. Using a permuted key $\mathbf{k}' = \Pi(\mathbf{k})$ for the post-processing would render error correction information useless for Eve, however, has to be random for each block and applied on both peers in secret. Thus, a huge amount of shared key material is required given the permutation has to be selected randomly from the $n!$ possible ones, which requires $O(n \log(n))$ bits to represent. In the end, if the selected permutation has to be communicated over the public channel via OTP the key balance is even worse than with syndrome encryption.

Padding. Padding the raw key with dummy bits could be used to hide error bits if combined with permutation. This corresponds to the technique of mixing raw key with dummy key bits. However, also in this case the positions and value of the dummy key bits have to be agreed on secretly by Alice and Bob, which also requires too many bits.

Finally, additional errors could be introduced only at Alice. Because the remaining error margin in practical CV-QKD is already very small, this technique can only hide small amount of information and substantially increases the computational work at the remote instance through the increased error rate.

In summary, all natural approaches for partially offloading RR with positive key rate to a single server in general seem unfeasible.

4.3.6 Verifiability of Outsourced IR

Besides the challenge of efficient yet secure outsourcing of information reconciliation, it is also important to have a means to efficiently check the correctness of the solution. This prevents from actively malicious behavior of the remote instance doing the actual work.

Fortunately, the problem of error decoding comes with an efficient algorithm to check the result:

1. Check if $\mathbf{eH}^\top = \mathbf{s_e}$, abort otherwise
2. Check if the weight of $\mathbf{e}$ is indeed below the threshold of the code, abort otherwise.

The firsts check can be easily computed by conducting the vector matrix multiplication and only requires $m \cdot n$ additions modulo 2 (XOR). The second check is even faster. The Hamming weight of $\mathbf{e}$ must be smaller than the distance of the code. This is guaranteed by the code used and the expected error rate from error estimation phase.

Ultimately, in the final confirmation phase an additional check is done to assure the key error probability, however, directly verifying the IR outsourcing step enables attribution of errors to external servers and flexible reaction besides aborting the whole process.

In summary, verifiability immediately follows from the nature of the problem. This makes protection against malicious remote servers possible with minimal effort and does not require a full re-computation by Alice.

4.3.7 Multiparty Computation Based Outsourcing

In the previous subsections we presented an efficient solution for offloading direct reconciliation (DR) to a single server and discussed problems for RR. Although relying on a single untrusted server seems the most desirable use case, it is natural to ask how efficient a multi-server configuration would be in cases where single server offloading is not possible. If multiple servers are available, ITS multiparty computation protocols (MPC) based on secret sharing—as introduced by Ben-Or et al. [72] and Chaum et al. [73]—can be used to obliviously compute arbitrary functions on sensitive data, thus they can also be used in CV-QKD because the inputs are kept private from the servers. The respective class of MPC protocols with ITS security operate in the honest majority setting, i.e., under the assumption that an adversary corrupts less than half of the MPC computing nodes. Besides the non-collusion assumption, the protocols also rely on secure channels, which can be assured by different means as discussed in use cases.

More concretely, in MPC a set of parties can jointly evaluate a function without leaking any information to any of the participating parties, beyond what can be derived from their own inputs and the computation result itself. Thus, MPC provides *input secrecy* (or *input privacy*), i.e., no party learns the input values of any other party, and *correctness*, i.e., the receiver of the result is ensured that the result is correct. In an honest-majority setting with less than half corrupt servers, ITS secure MPC protocols are among the most performant approaches for computing on encrypted data and achieve practical performance in many application scenarios.

We therefore looked into the problem of MPC-based information reconciliation with ITS security on the basis of secret sharing [100]. If IR is done in MPC, the decoding can be done without learning anything about the error syndrome $\mathbf{s}_e$ (private input) or error vector $\mathbf{e}$ (private output), but the parity check matrix can still be kept in clear. In this model the peer (Alice for RR) offloading IR is encoding the error syndrome as private input for the MPC system. The MPC system then obliviously computes the bit flip vector by executing a distributed protocol realizing a privacy preserving LDPC decoder. The result of the computation is secret shared among the MPC nodes after the protocol run and sent back to the peer (Alice), who can reconstruct it.

To demonstrate the feasibility of the solution, we study the practical efficiency of error decoding for low-density parity-check (LDPC) codes in an existing MPC framework and estimate the performance that can be achieved. To the best of our knowledge, this is the first time this problem is considered: The only known related work has been presented by Raeini and Nojoumian [281], who however only considered Berlekamp-Welch decoding for Reed-Solomon codes.

In general, we distinguish two main types of message-passing algorithms for LDPC decoding: bit-flipping algorithms and belief propagation [282]. The decoding approach typically used in QKD is from the category belief propagation (BP), and specifically uses sum-product mechanisms to update beliefs, an approach which works very efficiently on plaintext data. Unfortunately, this approach is not well suited for direct application in MPC. This is because the algorithm works on floating point numbers and uses trigonometric functions in the belief update part, both being very inefficient in MPC. Thus, to initiate the research topic we focused on bit-flipping algorithms (BF) first in our implementation approach, because it seemed more promising, although they are suffering from inferior performance in terms of information rate. BF algorithms have a very simple structure and work extremely fast, e.g., if implemented in hardware.

The bit-flipping algorithm is a non-probabilistic hard-input hard-output decoding algorithm and works on the Tanner graph representation of the code. The messages passing forth and back are all binary. The main structure of the BF algorithm is similar for all variants. In a first step, the variable nodes send their current value to the check nodes. Then the check nodes compute its value and feed back the result to the adjacent variable nodes signaling if the check is valid. After each variable node received the check bits from all connected check nodes the current guess for the code word is updated. Different approaches exist to update the variable nodes and to the best of our knowledge no optimized codes and methods for the particular case of QKD have been studied or analyzed. Therefore, we selected one of the most prominent solutions—Gallager's Algorithm [283]—to demonstrate feasibility and applied it to (non-optimized) codes available for BP algorithms.

We developed a MPC version for the bit flip decoding algorithm which is shown in Figure 4.1, and where $[[\cdot]]$ denote variables which are processed in the encrypted domain and are therefore kept confidential. The implementation assures that the codeword itself as well as related information is only kept in secret shared form among the parties and never revealed during computation. On a high level the algorithm performs message passing in a Tanner graph defined by the parity check matrix H with binary variables represented by public integers with values 0 and 1. We encode the bits on integers, because ITS-MPC work on algebraic circuits with additions being almost for free compared and multiplications requiring interactions between nodes. This allows us to quickly count the number of ones in a vector of bits and also enables exclusive-or over vectors by reducing modulo 2 after summing them up. The algorithm comprises 4 major steps which are iteratively repeated until a valid codeword is found or maximum of iterations is reached:

- In the first step, variable nodes pass their values to the check nodes where they are combined to compute the check value, which is 0 when the check is fulfilled and 1 if not.
- In the second step, the values of the check nodes are passed back to the variable nodes where they are aggregated, i.e., the number of check nodes not satisfied are counted for each variable node.
- Thirdly, the algorithm terminates if all check nodes are 0.
- Finally, the algorithm computes which variable nodes have to be flipped. Here we used Gallager's Algorithm B [283] in our implementation, which basically compares the counts computed in step two against a threshold value to decide which bits are flipped. Although the threshold value to compare to is public, the comparison has to be done obliviously to protect the variable node state as well as the bit flip information.

From a performance point of view, all steps except the last one are extremely fast in MPC, given that additions are only local operations and do not need any communication among the MPC peers and the reductions in step 1 can be done in parallel. The oblivious comparisons necessary to decide for each bit if it has to be flipped or not are the costly operations and are limiting the throughput in a MPC implementation. However, modern highly optimized MPC systems like MP-SPDZ[2]are able to achieve good performance even for this task as the results in Figure 4.2 show. This results indicate that MPC-based real-time decoding for QKD is possible.

Clearly, to achieve the best performance, optimized codes must be studied and designed in tandem with MPC protocols [8]. Also, BP based alternatives to sum-product decoding should be studied to see how fast MPC versions of belief propagation methods can be pushed. Additionally, for CV-QKD approximation approaches combined with multiedge-type codes [272] seem promising for fast MPC implementation. Nevertheless, our experiment already shows first results and paves the way for practical rates.

Finally, optimization-based decoding would also be possible as an alternative to message passing algorithms, i.e., by leveraging linear programming (LP). In LP decoding [284, 285] the maximum likelihood decoding problem is formulated as linear program. Thus, it is possible to decode a symbol by solving an associated LP with conventional approaches, e.g., with a simplex algorithm where also MPC versions exist [214]. However, for the QKD use case with block sizes k in the range of 10^4 to 10^6 bits and high error rates, the formulation would lead to a relatively large simplex tableau. Very low rates can be expected for this solution approach given the measured performance for MPC-based LP solving reported in [10].

[2]`https://github.com/data61/MP-SPDZ`, accessed 2023-01-10.

Algorithm 4.1: MPC version of bit flip decoding with allager's Algorithm B

```
Input: $[[c]]$ being the codeword to correct as list of $0/1$ bits $[[c_i]]$ stored in
       secure integer
       $h$ representing H as adjacency list of a Tanner graph with $h_i$ being a
index
       list of ones in row $i$ of H
       $t$ is a vector containing the number of 1s of each column in H
       $n, k, p$ as code parameters
       $iter_{max}$ iteration limit
Result: $[[c]]$ the corrected codeword after updating
$iter \leftarrow 0$
while $True$ do
    $iter \leftarrow iter + 1$
    /* step 1:  calculate sum at check nodes                    */
    for $row \in h$ do
        $[[v_i]] \leftarrow \sum_{i \in row} [[c_i]]$
        $[[v_i]] \leftarrow [[v_i]]\ (mod\ 2)$
    /* step 2:  count failed checks for every code bit          */
    $[[f]] \leftarrow [0 \ldots 0]$
    for $i \leftarrow 1$ to $p$ do
        for $node \in h_i$ do
            $[[f_{node}]] \leftarrow [[f_{node}]] + [[v_i]]$
    /* step 3:  if all checknodes are 0 then exit               */
    $[[f_{sum}]] \leftarrow \sum_{i \in \{1,..,n\}} [[f_i]]$
    if $f_{sum} = 0$ or $iter > iter_{max}$ then
        break while
    /* step 4:  calculate and apply bit flip vector             */
    $lvl = 1$
    while $True$ do
        for $j \leftarrow 1$ to $n$ do
            $[[b_j]] \leftarrow [[f_j]] > \lfloor (t_j/(lvl+1)) \rfloor$
        $[[b_{sum}]] = \sum [[b_i]]$
        if $b_{sum} = 0$ then $lvl \leftarrow lvl + 1$
        else
            $[[c]] \leftarrow [[c]] + [[b]]$
            break while
```

block size	bit width	circuit depth	time [s]	data [MB]	rounds	bitrate@10iter [bps]
1000	4	9	0.06	3.0	65	1571
1000	8	11	0.09	3.8	80	1116
10000	4	9	0.11	4.4	85	8932
10000	8	11	0.14	4.7	95	7054
100000	4	9	1.2	44	805	8354
100000	8	11	1.4	47	846	7117

Table 4.2: Performance comparison of MPC-based oblivious bit flip vector calculation as described in algorithm 4.1 (step 4) for LDPC decoding. The measurements were done in MP-SPDZ with *sharmir.sh* for three nodes with different block sizes and bit width of secure integer, circuit depth is then the multiplicative depth of the MPC circuit as processed by MP-SPDZ VMs and time is the time to compute the circuit. Data is the amount of information exchanged over the network during rounds of communication required by the VMs as reported by MP-SPDZ. Bitrate@10iter is the estimated throughput which can be achieved based on the measurements. The values show that the kilobit per second regime is feasible even without optimizations and block level parallelization.

4.3.8 Offload Privacy Amplification

Privacy amplification (PA) is another important step in the in the post-processing stack, cf. Section 4.1.2. It also requires a public channel for communication and is typically based on application of a randomly selected hash of a universal hash family thus achieving information theoretical secure randomness extraction. PA is used to extract the mutual information between Alice and Bob such that the adversary Eve is left without any information, except for a negligible error which can be made arbitrarily small.

Although the underlying matrix-vector multiplication seems rather efficient, because of finite key effects and its influence on the secure key rate large block length have to be used [280]. Therefore, also this step is computationally very demanding [262] and solutions to entirely offload this task from the device, or at least from the trusted area within a device, would be desirable.

In a PA protocol, Alice randomly selects a hash from a family of universal hashes with the right compression rate—based on Eves potential knowledge on the key—and communicates the selected function publicly to Bob. Both peers then apply the same function on the local reconciled key and arrive at the final shared key. The main property of a universal hash family is that they guarantee a low number of collisions even if the input is chosen by an adversary.

Because the block length in QKD is large, the complexity of the universal hashes is also relevant. One family of strongly universal hashes is given by multiplication of the raw key with a random matrix which would need a lot of randomness. Nevertheless, to reduce the randomness needed, a Toeplitz matrix can also used for PA, which requires only $n+m$ random bits compared to the $n \cdot m$ for a random matrix. The use of the Toeplitz matrix also reduces the computational effort for PA, because the diagonal structure also enables the use of a number theoretical transform for faster processing of the vector matrix product.

Assume that $\mathbf{k}'_A = \mathbf{k}'_B = \mathbf{k}'$ is the reconciled key at Alice and Bob, respectively, with length n and $\mathbf{k}$ is the final keys of length m. Then PA works as follows:

1. Alice randomly generates a uniform string of length $n+m-1$ defining the Toeplitz matrix $\mathbf{T}$ and sends it to Bob.
2. Alice computes $\mathbf{k} = \mathbf{k}'\mathbf{T}$ as final key.
3. Bob receives $\mathbf{T}$ from Alice and also computes his key as $\mathbf{k} = \mathbf{k}'\mathbf{T}$.

Thus, both parties do the same vector matrix multiplication to shrink the identical keys from n to m bits, where the ration n/m for CV-QKD is computed as in [280] and for DV-QKD as shown in [275].

If we want to offload PA, we would have to offload the core vector matrix multiplication which reduces the n raw-key bits to m final bits. The ratio is already known at the beginning of the PA step, but the Toeplitz matrix has to be generated for each block and exchanged in clear, which makes offloading a problem. Encrypting the PA matrix is not an option, it can be immediately seen that the key balance is negative if the encryption key for the matrix is longer than the raw key processed. Thus, hiding the input/output keys while still offloading the computation is not feasible in a single sever model.

However, if multiple servers are available, a very efficient protocol is possible, by offloading to two or more severs in a non-interactive way, i.e., without requiring the servers to communicate. The protocol is shown in Figure 4.12. The peer shares the raw key into n parts with a linear secret sharing scheme—working over $\mathbb{F}_2$ or a larger prime field $\mathbb{F}_p$—, and sends them to the servers (one share per server). The servers compute the $[\mathbf{k}] = [\mathbf{k}]\mathbf{T}$ where $[]$ denotes the sharing of a value and the multiplications and additions are done on the shares without interaction between the servers. The linearity of the sharing is used here. The result is then sent back to Alice and reconstructs to the final key. For the case of prime fields Alice additionally reduces the result vector *mod* 2.

The security of the protocol against passive adversary is governed by the security of the underlying secret sharing scheme: Because the parties do not interact with each other but only communicate with the peer, they cannot learn any information about the final key as long as an ITS linear secret sharing is used, e.g., additive or Shamir secret sharing [100]. The computational effort of the solution is the same for every server which would also be the same as for local computation.

Unfortunately, the `REM-PA` protocol does not provide efficient verifiability beside recomputation or spot checking, and therefore efficient protection against active attackers cannot be easily achieved during the PA phase. However, if the confirmation round

Protocol `REM-PA`:

QKD-Peer Q:

1. Generates a sharing of $[\mathbf{k}]$ by calling *share* of a linear secret sharing algorithm
2. Sends shares $[\mathbf{k}]_i$ to Party P_i
3. Receive enough shares $[\mathbf{k}]'_i$ to *reconstruct* $\mathbf{k}'$
4. Reduce elements of vector $\mathbf{k}' \mathit{mod}\ 2$
5. Return $\mathbf{k}'$

MPC-Party P_i:

1. Receive share of key string $[\mathbf{k}]_i$
2. Calculate $[\mathbf{k}']_i = [\mathbf{k}]_i \mathbf{T}$ with conventional PA algorithm
3. Send $[\mathbf{k}']_i$ back to Q

Figure 4.12: `REM-PA` Protocol

is shifted after PA, it will detect errors in the keys and prevent from erroneous keys by aborting the protocol. Thus, it can also detect malicious behavior of the external servers, but not directly attribute the errors to them. Additionally, a secure channel is required to distribute the shares to the severs. However, contrary to the MPC based LDPC decoding not interaction between servers is required.

4.3.9 Use Cases

To answer why offloading computationally intensive tasks is interesting at all, we present the expected benefits in general and discuss advantages for certain networking scenarios.

The overall goal which can be achieved are savings in energy and/or cost at the device side beneficially impacting the cost-effectiveness of the end-user equipment. Therefore, if the devices are simpler and require less computational power, the cost savings could be substantial, e.g., in cases where the user buys the equipment. Compared to data center environments, the devices are also less energy-efficient for running computation-intensive tasks. If this part can be offloaded to a more efficient data center also the overall operational cost can be lowered in addition. Therefore, dedicated cloud solutions which further pool information reconciliation for a larger amount could further help to reduce energy consumption. By regularly updating the external hardware resources, the system can benefit from Moore's law and the continuous drop in cost of compute resources. They can even be shifted flexibly between different locations and data centers to optimize energy usage and cost if more offerings are available. In general, it would be even possible to leverage public cloud services for REM-IR, which requires no trust assumption at all about the environment. Because of these arguments, we think the ability to offload and relocate computationally intensive tasks also leads to higher energy efficiency of compute resources.

Additionally, it could also lead to more flexibility on the QKD level, i.e., QKD as a service. The virtualization of the computationally expensive post-processing tasks could be convenient in the future. Not all optical network units (ONU) have access to QKD functionality but the same hardware may be used for coherent passive optical networks (PON) and CV-QKD [286, 287]. So, we may provide QKD to them by just allocating additional processing resources while also switching their software-defined transceiver into QKD mode. Furthermore, networking equipment is installed for longer times and not often updated. This is even more true for high-cost security-certified equipment, because upgrading security-certified equipment is a cumbersome and costly process which typically requires re-certification. Being able to update certain non-critical components without needing to exchange or re-certify the core QKD device hardware can greatly simplify the upgrade process.

To show how the advantages relate to concrete use cases we quickly mention three examples.

Access networks. In the case of access networks, we find constraint resources (computing energy) and the network units must be low-cost because they are the driving cost factor, especially, because only very low key rates are typically required (AES key refreshing). If computational resources are pooled in such a scenario, cost can be substantially reduced, not just in case of a reduced user subscription ratio, but also due to time sharing of centralized CPU resources. Furthermore, because the optical part is low energy the dominant cost and energy factor is when a CPU is partially idle, which should be avoided.

Satellite communication. Satellites have a particularly long lifetime (20-30 years) and have to be remotely operated and maintained. They also have limited access to energy resources and reducing energy consumption is of paramount interest. This is especially true if low-cost (mobile) earth stations should be supported or even for inter satellite links. Offloading post-processing can make satellite transceiver possible and increase the connectivity for individual satellites.

Integrated COTS Hardware. Finally, beside the evident advantages of outsourcing protocols like REM-IR to data centers, the concept can also be interesting when applied within the device. QKD devices are complex systems [49] and comprise many different components which makes security auditing and certification very hard. To achieve strong security guarantees only trustworthy hardware and software can be used to process key material in plaintext [58]. Furthermore, to prevent from side channel attacks and backdoors it would be desirable to reduce the number of trusted components and the complexity of the secure environment in a device as good as possible. Therefore, if the components processing sensitive key material can be reduced, this results in a smaller attack surface, simplifies security analysis and helps in the certification process. The

MPC-based protocols presented can be used for this purpose, i.e., to reduce the trusted environment on the device architecture level with all its benefits. Within a device it is also feasible to realize the secure channels required in the MPC model. Thus, it would allow for the integration of COTS hardware in QKD systems only processing keys in encrypted form.

4.4 QKD Authentication with Non-ITS MAC

During our work on timing synchronization we also discovered a new attack on an specific authentication method developed at AIT and used on the communication channel. It was originally proposed in [64] to save key material and make QKD more efficient by leveraging non-ITS message authentication as sub-protocol. The protocol was developed to improve the net key balance for QKD by reducing key consumption for authentication of short messages, but partial attacks emerged after publication [288, 55] and made some weaknesses evident. However, it was only during this thesis that the author developed a new attack strategy which led to a full break of the original protocol and variants thereof. The full analysis was presented in [14] where all possible protocol scenarios where discussed. Before we present the attack we recap the non-ITS authentication protocol and also introduce the BB84 protocol with post-processing in more detail.

4.4.1 Overview of Non-ITS MAC Protocol

The standard cryptographic approach ensuring authenticity of communication messages against malicious attackers is to use a message authentication code (MAC) [289]. A convenient class of MACs are *systematic* MACs which replace the original message with a concatenation of the message itself and an additional tag which is the image of a keyed hash function applied to the message. It is well-known that Strongly Universal$_2$ (SU$_2$), and more generally Almost Strongly Universal$_2$ (ASU$_2$) hashing is an ITS primitive that can be used to calculate systematic MAC tags. This is also the standard method for channel authentication in QKD.

The authentication mechanism proposed in Ref. [64] aimed to consume less key than ASU$_2$ authentication. The intended goal was a positive key balance for the combination QKD plus authentication even in realizations that use (relatively) short blocks in the post processing step. Note that later experimental progress has made these objectives not so relevant, as short key blocks are no longer necessary from an implementation perspective [290]. Still, a complete security analysis of the authentication mechanism of [64] is intriguing from a theoretical point of view as the mechanism has interesting properties not shared by any of the methods mentioned above.

To start with, we summarize the proposal and introduce some notation. The basic principle of the protocol is depicted in Figure 4.13. The proposal relies on a two-step hash function evaluation: $t = g_K(m) := h_K(f(m))$, where $f : \mathcal{M} \to \mathcal{Z}$ is a publicly known hash function and $h_K : \mathcal{Z} \to \mathcal{T}$ belongs to an SU_2 hash function family $\mathcal{H}$. Here, $\mathcal{M}$ is the set of messages to be authenticated, $\mathcal{Z}$ is an intermediate set of strings, and $\mathcal{T}$ is the set of tags with $|\mathcal{M}| \gg |\mathcal{Z}| > |\mathcal{T}|$.

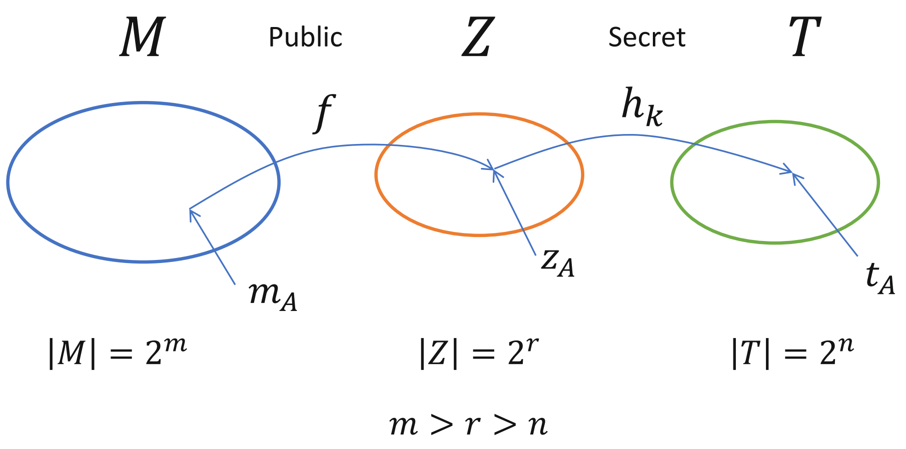


Figure 4.13: Principle of non-ITS MAC algorithm as proposed in [64]. An intermediate public hash is used first to compress message and subsequently input in a ITS secure MAC based on universal two family of hash functions.

The proposed two-step authentication method is not information-theoretically secure, because a known classical hash function is used within the protocol, but it was argued that nevertheless it does not compromise the security of QKD. The security of the above protocols is informally given by the following two properties.

Insertion of messages is ruled out. Assume that Eve attempts to calculate or guess the tag for a fixed message $m^{\mathbf{E}}$ that she wants to insert. In that case she has a success probability of $1/|\mathcal{T}|$ (irrespective of her computing power). This is because the key K which identifies the SU_2 hash function is not known to her. Thus, the authentication mechanism is (first-)preimage resistant, i.e. knowledge of the authentication tag alone does not allow to find messages yielding the same tag.

Substitution with given messages is ruled out. Let us further assume, Eve has intercepted a (valid) message-tag pair $(m^{\mathbf{A}}, t)$ from Alice and wants to substitute it with her *fixed* message $m^{\mathbf{E}}$ and some tag. Then Eve's chances increase slightly because she now has access to the intermediate value $f(m^{\mathbf{A}})$, and can check if $f(m^{\mathbf{A}}) = f(m^{\mathbf{E}})$. If there is a collision, Eve knows that $(m^{\mathbf{E}}, t)$ is a valid message-tag pair and can just send this, otherwise she guesses the tag as above. The total probability of success is now bounded by the guessing probability plus the collision probability, and assuming that $m^{\mathbf{A}}$ is random to Eve and that f is a good hash function, the collision probability is low (for details see [64]).

So this two-step authentication works well in a situation when Eve is given a fixed message $m^{\mathbf{E}}$ to generate the tag for. One immediate consequence is that Eve cannot perform the straightforward MITM attack with significant success probability since she would need to generate tags for messages $m^{\mathbf{E}}$ from her devices without knowledge of K, for which case the above bound applies.

The weakness. However, the protocol also has a weakness which turned out to be useful in breaking the approach. One should note that using the intercepted message-tag pair $(m^{\mathbf{A}},t)$ and enough computational power, Eve can in principle search for other preimages of t under f. If she can find (at least) one message $\tilde{m}^{\mathbf{E}}$ such that $f(m^{\mathbf{A}}) = f(\tilde{m}^{\mathbf{E}})$ then $h_K(f(m^{\mathbf{A}})) = h_K(f(\tilde{m}^{\mathbf{E}}))$ and therefore $(\tilde{m}^{\mathbf{E}},t)$ is a valid message-tag pair *for any key* K. She can now replace $m^{\mathbf{A}}$ with $\tilde{m}^{\mathbf{E}}$ with success probability of 100%. Now the question is if this (one of these) $\tilde{m}^{\mathbf{E}}$ can be used in place of the message $m^{\mathbf{E}}$. It would seem that, if Eve strictly follows the appropriate QKD protocol (random settings, best possible bit error rate, ...), this is not possible.

However, Eve is not forced to follow the precise requirements of the QKD protocol [291]; she only needs to make it seem to Alice and Bob that she does so. For example, Eve does not need to use random settings (e.g. preparation bases and raw keys), or even correctly send all settings she used. If it helps her, she can use a fixed sequence of settings or report other settings for some qubits than the ones actually used. This constitutes the basis for a sophisticated MITM attack that can break simplified QKD protocols. However, in this section we propose a different approach which targets the quantum transmission together with the sifting phase and achieves a full break with high probability.

4.4.2 Generic BB84 with Immediate Message Authentication

For our analysis we state a typical BB84 protocol also falling in the category of prepare-and-measure protocols. Each individual message is authenticated by the method laid out before. It is, however, straightforward to apply the attack discussed below to the case of an entanglement based protocol. As usual, Alice and Bob must provide for time synchronization between the state preparation and the measurement.

It is implicitly assumed, that on receiving messages Alice and Bob only accept valid message tag pairs of incoming messages and abort otherwise. In general, the BB84 protocol can be divided in two separate parts: (S) quantum state transmission and sifting which is heavily dependent on the quantum level implementation, and (P) post processing consisting of more generic classical building blocks like error estimation, error correction, confirmation, and privacy amplification. For our attack we only have to interfere in (S) and rely on correct operation of part (P) which takes the result of (S) (i.e. the sifted keys) as input.

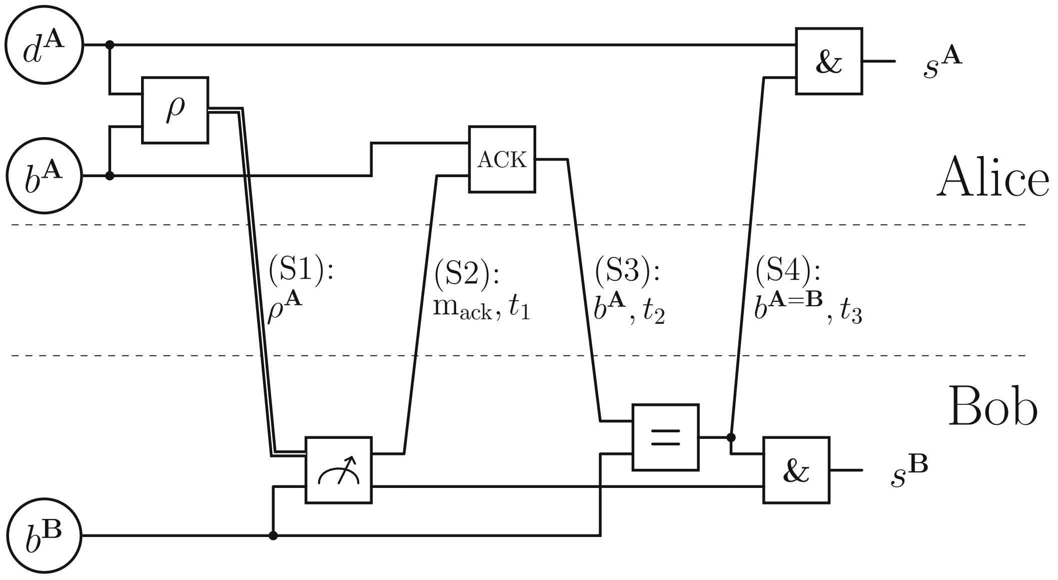

$d^{\mathbf{A}}$
$b^{\mathbf{A}}$
$b^{\mathbf{B}}$
ρ
ACK
&
$s^{\mathbf{A}}$
Alice
(S1): $\rho^{\mathbf{A}}$
(S2): $\mathrm{m}_{\mathrm{ack}}, t_1$
(S3): $b^{\mathbf{A}}, t_2$
(S4): $b^{\mathbf{A=B}}, t_3$
Bob
=
&
$s^{\mathbf{B}}$

Figure 4.14: BB84 Protocol (Quantum exchange and sifting only). Time flow is from left to right. Single (double) lines represent classical (quantum) communication. Local protocol actions are depicted by boxes: ρ depicts state preparation, the indicator is a quantum measurement device, the ACK box denotes that Alice waits for Bob's message until she continues with the protocol, = denotes the calculation of identical bases, & denotes the filtering of signals (in different bases).

The basic protocol flow for the BB84 quantum state transmission and sifting are shown in Figure 4.14. To run the protocol Alice and Bob have to be given three pre-shared keys (K_1, K_2, K_3) to authenticate messages on the classical channel, i.e., messages (S2), (S3), and (S4). Thus, it is assumed that all messages are protected by the respective authentication algorithm. The relevant protocol phase comprises the following steps:

(Sa) Alice creates two random bit strings of length N: her raw key $d^{\mathbf{A}}$ (data bits), and the bases string $b^{\mathbf{A}}$ (preparation bases), $d^{\mathbf{A}}, b^{\mathbf{A}} \in_r \{0,1\}^N$. For all pairs of bits $\left(d_k^{\mathbf{A}}, b_k^{\mathbf{A}}\right)$ Alice generates the corresponding quantum states $\rho_k^{\mathbf{A}} \in \{\rho^0, \rho^1, \rho^2, \rho^3\}$. Using quantum channel $\mathcal{Q}$, Alice sends the quantum state $\rho^{\mathbf{A}} = \bigotimes_{k=1}^{N} \rho_k^{\mathbf{A}}$ ("string" of all $\rho_k^{\mathbf{A}}$'s) as (S1) to Bob, i.e., (S1) := $(\rho^{\mathbf{A}})$

(Sb) Bob creates a random bases string $b^{\mathbf{B}} \in_r \{0,1\}^N$. Bob measures $\rho^{\mathbf{A}}$ in bases $b^{\mathbf{B}}$ and obtains $d^{\mathbf{B}} \in \{0,1,\mathit{empty}\}^N$, where *empty* corresponds to no measurement result at Bob, e.g., due to absorption in the channel, or the imperfection of the detectors. For all k with $d_k^{\mathbf{B}} = \mathit{empty}$, Bob sets $b_k^{\mathbf{B}} = \mathit{empty}$. Using the classical channel $\mathcal{C}$, Bob sends an acknowledgement message (S2) to Alice, with, (S2) := $(\mathrm{m}_{\mathrm{ack}}, g_{K_1}(\mathrm{m}_{\mathrm{ack}}))$

(Sc) Alice waits until she has received (S2), ensuring that the measurements have been finished before bases exchange is performed. If correctly received, Alice sends her bases string with authentication tag to Bob using $\mathcal{C}$, i.e. (S3) $:= b^{\mathbf{A}}, g_{K_2}(b^{\mathbf{A}})$.

(Sd) After receiving (S3), Bob calculates a bit string $b^{\mathbf{A=B}}$, such that $b_k^{\mathbf{A=B}} = 1$, if $b_k^{\mathbf{A}} = b_k^{\mathbf{B}}$, and $b_k^{\mathbf{A=B}} = 0$, otherwise, for $1 \leq k \leq N$. Thus, if Alice and Bob prepared and measured $\rho_k^{\mathbf{A}}$ in the same basis $b_k^{\mathbf{A=B}} = 1$, and $b_k^{\mathbf{A=B}} = 0$ if either the bases were different, or Bob did not measure a signal. Additionally, Bob removes from $d^{\mathbf{B}}$ all bits $d_k^{\mathbf{B}}$ where $b_k^{\mathbf{A=B}} = 0$ and obtains $s^{\mathbf{B}}$. Finally, using $\mathcal{C}$, Bob sends the information about selected bits as (S4) to Alice, i.e., (S4) $:= b^{\mathbf{A=B}}, g_{K_3}(b^{\mathbf{A=B}})$

(Se) Upon receiving (S4), Alice removes from $d^{\mathbf{A}}$ all bits $d_k^{\mathbf{A}}$ where $b_k^{\mathbf{A=B}} = 0$ and obtains $s^{\mathbf{A}}$.

After the steps Sa-Se the post processing steps are executed as follows: error estimation, correction and confirmation information is computed by Alice and sent to bob (Pa), Bob corrects key for error and confirms it before sending back the real error rate (Pb), Alice selects an adequate privacy amplification function, sends it to Bob and also applies it locally (Pc), and Bob also applies the privacy amplification function to the local key to arrive at the same key as Alice with high probability.

4.4.3 Man-in-the-Middle Attack

In our initial version of the attack, we leverage a rather simple idea to attack the quantum communication and sifting phase. Besides performing a conventional man-in-the-middle attack, we introduce the concept of an approximate attack to break one of the security assumptions from the original protocol. In fact, we argue that it is possible for an unbounded adversary Eve to search for an approximate message leading to the same tag in the first stage hashing step which is useful in later steps and eventually lead to common keys with Alice and Bob respectively.

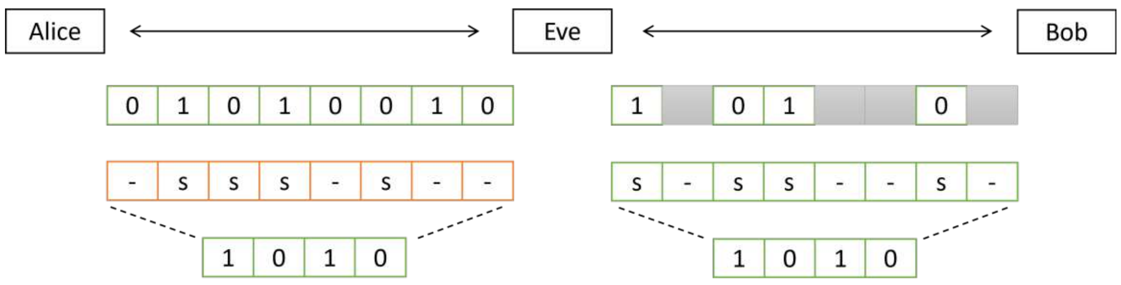


Figure 4.15: Overview of subsequence selection method. After the attack on the quantum (optical) channel Eve arrives at a state where she shares a key with Alice fully known to her and a different but smaller key with Bob. To successfully attack the protocol Eve has to send Alice a new sifting message selecting a subsequence corresponding to Bobs key. Eve succeeds if she can find a useful (approximate) collision for in the first stage non-ITS MAC of the authentication protocol.

The basic pattern of the attack is to learn all events from Alice by measuring after the basis have been revealed and to force Alice and Bob to agree on a common key although they did not work with the same qubits. To do so, Eve has to send random signals to Bob first in order to provoke Alice to release her bases after Bob acknowledged the measurements. After the attack on the quantum channel Alice knows the signal sent by Alice through its own measurement and the bases measured by Bob from the sifting message of Bob. The main task of Eve is then to exchange the sifting message of Bob to fit own needs, i.e., to remove all events where Eve and Bob had different bases. If Eve manages to make Alice only keep event where Eve and Bob where in the same basis, she will learn the key without being noticed and successfully break the protocol. To make this attack possible, Eve has to be able to find valid collisions close to messages she would like to exchange, and she must be able to extract a subsequence/substring from the key shared with Alice to map the key shared with Bob. The subsequence approach is essential for this attack and is shown in Figure 4.15.

In the following we describe the attack for the specific protocol introduced as also depicted in Figure 4.16:

(Sa) Alice performs step (Sa) of the protocol (prepares $\rho^{\mathbf{A}}$ and sends it in (S1)).

(Sa') Eve intercepts (S1) from Alice and stores $\rho^{\mathbf{A}}$ in her quantum memory, e.g, simple delay fibre. Then Eve performs exactly as Alice in step (a) of the protocol: Eve determines random $d^{\mathbf{E}}$ and $b^{\mathbf{E}}$, prepares a state $\rho^{\mathbf{E}}$ and sends it in (S1') to Bob.

(Sb) Bob performs step (Sb) of the protocol measuring the state Eve has prepared, $\rho^{\mathbf{E}}$, instead of $\rho^{\mathbf{A}}$, as in the protocol (in the following denoted as $\rho^{\mathbf{A}} \to \rho^{\mathbf{E}}$). Bob also sends the acknowledgement (S2) to Alice now that he measured a signal.

(Sc) Alice performs step (Sc) of the protocol. She sends (S3).

(Sc') Eve intercepts (S3), i.e. $b^{\mathbf{A}}$ and the corresponding tag $t_2 := g_{K_2}(b^{\mathbf{A}})$, and measures her quantum memory in bases $b^{\mathbf{A}}$ and obtains an identical copy of Alice's raw key, $d^{\mathbf{A}}$.

(Sc") Eve determines $\tilde{b}^{\mathbf{E}}$ (e.g. using an exhaustive search), such that the intercepted t_2 validates $\tilde{b}^{\mathbf{E}}$ and $d_H(\tilde{b}^{\mathbf{E}}, b^{\mathbf{E}})$ is small (cf. Corollary 4.4.2), and sends (S3') to Bob.

(Sd) Bob performs step (Sd) of the protocol based on (S3') received ($b^{\mathbf{A}} \to \tilde{b}^{\mathbf{E}}$, $b^{\mathbf{A=B}} \to b^{\mathbf{B=E}}$), obtains $s^{\mathbf{B}}$ and sends message (S4).

(Sd') Eve intercepts (S4), i.e. $b^{\mathbf{B=E}}$ and the corresponding tag $t_3 := g_{K_3}(b^{\mathbf{B=E}})$. Eve removes from $d^{\mathbf{E}}$ all bits $d_k^{\mathbf{E}}$ where $b_k^{\mathbf{B=E}} = 0$ and obtains $s^{\mathbf{E}\leftrightarrow\mathbf{B}} \approx s^{\mathbf{B}}$ (in general $s^{\mathbf{E}\leftrightarrow\mathbf{B}} \neq s^{\mathbf{B}}$ because Eve had to send $\tilde{b}^{\mathbf{E}}$ instead of her true basis choice $b^{\mathbf{E}}$ in step (Sd")).

(Sd") Using the algorithm detailed in 4.2, Eve searches for a subsequence of $d^{\mathbf{A}}$ that coincides with $s^{\mathbf{E}\leftrightarrow\mathbf{B}}$ and calculates $b^{\mathbf{A=E}}$ such that in Alice's next step, (Se), Alice would create $s^{\mathbf{A}} \approx s^{\mathbf{E}\leftrightarrow\mathbf{B}}$ as her sifted key. Typically Eve will have to allow for $O(\sqrt{n})$ bits that will be different between $s^{\mathbf{A}} = s^{\mathbf{E}\leftrightarrow\mathbf{A}}$ and $s^{\mathbf{E}\leftrightarrow\mathbf{B}}$ (see Lemma 4.4.3).

(Sd"') As in step (Sd") Eve determines $\tilde{b}^{\mathbf{A=E}}$ with small Hamming distance to $b^{\mathbf{A=E}}$, this time validated by t_3 obtained in step (Sd'), calculates the actual sifted key of Alice, $s^{\mathbf{E}\leftrightarrow\mathbf{A}} \approx s^{\mathbf{E}\leftrightarrow\mathbf{B}}$ and sends (S4') to Alice.

(Se) **A** performs step (Se) of the protocol ($b^{\mathbf{A}=\mathbf{B}} \to \tilde{b}^{\mathbf{A}=\mathbf{E}}$) and obtains $s^{\mathbf{A}} = s^{\mathbf{E}\leftrightarrow\mathbf{A}}$.

After interfering on the sifting stage, Eve has almost reached her goal, as now $s^{\mathbf{A}} \approx s^{\mathbf{E}} \approx s^{\mathbf{B}}$ holds. In the following steps of the protocol Eve only listens to Alice's messages on the classical channel for information reconciliation (e.g. syndrome for error correction), confirmation and privacy amplification to correct and check her sifted key $s^{\mathbf{E}}$ and privacy amplify it to her final key $K^{\mathbf{E}}$ which is identical to Alice's and Bob's key, i.e. $K^{\mathbf{A}} = K^{\mathbf{E}} = K^{\mathbf{B}}$. Thus, with Eve knowing all bits of the generated key she succeeded in fully breaking the protocol. Furthermore, Alice and Bob are not able to detect the attack and abort key generation.

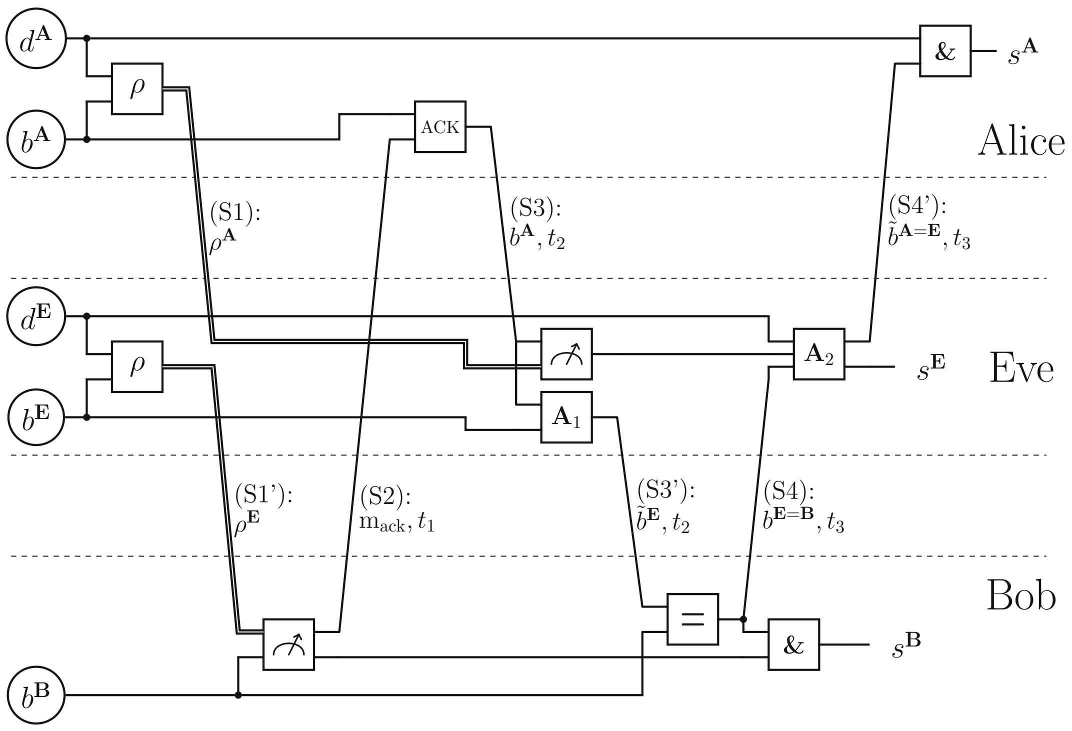

$d^{\mathbf{A}}$
$b^{\mathbf{A}}$
ρ
ACK
&
$s^{\mathbf{A}}$
Alice
(S1): $\rho^{\mathbf{A}}$
(S3): $b^{\mathbf{A}}, t_2$
(S4'): $\tilde{b}^{\mathbf{A}=\mathbf{E}}, t_3$
$d^{\mathbf{E}}$
$b^{\mathbf{E}}$
ρ
$\mathbf{A}_1$
$\mathbf{A}_2$
$s^{\mathbf{E}}$
Eve
(S1'): $\rho^{\mathbf{E}}$
(S2): $\mathrm{m}_{\mathrm{ack}}, t_1$
(S3'): $\tilde{b}^{\mathbf{E}}, t_2$
(S4): $b^{\mathbf{E}=\mathbf{B}}, t_3$
Bob
=
&
$s^{\mathbf{B}}$
$b^{\mathbf{B}}$

Figure 4.16: Attack on Protocol BB84Time flow is from left to right.

As shown in [14], the attacks can be adapted to all variations found in the field ranging from swapping roles between Alice and Bob in the post-processing or delaying authentication of certain protocol steps.

4.4.4 Success Prob. for Approximate and Correlation Attacks

After presenting the main ideas of the attack and a concrete attack strategy we now show that the attacks are practical in the sense that approximate messages and subsequence exist. The essence of the attack is to intercept a valid message-tag pair (sent by Alice or Bob) and—using large computational resources and/or leveraging weaknesses of the public hash function algorithm—find further preimages of the tag (messages that hash to the same hash value as the intercepted message) that are useful for the eavesdropper. Additionally, it must be feasible to find a subsequence in a given random sting to map a shorter different key.

4.4.5 Finding Hash Collisions in a Set of Messages

Assume that Eve has intercepted a message-tag pair $(m^{\mathbf{A}}, t)$ from Alice. The following lemma gives a lower bound for the probability that (under a mild assumption) a set $\mathcal{M}$ of messages contains at least a single message $m^{\mathbf{E}}$ that collides with $m^{\mathbf{A}}$, i.e. $h_K(f(m^{\mathbf{E}})) = t$.

Lemma 4.4.1. Let us assume that f maps all messages in $\mathcal{M}$ randomly onto $\mathcal{Z}$. Then the probability that at least one of the messages in $\mathcal{M}$ is validated by the given tag $t = h_K(f(m^{\mathbf{A}}))$ is

$$\mathcal{P}^{\text{succ}}_{\text{coll}} = \Pr\left\{\exists m^{\mathbf{E}} \in \mathcal{M} : h_K(f(m^{\mathbf{E}})) = t)\right\} > 1 - \exp\left(-|\mathcal{M}||\mathcal{Z}|^{-1}\right).$$

Proof. By assumption, the probability that f maps any (randomly chosen) message m of $\mathcal{M}$ onto any fixed value z of $\mathcal{Z}$ is $1/|\mathcal{Z}|$:

$$m \in_R \mathcal{M}, \forall z \in \mathcal{Z} : \Pr\{f(m) = z\} = 1/|\mathcal{Z}|. \tag{4.2}$$

Applying h_K to $f(m)$ and z in the argument of Pr (which potentially increases the value of the probability), setting $z = f(m^{\mathbf{A}})$, and using $t = h_K(f(m^{\mathbf{A}}))$ we obtain

$$m \in_R \mathcal{M} : \Pr\{h_K(f(m)) = t\} \geq 1/|\mathcal{Z}|. \tag{4.3}$$

Consequently, the probability that t authenticates at least one message of all $|\mathcal{M}|$ different messages in $\mathcal{M}$ is at least $1 - \left(1 - |\mathcal{Z}|^{-1}\right)^{|\mathcal{M}|}$, and using that $(1 - 1/n)^n < e^{-1}$ for $n > 1$ finishes the proof. □

If desired, $1/|\mathcal{Z}|$ can be replaced by any lower bound on the probability to allow for non-uniform distributions. Since no assumptions on the computational power of Eve are imposed, she will be able to find with probability $\mathcal{P}^{\text{succ}}_{\text{coll}}$ such an $m^{\mathbf{E}}$. Note, that $|\mathcal{M}| = |\mathcal{Z}|$ is sufficient to get $\mathcal{P}^{\text{succ}}_{\text{coll}} > 0.63$.

4.4.6 Finding Hash Collisions with Small Hamming Distance

Assume that Eve has intercepted a message-tag pair $(m^{\mathbf{A}}, t)$ from Alice. The following corollary of Lemma 4.4.1 states that (under a mild assumption) for any fixed message

$m^{\mathbf{E}}$, that Eve would like to send, there exists with probability almost 1 a message $\tilde{m}^{\mathbf{E}}$, such that (i) $\tilde{m}^{\mathbf{E}}$ is almost identical to $m^{\mathbf{E}}$, i.e. $\tilde{m}^{\mathbf{E}}$ has *small Hamming distance* to $m^{\mathbf{E}}$, and (ii) $(\tilde{m}^{\mathbf{E}}, t)$ will be accepted as authentic, i.e. $h_K(f(\tilde{m}^{\mathbf{E}})) = t$.

Corollary 4.4.2. Let $\mathcal{B}$ be the closed ball of all messages m having a Hamming distance to $m^{\mathbf{E}}$ not exceeding w:

$$\mathcal{B} = \left\{ m : d_H(m, m^{\mathbf{E}}) \le w \right\},$$

and let us assume that f maps all messages in $\mathcal{B}$ randomly onto $\mathcal{Z}$. Then the probability that at least one of the messages in $\mathcal{B}$ is validated by the given tag $t = h_K(f(m^{\mathbf{A}}))$ is

$$\mathcal{P}^{\text{succ}}_{\text{coll}} = \Pr\left\{ \exists \tilde{m}^{\mathbf{E}} \in \mathcal{B} : h_K(f(\tilde{m}^{\mathbf{E}})) = t \right\} > 1 - \exp\left(-|\mathcal{B}||\mathcal{Z}|^{-1}\right).$$

For simplicity we can loosen the bound and replace $|\mathcal{B}|$ by $\binom{\ell}{w} < |\mathcal{B}|$, where ℓ is the length of the binary message $m^{\mathbf{E}}$.

Proof. The proof follows from Lemma 4.4.1 by setting $\mathcal{M} = \mathcal{B}$. Finally, $|\mathcal{B}| = \sum_{k=0}^{w} \binom{\ell}{k} > \binom{\ell}{w}$. □

If desired, $1/|\mathcal{Z}|$ can be replaced by any lower bound on the probability to allow for non-uniform distributions. Since no assumptions on the computational power of Eve are imposed, she will be able to find with probability $\mathcal{P}^{\text{succ}}_{\text{coll}}$ such an $\tilde{m}^{\mathbf{E}}$. For typical parameters, e.g. $|\mathcal{Z}| = 2^{256}$, and $\ell = 2^{12}$ (2^{13}, 2^{14}, 2^{15}, 2^{16}, 2^{17}), a Hamming distance $w = 32$ (28, 25, 22, 20, 19) is sufficient to reach a success probability of 99.9%.

4.4.7 Subsequence Problem

Eve is given two fixed bit sequences, $s^{\mathbf{E}\leftrightarrow\mathbf{B}}$ (sifted key that Eve wants to achieve) and $d^{\mathbf{A}}$ (the raw key of Alice). Her goal is to find a *subsequence* of $d^{\mathbf{A}}$ that coincides with $s^{\mathbf{E}}$.

Algorithm that finds a subsequence. First we give a simple algorithm that takes two sequences $s = s_1|s_2|\dots|s_m$, $S = S_1|S_2|\dots|S_n$ as inputs and returns the index set $\mathcal{J} = \{j_1, \dots, j_m\} = \{j_i : S_{j_i} = s_i\}$ if s is a subsequence of S (denoted $s \preccurlyeq S$).

Probability for finding a subsequence in a random sequence. We assume that both sequences consist of i.i.d. Bernoulli trials with $p(0) = p(1) = 1/2$ and calculate the (success) probability that $s \preccurlyeq S$.

$s \preccurlyeq S$ iff S is of the form

$$S = \bar{s}_1|\dots|\bar{s}_1|s_1|\bar{s}_2|\dots|\bar{s}_2|s_2|\dots|\bar{s}_m|\dots|\bar{s}_m|s_m|x_1|x_2|\dots. \tag{4.4}$$

Here, $\bar{s}_j$ denotes the negation of s_j (written above as s_j to improve readability), while each x_i can independently take value 0 or 1. All sequences $\bar{s}_j|\dots|\bar{s}_j$ are optional. Let S

Algorithm 4.2: Find Subsequence (s, S)

```
Input: two non-empty binary sequences $K$ and $S$.
Output: index set $\mathcal{J}$ if $K$ is a subsequence of $S$, else $\emptyset$.
1 $i \leftarrow 1$, $j \leftarrow 1$, $m \leftarrow \text{length}(s)$, $n \leftarrow \text{length}(S)$, $\mathcal{J} \leftarrow \emptyset$
2 do
3     if $K_i = S_j$ then
4         $\mathcal{J} \leftarrow \mathcal{J} \cup \{j\}$                //store position
5         $i \leftarrow i+1$                  //compare next bit of $s$
6     $j \leftarrow j+1$                      //compare next bit of $S$
7 while ($i \leq m$ and $j \leq n$)      //neither end of $s$ nor end of $S$ reached
8 if $i \leq m$ then return $\emptyset$  //end of $s$ not reached, but end of $S$ reached
9 return $\mathcal{J}$
```

be the *number of different* valid sequences, i.e. sequences S, that contain s as a subsequence. Obviously S does not depend on s, but only on m and n. To calculate S we can therefore choose s to be the all zero sequence of length m. Consequently, S is equal to the number of different binary sequences of length n that contain at least m zeroes. The success probability

$$\text{Prob}\{s \preccurlyeq S\} = \mathsf{S}/2^n = 2^{-n} \sum_{l=m}^{n} \binom{n}{l}. \tag{4.5}$$

Application to Eve's attack. Note that Eve wants to find the sifted key $s^{\mathbf{E}\leftrightarrow\mathbf{B}}$ in Alice's raw key $d^{\mathbf{A}}$. If both bases are used with equal probability (as in standard symmetric BB84), then $m \approx n/2$. Obviously,

$$\text{Prob}\{s \preccurlyeq S\} > \frac{1}{2} \Leftarrow m \leq \lfloor n/2 \rfloor. \tag{4.6}$$

However, it is not necessary, that s is an *exact* subsequence of S. We can allow for some errors that will be removed during the subsequent error correction step. Using Hoeffding's inequality (Theorem 1 in Ref. [292]) we can give a non-tight (but exponential) lower bound on $\text{Prob}\{\tilde{s} \preccurlyeq S\}$ if we allow for approximately k errors in the resulting subsequence $\tilde{s}$:

$$\text{Prob}\{\tilde{s} \preccurlyeq S\} \geq 1 - \exp\left(-\frac{2k^2}{n}\right) \Leftarrow \tilde{m} = \lfloor n/2 \rfloor - k. \tag{4.7}$$

Here, only $\tilde{m}$ bits of s form a subsequence of S. For moderate values of k this probability reaches almost unity.

4.4.8 On the Practicality of the Attack

Finally, we are able to give some concrete figures for attacking the presented BB84 protocol with typical parameters as used in our various experiments.

Attacking the sifting stage—hiding in the noise. Let us assume that during the sifting stage the legitimate parties will exchange messages that contain one bit per preparation/measurement basis (time slot). Let us assume further that Eve can successfully attack the protocol (as discussed above), if she can substitute such a message, say $m^{\mathbf{A}}$, with a sifting message of her choice, say $m^{\mathbf{E}}$. From Corollary 4.4.2 it follows that if Eve replaces $m^{\mathbf{A}}$ with $\tilde{m}^{\mathbf{E}}$ instead of $m^{\mathbf{E}}$, she will introduce at this step (at most) an additional error $\varepsilon = w/\ell \approx 0.78\%$ (0.34%, 0.15%, 0.067%, 0.031%, 0.014%) (with parameters from above; in the worst case each modified basis bit could result in one flipped sifted key bit). This strategy allows Eve to hide the substitution of sifting messages in the usual noise on the quantum channel, since the following error correction step will also remove these small additional deviations. Obviously, the larger the message length ℓ, the easier Eve's task is.

Correlating the sifted keys of Alice and Bob. Assume for the moment that Eve has intercepted the quantum bits from Alice and has saved them into her quantum memory. Assume further that she managed to fool Alice, so that Alice announces her the corresponding preparation bases. Then Eve can measure the quantum bits and get Alice's raw key.

The strongest of the presented attacks is based on the fact that once Eve knows the raw key of Alice, she can by using a modification of Bob's sifting message ensure with high probability that the complete sifted key of Alice will be almost identical to that of Bob (cf. description of Protocol 1 and step (Se”) of the attack against it.).

Lemma 4.4.3. Let $d^{\mathbf{A}} \in_R \{0,1\}^n$ be the raw key that Alice has used to prepare her quantum bits. Once Eve knows $d^{\mathbf{A}}$ she can determine $\lfloor n/2 \rfloor - k$ bits of any fixed sifted key $s^{\mathbf{E}}$ that she wants Alice to create with probability

$$\mathcal{P}^{\text{succ}}_{\text{sift-attack}} \geq 1 - \exp\left(-\frac{2k^2}{n}\right) \tag{4.8}$$

by replacing Bob's sifting message with a message $b^{\mathbf{A}=\mathbf{E}}$ that she has prepared.

Eve's attack will succeed if a *subsequence* of $s^{\mathbf{E}}$ (derived by deleting some elements without changing the order) of length at least $\lfloor n/2 \rfloor - k$ is also a *subsequence* of $d^{\mathbf{A}}$. The proof and a simple and efficient algorithm to generate $b^{\mathbf{A}=\mathbf{E}}$ is given in 4.4.7. Note, that $k = O(\sqrt{n})$ is sufficient for $\mathcal{P}^{\text{succ}}_{\text{sift-attack}} \approx 1$.

The presented attack also works without access to a quantum memory for Eve and immediate measurement of Alice's signal with random basis selection, however, it will results in a smaller success probability in the ideal transition model for the subsequence problem. Nevertheless, if additional attenuation as found in practical systems is assumed, the same arguments can be used to achieve a probability of almost one for a successful attack.

Summary. We have shown that the attack strategy presented is feasible for a powerful adversary and not only of theoretical interest. Being able to generate approximate messages in the sifting stage and to correlate different raw keys by subsequence selection fully breaks the proposed approach for all possible protocol variations. If the modifications can be hidden in the noise—which is later corrected for in the post-processing steps—it enables Eve to successfully mount a man-in-the-middle attack for the proposed non-ITS authentication protocol. Although, different countermeasures have been developed to cope with the presented approach, it is recommended to only rely on a ITS-MAC for future designs and not use the researched non-ITS protocol. The proposed countermeasures are still not ITS and even worse, they have to be designed for particular protocol variants and are not generally viable.

4.5 QKD Integration and Discussion

In the last part of this chapter, we discuss how QKD systems can be integrated with different applications. We will quickly explain how QKD integrates with secure channel protocols in its own right. Additionally, we analyze how secure channel can support long-term secure storage and advanced data sharing scenarios, as demonstrated in a medical use case. Finally, we look how QKD can augment MPC. In general, a huge effort is spent on all levels to push integration of QKD technology [293] and we will only briefly discuss most relevant options.

4.5.1 QKD for Secure Channels

For the realization of secure communication based on QKD two major scenarios should be distinguished which differ in the underlying connectivity on the quantum layer, i.e., direct or indirect.

Link encryption with QKD. The most evident usage for QKD generated keys is link encryption, i.e., the implementation of a secure channel between two peers directly connected through a quantum channel. Integration of QKD on all layers of the network stack have been shown in the past ranging from layer 2 MACsec [294] to IPsec [57] up to transport layer TLS [118] and QUICK [119]. In essence, they follow a common idea and use QKD keys during the key exchange phase where subsequent symmetric encryption keys are negotiated. QKD is therefore used as an ITS replacement for asymmetric key exchange mechanisms (KEM) used in respective protocols. Although it would be possible to use QKD keys for one-time pad encryption (OTP), this mode of operation cannot be widely found in the field, because of the limited key bandwidth and compatibility issues with existing protocols.

However, recently a new trend emerged for QKD integration into link encryption. Contrary to replacing key exchange on an ad hoc basis, novel provably secure hybrid protocols have been presented which allow for a cryptographically sound combination of keys generated by different methods. A hybrid key exchange has been proposed in [295], which seamlessly combine the different keys such that the strength of the different technologies are preserved. This approach has also been extended to post-quantum cryptography [296], such that we don't have to choose between technologies but can use them concurrently to get the best of all worlds.

Quantum key distribution networks. One of the main problems of QKD from a user perspective is the intrinsic distance limitation for direct peer to peer key generation. The limitation stems from the attenuation encountered when transmitting photons through media together with the security properties of QKD protocols. Given that the security of QKD is based on the no-cloning property of quantum information which prevents from perfectly copying qbits, this property also prevents from amplifying signals along the way. The only possibility to overcome this problem would be by usage of quantum repeaters, however, quantum repeaters are not expected to be practical in the distant future [245]. Therefore, alternative solutions were developed in the last years to address the limitations in the connectivity of QKD.

On the one hand, *trusted repeater networks* have been proposed and demonstrated. If longer distances have to be bridged and secure channels have to be established between cities, countries or even continents, the physical limitations of QKD introduce real challenges. Trusted repeater networks are currently the only feasible solutions to overcome the distance limitation and the first step in building a quantum internet [297]. The basic idea behind trusted repeater networks is to concatenate QKD links on the classical layer, i.e., key management layer. Trusted repeaters nodes are generating keys with adjacent nodes in a network but have to translate encrypted messages from one link key to another, if secure traffic is forwarded. Thus, in pure QKD trusted repeater networks, any repeater can see the plaintext of forwarded messages and needs to be fully trusted, as the name already suggests. Hence, end-to-end security is not possible in this type of network and security depends on the implementation and operation of intermediary nodes, which is a substantial downgrade. The first trusted repeater networks were presented in [53, 298] and are still under active development. Additionally, tamper-proof hardware for long distance hop-by-hop forwarding has been proposed to operate trusted repeater even in less trusted environments.

On the other hand, practical *end-to-end security for QKD networks* is researched to overcome the limitations of trusted repeater networks with different strategies. If peers in a trusted repeater network communicate over multiple disjoint path and encode the messages with secret sharing, ITS secure communication can be re-established under the non-collusion assumption. The concept is known as perfectly secure message transmission (SMT) and is well understood, but was never applied in practice for conventional networks. However, it is very attractive for QKD networks. If adequate routing and forwarding mechanisms are developed, it can significantly increase security for larger QKD networks which are densely connected.

Alternatively, space based long-distance QKD has been proposed in the literature and already experimentally demonstrated. However, form a networking perspective this is still a trusted repeater network, because the satellite as well as the ground stations will be trusted intermediaries for practical applications.

Additionally, end-to-end security for QKD networks can also be achieved in combination with post-quantum cryptography (PQC), although not in an ITS way. Nevertheless, this combination can be useful in many application scenarios, e.g., on the last mile in access networks or in wireless radio networks, and can also be used within the QKD control plane or to provide end-to-end authentication. If combined in a seamless way, it can complement QKD where optical channels are not available but still ensure quantum-safe security.

4.5.2 QKD Integration with Secure Storage

After discussing ITS secure communication, it is natural to ask how it can be integrated with the proposed storage solution. In fact, the integration turns out to be straight forward. The ARCHISTAR storage system assumed ITS secure channels to prevent from attacks during up-/download operations. Thus, if QKD is used to secure the links, a fully ITS system can be achieved.

We evaluated the combination in a demonstration for ITS secure sharing of medial data. The use case was based on a digital pathology workflow where different pathologists and additional healthcare professionals are involved [299]. To achieve ITS secure data sharing between two hospitals we combined the ARCHISTAR-Proxy from Section 2.4 with QKD link encryptors to do a real-world evaluation. In Figure 4.17 the network structure and geographical distribution is shown. We used two data center which were connected to two hospitals each over dedicated QKD links. On each side we run a ARCHISTAR-Proxy configured with 2-out-of-2 secret sharing. In the demonstration, medical institutions from two different organizations were collaborating on digital pathology data in real-time. Given the huge amount of data produced in digital pathology, mainly due to managing high resolution whole slide images (WSI), not only ITS secure schemes were used but also alternative quantum-safe symmetric bulk encryption methods. This could be achieved by defining different security policies for different buckets in the proxy service. In particular, perfectly secure secret sharing (PSS) was used for sensitive information and computation secure algorithms (CSS) for

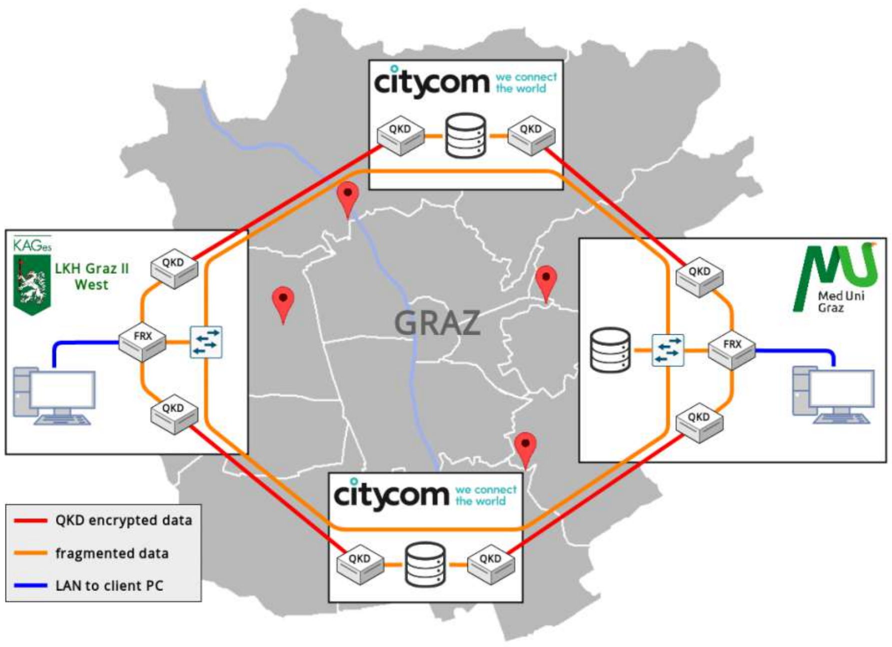


Figure 4.17: Basic concept of OpenQKD use case in Graz dealing with data sharing of medical information between two hospitals (cf. [115]).

bulk data which was less sensitive but still contained a lot of private information about patients. Additionally, different access control policies were enforced by using read only properties for certain buckets, thus further reducing the attack surface.

The basic structure of the deployment for the demonstrator is shown in Figure 4.18. It was tested for about 2 month and the final results from a user perspective were presented in [115]. From a technical perspective, we achieved to fulfill the user requirements and show ITS secure data sharing platform as feasible and can even increase user acceptance.

Finally, looking at the communication pattern in detail, it shows that we it also resembled non-interactive (single round) secure message transmission [300]. Data communicated between the hospitals are secret shared and communicated over two disjoint paths leaving all QKD peers without information as long as they don't collude. However, the options and possibilities are much richer compared to a configuration using plain SMT to connect the hospitals and a local server for data storage.

4.5.3 QKD Integration with Multiparty Computation

In the previous section we showed how QKD can be integrated with secure channel protocols and more complex data storage and sharing scenarios. Therefore, it is natural to ask how QKD can improve security in secure data processing scenarios. For this thesis we did not do any evaluation in a real testbed, but we show first results of our

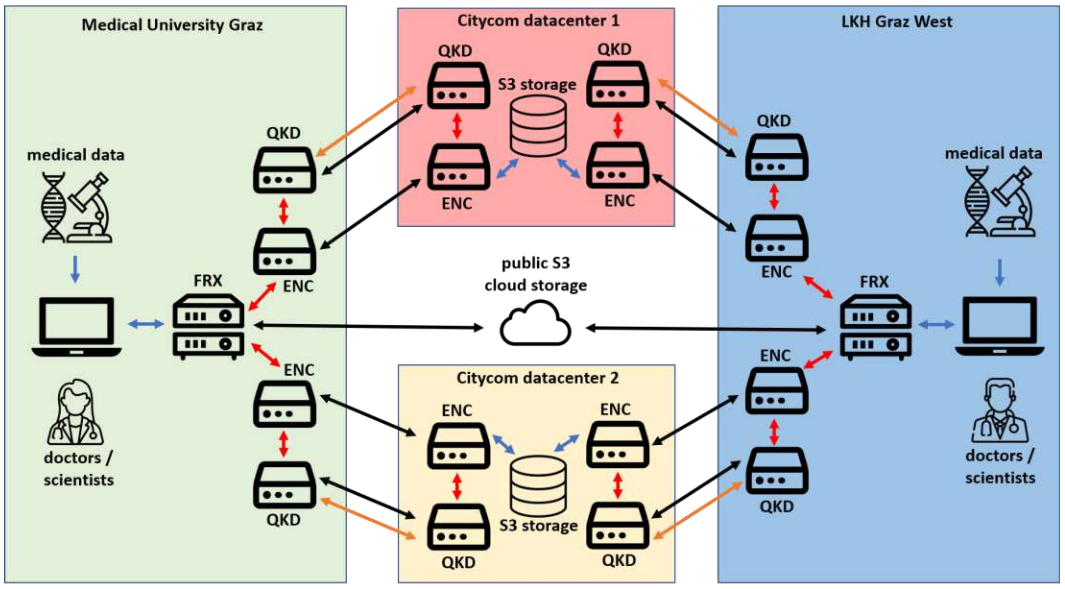


Figure 4.18: Components used to achieve secure data storage and sharing in the cloud based on the combination of secret sharing and QKD. Data shared between hospitals flows securely over two disjoint paths protected by hop-by-hop QKD based link encryption. The use case is leveraging the ARCHISTAR-Proxy as explained in [115].

research. In the following we give a quick overview of ways to achieve QKD assisted MPC, i.e., we show ways to leverage QKD for improved MPC security.

As discussed in Chapter 3, MPC protocols with ITS security are based on secret sharing. In that sense it is an ITS method for privacy preserving computation. However, all ITS secure protocols rely on the existence of secure channels which cannot be realized from traditional cryptography. ITS secure key agreement from algorithmic approaches is not possible. Therefore, the natural way is to use QKD to establish ITS secure channels to be used for MPC communication. This combination is straight forward and is feasible, similar to the extension of the distributed storage system. The only thing to do is to estimate the communication complexity for the compute task in question in order to see if the respective QKD system is capable to deliver the required amount of key.

Interestingly, there is another important point of contact in MPC protocols where QKD can help indirectly. This is due to the fact, that QKD is technically related to quantum oblivious transfer (QOT). In a nutshell, the very same hardware can be used to realize either functionality (QKD or QOT) by simply modifying the post-processing protocol. Therefore, QOT is another primitive from the quantum world which can be practically realized. Its security is well understood [301] and has also been shown composable [302], although only under hardness of symmetric cryptography. Nevertheless, it is an improvement compared to standard OT which requires asymmetric cryptography.

QOT can be used in the pre-processing phase of dishonest majority MPC protocols. Especially SPDZ [182], which uses an ITS-MAC in combination with secret sharing in its online phase, can benefit from the extension with QOT. If QOT is used in an OT type of pre-processing protocol like MASCOT [303], the security can be increased, because no asymmetric cryptography is required anymore. However, more research is required to understand the detailed requirements, assumptions, and consequences for the mentioned combination.

Chapter 5

Summary, Conclusion and Outlook

This work is dedicated to the direct application of information-theoretic cryptography in modern ICT applications. The goal was to leverage the strong security guarantees against an unbounded adversary and provide systems with long-term security. Such systems would not be affected by the advent of scalable quantum computers or any other future computing paradigm. However, pure ITS systems may not always be necessary and there are always some trade-offs to make. Nevertheless, we wanted to build systems which preserve the ITS core properties as good as possible but are at the same time practically efficient and relevant. As a rule, we wanted to keep free from the use of asymmetric cryptography and tried to push the boundary for symmetric systems. In fact, the systems designed are distributed and rely their security on the non-collusion assumption rather than on number theoretical assumptions.

During the work on this thesis post-quantum cryptography made enormous progress, which will replace currently used asymmetric cryptography in the near future. However, there is still a lot of research ahead on the topic and first candidates were just recently standardized by National Institute of Standards and Technology after a four-year competition. There is also still a lot of uncertainty in the hardness assumptions used for new algorithms when it comes to resistance against quantum computers. Relying only on ITS and symmetric cryptography was therefore considered a conservative approach and we explored how far we can get in terms of practical systems.

Interestingly, we were able to build efficient systems with strong security for all three data protection realms in the data lifecycle. We researched ways to protect data at rest, during processing and in transit and designed new systems and platforms to directly leverage ITS primitives to achieve long-term security.

Firstly, to protect *data at rest* we designed a novel secure multi-cloud storage system by integrating secret sharing with a Byzantine fault-tolerant protocol. We analyzed and optimized performance on the network layer as well as researched efficient software and hardware encoding engines for most relevant secret sharing schemes. Additionally, we developed novel efficient mechanisms to remotely check data consistency with requiring only little communication on the network and simple computation at the severs. The protocol can be even executed by untrusted parties and therefore enables public auditing. Additionally, the batching auditing approach has also been extended to provide efficient batch verifiable secret sharing. In essence, we were able to show that modern data sharing applications can be realized in a multi-cloud setting with strong security guarantees and advanced features.

Secondly, to protect *data during processing* we leveraged information-theoretic multiparty computation to build novel decentralized but privacy preserving marketplaces. We showed the feasibility of using publicly verifiable multiparty computation to achieve end-to-end authenticity for auction platforms in smart manufacturing. Adding verifiability was necessary due to the limited scalability of MPC for serious computational tasks which prevented users in the system from running their own nodes. Being able to verify the correctness of a privacy preserving computation turned out to be important to strengthen users' trust into the system and increase their willingness to participate. We also showed that even multi-stakeholder optimizations are possible for solving the assignment problem in a dedicated use case for air traffic management. In summary, we were able to show that MPC nicely extends our work on secret sharing based storage and can be used to build modern decentralized data markets with built-in privacy and data sovereignty, but also end-to-end authenticity.

Finally, to protect *data in transit* we worked on certain aspects of quantum key distribution, which is mainly used as link encryption tool in combination with one-time-pad encryption or symmetric ciphers. Because it is the only known ITS key exchange protocol, it nicely complements the work done on the other topics. Concretely, for a QKD system at AIT we developed a software-based synchronization system which is computationally very efficient and suitable to run on constrained embedded devices. To further reduce the computational effort on QKD devices we researched ways to offload computationally demanding tasks of post-processing in QKD, i.e., we showed that offloading error correction to a single untrusted sever is possible for discrete variable QKD. These results should help to increase energy efficiency of QKD devices and make them more flexible in usage. Unfortunately, we could not confirm the security of a very efficient message authentication protocol introduced in earlier work, instead we were able to develop an attack which fully broke the protocol based non-ITS MAC. Finally, we analyzed possibilities to combine QKD with other results in this work. In fact, we were able to demonstrate the combination of the ARCHISTAR-Proxy with QKD in a real-world medical use case, where digital pathology data was exchanged between two hospitals over multiple clouds and paths.

In conclusion, we achieved very positive research results and show that ITS cryptography can contribute to the security of modern ICT systems. Emerging applications in the Internet of Things are well suited due to its distributed nature and open up many new opportunities. By leveraging the non-collusion assumption, it is possible to design systems which are long-term secure and can also provide increased robustness. The achieved performance figures for evaluated use cases are very promising and the limitations were mainly induced by communication. Additionally, using multiple clouds is getting more and more the norm and many infrastructure providers are already available to really implement the proposed concepts. In fact, two start-ups already licensed results of this thesis and are working on the commercialization thereof. The company FragmentiX[1] is using the research results in the domain of multi-cloud storage to develop novel storage products, also in combination with QKD as demonstrated in the medical use case. The company Catch.Direct[2] is using our ideas on end-to-end verifiable MPC to develop a commercial data market for outsourcing manufacturing capacities. Not to forget the enormous progress made in the development of QKD networks on a European scale, which are also expected to be available for customers within the next years. Hence, we believe that the research results are very relevant and have the potential to be integrated into commercial solutions in the upcoming years.

We also identified many interesting new topics which will hopefully be covered in future work. The performance models developed for BFT protocols can be further advanced in different directions, e.g., modeling latency or view changes. Additionally, many new BFT protocols have been proposed in recent years which may also benefit from our modelling approach. Interestingly, during our work with MPC implementations we also found that the network layer is not accurately covered and especially the interplay with topics from the BFT world are not considered, e.g., correct broadcast protocol implementations. We think, that researching at the interface between BFT and MPC can lead to many new insights and better software implementations. Also, the extension with verifiability aspects for both, storage and processing, turned out to be very fruitful an should be continued. It can be a strong argument to foster collaboration of mutually distrusted parties. In the future, the development of NIZK systems which provide quantum-safe soundness together with perfect zero-knowledge will be important, which should be transitioned to once quantum computers are really available. Finally, most interesting topics to push forward from our work on QKD seem the combination of MPC with QOT based on QKD hardware, as well as MPC based offloading of post-processing to reduce the attack surface of QKD devices.

[1]`https://fragmentix.com/`, accessed 2023-01-10.
[2]`https://www.catch.direct/`, accessed 2023-01-10.

Chapter 6

Bibliography

Thesis Results Published

In this section the publications resulting from research during this thesis are listed. The author substantially contributed to published work below and results achieved by the author are part of the thesis.

[1] Thomas Lorünser, Andreas Happe, and Daniel Slamanig. "ARCHISTAR: Towards Secure and Robust Cloud Based Data Sharing". In: *7th IEEE International Conference on Cloud Computing Technology and Science, CloudCom 2015, Vancouver, BC, Canada, November 30 - December 3, 2015*. IEEE Computer Society, 2015, pp. 371–378. DOI: `10.1109/CloudCom.2015.71`.

[2] Thomas Lorünser, Benjamin Rainer, and Florian Wohner. "Towards a Performance Model for Byzantine Fault Tolerant Services". In: *Proceedings of the 12th Intern. Conference on Cloud Computing and Services Science, CLOSER 2022,, Online Streaming, April 27-29, 2022*. Ed. by Maarten van Steen, Donald Ferguson, and Claus Pahl. SCITEPRESS, 2022, pp. 178–189. DOI: `10.5220/0011041600003200`.

[3] Andreas Happe, Florian Wohner, and Thomas Lorünser. "The Archistar Secret-Sharing Backup Proxy". In: *Proceedings of the 12th International Conference on Availability, Reliability and Security, Reggio Calabria, Italy, August 29 - September 01, 2017*. ACM, 2017, 88:1–88:8. DOI: `10.1145/3098954.3104055`.

[4] Jakob Stangl, Thomas Lorünser, and Sai Manoj Pudukotai Dinakarrao. "A fast and resource efficient FPGA implementation of secret sharing for storage applications". In: *2018 Design, Automation & Test in Europe Conference & Exhibition, DATE 2018, Dresden, Germany, March 19-23, 2018*. Ed. by Jan Madsen and Ayse K. Coskun. IEEE, 2018, pp. 654–659. DOI: `10.23919/DATE.2018.8342091`.

[5] Thomas Lorüenser and Stephan Krenn. "Method for checking the availability and integrity of a distributed data object". Austrian pat. AT518910B1. 2016.

[6] Denise Demirel et al. "Efficient and Privacy Preserving Third Party Auditing for a Distributed Storage System". In: *11th International Conference on Availability, Reliability and Security, ARES 2016, Salzburg, Austria, August 31 - September 2, 2016*. IEEE Computer Society, 2016, pp. 88–97. DOI: `10.1109/ARES.2016.88`.

[7] Stephan Krenn, Thomas Lorünser, and Christoph Striecks. "Batch-verifiable Secret Sharing with Unconditional Privacy". In: *Proceedings of the 3rd Intern. Conference on Information Systems Security and Privacy, ICISSP 2017, Porto, Portugal, February 19-21, 2017*. Ed. by Paolo Mori, Steven Furnell, and Olivier Camp. SciTePress, 2017, pp. 303–311. DOI: `10.5220/0006133003030311`.

[8] Thomas Lorünser and Florian Wohner. "Performance Comparison of Two Generic MPC-frameworks with Symmetric Ciphers". In: *Proceedings of the 17th International Joint Conference on e-Business and Telecommunications, ICETE 2020 - Volume 2: SECRYPT, Lieusaint, Paris, France, July 8-10, 2020*. Ed. by Pierangela Samarati et al. ScitePress, 2020, pp. 587–594. DOI: `10.5220/0009831705870594`.

[9] Thomas Lorünser, Florian Wohner, and Stephan Krenn. "A Privacy-Preserving Auction Platform with Public Verifiability for Smart Manufacturing". In: *Proceedings of the 8th International Conference on Information Systems Security and Privacy, ICISSP 2022, Online Streaming, February 9-11, 2022*. Ed. by Paolo Mori, Gabriele Lenzini, and Steven Furnell. SCITEPRESS, 2022, pp. 637–647. DOI: `10.5220/0011006700003120`.

[10] Thomas Lorünser, Florian Wohner, and Stephan Krenn. "A Verifiable Multiparty Computation Solver for the Linear Assignment Problem: And Applications to Air Traffic Management". In: *Proceedings of the 2022 on Cloud Computing Security Workshop, CCSW 2022, Los Angeles, CA, USA, 7 November 2022*. Ed. by Francesco Regazzoni and Marten van Dijk. ACM, 2022, pp. 41–51. DOI: `10.1145/3560810.3564263`.

[11] Thomas Lorünser, Andreas Happe, and Andreas Poppe. "Timing synchronization with photon pairs for quantum communications". In: *2013 Conference on Lasers and Electro-Optics Europe and International Quantum Electronics Conference, CLEO/Europe-IQEC 2013*. San Jose, CA , USA, May 2013, p. 1. DOI: `10.1109/CLEOE-IQEC.2013.6801698`.

[12] Thomas Lorünser et al. "Method for creating and distributing cryptographic keys". Austrian pat. AT519476B1. 2017.

[13] Thomas Lorünser et al. “On the Security of Offloading Post-Processing for Quantum Key Distribution”. In: *Entropy* 25.2 (2023). ISSN: 1099-4300. DOI: 10.3390/e25020226.

[14] Christoph Pacher et al. “Attacks on quantum key distribution protocols that employ non-ITS authentication”. In: *Quantum Inf. Process.* 15.1 (2016), pp. 327–362. DOI: 10.1007/s11128-015-1160-4.

Related Publications and Early Work

Additional publications of the author published during the last years but not directly part of the thesis are listed below together with research work published in his early career.

[15] Christoph G. Schuetz et al. “A Distributed Architecture for Privacy-Preserving Optimization Using Genetic Algorithms and Multi-party Computation”. In: *Cooperative Information Systems - 28th International Conference, CoopIS 2022, Bozen-Bolzano, Italy, October 4-7, 2022, Proceedings*. Ed. by Mohamed Sellami et al. Vol. 13591. Lecture Notes in Computer Science. Springer, 2022, pp. 168–185. DOI: 10.1007/978-3-031-17834-4_10.

[16] Christoph G. Schuetz et al. “An Auction-Based Mechanism for a Privacy Preserving Marketplace for ATFM Slots”. In: *ICAS Congress 2022*. 2022, pp. 1–14.

[17] Stephan Krenn and Thomas Lorünser. “Single-Use Delegatable Signatures Based on Smart Contracts”. In: *ARES 2021: The 16th International Conference on Availability, Reliability and Security, Vienna, Austria, August 17-20, 2021*. Ed. by Delphine Reinhardt and Tilo Müller. ACM, 2021, 40:1–40:7. DOI: 10.1145/3465481.3469192.

[18] Thomas Lorünser, Christoph G. Schütz, and Eduard Gringinger. “SlotMachine - A Privacy-preserving Marketplace for Slot Management”. In: *ERCIM News* 2021.126 (2021).

[19] Christoph G Schuetz et al. “A Privacy-Preserving Marketplace for Air Traffic Flow Management Slot Configuration”. In: *2021 IEEE/AIAA 40th Digital Avionics Systems Conference (DASC)*. 2021, pp. 1–9. DOI: 10.1109/DASC52595.2021.9594401.

[20] Thomas Lorünser, Niki Dürk, and Stephan Puxkandl. “FLEXPROD - Flexible Optimizations of Production with Secure Auctions”. In: *ERCIM News* 2020.123 (2020).

[21] Thomas Lorünser, Stephan Krenn, and Roland Kammerer. "High Performance Software Defined Storage for the Cloud". In: *ERCIM News 119* (2019). `https://ercim-news.ercim.eu/en119/research-and-innovation/high-performance-software-defined-storage-for-the-cloud`.

[22] Erik Framner et al. "Making secret sharing based cloud storage usable". In: *Inf. Comput. Secur.* 27.5 (2019). DOI: `10.1108/ICS-01-2019-0016`.

[23] Leon Sell, Henrich C. Pöhls, and Thomas Lorünser. "C3S: Cryptographically Combine Cloud Storage for Cost-Efficient Availability and Confidentiality". In: *2018 IEEE International Conference on Cloud Computing Technology and Science, CloudCom 2018, Nicosia, Cyprus, December 10-13, 2018*. IEEE Computer Society, 2018, pp. 230–238. DOI: `10.1109/CloudCom2018.2018.00052`.

[24] Thomas Lorünser, Eva María Muñoz, and Marco Decandia Brocca. "Secure and Robust Multi-Cloud Storage for the Public Sector". In: *ERCIM News* 2018.114 (2018).

[25] Stephan Krenn, Thomas Lorünser, and Christoph Striecks. "Engineering Cryptography for Security and Privacy in the Cloud". In: *ERCIM News* 2018.113 (2018).

[26] David Derler et al. "Revisiting Proxy Re-encryption: Forward Secrecy, Improved Security, and Applications". In: Lecture Notes in Comp. Science 10769 (2018). Ed. by Michel Abdalla and Ricardo Dahab, pp. 219–250. DOI: `10.1007/978-3-319-76578-5\_8`.

[27] Stephan Krenn et al. "Towards Attribute-Based Credentials in the Cloud". In: *Cryptology and Network Security - 16th International Conference, CANS 2017, Hong Kong, China, November 30 - December 2, 2017, Revised Selected Papers*. Ed. by Srdjan Capkun and Sherman S. M. Chow. Vol. 11261. Lecture Notes in Computer Science. Springer, 2017, pp. 179–202. DOI: `10.1007/978-3-030-02641-7\_9`.

[28] Thomas Lorünser et al. "CryptSDLC: Embedding Cryptographic Engineering into Secure Software Development Lifecycle". In: *Proceedings of the 13th International Conference on Availability, Reliability and Security, ARES 2018, Hamburg, Germany, August 27-30, 2018*. Ed. by Sebastian Doerr et al. ACM, 2018, 4:1–4:9. DOI: `10.1145/3230833.3233765`.

[29] Pasquale Chiaro et al. "Secure and Privacy-Friendly Storage and Data Processing in the Cloud". In: *Privacy and Identity Management. The Smart Revolution - 12th IFIP WG 9.2, 9.5, 9.6/11.7, 11.6/SIG 9.2.2 International Summer School, Ispra, Italy, September 4-8, 2017, Revised Selected Papers*. Ed. by Marit Hansen et al. Vol. 526. IFIP Advances in Information and Communication Technology. Springer, 2017, pp. 153–169. DOI: `10.1007/978-3-319-92925-5\_10`.

[30] Erkuden Rios et al. *Cloud technology options towards Free Flow of Data*. Tech. rep. AKKA, France, 2017. DOI: 10.13140/RG.2.2.34366.38724.

[31] Thomas Lorünser et al. "Wirkungsvolle Kryptographie-Lösungen für Cloud Computing". In: *Elektrotechnik und Informationstechnik* 134.7 (Nov. 2017), pp. 364–369. ISSN: 0932383X. DOI: 10.1007/s00502-017-0519-x.

[32] Andreas Happe, Stephan Krenn, and Thomas Lorünser. "Malicious Clients in Distributed Secret Sharing Based Storage Networks". In: *Security Protocols XXIV - 24th International Workshop, Brno, Czech Republic, April 7-8, 2016, Revised Selected Papers*. Ed. by Jonathan Anderson et al. Vol. 10368. Lecture Notes in Computer Science. Springer, 2016, pp. 206–214. DOI: 10.1007/978-3-319-62033-6_23.

[33] Daniel Slamanig, Agi Karyda, and Thomas Lorünser. "PRISMACLOUD - Privacy and Security Maintaining Services in the Cloud". In: *ERCIM News* 2016.104 (2016).

[34] Alaa Sarah Alaqra et al. "Signatures for Privacy, Trust and Accountability in the Cloud: Applications and Requirements". In: *Privacy and Identity Management. Time for a Revolution? - 10th IFIP WG 9.2, 9.5, 9.6/11.7, 11.4, 11.6/SIG 9.2.2 International Summer School, Edinburgh, UK, August 16-21, 2015, Revised Selected Papers*. Ed. by David Aspinall et al. Vol. 476. IFIP Advances in Information and Communication Technology. Springer, 2015, pp. 79–96. DOI: 10.1007/978-3-319-41763-9_6.

[35] Andreas Happe and Thomas Lorünser. "Exchanging Database Writes with modern Cryptography". In: *The First International Conference on Advances in Cyber-Technologies and Cyber-Systems CYBER2016*. Italy, 2016.

[36] Bernd Zwattendorfer, Stephan Krenn, and Thomas Lorünser. "Secure and Privacy-Preserving Identity Management in the Cloud". In: *ERCIM News* 2016.104 (2016).

[37] Thomas Lorünser et al. "A New Architecture for Developing Cryptographic Cloud Services". In: *ERCIM News* 2016.106 (2016).

[38] Bernhard Schrenk et al. "Fully-passive resiliency switch for agile PON restoration". In: *Optical Fiber Communications Conference and Exhibition, OFC 2015, Los Angeles, CA, USA, March 22-26, 2015*. IEEE, 2015, pp. 1–3. DOI: 10.1364/OFC.2015.W1J.6.

[39] Thomas Lorünser, Thomas Länger, and Daniel Slamanig. "Cloud Security and Privacy by Design". In: *E-Democracy - Citizen Rights in the World of the New Computing Paradigms - 6th Intern. Conference, E-Democracy 2015, Athens, Greece, December 10-11, 2015, Proceedings*. Ed. by Sokratis K. Katsikas and

Alexander B. Sideridis. Vol. 570. Communications in Computer and Information Science. Springer, 2015, pp. 202–206. DOI: 10.1007/978-3-319-27164-4_16.

[40] Bernhard Schrenk et al. "Passive ROADM Flexibility in Optical Access With Spectral and Spatial Reconfigurability". In: *IEEE J. Sel. Areas Commun.* 33.12 (2015), pp. 2837–2846. DOI: 10.1109/JSAC.2015.2478719.

[41] Thomas Lorünser et al. "Towards a New Paradigm for Privacy and Security in Cloud Services". In: *Cyber Security and Privacy - 4th Cyber Security and Privacy Innovation Forum, CSP Innovation Forum 2015, Brussels, Belgium, April 28-29, 2015, Revised Selected Papers*. Ed. by Frances Cleary and Massimo Felici. Vol. 530. Communications in Computer and Information Science. Springer, 2015, pp. 14–25. DOI: 10.1007/978-3-319-25360-2_2.

[42] Aleksandar Hudic et al. "A Multi-layer and MultiTenant Cloud Assurance Evaluation Methodology". In: *IEEE 6th International Conference on Cloud Computing Technology and Science, CloudCom 2014, Singapore, December 15-18, 2014*. IEEE Computer Society, 2014, pp. 386–393. DOI: 10.1109/CloudCom.2014.85.

[43] Bernhard Schrenk, Thomas Lorünser, and Thomas Zemen. "Towards spectrum-programmable, mesh-enabled mobile xHaul through reconfigurable WDM overlay in fully-passive networks". In: *17th International Conference on Transparent Optical Networks, ICTON 2015, Budapest, Hungary, July 5-9, 2015*. IEEE, 2015, pp. 1–4. DOI: 10.1109/ICTON.2015.7193593.

[44] Oliver Maurhart et al. "New release of an open source QKD software: design and implementation of new algorithms, modularization and integration with IPSec". In: *QCRYPT 2013, 3rd Annual Conference on Quantum Cryptography*. University of Waterloo, Canada, 2013.

[45] Oliver Maurhart et al. "Quantum Key Distribution Software maintained by AIT". In: *QCMC - 11th International Conference on Quantum Communication, Measurement and Computing*. Vienna, Austria, 2012, p. 247.

[46] Oliver Maurhart et al. "QKD software architecture and systems integration with classical communication infrastructure". In: *QCRYPT 2012 - 2nd international conference on quantum cryptography*. Singapur, 2012.

[47] Aysajan Abidin et al. "Quantum cryptography and authentication with low key-consumption". In: *SPIE 8189*. 2011, pp. 818916–818916–7. DOI: 10.1117/12.898344.

[48] Christoph Pacher et al. "Hacking QKD protocols that employ non-ITS authentication". In: *QCRYPT 2011: First Annual Conference on Quantum Cryptography*. Zurich, Switzerland, 2011.

[49] Alexander Treiber et al. “Fully automated entanglement-based quantum cryptography system for telecom fiber networks”. In: *New Journal of Physics* 11.4 (Jan. 2009), p. 20. ISSN: 1367-2630. DOI: 10.1088/1367-2630/11/4/045013.

[50] A Poppe et al. “Results from the SECOQC quantum-key-distribution network”. In: *CLEO/Europe - EQEC 2009 - European Conference on Lasers and Electro-Optics and the European Quantum Electronics Conference*. Vol. 209. 2008. Vienna, Austria: IEEE, June 2009, pp. 1–1. ISBN: 978-1-4244-4079-5. DOI: 10.1109/CLEOE-EQEC.2009.5192790.

[51] A. Treiber et al. “Reliable hands-off entanglement-based QKD system for fiber networks”. In: *CLEO/Europe - EQEC 2009 - European Conference on Lasers and Electro-Optics and the European Quantum Electronics Conference*. Vienna, Austria: IEEE, June 2009, pp. 1–1. ISBN: 978-1-4244-4079-5. DOI: 10.1109/CLEOE-EQEC.2009.5194792.

[52] M Peev et al. “The SECOQC Quantum-Key-Distribution network in Vienna”. In: *Optical Fiber Communication - incudes post deadline papers, 2009. OFC 2009. Conference on*. Mar. 2009, pp. 1–3.

[53] M Peev et al. “The SECOQC quantum key distribution network in Vienna”. In: *New Journal of Physics* 11.7 (July 2009), p. 075001. ISSN: 1367-2630. DOI: 10.1088/1367-2630/11/7/075001.

[54] O Maurhart et al. “Node Modules and Protocols for the Quantum-Back-Bone of a QKD Network”. In: *CLEO/Europe - EQEC 2009 - European Conference on Lasers and Electro-Optics and the European Quantum Electronics Conference*. Vienna, Austria: IEEE, 2009, pp. 1–2. ISBN: 9783800731732.

[55] M. Peev et al. “Response To "Vulnerability of ’a Novel Protocol-Authentication Algorithm Ruling Out a Man-In-The-Middle Attack In Quantum Cryptography"”. In: *International Journal of Quantum Information* 7.7 (2009), p. 1401. ISSN: 0219-7499. DOI: 10.1142/S0219749909005729.

[56] Andreas Neppach et al. “Key Management of Quantum Generated Keys in IPsec”. In: *SECRYPT 2008, Proceedings of the International Conference on Security and Cryptography, Porto, Portugal, July 26-29, 2008, SECRYPT is part of ICETE - The International Joint Conference on e-Business and Telecommunications*. Ed. by Eduardo Fernández-Medina, Manu Malek, and Javier Hernando. INSTICC Press, 2008, pp. 177–183.

[57] Johannes Wolkerstorfer, Alexander Szekely, and Thomas Lorünser. “IPsec Security Gateway for Gigabit Ethernet”. In: *Austrochip 2008*. Ed. by Christoph Lackner et al. 2008, pp. 19–24.

[58] Thomas Lorünser et al. “Security processor with quantum key distribution”. In: *19th IEEE International Conference on Application-Specific Systems, Architectures and Processors, ASAP 2008, July 2-4, 2008, Leuven, Belgium*. IEEE Computer Society, 2008, pp. 37–42. DOI: `10.1109/ASAP.2008.4580151`.

[59] Hannes Hübel et al. “High-fidelity transmission of polarization encoded qubits from an entangled source over 100 km of fiber”. In: *Optics Express* 15.12 (Jan. 2008), p. 7. ISSN: 1094-4087. DOI: `10.1364/OE.15.007853`. eprint: `0801.3620`.

[60] H. Hubel et al. “Robustness of polarization entanglement for long distance QKD”. In: *Lasers and Electro-Optics, 2007 and the Intern. Quantum Electronics Conference. CLEOE-IQEC 2007. European Conference on*. IEEE, June 2007, pp. 1–1. ISBN: 978-1-4244-0930-3. DOI: `10.1109/CLEOE-IQEC.2007.4387007`.

[61] A Poppe et al. “Quantum key distribution over WDMs and optical switches to combine the quantum channel with synchronization channels”. In: *33rd European Conference and Exhibition on Optical Communication ECOC 2007*. 1. Iee, 2007, pp. 947–947. ISBN: 9783800730421. DOI: `10.1049/ic:20070343`.

[62] Stephan Bettelli et al. “Effect of double pair emission to entanglement based QKD”. In: *European Conference on Lasers and Electro-Optics, 2007 and the International Quantum Electronics Conference. CLEOE-IQEC 2007*. IEEE, June 2007, pp. 1–1. ISBN: 978-1-4244-0930-3, 978-1-4244-0931-0. DOI: `10.1109/CLEOE-IQEC.2007.4387006`.

[63] A. Fedrizzi et al. “Practical quantum key distribution with polarization entangled photons”. In: *EQEC ’05. European Quantum Electronics Conference, 2005*. IEEE, 2005, pp. 303–303. ISBN: 0-7803-8973-5. DOI: `10.1109/EQEC.2005.1567469`.

[64] M. Peev et al. “A Novel Protocol-Authentication Algorithm Ruling Out a Man-in-the-Middle Attack in Quantum Cryptography”. In: *Intern. Journal of Quantum Inform. (IJQI)* 3.1 (July 2004), p. 4. DOI: `10.1142/S0219749905000797`. eprint: `0407131` (quant-ph).

[65] A Poppe et al. “Practical quantum key distribution with polarization entangled photons.” In: *Optics Express* 12.16 (2004), pp. 3865–3871.

[66] Roland Lieger et al. “Embedding quantum cryptography on DSP-boards”. In: *2004 12th European Signal Processing Conference, Vienna, Austria, September 6-10, 2004*. IEEE, 2004, pp. 2027–2030.

General References

[67] World Economic Forum. *Deep shift: technology tipping points and societal impact*. 2015. URL: https://www3.weforum.org/docs/WEF_GAC15_Technological_Tipping_Points_report_2015.pdf (visited on 01/10/2023).

[68] Thomas Ristenpart et al. “Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds”. In: *Proceedings of the 2009 ACM Conference on Computer and Communications Security, CCS 2009, Chicago, Illinois, USA, November 9-13, 2009*. ACM, 2009, pp. 199–212. DOI: 10.1145/1653662.1653687.

[69] Yinqian Zhang et al. “Cross-Tenant Side-Channel Attacks in PaaS Clouds”. In: *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*. Ed. by Gail-Joon Ahn, Moti Yung, and Ninghui Li. ACM, 2014, pp. 990–1003. DOI: 10.1145/2660267.2660356.

[70] Raj Samani, Brian Honan, and Jim Reavis. “Cloud Security Alliance Research”. In: *CSA Guide to Cloud Computing*. 2015. DOI: 10.1016/b978-0-12-420125-5.00008-x.

[71] ENISA. *Cloud Computing Risk Assessment*. 2009. URL: https://www.enisa.europa.eu/publications/cloud-computing-risk-assessment (visited on 01/10/2023).

[72] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. “Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation (Extended Abstract)”. In: *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*. Ed. by Janos Simon. ACM, 1988, pp. 1–10. DOI: 10.1145/62212.62213.

[73] David Chaum, Claude Crépeau, and Ivan Damgård. “Multiparty Unconditionally Secure Protocols (Extended Abstract)”. In: *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*. Ed. by Janos Simon. ACM, 1988, pp. 11–19. DOI: 10.1145/62212.62214.

[74] Claude E. Shannon. “Communication theory of secrecy systems”. In: *Bell Syst. Tech. J.* 28.4 (1949), pp. 656–715. DOI: 10.1002/j.1538-7305.1949.tb00928.x.

[75] ISO/IEC 4922-1. *Information security — Secure multiparty computation — Part 1: General*. https://www.iso.org/standard/80508.html. 2022.

[76] Cloud Security Alliance. *The Treacherous 12 Cloud Computing Top Threats in 2016*. 2016. URL: `https://cloudsecurityalliance.org/artifacts/the-treacherous-twelve-cloud-computing-top-threats-in-2016/` (visited on 12/12/2022).

[77] Mahesh Kallahalla et al. "Plutus: Scalable Secure File Sharing on Untrusted Storage". In: *Proceedings of the FAST '03 Conference on File and Storage Technologies, March 31 - April 2, 2003, Cathedral Hill Hotel, San Francisco, California, USA*. Ed. by Jeff Chase. USENIX, 2003.

[78] Shucheng Yu et al. "Achieving Secure, Scalable, and Fine-grained Data Access Control in Cloud Computing". In: *INFOCOM 2010. 29th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 15-19 March 2010, San Diego, CA, USA*. IEEE, 2010, pp. 534–542. DOI: `10.1109/INFCOM.2010.5462174`.

[79] Amit Sahai and Brent Waters. "Fuzzy Identity-Based Encryption". In: *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*. Ed. by Ronald Cramer. Vol. 3494. Lecture Notes in Computer Science. Springer, 2005, pp. 457–473. DOI: `10.1007/11426639\_27`.

[80] Vipul Goyal et al. "Attribute-based encryption for fine-grained access control of encrypted data". In: *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, Alexandria, VA, USA, October 30 - November 3, 2006*. Ed. by Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati. ACM, 2006, pp. 89–98. DOI: `10.1145/1180405.1180418`.

[81] Matt Blaze, Gerrit Bleumer, and Martin Strauss. "Divertible Protocols & Atomic Proxy Cryptography". In: *Advances in Cryptology - EUROCRYPT '98, International Conference on the Theory and Application of Cryptographic Techniques, Espoo, Finland, May 31 - June 4, 1998, Proceeding*. Ed. by Kaisa Nyberg. Vol. 1403. Lecture Notes in Computer Science. Springer, 1998, pp. 127–144. DOI: `10.1007/BFb0054122`.

[82] Giuseppe Ateniese et al. "Improved Proxy Re-Encryption Schemes with Applications to Secure Distributed Storage". In: *Proceedings of the Network and Distributed System Security Symposium, NDSS 2005, San Diego, California, USA*. The Internet Society, 2005.

[83] Matthew Pirretti et al. "Secure attribute-based systems". In: *J. Comput. Secur.* 18.5 (2010), pp. 799–837. DOI: `10.3233/JCS-2009-0383`.

[84] Seny Kamara and Kristin E. Lauter. “Cryptographic Cloud Storage”. In: *Financial Cryptography and Data Security, FC 2010 Workshops, RLCPS, WECSR, and WLC 2010, Tenerife, Canary Islands, Spain, January 25-28, 2010, Revised Selected Papers*. Ed. by Radu Sion et al. Vol. 6054. Lecture Notes in Computer Science. Springer, 2010, pp. 136–149. DOI: 10.1007/978-3-642-14992-4_13.

[85] John Bethencourt, Amit Sahai, and Brent Waters. “Ciphertext-Policy Attribute-Based Encryption”. In: *2007 IEEE Symposium on Security and Privacy (S&P 2007), 20-23 May 2007, Oakland, California, USA*. IEEE Computer Society, 2007, pp. 321–334. DOI: 10.1109/SP.2007.11.

[86] Amit Sahai, Hakan Seyalioglu, and Brent Waters. “Dynamic Credentials and Ciphertext Delegation for Attribute-Based Encryption”. In: *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*. Ed. by Reihaneh Safavi-Naini and Ran Canetti. Vol. 7417. Lecture Notes in Computer Science. Springer, 2012, pp. 199–217. DOI: 10.1007/978-3-642-32009-5_13.

[87] Johannes Braun et al. “Long term confidentiality: a survey”. In: *Des. Codes Cryptogr.* 71.3 (2014), pp. 459–478. DOI: 10.1007/s10623-012-9747-6.

[88] Daniel Slamanig and Christian Hanser. “On cloud storage and the cloud of clouds approach”. In: *7th International Conference for Internet Technology and Secured Transactions, ICITST 2012, London, United Kingdom, December 10-12, 2012*. Ed. by Nick Savage, Safwan El Assad, and Charles A. Shoniregun. IEEE, 2012, pp. 649–655.

[89] Mennan Selimi and Felix Freitag. “Tahoe-LAFS Distributed Storage Service in Community Network Clouds”. In: *2014 IEEE Fourth International Conference on Big Data and Cloud Computing, BDCloud 2014, Sydney, Australia, December 3-5, 2014*. IEEE Computer Society, 2014, pp. 17–24. DOI: 10.1109/BDCloud.2014.24.

[90] Josef Spillner et al. “Information Dispersion over Redundant Arrays of Optimal Cloud Storage for Desktop Users”. In: *IEEE 4th International Conference on Utility and Cloud Computing, UCC 2011, Melbourne, Australia, December 5-8, 2011*. IEEE Computer Society, 2011, pp. 1–8. DOI: 10.1109/UCC.2011.11.

[91] Hussam Abu-Libdeh, Lonnie Princehouse, and Hakim Weatherspoon. “RACS: a case for cloud storage diversity”. In: *Proceedings of the 1st ACM Symposium on Cloud Computing, SoCC 2010, Indianapolis, Indiana, USA, June 10-11, 2010*. Ed. by Joseph M. Hellerstein, Surajit Chaudhuri, and Mendel Rosenblum. ACM, 2010, pp. 229–240. DOI: 10.1145/1807128.1807165.

[92] Alysson Neves Bessani et al. "DepSky: Dependable and Secure Storage in a Cloud-of-Clouds". In: *ACM Trans. Storage* 9.4 (2013), p. 12. DOI: 10.1145/2535929.

[93] Hugo Krawczyk. "Secret Sharing Made Short". In: *Advances in Cryptology - CRYPTO '93, 13th Annual International Cryptology Conference, Santa Barbara, California, USA, August 22-26, 1993, Proceedings*. Ed. by Douglas R. Stinson. Vol. 773. Lecture Notes in Computer Science. Springer, 1993, pp. 136–146. DOI: 10.1007/3-540-48329-2_12.

[94] Raluca Ada Popa et al. "Enabling Security in Cloud Storage SLAs with CloudProof". In: *2011 USENIX Annual Technical Conference, Portland, OR, USA, June 15-17, 2011*. Ed. by Jason Nieh and Carl A. Waldspurger. USENIX Association, 2011.

[95] Miguel Castro and Barbara Liskov. "Practical byzantine fault tolerance and proactive recovery". In: *ACM Trans. Comput. Syst.* 20.4 (2002), pp. 398–461. DOI: 10.1145/571637.571640.

[96] Danyang Zhuo et al. "Machine fault tolerance for reliable datacenter systems". In: *Asia-Pacific Workshop on Systems, APSys'14, Beijing, China, June 25-26, 2014*. ACM, 2014, 3:1–3:7. DOI: 10.1145/2637166.2637235.

[97] Miguel Correia, Nuno Ferreira Neves, and Paulo Verıssimo. "BFT-TO: Intrusion Tolerance with Less Replicas". In: *Comput. J.* 56.6 (2013), pp. 693–715. DOI: 10.1093/comjnl/bxs148.

[98] Ramakrishna Kotla et al. "Zyzzyva: speculative byzantine fault tolerance". In: *Proceedings of the 21st ACM Symposium on Operating Systems Principles 2007, SOSP 2007, Stevenson, Washington, USA, October 14-17, 2007*. Ed. by Thomas C. Bressoud and M. Frans Kaashoek. ACM, 2007, pp. 45–58. DOI: 10.1145/1294261.1294267.

[99] Giuliana Santos Veronese et al. "Spin One's Wheels? Byzantine Fault Tolerance with a Spinning Primary". In: *28th IEEE Symposium on Reliable Distributed Systems (SRDS 2009), Niagara Falls, New York, USA, September 27-30, 2009*. IEEE Computer Society, 2009, pp. 135–144. DOI: 10.1109/SRDS.2009.36.

[100] Adi Shamir. "How to Share a Secret". In: *Commun. ACM* 22.11 (1979), pp. 612–613. DOI: 10.1145/359168.359176.

[101] Michael O. Rabin. "Efficient dispersal of information for security, load balancing, and fault tolerance". In: *J. ACM* 36.2 (1989), pp. 335–348. DOI: 10.1145/62044.62050.

[102] Daniel J Bernstein. *ChaCha, a variant of Salsa20*. 2008. URL: http://cr.yp.to/chacha/chacha-20080128.pdf (visited on 12/12/2022).

[103] Tal Rabin and Michael Ben-Or. "Verifiable Secret Sharing and Multiparty Protocols with Honest Majority (Extended Abstract)". In: *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washington, USA*. Ed. by David S. Johnson. ACM, 1989, pp. 73–85. DOI: `10.1145/73007.73014`.

[104] Daniel J. Bernstein. "The Poly1305-AES Message-Authentication Code". In: *Fast Software Encryption: 12th International Workshop, FSE 2005, Paris, France, February 21-23, 2005, Revised Selected Papers*. Ed. by Henri Gilbert and Helena Handschuh. Vol. 3557. Lecture Notes in Computer Science. Springer, 2005, pp. 32–49. DOI: `10.1007/11502760\_3`.

[105] Nikos Chondros, Konstantinos Kokordelis, and Mema Roussopoulos. "On the Practicality of Practical Byzantine Fault Tolerance". In: *Middleware 2012 - ACM/IFIP/USENIX 13th International Middleware Conference, Montreal, QC, Canada, December 3-7, 2012. Proceedings*. Ed. by Priya Narasimhan and Peter Triantafillou. Vol. 7662. Lecture Notes in Computer Science. Springer, 2012, pp. 436–455. DOI: `10.1007/978-3-642-35170-9\_22`.

[106] Jae Kwon. *TenderMint : Consensus without Mining (Software)*. 2014. URL: `https://tendermint.com/` (visited on 07/12/2021).

[107] Maofan Yin et al. "HotStuff: BFT Consensus with Linearity and Responsiveness". In: (2019). Ed. by Peter Robinson and Faith Ellen, pp. 347–356. DOI: `10.1145/3293611.3331591`.

[108] Andrew Miller et al. "The Honey Badger of BFT Protocols". In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*. Ed. by Edgar R. Weippl et al. ACM, 2016, pp. 31–42. DOI: `10.1145/2976749.2978399`.

[109] Divya Gupta, Lucas Perronne, and Sara Bouchenak. "BFT-Bench: Towards a Practical Evaluation of Robustness and Effectiveness of BFT Protocols". In: *Distributed Applications and Interoperable Systems - 16th IFIP WG 6.1 International Conference, DAIS 2016, Held as Part of the 11th International Federated Conference on Distributed Computing Techniques, DisCoTec 2016, Heraklion, Crete, Greece, June 6-9, 2016, Proceedings*. Ed. by Márk Jelasity and Evangelia Kalyvianaki. Vol. 9687. Lecture Notes in Computer Science. Springer, 2016, pp. 115–128. DOI: `10.1007/978-3-319-39577-7\_10`.

[110] Harish Sukhwani et al. "Performance Modeling of PBFT Consensus Process for Permissioned Blockchain Network (Hyperledger Fabric)". In: *36th IEEE Symposium on Reliable Distributed Systems, SRDS 2017, Hong Kong, Hong Kong, September 26-29, 2017*. IEEE Computer Society, 2017, pp. 253–255. DOI: `10.1109/SRDS.2017.36`.

[111] Andras Varga. *TCP Tahoe/Reno/NewReno strange behaviors*. 2015. URL: https://github.com/inet-framework/inet/issues/75 (visited on 07/06/2020).

[112] Stephen Hemminger. *Network Emulation with NetEm*. 2005. URL: https://wiki.linuxfoundation.org/networking/netem (visited on 04/06/2018).

[113] Adorjan Matt. *AWS Latency Monitoring*. 2020. URL: https://www.cloudping.co/grid (visited on 12/10/2020).

[114] Jing Han et al. "Survey on NoSQL database". In: *Pervasive computing and applications (ICPCA), 2011 6th international conference on*. IEEE. 2011, pp. 363–366.

[115] Bernhard Zatoukal et al. "OpenQKD Use-case for Securing Sensitive Medical Data at rest and in transit". In: *2021 Conference on Lasers and Electro-Optics Europe & European Quantum Electronics Conference (CLEO/Europe-EQEC)*. 2021, p. 1. DOI: 10.1109/CLEO/Europe-EQEC52157.2021.9542590.

[116] Marc Shapiro et al. "Conflict-Free Replicated Data Types". In: *Stabilization, Safety, and Security of Distributed Systems - 13th International Symposium, SSS 2011, Grenoble, France, October 10-12, 2011. Proceedings*. Ed. by Xavier Défago, Franck Petit, and Vincent Villain. Vol. 6976. Lecture Notes in Computer Science. Springer, 2011, pp. 386–400. DOI: 10.1007/978-3-642-24550-3_29.

[117] Manuel Barbosa et al. "Secure Conflict-free Replicated Data Types". In: *ICDCN '21: International Conference on Distributed Computing and Networking, Virtual Event, Nara, Japan, January 5-8, 2021*. ACM, 2021, pp. 6–15. DOI: 10.1145/3427796.3427831.

[118] Mohamed Elboukhari, Mostafa Azizi, and Abdelmalek Azizi. "Improving TLS Security By Quantum Cryptography". In: *International Journal of Network Security & Its Applications* 2.3 (2010), pp. 87–100. ISSN: 09752307. DOI: 10.5121/ijnsa.2010.2306.

[119] Peng Yan and Nengkun Yu. "The QQUIC Transport Protocol: Quantum-Assisted UDP Internet Connections". In: *Entropy* 24.10 (2022), p. 1488. DOI: 10.3390/e24101488.

[120] Douglas Stebila, Scott Fluhrer, and Shay Gueron. *Hybrid key exchange in TLS 1.3*. 2022. URL: https://www.ietf.org/archive/id/draft-ietf-tls-hybrid-design-04.txt (visited on 01/10/2023).

[121] Aleksandar Hudic et al. "Towards a Unified Secure Cloud Service Development and Deployment Life-Cycle". In: *11th International Conference on Availability, Reliability and Security, ARES 2016, Salzburg, Austria, August 31 - September 2, 2016*. IEEE Computer Society, 2016, pp. 428–436. DOI: 10.1109/ARES.2016.73.

[122] Johannes Wolkerstorfer. “Secret-Sharing Hardware Improves the Privacy of Network Monitoring”. In: *Data Privacy Management and Autonomous Spontaneous Security - 5th International Workshop, DPM 2010 and 3rd International Workshop, SETOP 2010, Athens, Greece, September 23, 2010, Revised Selected Papers*. Ed. by Joaquín García-Alfaro et al. Vol. 6514. Lecture Notes in Computer Science. Springer, 2010, pp. 51–63. DOI: 10.1007/978-3-642-19348-4_5.

[123] Pei Luo et al. “Hardware Implementation of Secure Shamir’s Secret Sharing Scheme”. In: *15th International IEEE Symposium on High-Assurance Systems Engineering, HASE 2014, Miami Beach, FL, USA, January 9-11, 2014*. IEEE Computer Society, 2014, pp. 193–200. DOI: 10.1109/HASE.2014.34.

[124] Martin Kirchner. “On The Applicability Of Secret Sharing Cryptography In Secure Cloud Services”. MA thesis. Technische Universität Wien, 2014.

[125] A. Abdallah and M. Salleh. “Secret sharing scheme security and performance analysis”. In: *2015 International Conference on Computing, Control, Networking, Electronics and Embedded Systems Engineering (ICCNEEE)*. 2015, pp. 173–180. DOI: 10.1109/ICCNEEE.2015.7381357.

[126] Su Chen et al. “Deploying Scalable and Secure Secret Sharing with GPU Many-Core Architecture”. In: *26th IEEE International Parallel and Distributed Processing Symposium Workshops & PhD Forum, IPDPS 2012, Shanghai, China, May 21-25, 2012*. IEEE Computer Society, 2012, pp. 1360–1369. DOI: 10.1109/IPDPSW.2012.173.

[127] Rogelio Adrian Hernandez-Becerril et al. “A GPU implementation of secret sharing scheme based on cellular automata”. In: *J. Supercomput.* 72.4 (2016), pp. 1291–1311. DOI: 10.1007/s11227-016-1646-6.

[128] Knut Wold and Chik How Tan. “Analysis and Enhancement of Random Number Generator in FPGA Based on Oscillator Rings”. In: *ReConFig’08: 2008 International Conference on Reconfigurable Computing and FPGAs, 3-5 December 2008, Cancun, Mexico, Proceedings*. IEEE Computer Society, 2008, pp. 385–390. DOI: 10.1109/ReConFig.2008.17.

[129] A. Karatsuba and Yu. Ofman. “Multiplication of multidigit numbers on automata”. In: *Soviet Physics - Doklady*. Vol. 7. Jan. 1963, pp. 595–596.

[130] Giuseppe Ateniese et al. “Provable data possession at untrusted stores”. In: *Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, October 28-31, 2007*. Ed. by Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson. ACM, 2007, pp. 598–609. DOI: 10.1145/1315245.1315318.

[131] Ari Juels and Burton S. Kaliski Jr. “Pors: proofs of retrievability for large files”. In: *Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, October 28-31, 2007*. Ed. by Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson. ACM, 2007, pp. 584–597. DOI: `10.1145/1315245.1315317`.

[132] Mehul A. Shah et al. “Auditing to Keep Online Storage Services Honest”. In: *Proceedings of HotOS’07: 11th Workshop on Hot Topics in Operating Systems, May 7-9, 2005, San Diego, California, USA*. Ed. by Galen C. Hunt. USENIX Association, 2007.

[133] Hovav Shacham and Brent Waters. “Compact Proofs of Retrievability”. In: *Advances in Cryptology - ASIACRYPT 2008, 14th International Conference on the Theory and Application of Cryptology and Information Security, Melbourne, Australia, December 7-11, 2008. Proceedings*. Ed. by Josef Pieprzyk. Vol. 5350. Lecture Notes in Computer Science. Springer, 2008, pp. 90–107. DOI: `10.1007/978-3-540-89255-7\_7`.

[134] David Cash, Alptekin Küpçü, and Daniel Wichs. “Dynamic Proofs of Retrievability via Oblivious RAM”. In: *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*. Ed. by Thomas Johansson and Phong Q. Nguyen. Vol. 7881. Lecture Notes in Computer Science. Springer, 2013, pp. 279–295. DOI: `10.1007/978-3-642-38348-9\_17`.

[135] Elaine Shi, Emil Stefanov, and Charalampos Papamanthou. “Practical dynamic proofs of retrievability”. In: *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS’13, Berlin, Germany, November 4-8, 2013*. Ed. by Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung. ACM, 2013, pp. 325–336. DOI: `10.1145/2508859.2516669`.

[136] Chaowen Guan et al. “Symmetric-Key Based Proofs of Retrievability Supporting Public Verification”. In: *Computer Security - ESORICS 2015 - 20th European Symposium on Research in Computer Security, Vienna, Austria, September 21-25, 2015, Proceedings, Part I*. Ed. by Günther Pernul, Peter Y. A. Ryan, and Edgar R. Weippl. Vol. 9326. Lecture Notes in Computer Science. Springer, 2015, pp. 203–223. DOI: `10.1007/978-3-319-24174-6\_11`.

[137] Nesrine Kaaniche and Maryline Laurent. “SHoPS: Set Homomorphic Proof of Data Possession Scheme in Cloud Storage Applications”. In: *2015 IEEE World Congress on Services, SERVICES 2015, New York City, NY, USA, June 27 - July 2, 2015*. Ed. by Liang-Jie Zhang and Rami Bahsoon. IEEE Computer Society, 2015, pp. 143–150. DOI: `10.1109/SERVICES.2015.29`.

[138] Thomas J. E. Schwarz and Ethan L. Miller. "Store, Forget, and Check: Using Algebraic Signatures to Check Remotely Administered Storage". In: *26th IEEE International Conference on Distributed Computing Systems (ICDCS 2006), 4-7 July 2006, Lisboa, Portugal*. IEEE Computer Society, 2006, p. 12. DOI: `10.1109/ICDCS.2006.80`.

[139] Reza Curtmola et al. "MR-PDP: Multiple-Replica Provable Data Possession". In: *28th IEEE Int. Conference on Distributed Computing Systems (ICDCS 2008), 17-20 June 2008, Beijing, China*. IEEE Computer Society, 2008, pp. 411–420. DOI: `10.1109/ICDCS.2008.68`.

[140] Cong Wang et al. "Ensuring data storage security in Cloud Computing". In: *17th International Workshop on Quality of Service, IWQoS 2009, Charleston, South Carolina, USA, 13-15 July 2009*. IEEE, 2009, pp. 1–9. DOI: `10.1109/IWQoS.2009.5201385`.

[141] Kevin D. Bowers, Ari Juels, and Alina Oprea. "HAIL: a high-availability and integrity layer for cloud storage". In: *Proceedings of the 2009 ACM Conference on Computer and Communications Security, CCS 2009, Chicago, Illinois, USA, November 9-13, 2009*. Ed. by Ehab Al-Shaer, Somesh Jha, and Angelos D. Keromytis. ACM, 2009, pp. 187–198. DOI: `10.1145/1653662.1653686`.

[142] Clémentine Gritti, Willy Susilo, and Thomas Plantard. "Efficient Dynamic Provable Data Possession with Public Verifiability and Data Privacy". In: *Information Security and Privacy - 20th Australasian Conference, ACISP 2015, Brisbane, QLD, Australia, June 29 - July 1, 2015, Proceedings*. Ed. by Ernest Foo and Douglas Stebila. Vol. 9144. Lecture Notes in Computer Science. Springer, 2015, pp. 395–412. DOI: `10.1007/978-3-319-19962-7\_23`.

[143] Michael Ben-Or et al. "Multi-Prover Interactive Proofs: How to Remove Intractability Assumptions". In: *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*. Ed. by Janos Simon. ACM, 1988, pp. 113–131. DOI: `10.1145/62212.62223`.

[144] Yair Frankel, Philip D. MacKenzie, and Moti Yung. "Robust Efficient Distributed RSA-Key Generation". In: *Proceedings of the Seventeenth Annual ACM Symposium on Principles of Distributed Computing, PODC '98, Puerto Vallarta, Mexico, June 28 - July 2, 1998*. Ed. by Brian A. Coan and Yehuda Afek. ACM, 1998, p. 320. DOI: `10.1145/277697.277779`.

[145] Ivan Damgård. *On Σ-Protocols*. Lecture on Cryptologic Protocol Theory; Faculty of Science, University of Aarhus. 2010.

[146] Fabrice Benhamouda et al. “Efficient Zero-Knowledge Proofs for Commitments from Learning with Errors over Rings”. In: *Computer Security - ESORICS 2015 - 20th European Symposium on Research in Computer Security, Vienna, Austria, September 21-25, 2015, Proceedings, Part I*. Ed. by Günther Pernul, Peter Y. A. Ryan, and Edgar R. Weippl. Vol. 9326. Lecture Notes in Computer Science. Springer, 2015, pp. 305–325. DOI: `10.1007/978-3-319-24174-6\_16`.

[147] Lloyd Welch and Elwyn Berlekamp. *Error Correction of Algebraic Block Codes*. US Patent #4,633,470. 1983.

[148] Mehrdad Nojoumian, Douglas R. Stinson, and Morgan Grainger. “Unconditionally secure social secret sharing scheme”. In: *IET Inf. Secur.* 4.4 (2010), pp. 202–211. DOI: `10.1049/iet-ifs.2009.0098`.

[149] Benny Chor et al. “Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults (Extended Abstract)”. In: *26th Annual Symposium on Foundations of Computer Science, Portland, Oregon, USA, 21-23 October 1985*. IEEE Computer Society, 1985, pp. 383–395. DOI: `10.1109/SFCS.1985.64`.

[150] Matthias Fitzi et al. “Round-Optimal and Efficient Verifiable Secret Sharing”. In: *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings*. Ed. by Shai Halevi and Tal Rabin. Vol. 3876. Lecture Notes in Computer Science. Springer, 2006, pp. 329–342. DOI: `10.1007/11681878\_17`.

[151] Rosario Gennaro et al. “The round complexity of verifiable secret sharing and secure multicast”. In: *Proceedings on 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece*. Ed. by Jeffrey Scott Vitter, Paul G. Spirakis, and Mihalis Yannakakis. ACM, 2001, pp. 580–589. DOI: `10.1145/380752.380853`.

[152] Jonathan Katz, Chiu-Yuen Koo, and Ranjit Kumaresan. “Improving the Round Complexity of VSS in Point-to-Point Networks”. In: *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part II - Track B: Logic, Semantics, and Theory of Programming & Track C: Security and Cryptography Foundations*. Ed. by Luca Aceto et al. Vol. 5126. Lecture Notes in Computer Science. Springer, 2008, pp. 499–510. DOI: `10.1007/978-3-540-70583-3\_41`.

[153] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. “Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation (Extended Abstract)”. In: *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*. Ed. by Janos Simon. ACM, 1988, pp. 1–10. DOI: `10.1145/62212.62213`.

[154] Danny Dolev et al. “Perfectly Secure Message Transmission”. In: *J. ACM* 40.1 (1993), pp. 17–47. DOI: 10.1145/138027.138036.

[155] Torben P. Pedersen. “Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing”. In: *Advances in Cryptology - CRYPTO ’91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*. Ed. by Joan Feigenbaum. Vol. 576. Lecture Notes in Computer Science. Springer, 1991, pp. 129–140. DOI: 10.1007/3-540-46766-1_9.

[156] Paul Feldman. “A Practical Scheme for Non-interactive Verifiable Secret Sharing”. In: *28th Annual Symposium on Foundations of Computer Science, Los Angeles, California, USA, 27-29 October 1987*. IEEE Computer Society, 1987, pp. 427–437. DOI: 10.1109/SFCS.1987.4.

[157] Michael Backes, Aniket Kate, and Arpita Patra. “Computational Verifiable Secret Sharing Revisited”. In: *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings*. Ed. by Dong Hoon Lee and Xiaoyun Wang. Vol. 7073. Lecture Notes in Computer Science. Springer, 2011, pp. 590–609. DOI: 10.1007/978-3-642-25385-0_32.

[158] Eiichiro Fujisaki and Tatsuaki Okamoto. “Practical and Provably Secure Scheme for Publicly Verifiable Secret Sharing and Its Applications”. In: *Advances in Cryptology - EUROCRYPT ’98, International Conference on the Theory and Application of Cryptographic Techniques, Espoo, Finland, May 31 - June 4, 1998, Proceeding*. Ed. by Kaisa Nyberg. Vol. 1403. Lecture Notes in Computer Science. Springer, 1998, pp. 32–46. DOI: 10.1007/BFb0054115.

[159] Mahabir Prasad Jhanwar, Ayineedi Venkateswarlu, and Reihaneh Safavi-Naini. “Paillier-based publicly verifiable (non-interactive) secret sharing”. In: *Designs, Codes and Cryptography* 73.2 (2014), pp. 529–546. DOI: 10.1007/s10623-014-9952-6.

[160] Berry Schoenmakers. “A Simple Publicly Verifiable Secret Sharing Scheme and Its Application to Electronic”. In: *Advances in Cryptology - CRYPTO ’99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*. Ed. by Michael J. Wiener. Vol. 1666. Lecture Notes in Computer Science. Springer, 1999, pp. 148–164. DOI: 10.1007/3-540-48405-1_10.

[161] Christian Cachin et al. “Asynchronous verifiable secret sharing and proactive cryptosystems”. In: *Proceedings of the 9th ACM Conference on Computer and Communications Security, CCS 2002, Washington, DC, USA, November 18-22,*

2002. Ed. by Vijayalakshmi Atluri. ACM, 2002, pp. 88–97. DOI: 10.1145/586110.586124.

[162] Ran Canetti and Tal Rabin. “Fast asynchronous Byzantine agreement with optimal resilience”. In: *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, May 16-18, 1993, San Diego, CA, USA*. Ed. by S. Rao Kosaraju, David S. Johnson, and Alok Aggarwal. ACM, 1993, pp. 42–51. DOI: 10.1145/167088.167105.

[163] Arpita Patra, Ashish Choudhary, and C. Pandu Rangan. “Simple and efficient asynchronous byzantine agreement with optimal resilience”. In: *Proceedings of the 28th Annual ACM Symposium on Principles of Distributed Computing, PODC 2009, Calgary, Alberta, Canada, August 10-12, 2009*. Ed. by Srikanta Tirthapura and Lorenzo Alvisi. ACM, 2009, pp. 92–101. DOI: 10.1145/1582716.1582736.

[164] Mihir Bellare, Juan A. Garay, and Tal Rabin. “Distributed Pseudo-Random Bit Generators - A New Way to Speed-Up Shared Coin Tossing”. In: *Proceedings of the Fifteenth Annual ACM Symposium on Principles of Distributed Computing, Philadelphia, Pennsylvania, USA, May 23-26, 1996*. Ed. by James E. Burns and Yoram Moses. ACM, 1996, pp. 191–200. DOI: 10.1145/248052.248090.

[165] Kamer Kaya and Ali Aydin Selçuk. “A Verifiable Secret Sharing Scheme Based on the Chinese Remainder Theorem”. In: *Progress in Cryptology - INDOCRYPT 2008, 9th International Conference on Cryptology in India, Kharagpur, India, December 14-17, 2008. Proceedings*. Ed. by Dipanwita Roy Chowdhury, Vincent Rijmen, and Abhijit Das. Vol. 5365. Lecture Notes in Computer Science. Springer, 2008, pp. 414–425. DOI: 10.1007/978-3-540-89754-5_32.

[166] Mihir Bellare, Juan A. Garay, and Tal Rabin. “Fast Batch Verification for Modular Exponentiation and Digital Signatures”. In: *Advances in Cryptology - EUROCRYPT ’98, International Conference on the Theory and Application of Cryptographic Techniques, Espoo, Finland, May 31 - June 4, 1998, Proceeding*. Ed. by Kaisa Nyberg. Vol. 1403. Lecture Notes in Computer Science. Springer, 1998, pp. 236–250. DOI: 10.1007/BFb0054130.

[167] Markus Stadler. “Publicly Verifiable Secret Sharing”. In: *Advances in Cryptology - EUROCRYPT ’96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding*. Ed. by Ueli M. Maurer. Vol. 1070. Lecture Notes in Computer Science. Springer, 1996, pp. 190–199. DOI: 10.1007/3-540-68339-9_17.

[168] Steven Malin. “Auswirkungen der Integration eines Secret Sharing Object Stores in eine IaaS Cloud am Beispiel von Archistar und OpenStack”. MA thesis. University of Applied Sciences Burgenland, 2018.

[169] Martin Weltler. “Datenschutz: Anwendung von Multy Party Computation auf Fitnesstracker”. Bachelor’s Thesis. University of Applied Sciences Burgenland, 2022.

[170] Marcella Hastings et al. “SoK: General Purpose Compilers for Secure Multi-Party Computation”. In: *2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019*. IEEE, 2019, pp. 1220–1237. DOI: 10.1109/SP.2019.00028.

[171] Oded Goldreich, Silvio Micali, and Avi Wigderson. “How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority”. In: *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*. Ed. by Alfred V. Aho. ACM, 1987, pp. 218–229. DOI: 10.1145/28395.28420.

[172] Donald Beaver. “Efficient Multiparty Protocols Using Circuit Randomization”. In: *Advances in Cryptology - CRYPTO ’91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*. Ed. by Joan Feigenbaum. Vol. 576. Lecture Notes in Computer Science. Springer, 1991, pp. 420–432. DOI: 10.1007/3-540-46766-1_34.

[173] Martin R. Albrecht et al. “Ciphers for MPC and FHE”. In: *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*. Ed. by Elisabeth Oswald and Marc Fischlin. Vol. 9056. Lecture Notes in Computer Science. Springer, 2015, pp. 430–454. DOI: 10.1007/978-3-662-46800-5_17.

[174] Martin Albrecht et al. *Ciphers for MPC and FHE*. Cryptology ePrint Archive, Paper 2016/687. https://eprint.iacr.org/2016/687. 2016.

[175] Lorenzo Grassi et al. “MPC-Friendly Symmetric Key Primitives”. In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*. Ed. by Edgar R. Weippl et al. ACM, 2016, pp. 430–443. DOI: 10.1145/2976749.2978332.

[176] Christian Rechberger, Hadi Soleimany, and Tyge Tiessen. “Cryptanalysis of Low-Data Instances of Full LowMCv2”. In: *IACR Trans. Symmetric Cryptol.* 2018.3 (2018), pp. 163–181. DOI: 10.13154/tosc.v2018.i3.163-181.

[177] Lorenzo Grassi et al. “On a Generalization of Substitution-Permutation Networks: The HADES Design Strategy”. In: *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part II*. Ed. by Anne Canteaut and Yuval Ishai. Vol. 12106. Lecture

Notes in Computer Science. Springer, 2020, pp. 674–704. DOI: 10.1007/978-3-030-45724-2_23.

[178] Martin R. Albrecht et al. "Feistel Structures for MPC, and More". In: *Computer Security - ESORICS 2019 - 24th European Symposium on Research in Computer Security, Luxembourg, September 23-27, 2019, Proceedings, Part II*. Ed. by Kazue Sako, Steve A. Schneider, and Peter Y. A. Ryan. Vol. 11736. Lecture Notes in Computer Science. Springer, 2019, pp. 151–171. DOI: 10.1007/978-3-030-29962-0_8.

[179] Berry Schoenmakers. "MPyC–Python Package for Secure Multiparty Computation". In: *Theory and Practice of Multi-Party Computation 2018 - TPMPC 2018*. Aarhus, 2018.

[180] Ivan Damgård et al. "Asynchronous Multiparty Computation: Theory and Implementation". In: *Public Key Cryptography - PKC 2009, 12th International Conference on Practice and Theory in Public Key Cryptography, Irvine, CA, USA, March 18-20, 2009. Proceedings*. Ed. by Stanislaw Jarecki and Gene Tsudik. Vol. 5443. Lecture Notes in Computer Science. Springer, 2009, pp. 160–179. DOI: 10.1007/978-3-642-00468-1_10.

[181] Marcel Keller. *Multi-Protocol SPDZ*. URL: https://github.com/data61/MP-SPDZ (visited on 12/10/2022).

[182] Ivan Damgård et al. "Multiparty Computation from Somewhat Homomorphic Encryption". In: *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*. Ed. by Reihaneh Safavi-Naini and Ran Canetti. Vol. 7417. Lecture Notes in Computer Science. Springer, 2012, pp. 643–662. DOI: 10.1007/978-3-642-32009-5_38.

[183] Jesper Buus Nielsen et al. "A New Approach to Practical Active-Secure Two-Party Computation". In: *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*. Ed. by Reihaneh Safavi-Naini and Ran Canetti. Vol. 7417. Lecture Notes in Computer Science. Springer, 2012, pp. 681–700. DOI: 10.1007/978-3-642-32009-5_40.

[184] Marcel Keller, Peter Scholl, and Nigel P. Smart. "An architecture for practical actively secure MPC with dishonest majority". In: (2013). Ed. by Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, pp. 549–560. DOI: 10.1145/2508859.2516744.

[185] Benny Pinkas et al. “Secure Two-Party Computation Is Practical”. In: *Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings*. Ed. by Mitsuru Matsui. Vol. 5912. Lecture Notes in Computer Science. Springer, 2009, pp. 250–267. DOI: `10.1007/978-3-642-10366-7\_15`.

[186] Toshinori Araki et al. “High-Throughput Semi-Honest Secure Three-Party Computation with an Honest Majority”. In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*. Ed. by Edgar R. Weippl et al. ACM, 2016, pp. 805–817. DOI: `10.1145/2976749.2978331`.

[187] Ivan Damgård and Marcel Keller. “Secure Multiparty AES”. In: *Financial Cryptography and Data Security, 14th International Conference, FC 2010, Tenerife, Canary Islands, Spain, January 25-28, 2010, Revised Selected Papers*. Ed. by Radu Sion. Vol. 6052. Lecture Notes in Computer Science. Springer, 2010, pp. 367–374. DOI: `10.1007/978-3-642-14577-3\_31`.

[188] Ivan Damgård et al. “Implementing AES via an Actively/Covertly Secure Dishonest Majority MPC Protocol”. In: *Security and Cryptography for Networks - 8th International Conference, SCN 2012, Amalfi, Italy, September 5-7, 2012. Proceedings*. Ed. by Ivan Visconti and Roberto De Prisco. Vol. 7485. Lecture Notes in Computer Science. Springer, 2012, pp. 241–263. DOI: `10.1007/978-3-642-32928-9\_14`.

[189] Christophe De Cannière and Bart Preneel. “Trivium”. In: *New Stream Cipher Designs - The eSTREAM Finalists*. Ed. by Matthew J. B. Robshaw and Olivier Billet. Vol. 4986. Lecture Notes in Computer Science. Springer, 2008, pp. 244–266. DOI: `10.1007/978-3-540-68351-3\_18`.

[190] Y. Nir and A. Langley. *ChaCha20 and Poly1305 for IETF Protocols*. 2015. URL: `https://tools.ietf.org/html/rfc7539` (visited on 12/12/2022).

[191] ISO. *Information technology – Security techniques – Lightweight cryptography – Part3: Stream ciphers (2012), ISO/IEC 29192:2012*. 2012.

[192] Anne Canteaut et al. “Stream Ciphers: A Practical Solution for Efficient Homomorphic Ciphertext Compression”. In: *J. Cryptol.* 31.3 (2018), pp. 885–916. DOI: `10.1007/s00145-017-9273-9`.

[193] Martin R. Albrecht et al. “MiMC: Efficient Encryption and Cryptographic Hashing with Minimal Multiplicative Complexity”. In: *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I*. Ed. by Jung Hee Cheon and Tsuyoshi Takagi.

Vol. 10031. Lecture Notes in Computer Science. 2016, pp. 191–219. DOI: 10.1007/978-3-662-53887-6_7.

[194] Peter Bogetoft et al. “Secure Multiparty Computation Goes Live”. In: *Financial Cryptography and Data Security, 13th International Conference, FC 2009, Accra Beach, Barbados, February 23-26, 2009. Revised Selected Papers*. Ed. by Roger Dingledine and Philippe Golle. Vol. 5628. Lecture Notes in Computer Science. Springer, 2009, pp. 325–343. DOI: 10.1007/978-3-642-03549-4_20.

[195] David Cerezo Sánchez. *Raziel: Private and Verifiable Smart Contracts on Blockchains*. Cryptology ePrint Archive, Paper 2017/878. https://eprint.iacr.org/2017/878. 2017.

[196] Hisham S. Galal and Amr M. Youssef. “Verifiable Sealed-Bid Auction on the Ethereum Blockchain”. In: *Financial Cryptography and Data Security - FC 2018 International Workshops, BITCOIN, VOTING, and WTSC, Nieuwpoort, Curaçao, March 2, 2018, Revised Selected Papers*. Ed. by Aviv Zohar et al. Vol. 10958. Lecture Notes in Computer Science. Springer, 2018, pp. 265–278. DOI: 10.1007/978-3-662-58820-8_18.

[197] Erik-Oliver Blass and Florian Kerschbaum. “Strain: A Secure Auction for Blockchains”. In: *Computer Security - 23rd European Symposium on Research in Computer Security, ESORICS 2018, Barcelona, Spain, September 3-7, 2018, Proceedings, Part I*. Ed. by Javier López, Jianying Zhou, and Miguel Soriano. Vol. 11098. Lecture Notes in Computer Science. Springer, 2018, pp. 87–110. DOI: 10.1007/978-3-319-99073-6_5.

[198] Ahmed E. Kosba and et al. “Hawk: The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts”. In: *Security and Privacy, SP 2016*. IEEE Computer Society, 2016, pp. 839–858. DOI: 10.1109/SP.2016.55.

[199] Hisham S. Galal and Amr M. Youssef. “Trustee: Full Privacy Preserving Vickrey Auction on Top of Ethereum”. In: Lecture Notes in Computer Science 11599 (2019). Ed. by Andrea Bracciali et al., pp. 190–207. DOI: 10.1007/978-3-030-43725-1_14.

[200] Marcin Andrychowicz et al. “Secure Multiparty Computations on Bitcoin”. In: *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*. IEEE Computer Society, 2014, pp. 443–458. DOI: 10.1109/SP.2014.35.

[201] Iddo Bentov and Ranjit Kumaresan. “How to Use Bitcoin to Design Fair Protocols”. In: *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part II*. Ed. by Juan A. Garay and Rosario Gennaro. Vol. 8617. Lecture Notes in

Computer Science. Springer, 2014, pp. 421–439. DOI: 10.1007/978-3-662-44381-1_24.

[202] Ranjit Kumaresan, Vinod Vaikuntanathan, and Prashant Nalini Vasudevan. "Improvements to Secure Computation with Penalties". In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*. Ed. by Edgar R. Weippl et al. ACM, 2016, pp. 406–417. DOI: 10.1145/2976749.2978421.

[203] Ranjit Kumaresan and Iddo Bentov. "Amortizing Secure Computation with Penalties". In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*. Ed. by Edgar R. Weippl et al. ACM, 2016, pp. 418–429. DOI: 10.1145/2976749.2978424.

[204] Carsten Baum, Ivan Damgård, and Claudio Orlandi. "Publicly Auditable Secure Multi-Party Computation". In: *Security and Cryptography for Networks - 9th International Conference, SCN 2014, Amalfi, Italy, September 3-5, 2014. Proceedings*. Ed. by Michel Abdalla and Roberto De Prisco. Vol. 8642. Lecture Notes in Computer Science. Springer, 2014, pp. 175–196. DOI: 10.1007/978-3-319-10879-7_11.

[205] Meilof Veeningen. "Pinocchio-Based Adaptive zk-SNARKs and Secure/Correct Adaptive Function Evaluation". In: *Progress in Cryptology - AFRICACRYPT 2017 - 9th International Conference on Cryptology in Africa, Dakar, Senegal, May 24-26, 2017, Proceedings*. Ed. by Marc Joye and Abderrahmane Nitaj. Vol. 10239. Lecture Notes in Computer Science. 2017, pp. 21–39. DOI: 10.1007/978-3-319-57339-7_2.

[206] Michael Backes et al. "ADSNARK: Nearly Practical and Privacy-Preserving Proofs on Authenticated Data". In: *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*. IEEE Computer Society, 2015, pp. 271–286. DOI: 10.1109/SP.2015.24.

[207] Ahmed E. Kosba et al. "MIRAGE: Succinct Arguments for Randomized Algorithms with Applications to Universal zk-SNARKs". In: *29th USENIX Security Symposium, USENIX Security 2020, August 12-14, 2020*. Ed. by Srdjan Capkun and Franziska Roesner. USENIX Association, 2020, pp. 2129–2146.

[208] Larry Ausubel and Paul Milgrom. "Ascending Auctions with Package Bidding". In: *Frontiers of Theoretical Economics* 1.1 (2002), Article 1.

[209] Aditya V. Sunderam and David C. Parkes. "Preference elicitation in proxied multiattribute auctions". In: *Proceedings 4th ACM Conference on Electronic Commerce (EC-2003), San Diego, California, USA, June 9-12, 2003*. Ed. by Daniel A. Menascé and Noam Nisan. ACM, 2003, pp. 214–215. DOI: 10.1145/779928.779965.

[210] Benny Moldovanu. “Auction Theory and Applications”. In: *The Bonn Journal of Economics* 1.1 (2012), pp. 53–64.

[211] DFLEX Project. *Demonstration Report (D2)*. SESAR Joint Untertaking project deliverable. 2014.

[212] Eurocontrol. *Implementing user-driven prioritisation process at Zurich airport*. 2019. URL: `https://www.eurocontrol.int/use-cases/implementing-user-driven-prioritisation-process-zurich-airport` (visited on 07/20/2022).

[213] Jiangtao Li and Mikhail J. Atallah. “Secure and Private Collaborative Linear Programming”. In: *2nd International ICST Conference on Collaborative Computing: Networking, Applications and Worksharing, CollaborateCom 2006, Atlanta, GA, USA, November 17-20, 2006*. Ed. by Enrico Blanzieri and Tao Zhang. IEEE, 2006. DOI: `10.1109/COLCOM.2006.361848`.

[214] Tomas Toft. “Solving Linear Programs Using Multiparty Computation”. In: *Financial Cryptography and Data Security, 13th International Conference, FC 2009, Accra Beach, Barbados, February 23-26, 2009. Revised Selected Papers*. Ed. by Roger Dingledine and Philippe Golle. Vol. 5628. Lecture Notes in Computer Science. Springer, 2009, pp. 90–107. DOI: `10.1007/978-3-642-03549-4\_6`.

[215] Jaideep Vaidya. “Privacy-preserving linear programming”. In: *Proceedings of the 2009 ACM Symposium on Applied Computing (SAC), Honolulu, Hawaii, USA, March 9-12, 2009*. Ed. by Sung Y. Shin and Sascha Ossowski. ACM, 2009, pp. 2002–2007. DOI: `10.1145/1529282.1529729`.

[216] Yuan Hong et al. *Privacy Preserving Linear Programming*. arXiv:1610.02339 [cs.CR]. 2016. DOI: `10.48550/ARXIV.1610.02339`.

[217] Ivan Damgård et al. “Confidential Benchmarking Based on Multiparty Computation”. In: *Financial Cryptography and Data Security - 20th International Conference, FC 2016, Christ Church, Barbados, February 22-26, 2016, Revised Selected Papers*. Ed. by Jens Grossklags and Bart Preneel. Vol. 9603. Lecture Notes in Computer Science. Springer, 2016, pp. 169–187. DOI: `10.1007/978-3-662-54970-4\_10`.

[218] Philippe Golle. “A Private Stable Matching Algorithm”. In: *Financial Cryptography and Data Security, 10th International Conference, FC 2006, Anguilla, British West Indies, February 27-March 2, 2006, Revised Selected Papers*. Ed. by Giovanni Di Crescenzo and Aviel D. Rubin. Vol. 4107. Lecture Notes in Computer Science. Springer, 2006, pp. 65–80. DOI: `10.1007/11889663\_5`.

[219] Matthew K. Franklin, Mark A. Gondree, and Payman Mohassel. “Improved Efficiency for Private Stable Matching”. In: *Topics in Cryptology - CT-RSA 2007, The Cryptographers’ Track at the RSA Conference 2007, San Francisco, CA, USA, February 5-9, 2007, Proceedings*. Ed. by Masayuki Abe. Vol. 4377. Lecture Notes in Computer Science. Springer, 2007, pp. 163–177. DOI: 10.1007/11967668_11.

[220] David Gale and Lloyd S. Shapley. “College Admissions and the Stability of Marriage”. In: *The American Mathematical Monthly* 120.5 (2013), pp. 386–391.

[221] Jack Doerner, David Evans, and Abhi Shelat. “Secure Stable Matching at Scale”. In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*. Ed. by Edgar R. Weippl et al. ACM, 2016, pp. 1602–1613. DOI: 10.1145/2976749.2978373.

[222] M. Sadegh Riazi et al. “Toward Practical Secure Stable Matching”. In: *Proc. Priv. Enhancing Technol.* 2017.1 (2017), pp. 62–78. DOI: 10.1515/popets-2017-0005.

[223] Andrew Chi-Chih Yao. “How to Generate and Exchange Secrets (Extended Abstract)”. In: *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*. IEEE Computer Society, 1986, pp. 162–167. DOI: 10.1109/SFCS.1986.25.

[224] Stefan Wüller et al. “Using Secure Graph Algorithms for the Privacy-Preserving Identification of Optimal Bartering Opportunities”. In: *Proceedings of the 2017 on Workshop on Privacy in the Electronic Society, Dallas, TX, USA, October 30 - November 3, 2017*. Ed. by Bhavani Thuraisingham and Adam J. Lee. ACM, 2017, pp. 123–132. DOI: 10.1145/3139550.3139557.

[225] Sebastiaan de Hoogh, Berry Schoenmakers, and Meilof Veeningen. “Certificate Validation in Secure Computation and Its Use in Verifiable Linear Programming”. In: *Progress in Cryptology - AFRICACRYPT 2016 - 8th International Conference on Cryptology in Africa, Fes, Morocco, April 13-15, 2016, Proceedings*. Ed. by David Pointcheval, Abderrahmane Nitaj, and Tajjeeddine Rachidi. Vol. 9646. Lecture Notes in Computer Science. Springer, 2016, pp. 265–284. DOI: 10.1007/978-3-319-31517-1_14.

[226] Berry Schoenmakers and Meilof Veeningen. “Universally Verifiable Multiparty Computation from Threshold Homomorphic Cryptosystems”. In: *Applied Cryptography and Network Security - 13th International Conference, ACNS 2015, New York, NY, USA, June 2-5, 2015, Revised Selected Papers*. Ed. by Tal Malkin et al. Vol. 9092. Lecture Notes in Computer Science. Springer, 2015, pp. 3–22. DOI: 10.1007/978-3-319-28166-7_1.

[227] Mustafa Akgül. “The Linear Assignment Problem”. In: *Combinatorial Optimization*. Ed. by Mustafa Akgül, Horst W Hamacher, and Süleyman Tüfekçi. Springer, 1992, pp. 85–122. ISBN: 978-3-642-77489-8.

[228] Dimitri P. Bertsekas. “Auction algorithms for network flow problems: A tutorial introduction”. In: *Comput. Optim. Appl.* 1.1 (1992), pp. 7–66. DOI: 10.1007/BF00247653.

[229] Rainer E. Burkard and Eranda Çela. “Linear Assignment Problems and Extensions”. In: *Handbook of Combinatorial Optimization*. Ed. by Ding-Zhu Du and Panos M. Pardalos. Springer, 1999, pp. 75–149. DOI: 10.1007/978-1-4757-3023-4_2.

[230] Mauro Dell’Amico and Paolo Toth. “Algorithms and codes for dense assignment problems: the state of the art”. In: *Discret. Appl. Math.* 100.1-2 (2000), pp. 17–48. DOI: 10.1016/S0166-218X(99)00172-9.

[231] Lyle Ramshaw and Robert E. Tarjan. *On Minimum-Cost Assignments in Unbalanced Bipartite Graphs*. HP Laboratories Technical Report, HPL-2012-40R1. 2012.

[232] Abdelrahaman Aly and Sara Cleemput. *A Fast, Practical and Simple Shortest Path Protocol for Multiparty Computation*. Cryptology ePrint Archive, Paper 2017/971. https://eprint.iacr.org/2017/971. 2017.

[233] Dimitri P Bertsekas. “Auction Algorithms”. In: *Encyclopedia of Optimization*. 2009.

[234] H. W. Kuhn. “The Hungarian method for the assignment problem”. In: *Naval Research Logistics Quarterly* 2.1-2 (1955). ISSN: 00281441. DOI: 10.1002/nav.3800020109.

[235] H. W. Kuhn. “Variants of the hungarian method for assignment problems”. In: *Naval Research Logistics Quarterly* 3.4 (1956). ISSN: 00281441. DOI: 10.1002/nav.3800030404.

[236] James Munkres. “Algorithms for the Assignment and Transportation Problems”. In: *Journal of the Society for Industrial and Applied Mathematics* 5.1 (1957). ISSN: 0368-4245. DOI: 10.1137/0105003.

[237] Dimitri P Bertsekas. “A distributed algorithm for the assignment problem”. In: *Lab. for Information and Decision Systems Working Paper, MIT* (1979).

[238] T. B. Boffey and Dimitri P. Bertsekas. “Linear Network Optimization: Algorithms and Codes.” In: *The Journal of the Operational Research Society* 45.4 (1994). ISSN: 01605682. DOI: 10.2307/2584223.

[239] Dimitri P Bertsekas. *Network optimization: continuous and discrete models*. Athena Scientific, 1998, p. 593. ISBN: 1886529027, 9781886529021.

[240] Roy Jonker and A. Volgenant. “A shortest augmenting path algorithm for dense and sparse linear assignment problems”. In: *Computing* 38.4 (1987), pp. 325–340. DOI: 10.1007/BF02278710.

[241] Manuel Blum, Paul Feldman, and Silvio Micali. “Non-Interactive Zero-Knowl. and Its Applications (Extended Abstract)”. In: *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*. Ed. by Janos Simon. ACM, 1988, pp. 103–112. DOI: 10.1145/62212.62222.

[242] Benedikt Bünz et al. “Bulletproofs: Short Proofs for Confidential Transactions and More”. In: *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*. IEEE Computer Society, 2018, pp. 315–334. DOI: 10.1109/SP.2018.00020.

[243] Charles H. Bennett and Gilles Brassard. “Quantum cryptography: Public key distribution and coin tossing”. In: vol. 560. 2014, pp. 7–11. DOI: 10.1016/j.tcs.2014.05.025.

[244] Eleni Diamanti et al. “Practical challenges in quantum key distribution”. In: *npj Quantum Information* 2.1 (2016), p. 16025. ISSN: 2056-6387. DOI: 10.1038/npjqi.2016.25.

[245] Bruno Huttner et al. “Long-range QKD without trusted nodes is not possible with current technology”. In: *npj Quantum Information* 8.1 (2022), p. 108. ISSN: 2056-6387. DOI: 10.1038/s41534-022-00613-4.

[246] D Lopez et al. “Madrid Quantum Communication Infrastructure: a testbed for assessing QKD technologies into real production networks”. In: *Optical Fiber Comm. Conference (OFC) 2021*. Optica Publishing Group, 2021, Th2A.4. DOI: 10.1364/OFC.2021.Th2A.4.

[247] Sana Amairi-Pyka. “Industrialization of quantum communication technologies”. In: *PhotonicsViews* 18.6 (2021), pp. 28–31. DOI: https://doi.org/10.1002/phvs.202170609.

[248] Ralf-Peter Braun and Marc Geitz. “The OpenQKD Testbed in Berlin”. In: *Asia Communications and Photonics Conference 2021*. Optica Publishing Group, 2021, p. M4C.2. DOI: 10.1364/ACPC.2021.M4C.2.

[249] Jörn Müller-Quade and Renato Renner. “Composability in quantum cryptography”. In: *New Journal of Physics* 11.8 (Aug. 2009), p. 85006. DOI: 10.1088/1367-2630/11/8/085006.

[250] Mark N Wegman and Larry Carter. “New Hash Functions and Their Use in Authentication and Set Equality”. In: *J. Comput. Syst. Sci.* 22 (1981), pp. 265–279.

[251] Christopher Portmann. “Key Recycling in Authentication”. In: *IEEE Trans. Inf. Theory* 60.7 (2014), pp. 4383–4396. DOI: `10.1109/TIT.2014.2317312`.

[252] H Zbinden et al. “Quantum cryptography”. In: *Applied Physics B* 67.6 (1998), pp. 743–748. ISSN: 1432-0649. DOI: `10.1007/s003400050574`.

[253] Norbert Lütkenhaus. “Estimates for practical quantum cryptography”. In: *Phys. Rev. A* 59.5 (1999), pp. 3301–3319. DOI: `10.1103/PhysRevA.59.3301`.

[254] Gerhard Humer et al. “A Simple and Robust Method for Estimating Afterpulsing in Single Photon Detectors”. In: *Journal of Lightwave Technology* 33.14 (2015), pp. 3098–3107. DOI: `10.1109/JLT.2015.2428053`.

[255] R Ursin et al. “Entanglement-based quantum communication over 144 km”. In: *Nature Physics* 3.7 (2007), pp. 481–486. ISSN: 17452473. DOI: `10.1038/nphys629`.

[256] Caleb Ho, Antia Lamas-Linares, and Christian Kurtsiefer. “Clock synchronization by remote detection of correlated photon pairs”. In: *New Journal of Physics* 11.4 (2009), p. 14.

[257] M Jacak et al. “Testing of feasibility of QKD EPR S405 Quelle system deployment in commercial metropolitan fiber networks”. In: *Research in Optical Sciences*. Optica Publishing Group, 2014, JW2A.40. DOI: `10.1364/HILAS.2014.JW2A.40`.

[258] Charles H Bennett, Gilles Brassard, and N David Mermin. “Quantum cryptography without Bell’s theorem”. In: *Phys. Rev. Lett.* 68.5 (Feb. 1992), pp. 557–559. DOI: `10.1103/PhysRevLett.68.557`.

[259] Jesús Martínez-Mateo et al. “Demystifying the information reconciliation protocol cascade”. In: *Quantum Inf. Comput.* 15.5&6 (2015), pp. 453–477. DOI: `10.26421/QIC15.5-6-6`.

[260] Thomas Brochmann Pedersen and Mustafa Toyran. “High performance information reconciliation for QKD with CASCADE”. In: *Quantum Inf. Comput.* 15.5&6 (2015), pp. 419–434. DOI: `10.26421/QIC15.5-6-4`.

[261] Haokun Mao, Yucheng Qiao, and Qiong Li. “High-Efficient Syndrome-Based LDPC Reconciliation for Quantum Key Distribution”. In: *Entropy* 23.11 (2021), p. 1440. DOI: `10.3390/e23111440`.

[262] Xiangyu Wang et al. “High-speed implementation of length-compatible privacy amplification in continuous-variable quantum key distribution”. In: *IEEE Photonics Journal* 10.3 (2018), pp. 1–10. ISSN: 19430655. DOI: `10.1109/JPHOT.2018.2824316`.

[263] Yang Li et al. “High-throughput GPU layered decoder of quasi-cyclic multi-edge type low density parity check codes in continuous-variable quantum key distribution systems.” In: *Scientific reports* 10.1 (Sept. 2020), p. 14561. ISSN: 2045-2322 (Electronic). DOI: 10.1038/s41598-020-71534-5.

[264] Shen-Shen Yang, Zhen-Guo Lu, and Yong-Min Li. “High-Speed Post-Processing in Continuous-Variable Quantum Key Distribution Based on FPGA Implementation”. In: *J. Lightwave Technol.* 38.15 (Aug. 2020), pp. 3935–3941.

[265] Shen-Shen Yang et al. “An FPGA-Based LDPC Decoder With Ultra-Long Codes for Continuous-Variable Quantum Key Distribution”. In: *IEEE Access* 9 (2021), pp. 47687–47697. DOI: 10.1109/ACCESS.2021.3065776.

[266] Zhiliang Yuan et al. “10-Mb/s Quantum Key Distribution”. In: *J. Lightwave Technol.* 36.16 (Aug. 2018), pp. 3427–3433.

[267] Shengjun Ren et al. “Demonstration of high-speed and low-complexity continuous variable quantum key distribution system with local local oscillator”. In: *Scientific Reports* 11.1 (2021), p. 9454. ISSN: 2045-2322. DOI: 10.1038/s41598-021-88468-1.

[268] Fadri Grünenfelder et al. “Performance and security of 5 GHz repetition rate polarization-based quantum key distribution”. In: *Applied Physics Letters* 117.14 (2020), p. 144003. DOI: 10.1063/5.0021468. eprint: https://doi.org/10.1063/5.0021468.

[269] Gilles Brassard and Louis Salvail. “Secret-Key Reconciliation by Public Discussion”. In: *Advances in Cryptology - EUROCRYPT ’93, Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings*. Ed. by Tor Helleseth. Vol. 765. Lecture Notes in Computer Science. Springer, 1993, pp. 410–423. DOI: 10.1007/3-540-48285-7_35.

[270] Christoph Pacher et al. “An information reconciliation protocol for secret-key agreement with small leakage”. In: *IEEE International Symposium on Information Theory, ISIT 2015, Hong Kong, China, June 14-19, 2015*. IEEE, 2015, pp. 730–734. DOI: 10.1109/ISIT.2015.7282551.

[271] David Elkouss et al. “Rate compatible protocol for information reconciliation: An application to QKD”. In: *2010 IEEE Information Theory Workshop on Information Theory (ITW 2010, Cairo)*. 2010, pp. 1–5. DOI: 10.1109/ITWKSPS.2010.5503195.

[272] Hossein Mani et al. “Multiedge-type low-density parity-check codes for continuous variable quantum key distribution”. In: *Phys. Rev. A* 103 (6 June 2021), p. 062419. DOI: 10.1103/PhysRevA.103.062419.

[273] David S. Slepian and Jack K. Wolf. “Noiseless coding of correlated information sources”. In: *IEEE Trans. Inf. Theory* 19.4 (1973), pp. 471–480.

[274] Aaron D. Wyner and Jacob Ziv. “The rate-distortion function for source coding with side information at the decoder”. In: *IEEE Trans. Inf. Theory* 22.1 (1976), pp. 1–10.

[275] Valerio Scarani et al. “The security of practical quantum key distribution”. In: *Reviews of Modern Physics* 81.3 (2009), pp. 1301–1350. ISSN: 00346861. DOI: 10.1103/RevModPhys.81.1301.

[276] David Elkouss et al. “Rate Compatible Protocol for Information Reconciliation: An application to QKD”. In: *CoRR* abs/1006.2660 (2010). arXiv: 1006.2660.

[277] Jesús Martínez-Mateo, David Elkouss, and Vicente Martin. “Blind reconciliation”. In: *Quantum Inf. Comput.* 12.9-10 (2012), pp. 791–812. DOI: 10.26421/QIC12.9-10-5.

[278] Ueli M. Maurer. “Secret key agreement by public discussion from common information”. In: *IEEE Trans. Inf. Theory* 39.3 (1993), pp. 733–742. DOI: 10.1109/18.256484.

[279] Fabian Furrer. “Reverse-reconciliation continuous-variable quantum key distribution based on the uncertainty principle”. In: *Phys. Rev. A* 90 (4 Oct. 2014), p. 042325. DOI: 10.1103/PhysRevA.90.042325.

[280] Anthony Leverrier, Frédéric Grosshans, and Philippe Grangier. “Finite-size analysis of a continuous-variable quantum key distribution”. In: *Physical Review A - Atomic, Molecular, and Optical Physics* 81.6 (2010). ISSN: 10502947. DOI: 10.1103/PhysRevA.81.062343.

[281] Mohammad G. Raeini and Mehrdad Nojoumian. “Secure error correction using multiparty computation”. In: (2018), pp. 468–473. DOI: 10.1109/CCWC.2018.8301702.

[282] Thomas J. Richardson and Rüdiger L. Urbanke. “The capacity of low-density parity-check codes under message-passing decoding”. In: *IEEE Trans. Inf. Theory* 47.2 (2001), pp. 599–618. DOI: 10.1109/18.910577.

[283] Robert G. Gallager. *Low-Density Parity-Check Codes*. MIT Press, Cambridge, MA. 1963.

[284] Jon Feldman, Martin J. Wainwright, and David R. Karger. “Using linear programming to Decode Binary linear codes”. In: *IEEE Trans. Inf. Theory* 51.3 (2005), pp. 954–972. DOI: 10.1109/TIT.2004.842696.

[285] Jon Feldman. “Decoding error-correcting codes via linear programming”. PhD thesis. Massachusetts Institute of Technology, Cambridge, MA, USA, 2003.

[286] Dinka Milovancev et al. “Ultra-Low Noise Balanced Rec. with 20 dB Quantum-to-Classical Noise Clearance at 1 GHz”. In: *European Conference on Optical Communication, ECOC 2021, Bordeaux, France, September 13-16, 2021*. IEEE, 2021, pp. 1–4.

[287] Dinka Milovancev et al. “Towards Integrating True Random Number Generation in Coherent Optical Transceivers”. In: *IEEE Journal of Selected Topics in Quantum Electronics* 26.5 (2020), pp. 1–8.

[288] T.∼Beth, J.∼Müller-Quade, and R.∼Steinwandt. “Cryptanalysis of a Practical Quantum Key Distribution With Polarization-Entangled Photons”. In: *Quantum Information and Computation* 5.3 (2005), pp. 181–186.

[289] Alfred J Menezes, Paul C Van Oorschot, and Scott A Vanstone. *Handbook of applied cryptography*. CRC press, 1996.

[290] M Sasaki et al. “Field test of quantum key distribution in the Tokyo QKD Network.” In: *Optics Express* 19.11 (2011), p. 21.

[291] Aysajan Abidin and Jan-Åke Larsson. “Vulnerability of “A novel protocol authentication algorithm ruling out a man-in-the-middle attack in quantum cryptography””. In: *International Journal of Quantum Information* 7.5 (Aug. 2009), pp. 1047–1052.

[292] Wassily Hoeffding. “Probability Inequalities for Sums of Bounded Random Variables”. In: *J. Amer. Statistical Assoc.* 58.301 (1963), pp. 13–30.

[293] Oskar van Deventer et al. “Towards European standards for quantum technologies”. In: *EPJ Quantum Technology* 9.1 (2022), p. 33. ISSN: 2196-0763. DOI: `10.1140/epjqt/s40507-022-00150-1`.

[294] Joo Yeon Cho and Andrew Sergeev. “Using QKD in MACsec for secure Ethernet networks”. In: *IET Quantum Communication* 2.3 (2021), pp. 66–73. DOI: `https://doi.org/10.1049/qtc2.12006`.

[295] Benjamin Dowling, Torben Brandt Hansen, and Kenneth G. Paterson. “Many a Mickle Makes a Muckle: A Framework for Provably Quantum-Secure Hybrid Key Exchange”. In: *Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020, Paris, France, April 15-17, 2020, Proceedings*. Ed. by Jintai Ding and Jean-Pierre Tillich. Vol. 12100. Lecture Notes in Computer Science. Springer, 2020, pp. 483–502. DOI: `10.1007/978-3-030-44223-1\_26`.

[296] Nimrod Aviram et al. *Practical (PQ) Key Combiners from One-Wayness and Applications to TLS*. Cryptology ePrint Archive, Paper 2022/065. `https://eprint.iacr.org/2022/065`. 2022.

[297] Stephanie Wehner, David Elkouss, and Ronald Hanson. “Quantum internet: A vision for the road ahead”. In: *Science* 362.6412 (2018), eaam9288. DOI: `10.1126/science.aam9288`.

[298] Louis Salvail et al. “Security of trusted repeater quantum key distribution networks”. In: *J. Comput. Secur.* 18.1 (2010), pp. 61–87. DOI: `10.3233/JCS-2010-0373`.

[299] Stephan W. Jahn, Markus Plass, and Farid Moinfar. “Digital pathology: Advantages, limitations and emerging perspectives”. In: *Journal of Clinical Medicine* 9.11 (2020), pp. 1–17. ISSN: 20770383. DOI: `10.3390/jcm9113697`.

[300] Yvo Desmedt and Yongge Wang. “Perfectly Secure Message Transmission Revisited”. In: Lecture Notes in Computer Science 2332 (2002). Ed. by Lars R. Knudsen, pp. 502–517. DOI: `10.1007/3-540-46035-7\_33`.

[301] Manuel B. Santos, Paulo Mateus, and Armando N. Pinto. “Quantum Oblivious Transfer: A Short Review”. In: *Entropy* 24.7 (2022), p. 945. DOI: `10.3390/e24070945`.

[302] Alex B. Grilo et al. “Oblivious Transfer Is in MiniQCrypt”. In: *Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17-21, 2021, Proceedings, Part II*. Ed. by Anne Canteaut and François-Xavier Standaert. Vol. 12697. Lecture Notes in Computer Science. Springer, 2021, pp. 531–561. DOI: `10.1007/978-3-030-77886-6\_18`.

[303] Marcel Keller, Emmanuela Orsini, and Peter Scholl. “MASCOT: Faster Malicious Arithmetic Secure Computation with Oblivious Transfer”. In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*. Ed. by Edgar R. Weippl et al. ACM, 2016, pp. 830–842. DOI: `10.1145/2976749.2978357`.