institute of telecommunications

Subliminal Channels in High-Speed Signatures

Master's Thesis

for obtaining the academic degree

Diplom-Ingenieur

as part of the study

Electrical Engineering and Information Technology

carried out by

Alexander Hartl

student number: 01125115

Institute of Telecommunications
at TU Wien

Supervision:
Univ. Prof. Dipl.-Ing. Dr.-Ing. Tanja Zseby
Dipl.-Ing. Robert Annessi, B.Sc

Acknowledgments

I want to express my gratitude for everyone who assisted or encouraged me in various ways during my studies.

In particular, I would like to thank Prof. Tanja Zseby and Dipl.-Ing. Robert Annessi for giving me the opportunity to write this thesis, for supporting me actively while I was working on it and for many valuable comments and suggestions.

I owe special thanks to my family, especially my parents Edith and Rudolf, for their endless support during all these years.

Thank you, Sabrina, for encouraging me and for so many cheerful hours in my life.

Abstract

One of the fundamental building blocks for achieving security in data networks is the use of digital signatures. A digital signature is a bit string which allows the receiver of a message to ensure that the message indeed originated from the apparent sender and has not been altered along the path. In certain cases, however, the functioning of signature schemes allows an adversary to additionally utilize the signature string as a hidden information channel. These channels are termed subliminal channels and have been known and tolerated since the 80s. Due to the recent progress in the development of high-speed signature algorithms, however, application scenarios for digital signatures become feasible that lead to a large exploitable bit rate for data exfiltration, given that the deployed signature scheme allows the utilization as subliminal channel.

This thesis shows how certain high-speed signature schemes can be exploited to carry hidden information. In particular, we analyse the recent EdDSA signature scheme, which yields substantial future potential, as well as the class of Multivariate Quadratic (MQ) signature schemes. We discuss how an adversary can proceed to embed and recover subliminal information and what bit rate the adversary can achieve for transmitting hidden information. Scenarios like signed NTP broadcasts, signed sensor data transmissions and the TLS key exchange are depicted, where the existence of a subliminal channel gives rise to new attack possibilities threatening network security. To confirm these findings we discuss the results of performed experiments, which attest a considerable subliminal bandwidth to the analysed signature schemes.

Furthermore, we depict several methods for preventing the exploitation of subliminal channels in EdDSA, but we have to conclude that none of them is viable in a practical situation, reinforcing the threats that originate from the described subliminal channels.

Abstract

Digitale Signaturen sind ein wesentlicher Bestandteil sicherer Kommunikation in modernen Datennetzwerken. Eine digitale Signatur ist ein Datenblock, der es dem Empfänger einer Nachricht erlaubt, zu prüfen, ob die Nachricht unverändert ist und tatsächlich vom vorgeblichen Absender stammt. Darüber hinaus ermöglichen es jedoch viele Signaturverfahren einem feindlich gesinnten Absender, die Signaturen als versteckte Kommunikationskanäle zu verwenden. Obwohl diese sogenannten Subliminal Channels seit den 80er Jahren bekannt sind, wurden sie bei der Protokollentwicklung oft toleriert. Durch die kürzlich erfolgte Entwicklung von hochperformanten Signaturverfahren werden allerdings Anwendungsgebiete von digitalen Signaturen möglich, welche einem Angreifer erhebliche verborgene Datenübertragungsraten ermöglichen, sofern die Signaturverfahren Subliminal Channels erlauben.

Diese Arbeit zeigt die notwendigen Schritte zum Einbetten und Wiedergewinnen von versteckter Information in EdDSA-Signaturen und in auf multivariaten Polynomen basierenden Signaturverfahren auf und diskutiert die hierbei erreichbaren Bandbreiten. Um die praktischen Auswirkungen der beschriebenen Subliminal Channels auf die Informationssicherheit zu erfassen, werden zudem Angriffsszenarien bei Anwendungsgebieten wie signierten NTP-Broadcasts, signierten Sensordatenübertragungen im Smart Grid und Schlüsselvereinbarung bei TLS beschrieben. Bei diesen Anwendungen ist der Einsatz der neuen Signaturverfahren wahrscheinlich und damit potentiell eine heimliche Datenübertragung erheblichen Ausmaßes möglich. Um die Ergebnisse empirisch zu bestätigen, werden einige der Angriffsszenarien praktisch umgesetzt. Die Ergebnisse dieser Experimente bescheinigen den untersuchten Subliminal Channels eine wesentliche Bedeutung für sicherheitskritische Umgebungen.

Für EdDSA im Speziellen werden außerdem verschiedene Methoden aufgezeigt, um die Ausnutzung jeglicher Subliminal Channels zu unterbinden, welche allerdings mit deutlichen negativen Auswirkungen auf den Signierprozess verbunden sind. Die beschriebenen Methoden sind daher in der Praxis nur in den wenigsten Fällen umsetzbar, womit von Subliminal Channels weiterhin ein Risiko ausgeht, das bei der Verwendung von EdDSA bzw. von Signaturverfahren im Allgemeinen beachtet werden muss.

Parts of this thesis have been published in the

Proceedings of the 9th International Workshop on Managing Insider Security Threats (MIST'17) [1]

and the

Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA) Vol. 9.1 [2].

Contents

List of Figures

List of Tables

List of Abbreviations

BGP Border Gateway Protocol.
DNS Domain Name System.
DSA Digital Signature Algorithm.
ECDSA Elliptic Curve Digital Signature Algorithm.
EdDSA Edwards-Curve Digital Signature Algorithm.
HFE Hidden Field Equations.
HTTP Hypertext Transfer Protocol.
IETF Internet Engineering Task Force.
IKE Internet Key Exchange.
MAC Message Authentication Code.
MQ Multivariate Quadratic.
MQQ Multivariate Quadratic Quasigroups.
NTP Network Time Protocol.
OFB Output Feedback.
PMU Phasor Measurement Unit.
PTP Precision Time Protocol.
RSA Rivest–Shamir–Adleman signature scheme.
SSH Secure Shell.
STS Stepwise Triangular Systems.
TLS Transport Layer Security.
UOV Unbalanced Oil and Vinegar.
VRF Verifiable Random Function.
VUF Verifiable Unpredictable Function.

List of Symbols

General Symbols

$\mathbb{F}_q$ Galois field of order q.
$\mathbb{Z}_L$ Set of nonnegative integers smaller L.
$\mathbb{Z}_L^*$ Set of positive integers smaller L.
B_s Subliminal bandwidth.
M Message.

Symbols for EdDSA

k	Private key.
a	Signing key.
A	Public key.
r	Random nonce.
R	Random nonce point.
S	Integer part of signature.
B	Base point.
L	Order of subgroup generated by B.

Symbols for MQ Signatures

$\boldsymbol{h}$	Message hash.
$\boldsymbol{s}$	Signature.
$\boldsymbol{x}$	Signature transformed by S.
$\boldsymbol{y}$	Message hash transformed by T.
$\boldsymbol{z}$	Vinegar variables.
$\hat{\boldsymbol{x}}$	Oil variables.
$\boldsymbol{\rho}$	Random data for minus modification.
$S(\boldsymbol{s})$	Affine mapping for signature.
$\boldsymbol{S}_i^{(l)}$	Linear part of S for equation i.
$S_i^{(a)}$	Affine part of S for equation i.
$F(\boldsymbol{x})$	Central mapping.
$T(\boldsymbol{y})$	Affine mapping for message hash.
$P(\boldsymbol{s})$	Composition of T, F and S.
$\boldsymbol{P}_i^{(q)}$	Quadratic part of P for equation i.
$\boldsymbol{P}_i^{(l)}$	Linear part of P for equation i.
$P_i^{(a)}$	Affine part of P for equation i.
n	Dimension of message hash.
m	Dimension of signature.
r	Number of removed public key equations.
v	Number of vinegar variables.
$\mathbb{E}$	nth order extension field of $\mathbb{F}_q$.
$\tilde{x}, \tilde{y}$	Extension field variables.
$\Phi(\tilde{x})$	Isomorphism between $\mathbb{E}$ and $\mathbb{F}_q^n$.

1 Introduction

The Internet becomes increasingly important for today's society and serves as a fundamental building block for modern technology. As an inevitable consequence, security and privacy in computer networks become more and more essential and a large number of techniques have been found for reaching these goals to a practically satisfactory extent. One of the basic elements for achieving secure network communication are digital signatures. A digital signature as used today is a bit string appended to a message to allow the receiver of the message to verify the message's origin and to protect its contents from modifications along the path. With these properties digital signatures can under certain conditions indeed be considered an electronic replacement for hand-written signatures. In fact, however, when used correctly, digital signatures allow a substantially higher level of security than hand-written signatures.

The obvious use case for digital signatures is signing emails or other electronic personal communication. In fact, however, digital signatures are used in a wide variety of further applications, in many cases without the user even knowing about the complex processes under the hood. Examples include protection of web traffic, guaranteeing genuineness of software or proving one's identity to remote servers. Algorithms for obtaining digital signatures are commonly referred to as signature schemes and an impressing number of such schemes have been invented so far for suiting the different needs of the application scenarios. Still, only very few of them, notably RSA [3], DSA [4] and ECDSA [5], are of widespread use today and have proved to provide a very high level of security also in practice.

While this seems all good, digital signatures can introduce a security risk that was first shown by Simmons [7, 6] and is often neglected or tolerated when deploying signatures. Simmons described the possibility of hiding data in signatures, hence transmitting it unnoticeably for any observer and allowing data to be exfiltrated. He called these hidden information channels *subliminal channels*.

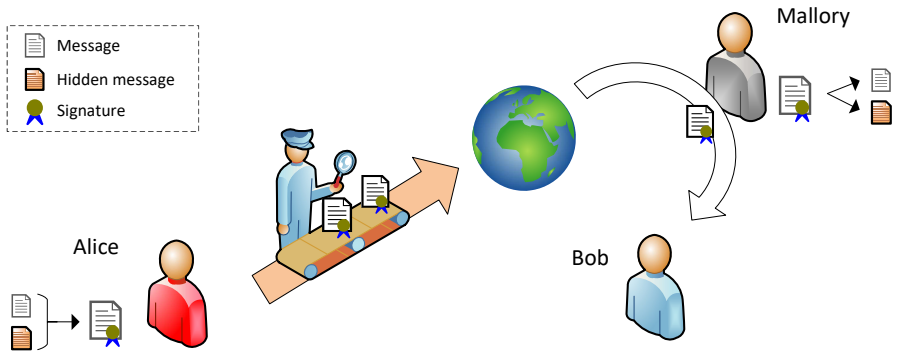

Figure 1: Illustration of subliminal channels.

Fig. 1 illustrates the functioning of subliminal channels. In this scenario, Alice uses the Internet to transmit a message to Bob. To allow Bob to verify that the message truly originates from her, Alice appends a digital signature. Alice furthermore wants to transmit a message to Mallory, but unlike her communication with Bob, she wants to make sure that nobody who is able to inspect the messages she sends, learns about her communication with Mallory or even suspects any hidden communication. Alice knows that Mallory can intercept her messages to Bob.
A subliminal channel in the signature scheme allows Alice to achieve this very task. Alice can encode her message to Mallory in the signature and Mallory will be able to recover her secret message from the signed overt message, mostly using a secret Alice has shared with Mallory upfront. The astonishing aspect of this approach is that neither Bob nor anybody who observes the communication will be suspicious, as the message is unchanged and the signature passes verification successfully. Alice' overt communication is fully visible and intelligible to someone inspecting her messages and ostensibly does not contain any suspicious content.
Let us, for example, assume that Alice is located in a security-critical, heavily monitored, enterprise network. She can use this channel to leak a document containing sensitive information without giving any monitoring technique the chance to unveil the data exfiltration or just expect its existence.
Subliminal channels have been identified in many signature schemes so far. However, recently new constructions of signature schemes raised the interest of the cryptographic community, which exhibit a very high performance considering signing and verification speed, meeting the demands of new application areas of digital signatures. Examples of such high-speed signatures are the Edwards-Curve Digital Signature Algorithm (EdDSA) [8], PFlash [9] or MQQ-SIG [10]. The question arises whether and, if so, how subliminal channels can also be established exploiting these new schemes.

1.1 Motivation

The concept of subliminal channels was introduced around the year 1980 [7, 6]. Subliminal channels, hence, have been known and tolerated for a notable time now without being attributed too much importance. In fact, throughout the current decade only a small amount of research has been conducted about identifying new subliminal channels or preventing their exploitation so far. The question arises why this is the case.
A reason why subliminal channels mostly were not (and often are still not) classified as major threats is their former rare use in network protocols. Only recently application scenarios have emerged, where signatures are transmitted with considerable frequency. Reasons why such application scenarios of signature schemes were not of great importance earlier, are twofold:

1. On the one hand, the large computational overhead of signatures prevented their high-rate use in protocols. Especially on low-power devices the signing and verification process of a large number of messages per time unit was computationally infeasible. With recent developments in the field of high-speed signature schemes, however, the computational costs for signing and verification drop significantly and new use cases become possible.

2. On the other hand, the deployment of an increasing amount of machine-to-machine communication is a phenomenon of present-day engineering. In the field of machine-to-machine communication often large communication demands occur. These communication demands frequently come in conjunction with high security requirements, giving rise to the use of signature schemes for authentication purposes.

Signatures are thus used in network protocols directly in addition to being used for signing messages transmitted by the users. Examples of such application scenarios include clock synchronization protocols, smart grid communication or communication in cyberphysical systems in general. In these scenarios signatures are transmitted with a high frequency. Hence, if the used schemes yield a subliminal channel, a vast bit rate might result for data exfiltration by an adversary and the consequences for information security might be unacceptable.
We therefore observe that unlike traditional use cases subliminal channels can constitute severe security threats for important applications of today. The risks that follow from subliminal channels have to be reevaluated.

1.2 Aims of this Thesis

In this work we discuss the possibility of subliminal channels for new high-speed signature schemes. One of such schemes is EdDSA. EdDSA is a recently developed scheme that allows both, high security and a superior signing and verification performance. It is, hence, an important alternative to currently used signature schemes and has substantial significance for future deployments.
Furthermore, schemes based on MQ cryptography have to be considered in this context. Even though this area of cryptography has not yet proven to yield secure signature schemes, it potentially yields several attractive properties, thus attracting the interest of the cryptographic community. For example, some of the proposed schemes like PFlash or MQQ-SIG have a very good signing and verification performance, which motivates investigating them in the present context.
The possibility of subliminal channels has not yet been researched for these signature schemes. Considering the risks that might follow from the possibility of clandestine data transmissions, hence, important security-related questions arise:

Do these schemes allow subliminal channels? What subliminal bandwidths do these channels yield? Do they constitute a real-world threat and is it possible to prevent their active exploitation by adversaries?
We will, hence, investigate the signature schemes' mathematical structures and point out possibilities for exploitation as subliminal channels. We will discuss where these signatures are likely to be used in the future and what risks can follow from their use by an adversary in these cases.

1.3 Methodological Approach

The goals of this thesis can be broken down into specific results about possibilities of subliminal channels in EdDSA or MQ-based signatures on the one hand, and general results about subliminal channels in high-speed signatures on the other hand. Furthermore, we enclose an empirical part to confirm these findings in practice. The following approaches were used for gathering information about subliminal channels and for judging their impact on information security.

Identifying Subliminal Channels

How can subliminal channels be identified? If a given message deterministically maps to a certain signature string using a given private key, the signer cannot influence the signing process to obtain different signature strings. To start with, we can therefore look out for randomness being used during signature generation, i.e. for data that is used during signing which is not derived from the message or the key but stems from a (pseudo)random number generator. Evidently, then the signer cannot hide information utilizing data transmissions he cannot influence.
In fact, however, subliminal channels can exist under the more general requirement that multiple valid signature strings can be found for a given message and public key. The signer in this case might be able to deviate from the usual signing process to find different valid signatures for the message, encoding information into the choice he makes. To prove that a signature scheme does not allow any subliminal channels, we thus have to show that only one signature can exist for any given message and public key.
In order for the subliminal channel to be well usable, the signer not only has to be able to modify the signature string to embed information, the intended subliminal receiver also has to be able to recover the information from the signature string. Hence, when directly encoding the subliminal information into randomness used throughout the signing process, we have to investigate ways to recover this randomness from the signature. It is then useful to assess if requirements have to be met for these recovery methods to function and, hence, if the subliminal channel is exploitable also in a practical scenario.

For this thesis the above methods are used for investigating MQ signatures for subliminal channels. For MQ signatures we thus first analyse if randomness is used during signature generation and, thereupon, analyse if this randomness can efficiently be used as a subliminal channel. Due to the current uncertainty of the viability of specific MQ-based signature schemes, we first conduct these considerations from a very general perspective, and eventually specialize them to particular proposed signature schemes.
On the other hand, for EdDSA, we have recourse to a significant amount of existing research for signature schemes like DSA or ECDSA which are to some degree similar to EdDSA, simplifying the search for subliminal channels. Instead of reinventing the wheel we thus first examine if the subliminal channels known from these signature schemes can be adopted for EdDSA. In this context it is also of main interest which methods for preventing the subliminal communication have been proposed for these preceding schemes. We thus review which prevention techniques have been proposed and if they can be used with EdDSA.

Impact Analysis

Given that the signature schemes yield subliminal channels, the question arises if they can be used in a way that threatens information security. We thus select application scenarios of high-speed signatures where a large number of messages is signed and transmitted per time unit giving rise to a large exploitable bit rate for data transmission by an adversary. Further properties of scenarios raising the security-related impact of our results, is the use in security-critical settings or a large number of receivers of the signed messages.
Finally, deployment of signatures in protocols with wide-spread use apparently yields potential security threats. We will thus evaluate the risks that result from the subliminal channel resulting from the use of EdDSA in the Transport Layer Security (TLS) [11, 12] protocol in comparison to other information hiding techniques.

Empirical Evaluation

In order to confirm our findings, the theoretical results have to be evaluated in practice. To keep the effort for performing this task low, we use existing (open-source) implementations of the signature algorithms and modify them to do the embedment and recovery of the subliminal information. The subliminal channels have to be tested with many different choices of overt and subliminal information to ensure proper functioning. Furthermore, the subliminal channels are tested in conjunction with different scenarios described in the course of impact analysis. On the other hand, the actual data that is transmitted in these scenarios is not of high importance in our case and also precise timing, which in practice might be

crucial when using high-speed signatures, does not have to be extremely accurate in our case. This justifies the use of a simplified setup deploying virtual machines for running the experiments instead of dedicated hardware.

1.4 Outline

After introducing the most important building blocks of cryptography in Chapter 2, we begin by depicting scenarios where high-speed signatures are likely to be used in Chapter 4 to determine the impact of any potential subliminal communication for information security. Examples of such scenarios are signed NTP [13] broadcasts or smart grid communication. We here discuss in which cases a subliminal channel is likely to be exploitable and what it might be used for by an adversary.
Having introduced the environment where the digital signatures will be used, Chapter 5 details on possibilities for establishing subliminal channels in EdDSA. We describe how both a broadband subliminal channel and a narrowband subliminal channel can be used, where the narrowband channel allows significantly less data to be transmitted, but also has less stringent requirements. Probably most interesting, we furthermore highlight some mitigation strategies for preventing any subliminal communication. We also depict limitations of the mitigation strategies. Unfortunately, due to these limitations the practical usability of the mitigation techniques is almost ruled out and limited to very specific situations.
Chapter 6 elaborates on possibilities for establishing subliminal channels exploiting MQ signatures. As this term refers to a technique for constructing signature schemes rather than a particular signature scheme, we show how different methods that are used for constructing MQ signatures can directly be exploited as subliminal channel or at least cause a subliminal channel to exist. We then specialize these findings to specific signature schemes that have been proposed up to now and discuss the resulting subliminal bandwidths.
In Chapter 7 we detail on TLS as a real-world protocol and analyse what impact the discovered subliminal channels can have on information security. Considering the extremely widespread use of TLS, this evaluation has a major significance. On the other hand, given the existence of other covert channels in TLS, the subliminal channels' importance might be not that high, depending on the scenario and the adversary's goal. We discuss in which cases a subliminal channel in signatures can be used in general and when it is reasonable for an adversary to exploit it.
Finally, in Chapter 8 we describe practical experiments we conducted to confirm our findings. In these experiments we implemented some of the use cases depicted in Chapter 4 and evaluated the practically achievable bit rate for adversarial communication.

2 Cryptographic Fundamentals

Security in data networks has to be considered in a wide variety of different environments. Depending on the particular environment and the type of data that has to be protected, certain aspects of information security are of tremendous importance while others might be less important. We can therefore define different security objectives to classify these needs. ISO 7498 [14] also introduces the term *security services* for these objectives.
For example, the following security objectives are the most important for protecting a communication process [15].

- *Confidentiality* means that the transmitted information can only be received by the intended recipient.
- *Integrity* means that the exchanged message can be ensured to not have been modified on the path, be it unintentionally as a result of transmission errors or intentionally by an adversary.
- *Authenticity* means that the communicating parties can be sure about the identity of their communication partners. In the present context we are mainly interested in *data origin authentication*, which implies integrity for a message but additionally ensures that the message originated from the alleged sender.
- *Non-repudiation* allows the recipient to prove to a third-party like, for example, a legal authority, that the message indeed originated from the alleged sender.

In modern system engineering these objectives are mostly achieved by using cryptographic methods. Cryptography is the "study of mathematical techniques related to aspects of information security" [15].
In the following sections we discuss several cryptographic techniques relevant for the remainder of this work.

2.1 Digital Signatures

This thesis is about digital signatures, which yield integrity, authenticity and non-repudiation by cryptographic means [16]. Hence, if Alice transmits a message to Bob and signs it using a digital signature scheme, Bob can be sure that the message indeed originated from Alice and was not altered along the path.

For using a digital signature, Alice first has to create a key pair, consisting of a *private key*[1] and a *public key*.
For the most important class of signature schemes, *signatures with appendix*, the private key can, from a very general perspective, be thought of as a set of mappings $S_r \colon \mathcal{M} \to \mathcal{S}$. Here, $\mathcal{M}$ is the set of all messages, $\mathcal{S}$ denotes a set of bit strings termed *signatures* and $r \in \mathcal{R}$, where $\mathcal{R}$ is called the *indexing set for signing*.
The public key can be thought of as a mapping $V \colon \mathcal{M} \times \mathcal{S} \to \{\text{valid, invalid}\}$, so that $V(M, S_r(M)) = \text{valid}$ for all $M \in \mathcal{M}$ and $r \in \mathcal{R}$ [15]. Alice communicates her public key to Bob in an authenticated (e.g. offline) way and keeps her private key secret. In order for the signature scheme to be secure, it must be computationally infeasible to find an $(M, s) \in \mathcal{M} \times \mathcal{S}$, that satisfies $V(M, s) = \text{valid}$ without knowing the private key.
With this setup Alice can allow Bob to ensure authenticity and integrity for her messages. For this purpose, she picks an $r \in \mathcal{R}$, computes $s = S_r(M)$ for her message M and transmits (M, s) to Bob. On receiving the message, Bob verifies if indeed $V(M, s) = \text{valid}$ and rejects the message otherwise.
Signature schemes can be classified into *randomized* and *deterministic* schemes. For a randomized signature scheme $|\mathcal{R}| > 1$, otherwise $|\mathcal{R}| = 1$.
Moreover, in the present context it is relevant to highlight different methods for combining digital signing with encryption. For this purpose three different methods can be distinguished [17]:

- When using the *Sign-then-Encrypt* method, the plaintext is first signed and encryption is applied to the signed message.
- When using the *Encrypt-then-Sign* method, the plaintext is first encrypted and a signature is appended to the ciphertext. The signature is computed over the ciphertext.
- When using the *Encrypt-and-Sign* method, the plaintext is first encrypted and a signature is appended to the ciphertext. However, in this case the signature is computed over the plaintext instead of the ciphertext.

2.2 Information Hiding

The possibility of leaking information using channels that originally were never intended to be used in this way, constitutes a security threat that is often overlooked in practice. Such methods can be coarsely classified as *steganographic techniques*. Steganographic methods make information "difficult to notice" in contrast to cryptographic methods which make information "difficult to recognize" [18].

[1] The private key is sometimes also termed the *secret key*.

One of the first methods of such kind exploiting digital technology was hiding information in digital media like pictures or movies [18], for example modulating the least significant bits of the pixels. Trying to exploit these methods for malware communication, however, the malware has to be able to modify user data. This might be difficult as the malware then has to determine which files are about to be transmitted, which in most cases is strongly dependent on the user's behaviour. Furthermore, such modifications are likely to be recognized as a user's files change.

Covert and Subliminal Channels

This thesis focuses on information hiding techniques exploiting network protocols and, in particular, cryptographic algorithms used in these protocols. Information hiding techniques exploiting network protocols are commonly termed *covert channels* [18]. Examples of covert channels include protocol header fields which are rarely used or explicitly contain random data. Also meta information like timing between packets, the order of packets or packet loss can be used, achieving good concealment properties as these phenomena also occur to a certain degree in the regular operation of data networks. Chapter 7 will provide several more detailed examples of covert channels in the context of the TLS protocol.
Information hiding techniques that exploit cryptographic schemes are commonly termed *subliminal channels* [6, 18]. Attractive carriers for subliminal channels are digital signatures as they seem innocuous, since they are usually not supposed to carry any information. Moreover, they are used in many scenarios for providing authenticity and integrity.
Adopting frequent notation, in this thesis we will term the data hidden in signatures the *subliminal information* and the transferable amount of subliminal information the *subliminal bandwidth* or just the bandwidth, when there is no risk of confusion. Furthermore, we term the transmitter of the subliminal information the *subliminal sender* and its recipient the *subliminal receiver*.
Subliminal channels in signatures are sometimes classified according to their subliminal bandwidth. *Broadband* subliminal channels allow (almost) all the signature's bits that are not used for providing its security against forgery to be used for carrying subliminal information. *Narrowband* subliminal channels allow a significantly lower subliminal bandwidth and typically yield just a few bits [19] per signature. Broadband subliminal channels have been identified in major signature schemes like DSA [19, 4] or ECDSA [20, 21, 5].

Unique Signatures

Finally, signature schemes that provably do not allow any subliminal channels are called *subliminal-free* or *unique* [21]. It is interesting to note that the problem of

finding subliminal-free signature schemes occurs in the context of pseudorandom number generation as well. A Verifiable Random Function (VRF) is a pseudorandom number generator that is parameterized by a private key and public key in such a way that the generator can only be queried for a particular seed if the private key is known, but at the same time everyone who knows the public key is able to verify if the VRF has been used in the correct way [22].
A related term is a Verifiable Unpredictable Function (VUF), which is a function that is hard to compute when not having the private key, but is not necessarily pseudorandom. A VUF can be used for constructing a VRF and is recognized to be conceptually equivalent to a unique signature scheme [23, 24]. Exemplary use cases of VRFs are a micropayment system [25] or a privacy-preserving escrow system [26]. Thus, the investigation of subliminal-free signature schemes appears important also in this respect.

2.3 Groups and Fields

Cryptographic algorithms are based on a multitude of different mathematical concepts. However, one of the most important concepts used in public key cryptography is that of finite groups and finite fields. The book [27] gives a good introduction. As these concepts are also essential for some of the algorithms described in this thesis, this section shall summarize the most important terms from [27].
A *group* $\mathcal{G}$ is a set equipped with a binary operation $*\colon \mathcal{G} \times \mathcal{G} \to \mathcal{G}$ satisfying the following properties:

- The operation $*$ is associative, i.e. $a * (b * c) = (a * b) * c \ \ \forall\, a, b, c \in \mathcal{G}$.
- There is an *identity element* $e \in \mathcal{G}$, for which $a * e = e * a = a \ \ \forall\, a \in \mathcal{G}$ holds.
- For all $a \in \mathcal{G}$ an *inverse element* $a^{-1} \in \mathcal{G}$ can be found, which satisfies $a * a^{-1} = a^{-1} * a = e$.

Furthermore, the following properties are of particular importance:

- If $a * b = b * a \ \ \forall\, a, b \in \mathcal{G}$, the group is called *abelian.*
- The group is called *cyclic* if a $a \in \mathcal{G}$ can be found, so that every $b \in \mathcal{G}$ can be represented by recursively applying the operation $*a$ a finite number of times to a, i.e. $b = a * a * \cdots * a$.
- The group is called *finite* if the number of elements it contains $|\mathcal{G}|$ is finite. $|\mathcal{G}|$ is then called the *order* of $\mathcal{G}$.

A *field* is a set $\mathcal{F}$ equipped with two binary operations $+$ and $\cdot$, so that the following properties hold:

- $(\mathcal{F}, +)$ is an abelian group. Its identity element will be denoted as 0.
- $(\mathcal{F}\backslash\{0\}, \cdot)$ is an abelian group.
- The operation $\cdot$ is associative.
- Distributivity holds for $+$ and $\cdot$, i.e. $a \cdot (b + c) = a \cdot b + a \cdot c$ and $(b + c) \cdot a = b \cdot a + c \cdot a \ \forall a, b, c \in \mathcal{F}$.

Similar to groups a field $\mathcal{F}$ is called finite if $|\mathcal{F}|$ is finite. $|\mathcal{F}|$ is then called the order of $\mathcal{F}$.

2.3.1 Galois Fields

Finite field are also called *Galois fields* and are of fundamental importance for this thesis. It can be shown that all finite fields with given order q are *isomorphic,* which means that a bijection $f\colon \mathcal{F}_1 \to \mathcal{F}_2$ can be found between two such fields $\mathcal{F}_1, \mathcal{F}_2$ that preserves the group operations, i.e. $f(a+b) = f(a)+f(b)$ and $f(a \cdot b) = f(a) \cdot f(b)$ for all $a, b \in \mathcal{F}_1$. This theorem gives rise to speaking of *the* finite field of order q (as opposed to *a* finite field). In this thesis we will denote the Galois field of order q as $\mathbb{F}_q$.
An example of finite fields is calculation modulo some prime integer p. The set $\mathcal{F}$ in this case consists of the integers $0, \ldots, (p-1)$ and the addition and multiplication operations $+$ and $\cdot$ are usual integer addition and multiplication, taking the result modulo p. It can be shown that for $(\mathcal{F}, +, \cdot)$ all field axioms hold.

2.3.2 Elliptic Curves

When using cryptographic algorithms based on finite groups, it is possible to use the multiplication operation of finite fields as described above for this purpose. As alternative it is possible to construct finite groups using *elliptic curves.* An elliptic curve over a field $\mathcal{F}$ is defined as [16]

$$\mathcal{E} = \{(x, y) \in \mathcal{F} \times \mathcal{F} : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6\}, \tag{1}$$

where $a_1, a_2, a_3, a_4, a_6 \in \mathcal{F}$ have to satisfy certain conditions (see [16]) to ensure that the curve is non-singular. Defining a suitable group operation certain elliptic curves can be shown to satisfy the group axioms and, as it turns out, use of elliptic curves for cryptography allows significantly shorter keys to achieve the same level of security [16].
An example of elliptic curves is *twisted Edwards curves,* which are defined over a finite field $\mathbb{F}_q$ with $2 \nmid q$ as [28]

$$\mathcal{E}_{a,d} = \{(x, y) \in \mathbb{F}_q \times \mathbb{F}_q : ax^2 + y^2 = 1 + dx^2y^2\} \tag{2}$$

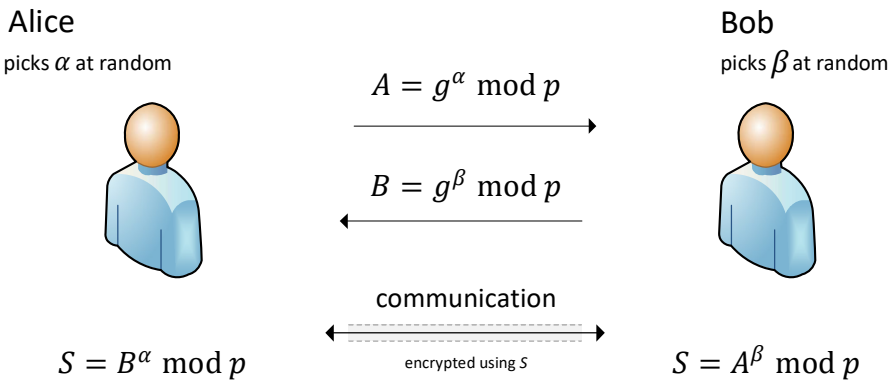

Figure 2: The Diffie-Hellman key exchange.

with distinct $a, d \in \mathbb{F}_q \backslash \{0\}$. Together with the addition operation

$$(x_1, y_1) + (x_2, y_2) = \left(\frac{x_1 y_2 + y_1 x_2}{1 + d x_1 x_2 y_1 y_2}, \frac{y_1 y_2 - a x_1 x_2}{1 - d x_1 x_2 y_1 y_2} \right) \tag{3}$$

these curves can be shown to satisfy the group axioms with identity element $(0, 1)$. The inverse element of (x, y) then is $(-x, y)$.

2.4 Discrete Logarithms and Diffie-Hellman

Many of the cryptographic schemes used today are based on the *discrete logarithm problem*, i.e. the problem of computing logarithms in cyclic groups. Considering the field of remainders modulo some prime p, for example, the *discrete exponential function* for base $g \in \{2, \ldots, p-1\}$ is defined as the function $y = g^k \bmod p$ for given $k \in \mathbb{Z}_p^*$. The *discrete logarithm function* inverts the discrete exponential function, i.e. for a given $y \in \mathbb{Z}_p^*$ it finds a $k \in \mathbb{Z}_p^*$, so that $y = g^k \bmod p$ holds. While the discrete exponential function can be computed efficiently in at most $2 \log_2 k$ steps using the square-and-multiply method [29], significantly more steps are required for computing discrete logarithms. For example, an important algorithm for performing this task is the Baby-Step-Giant-Step algorithm [29], which needs at most $\sqrt{p}$ steps to find a result. For large p, finding a solution quickly becomes infeasible.

An important example of algorithms based on the discrete logarithm problem is the Diffie-Hellman key exchange [30] depicted in Fig. 2. This protocol allows two communicating parties to agree on a common secret for encrypting their communication without the need for any upfront key exchange, if it can be assumed that an attacker is unable to unnoticeably alter the key exchange messages.

The algorithm uses a public prime number p and a public $g \in \{2, \ldots, p-1\}$. During algorithm execution Alice chooses a secret number $\alpha \in \mathbb{Z}_p^*$ at random and

calculates $A = g^\alpha \bmod p$. Equally, Bob chooses a secret $\beta \in \mathbb{Z}_p^*$ and calculates $B = g^\beta \bmod p$. Now Alice and Bob transmit A and B to the respective other. Considering the discrete logarithm problem, neither Alice is able to deduce Bob's secret β from B, nor is Bob able to deduce α from A, nor is an attacker who intercepts A and B able to find any of the secrets. However, Alice and Bob are able to find a common secret S as $S = A^\beta \bmod p = g^{\alpha\beta} \bmod p = B^\alpha \bmod p$. They can thus use S for encrypting their communication.
While Diffie-Hellman key exchange would be broken if computing discrete logarithms were easy, the reverse is not necessarily true. There might be a method for computing $g^{\alpha\beta} \bmod p$ from $g^\alpha \bmod p$ and $g^\beta \bmod p$. This problem is called the *Diffie-Hellman problem* and is also believed to be hard [15].
Sometimes the functioning of the Diffie-Hellman key exchange is further differentiated according to the lifespan of the participants' secrets (see e.g. [11]). For *static* Diffie-Hellman the same secrets α, β are used for all connections, allowing the values A, B to be announced as public keys. For *ephemeral* Diffie-Hellman these values are chosen anew for each connection.

2.5 EdDSA Signatures

The Edwards-Curve Digital Signature Algorithm (EdDSA) [8] constitutes the first signature algorithm that shall be analysed throughout this thesis. It was introduced in 2011 by Bernstein et al. as a well performing alternative to today's signature schemes in terms of speed and security. The algorithms use point addition on the twisted Edwards curve

$$\mathcal{E} = \left\{(x, y) \in \mathbb{F}_q \times \mathbb{F}_q : -x^2 + y^2 = 1 + dx^2y^2\right\}. \tag{4}$$

The scheme has several parameters: the prime q, the parameter $d \in \mathbb{F}_q$ defining the curve, a base point $B \in \mathcal{E}$, the order of the subgroup generated by B denoted as L, a cofactor 2^c with integer c such that $2^c L = |\mathcal{E}|$ (the number of points on the curve) and an integer $b \in \mathbb{N}$, the length of the private key, where $b \approx \log_2 L$. Furthermore, cryptographic hash functions H, H_a and H_r are used which we here model as directly producing values in $\mathbb{Z}_L$. For Ed25519, which is EdDSA used together with the curve Curve25519 from [31], these parameters are standardized in [32]. The RFC defines a further scheme, Ed448, which uses an untwisted Edwards curve [33] and provides 224-bit security rather than 128-bit. Ed25519 and Ed448 yield signature lengths of 512 bit and 912 bit, respectively.
The private key consists of a b-bit string k. It is mapped to a number $a = H_a(k)$. Knowledge of a is sufficient for producing valid signatures, which justifies considering a the actual secret for generating signatures. To differentiate it from the private key k, we will hence call a the *signing key*. The public key consists of a point on the curve $A = aB$.

To generate a signature for a message M, first a nonce value

$$r = H_r(k, M) \tag{5}$$

has to be derived. The signature consists of two parts: (1) a point $R = rB$ and (2) a number $S = r + H(R, A, M)a \bmod L$. For verification the receiver has to check the group equation

$$2^c SB = 2^c R + 2^c H(R, A, M)A. \tag{6}$$

EdDSA is based on a digital signature scheme that was first described by Schnorr [34]. A main concern when using this kind of signatures is that r has to be chosen unpredictably [8]. Indeed, if an adversary can guess r correctly for a signed message (M, S, R), then the signing key a can simply be computed as $a = (S - r)/H(R, A, M) \bmod L$ [8], where A and L can be considered public and, thus, known by the adversary. Furthermore, if the same nonce value has been used for generating signatures of different messages M_1 and M_2, the signing key a can be found as well as $a = (S_1 - S_2)/(H(R, A, M_1) - H(R, A, M_2)) \bmod L$. Both issues can be addressed by deriving r from the message and the private key as done for EdDSA (see Eq. (5)). This is in contrast to ECDSA, where the issue of choosing an appropriate value for r is left to the implementation, which has to use a (pseudo)random number generator for this purpose.

2.6 Signatures based on MQ Cryptography

In addition to EdDSA, signature schemes based on MQ cryptography are relevant for this work and shall be introduced in this section.
Signature schemes and, more generally, public key cryptography systems need to be based on problems that are mathematically hard to solve. All such systems that are widely used today are based on just very few problems. To be precise, DSA, EdDSA and similar schemes are based on the discrete logarithm problem. RSA is based on the problem of factoring large integers. The problem of solving systems of polynomial equations in finite fields would be a possible alternative to today's cryptographic environment. When used for cryptography the used polynomials are usually restricted to degree two and, hence, the problem is called the Multivariate Quadratic polynomials problem (MQ problem). The resulting signature schemes might have particularly attractive properties: On the one hand, some candidates that were developed up to now showed to have a very good performance in terms of signing and verification speed compared to established schemes. Furthermore, in contrast to both integer factorization and the discrete logarithm problem, no quantum algorithm is known that could threaten the schemes' security if progress in quantum computing is made. For these reasons, finding a signature scheme based on multivariate cryptography is an active research area.

Unfortunately, most candidates that were found so far showed to be insecure. In order to find new schemes, it has become common to apply modifications to existing, broken schemes. Therefore, a large number of modifications have been developed to a comparatively low number of basic trapdoors. The most promising candidate that was found up to now is Hidden Field Equations (HFE) [35]. The basic trapdoor was broken in [36, 37, 38]. However, in conjunction with appropriate modifications a secure signature scheme might result. To be precise, the *minus modification* and *vinegar variables* seem promising (HFEv-). Both modifications will be discussed in Section 2.6.3. HFEv- is currently being evaluated by the PQCRYPTO project for post-quantum security [39].

2.6.1 Signature Generation and Verification

Fig. 3 depicts the basic functioning of MQ signature schemes [40]. The public key of MQ signatures consists of a quadratic polynomial system $P\colon \mathbb{F}_q^m \to \mathbb{F}_q^n$, where $q \in \mathbb{N}$ is the order of the underlying Galois field, $m \in \mathbb{N}$ is the dimension of the signature and $n \in \mathbb{N}$ is the dimension of the used message hash value. Inversion of P is expected to be infeasible for an attacker who does not know how P was constructed. The private key contains this information and usually consists of two bijective mappings $S\colon \mathbb{F}_q^m \to \mathbb{F}_q^m$ and $T\colon \mathbb{F}_q^n \to \mathbb{F}_q^n$ and a central quadratic mapping $F\colon \mathbb{F}_q^m \to \mathbb{F}_q^n$, so that $P = T \circ F \circ S$.
For signing and verification a cryptographic hash function H is applied to the message, yielding the value $\boldsymbol{h} = H(M)$, where $\boldsymbol{h} \in \mathbb{F}_q^n$. In the course of signature generation the signer has to find a vector $\boldsymbol{s} \in \mathbb{F}_q^m$, so that $\boldsymbol{h} = P(\boldsymbol{s})$. To find such a vector, the signer first computes $\boldsymbol{y} = T^{-1}(\boldsymbol{h})$. In the next step he tries to find a vector $\boldsymbol{x} \in \mathbb{F}_q^m$, for which $\boldsymbol{y} = F(\boldsymbol{x})$ holds. F is a quadratic function. Hence, if it consisted of polynomials with random coefficients, this problem would be as hard as solving $\boldsymbol{h} = P(\boldsymbol{s})$ in the first place and infeasible to solve. However, F is constructed with a particular structure that allows inversion in a straight-forward manner as discussed in Section 2.6.2. Finally, knowing $\boldsymbol{x}$, the signer is able to find $\boldsymbol{s}$ as $\boldsymbol{s} = S^{-1}(\boldsymbol{x})$.

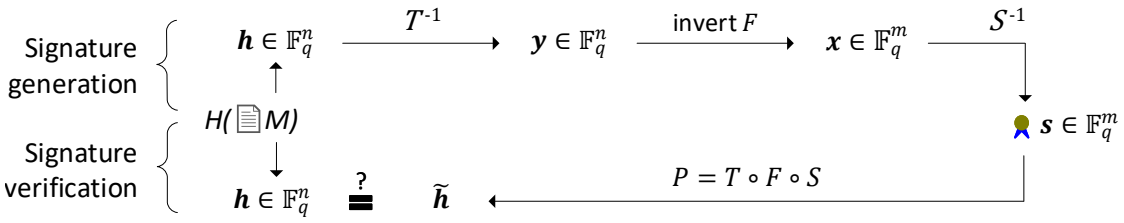

Figure 3: The basic operation principle of MQ signature schemes.

Knowing a signature $\boldsymbol{s}$, verification is simple. By applying $P = T \circ F \circ S$ to $\boldsymbol{s}$, the signed message hash $\boldsymbol{h}$ can be regained and compared to the actual hash $H(M)$. An attacker who tries to forge a signature only has P and is confronted with the problem of finding an $\boldsymbol{s}$ that solves $\boldsymbol{h} = P(\boldsymbol{s})$. If these quadratic polynomials had random coefficients, this would be hard, corresponding to the MQ problem. The attack approaches developed so far for many MQ signature schemes target at exploiting the structure of F that is meant to be hidden by the use of S and T. Unfortunately, attacks have been found against all trapdoors found so far. Instead of devising completely new constructions it has thus become common to modify existing (broken) schemes to yield a more secure signature scheme.

2.6.2 Trapdoors

There are different approaches for constructing the central mapping F. The survey paper [40] gives a good overview of the most important ones. In this section the approaches will be discussed but not described in too much detail. For detailed information the reader is referred to the original publications referenced throughout the following sections. If no modification is applied, attacks have been found against all of the below trapdoors.

Unbalanced Oil and Vinegar

The initial idea of schemes based on vinegar variables was by Patarin in [41]. In the course of signature generation the signer has to invert the equations

$$F_i(\hat{\boldsymbol{x}}, \boldsymbol{z}) = \sum_{j=1}^{n}\sum_{k=1}^{v} \gamma_{ijk}\hat{x}_j z_k + \sum_{j=1}^{v}\sum_{k=1}^{v} \lambda_{ijk} z_j z_k + \sum_{j=1}^{n} \xi_{ij}\hat{x}_j + \sum_{j=1}^{v} \xi'_{ij} z_j + \delta_i, \qquad (7)$$

where $\gamma_{ijk}, \lambda_{ijk}, \xi_{ij}, \xi'_{ij}, \delta_i \in \mathbb{F}_q$ are chosen randomly by the signer during key generation. Furthermore, $\boldsymbol{z} \in \mathbb{F}_q^v$ denote the $v \in \mathbb{N}$ vinegar variables that are chosen at random by the signer for each signature. Fixing these variables the equations (7) turn into an affine equation system in the variables $\hat{x}_i$, that are referred to as oil variables. The equation system is nonsingular with non-negligible probability and, when this is the case, can easily be inverted by the signer. Otherwise he tries different values for the vinegar variables until he finds an invertible equation system. Finally, the signature is formed by transforming both the oil and the vinegar variables using the inverse of the secret affine mapping $S\colon \mathbb{F}_q^{n+v} \to \mathbb{F}_q^{n+v}$, thus hiding the vinegar variables. Note that for the notation introduced in Fig. 3, we set $m = n + v$ and $\boldsymbol{x}^T = (\,\hat{\boldsymbol{x}}^T\, \boldsymbol{z}^T\,)$.
While we have as many oil variables as vinegar variables in the case of the initial, balanced scheme [41], we have more vinegar than oil variables for Unbalanced Oil and Vinegar schemes [42], which were proposed to yield more security, after balanced oil and vinegar schemes had been cryptanalyzed.

Matsumoto-Imai Scheme

The Matsumoto-Imai Scheme (C^*) was first described in [43, 44]. It uses an extension field $\mathbb{E}$ of $\mathbb{F}_q$ to define the mapping F. The central polynomials F are expressed as $\Phi \circ \tilde{F} \circ \Phi^{-1}$, where $\Phi \colon \mathbb{E} \to \mathbb{F}_q^n$ is a secret bijection between an n-dimensional vector of the ground field and the extension field $\mathbb{E}$ of degree n. In the case of C^*, $\tilde{F}$ is the monomial

$$\tilde{F}(\tilde{x}) = \tilde{x}^{q^\lambda+1}, \tag{8}$$

where $\tilde{x} \in \mathbb{E}$ and λ is an integer, so that $q^n - 1$ and $q^\lambda + 1$ are coprime. The latter condition ensures that equation (8) can easily be solved for $\tilde{x}$. If the condition is met, an $h = (q^\lambda+1)^{-1} \bmod (q^n-1)$ can be found and the inverse can be computed as

$$\tilde{F}^{-1}(\tilde{y}) = \tilde{y}^h. \tag{9}$$

Hidden Field Equations

Similarly to the trapdoor C^*, Hidden Field Equations (HFE) uses an extension field $\mathbb{E}$ to define F. C^* has been generalized to HFE by Patarin [35] after being cryptanalyzed. Instead of a monomial, HFE uses a polynomial

$$\tilde{F}(\tilde{x}) = \sum_{\substack{0 \le i,j \le d \\ q^i+q^j \le d}} C_{i+1,j+1} \tilde{x}^{q^i+q^j} + \sum_{\substack{0 \le i \le d \\ q^i \le d}} B_{i+1} \tilde{x}^{q^i} + A \tag{10}$$

with $C_{i,j}, B_i, A \in \mathbb{E}$ and a degree $d \in \mathbb{N}$. In theory, the mapping could be constructed to be a bijection by choosing a permutation polynomial for $\tilde{F}(\tilde{x})$. However, Patarin assumed it to be very difficult to find suitable permutation polynomials, so the coefficients of $\tilde{F}(\tilde{x})$ are usually chosen at random. Hence, the most important difference to C^* in this context is that in general F is no bijection anymore. This means that for some messages there might be multiple signatures and for others there might be none at all. To be able to sign all possible messages, it is necessary to include randomness in the signed data using, for example, the minus or vinegar variables modifications described in Section 2.6.3.

During the signing process, Berlekamp's algorithm can be used for finding the roots of the polynomial Eq. (10) but also other methods have been proposed to invert the central mapping [35, 45]. In total, however, inversion of the central mapping is not as efficient as for C^*.

At present, HFE is one of the most preferred candidates for constructing secure signature schemes by applying appropriate modifications.

Multivariate Quadratic Quasigroups

Gligoroski, Markovski, and Knapskog proposed a trapdoor based on Multivariate Quadratic Quasigroups (MQQ) in [46, 47]. We here describe the slightly modified variant described in [10].
In MQQ the mapping $F\colon \mathbb{F}_2^n \to \mathbb{F}_2^n$ is constructed from a quasigroup $(\mathcal{G}, *)$ of order 2^d with $d \in \mathbb{N}$ a divisor of n as follows: First the input vector $\boldsymbol{x} \in \mathbb{F}_2^n$ is divided into n/d vectors in $\mathbb{F}_2^d$, which are interpreted as n/d group elements $X_1, \ldots, X_{n/d} \in \mathcal{G}$. The variables $Y_1, \ldots, Y_{n/d} \in \mathcal{G}$ are then obtained according to

$$Y_i = \begin{cases} X_i, & \text{if } i = 1 \\ X_{i-1} * X_i, & \text{if } i > 1 \wedge i \text{ odd} \\ X_i * X_{i-1}, & \text{if } i > 1 \wedge i \text{ even} \end{cases} . \tag{11}$$

Quasigroups that are used in MQQ show the bilinearity property and allow the operation $*$ to be executed in terms of simple matrix multiplications and to be inverted in a straight-forward way.
Finally, the variables Y_i are interpreted as n/d vectors in $\mathbb{F}_2^d$, which are concatenated to form the output $\boldsymbol{y} \in \mathbb{F}_2^n$ of the mapping $F(\boldsymbol{x})$.

Stepwise Triangular Systems

Stepwise Triangular Systems (STS) use a layered approach for inverting a system of multivariate equations. STS can further be divided into general STS and the more important regular STS.
In regular STS the inversion of the central mapping proceeds in $L \in \mathbb{N}$ layers, where n must be a multiple of L. In each layer $l = 1, \ldots, L$ the signer solves the system of $r = n/L$ polynomials

$$\begin{aligned} F_{(l-1)r+1}(\boldsymbol{x}) &= F'_{(l-1)r+1}(x_1, \ldots, x_{lr}) \\ &\vdots \\ F_{lr}(\boldsymbol{x}) &= F'_{lr}(x_1, \ldots, x_{lr}) \end{aligned}$$

where for layers $l > 1$ the variables $x_1, \ldots, x_{(l-1)r}$ are already known from the previous layers, so the equations have to be solved in the r unknowns $x_{(l-1)r+1}, \ldots, x_{lr}$. The coefficients of the polynomials are chosen in such a way to obtain a bijective mapping in each layer. This allows fast signing and results in the overall central mapping to be bijective as well.
In general STS the number of polynomials and used variables differs in each layer and bijectivity might be lost.

2.6.3 Modifications

Attacks have been found for all (unmodified) trapdoors described above. It has thus become common to apply modifications to these trapdoors to attain a (more) secure signature scheme. In this section only modifications are listed that aim to improve the schemes' security and not to improve their performance or key sizes.

The Minus Modification

One of the most popular methods is simply removing some of the public key polynomials [40], allowing only a part of the message hash to be recovered from the signature using the public key. Despite being very simple the modification prevents important attacks against MQ schemes and has been used in many proposed signature schemes.

Vinegar Variables

The HFE trapdoor can be enhanced by the use of vinegar variables [42]. In this case the signer picks $v \in \mathbb{N}$ vinegar variables $\boldsymbol{z} \in \mathbb{F}_q^v$ at random and the polynomial (10) is adapted according to

$$\tilde{F}(\tilde{x}, \boldsymbol{z}) = \sum_{\substack{0 \leq i,j \leq d \\ q^i + q^j \leq d}} C_{i+1,j+1} \tilde{x}^{q^i + q^j} + \sum_{\substack{0 \leq i \leq d \\ q^i \leq d}} B_{i+1}(\boldsymbol{z}) \tilde{x}^{q^i} + A(\boldsymbol{z}), \tag{12}$$

so B_i and A are now functions of the vinegar variables. Similar to UOV, $\hat{\boldsymbol{x}} = \Phi(\tilde{x})$ and $\boldsymbol{z}$ are transformed together by the affine mapping $S\colon \mathbb{F}_q^{n+v} \to \mathbb{F}_q^{n+v}$ to form the signature. To retain MQ shape, $B_i(\boldsymbol{z})$ have to be affine functions and $A(\boldsymbol{z})$ has to be a quadratic function.

Projection

Projection is also known as "fixing" [48]. The public key polynomials P are partly evaluated by setting $f \in \mathbb{N}$ elements of $\boldsymbol{x}$ to specific values during key generation, where $f < n$. As these f elements have to match by chance during signature generation, the signer has a probability of q^{-f} to obtain a valid signature when inverting the central mapping one time.

The Plus Modification

The plus method can be seen as the opposite to the minus modification. The signer adds $a \in \mathbb{N}$ random equations during key generation to the polynomials F [49]. For inverting the central mapping one time, we obtain a probability of q^{-a} for the random equations to hold as well.

Internal Perturbation

This modification aims to prevent common attacks by perturbing the structure by modifying the central mapping to

$$F'(\boldsymbol{x}) = F(\boldsymbol{x}) + \pi(l(\boldsymbol{x})), \tag{13}$$

where $l\colon \mathbb{F}_q^m \to \mathbb{F}_q^w$ with $w \in \mathbb{N}$ is an affine mapping and $\pi\colon \mathbb{F}_q^w \to \mathbb{F}_q^n$ is a system of quadratic functions [40, 50]. The coefficients of the affine mapping l and the quadratic polynomials π are chosen at random during key generation and, hence, the polynomials π cannot be inverted easily. Instead, a value for $l(\boldsymbol{x})$ has to be guessed during signing. Having the value for $l(\boldsymbol{x})$ fixed allows to subtract $\pi(l(\boldsymbol{x}))$ and invert the central mapping for $\boldsymbol{x}$. Finally, $\boldsymbol{x}$ is checked to produce the guessed value for $l(\boldsymbol{x})$ and the process is repeated otherwise. For inverting the central mapping one time, we obtain a probability of q^{-w} for the guessed values to be correct.

3 Related Work

Information hiding is a fascinating topic which has given rise to an impressive amount of research and a variety of different methods to conceal information. An introduction and overview is presented in [18]. In the following we point out important literature related to information hiding techniques discussed in this thesis.

Subliminal Channels

The concept of subliminal channels was introduced in 1978 and first publicly described in 1984 by Simmons [7, 6]. Simmons imagined two prisoners who are allowed to communicate with each other in terms of messages. As the prison warden aims to prevent the prisoners from coordinating an escape plan, he only passes on messages that are unencrypted so that he can read them. On the other hand, the prisoners fear the warden forging messages from the respective other such that they insist on the communication being authenticated using signatures. With this setting in mind, Simmons showed that information can be embedded in the signature such that it does not hamper successful verification of the signature.

Simmons furthermore showed how to construct narrowband channels that allow transmitting only a few bit of information as well as broadband channels that allow a significant amount of subliminal information to be added to a signature [19]. Broadband channels often require the receiver of the hidden information to know the signer's private key to recover the subliminal information. A noteworthy exception to this rule is the Newton channel, which was shown by Anderson et al. for the ElGamal signature scheme specifically [51]. When using the Newton channel, the signer unveils only as many bits of information of his private key to the subliminal receiver as should be available for the subliminal channel afterwards.

So far, subliminal channels have been shown to exist in many traditional signature schemes like the Digital Signature Algorithm (DSA) [19, 4], the Elliptic Curve Digital Signature Algorithm (ECDSA) [20, 21, 5], RSA [52, 53, 3] or the Rabin signature scheme [7, 54], and finding a mode of operation that is provably subliminal-free often turns out to be a difficult task. For example, for Schnorr signatures Zhang et al. [55] proposed an interactive protocol for achieving subliminal-freeness and Dong and Xiao [21] proposed an interactive protocol for ECDSA for the same purpose. Furthermore, Bohli, Vasco, and Steinwandt [20] proposed a method for constructing non-interactive subliminal-freeness proofs for generated ECDSA signatures. Two of these methods will be discussed in detail in Chapter 5.

SETUP and ASA Attacks

The concept of subliminal channels is related to SETUP (Secretly Embedded Trapdoor with Universal Protection) attacks that were introduced by Young and Yung [56]. When performing a SETUP attack, an adversary replaces a cryptographic algorithm on a victim's device by an altered algorithm aiming to break its security. In the context of digital signatures this means that the modified signing algorithm leaks the signer's private key to the adversary. The same authors also introduced the realm of Kleptography which is defined as the "study of stealing information securely and subliminally" [56].
Recently attacks based on modifying cryptographic algorithms attracted anew research interest and are now called Algorithm-Substitution Attacks (ASAs) [57, 58] and Subversion Attacks (SAs) [58]. Most noteworthy, the paper [58] provides a detailed discussion about the requirements for a signature scheme to be susceptible to SAs.

MQ Signatures

A good introduction to the mathematical structure of MQ signatures is given by [40]. The paper also discusses some important attacks on the schemes and details on the different methods for constructing MQ signatures. The most important trapdoor HFE was introduced by Patarin [35]. Other important publications include the concept of vinegar variables [41] or HFE's predecessor C^* [44], which was the first important scheme based on MQ cryptography.
Noteworthy examples of proposed signature schemes include QUARTZ [45], GUI [59], PFlash [9], Rainbow [60] or MQQ-SIG [10]. The possibility of subliminal channels has not yet been researched for these signature schemes and will be treated in Chapter 6.

TLS Covert Channels

Some research has been conducted into exploring information hiding techniques exploiting the TLS [11, 12] protocol. Goh et al. [61] discussed how several protocol fields can be exploited to allow an eavesdropper to successfully perform a key recovery attack. Moreover, covert channels in TLS are mentioned and analysed in several further publications [61, 62, 63, 64, 65].
Considering subliminal channels in TLS, on the other hand, only the key exchange method using RSA has been documented by Gołębiewski, Kutyłowski, and Zagórski [64] to allow a subliminal channel, even though the authors do not use the term "subliminal channel" for their method.
Chapter 7 will give a more detailed overview of information hiding techniques usable in TLS and of the relevant literature.

4 Adversarial Communication Scenarios

Digital signatures are used for securing communication in a variety of different scenarios, ranging from signing documents or emails to their use for providing authentication of communication partners in complex security protocols. In particular, due to its excellent properties regarding performance and security, EdDSA has been proposed or is already in use for many applications, protocols and use cases.
In addition to these existing use cases, high-speed signatures have been proposed for scenarios where a sender transmits a large number of packets and needs to ensure integrity and data origin authentication [66, 67, 68]. Such scenarios were not possible with traditional signature schemes like RSA, DSA or ECDSA due to their high computational requirements for the signing and/or the signature verification process. EdDSA, on the other hand, is an excellent choice for such scenarios as it is fast and lightweight enough to be applicable to scenarios with a high rate of signed messages and limited resources, such as broadcast clock synchronization and sensor data collection in the smart grid.
Given that the deployed high-speed signature scheme yields subliminal channels, however, the signature provides the possibility to establish secret communication channels to an adversary, resulting in severe security issues. Such scenarios are highlighted and analysed in this section.

4.1 Preconditions

As will be described in Chapters 5 and 6, the subliminal receiver has to receive secret key material from the signer to be able to use broadband subliminal channels for both EdDSA and MQ signatures. This is a frequent requirement for the use of subliminal channels. Additionally one of the following requirements has to be met:

1. The communication is unencrypted.

2. The communication is encrypted using the Encrypt-then-Sign method. In this case the subliminal receiver does not necessarily know the transmitted message. However, as the signature scheme is applied to the ciphertext of the message and the signature itself is unencrypted, the subliminal information can be recovered nevertheless.

3. For MQ signatures, recovery of the subliminal information is also possible if the communication is encrypted using the Encrypt-and-Sign method. When using the Encrypt-and-Sign method the signature itself is unencrypted and available to the subliminal receiver, but the signed message is unavailable when not being able to perform the decryption.

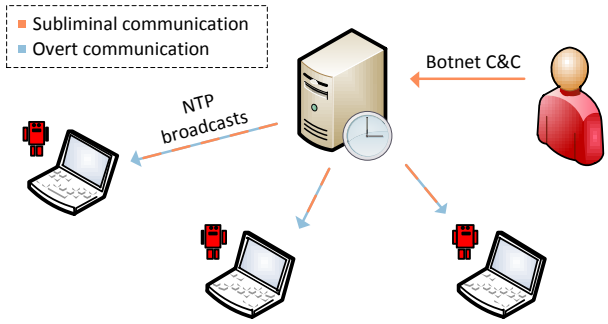

Figure 4: Botnet Command and Control using signed NTP broadcasts.

4. The subliminal receiver knows the decryption key or is able to break the encryption.

While (1) is a reasonable assumption for some application scenarios where confidentiality is of no importance, (4) can be achieved by leaking the decryption key upfront using, for example, a narrowband subliminal channel which is usable also for encrypted signatures.

4.2 Clock Synchronization Protocols

Accurate time information becomes increasingly important for the proper functioning of critical infrastructure. Consequently, protocols like the Network Time Protocol (NTP) [13] or the Precision Time Protocol (PTP) [69] are used for providing clock synchronization. If time synchronization is performed over insecure network links, however, data origin authentication is necessary for the protocol packets as otherwise an adversary might be given the chance to modify the transmitted time information [66, 70].

A method for authenticating NTP or PTP packets not only has to provide appropriate security, but also has to allow very fast signature generation and verification to keep the influence on the error of the transmitted time information low. High-speed signature schemes have been proposed to be used for this purpose [66, 67, 68]. Confidentiality is usually irrelevant for time information and, hence, encryption is likely to be omitted to safe resources. The signatures used for securing NTP and PTP packets thus provide ideal conditions for exploitation as a subliminal channel.

Time synchronization can happen either in unicast or in broadcast mode. As the impact of a subliminal channel is very different for these two cases, we here handle them separately.

Unicast Mode

In unicast mode a client regularly exchanges few messages with a time server, where poll intervals for NTPv4 range from 16 s to 36 h [13]. Signing each of these messages individually, the bit rate of the resulting subliminal channel may suffice for leaking sensible information, which is particularly severe if synchronization is performed across the Internet, thus leaving the protected network of a company.
In unicast mode it is essentially possible to achieve origin authentication by using MACs based on symmetric cryptography. However, signatures might still be used for sake of simplicity, e.g. to avoid key management and/or key agreement.

Broadcast Mode

In broadcast mode, time information is broadcast in regular intervals across a company's network. As these broadcasts occur in regular intervals, the amount of data that can be transferred using the subliminal channel is large when observing a large-enough time span. Furthermore, for receiving the leaked data, an adversary does not have to take special measures for eavesdropping on the signed messages as any network access suffices to become a regular receiver of the messages.
Finally, also in scenarios where the adversary wants to reach a large number of network nodes the signed time broadcasts yield an attractive subliminal channel. As example, Fig. 4 depicts the operation of a botnet. If the adversary has managed to install malware on a large number of network nodes and also has compromised the time server, he can use the subliminal channel for transmitting command and control messages to his bots. Approaches to discover the botnet by detecting the command and control communication then are foredoomed.

4.3 Smart Grid Communication

Smart grids enhance electrical grids by information technology to optimize grid operations. Power grids have to meet highest requirements in availability and quality of supply. As a consequence, for smart grids, high requirements for availability, integrity, authenticity and possibly also confidentiality, have to be considered.
Digital signatures are thus of fundamental importance for providing authentication. The use of high-speed signatures is favourable in many situations when signatures shall be processed on low-power hardware like sensor devices, if a large amount of continuously transmitted data has to be signed or if low latency is important.
In such a setting there are several reasons for why a subliminal channel has a significant impact on information security. The communicating partners often store sensible data like maintenance schedules, configuration parameters or even key material. Among others, this data can be used for preparing an attack on the critical infrastructure of the grid. Furthermore, real-time applications require data

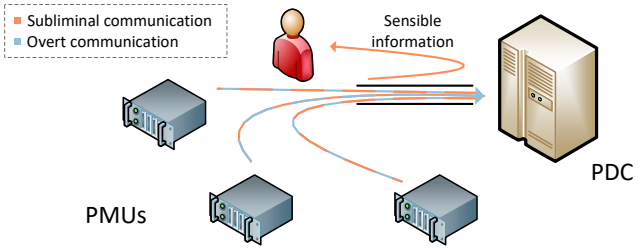

Figure 5: Information leakage using signed phasor measurements.

to be transmitted with a large frequency. Signing each of these packets individually, a vast bit rate results for data exfiltration. Finally, due to the widespread deployment of sensor devices or other smart grid components, an adversary faces a widespread infrastructure for mounting attacks. The homogeneous hardware and configuration of many of these devices allows malware to spread more easily.

Phasor Measurement Units

An example of smart grid applications where signatures can lead to a vast bit rate for data exfiltration is the transmission of measurements by Phasor Measurement Units (PMUs). These devices measure the phasor of electrical current and voltage and transmit the measurements to Phasor Data Concentrators (PDCs), finally supporting control decisions for grid operations (see Fig. 5).
The measurements have to be transmitted in real-time, where they usually are transmitted with a reporting rate of 10 to 50 measurements per second in a 50 Hz grid and 10 to 60 measurements per second in a 60 Hz grid, but higher reporting rates like 120 measurements per second might also be used [71]. Authenticity and integrity are crucial as otherwise sensor data might be altered in a man in the middle attack and wrong control decisions might result. Confidentiality of phasor measurements in the public grid is of little importance and, hence, encryption might be omitted to reduce computational effort. The use of high-speed signatures for signing transmissions with such a high rate of signed packets seems natural. Finally, measurement data might also be multicast, yielding advantages comparable to broadcast time information described in the previous Section 4.2 for an adversary.
Hence, also this scenario yields ideal conditions for the establishment of subliminal channels and the exfiltration of large amounts of data.

4.4 Traditional Use Cases

The above scenarios mainly benefit from the high-speed property of signature schemes, giving rise to a large bit rate for adversarial data transmissions. However, the signature schemes of course can also be used in traditional use cases, where high-speed signing is not vital.

Table 1: Deployment of EdDSA.

	Service	Purpose of EdDSA	Possible abuse
Protocols	Tor[1]	Authenticate hidden services	Deanonymization by leaking identity information
	DNSCrypt[2]	Authenticate DNS servers	Botnet C&C
	DNSSEC [72]	Sign DNS records	Hiding illegal content
	TLS [73, 12] SSH [74, 75] IKE [76, 77]	Authenticate server and/or client	Data exfiltration
Blockchains	Monero[3] Ripple[4] Stellar[5] Tendermint[6] Lisk[7]	Prove coin ownership Authenticate validators	Hiding illegal content Leaking secret keys Deanonymization
Tools	GnuPG[8]	Sign emails	Data exfiltration
	signify[9]	Sign files and software	Hiding illegal content

[1] https://www.torproject.org/
[2] https://dnscrypt.info/
[3] https://getmonero.org/
[4] https://ripple.com/
[5] https://www.stellar.org/
[6] https://tendermint.com/
[7] https://lisk.io/
[8] https://www.gnupg.org/
[9] https://www.openbsd.org/papers/bsdcan-signify.html

In this section we mainly focus on EdDSA as signature scheme, because it is widely deployed already and its deployment is likely to increase further in future due to its very appealing properties. For the class of MQ signatures, on the other hand, no scheme has been confirmed by the cryptographic community yet. However, use cases for such signatures are quite similar to EdDSA, so the results are also relevant for MQ signatures.

EdDSA has been proposed for and is already in use in a large number of applications. Table 1 lists some examples for the present deployment of EdDSA[2] and points out possible intents for exploiting the signatures' subliminal channel. Signatures

[2] A more extensive list can be found at https://ianix.com/pub/ed25519-deployment.html.

that are generated dynamically as in the course of network protocols, could be exploited for malware communication or data exfiltration, static signatures that are created once and then are publicly available on the Internet, might be exploited for hiding and clandestinely transmitting illegal or otherwise unwanted content.
In the following some example use cases for digital signatures will be highlighted in detail.

Security Extensions for Internet Protocols

Nested signatures are used for achieving path validation for the Border Gateway Protocol (BGP). In this case, a subliminal channel provides the possibility for clandestine information exchange between BGP routers. Signatures can also be used for providing security for the DNS. In particular, the DNSSEC extension [72] and DNSCrypt both support EdDSA. As DNS lookups occur frequently for some applications, a large bit rate is possible for adversarial communication.

Network Security Protocols

There are several protocols that allow confidentiality and authenticity for communication over insecure network links such as over the Internet. In such protocols signatures are used for achieving authenticity of the communication partners.
EdDSA is supported in the Secure Shell (SSH) [74, 75] protocol, TLS version 1.3 [12] and has been proposed for TLS version 1.2 [73]. It furthermore has been proposed for Internet Key Exchange (IKE) [76, 77] which is used for the key exchange for the IPsec [78] protocol.
Due to performance reasons, however, signatures in network security protocols are mainly used during the initial connection establishment phase for key agreement. Later, authenticity is achieved by means of symmetric cryptography. Hence, only small amounts of data can be transmitted by exploiting a subliminal channel. The use of high-speed signatures in most cases is not essential but might be desirable to speed up the connection establishment.
Due to the extremely widespread use of TLS, Chapter 7 will discuss in detail how subliminal channels can be exploited and what impact an exploitation can have on security also considering other possible covert and subliminal communication techniques.

Official Documents

As highlighted in [20] a subliminal channel can be used for placing information in digital signatures of passports. The same applies to digital health insurance cards. The issuing instance can embed sensible information without the owner's knowledge, significantly threatening the owner's privacy.

5 Subliminal Channels in EdDSA

The highly attractive properties of the EdDSA signature scheme give rise to its widespread use in current and future deployments. Judging from Chapter 4, however, a subliminal channel in EdDSA signatures can pose a significant threat to information security. We will therefore evaluate in this chapter the possibilities for subliminal communication EdDSA yields and the mitigation strategies to prevent their active exploitation by an adversary.
Taking a first glance at the signing procedure described in Section 2.5, signing is deterministic and therefore seems unlikely to allow the embedment of subliminal information. Taking a closer look, however, we note that for the purpose of embedment of subliminal information we can alter the calculation rule Eq. (5) for the nonce r. In fact, for the reasons pointed out in Section 2.5 use of a cryptographic hash function as proposed in [8] and standardized in [32] for deriving the nonce value r is reasonable for obtaining a secure signature scheme. However, as the calculation of the nonce involves the private key, a verifier has no means to test if the method has indeed been used. Using a different value for the nonce does not harm the successful verifiability of the produced signature in any way. Therefore, an arbitrary (random) value could be used instead for r. As a consequence, both a broadband subliminal channel with a subliminal bandwidth of almost half the signature's bits, and a narrowband subliminal channel with a subliminal bandwidth of just a few bits per signature, can be established exploiting EdDSA signatures.

5.1 The Broadband Channel

Like in other signature schemes that are based on the discrete logarithm problem the (random) nonce r can be calculated with little effort from a signed message (M, S, R) if the signing key a is known as

$$r = S - H(R, A, M)a \bmod L, \tag{14}$$

where A and L can be considered known. Hence, the value of r can be used as a subliminal channel by encoding covert data into it on the sender side. This data can then be recovered using Eq. (14) by anyone who holds the signing key a and is able to intercept the message and its signature. Since information can only be encoded in the residue class modulo L, the subliminal channel has a theoretical subliminal bandwidth of $\log_2 L$ bits per signature. For Ed25519 this corresponds to a subliminal bandwidth of $\log_2 L \approx 252$ bit per signature. For Ed448 this corresponds to a subliminal bandwidth of $\log_2 L \approx 447$ bit per signature.
The setting for this broadband subliminal channel is depicted in Fig. 6. The sender cooperates with a subliminal receiver, who can be co-located with the receiver or

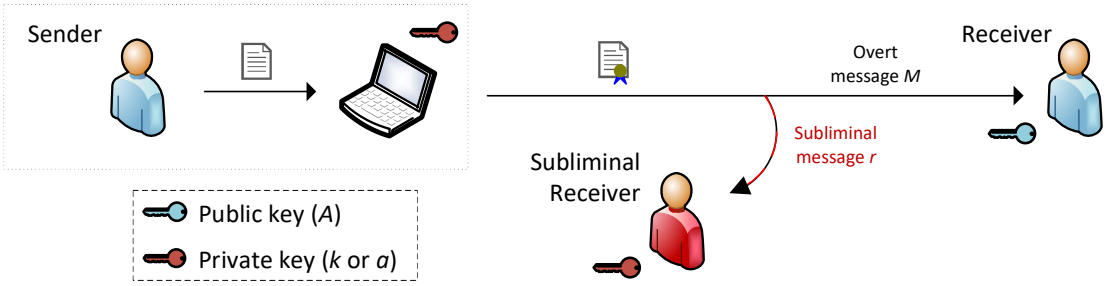

Figure 6: A broadband subliminal channel in EdDSA.

reside somewhere on the network path. It is required that the receiver of the subliminal information also knows the signing key a. For this, we can distinguish two cases: (1) the legitimate sender wants to transmit subliminal information intentionally and (2) the legitimate sender has been compromised and the subliminal message is inserted by malware that has access to the signing process.
For case (1), the sender directly shares the signing key with the subliminal receiver before the subliminal communication starts. By knowing the signing key, the subliminal receiver would also be capable of forging arbitrary signatures on behalf of the sender. Nevertheless, for subliminal communication scenarios we assume that the sender of the subliminal information and the subliminal receiver collude and that it is therefore reasonable to assume that they share the private key k (to derive the signing key) or the signing key a upfront. For case (2), the adversary needs to clandestinely leak the signing key to the receiver. For the attack scenarios described in Chapter 4 the key could be leaked by using the narrowband channel described below.
A further requirement is that both the signature and the signed message are known to the subliminal receiver. Usability of the subliminal channel is thus ruled out if communication is encrypted after the signature has been attached and the decryption key is unknown to the subliminal receiver. On the other hand, when using the Encrypt-then-Sign method, exploitation of the subliminal channel is easy. In this case the signed message is the ciphertext of the message and the signature itself is unencrypted. Hence, the nonce value can be recovered without the need to decrypt the ciphertext.

5.2 A Narrowband Channel

Besides the broadband subliminal channel described above, it is also possible to use the signature as a narrowband subliminal channel. The following procedure constitutes a very general approach for establishing a subliminal channel exploiting signature schemes that either explicitly consume randomness for signature genera-

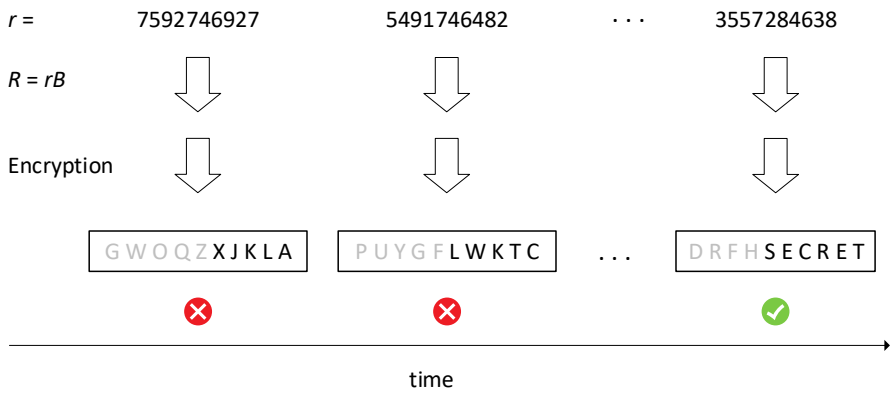

Figure 7: A narrowband subliminal channel: Embedment of the string 'secret' in EdDSA signatures.

tion (randomized signature schemes) or implicitly allow multiple valid signatures for the same message.

The signer tries to make the encoded representation of the signature show a specific bit pattern that corresponds to the subliminal information. In our case, the signer could, for example, aim to make the last byte of the encoded representation of R equal to the subliminal information. Since computing discrete logarithms is infeasible, he cannot directly find a nonce value, for which R has the desired properties. However, trying a large-enough number of different, randomly picked values for r, a suitable value can eventually be found by chance.

If B_s denotes the intended subliminal bandwidth in bits per signature, the signer has a probability of 2^{-B_s} that for a particular choice of the nonce r, R shows the desired properties. The process of searching for a suitable value of r thus represents a sequence of stochastically independent Bernoulli trials with success probability 2^{-B_s}. Hence, the number of trials needed for finding a suitable value obeys a geometric distribution with mean 2^{B_s}. Due to the exponential growth of the computational effort with the subliminal bandwidth B_s, it is thus not possible to use a significant portion of the signature's bits for the subliminal information, which explains the classification as narrowband channel.

The benefits of this method as compared to the broadband channel are that the subliminal receiver does neither have to know the signing key a, nor the signed message. Note, furthermore, that this narrowband channel is exploitable even if communication is encrypted under the mild assumption that the subliminal receiver can locate the ciphertext of the signature in the encrypted data. The signer then tries to make the ciphertext of the encoded representation of R show the intended bit pattern. To reach this goal, he proceeds as described above. Fig. 7 illustrates this process.

5.3 Combining Broadband and Narrowband Channel

The narrowband and broadband subliminal channels described above can also be combined to achieve a large subliminal bandwidth to subliminal receivers for which the requirements of the broadband channel are met, and a small subliminal bandwidth to other subliminal receivers. In this case the signer does not use all of the available bits of the broadband subliminal channel but leaves some of them free to be able to vary r as described in the previous Section 5.2. It has to be noted, however, that due to the stochastic nature of this process, the subliminal bandwidths of narrowband and broadband channel sum up to little less than what is available when using the broadband channel alone.
To find the usable bandwidth, let $B_{s,b}, B_{s,n} \in \mathbb{N}$ denote the broadband and narrowband subliminal bandwidth in bits per signature, respectively. Furthermore, let $\delta \in \mathbb{N}$ denote the difference of bandwidths that are available from using both channels together and only the broadband channel, i.e. $252 = B_{s,b} + B_{s,n} + \delta$ in the case of Ed25519. The signer has a probability of $2^{-B_{s,n}}$ of finding a signature with the intended subliminal information when trying one value of r. As he can try $2^{252-B_{s,b}} = 2^{B_{s,n}+\delta}$ different values of r, the signer thus has a probability of $P_f = (1 - 2^{-B_{s,n}})^{2^{B_{s,n}+\delta}}$ for signing to fail with the intended subliminal information. With the approximation $(1+\frac{x}{n})^n \approx \exp(x)$ which holds for large n, we obtain $P_f \approx \exp(-2^{\delta})$ for large $2^{B_{s,n}}$.
For example, for $P_f = 10^{-6}$ we obtain $\delta = 3.79 \approx 4$ bits, so the signer can use about 4 bit less when using both broadband and narrowband channel compared to when using the broadband channel alone.

5.4 Mitigating Subliminal Communication

Chapter 4 has shown that the subliminal channels described above constitute a severe security problem in certain situations. It therefore is an important question if it is possible to detect the abuse of signatures for this purpose or even use the signatures in a setting that renders exploitation of the subliminal channels impossible.

5.4.1 Ensuring Subliminal-Free Signatures

Some approaches that aim to prevent subliminal communication while retaining compatibility with the usual signature verification algorithm have been proposed for other signature schemes. In this section, we investigate whether these methods for other signature schemes can also be applied to EdDSA signatures to protect them against the establishment of subliminal channels.
To elaborate on the mitigation possibilities we refer to the warden scenario Fig. 8

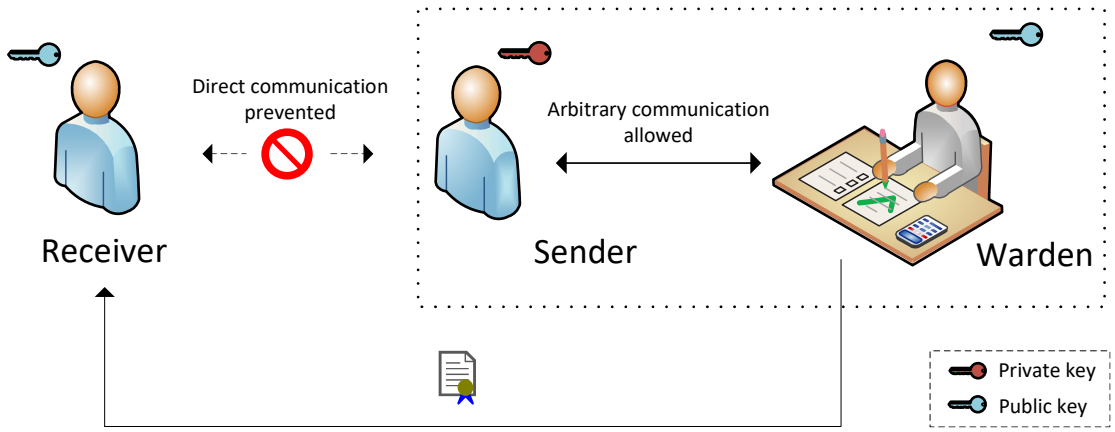

Figure 8: Warden scenario to achieve subliminal-freeness.

that was introduced by Simmons [6]. The sender may only communicate with a warden to transmit a message to a receiver. The warden is located between sender and receiver and monitors the communication and filters inappropriate messages. The warden thus has to forward the messages but will do so only if he approves with the messages' contents. Obviously, he insists on the communication being unencrypted as he otherwise would not be able to investigate the contents. Finally, if the sender uses signatures to authenticate his messages, the warden will only forward the messages if he can be sure that no subliminal information is hidden in the signatures. As described in Chapter 3, Simmons thought of two prisoners trying to coordinate an escape plan as example for this setting.
Since the subliminal channel in EdDSA is employed in the random number used to generate the signature, the verification process in general would not show any irregularities when exploiting the subliminal channel. Hence, the question arises, how the signer can prove subliminal-freeness to the warden without disclosing information about his signing key to the warden.
Table 2 summarizes the three distinct approaches to mitigate subliminal communication described below with respect to their advantages and drawbacks.

Preannounced Nonce Points

In some situations the following simple mitigation strategy might be sufficient. Instead of computing the nonce values during signing as described above, the warden requires the signer to generate and announce a list of nonce points R before the signer can possibly have the information that he would transmit using the subliminal channel during signing. During signature generation, the signer then has to use the values in the same order as they appear in the list. With the nonce

Table 2: Approaches to ensure subliminal-freeness in EdDSA.

Approach	**Applicability**	**Advantages and drawbacks**
Preannounced nonce points	Elgamal and Schnorr signatures	+ Simple + Low computational requirements - Limited number of transmitted messages - Subliminal information embeddable during list computation - Storage requirements for warden
Interactive method [55]	Schnorr signatures	+ Small bandwidth requirements - Participation of warden required - Several messages need to be exchanged - Need for bidirectional communication
Non-interactive method [20]	Proving pseudo-randomness of a curve point	+ Simple communication pattern + Feasible for offline scenarios - Huge prove size - Significant computational requirements

r being fixed a priori, the signature indeed becomes unique in the sense that for a given message just one possible value for S remains.

A variant of this approach would be to index the list by a number derived deterministically from the message. However, this approach causes an increasing number of messages to be unsignable as the corresponding signature would require a nonce that was already used earlier and would therefore make the signature reveal the signing key.

This mitigation strategy has several disadvantages. First of all, due to the limited number of usable R-values, the number of distinct messages that can be signed is equally limited. When using the indexed variant, this is even more serious, as the number of unsignable messages would become significantly large once a certain amount of values has been used. Secondly, the warden needs to store the list of R-values, which might lead to significant storage requirements. For each message the signer should be able to sign, 32 B of storage capacity are required. The method also introduces a state into signing, which might cause security issues. The most important drawback is the fact that the subliminal channel is just shifted to an earlier time instant – subliminal information can still be embedded during generation of R-values.

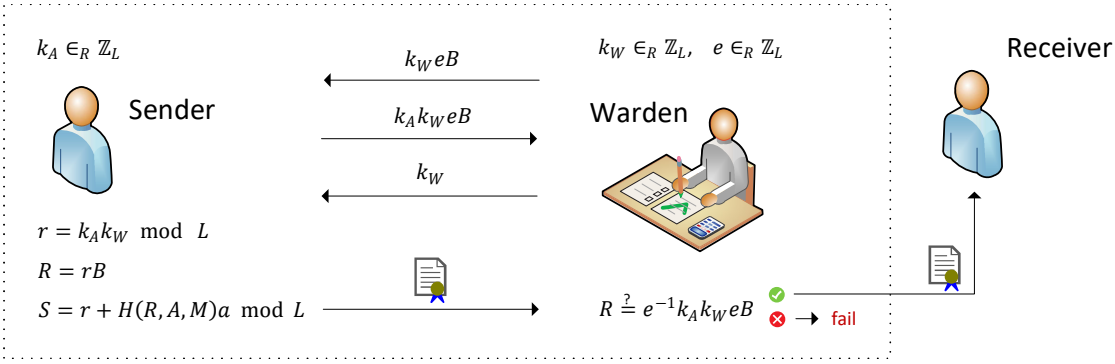

Figure 9: An interactive method for achieving subliminal-freeness.

An Interactive Approach

Zhang et al. proposed an interactive scheme for subliminal-free signing for Schnorr signatures, in which the warden actively contributes to signature generation [55]. To prevent a subliminal channel, a total of six messages have to be exchanged between signer and warden for each signature. The scheme is shown to be secure against existential forgery if the computational Diffie-Hellman assumption holds. Furthermore, embedding subliminal information in the signature is shown to be as hard as computing discrete logarithms for the signer.
Since the EdDSA scheme is derived from Schnorr signatures, this method can be adopted to be used also as mitigation strategy for EdDSA. Nevertheless, the major drawbacks of the scheme are the large number of messages to be exchanged and the computational effort required both at the signer and at the warden for generating signatures. These drawbacks conflict substantially with the requirements in typical scenarios where EdDSA can be employed: low computational effort, fast signing and verification, and small overhead. Furthermore in some situations only unidirectional communication between signer and warden is possible.
Fig. 9 shows a simplified version of the protocol adopted to EdDSA[3].

A Non-Interactive Approach

Bohli, Vasco, and Steinwandt described a method for making ECDSA subliminal-free with proof that does not require active participation of the warden [20]. Instead, the nonce r required by the signature scheme is generated deterministically from the message and a proof is given to the warden that the value has indeed been derived correctly. This proof does not provide any means for the warden for deriving the nonce, and, hence, computing the signing key.

[3]The original publication additionally modifies key generation and is able to ensure that the signer has to cooperate with the warden for signature generation.

For generating a nonce value during signing, first a hash value $\boldsymbol{h} \in \{0,1\}^m$ is computed from the message. The nonce is then derived using Naor and Reingold's pseudo random function [79] as

$$r(\boldsymbol{h}) = g^{a_{m+1} \prod_{1 \leq i \leq m, h_i = 1} a_i} \bmod p \bmod L.$$

In this equation, p is a prime number and g is the generator of a cyclic group of prime order q, where p, q and g are public. The vector $\boldsymbol{a} \in \mathbb{Z}_q^{m+1}$ is an additional secret for signature generation which must not be confused with the private key k or signing key a needed for signing. These variables are chosen once during key generation.
The signer computes commitments for $\boldsymbol{a}$ and shows them to the warden during key generation. During signing the signer can then compute zero-knowledge proofs that show that the nonce has in fact been computed in the correct way. Subliminal-freeness is proven only for the signature itself. The proof could thus contain a subliminal channel and must be stripped off by the warden after verification.
This provably subliminal-free signature scheme is formulated for ECDSA. Since it solves the general problem of proving that the nonce was computed deterministically from the message without disclosing the nonce value itself, it can equally be applied to EdDSA. Compared to the previous interactive approach, the method has the advantage of a simple unidirectional communication pattern between signer and warden which even qualifies for certain offline scenarios. In fact, Bohli, Vasco, and Steinwandt proposed to use the scheme for passports where it should be possible for the passport's holder to make sure that the issuing party has not embedded information in the signature. On the downside, since one proof (for a security level of 128 bit) takes several megabytes, the bandwidth requirements between signer and warden are high. In particular, in the context of network protocols these bandwidth requirements usually are substantially too high.
Alternatively, if the requirements for the warden are relaxed to being able to prove the existence of a subliminal channel when examining a random sample of signed messages, the scheme is suitable for scenarios where it is not feasible to place a warden in a man-in-the-middle position: The signer can be obliged to offer the proofs for the generated signatures on a protected interface. A signature that has been intercepted unnoticeably can then be tested for having been generated correctly. In this scenario the signer does not have to compute and store the proofs for all signatures, which would cause very high storage requirements. Instead, when requesting a proof the warden can provide the signer with the message in question, which suffices to reproduce the signature and generate the corresponding proof. In this case the signer must make sure that the warden already has a valid signature for the message, as he would otherwise sign arbitrary messages on behalf of the warden.

5.4.2 Detecting Subliminal Communication

Although EdDSA cannot be made subliminal-free (without introducing serious disadvantages), there may still exist ways to detect the subliminal data transfer or at least to check whether subliminal data exchange can be suspected. In this section, we highlight situations in which the subliminal channel can lead to observable suspicious patterns that help to detect the transmission of subliminal information.

Identical Messages

Due to the deterministic calculation of the nonce r (see Eq. (5)), a specific message produces the same signature independent of how often the message is transmitted. If the same message is transmitted twice and r has not been derived from the messages but carries two distinct subliminal messages, this abuse can be detected by the fact that the signatures differ while the messages are identical. A warden who monitors the communication can notice that two signatures are different even though they were generated for identical messages using the same key. From this observation, the warden can then deduce that subliminal information has been transferred.
To prevent this detection method, a subliminal sender would need to check if an identical message has been sent before. If this is the case, he would use the same nonce value to not raise suspicion. The subliminal receiver can just discard any subliminal information received in duplicated messages. Obviously, this method increases storage requirements for both, subliminal sender and subliminal receiver, significantly.

Predictable Nonce Values

As described in Section 2.5, it is of utmost importance for the security of the signature system to sustain unpredictability of the nonce value r. However, when directly encoding the unencrypted subliminal information into the nonce value, r might regularly assume predictable values like, for example, small values or even become equal to zero, depending on the subliminal information that is being transmitted. Detection of such values can, therefore, not only lead to detection of the subliminal channel, but also allow an eavesdropper to recover the signing key a.
The subliminal sender can mitigate this problem, though, by encrypting the subliminal information. In fact, encryption is often performed for covert and subliminal channels to prevent others from being able to read the transferred information in case the channel is detected. It is noteworthy, however, that unlike signature schemes that consume true randomness, encryption in this case cannot prevent detection of the subliminal channel in case that someone knows the private key k

(e.g., in a special administratory position). In this case the private key k together with the message can be used to verify if r has been computed conforming to Eq. (5).

Repeating Nonce Values

As explained in Section 2.5, the signing key can be recovered from signatures for distinct messages which are using the same nonce value r. The subliminal sender therefore has to ensure that this case does not occur, if r is used for subliminal information instead of deriving it from the message. Depending on the type of information that is to be transmitted covertly, repeating values of r might occur especially if naive encryption methods are used. To counter this problem, a block cipher with a suitable operational mode like Output Feedback (OFB) can be used for the covert information to significantly reduce the probability of recurring random values [15]. The initialization vector for encryption can be derived from the overt message in a manner similar to Eq. (5).

6 Subliminal Channels in MQ Signatures

MQ signatures constitute a novel approach for constructing signature schemes compared to traditional signature schemes like RSA or (EC)DSA. As MQ signatures are based on the problem of solving systems of polynomial equations in finite fields, the resulting signature schemes are likely to survive progress in quantum computing. Furthermore, the signatures are very fast in verification and sometimes also in generation. For these reasons, MQ-based signature and encryption algorithms are an active research area.
In a post-quantum era a secure signature scheme that additionally yields the subliminal-freeness property might be of particular importance:

- As described in previous chapters a subliminal channel allows an attacker to transmit sensitive information unnoticeably. This problem is especially severe if the scheme is performant enough to allow a large number of signatures in little time and if the subliminal channel is broadband, i.e. the hidden data makes up a significant portion of the signature's bits.

- Subliminal-free signature schemes are closely related to the concept of VUFs and VRFs [23, 24]. All constructions of such functions that were found up to date are based on either the problem of factoring large integers [22] or the discrete logarithm problem [80, 81, 23] and thus would be broken as soon as appropriate progress in quantum computing is made.

Therefore, we here investigate the existence of subliminal channels in MQ signature schemes. For this purpose we first analyse where randomness is used throughout the signing process and, thereupon, how information can be encoded in and recovered from this randomness. Finally, the considerations will be extended to some practically proposed signature schemes.

6.1 Randomness in MQ Signatures

The basic operation principle of MQ signature schemes depicted in Section 2.6 creates the impression that these schemes deterministically map a given message to a signature, leaving no space for subliminal information. However, on the contrary, almost all MQ signature schemes devised so far explicitly use randomness throughout the signing process. Among the schemes that are currently considered secure, all use randomness.
Reasons for introducing randomness in the signing process can be classified into two groups. The first group of methods explicitly includes random data in the signature to achieve its security. The second group does not directly introduce randomness but reduces the probability of the central mapping to be invertible for

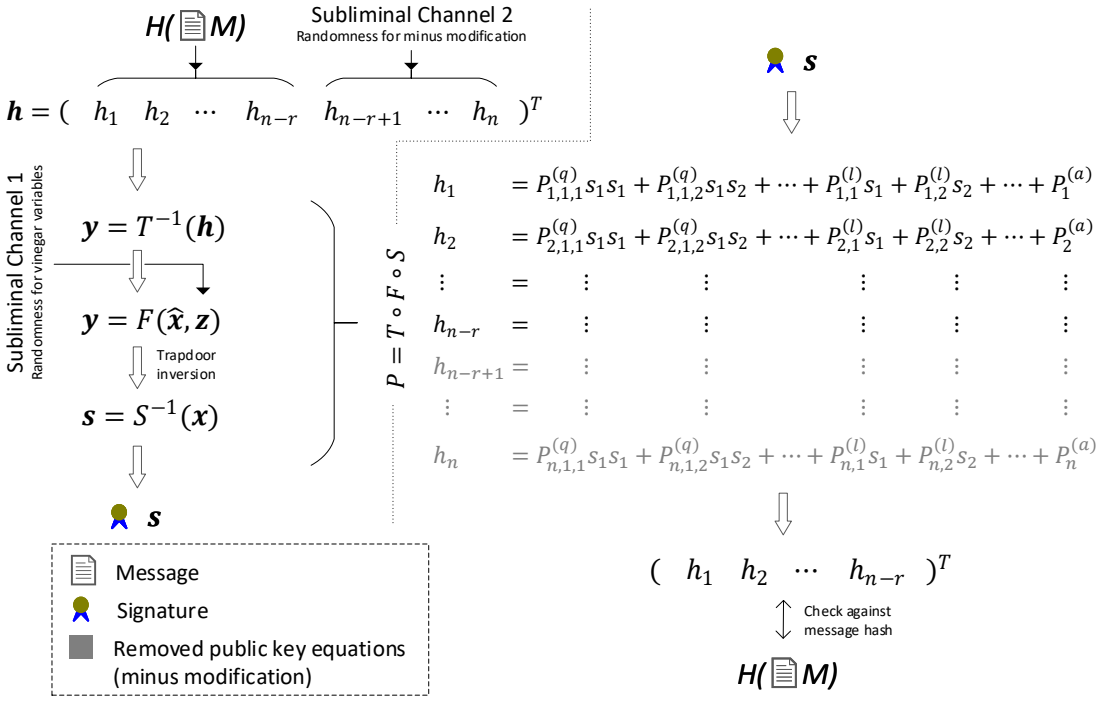

Figure 10: Subliminal channels exploiting the vinegar variables and minus modifications: Signature generation (left) and verification (right).

a message. Hence, randomness has to be included in the signature to make sure that the signer is able to find a suitable signature.
Table 3 summarizes the methods for constructing MQ schemes with respect to this classification. It shows that most trapdoors and modifications either explicitly require randomness during signing or lead to a non-surjective mapping, requiring to include randomness to provide signatures for arbitrary messages. Only the trapdoors listed as bijective would thus be suitable for constructing a subliminal-free signature scheme.
Fig. 10 shows the use of randomness in the signing process when using the minus modification and vinegar variables, allowing exploitation as subliminal channels as described below.

6.1.1 Explicit Randomness

The group of methods that explicitly introduce randomness consists of the UOV trapdoor, the minus modification and the vinegar variables modification. When using schemes with vinegar variables $\boldsymbol{z}$, the signer chooses random values for $\boldsymbol{z}$ and tries to invert the central mapping F with this choice. Exploiting $\boldsymbol{z}$ to encode

Table 3: Randomness in MQ signatures.

	Exp. Randomness	**Non-Surjective**	**Bijective**
Trapdoors	UOV	HFE General STS	C^* MQQ Regular STS
Modifications	Minus modification Vinegar variables	Projection Int. perturbation Plus modification	

information, the vinegar variables can be used as subliminal channel. However, most constructions require multiple runs of the inversion of F with different choices of vinegar variables to be able to find one signature with large-enough probability. Hence, few of the vinegar variables have to remain free to be able to vary them to find a valid signature.

6.1.2 Loss of Surjectivity

There are many methods for constructing MQ signature schemes that cause the trapdoor to not be surjective. This means that, without any further measures, there would be messages for which no signature exists. In particular, this group includes the HFE trapdoor and general STS, the projection modification, the internal perturbation modification and the plus modification. As described in Section 2.6.2 the HFE trapdoor would require a permutation polynomial for $\tilde{F}(\tilde{x})$ to constitute a bijective mapping.

To guarantee the existence of a signature with large-enough probability for every message, randomness has to be included during signature generation. Usually this is accomplished by using either the minus modification or the vinegar variables modification.

It is interesting to note that the modifications that lead to the loss of surjectivity, can not only cause the existence of a subliminal channel but may also *reduce* the subliminal bandwidth that results from, for example, other modifications like the minus or vinegar variables modification. This is because the probability of finding a signature with a particular choice of random data is reduced, requiring the signer to be able to vary more variables to find a signature with large-enough probability. Yet, use of these modifications for this sole purpose is usually not justified, as they come with significant drawbacks in terms of signing speed.

6.1.3 Subliminal-Free Trapdoors

Comparing the methods mentioned in the prior two sections with Section 2.6.2 we see that only the C^*, STS and MQQ trapdoors in unmodified form and not a single modification are suitable to construct subliminal-free signatures. The subliminal-freeness of these methods is due to the fact that these trapdoors are specifically constructed to be bijective. Nevertheless, all signature schemes using these unmodified trapdoors suitable for subliminal-free signatures, have been broken.
Furthermore, for the particular case of STS we have to note that bijectivity only applies to regular STS, which is constructed to have a bijective mapping in each layer. If, for example, a layer of a general STS construction yields more unknowns than equations, multiple signatures can be found for a message and bijectivity is lost.

6.2 Subliminal Channel 1: Recovering Vinegar Variables

Both the UOV trapdoor and the vinegar variables modification have in common that, for the purpose of signature generation, the signer first picks v vinegar variables at random and afterwards inverts the central mapping $F(\hat{\boldsymbol{x}}, \boldsymbol{z}) = \boldsymbol{y}$ using this particular choice of vinegar variables. Knowing $\hat{\boldsymbol{x}}$ and $\boldsymbol{z}$ the signer can then compute the signature by using the inverse of the affine mapping S. Writing S as $S(\boldsymbol{s}) = \boldsymbol{S}^{(l)}\boldsymbol{s} + \boldsymbol{S}^{(a)}$ with $\boldsymbol{S}^{(l)}$ and $\boldsymbol{S}^{(a)}$ the linear and affine parts of S, respectively, the signer computes $\boldsymbol{s} = {\boldsymbol{S}^{(l)}}^{-1}(\boldsymbol{x} - \boldsymbol{S}^{(a)})$.
Evidently, for everyone knowing S it is easy to recover the vinegar and oil variables as

$$\boldsymbol{x} = \begin{pmatrix} \hat{\boldsymbol{x}} \\ \boldsymbol{z} \end{pmatrix} = \boldsymbol{S}^{(l)}\boldsymbol{s} + \boldsymbol{S}^{(a)}, \tag{15}$$

or, partitioning $\boldsymbol{S}^{(l)}$ as $\boldsymbol{S}^{(l)} = \begin{pmatrix} \boldsymbol{S}_x \\ \boldsymbol{S}_z \end{pmatrix}$ with $\boldsymbol{S}_x \in \mathbb{F}_q^{n\times m}$ and $\boldsymbol{S}_z \in \mathbb{F}_q^{v\times m}$, the vinegar variables can be computed as $\boldsymbol{z} = \boldsymbol{S}_z\boldsymbol{s} + \boldsymbol{S}^{(a)}$. Hence, it suffices if the signer passes the subliminal receiver the affine mapping consisting of $\boldsymbol{S}_z$ and $\boldsymbol{S}^{(a)}$ upfront to allow him to recover the vinegar variables and exploit them as a very efficient subliminal channel.
It is not surprising, however, that by sharing $\boldsymbol{S}_z$ and $\boldsymbol{S}^{(a)}$ the signature scheme's security is significantly reduced considering attacks performed by the subliminal receiver. To see that, we note that by setting $\boldsymbol{z}$ to some arbitrary value the subliminal receiver is able to obtain a linear equation system in s_i from $\boldsymbol{z} = \boldsymbol{S}_z\boldsymbol{s} + \boldsymbol{S}^{(a)}$. He can, hence, express v of the variables s_i and substitute them into the public key equations. The resulting altered public key is the public key of the original key

with fixed vinegar variables. However, by fixing the vinegar variables, the HFEv signature system Eq. (12) reverts to the original HFE shape Eq. (10). The new public key therefore corresponds to unmodified HFE and can be attacked with key recovery attacks that have been found for HFE. For the UOV trapdoor the central mapping Eq. (7) even turns into an affine mapping, making it trivial for the subliminal receiver to forge signatures for arbitrary messages.

Hence, to retain a certain level of security against attacks performed by the subliminal receiver, the signer must not pass on all lines of $\boldsymbol{S}_z$ and, therefore, cannot use all vinegar variables to encode subliminal information. If there is not sufficient trust among the subliminal sender and the subliminal receiver, the sender thus has to find a trade-off between achievable subliminal bandwidth and security against attacks performed by the subliminal receiver.

6.3 Subliminal Channel 2: Using the Minus Modification

The minus modification is a popular method for improving an MQ signature scheme's security.

The public key of an unmodified MQ signature scheme consists of n multivariate equations $h_i = P_i(\boldsymbol{s})$, where $i = 1, \ldots, n$. When deploying the minus modification, we remove r of the equations from the public key. The corresponding portion of the message hash $\boldsymbol{h}$ can be filled with arbitrary data by the signer. In fact, the signer even has to fill it with random data as otherwise an adversary is able to reconstruct the removed public key equations as soon as he has observed enough signed messages [10].

The public key of the modified scheme now consists of the equations $h_i = P_i(\boldsymbol{s})$ with $i = 1, \ldots, (n - r)$ and the removed equations read $\rho_i = P_{n-r+i}(\boldsymbol{s})$, where $i = 1, \ldots, r$ and $\boldsymbol{\rho} \in \mathbb{F}_q^r$ denotes the random data. Hence, using the unmodified public key, $\boldsymbol{\rho}$ can easily be recovered from the signature, making it possible to use $\boldsymbol{\rho}$ as subliminal channel. To use this subliminal channel, the signer has to share the removed equations $P_{n-r+1}(\boldsymbol{s}), \ldots, P_n(\boldsymbol{s})$ with the subliminal receiver upfront.

Obviously, by receiving the removed equations the modified scheme reverts to the unmodified scheme for the subliminal receiver. Hence, for the subliminal receiver attacks become possible that were meant to be prevented by the use of the minus modification. The signer can retain a certain level of security, however, by using only part of the removed equations as subliminal channel and keeping the remaining ones secret. Hence, also in this case a trade-off between subliminal bandwidth and security against attacks performed by the subliminal receiver can be achieved.

Table 4: Subliminal bandwidths of proposed MQ signature schemes.

Scheme	Trapdoor	Broken	Signature length	Subliminal bandwidth
QUARTZ [45]	HFEv-	no	128 bit	~12 bit/sig. (9%)
Gui-127 [59]	HFEv-	no	163 bit	~24 bit/sig. (15%)
Rainbow [60]	UOV-, STS	yes	264 bit	~46 bit/sig. (17%)
SFlash [82]	C^*-	yes	469 bit	77 bit/sig. (16%)
PFlash(GF16,94,30,1) [9]	pC^*-	no	372 bit	~108 bit/sig. (29%)
MQQ-SIG (256 bit) [10]	MQQ-	yes	256 bit	128 bit/sig. (50%)

6.4 Narrowband Channels

As described in Section 5.2, a very general approach for establishing a subliminal channel for randomized signature schemes is varying the random value until the signature string shows a specific bit pattern that corresponds to the intended subliminal information. As randomness is used throughout the signing process for (almost) all MQ signature schemes, this approach can also be used for MQ signatures. The requirement is, however, that there are enough random bits to ensure that a signature with the intended subliminal information can be found with large probability.
The advantage of this method compared to the two subliminal channels above is that the signer does not have to disclose any of his secret information. Furthermore, as described in Section 5.2 the subliminal channel can also be used if the signature is encrypted with a key that is unknown to the subliminal receiver.
Similar to Section 5.3 it is also possible to combine this narrowband channel with the two above methods.

6.5 Existing MQ Signature Schemes

To get an understanding of how much data can be exfiltrated using subliminal channels in MQ signature schemes, we analyse algorithms for which an implementation exists or at least a practical set of parameters has been proposed. It shall be stressed that we aim to analyse techniques that are used for constructing MQ signature schemes rather than concrete algorithms. Hence, even though many of the schemes described below have been broken, the results have a certain relevance for signature schemes that are going to be developed and are likely to be constructed in a similar way. Table 4 shows the results.

In the following we do not distinguish between truly random data and pseudo-random data that was deterministically derived from the message using a cryptographic hash function and a secret key (see e.g. the calculation of r in Chapter 2.5). QUARTZ [45] uses the HFE trapdoor in conjunction with the minus modification and vinegar variables modification. To avoid birthday attacks, QUARTZ uses 4 rounds and uses the trapdoor in each round. The corresponding portion of the message hash and the vinegar variables are filled with random data. Hence, subliminal information can be embedded and recovered as described in Section 2.6.3. This way 7 bits of subliminal information (4 for vinegar variables and 3 due to the minus modification) could be embedded each round. It has to be noted, however, that, as described in Section 2.6.2 the trapdoor cannot be inverted for each possible choice of random data. The signer therefore has to be able to vary it to a certain degree to perform multiple attempts to eventually find a signature, varying the random data each time. In fact, the probability for the trapdoor to not be invertible in one round for a particular choice of random data is $1/e$ [45]. Reserving B_r bits per round for subliminal information we can try 2^{7-B_r} different values for the random data in each round. Having 4 rounds we then have a probability of

$$P_f = 1 - \left(1 - \left(\frac{1}{e}\right)^{2^{7-B_r}}\right)^4 \tag{16}$$

for signing to fail. For a (arbitrarily chosen) value of $P_f = 10^{-6}$ we obtain $B_r = 3.07 \approx 3$ bit per round and a total subliminal bandwidth of 12 bit per signature.
The Gui signature scheme [59] is an improved version of QUARTZ and has a very similar design. The authors propose three different parametrizations of the scheme. Gui-127, which achieves a security level of 123 bit, operates in 4 rounds like QUARTZ but uses in each round 6 vinegar variables and removes 4 equations in $\mathbb{F}_2$, hence achieving 10 bit of subliminal information per round when neglecting the need to invert the trapdoor multiple times. Using the same calculation as above, we obtain $B_r = 6.07 \approx 6$ bit per round and a total subliminal bandwidth of 24 bit per signature.
Rainbow [60] uses the UOV trapdoor in a layered approach. We can consider it as a combination of the UOV and STS trapdoors. In a total of 4 layers the scheme uses 6, 12, 17 and 22 vinegar variables, respectively. For Rainbow, the vinegar and oil variables of layer i are the vinegar variables of layer $i + 1$, which is why randomness is only used for the initial 6 vinegar variables. Having the appropriate equations, these variables can be fully reconstructed from a signature. Hence, as the scheme operates in $\mathbb{F}_{256}$, a maximum subliminal bandwidth of 48 bit per signature results. Also in this case the invertibility of the trapdoor is not guaranteed (see Section 2.6.2) narrowing the exploitable bandwidth. The probability for the matrix forming a UOV trapdoor to be invertible is $\prod_{n=1}^{N-1}(1 - q^{-n})$, where q denotes the

order of the Galois field and N the dimension of the matrix [41]. In our case we therefore obtain a probability of

$$P_s = \prod_{n=1}^{11}\left(1-\frac{1}{256^n}\right)\prod_{n=1}^{16}\left(1-\frac{1}{256^n}\right)\prod_{n=1}^{21}\left(1-\frac{1}{256^n}\right)\prod_{n=10}^{32}\left(1-\frac{1}{256^n}\right) \quad (17)$$

for the signature generation to succeed for a particular choice of vinegar variables. Reserving B_s bits for subliminal information we thus obtain a probability of $P_f = (1-P_s)^{2^{48-B_s}}$ for no signature to exist. Again targeting a probability of $P_f = 10^{-6}$ we obtain

$$B_s = 48 - \log_2 \frac{-6}{\log_{10}(1-P_s)} \approx 46 \text{ bit/signature} \quad (18)$$

as subliminal bandwidth.

Furthermore, we analysed two schemes that use the minus modification and use trapdoors with guaranteed invertibility, independent of the choice of the random variables. Hence, all random bits can be used for subliminal information. One is the SFlash scheme [82]. The scheme was recommended by the NESSIE project for low cost smart cards (and broken shortly afterwards), where efficient operation is of high importance, but the size of the public key is not an issue [83]. The scheme uses the C^* trapdoor and removes 11 equations from the public key. As it operates in $\mathbb{F}_{128}$, we obtain a subliminal bandwidth of 77 bit per signature.

The PFlash signature scheme [9] extends SFlash by the use of projection, fixing one of the signature variables. The probability of finding a signature with a particular choice of random data thus drops to q^{-1}, where for PFlash $q = 16$. Again setting the probability of not being able to find a signature for a particular choice of subliminal information to $P_f = (1-q^{-1})^{2^{4r-B_s}} = 10^{-6}$, we obtain for an exemplary parametrization with $n = 94$ and $r = 30$ a subliminal bandwidth of 108 bit per signature.

Finally, the MQQ trapdoor which was proposed in [46, 47] was enhanced by the minus modification after being broken. Hence, the authors proposed to remove half of the public key equations, resulting in MQQ-SIG [10]. If we, for example, use the signature scheme to produce a signature with a length of 256 bit an exploitable subliminal bandwidth of 128 bit per signature results.

6.6 Properties of MQ Subliminal Channels

Among all methods for constructing MQ signature schemes only very few showed to be subliminal-free. In particular, the approaches that are considered most secure today allow a large subliminal bandwidth. Furthermore, the embedment and recovery processes work remarkably easy and efficiently in all algorithms that were considered.

It is noteworthy that subliminal channels for MQ signatures might have particularly attractive properties for an attacker: In contrast to broadband subliminal channels of DSA-like signature schemes the possibility to decode the subliminal information does not directly coincide with the possibility of signing arbitrary messages. If the total available bandwidth is used, however, attacks become possible for the subliminal receiver allowing signature forging. On the other hand, by using just part of the bandwidth, a trade-off can be achieved between subliminal bandwidth and security against attacks performed by the subliminal receiver. Furthermore, by passing on different parts of the set of hidden equations, it is possible to transmit different subliminal information to multiple receivers, who thereby are unable to decode the information that is not intended for them.
Similar to the broadband subliminal channel described for EdDSA, encryption of the signature might pose a problem when trying to exploit the subliminal channel. In particular, when using the Sign-then-Encrypt method, the signature string is unavailable if the subliminal receiver is not able to perform the decryption, which prevents the recovery techniques for random data depicted in Sections 6.2 and 6.3. When using the Encrypt-then-Sign method, however, exploitation of the subliminal channel is possible, as in this case the signature itself is unencrypted. Unlike EdDSA, furthermore, exploitation of the subliminal channel is also possible when using the Encrypt-and-Sign method, as the signed message is not required for recovery of the subliminal information.
Considering these results it appears to be an important task to find subliminal-free MQ signature schemes. However, even if a subliminal-free MQ signature scheme was found a further problem has to be considered: The subliminal-freeness can only be guaranteed if the private and public key are known to have been generated in the proper way. For a subliminal-free scheme the public key consists of a bijective mapping $P\colon \mathbb{F}_q^n \to \mathbb{F}_q^n$. Bijectivity ensures that a given message only has a single valid signature, rendering any subliminal communication impossible. However, without knowing how P was constructed, it cannot be verified if it indeed is a bijection and, hence, if signing is indeed subliminal-free. This is similar to the RSA signature scheme where uniqueness of signatures can only be guaranteed if the signer's modulus is known to be the product of two primes [84, 85].

6.7 Mitigation Techniques

For mitigating subliminal communication it might be possible to construct subliminal-freeness proofs like described in Section 5.4 for EdDSA. However, it has to be noted that mitigation strategies that use zero-knowledge proofs appear to be significantly more difficult to construct for MQ signatures, if security in a post-quantum era is a reason for using these schemes. This is because many zero-knowledge proofs rely fundamentally on the hardness assumption of discrete

logarithms. For post-quantum cryptography this assumption can no longer be assumed to hold.
Similar to EdDSA it is possible to make the signing process deterministic by deriving random values from a cryptographic hash function as described for EdDSA in Eq. (5). Since for most MQ signatures not all possible choices of random data lead to a valid signature, in this case it must be possible to deterministically create a sequence of random values, which will be tried consecutively during signing, instead of just one value.
With a deterministic signing process, subliminal communication can be detected when different signatures are generated for two identical messages. However, as described in Section 5.4.2 it is easily possible for an adversary to circumvent this detection technique.

7 TLS Information Hiding

Armed with the findings from the previous chapters, we can now take a closer look at the implications of subliminal channels. For this purpose we select TLS as a real-world protocol and investigate how the subliminal channels contribute to the big picture of information hiding techniques. After all, even if subliminal channels exist, the benefits an adversary gains by their use might be of minor interest if the protocol itself allows vast amounts of covert channels. As it will be shown in this chapter, however, the exploitation of subliminal channels indeed can be very favourable for an adversary.
TLS is the prevalent protocol for protecting communications on today's Internet. In fact, more than half of the total web traffic is already encrypted using TLS [86]. It is this wide deployment that justifies taking a closer look at the possibilities for covert data transmissions using TLS and their security-related implications. Data exfiltration is possible not only in a large amount of different environments but, as TLS usually is used over the Internet, can cross long distances.
On the other hand, a main purpose of TLS is providing confidentiality, i.e. user data is encrypted. A monitoring supervisor who cannot decrypt the communicated contents is not able to investigate these contents for adversarial data transmissions. If data can be hidden in the encrypted payload, the question arises if the additional possibility for covert data transmissions can have any impact on security. We give few reasons for why this question has to be answered in the affirmative.

- By conducting a failing TLS handshake an adversary is able to give the impression that no transmission of data has occurred. Using covert or subliminal channels he, however, could indeed clandestinely exchange data already during the initial TLS handshake.
- In many cases the operation of TLS is performed by third-party libraries instead of by the communicating applications themselves. Exploiting covert or subliminal channels, these libraries are able to communicate unnoticeably without having to alter user data.
- In certain situations like, for example, for monitoring in financial institutions [87], it is reasonable to assume that decryption can be performed by administrators who intercept the communication. Malicious communication using the encrypted user data could thus be detected.
- To be able to communicate using encrypted data (in an efficient manner), the receiver has to hold the decryption key. In scenarios where data should be leaked to an eavesdropper on the path (see e.g. Fig. 1), this requirement often is not satisfied. Certain covert and subliminal channels are exploitable without this requirement. An important use case for this scenario is leaking the decryption key for user data to the eavesdropper.

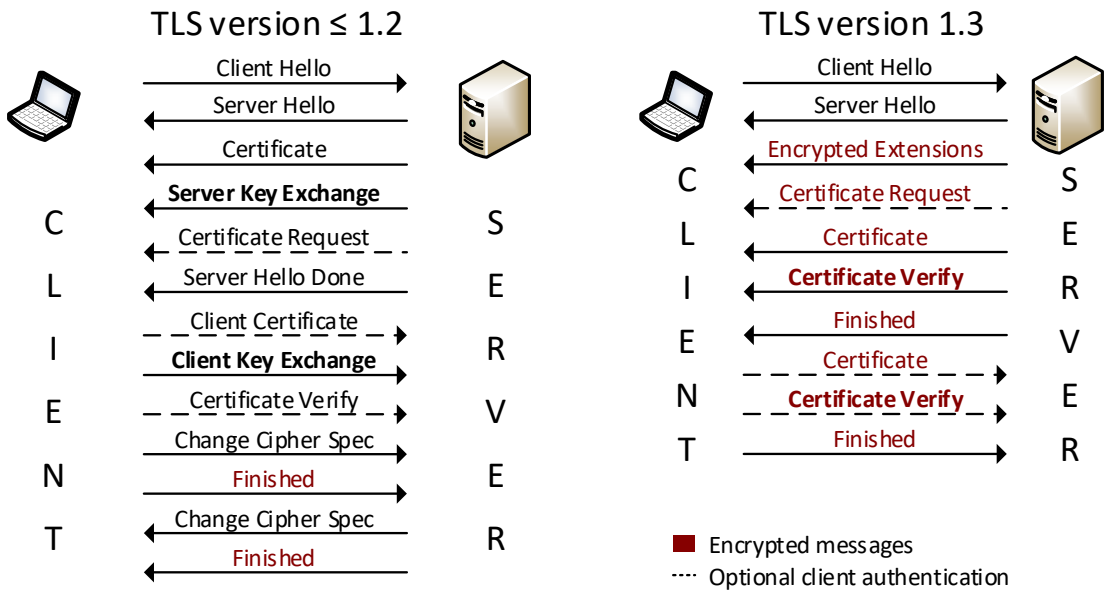

Figure 11: The TLS handshake.

Therefore, in this chapter we will discuss the exploitable covert and subliminal channels in TLS. EdDSA is used as example because it will be supported by upcoming versions of TLS [73, 12], but our results can be applied to other signature schemes as well if they yield subliminal channels.

7.1 The TLS Handshake

The TLS handshake consists of a number of messages exchanged between client and server primarily at the beginning of a connection for agreeing on common connection parameters and cryptographic algorithms, for authenticating the server and, optionally, the client, and for agreeing on a common symmetric secret. In the present context it is of particular importance which parts of the handshake are encrypted with the agreed upon symmetric secret, for example from the Diffie-Hellman key exchange. This is because an important use case of covert and subliminal channels for an adversary is leaking data to a passive eavesdropper on the path, who usually does not know the agreed upon secret. Hence, if data fields are encrypted they cannot (directly) be used as a covert channel.

Fig. 11 shows the handshake for both TLS 1.2 [11] and the new draft version TLS 1.3 [12]. As can be seen from the figure, in TLS 1.3 significantly more handshake data is encrypted. In fact, encryption is applied as soon as the common secret is available.

Table 5: Covert channels in SSL 3.0 and TLS described in literature.

Altered feature	Protocol version	Capacity	Difficulty of detection[1]	References
Client timestamp	≤ TLS 1.2	low	hard	[61]
Client randomness	all	high	hard	[61, 62, 63, 64, 65]
Server timestamp	≤ TLS 1.2	low	hard	[61]
Server randomness	all	high	hard	[61]
Session id	≤ TLS 1.2	high	hard	[61]
Encryption padding	SSL 3.0	high	hard	[61]
Contents of X.509 certificate	all[2]	high	easy	[63]
Cipher suites list	all	low	easy	[61, 63]

[1] Assumes the covert data sequence to be indistinguishable from random data.
[2] For TLS 1.3 only if the encryption key can be leaked to the receiver of the covert data.

7.2 Covert Channels

Due to its complex design, TLS yields numerous covert channels. Table 5 shows TLS covert channels that have been described in literature so far and Table 6 shows further covert channels that have not been described so far but whose use might in some cases yield advantages to an adversary over using the other covert channels (alone) like, for example, for maximizing the total usable covert bandwidth. Most of the covert channels are unencrypted and, hence, are easily exploitable in various scenarios.

Most publications propose to use the randomness field that both client and server transmit in their *Hello* messages as covert channels. These fields contain 32 B of random data for TLS 1.3 and 28 B of random data for prior TLS versions. The exploitation of this randomness field is particularly attractive because it yields a large bandwidth and, if done properly, remains undetectable as encrypted covert information can be assumed to be indistinguishable from a truly random byte sequence. For the same reason the session id field is a good candidate for covert data transmission, at least when requiring the covert data to flow from TLS server to TLS client.

Also, the timestamp values in the *Hello* messages can be used as a covert channel and were proposed to be used by Goh et al. [61]. These timestamps are encoded as Unix time, i.e. as seconds that have elapsed since 1 January 1970 [11]. Assuming

Table 6: Unidentified covert channels in SSL 3.0 and TLS.

Altered feature	Protocol version	Capacity	Difficulty of detection[1]
Client TLS version	all	low	easy
Server TLS version	all	low	easy
Timing between packets	all	high	hard
Length of application data records	all	high	hard
Extensions list	TLS	low	easy
List of compression algorithms	$\leq$ TLS 1.2	low	easy
Frequency of renegotiations	$\leq$ TLS 1.2	low	hard
Frequency of rekeyings	TLS 1.3[2]	low	hard

[1] Assumes the covert data sequence to be indistinguishable from random data.
[2] If the encryption key can be leaked to the receiver of the covert data.

that client and server do not need to be synchronized to very accurate clocks, the timestamps can be expected to deviate from real time by a certain amount. Hence, small changes of these values will go unnoticed and information can therefore be transmitted in the least significant bits of the timestamp. The resulting covert channel will be hard to detect but the bandwidth will be just a few bits per handshake. When using a significant portion of the timestamp value to transmit covert information, on the other hand, the channel will be easy to detect.

Moreover, there are some fields in the TLS protocol that could be used as covert channel but which would be detectable easily because in normal operation the same value occurs across different connections. For example, the *Hello* messages contain various lists to announce supported algorithms and extensions. Information can be encoded both into the entries of these lists and into their order.

Scott proposed to use the transmitted X.509 certificate as a carrier for covert information [63]. The exploitation of the X.509 certificate is on the one hand suspicious due to the varying certificates' contents. On the other hand, in many cases server operators only own a single valid certificate usable for authentication. To avoid failing TLS connections on machines of legitimate users due to an invalid certificate, the covert channel in Scott's example is only used if the client indicates it by using the covert channel in the client randomness field.

Finally, also meta information like the timing between packets or the frequency of renegotiations can be used as covert channels. In the former case even a large exploitable bandwidth might be possible as covert information can be transmitted during the whole TLS connection instead of just during the initial handshake.

Table 7: Subliminal channels in SSL 3.0 and TLS.

Exploited scheme	Protocol version	Bandwidth	Reference
Key exchange using RSA	$\leq$ TLS 1.2	narrow/broad[1]	[64]
Diffie-Hellman key exchange	all	narrow	
DSA	$\leq$ TLS 1.2	narrow/broad[1]	[19]
ECDSA	TLS[2]	narrow/broad[1]	[20]
EdDSA	TLS[2]	narrow/broad[1]	This thesis

[1] Depending on the receiver of the covert data possessing the corresponding private key.
[2] For TLS 1.3 only if the encryption key can be leaked to the receiver of the covert data.

7.3 Subliminal Channels

Subliminal channels hide information exploiting cryptographic algorithms. Compared to the above covert channels, significantly fewer subliminal channels have been identified in TLS. Table 7 lists the cryptographic schemes that can be exploited. All these subliminal channels exploit schemes that are executed for performing the initial key exchange. As these public key algorithms are slow compared to symmetric cryptography, TLS is able to pick up an agreed upon secret from a previous connection (session resumption). In this case none of the following algorithms is used, preventing the subliminal channels. However, it is quite reasonable to assume that an adversary, who is able to modify cryptographic algorithms to embed subliminal information, is also able to modify protocol messages to prevent session resumption.

7.3.1 RSA Ciphertext

Gołębiewski, Kutyłowski, and Zagórski [64] describe a method for exploiting the TLS premaster secret as a hidden information channel if RSA is used for the key exchange. If RSA is used, the premaster secret is chosen by the client, encrypted under the server's public key and sent to the server in the *Client Key Exchange* message. Assuming that neither client nor subliminal receiver hold the server's private key, the value cannot be used as a broadband channel. The subliminal receiver is not able to decrypt the ciphertext. Thus, the subliminal information would have to be contained in the ciphertext instead of the premaster secret itself. On the other hand, lacking the server's private key also the client is not able to choose the premaster secret in a way that makes the ciphertext equal to the

intended subliminal information. The client can, however, repeatedly try different values for the premaster secret until he finds a value that suits his needs. Then the encrypted value would show a pattern that corresponds to the intended subliminal information, like, for example, the last byte being equal to a given value. This channel is similar to the narrowband subliminal channel in EdDSA described in Section 5.2 and only allows a small bandwidth.
The encrypted premaster secret can be used as a broadband subliminal channel if either the client or the subliminal receiver knows the server's private key. Note, however, that this scenario requires collaboration of all three instances, i.e. client, server and subliminal receiver.
Since TLS 1.3 no longer supports key exchange using RSA, this subliminal channel is no longer usable.

7.3.2 Diffie-Hellman Parameters

Another possibility similar to the approach described in the previous Section 7.3.1 or to the narrowband subliminal channel in Section 5.2 is to exploit the exchange of parameters in the Diffie-Hellman key exchange. Using Diffie-Hellman is one option for key negotiation in TLS. Section 2.4 introduced the basic functioning of the Diffie-Hellman key exchange. A narrowband subliminal channel can be established using a trial-and-error approach: Trying different values for α, a value of $g^\alpha \bmod p$ can be found that shows the desired bit pattern by chance, allowing the client to transmit few bits of subliminal information. Similarly, a value of $g^\beta \bmod p$ can be found by the server by chance by trying different values for β to transmit few bits of subliminal information.
However, it is not possible to use the Diffie-Hellman key exchange as a broadband subliminal channel, as it is not feasible to directly encode subliminal information in the transmitted value $g^\alpha \bmod p$ or $g^\beta \bmod p$, at least not if the connection establishment is meant to succeed. The discrete logarithm α has to be known by the client to calculate the joint secret key. Thus, if the client wants to transmit information directly in $g^\alpha \bmod p$ he would need to find an α that generates the subliminal message $g^\alpha \bmod p$. Since computing the discrete logarithm is infeasible, the value $g^\alpha \bmod p$ cannot be chosen equal to the intended subliminal information. The same considerations hold for the reverse direction where the server wants to transmit subliminal information in $g^\beta \bmod p$.

7.3.3 Digital Signatures

Digital signatures can provide the opportunity to establish broadband subliminal channels in TLS. During the handshake signatures are used to prove the identity of the server and, optionally, of the client. Signatures furthermore ensure the integrity

of the Diffie-Hellman key exchange and are an essential building block for the certificate chain used to verify the server's identity based on a set of certificate authorities.

If ephemeral Diffie-Hellman is used for key exchange, the authenticity of the server is verified using a signature. In protocol versions prior to TLS 1.3, the signature is transmitted together with the server's Diffie-Hellman parameters in the *Server Key Exchange* message. Besides these parameters the signed data only contains the random values from the client's and the server's *Hello* messages. As visible from Fig. 11 almost the complete handshake is unencrypted and both the message and its signature, which are needed for recovery of the subliminal information, are therefore known by an eavesdropper. Furthermore, as described in Section 5.4.2, the inclusion of the client's and server's random values in the signed data hampers detection of the subliminal channel, as in normal protocol operation recurring signed messages will occur with negligibly low probability.

The client can also be authenticated using a certificate. In this case the corresponding signature is transmitted in the *Certificate Verify* message and is computed over all handshake messages up to the *Certificate Verify*. Therefore, when using client authentication, subliminal channels exist in both directions, otherwise only the subliminal channel originating from the server can be exploited.

In contrast to earlier TLS versions, the use of ephemeral Diffie-Hellman is enforced in the upcoming version TLS 1.3 with the associated parameters being exchanged already in the client's and the server's *Hello* messages. The signatures used for authentication are exchanged in the *Certificate Verify* messages. The signed data now contains the entire handshake up to the respective *Certificate Verify* message for both server and client. The most important difference in TLS 1.3 compared to earlier versions is the fact that the handshake data is now encrypted as soon as the shared secret from the key exchange algorithm is available. Therefore, all messages following the *Server Hello* are unavailable to passive eavesdroppers that do not know the encryption key. An eavesdropper can see neither the signed message nor the signature and therefore cannot recover the subliminal information using Eq. (14). For this reason, to use the exchange of signatures as a broadband subliminal channel, the subliminal receiver additionally needs to know the encryption key. The narrowband subliminal channel remains usable in TLS 1.3 even if decryption cannot be performed, though.

Finally, the signatures used for building the certificate chain can equally be used to carry hidden data in the course of the TLS handshake. However, as described in Section 7.2, this approach might be easy to detect in comparison to above mentioned signatures or even cause connections from legitimate users to fail.

Table 7 provides references for subliminal channels discovered that may be used in TLS. Current TLS deployments support either DSA or ECDSA when using the

Diffie-Hellman key exchange. EdDSA is one of the new signature scheme options available in TLS 1.3 [12] and is also proposed to be used with version 1.2 and earlier TLS versions [73]. Therefore, it is quite likely that EdDSA is implemented in current and future implementations of TLS and the subliminal channel in EdDSA can be utilized to transmit hidden information.

7.4 Mitigation Techniques

The above sections depict various ways for clandestinely transmitting information exploiting TLS. Is it possible to prevent any adversarial communication? Evidently, to ensure the correct use of the protocol fields, an instance is required that supervises the communication. Let use, hence, reuse the warden scenario from Section 5.4. In Fig. 12 the warden server ensures that the protocol is used in the correct way, but does not own the TLS servers' private keys.

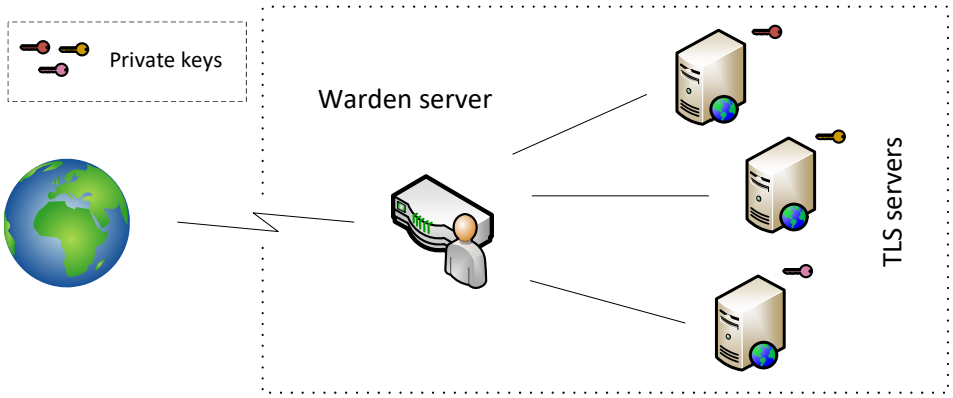

Figure 12: Warden scenario for TLS servers.

Covert Channels

In this setting the majority of covert communication can be prevented in a rather straightforward way. On the one hand, fields like the server TLS version, the X.509 certificate or the extensions list solely depend on the servers' configuration and, hence, stay constant for different clients. Hence, this configuration can be enforced by the warden server and any connection that differs in this respect can be rejected. On the other hand, random data like the server randomness field or the session id can be derived deterministically from carefully selected seed values like the client's IP address, a timestamp or a sequence number. Alternatively, to make sure the warden server is not able to weaken the encryption by influencing these random values, a VRF might be used, for which the TLS servers own the private keys.

Mitigation Techniques for EdDSA

For mitigating subliminal communication using EdDSA signatures, the mitigation techniques described in Section 5.4 can be used.
As described in Section 5.4, using preannounced nonce points is possible only in very specific situations and causes large storage requirements on the warden server. The interactive method is well usable but slows down the connection establishment by a certain amount, because several messages have to be exchanged between warden server and TLS server. The requirement for bidirectional communication is likely to be satisfied in this case. The non-interactive method, finally, is not likely to be a feasible approach in this case, at least not if proofs have to be generated in real-time. The connection establishment would be delayed by a too large amount for the non-interactive approach due to the large computational overhead.
Compared to the above techniques for preventing covert channels, the effort for preventing subliminal channels in EdDSA is significantly larger. Note, furthermore, that while TLS 1.3 limits the usability of the subliminal channel by encrypting larger amounts of handshake data, it also prevents the techniques for preventing the subliminal channel depicted here, if the warden server is not able to decrypt the signatures.

7.5 The Impact of Subliminal Channels in TLS

Tables 5, 6 and 7 showed that there are numerous techniques for information hiding exploitable in TLS, which differ with respect to the bandwidth they yield, their concealment properties, the algorithms they can be used with and the necessity of private keys for recovery of the information. Hence, the choice of covert or subliminal channels an attacker will use, depends on the specific scenario.

Bandwidth

Most channels described above allow data to be injected in messages exchanged during the initial handshake. The amount of data is therefore large if a large amount of connection establishments occurs with a comparatively low amount of overt data being exchanged afterwards. Examples are applications for updating news headlines, social networks or software update requests with small poll intervals. For such scenarios the subliminal channels during key exchange allow a significant usable bandwidth for hidden data. When using TLS 1.2 with EdDSA, 28 B can be transmitted from the server to the client using the randomness field and 31 B can be transmitted using the broadband subliminal channels EdDSA yields. The contribution of the subliminal channel thus constitutes a significant amount.

If, on the other hand, connections are likely to be kept alive during long periods, exchanging large amounts of data, covert channels like timing channels are more suitable for maximizing the amount of secretly transmittable data.

Concealment Properties

Considering detectability, the subliminal channels turn out to be particularly attractive to an adversary. If fields exchanged throughout the normal operation of the cryptographic algorithms are directly derived from random data, there is no way to determine, if specific values have been used for these random values.
Even though the legitimate signing process for EdDSA is deterministic, as described in Section 5.4.2, the exploitation of the subliminal channel in EdDSA can still hardly be detected if the attacker behaves smart. In particular, recurring signed messages will occur with negligibly low probability in normal protocol operation as the signed data includes the random numbers exchanged in the client's and the server's *Hello*s.
Hence, compared to some of the covert channels, the subliminal channel also yields very good concealment properties. Apparently, also the randomness values in the client's and server's *Hello* messages show this property if the covert message is encrypted before being included in the randomness field. However, the subliminal channels might still yield advantages compared to covert channels for an adversary, since as described in Section 7.4 also prevention of the hidden communication is more difficult and accompanied by drawbacks like a larger delay for connection establishment.

Requirements

The most serious requirement for the broadband subliminal channels is the necessity for the subliminal receiver to know the corresponding private keys. Beneath this requirement, the broadband subliminal channels furthermore require that session resumption is not used for agreeing on a common key, as in this case no public key algorithms are used. In most cases preventing session resumption is not likely to pose problems to the adversary. On the other hand, while the subliminal channels are unlikely to be detected, the prevention of session resumption could be suspicious behaviour.

Location of the Covert Receiver

Finally, it has to be noted that the new version TLS 1.3 introduces some protocol modifications that change the applicability of some of the described covert and subliminal channels. Beneath dropping the support for RSA as key exchange method, the major difference in our context is that large parts of the handshake are

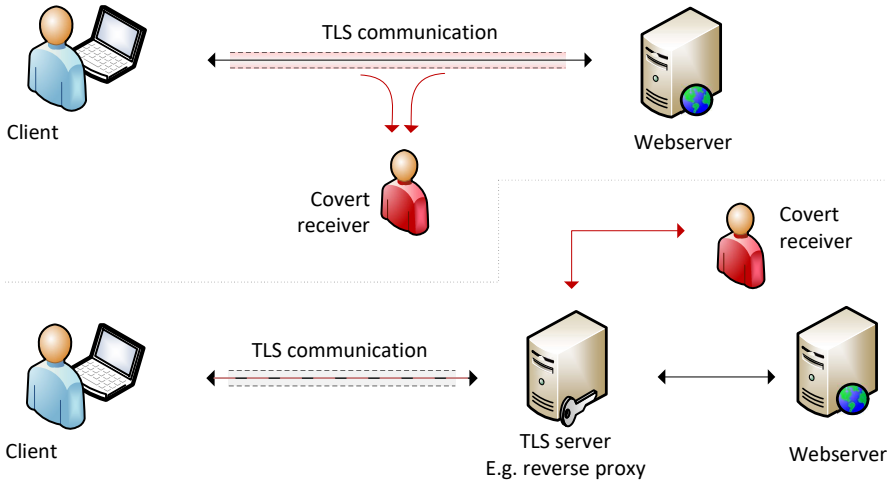

Figure 13: Scenarios for hidden communication exploiting the TLS protocol.

encrypted. For discussing the usable information hiding methods we thus have to distinguish the two scenarios depicted in Fig. 13. In the case depicted on the top of the figure, the receiver of the hidden information is located on the path between client and server and eavesdrops on the connection, while in the other case he is co-located with the TLS server. In the latter case it can be further differentiated if the TLS session is terminated by the application server itself or by a different server as depicted in the figure.

If the receiver eavesdrops on the connection, he is at first not able to perform decryption, thus preventing direct use of the encrypted fields for hiding information and, in particular, ruling out the broadband subliminal channel EdDSA yields. This might be worked around by performing a key-recovery attack first, which, however, obviously poses a major restriction. Moreover, the use of the broadband subliminal channel for the very purpose of performing a key-recovery, is not possible, because the decryption key must be known before the subliminal channel can be used.

If, on the other hand, the hidden information exchange is meant to take place between client and server, decryption can be performed by the receiver and the broadband channel is perfectly usable. An exemplary use case of hiding information exchange between client and server is concealing the channel from a monitoring party that is able to decrypt user data. Yet, usability in this case appears to be more limited.

Concluding, while the subliminal channel in EdDSA significantly improves an attacker's possibilities for information hiding in TLS for versions prior to version 1.3, for the new version exploitation of the subliminal channel by an attacker is justified for few use cases only.

8 Experimental Results

We performed experiments to confirm the functioning of the broadband channels in practice. To retain best possible proximity to a practical situation, we used the scenarios highlighted in Chapter 4 as basis. Our goals when performing the experiments were to prove the existence of the subliminal channel, to get an impression of the difficulty of exploiting the subliminal channel and to find the bit rate with which data can be leaked in practice.

8.1 NTP Broadcasts

Annessi, Fabini, and Zseby [66] proposed to use EdDSA to sign NTP broadcasts. We here reused their experimental setup to investigate the subliminal channel, which is depicted in Fig. 14. Instead of modifying the source code of the NTP server and client processes, the tasks of signature generation and verification are thus performed by network bridges between server and client. Therefore, the subliminal information can be embedded by the signer bridge. The subliminal receiver can be located anywhere on the broadcast domain between signer and verifier bridge.

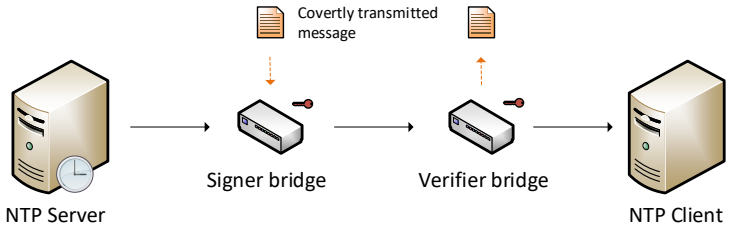

Figure 14: Experimental setup for Section 8.1.

Server and client were running Debian 'Jessie' as operating system and the network bridges were running Debian 'Stretch'. The insertion and removal of signatures was performed with iptables using the nfqueue target. We could thus use a userspace C application to process the packets. We used the cryptographic primitives from the NaCl[4] library in version 20110221 to perform the tasks of signature generation and verification. For the task of recovery of the subliminal information we had to enhance the library by a routine for performing finite field subtractions. Apart from this modification, substitution and recovery of the nonce value was straightforward.
In a practical setting, the handling of partial bytes is difficult. Hence, for sake of simplicity, we transmitted 248 bit instead of the theoretically possible 252 bit of subliminal information.

[4] https://nacl.cr.yp.to/

Using this setup the subliminal channel was proven functional with the expected subliminal bandwidth. Without modifying its source code, the NTP process allows the broadcasting interval of time information to be configured to not less than 8 seconds. We therefore were able to observe a subliminal bit rate of 3.9 B/s.

8.2 PMU Sensor Data Transmissions

For the second experiment we reused the signer and verifier bridges from the previous setup Fig. 14 to investigate signing sensor data transmissions by PMUs. Instead of the NTP server we thus used a 1133A Power Sentinel by Arbiter Systems as data source. When used with the manufacturer's proprietary PowerSentinelCSV protocol, the device is able to send 10 UDP packets of measurement data per second. Adding an EdDSA signature to each packet, we were thus able to transmit 310 B/s of subliminal data, which constitutes a considerable bit rate for data exfiltration. Other protocols for the same task like the IEEE C37.118 allow a comparable measurement frequency, resulting in a similar bit rate for data exfiltration.

8.3 TLS Key Exchange

As described in Section 7, TLS can use signatures to ensure authenticity of the communicating partners and integrity of the key exchange. We performed an experiment to show the practical exploitability of these signatures as subliminal channel when using EdDSA. For this purpose, we used the nginx[5] webserver in version 1.13.0, a simple HTTP client application, both compiled with Google's OpenSSL fork BoringSSL[6]. We chose this library because of its support of both Ed25519 and a draft version of TLS 1.3. Also in this case we had to enhance the library by a function that performs field subtractions. In addition to this modification, just minor modifications were necessary for both TLS 1.2 and TLS 1.3 to be able to exploit the subliminal channel.
Hence, also for the TLS handshake the subliminal channel was proven easily exploitable allowing 31 B of data to be transmitted per key exchange when, again, avoiding to transmit partial bytes. If client authentication is used and thus signatures are exchanged in both directions, the subliminal channel can be exploited in both directions as well.

8.4 MQQ-SIG: A Practical MQ Subliminal Channel

To verify the practical feasibility of the described MQ subliminal channels, we investigated a subliminal channel in the MQQ-SIG signature scheme by Gligoroski

[5] http://nginx.org/
[6] https://boringssl.googlesource.com/boringssl/

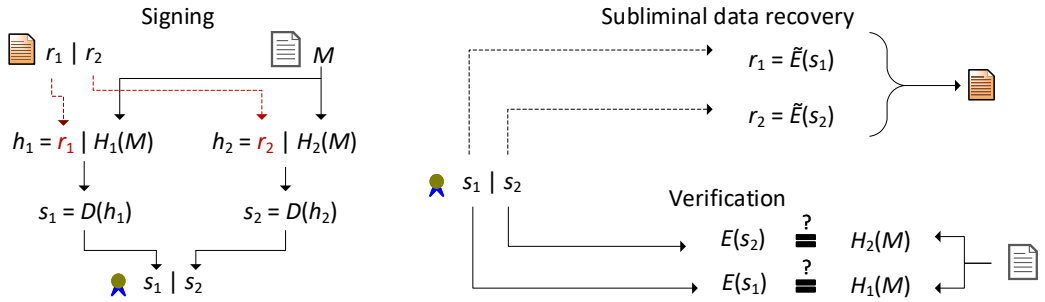

Figure 15: Embedment of subliminal channels in MQQ-SIG signatures.

et al. and developed a proof-of-concept. MQQ-SIG was found in 2012 and broken 3 years later in [88]. It uses the MQQ trapdoor and the minus modification to improve its security.

The scheme uses the usual composition $P = T \circ F \circ S$ for signature generation and verification. T and S are a linear and an affine mapping, respectively, that are constructed from a random nonsingular matrix $\boldsymbol{S} \in \{0,1\}^{n \times n}$ and a random vector $\boldsymbol{v} \in \{0,1\}^n$ according to

$$T(\boldsymbol{y}) = \boldsymbol{S}\boldsymbol{y} \qquad \text{and}$$
$$S(\boldsymbol{s}) = \boldsymbol{S}\boldsymbol{s} + \boldsymbol{v}.$$

The signature scheme uses the MQQ trapdoor, i.e. the mapping $F\colon \{0,1\}^n \to \{0,1\}^n$ is constructed by interpreting the input vector as $n/8$ elements X_i of a quasigroup of order 2^8, which are mapped according to Eq. (11).

Following this construction, P consists of n quadratic polynomials. The signer is able to find $D(\boldsymbol{y}) = P^{-1}(\boldsymbol{y})$. The second half of the polynomials of P forms the mapping $E(\boldsymbol{s})$, which is used as public key.

To generate a signature, two cryptographic hash functions of length $n/2$ are applied to the message. The results are prefixed with random bits to have a length of n bits resulting in the vectors $\boldsymbol{h}_1, \boldsymbol{h}_2 \in \{0,1\}^n$. The signature consists of the vectors $\boldsymbol{s}_1 = D(\boldsymbol{h}_1)$ and $\boldsymbol{s}_2 = D(\boldsymbol{h}_2)$. For verification of a signature, $E(\boldsymbol{s})$ is applied to both $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$ and the result is checked to be equal to the original hash values of the message. Fig. 15 illustrates the functioning of MQQ-SIG.

The scheme as described here uses the minus modification, which was introduced after the cryptanalysis of unmodified MQQ. Instead of all n polynomials just $n/2$ are published as public key and the corresponding parts of $\boldsymbol{h}_1$ and $\boldsymbol{h}_2$ are padded with random data. This way key recovery attacks like [89, 90] shall be prevented. However, as described in Section 6.3, it is remarkably easy to use the signature as a subliminal channel if the minus modification is used. The signer just has to pass on the missing public key equations, which then can be used as a function $\tilde{E}(\boldsymbol{s})$ that recovers the subliminal information.

Table 8: Experimental results.

Scenario	**Amount of data**
NTP broadcasts	max. 3.9 B/s
PMU measurements	310 B/s
TLS handshake	31 B per handshake
MQQ-SIG	8 B per signature

To verify that this subliminal channel works in practice, we used the reference implementation of the signature algorithms available from the SUPERCOP project[7] with a signature length of 256 bit. We modified the key generation code to not only output half of the equations P as public key, but also output a key for recovery of the subliminal information. Furthermore, we modified the signing algorithm to use the subliminal information as random data for signing. With these preparations recovery of the subliminal information is straight-forward: The same algorithm as for verification can be reused, feeding it with the subliminal recovery key instead of the public key and outputting the subliminal information before comparison with the message hash.
In this setting the subliminal channel was proven operational with the predicted subliminal bandwidth. By default, the implementation removed just 1/4 of the public key equations instead of half of them as described above. Hence, a subliminal bandwidth of 64 bit per signature was possible.

8.5 Summary

The above experiments showed that also in a practical setting the subliminal channels can be exploited easily. In particular, for the embedment process in EdDSA signatures just few lines of code have to be modified to use the subliminal information in place of the deterministically computed nonce value. The recovery process is slightly more complex as a function for performing finite field subtractions has to be implemented, but does not pose a burden for any reasonable adversary.
Similarly, for MQQ-SIG the exploitation of the subliminal channel was proven easily achievable by reasonable adversaries.
Table 8 summarizes the results of the described experiments. The bit rate with which data can be leaked is large especially for sensor data transmissions.

[7] https://bench.cr.yp.to/supercop.html

9 Conclusions

In Fig. 1 Alice tried to clandestinely transmit a secret document from her company's network to Mallory, an eavesdropper outside. Considering the arguments given throughout this thesis we now see that, if the communication is secured using EdDSA or an MQ-based signature like, e.g., PFlash, chances are good that she will be able to reach this goal by using a subliminal channel. In fact, a good part of the signatures' bits can be exploited to carry information.
In particular, we showed that for the two variants of EdDSA standardized by the IETF, Ed25519 and Ed448, the resulting broadband subliminal bandwidths are 252 bit per signature and 447 bit per signature, respectively, so almost 50% of a signature's bits can be exploited to carry information. We furthermore presented a narrowband subliminal channel which has requirements that can be met easily by any adversary.
Considering the use of EdDSA in new versions of the frequently used TLS protocol, serious scenarios for data exfiltration result. Even though the protocol yields numerous covert channels, the subliminal channel in certain situations allows an adversary to leak even more data and simultaneously achieve even better concealment properties. In the light of these data exfiltration scenarios, our investigation of mitigation strategies for subliminal communication in EdDSA is a necessity. Unfortunately, among all methods for preventing the subliminal communication, none turned out to be a universal solution for practical scenarios, again reinforcing the risks that follow from the subliminal channels we showed up.
We came to a very similar result for the class of signatures based on MQ cryptography. Almost all methods for constructing secure variants of such signature schemes use randomness throughout the signing process, giving rise to the possibility of subliminal channels. We showed how the minus modification and vinegar variables can be exploited to gain very efficient subliminal channels that furthermore yield attractive properties for the adversary. These very modifications are currently the most promising candidates for building secure MQ signature schemes. Based on these findings, we analysed existing MQ signature schemes and found subliminal bandwidths ranging from approximately 12 bit per signature for QUARTZ to 128 bit per signature for MQQ-SIG.
While the subliminal channels allow serious threats for traditional use cases like authentication in TLS, the situation becomes even worse when considering high-speed signing scenarios, for which EdDSA and certain MQ schemes are predestined. Due to the high number of transmitted signatures, vast amounts of information can be leaked in this case. As examples of such scenarios, we presented signing of NTP broadcasts and smart grid phasor measurements and we backed our argumentation by practical experiments, achieving 310 B/s for data exfiltration using signed PMU measurements, which is a substantial rate in a security-critical environment.

9.1 Contributions

Even though the concept of subliminal channels has been known for a long time, it often did not receive much attention when designing network protocols or otherwise deploying digital signatures so far. In this thesis we showed the risks arising due to this approach which can evolve into serious security-related issues. In more detail, the contributions of this thesis can be summarized as follows:

- In general, we analysed scenarios where high-speed signing is necessary and we came to the conclusion that in these scenarios severe attacks become possible if the used signature scheme allows the establishment of a subliminal channel. Even if we ascribed subliminal channels a low impact on information security for traditional signing scenarios, our results show that the risks emerging from subliminal channels in these scenarios are immense and need to be evaluated when designing network protocols for security-critical environments.
- In particular, we showed how EdDSA signatures can be exploited to establish subliminal channels. While the existence of a subliminal channel is not surprising considering EdDSA's resemblance to prior schemes like DSA and ECDSA, our evaluation depicts in detail how the subliminal channels can be exploited both in theory and practice. Moreover, we surveyed techniques for preventing the subliminal communication for EdDSA.
- We furthermore showed that the whole class of MQ signatures is prone to the existence of subliminal channels due to the inevitable need for randomness in almost all methods for constructing secure MQ signature schemes. Similar to currently known subliminal channels of e.g. (EC)DSA, the subliminal information can be recovered in a very easy and efficient manner. Moreover, unlike most currently known broadband subliminal channels, the adversary can achieve a trade-off between subliminal bandwidth and resistance against forgery attacks performed by the subliminal receiver, which increases the attractiveness of the subliminal channels for an adversary.
- We evaluated the bit rates with which data can be exfiltrated exploiting different signature schemes and in several real-world scenarios. We performed an empirical investigation of several of the subliminal channel scenarios, proving the practicability of data exfiltration and confirming the attainable bit rates also in practice.

In a world where information technology becomes increasingly important, these are important results for information security and privacy. Cyberphysical systems like sensor networks or the Internet of Things are emerging just now and will play a crucial role in the future. Hence, the time is now to think about the right ways for ensuring secure communication of these devices.

Protocol designers can therefore be given the advice to evaluate carefully whether a subliminal channel constitutes a security threat for the intended application of digital signatures and, when this is the case, refrain from using the new signature schemes, even if they have very desirable properties. Possible alternatives are using one of the few subliminal-free signature schemes or, if possible, using symmetric cryptography for achieving authenticity for scenarios where non-repudiation is not required.
Encryption of signatures, on the other hand, is only to a limited extent useful for preventing subliminal communication. While the Sign-then-Encrypt method reduces the usability of broadband subliminal channels to scenarios where the subliminal receiver knows the decryption key, it still remains possible to use a narrowband subliminal channel if this requirement cannot be satisfied.

9.2 Limitations

Incorporating new fields of cryptography, there is a vast number of methods for constructing signature schemes. In this thesis we therefore had to focus on some of them for investigating the possibility of subliminal channels. The particular choice of EdDSA and MQ-based signatures was due to their very attractive properties allowing their use in high-speed signing scenarios. However, also other signature schemes might be suitable for this purpose, heavily depending on the needs of a specific use case.
For the specific case of MQ signatures, furthermore, we detailed on possibilities for exploiting the subliminal channels while only briefly discussing mitigation techniques. In fact, the invention of mitigation techniques based on zero-knowledge proofs like described for EdDSA seems like a feasible task. However, basing these techniques on the discrete logarithm problem would be of limited sense, assuming post-quantum security is a desired property, as the subliminal-freeness proofs in this case would not be able to retain the signature schemes' post-quantum security. Consequently, such zero-knowledge techniques would have to be based on, e.g., the MQ problem as well. While few such zero-knowledge techniques have been proposed already, there is only a very small amount of research so far a subliminal-freeness proof could build on.
Finally, when evaluating our results empirically, we mainly focused on the embedment and recovery process of the subliminal information, justifying a simplified experimental setup. Our analysis thus did not cover additional steps such as leaking secret keys or substituting signing code on sending devices using, for example, malware.
These limitations show us that a large amount of future research is required in the area of subliminal channels.

9.3 Future Work

Considering the major impact of subliminal channels on information security, substantial further research is required for finding ways to mitigate the subliminal channels while retaining at least some of the signature schemes' attractive properties. For example, despite its drawbacks, the interactive method for making EdDSA subliminal-free might in some cases constitute a reasonable approach. It is thus required to realize the method as a network protocol and to measure what additional delay and computational requirements it causes for signing. Furthermore, for Dong and Xiao's non-interactive method it would be interesting to investigate if the used zero-knowledge proofs can be optimized to yield smaller proof sizes and be computationally less demanding.
Furthermore, most methods for constructing MQ signature schemes allow subliminal channels. In particular, the schemes currently considered most secure, allow a number of subliminal channels. Even if these signature schemes turn out to be secure, our considerations motivate also focusing on bijective trapdoors like STS or C^* for further research to find subliminal-free signature schemes or to come up with entirely new constructions for MQ signatures. On the other hand, subliminal-freeness for this kind of signatures could be achieved using mitigation techniques comparable to those presented for EdDSA. Assuming that the deployed zero-knowledge proofs should be post-quantum secure, future work is required to build zero-knowledge proofs based on, e.g., the MQ problem.
Considering novel approaches for constructing signature schemes in general, the most obvious task is to extend our considerations to further classes of cryptography. Beneath MQ cryptography, also hash-based, code-based and lattice-based cryptography yield several interesting algorithms (see e.g. [91]). The possibility of subliminal channels has not been researched to a large extent for these algorithms, which, however, will be necessary at the latest when one of them turns out to be a reasonable choice for achieving authenticity for future deployments.
From the current point of view, however, the best approach to avoid the risks resulting from subliminal channels is to fall back to a subliminal-free signature scheme. It thus has to be evaluated which signature schemes are possible for particular scenarios and which of them qualify for the different use cases highlighted throughout this thesis. Even though for the depicted high-speed signing scenarios the use of slow signature schemes seems hardly possible, for many signature scenarios with less stringent performance requirements the use of a traditional, subliminal-free signature scheme can mitigate the security threats resulting from subliminal channels.

References

[1] Alexander Hartl, Robert Annessi, and Tanja Zseby. "A Subliminal Channel in EdDSA: Information Leakage with High-Speed Signatures". In: *Proc. of the 9th ACM CCS International Workshop on Managing Insider Security Threats (MIST'17), Dallas, USA*. ACM, Oct. 2017, pp. 67–78.

[2] Alexander Hartl, Robert Annessi, and Tanja Zseby. "Subliminal Channels in High-Speed Signatures". In: *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)* 9.1 (Mar. 2018), pp. 30–53.

[3] Ronald L. Rivest, Adi Shamir, and Len Adleman. "A method for obtaining digital signatures and public-key cryptosystems". In: *Communications of the ACM* 21.2 (Feb. 1978), pp. 120–126.

[4] Taher ElGamal. "A public key cryptosystem and a signature scheme based on discrete logarithms". In: *Advances in Cryptology – Proc. of the CRYPTO'84*. LNCS vol. 196. Springer, Aug. 1984, pp. 10–18.

[5] Don Johnson, Alfred Menezes, and Scott Vanstone. "The elliptic curve digital signature algorithm (ECDSA)". In: *International Journal of Information Security* 1.1 (Aug. 2001), pp. 36–63.

[6] Gustavus J. Simmons. "The Prisoners' Problem and the Subliminal Channel". In: *Advances in Cryptology – Proc. of the CRYPTO'83*. LNCS vol. 1440. Springer, Jan. 1984, pp. 51–67.

[7] Gustavus J. Simmons. "The history of subliminal channels". In: *IEEE Journal on Selected Areas in Communications* 16.4 (May 1998), pp. 452–462.

[8] Daniel J. Bernstein et al. "High-Speed High-Security Signatures". In: *Proc. of the 13th International Workshop on Cryptographic Hardware and Embedded Systems (CHES'11), Nara, Japan*. LNCS vol. 6917. Springer, Sept. 2011, pp. 124–142.

[9] Ming-Shing Chen, Bo-Yin Yang, and Daniel Smith-Tone. "PFLASH - Secure Asymmetric Signatures on Smart Cards". In: *Proc. of the Lightweight Cryptographic Workshop 2015, Gaithersburg, USA*. NIST, July 2015.

[10] Danilo Gligoroski et al. "MQQ-SIG: An Ultra-fast and Provably CMA Resistant Digital Signature Scheme". In: *Proc. of the 3rd International Conference on Trusted Systems (INTRUST'11), Beijing, China*. LNCS vol. 7222. Springer, Nov. 2012, pp. 184–203.

[11] T. Dierks and E. Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.2*. RFC 5246 (Proposed Standard). IETF, Aug. 2008. URL: http://www.ietf.org/rfc/rfc5246.txt.

[12] Eric Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.3*. Internet-Draft draft-ietf-tls-tls13-28. IETF, Mar. 2018. URL: http://www.ietf.org/internet-drafts/draft-ietf-tls-tls13-28.txt.

[13] D. Mills et al. *Network Time Protocol Version 4: Protocol and Algorithms Specification*. RFC 5905 (Proposed Standard). IETF, June 2010. URL: http://www.ietf.org/rfc/rfc5905.txt.

[14] ISO. *Information processing systems – Open Systems Interconnection – Basic Reference Model – Part 2: Security Architecture*. Standard. Geneva, Switzerland: International Organization for Standardization, 1989.

[15] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, Oct. 1996. ISBN: 978-0-849-38523-0.

[16] I. Blake, G. Seroussi, and N. Smart. *Elliptic Curves in Cryptography*. London Mathematical Society Lecture Note Series. Cambridge: Cambridge University Press, 1999. ISBN: 978-0-521-65374-9.

[17] Jee Hea An. "Authenticated Encryption in the Public-Key Setting: Security Notions and Analyses". In: *IACR Cryptology ePrint Archive* (June 2001). URL: http://eprint.iacr.org/2001/079.

[18] Wojciech Mazurczyk et al. *Information Hiding in Communication Networks: Fundamentals, Mechanisms, Applications, and Countermeasures*. 1st. Wiley-IEEE Press, 2016. ISBN: 978-1-118-86169-1.

[19] Gustavus J. Simmons. "Subliminal Communication is Easy Using the DSA". In: *Advances in Cryptology – Proc. of the EUROCRYPT'93*. LNCS vol. 765. Springer, May 1993, pp. 218–232.

[20] Jens-Matthias Bohli, Maria Isabel Gonzalez Vasco, and Rainer Steinwandt. "A subliminal-free variant of ECDSA". In: *Proc. of the 8th International Workshop on Information Hiding (IH'06), Alexandria, USA*. LNCS vol. 4437. Springer, July 2006, pp. 375–387.

[21] Q. Dong and G. Xiao. "A Subliminal-Free Variant of ECDSA Using Interactive Protocol". In: *Proc. of the 2010 International Conference on E-Product E-Service and E-Entertainment (ICEEE'10), Henan, China*. IEEE, Nov. 2010.

[22] Silvio Micali, Michael Rabin, and Salil Vadhan. "Verifiable random functions". In: *Proc. of the 40th Annual Symposium on Foundations of Computer Science (FOCS'99), New York, USA*. IEEE, Oct. 1999, pp. 120–130.

[23] Yevgeniy Dodis and Aleksandr Yampolskiy. "A Verifiable Random Function with Short Proofs and Keys". In: *Proc. of the 8th International Workshop on Theory and Practice in Public Key Cryptography (PKC'05), Les Diablerets, Switzerland*. LNCS vol. 3386. Springer, Jan. 2005, pp. 416–431.

[24] Shiuan-Tzuo Shen, Amir Rezapour, and Wen-Guey Tzeng. "Unique Signature with Short Output from CDH Assumption". In: *Proc. of the 9th International Conference on Provable Security, Kanazawa, Japan*. LNCS vol. 9451. Springer, Nov. 2015, pp. 475–488.

[25] Silvio Micali and Ronald L. Rivest. "Micropayments Revisited". In: *Topics in Cryptology – Proc. of the RSA Conference 2002, San Jose, USA*. LNCS vol. 2271. Springer, Feb. 2002, pp. 149–163.

[26] Stanislaw Jarecki and Vitaly Shmatikov. "Handcuffing Big Brother: an Abuse-Resilient Transaction Escrow Scheme". In: *Advances in Cryptology – Proc. of the EUROCRYPT'04*. LNCS vol. 3027. Springer, May 2004, pp. 590–608.

[27] R. Lidl and H. Niederreiter. *Introduction to Finite Fields and Their Applications*. Cambridge: Cambridge University Press, 1986. ISBN: 978-0-521-30706-2.

[28] Daniel J. Bernstein et al. "Twisted Edwards Curves". In: *Progress in Cryptology – Proc. of the AFRICACRYPT'08*. LNCS vol. 5023. Springer, June 2008, pp. 389–405.

[29] Albrecht Beutelspacher, Jörg Schwenk, and Klaus-Dieter Wolfenstetter. *Moderne Verfahren der Kryptographie: Von RSA zu Zero-Knowledge*. 6th ed. Vieweg, 2006. ISBN: 978-3-834-80083-1.

[30] Whitfield Diffie and Martin Hellman. "New directions in cryptography". In: *IEEE transactions on Information Theory* 22.6 (1976), pp. 644–654.

[31] Daniel J. Bernstein. "Curve25519: New Diffie-Hellman Speed Records". In: *Proc. of the 9th International Conference on Theory and Practice in Public-Key Cryptography (PKC'06), New York, USA*. LNCS vol. 3958. Springer, Apr. 2006, pp. 207–228.

[32] S. Josefsson and I. Liusvaara. *Edwards-Curve Digital Signature Algorithm (EdDSA)*. RFC 8032 (Informational). IETF, Jan. 2017. URL: http://www.ietf.org/rfc/rfc8032.txt.

[33] Mike Hamburg. "Ed448-Goldilocks, a new elliptic curve". In: *IACR Cryptology ePrint Archive* (June 2015). URL: http://eprint.iacr.org/2015/625.

[34] Claus P. Schnorr. "Efficient Identification and Signatures for Smart Cards". In: *Advances in Cryptology – Proc. of the CRYPTO'89*. LNCS vol. 435. Springer, Aug. 1989, pp. 239–252.

[35] Jacques Patarin. "Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): Two New Families of Asymmetric Algorithms". In: *Advances in Cryptology – Proc. of the EUROCRYPT'96*. LNCS vol. 1070. Springer, May 1996, pp. 33–48.

[36] Aviad Kipnis and Adi Shamir. "Cryptanalysis of the HFE Public Key Cryptosystem by Relinearization". In: *Advances in Cryptology – Proc. of the CRYPTO'99*. LNCS vol. 1666. Springer, Aug. 1999, pp. 19–30.

[37] Jean-Charles Faugère and Antoine Joux. "Algebraic Cryptanalysis of Hidden Field Equation (HFE) Cryptosystems Using Gröbner Bases". In: *Advances in Cryptology – Proc. of the CRYPTO'03*. LNCS vol. 2729. Springer, Aug. 2003, pp. 44–60.

[38] Andrey V Sidorenko and Ernst M Gabidulin. "The weak keys for HFE". In: *Proceedings of the Seventh International Symposium on Communication Theory and Applications (ISCTA03)*. 2003, pp. 239–244.

[39] Daniel Augot, Lejla Batina, et al. "Initial recommendations of long-term secure post-quantum systems". In: *PQCrypto* (Sept. 2015). URL: http://pqcrypto.eu.org/docs/initial-recommendations.pdf.

[40] Christopher Wolf and Bart Preneel. "Taxonomy of Public Key Schemes based on the problem of Multivariate Quadratic equations". In: *IACR Cryptology ePrint Archive* (Mar. 2005). URL: http://eprint.iacr.org/2005/077.

[41] Jacques Patarin. "The oil and vinegar signature scheme". In: *Proc. of the Dagstuhl Workshop on Cryptography*. Sept. 1997.

[42] Aviad Kipnis, Jacques Patarin, and Louis Goubin. "Unbalanced Oil and Vinegar Signature Schemes". In: *Advances in Cryptology – Proc. of the EUROCRYPT'99*. LNCS vol. 1592. Springer, May 1999, pp. 206–222.

[43] Hideki Imai and Tsutomu Matsumoto. "Algebraic methods for constructing asymmetric cryptosystems". In: *Proc. of the 3rd International Conference on Algebraic Algorithms and Error-Correcting Codes (AAECC-3), Grenoble, France*. LNCS vol. 229. Springer, July 1986, pp. 108–119.

[44] Tsutomu Matsumoto and Hideki Imai. "Public Quadratic Polynomial-Tuples for Efficient Signature-Verification and Message-Encryption". In: *Advances in Cryptology – Proc. of the EUROCRYPT'88*. LNCS vol. 330. Springer, May 1988, pp. 419–453.

[45] Nicolas Courtois, Louis Goubin, and Jacques Patarin. *Quartz, an asymmetric signature scheme for short signatures on PC – Primitive specification and supporting documentation*. Oct. 2001.

[46] Danilo Gligoroski, Smile Markovski, and Svein J. Knapskog. "A Public Key Block Cipher Based on Multivariate Quadratic Quasigroups". In: *IACR Cryptology ePrint Archive* (July 2008). URL: http://eprint.iacr.org/2008/320.

[47] Danilo Gligoroski, Smile Markovski, and Svein Johan Knapskog. "Multivariate Quadratic Trapdoor Functions Based on Multivariate Quadratic Qua-

sigroups". In: *Proc. of the American Conference on Applied Mathematics (MATH'08), Cambridge, USA*. WSEAS Press, Mar. 2008, pp. 44–49.

[48] Jintai Ding et al. "Breaking the Symmetry: a Way to Resist the New Differential Attack". In: *IACR Cryptology ePrint Archive* (Sept. 2007). URL: http://eprint.iacr.org/2007/366.

[49] Jacques Patarin, Louis Goubin, and Nicolas Courtois. "C-+* and HM: Variations Around Two Schemes of T. Matsumoto and H. Imai". In: *Advances in Cryptology – Proc. of the ASIACRYPT'98*. LNCS vol. 1514. Springer, Oct. 1998, pp. 35–50.

[50] Jintai Ding and Dieter Schmidt. "Cryptanalysis of HFEv and Internal Perturbation of HFE". In: *Proc. of the 8th International Workshop on Public Key Cryptography (PKC'05), Les Diablerets, Switzerland*. LNCS vol. 3386. Springer, Jan. 2005, pp. 288–301.

[51] Ross Anderson et al. "The Newton channel". In: *Proc. of the First International Workshop on Information Hiding (IH'96), Cambridge, U.K.* LNCS vol. 1174. Springer, May 1996, pp. 151–156.

[52] Xianfeng Zhao and Ning Li. "Reversible Watermarking with Subliminal Channel". In: *Proc. of the 10th International Workshop on Information Hiding (IH'08), Santa Barbara, USA*. LNCS vol. 5284. Springer, May 2008, pp. 118–131.

[53] Jens-Matthias Bohli and Rainer Steinwandt. "On Subliminal Channels in Deterministic Signature Schemes". In: *Proc. of the 7th International Conference on Information Security and Cryptology (ICISC'04), Seoul, Korea*. LNCS vol. 3506. Springer, Dec. 2005, pp. 182–194.

[54] Michael O. Rabin. *Digitalized signatures and public-key functions as intractable as factorization*. Tech. rep. Cambridge, USA: Massachusetts Institute of Technology, 1979.

[55] Yinghui Zhang et al. "Provably Secure and Subliminal-Free Variant of Schnorr Signature". In: *Proc. of the International Conference on Information and Communication Technology (ICT-EurAsia'13), Yogyakarta, Indonesia*. LNCS vol. 7804. Springer, Mar. 2013, pp. 383–391.

[56] Adam Young and Moti Yung. "The Dark Side of "Black-Box" Cryptography or: Should We Trust Capstone?" In: *Advances in Cryptology – Proc. of the CRYPTO'96*. LNCS vol. 1109. Springer, Aug. 1996, pp. 89–103.

[57] Mihir Bellare, Kenneth G. Paterson, and Phillip Rogaway. "Security of Symmetric Encryption against Mass Surveillance". In: *Advances in Cryptology – Proc. of the CRYPTO'14*. LNCS vol. 8616. Springer, Aug. 2014, pp. 1–19.

[58] Giuseppe Ateniese, Bernardo Magri, and Daniele Venturi. "Subversion-Resilient Signature Schemes". In: *Proc. of the 22nd ACM Conference on*

Computer and Communications Security (CCS'15), Denver, USA. ACM, Oct. 2015, pp. 364–375.

[59] Albrecht Petzoldt et al. "Design Principles for HFEv- Based Multivariate Signature Schemes". In: *Advances in Cryptology – Proc. of the ASIACRYPT'15*. LNCS vol. 9452. Springer, Nov. 2015, pp. 311–334.

[60] Jintai Ding and Dieter Schmidt. "Rainbow, a New Multivariable Polynomial Signature Scheme". In: *Proc. of the 3rd International Conference on Applied Cryptography and Network Security (ACNS'05), New York, USA*. LNCS vol. 3531. Springer, June 2005, pp. 164–175.

[61] Eu-Jin Goh et al. "The Design and Implementation of Protocol-Based Hidden Key Recovery". In: *Proc. of the 6th International Conference on Information Security (ISC'03), Bristol, U.K.* LNCS vol. 2851. Springer, Oct. 2003, pp. 165–179.

[62] Justin Merrill and Daryl Johnson. "Covert Channels in SSL Session Negotiation Headers". In: *Proc. of the 13th International Conference on Security and Management (SAM'15), Las Vegas, USA*. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), July 2015, pp. 70–72.

[63] Carlos Scott. *Network Covert Channels: Review of Current State and Analysis of Viability of the use of X.509 Certificates for Covert Communications*. Tech. rep. Roal Holloway, University of London, Jan. 2008.

[64] Zbigniew Gołębiewski, Mirosław Kutyłowski, and Filip Zagórski. "Stealing secrets with SSL/TLS and SSH – Kleptographic attacks". In: *Proc. of the 5th International Conference on Cryptology and Network Security (CANS'06), Suzhou, China*. LNCS vol. 4301. Springer, Dec. 2006, pp. 191–202.

[65] Adam L. Young and Moti M. Yung. "Space-Efficient Kleptography Without Random Oracles". In: *Proc. of the 9th International Workshop on Information Hiding (IH'07), Saint Malo, France*. LNCS vol. 4567. Springer, June 2007, pp. 112–129.

[66] Robert Annessi, Joachim Fabini, and Tanja Zseby. *SecureTime: Secure Multicast Time Synchronization*. May 2017. eprint: arXiv:1705.10669.

[67] E. Itkin and A. Wool. "A security analysis and revised security extension for the precision time protocol". In: *Proc. of the IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS'16), Stockholm, Sweden*. IEEE, Sept. 2016.

[68] Eyal Itkin and Avishai Wool. *A Security Analysis and Revised Security Extension for the Precision Time Protocol*. Mar. 2016. eprint: arXiv:1603.00707.

[69] *IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*. IEEE Std. 1588. 2008.

[70] R. Annessi, J. Fabini, and T. Zseby. "It's about Time: Securing Broadcast Time Synchronization with Data Origin Authentication". In: *Proc. of the 26th International Conference on Computer Communication and Networks (ICCCN'17), Vancouver, Canada*. IEEE, July 2017.

[71] *IEEE Standard for Synchrophasor Measurements for Power Systems*. IEEE Std. C37.118.1. 2011.

[72] O. Sury and R. Edmonds. *Edwards-Curve Digital Security Algorithm (EdDSA) for DNSSEC*. RFC 8080 (Proposed Standard). IETF, Feb. 2017. URL: http://www.ietf.org/rfc/rfc8080.txt.

[73] Yoav Nir, Simon Josefsson, and Manuel Pegourie-Gonnard. *Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS) Versions 1.2 and Earlier*. Internet-Draft draft-ietf-tls-rfc4492bis-17. IETF, May 2017. URL: http://www.ietf.org/internet-drafts/draft-ietf-tls-rfc4492bis-17.txt.

[74] T. Ylonen and C. Lonvick. *The Secure Shell (SSH) Protocol Architecture*. RFC 4251 (Proposed Standard). IETF, Jan. 2006. URL: http://www.ietf.org/rfc/rfc4251.txt.

[75] Ben Harris and Loganaden Velvindron. *Ed25519 public key algorithm for the Secure Shell (SSH) protocol*. Internet-Draft draft-ietf-curdle-ssh-ed25519-02. IETF, Feb. 2018. URL: http://www.ietf.org/internet-drafts/draft-ietf-curdle-ssh-ed25519-02.txt.

[76] C. Kaufman et al. *Internet Key Exchange Protocol Version 2 (IKEv2)*. RFC 7296 (Internet Standard). IETF, Oct. 2014. URL: http://www.ietf.org/rfc/rfc7296.txt.

[77] Yoav Nir. *Using Edwards-curve Digital Signature Algorithm (EdDSA) in the Internet Key Exchange (IKEv2)*. Internet-Draft draft-ietf-ipsecme-eddsa-04. IETF, Oct. 2017. URL: http://www.ietf.org/internet-drafts/draft-ietf-ipsecme-eddsa-04.txt.

[78] S. Frankel and S. Krishnan. *IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap*. RFC 6071 (Informational). IETF, Feb. 2011. URL: http://www.ietf.org/rfc/rfc6071.txt.

[79] M. Naor and O. Reingold. "Number-theoretic constructions of efficient pseudo-random functions". In: *Proc. of the 38th Annual Symposium on Foundations of Computer Science (FOCS'97), Miami Beach, Florida*. IEEE, Oct. 1997, pp. 458–467.

[80] Anna Lysyanskaya. "Unique Signatures and Verifiable Random Functions from the DH-DDH Separation". In: *Advances in Cryptology – Proc. of the CRYPTO'02*. LNCS vol. 2442. Springer, Aug. 2002, pp. 597–612.

[81] Yevgeniy Dodis. "Efficient Construction of (Distributed) Verifiable Random Functions". In: *Proc. of the 6th International Workshop on Practice and Theory in Public Key Cryptography (PKC'03), Miami, USA*. LNCS vol. 2567. Springer, Jan. 2002, pp. 1–17.

[82] Nicolas Courtois, Louis Goubin, and Jacques Patarin. "SFLASHv3, a fast asymmetric signature scheme". In: *IACR Cryptology ePrint Archive* (Oct. 2003). URL: http://eprint.iacr.org/2003/211.

[83] *Portfolio of recommended cryptographic primitives*. NESSIE consortium. Feb. 2003.

[84] Mihir Bellare and Moti Yung. "Certifying Permutations: Noninteractive zero-knowledge based on any trapdoor permutation". In: *Journal of Cryptology* 9.3 (June 1996), pp. 149–166.

[85] Saqib A. Kakvi, Eike Kiltz, and Alexander May. "Certifying RSA". In: *Advances in Cryptology – Proc. of the ASIACRYPT'12*. LNCS vol. 7658. Springer, Dec. 2012, pp. 404–414.

[86] Klint Finley. *Half the Web Is Now Encrypted. That Makes Everyone Safer*. Accessed: 2018-06-05. 2017. URL: https://www.wired.com/2017/01/half-web-now-encrypted-makes-everyone-safer/.

[87] Hanno Böck. *Die Zukunft der Netzverschlüsselung*. Accessed: 2018-06-05. 2016. URL: https://www.golem.de/news/tls-1-3-die-zukunft-der-netzverschluesselung-1612-124724.html.

[88] Jean-Charles Faugère et al. "A Polynomial-Time Key-Recovery Attack on MQQ Cryptosystems". In: *Proc. of the 18th IACR International Conference on Practice and Theory in Public-Key Cryptography (PKC'15), Gaithersburg, USA*. LNCS vol. 9020. Springer, Mar. 2015, pp. 150–174.

[89] Mohamed Saied Emam Mohamed et al. "Algebraic Attack on the MQQ Public Key Cryptosystem". In: *Proc. of the 8th International Conference on Cryptology and Network Security (CANS'09), Kanazawa, Japan*. LNCS vol. 5888. Springer, Dec. 2009, pp. 392–401.

[90] Jean-Charles Faugère et al. "Analysis of the MQQ Public Key Cryptosystem". In: *Proc. of the 9th International Conference on Cryptology and Network Security (CANS'10), Kuala Lumpur, Malaysia*. LNCS vol. 6467. Springer, Dec. 2010, pp. 169–183.

[91] Daniel J. Bernstein, Johannes Buchmann, Erik Dahmen, et al. *Post-Quantum Cryptography*. Springer, Nov. 2008. ISBN: 978-3-540-88701-0.

Statement on Academic Integrity

Hiermit erkläre ich, dass die vorliegende Arbeit gemäß dem Code of Conduct – Regeln zur Sicherung guter wissenschaftlicher Praxis (in der aktuellen Fassung des Mitteilungsblattes Nr. 26/2007 der TU Wien), insbesondere ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel, angefertigt wurde. Die aus anderen Quellen direkt oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet.
Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder in ähnlicher Form in anderen Prüfungsverfahren vorgelegt.

Wien, Juni 2018

Alexander Hartl