Foundations of Coin Mixing Services

Noemi Glaeser
University of Maryland
College Park, MD, USA
Max Planck Insitute for Security and Privacy
Bochum, Germany
nglaeser@umd.edu

Matteo Maffei
TU Wien
Christian Doppler Laboratory Blockchain Technologies for the Internet of Things
Vienna, Austria
matteo.maffei@tuwien.ac.at

Giulio Malavolta
Max Planck Institute for Security and Privacy
Bochum, Germany
giulio.malavolta@hotmail.it

Pedro Moreno-Sanchez
IMDEA Software Institute
Madrid, Spain
pedro.moreno@imdea.org

Erkan Tairi
TU Wien
Christian Doppler Laboratory Blockchain Technologies for the Internet of Things
Vienna, Austria
erkan.tairi@tuwien.ac.at

Sri AravindaKrishnan Thyagarajan
Carnegie Mellon University
Pittsburgh, PA, USA
t.srikrishnan@gmail.com

ABSTRACT

Coin mixing services allow users to mix their cryptocurrency coins and thus enable unlinkable payments in a way that prevents tracking of honest users' coins by both the service provider and the users themselves. The easy bootstrapping of new users and backwards compatibility with cryptocurrencies (such as Bitcoin) with limited support for scripts are attractive features of this architecture, which has recently gained considerable attention in both academia and industry.

A recent work of Tairi et al. [IEEE S&P 2021] formalizes the notion of a coin mixing service and proposes $\mathrm{A}^2\mathrm{L}$, a new cryptographic protocol that simultaneously achieves high efficiency and interoperability. In this work, we identify a gap in their formal model and substantiate the issue by showing two concrete counterexamples: we show how to construct two encryption schemes that satisfy their definitions but lead to a completely insecure system.

To amend this situation, we investigate secure constructions of coin mixing services. First, we develop the notion of blind conditional signatures (BCS), which acts as the cryptographic core for coin mixing services. We propose game-based security definitions for BCS and propose $\mathrm{A}^2\mathrm{L}^+$, a modified version of the protocol by Tairi et al. that satisfies our security definitions. Our analysis is in an idealized model (akin to the algebraic group model) and assumes the hardness of the one-more discrete logarithm problem. Finally, we propose $\mathrm{A}^2\mathrm{L}^{\mathrm{UC}}$, another construction of BCS that achieves the stronger notion of UC-security (in the standard model), albeit with a significant increase in computation cost. This suggests that constructing a coin mixing service protocol secure under composition requires more complex cryptographic machinery than initially thought.

CCS CONCEPTS

• **Security and privacy** → **Cryptanalysis and other attacks**; **Mathematical foundations of cryptography**.

KEYWORDS

blockchain, cryptocurrencies, coin mixing services, cryptanalysis, game-based security, universal composability

1 INTRODUCTION

Bitcoin and cryptocurrencies sharing Bitcoin's core principles have attained huge prominence as decentralized and publicly verifiable payment systems. They have attracted not only cryptocurrency enthusiasts but also banks [5], leading IT companies (e.g., Facebook and PayPal), and payment providers such as Visa [20]. At the same time, the initial perception of payment unlinkability based on pseudonyms has been refuted in numerous academic research works [39, 51], and the blockchain surveillance industry [30] demonstrates this privacy breach in practice. This has led to a large amount of work devoted to providing a privacy-preserving overlay to Bitcoin in the form of *coin mixing* protocols [6, 26].

Decentralized coin mixing protocols such as CoinJoin [1] or CoinShuffle [45–47] allow a set of mutually distrusting users to mix their coins to achieve *unlinkability*: that is, the coins cannot be linked to their initial owners even by malicious participants. These protocols suffer from a common drawback, the *bootstrapping problem*, i.e., how to find a set of participants to execute the protocol. In fact, while a high number of participants is desirable to improve the anonymity guarantees provided by the coin mixing protocol, such a high number is at the same time undesirable as it results in poor scalability and makes bootstrapping harder.

An alternative mechanism is one in which a third party, referred to as the *hub*, alleviates the bootstrapping problem by connecting users that want to mix their coins. Moreover, the hub itself can provide a coin mixing service by acting as a tumbler. In more detail, users send their coins to the hub, which, after collecting all the coins, sends them back to the users in a randomized order, thereby providing unlinkability for an observer of such transfers (e.g., an observer of the corresponding Bitcoin transactions).

Synchronization Puzzles. There are numerous reported cases of "exit scams" by mixing services which took in new payments but stopped providing the mixing service [52]. This has prompted the design of numerous cryptographic protocols [2, 14, 32, 54] to remove trust from the hub, providing a trade-off between trust assumptions, minimum number of transactions, and Bitcoin compatibility [31]. Of particular interest is the work by Heilman et al. [31], which lays the groundwork for the core cryptographic primitive which can be used to build a mixing service. This primitive, referred to as a *synchronization puzzle*, enables unlinkability from even the view of a corrupt hub. However, Heilman et al. only present informal descriptions of the security and privacy notions of interest. Furthermore, the protocol proposed (TumbleBit) relies on hashed time-lock contracts (HTLCs), a smart contract incompatible with major cryptocurrencies such as Monero, Stellar, Ripple, MimbleWimble, and Zerocash (shielded addresses), lowering the interoperability of the solution.

The recent work of Tairi et al. [53] attempts to overcome both of these limitations. It gives formal security notions for a synchronization puzzle in the universal composability (UC) framework [16]. It also provides an instantiation of the synchronization puzzle (called A^2L) that is simultaneously more efficient and more interoperable than TumbleBit, requiring only timelocks and digital signature verification from the underlying cryptocurrencies.

In this work, we identify a gap in their security analysis, and we substantiate the issue by presenting two concrete counterexamples: there exist two encryption schemes (secure under standard cryptographic assumptions) that satisfy the prerequisites of their security notions, yet yield completely insecure systems. This shows that our understanding of synchronization puzzles as a cryptographic primitive is still inadequate. Establishing firm foundations for this important cryptographic primitive requires us to rethink this object from the ground up.

1.1 Our Contributions

We summarize the contributions of this work below.

Counterexamples. First, we identify a gap in the security model of the synchronization puzzle protocol A^2L [53], presenting two concrete counterexamples (Section 3). Specifically, we show that there exist underlying cryptographic building blocks that satisfy the prerequisites stated in A^2L, yet they allow for:

- a *key recovery attack*, in which a user can learn the long-term secret decryption key of the hub;
- a *one-more signature attack*, in which a user can obtain n signed transactions from the hub while only engaging in $n-1$ successful instances of signing a transaction which pays the hub. In other words, the user obtains n coins from the hub while the hub receives only $n-1$ coins.

Both attacks run in polynomial time and succeed with overwhelming probability.

Definitions. To place the synchronization puzzle on firmer foundations, we propose a new cryptographic notion that we call *blind conditional signatures (BCS)*. Our new notion intuitively captures the functionality of a *synchronization puzzle* from [31, 53]. BCS is a simple and easy-to-understand tool, and we formalize its security notions both in the *game-based* (Section 4.1) and *universal composability* (Section 5) setting. The proposed game-based definitions for BCS are akin to the well-understood standard security notions for regular blind signatures [19, 48]. We hope that this abstraction may lay the foundations for further studies on this primitive in all cryptocurrencies, scriptless or not.

Constructions. We give two constructions, one that satisfies our game-based security guarantees and one that is UC-secure. Both require only the same limited functionality as A^2L from the underlying blockchain. In more detail:

- We give a modified version of A^2L (Sections 4.2 and 4.3) which we refer to as A^2L^+ that satisfies the game-based notions (Section 4.1) of BCS, albeit in the *linear-only encryption (LOE)* model [29]. In this model, the attacker does not directly have access to a homomorphic encryption scheme; instead, it can perform the legal operations by querying the corresponding oracles. This is a strong model with a non-falsifiable flavor, similar to the generic/algebraic group model [24, 37, 50].
- We then provide a less efficient construction A^2L^{UC} that securely realizes the UC notion of BCS (Section 5). This scheme significantly departs from the construction paradigm of A^2L and is based on general-purpose cryptographic tools such as secure two-party computation (2PC).

Our results hint at the fact that achieving UC-security for a synchronization puzzle requires a radical departure from current construction paradigms, and it is likely to lead to less efficient schemes. On the other hand, we view the game-based definitions (a central contribution of our work) as a reasonable middle ground between security and efficiency.

1.2 Technical Overview

To put our work into context, we give a brief overview of A^2L [53] recast as a synchronization puzzle (a notion first introduced in [31]), and discuss how it can be used as a coin mixing protocol. We then outline the vulnerabilities in A^2L and discuss how to fix them using the tools that we develop in this work.

Synchronization Puzzles. A synchronization puzzle protocol is a protocol between three parties: Alice, Bob, and Hub (refer to Figure 1 for a pictorial description). The synchronization puzzle begins with Hub and Bob executing a *puzzle promise* protocol (step 1) with respect to some message, m_{HB} such that Bob receives a puzzle τ that contains a signature s (at this point still hidden) on m_{HB}. Bob wishes to solve the puzzle and obtain the embedded signature. To do this, he sends the puzzle τ privately to Alice (step 2), who executes a *puzzle solve* protocol (step 3) with Hub with respect to some message m_{AH} such that, at the end of the protocol, Alice obtains the signature s, whereas Hub obtains a signature s' on m_{AH}. Alice then sends the signature s privately to Bob (step 4). Such a protocol must satisfy the following properties.

Blindness: The puzzle solve protocol does not leak any information to Hub about τ, and Hub *blindly* helps solve the puzzle. This ensures that Hub cannot link puzzles across interactions.

Unlockability: If step 3 is successfully completed, then the secret s must be a valid secret for Bob's puzzle τ. This guarantees that Hub cannot learn a signature on m_{AH}, without at the same time revealing a signature on m_{HB}.

Unforgeability: Bob cannot output a valid signature on m_{HB} before Alice interacts with the Hub.

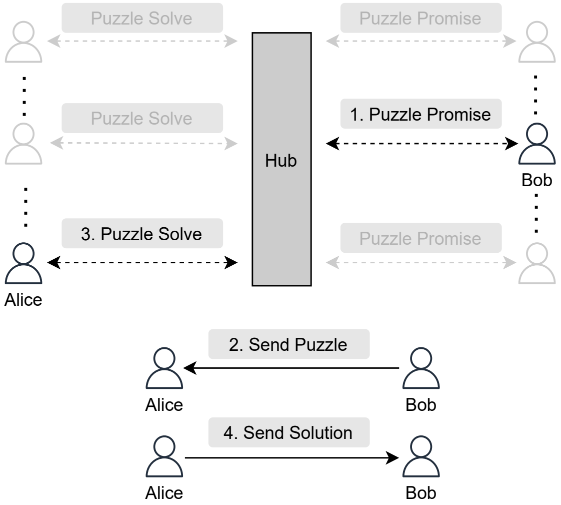

Figure 1: Protocol flow of the synchronization puzzle, the underlying cryptographic mechanism of Tumblebit and A^{2}L. Our approach in Blind Conditional Signatures follows a similar execution. Dotted double-edged arrows indicate 2-party protocols. Solid arrows indicate secure point-to-point communication.

Towards a Coin Mixing Service. As shown in [31, 53], the synchronization puzzle is the cryptographic core of a coin mixing service. First, Alice and Bob define the messages

$$m_{AH} : (A \xrightarrow{v} H) \text{ and } m_{HB} : (H \xrightarrow{v} B)$$

where $(U_i \xrightarrow{v} U_j)$ denotes a cryptocurrency payment (e.g., on-chain transaction or a payment over payment channels) that transfers v coins from U_i to U_j. Second, Alice and Bob run the synchronization puzzle protocol with Hub to synchronize the two aforementioned transfers. Here, the signatures s and s' are the ones required to validate the transactions defined by m_{AH} and m_{HB}. The anonymity of mixing follows from the fact that multiple pairs of users are executing the synchronization puzzle simultaneously with Hub, and Hub cannot link its interaction on the left to the corresponding interaction on the right. Throughout the rest of this work, we mainly focus on the synchronization puzzle as a cryptographic primitive. The application of a coin mixing protocol follows as prescribed in prior works [31, 53].

The A^{2}L System. In A^{2}L, the *blindness* property is achieved by making use of a re-randomizable linearly homomorphic (CPA-secure) encryption. The puzzle τ contains a ciphertext $c \leftarrow \mathsf{Enc}(\mathsf{ek}_H, s)$ encrypting the signature s under the encryption key ek_H of Hub. During the puzzle solve step, Alice first re-randomizes the ciphertext (and the underlying plaintext)

$$c \xrightarrow{r} c' = \mathsf{Enc}(\mathsf{ek}_H, s + r)$$

with a random scalar r. Hub then decrypts c' to obtain $s + r$, which in turn reveals a signature s' on m_{AH}.[1] Alice can then strip off the re-randomization factor r and send s to Bob later in step 4. In the analysis, it is argued that the CPA-security of the encryption scheme ensures unforgeability, whereas the re-randomization process guarantees blindness. Unfortunately, we show in this work that this claim is flawed.

Counterexamples. We observe that the encryption scheme is only CPA-secure, and the Hub is offering a decryption oracle in disguise. In these settings, the right notion of security is the stronger CCA-security, which accounts exactly for this scenario. However, CCA-security is at odds with blindness, since we require the scheme to be (i) linearly homomorphic and (ii) publicly re-randomizable.[2] We then substantiate this concern by showing two counterexamples. Specifically, we show that there exist two encryption schemes that satisfy the prerequisites spelled out by A^{2}L, but enable two concrete attacks against the protocol. Depending on the scheme, we can launch one of the following attacks:

- A key recovery attack that completely recovers the long-term secret key of the hub, i.e., the decryption key dk_H.
- A one-more signature attack that allows one to obtain $n + 1$ signatures on transactions from Hub to Bob, while only revealing n signatures on transactions from Alice to Hub. Effectively, this allows one to steal coins from the hub.

We stress that both these schemes are specifically crafted to make the protocol fail: their purpose is to highlight a gap in the security model of A^{2}L. As such, they do not imply that A^{2}L as implemented is insecure, although we cannot prove it secure either. For a detailed description of the attacks, we refer the reader to Section 3.2.

Can We Fix This? In light of our attacks, the natural question is whether we can establish formally rigorous security guarantees for the (appropriately patched) A^{2}L system. While it seems unlikely that A^{2}L can achieve UC-security (more discussion on this later), we investigate whether it satisfies some weaker, but still meaningful, notion of security. Our main observation here is that a weak notion of CCA-security for encryption schemes suffices to provide formal guarantees for A^{2}L. This notion, which we refer to as *one-more CCA-security*, (roughly) states that it is hard to recover the plaintexts of n ciphertexts while querying a decryption oracle at most $n - 1$ times. Importantly, this notion is, in principle, not in conflict with the homomorphism/re-randomization requirements, contrary to standard CCA-security.

Towards establishing a formal analysis of A^{2}L, we introduce the notion of blind conditional signatures (BCS) as the cryptographic cornerstone of a synchronization puzzle. We propose game-based definitions (Section 4.1) similar in spirit to the well-established security definitions of regular blind signatures [19, 48]. We then prove that A^{2}L$^+$, our appropriately modified version of A^{2}L, satisfies these definitions (Section 4.2). Our analysis comes with an important caveat: we analyze the security of our scheme in the *linear-only encryption model*. This is a model introduced by Groth [29] that only models adversaries that are restricted to perform "legal" operations on ciphertexts, similarly to the generic/algebraic group

[1] This is achieved via the notion of *adaptor signatures*, but for the sake of this overview we ignore the exact details of this aspect.

[2] It is well known that no encryption scheme that satisfies either of these properties can be CCA-secure.

model. While this is far from a complete analysis, it increases our confidence in the security of the system. [3]

UC-Security. The next question that we set out to answer is whether we can construct a synchronization puzzle that satisfies the strong notion of UC-security. We do not know how to prove that A^{2}L (or A^{2}L$^+$) is secure under composition, which is why we prove A^{2}L$^+$ secure only in the game-based setting. The technical difficulty in proving UC-security is that blindness is unconditional, and we lack a "trapdoor mechanism" that allows the simulator to link adversarial sessions during simulation in the security analysis; the proof of UC-security in [53] is flawed due to this same reason. Thus, in Section 5.2 we develop a different protocol (called A^{2}L$^{\text{UC}}$) that we can prove UC-secure in the standard model. The scheme relies on standard general-purpose cryptographic tools, such as 2PC, and incurs a significant increase in computation costs. We stress that we view this scheme as a proof-of-concept, and leave further improvements for practical efficiency as an open problem. We hope that the scheme will shed some light on the barriers that need to be overcome in order to construct a practically efficient UC-secure synchronization puzzle.

1.3 Related Work

We recall some relevant related work in the literature.

Unlinkable Transactions. CoinJoin [1], Coinshuffle [45–47], and Möbius [38] are coin mixing protocols that rely on interested users coming together and making an on-chain transactions to mix their coins. These proposals suffer from the bootstrapping problem (users having to find other interested users for the mix) in addition to requiring custom scripting language support from the underlying currency and completing the mix with on-chain transactions. Perun [23] and mixEth [49] are mixing solutions that rely on Ethereum smart contracts to resolve contentions among users. An alternate design choice is to incorporate coin unlinkability natively in the currency. Monero [35] and Zcash [10] are the two most popular examples of currencies that allow for unlinkable transactions without any special coin mixing protocol. This is enabled by complex on-chain cryptographic mechanisms that are not supported in other currencies.

RCCA Security. A security notion related to one-more CCA is that of re-randomizable *Replayable CCA (RCCA)* encryption scheme [43]. The notion guarantees security even if the adversary has access to a decryption oracle, but only for ciphertexts that do not decrypt to the challenge messages. This is slightly different from what we require in our setting, since in our application the adversary will always query the oracle on encryption of new (non-challenge) messages (because of the plaintext re-randomization). This makes it challenging to leverage the guarantees provided by this notion in our analysis.

2 PRELIMINARIES

We denote by $n \in \mathbb{N}$ the security parameter and by $x \leftarrow \mathcal{A}(\mathsf{in}; r)$ the output of the algorithm $\mathcal{A}$ on input in using $r \leftarrow_{\$} \{0,1\}^*$ as its randomness. We often omit this randomness and only mention it explicitly when required. We say that an algorithm is (non-uniform) PPT if it runs in probabilistic polynomial time. We say that a function is *negligible* if it vanishes faster than any polynomial.

Digital Signature. A digital signature scheme $\Pi_{\mathsf{DS}} := (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$ has a key generation algorithm $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^n)$ that outputs a verification-signing key pair. The owner of the signing key sk can compute signatures on a message m by running $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$, which can be publicly verified using the corresponding verification key vk by running $\mathsf{Vf}(\mathsf{vk}, m, \sigma)$. We require that the digital signature scheme satisfies the standard notion of strong existential unforgeability [28].

Hard Relations. We recall the notion of a hard relation R with statement/witness pairs (Y, y). We denote by $\mathcal{L}_R$ the associated language defined as $\mathcal{L}_R := \{Y \mid \exists y,\ (Y, y) \in R\}$. The relation is called a hard relation if the following holds: (i) There exists a PPT sampling algorithm $\mathsf{GenR}(1^n)$ that outputs a statement/witness pair $(Y, y) \in R$; (ii) The relation is poly-time decidable; (iii) For all PPT adversaries $\mathcal{A}$ the probability of $\mathcal{A}$ on input Y outputting a witness y is negligible. In this work we use the discrete log language $\mathcal{L}_{\mathsf{DL}}$ defined with respect to a group $\mathbb{G}$ with generator g and order p. The language is defined as $\mathcal{L}_{\mathsf{DL}} := \{Y \mid \exists y \in \mathbb{Z}_p,\ Y = g^y\}$ with corresponding hard relation R_{DL}.

Adaptor Signatures. Adaptor signatures [3] let users generate a pre-signature on a message m which by itself is not a valid signature, but can later be adapted into a valid signature using knowledge of some secret value. More precisely, an adaptor signature scheme $\Pi_{\mathsf{ADP}} := (\mathsf{KGen}, \mathsf{PreSig}, \mathsf{PreVf}, \mathsf{Adapt}, \mathsf{Vf}, \mathsf{Ext})$ is defined with respect to a signature scheme Π_{DS} and a hard relation R. The key generation algorithm is the same as in Π_{DS} and outputs a key pair $(\mathsf{vk}, \mathsf{sk})$. The pre-signing algorithm $\mathsf{PreSig}(\mathsf{sk}, m, Y)$ returns a pre-signature $\tilde{\sigma}$ (we sometimes also refer to this as a partial signature). The pre-signature verification algorithm $\mathsf{PreVf}(\mathsf{vk}, m, Y, \tilde{\sigma})$ verifies if the pre-signature $\tilde{\sigma}$ is correctly generated. The adapt algorithm $\mathsf{Adapt}(\tilde{\sigma}, y)$ transforms a pre-signature $\tilde{\sigma}$ into a valid signature σ given the witness y for the instance Y of the language $\mathcal{L}_R$. The verification algorithm Vf is the same as in Π_{DS}. Finally, we have the extract algorithm $\mathsf{Ext}(\tilde{\sigma}, \sigma, Y)$ which, given a pre-signature $\tilde{\sigma}$, a signature σ, and an instance Y, outputs the witness y for Y. This can be formalized as *pre-signature correctness*.

Definition 2.1 (Pre-signature Correctness). An adaptor signature scheme Π_{ADP} satisfies pre-signature correctness if for every $n \in \mathbb{N}$, every message $m \in \{0,1\}^*$, and every statement/witness pair $(Y, y) \in R$, the following holds:

$$\Pr\left[\begin{array}{c|l} \mathsf{PreVf}(\mathsf{vk}, m, Y, \tilde{\sigma}) = 1 & \\ \wedge & (\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KGen}(1^n) \\ \mathsf{Vf}(\mathsf{vk}, m, \sigma) = 1 & \tilde{\sigma} \leftarrow \mathsf{PreSig}(\mathsf{sk}, m, Y) \\ \wedge & \sigma := \mathsf{Adapt}(\tilde{\sigma}, y) \\ (Y, y') \in R & y' := \mathsf{Ext}(\sigma, \tilde{\sigma}, Y) \end{array}\right] = 1.$$

In terms of security, we want standard unforgeability even when the adversary is given access to pre-signatures with respect to the signing key sk.

[3] We resort to the LOE model because of the seemingly inherent conflict between linear homomorphism and CCA-like security, both of which are needed for our application (in our setting, the adversary has access to something akin to a decryption oracle). Indeed, even proving that ElGamal encryption is CCA1-secure in the standard model is a long-standing open problem, and we believe that the A^{2}L approach would inherently hit this barrier without some additional assumption.

```
$\mathsf{aSigForge}_{\mathcal{A},\Pi_{\mathsf{ADP}}}(n)$
  $Q := \emptyset$
  $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KGen}(1^n)$
  $m \leftarrow \mathcal{A}^{\mathcal{O}_{\mathrm{S}}(\cdot), \mathcal{O}_{\mathrm{pS}}(\cdot,\cdot)}(\mathsf{vk})$
  $(Y, y) \leftarrow \mathsf{GenR}(1^n)$
  $\tilde{\sigma} \leftarrow \mathsf{PreSig}(\mathsf{sk}, m, Y)$
  $\sigma \leftarrow \mathcal{A}^{\mathcal{O}_{\mathrm{S}}(\cdot), \mathcal{O}_{\mathrm{pS}}(\cdot,\cdot)}(\tilde{\sigma}, Y)$
  return $(m \notin Q \wedge \mathsf{Vf}(\mathsf{vk}, m, \sigma))$

$\mathcal{O}_{\mathrm{S}}(m)$
  $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$
  $Q := Q \cup \{m\}$
  return $\sigma$

$\mathcal{O}_{\mathrm{pS}}(m, Y)$
  $\tilde{\sigma} \leftarrow \mathsf{PreSig}(\mathsf{sk}, m, Y)$
  $Q := Q \cup \{m\}$
  return $\tilde{\sigma}$
```

Figure 2: Unforgeability experiment of adaptor signatures

Definition 2.2 (Unforgeability). An adaptor signature scheme Π_{ADP} is aEUF-CMA secure if for every PPT adversary $\mathcal{A}$ there exists a negligible function negl such that

$$\Pr\left[\mathsf{aSigForge}_{\mathcal{A},\Pi_{\mathsf{ADP}}}(n) = 1\right] \leq \mathsf{negl}(n),$$

where the experiment $\mathsf{aSigForge}_{\mathcal{A},\Pi_{\mathsf{ADP}}}$ is defined as in Figure 2.

We also require that, given a pre-signature and a witness for the instance, one can always adapt the pre-signature into a valid signature (*pre-signature adaptability*).

Definition 2.3 (Pre-signature Adaptability). An adaptor signature scheme Π_{ADP} satisfies pre-signature adaptability if for any $n \in \mathbb{N}$, any message $m \in \{0,1\}^*$, any statement/witness pair $(Y, y) \in R$, any key pair $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KGen}(1^n)$, and any pre-signature $\tilde{\sigma} \leftarrow \{0,1\}^*$ with $\mathsf{PreVf}(\mathsf{vk}, m, Y, \tilde{\sigma}) = 1$, we have:

$$\Pr[\mathsf{Vf}(\mathsf{vk}, m, \mathsf{Adapt}(\tilde{\sigma}, y)) = 1] = 1.$$

Finally, we require that, given a valid pre-signature and a signature with respect to the same instance, one can efficiently extract the corresponding witness (*witness extractability*).

Definition 2.4 (Witness Extractability). An adaptor signature scheme Π_{ADP} is witness extractable if for every PPT adversary $\mathcal{A}$, there exists a negligible function negl such that

$$\Pr\left[\mathsf{aWitExt}_{\mathcal{A},\Pi_{\mathsf{ADP}}}(n) = 1\right] \leq \mathsf{negl}(n),$$

where the experiment $\mathsf{aWitExt}_{\mathcal{A},\Pi_{\mathsf{ADP}}}$ is defined as in Figure 3.

```
$\mathsf{aWitExt}_{\mathcal{A},\Pi_{\mathsf{ADP}}}(n)$
  $Q := \emptyset$
  $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KGen}(1^n)$
  $(m, Y) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathrm{S}}(\cdot), \mathcal{O}_{\mathrm{pS}}(\cdot,\cdot)}(\mathsf{vk})$
  $\tilde{\sigma} \leftarrow \mathsf{PreSig}(\mathsf{sk}, m, Y)$
  $\sigma \leftarrow \mathcal{A}^{\mathcal{O}_{\mathrm{S}}(\cdot), \mathcal{O}_{\mathrm{pS}}(\cdot,\cdot)}(\tilde{\sigma})$
  $y' := \mathsf{Ext}(\sigma, \tilde{\sigma}, Y)$
  return $(m \notin Q \wedge (Y, y') \notin R$
    $\wedge\ \mathsf{Vf}(\mathsf{vk}, m, \sigma))$

$\mathcal{O}_{\mathrm{S}}(m)$
  $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$
  $Q := Q \cup \{m\}$
  return $\sigma$

$\mathcal{O}_{\mathrm{pS}}(m, Y)$
  $\tilde{\sigma} \leftarrow \mathsf{PreSig}(\mathsf{sk}, m, Y)$
  $Q := Q \cup \{m\}$
  return $\tilde{\sigma}$
```

Figure 3: Witness extractability experiment for adaptor signatures

```
$\mathcal{O}^{\mathsf{Gen}}(i)$
  $\mathsf{ek}_i \leftarrow\$ \{0,1\}^n$
  Enter $(i, \mathsf{ek}_i)$ into table $K$
  return $\mathsf{ek}_i$

$\mathcal{O}^{\mathsf{Enc}}(\mathsf{ek}_i, m)$
  $c_j \leftarrow\$ \{0,1\}^n$
  Enter $(m, c_j)$ into table $M_i$
  return $c_j$

$\mathcal{O}^{\mathsf{Dec}}(\mathsf{ek}_i, c)$
  if $(\cdot, c) \notin M_i$ then return $\perp$
  else
    Look up $m$ corresponding to $c$ in $M_i$
    return $m$

$\mathcal{O}^{\mathsf{Add}}(\mathsf{ek}_i, c_0, c_1)$
  Look up $m_0, m_1$ corresponding to $c_0, c_1$ in table $M_i$
  $\tilde{c} \leftarrow\$ \{0,1\}^n$
  Enter $(m_0 + m_1, \tilde{c})$ into table $M_i$
  return $\tilde{c}$
```

Figure 4: Linear-only encryption oracles

Combining the three properties described above, we can define a secure adaptor signature scheme as follows.

Definition 2.5 (Secure Adaptor Signature Scheme). An adaptor signature scheme Π_{ADP} is secure if it is aEUF-CMA secure, pre-signature adaptable, and witness extractable.

Linear-Only Homomorphic Encryption. A public-key encryption scheme $\Pi_{\mathsf{E}} := (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ allows one to generate a key pair $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{KGen}(1^n)$ that allows anyone to encrypt messages as $c \leftarrow \mathsf{Enc}(\mathsf{ek}, m)$ and allows only the owner of the decryption key dk to decrypt ciphertexts as $m \leftarrow \mathsf{Dec}(\mathsf{dk}, c)$. We require that Π_{E} satisfies perfect correctness and the standard notion of CPA-security [27]. We say that an encryption scheme is *linearly homomorphic* if there exists some efficiently computable operation $\circ$ such that $\mathsf{Enc}(\mathsf{ek}, m_0) \circ \mathsf{Enc}(\mathsf{ek}, m_1) \in \mathsf{Enc}(\mathsf{ek}, m_0 + m_1)$, where addition is defined over $\mathbb{Z}_p$. The α-fold application of $\circ$ is denoted by $\mathsf{Enc}(\mathsf{ek}, m)^\alpha$.

Linear-only encryption (LOE) is an idealized model introduced by Groth [29] as "generic homomorphic cryptosystem". Here, homomorphic encryption is modeled by giving access to oracles instead of their corresponding algorithms. A formal description of the oracles is given in Figure 4. We note that although we do not model such an algorithm explicitly, this model allows for (perfect) ciphertext re-randomization by homomorphically adding 0 to the desired ciphertext.

Non-Interactive Zero-Knowledge. Let $\mathcal{R} : \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}$ be an NP-witness-relation with corresponding NP-language $\mathcal{L} := \{x \mid \exists w \text{ s.t. } \mathcal{R}(x, w) = 1\}$. A non-interactive zero-knowledge proof system $\mathsf{NIZK} := (\mathsf{Setup}, \mathsf{P}, \mathsf{V})$ for the relation $\mathcal{R}$ is initialized with a setup algorithm $\mathsf{Setup}(1^n)$ that, on input the security parameter, outputs a common reference string crs and a trapdoor td. A prover can show the validity of a statement x with a witness w by invoking $\mathsf{P}(\mathsf{crs}, x, w)$, which outputs a proof π. The proof π can be efficiently checked by the verification algorithm $\mathsf{V}(\mathsf{crs}, x, \pi)$. We require a NIZK system to be (1) *zero-knowledge*, i.e., there exists a simulator $\pi \leftarrow \mathsf{Sim}(\mathsf{td}, x)$ that computes valid proofs without the knowledge of the witness, (2) *sound*, i.e., it is infeasible for an adversary to output a valid proof for a statement $x \notin \mathcal{L}$, and (3) *UC-secure*, i.e., one can efficiently extract from the proofs computed by the adversary a valid witness (with the knowledge of the trapdoor td), even in the presence of simulated proofs. For formal security definitions, we refer the reader to [15, 22].

One-More DL. We recall the one-more discrete logarithm (OMDL) assumption [7, 9].

Definition 2.6 (One-More Discrete Logarithm (OMDL) Assumption). Let $\mathbb{G}$ be a uniformly sampled cyclic group of prime order p and g a random generator of $\mathbb{G}$. The one-more discrete logarithm (OMDL) assumption states that for all $n \in \mathbb{N}$ there exists a negligible function $\mathsf{negl}(n)$ such that for all PPT adversaries $\mathcal{A}$ making at most $q = \mathsf{poly}(n)$ queries to $\mathsf{DL}(\cdot)$, the following holds:

$$\Pr\left[\ \forall i : x_i = r_i \ \middle|\ \begin{array}{c} r_1, \ldots, r_{q+1} \leftarrow\$\ \mathbb{Z}_p \\ \forall i \in [q+1], h_i \leftarrow g^{r_i} \\ \{x_i\}_{i \in [q+1]} \leftarrow \mathcal{A}^{\mathsf{DL}(\cdot)}(h_1, \ldots, h_{q+1}) \end{array} \right] \leq \mathsf{negl}(n),$$

where the $\mathsf{DL}(\cdot)$ oracle takes as input an element $h \in \mathbb{G}$ and returns x such that $h = g^x$.

3 COUNTEREXAMPLES OF $\mathrm{A}^2\mathrm{L}$

In the following, we recall the $\mathrm{A}^2\mathrm{L}$ system and present two counterexamples to their main theorem.

3.1 Description of $\mathrm{A}^2\mathrm{L}$

$\mathrm{A}^2\mathrm{L}$ is defined over the following cryptographic schemes:

- A digital signature scheme Π_{DS}, a hard relation R_{DL} for a group $(\mathbb{G}, g, p)$ with generator g and prime order p, and the corresponding adaptor signature scheme Π_{ADP}.
- A linearly homomorphic re-randomizable CPA-secure encryption scheme Π_{E}.[4]
- A NIZK proof system $\Pi_{\mathsf{NIZK}} := (\mathsf{Setup}, \mathsf{P}, \mathsf{V})$ for the language

$$\mathcal{L} := \{(\mathsf{ek}, Y, c) \mid \exists s \text{ s.t. } c \leftarrow \Pi_{\mathsf{E}}.\mathsf{Enc}(\mathsf{ek}, s) \wedge Y = g^s\}.$$

The protocol has three parties: Alice, Bob, and Hub. At the beginning of the system, Hub runs the setup (as described in Figure 9) to generate its keys, which are the keys for the (CPA-secure) encryption scheme Π_{E}. The protocol then consists of a promise phase and a solving phase.

Puzzle Promise. In the promise phase (Figure 12), Hub generates a pre-signature $\tilde{\sigma}_{HB}^{H}$ on a common message m_{HB} with respect to a uniformly sampled instance $Y := g^s$. Hub also encrypts the witness s in the ciphertext $c \leftarrow \Pi_{\mathsf{E}}.\mathsf{Enc}(\mathsf{ek}_H, s)$ under its own encryption key ek_H. Hub gives Bob the tuple $(Y, c, \pi, \tilde{\sigma}_{HB}^{H})$, where π is a NIZK proof that certifies the ciphertext c encrypts s. Bob verifies that the NIZK proof and the pre-signature are indeed valid. If so, he chooses a random $r \leftarrow\$\ \mathbb{Z}_q$ and re-randomizes the instance Y to $Y' := Y \cdot g^r$ and also re-randomizes the ciphertext c as $c' \leftarrow \Pi_{\mathsf{E}}.\mathsf{Rand}(c, r)$. The puzzle is set to $\tau := (r, m_{HB}, \tilde{\sigma}_{HB}^{H}, (Y, c), (Y', c'))$.

Puzzle Solve. Bob sends the puzzle τ privately to Alice, who now executes the puzzle solve protocol with Hub (Figure 13). Alice samples a random r' and further re-randomizes the instance Y' as $Y'' := Y' \cdot g^{r'}$ and the ciphertext c' as $c'' \leftarrow \Pi_{\mathsf{E}}.\mathsf{Rand}(c', r')$. She then generates a pre-signature $\tilde{\sigma}_{AH}^{A}$ on a common message m_{AH} with respect to the instance Y''. She sends the tuple $(Y'', c'', \tilde{\sigma}_{AH}^{A})$ to Hub, who decrypts c'' using the decryption key dk_H to obtain s''. Hub then adapts the pre-signature $\tilde{\sigma}_{AH}^{A}$ to σ_{AH}^{A} using s'' and ensures its validity. It then sends the signature σ_{AH}^{A} to Alice, who extracts the witness for Y'' as $s'' \leftarrow \Pi_{\mathsf{ADP}}.\mathsf{Ext}(\tilde{\sigma}_{AH}^{A}, \sigma_{AH}^{A}, Y'')$. Alice removes the re-randomization factor to obtain the solution $s' := s'' - r'$ for the instance Y'. Alice finally sends s' privately to Bob, who opens the puzzle τ by computing the witness $s := s' - r$ and adapting the pre-signature $\tilde{\sigma}_{HB}^{H}$ (given by Hub in the promise phase) to the signature σ_{HB}^{H}.

3.2 Counterexamples

Next, we describe two cryptographic instantiations of $\mathrm{A}^2\mathrm{L}$ that satisfy the formal definitions, yet enable two attacks. For the purpose of these attacks, it suffices to keep in mind that Hub offers the sender party (Alice) access to the following oracle, which we refer to as $\mathcal{O}^{\mathsf{A^2L}}_{\mathsf{dk}, \Pi_{\mathsf{E}}, \Pi_{\mathsf{ADP}}}$. On input a verification key vk, a message m, a group element h, a ciphertext c, and a partial signature $\tilde{\sigma}$, the oracle behaves as follows:

- Compute $\tilde{x} \leftarrow \Pi_{\mathsf{E}}.\mathsf{Dec}(\mathsf{dk}, c)$.
- Compute $\sigma' \leftarrow \Pi_{\mathsf{ADP}}.\mathsf{Adapt}(\tilde{\sigma}, \tilde{x})$.
- If $\Pi_{\mathsf{ADP}}.\mathsf{Vrfy}(\mathsf{vk}, m, \sigma') = 1$, return σ'.
- Else return $\perp$.

Note that returning σ' implicitly reveals $\tilde{x}$, since $\Pi_{\mathsf{ADP}}.\mathsf{Ext}(\tilde{\sigma}, \sigma', h) = \tilde{x}$. It is also useful to observe that providing a valid pre-signature to the $\mathrm{A}^2\mathrm{L}$ oracle is trivial for an adversary: generating a pre-signature that is valid when adapted with a value x requires only knowledge of the party's own signing key and of a value $h = g^x$. The leakage offered by this oracle (and indeed the existence of this leakage) is not addressed in $\mathrm{A}^2\mathrm{L}$'s proof of security.

Key Recovery Attack. In our first attack, we completely recover the decryption key dk of the hub by simply querying the oracle $\mathcal{O}^{\mathsf{A^2L}}_{\mathsf{sk}, \Pi_{\mathsf{E}}, \Pi_{\mathsf{ADP}}}$ n times. For this attack, we assume that the encryption scheme Π_{E} is (in addition to being re-randomizable and CPA-secure as required by $\mathrm{A}^2\mathrm{L}$):

- Linearly homomorphic over $\mathbb{Z}_p$.
- Circular secure for bit encryption, i.e., the scheme is CPA-secure even given the bitwise encryption of the decryption key $\mathsf{Enc}(\mathsf{ek}, \mathsf{dk}_1), \ldots, \mathsf{Enc}(\mathsf{ek}, \mathsf{dk}_n)$.
- The above-mentioned ciphertexts $(c_1, \ldots, c_n) := (\mathsf{Enc}(\mathsf{ek}, \mathsf{dk}_1), \ldots, \mathsf{Enc}(\mathsf{ek}, \mathsf{dk}_n))$ are included in the encryption key ek.

Such schemes can be constructed from a variety of standard assumptions [13]. It is easy to see that these additional requirements do not contradict the initial prerequisites of the scheme.

The attack is shown in Algorithm 1. Note that, for a signing key pair in the i-th iteration, if the $\mathcal{O}^{\mathsf{A^2L}}$ oracle returns $y \neq \perp$, this means that in the coin mixing layer, the Hub has obtained a valid y and thus

[4] Technically, [53] uses a different abstraction called "re-randomizable puzzle". However, it is not hard to see that a re-randomizable linearly homomorphic encryption scheme satisfies this notion. For completeness, we show this in Appendix A.

Algorithm 1 Key Recovery Attack

```
Input: Hub's ek along with the cipheretexts $(c_1, \ldots, c_n)$
Initialize key guess $\mathsf{dk}' := 0^n$
for $i \in 1 \ldots n$ do
    Sample $x \leftarrow\$ \mathbb{Z}_p$ and compute $h := g^x$
    Sample a fresh signing key $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^n)$
    Set $c'_i := \Pi_{\mathsf{E}}.\mathsf{Enc}(\mathsf{ek}, x) \circ c_i = \Pi_{\mathsf{E}}.\mathsf{Enc}(\mathsf{ek}, x + \mathsf{dk}_i)$
    Compute $\tilde{\sigma}_i \leftarrow \Pi_{\mathsf{ADP}}.\mathsf{PreSig}(\mathsf{sk}, m, h)$
    Query $y \leftarrow \mathcal{O}^{\mathsf{A^2L}}_{\mathsf{dk},\Pi_{\mathsf{E}},\Pi_{\mathsf{ADP}}}(\mathsf{vk}, m, h, c'_i, \tilde{\sigma}_i)$
    If $y = \perp$ set $\mathsf{dk}'_i := 1$
end for
return $\mathsf{dk}'$
```

Algorithm 2 One-More Signature Attack

```
Input: Bob's ciphertexts $(c_1, \ldots, c_{q+1})$ and group elements $(h_1, \ldots, h_{q+1})$,
    where $c_j = \Pi_{\mathsf{E}}.\mathsf{Enc}(\mathsf{ek}, x_j)$ and $h_j := g^{x_j}$, and Hub's ek
Initialize guess $x'_1 := 0^n$ and a counter $i := 1$
for $i = 1 \ldots n$ do
    Sample a fresh signing key $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^n)$
    Compute $c'_1 \leftarrow \Pi_{\mathsf{E}}.\mathsf{CFlip}(\mathsf{ek}, i, c_1)$
    Sample $(r_1^{(i)}, \ldots, r_{q+1}^{(i)}) \leftarrow\$ \mathbb{Z}_p^{q+1}$
    Compute $c' := (c'_1)^{r_1^{(i)}} \circ (c_2)^{r_2^{(i)}} \cdots \circ (c_{q+1})^{r_{q+1}^{(i)}}$
    Compute $h' := \prod_{j=1}^{q+1} h_j^{r_j^{(i)}}$
    Sign $\tilde{\sigma} \leftarrow \Pi_{\mathsf{ADP}}.\mathsf{PreSig}(\mathsf{sk}, m, h')$
    Query $y_i \leftarrow \mathcal{O}^{\mathsf{A^2L}}_{\mathsf{dk},\Pi_{\mathsf{E}},\Pi_{\mathsf{ADP}}}(\mathsf{vk}, m, h', c', \tilde{\sigma})$
    If $y_i = \perp$ set $x'_{1,i} := 1$
end for
Continue querying (without updating $x'_1$) until $q$ non-$\perp$ queries have
    been made
For all $i$ corresponding to a non-$\perp$ query, set $E_i$ to be the equation
    $y_i - r_1^{(i)} x'_1 = r_2^{(i)} x'_2 + \ldots + r_{q+1}^{(i)} x'_{q+1}$
Solve $(E_1, \ldots, E_q)$ for $(x'_2, \ldots, x'_{q+1})$
return $(x'_1, x'_2, \ldots, x'_{q+1})$
```

obtains Alice's (adversary's) signature on a transaction. Due to one-time use of keys in this (cryptocurrency) layer, the attacker therefore cannot reuse the same signing key pair in another iteration for a different message (transaction). Therefore, it is necessary that the attacker (Alice) sample n signing keys to account for every iteration being a non-$\perp$ query to $\mathcal{O}^{\mathsf{A^2L}}$. This is realized in the real world by the attacker having n different sessions (of coin mixing), one for each vk_i, with Hub.

Observe that the response of the oracle is $\perp$ if and only if $\mathsf{dk}_i = 1$, since $h = g^x \neq g^{x+1}$. On the other hand, if $\mathsf{dk}_i = 0$, then the oracle always returns a valid adapted signature σ'. Thus, the attack succeeds with probability 1.

One-More Signature Attack. We present a different attack, where we impose different assumptions on the encryption scheme Π_{E}. We discuss later in the section why these assumptions do not contradict the pre-requisites of the A^{2}L scheme. Specifically, in addition to A^{2}L's requirement that the scheme is perfectly re-randomizable and CPA-secure, we assume that it is:

- Linearly homomorphic over $\mathbb{Z}_p$.
- Supports homomorphic evaluation of the *conditional bit flip* (CFlip) function, defined as

$$\Pi_{\mathsf{E}}.\mathsf{CFlip}(\mathsf{ek}, i, \mathsf{Enc}(\mathsf{ek}, x)) := \mathsf{Enc}(\mathsf{ek}, y)$$
$$\text{where } \begin{cases} y = x & \text{if } x_i = 0 \\ y = x \oplus e_i & \text{if } x_i = 1 \end{cases}$$

  and e_i is the i-th unit vector.

The objective of the attack is to steal coins from the hub in the coin mixing protocol. Specifically, at the A^{2}L level, the attacker will solve $q + 1$ puzzles by querying the puzzle solver interface successfully only q times. Note that we do not count unsuccessful (i.e., the oracle returns $\perp$) queries, since those non-accepting queries do not correspond to any payment from Alice's side.

The attack is shown in Algorithm 2. We assume (for convenience) that $q \geq n$ and that $\mathbb{Z}_p \leq 2^n$ and therefore $x_j \in \{0, 1\}^n$. Observe that the attack makes at most q successful queries to the oracle, so all we need to show is that the success probability is high enough. First, we argue that the attack recovers the correct $x'_1 = x_1$ with probability 1. If the i-th bit $x_{1,i} = 0$, then the CFlip operation does not alter the content of the ciphertext and therefore

$$c' = \mathsf{Enc}\left(\mathsf{ek}, \sum_{j=1}^{q+1} r_j^{(i)} \cdot x_j\right) \text{ and } h' = \prod_{j=1}^{q+1} h_j^{r_j^{(i)}} = g^{\sum_{j=1}^{q+1} r_j^{(i)} \cdot x_j}$$

so the oracle always returns a non-$\perp$ response. On the other hand, if $x_{1,i} = 1$, then the above equality does not hold and therefore $\mathcal{O}^{\mathsf{A^2L}}_{\mathsf{sk},\Pi_{\mathsf{E}},\Pi_{\mathsf{ADP}}}$ always returns $\perp$.

This querying strategy is repeated for every bit of x'_1 and continued on x_2, etc., until q non-$\perp$ queries have been made. Because $q \geq n$, the attacker will have learned all n bits of x'_1 by this point. Thus, the set of equations $(E_1, \ldots, E_q)$ has exactly q unknowns. Since the coefficients are uniformly chosen, the equations are, with all but negligible probability, linearly independent. Since $\mathbb{Z}_p$ is a field, the solution is uniquely determined and can be found efficiently via Gaussian elimination.

N-More Signatures. The described attack is in fact even stronger than shown. Using this method, an attacker $\mathcal{A}$ can use q queries, where $\lfloor q \rfloor = Nn$, to recover $N + q$ plaintexts. $\mathcal{A}$ does this by using Nn queries to recover the first N plaintexts $x_1, \ldots, x_N$ and Nn equations as described previously (once it has flipped all n bits in x_1, it starts flipping bits in x_2, and so on). Using its remaining queries, it obtains $q - Nn$ more equations (either by continuing to flip bits in further ciphertexts, which are however wasted, or by simply choosing new values r_i for the linear combinations) for a total of q equations. Using Gaussian elimination, it can recover the remaining q plaintexts $x_{N+1}, \ldots, x_{N+q}$. Taken with the plaintexts $x_1, \ldots, x_N$ that were recovered bit-by-bit, the attacker has learned $N + q$ plaintexts.

Instantiations. We now justify our additional assumptions on the encryption scheme Π_{E} by describing suitable instantiations that satisfy all the requirements. Clearly, if the scheme is fully-homomorphic [25] then it supports both linear functions over $\mathbb{Z}_p$ and conditional bit flips. However, we show that even a linear homomorphic encryption (over $\mathbb{Z}_p$) can suffice to mount our attack. Specifically, given a CPA-secure linearly homomorphic encryption scheme (KGen^*, Enc^*, Dec^*), we define a *bitwise* encryption scheme (KGen, Enc, Dec) as follows:

- $\mathsf{KGen}(1^n)$: Return the output of $\mathsf{KGen}^*(1^n)$.
- $\mathsf{Enc}(\mathsf{ek}, x)$: Parse x as $(x^{(1)}, \dots, x^{(n)})$ and return $(\mathsf{Enc}^*(\mathsf{ek}, x^{(1)}), \dots, \mathsf{Enc}^*(\mathsf{ek}, x^{(n)}))$.
- $\mathsf{Dec}(\mathsf{dk}, c)$: Parse c as $(c^{(1)}, \dots, c^{(n)})$ and return $\sum_{i=1}^{n} 2^{i-1} \cdot \mathsf{Dec}^*(\mathsf{dk}, c^{(i)})$.

It is easy to show that the new scheme is CPA-secure via a standard hybrid argument.

Next, we argue that one can efficiently implement the conditional bit flip operation (CFlip) over such ciphertexts. Given a ciphertext $c = (c^{(1)}, \dots, c^{(n)})$, we can conditionally flip the i-th bit by computing

$$(c^{(1)}, \dots, \underbrace{\mathsf{Enc}^*(\mathsf{ek}, 0)}_{i\text{-th ciphertext}}, \dots, c^{(n)}).$$

This is a correctly formed ciphertext, since the conditional bit flip always sets the i-th bit to 0 and leaves the other positions untouched.

Finally, we need to argue that the encryption scheme is still linearly homomorphic over $\mathbb{Z}_p$. Note that this does not follow immediately from the fact that $(\mathsf{KGen}^*, \mathsf{Enc}^*, \mathsf{Dec}^*)$ is linearly homomorphic, since the new encryption algorithm decomposes the inputs bitwise. Nevertheless, we show this indeed holds for the case of two ciphertexts $c = (c^{(1)}, \dots, c^{(n)})$ and $d = (d^{(1)}, \dots, d^{(n)})$ encrypting x and y, respectively. The general case follows analogously. To homomorphically compute $\alpha x + \beta y$, where $(\alpha, \beta) \in \mathbb{Z}_p^2$, we compute

$$\left(\left(\bigcirc_{i=1}^{n} (c^{(i)})^{2^{i-1}}\right)^{\alpha} \circ \left(\bigcirc_{i=1}^{n} (d^{(i)})^{2^{i-1}}\right)^{\beta}, \mathsf{Enc}^*(\mathsf{ek}, 0), \dots, \mathsf{Enc}^*(\mathsf{ek}, 0)\right).$$

A routine calculation shows that this ciphertext correctly decrypts to the desired result $\alpha x + \beta y$.

4 BLIND CONDITIONAL SIGNATURES

In the following we formally define and instantiate blind conditional signatures, the central cryptographic notion for coin mixing services. Our goal here is to give a simple and easy-to-understand formalization of a synchronization puzzle.

4.1 Definitions

A blind conditional signature (BCS) is executed among users Alice, Bob, and Hub. The interfaces and associated security properties are defined below.

Definition 4.1 (Blind Conditional Signature). A blind conditional signature $\Pi_{\mathsf{BCS}} := (\mathsf{Setup}, \mathsf{PPromise}, \mathsf{PSolver}, \mathsf{Open})$ is defined with respect to a signature scheme $\Pi_{\mathsf{DS}} := (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$ and consists of the following efficient algorithms.

- $(\tilde{\mathsf{ek}}, \tilde{\mathsf{dk}}) \leftarrow \mathsf{Setup}(1^n)$: The setup algorithm takes as input the security parameter 1^n and outputs a key pair $(\tilde{\mathsf{ek}}, \tilde{\mathsf{dk}})$.
- $(\bot, \{\tau, \bot\}) \leftarrow \mathsf{PPromise}\left\langle \begin{matrix} H\left(\tilde{\mathsf{dk}}, \mathsf{sk}^H, m_{HB}\right) \\ B\left(\tilde{\mathsf{ek}}, \mathsf{vk}^H, m_{HB}\right) \end{matrix} \right\rangle$: The puzzle promise algorithm is an interactive protocol between two users H (with inputs the decryption key $\tilde{\mathsf{dk}}$, the signing key sk^H, and a message m_{HB}) and B (with inputs the encryption key $\tilde{\mathsf{ek}}$, the verification key vk^H, and a message m_{HB}) and returns $\bot$ to H and either a puzzle τ or $\bot$ to B.
- $(\{(\sigma^*, s), \bot\}, \{\sigma^*, \bot\}) \leftarrow \mathsf{PSolver}\left\langle \begin{matrix} A\left(\mathsf{sk}^A, \tilde{\mathsf{ek}}, m_{AH}, \tau\right) \\ H\left(\tilde{\mathsf{dk}}, \mathsf{vk}^A, m_{AH}\right) \end{matrix} \right\rangle$: The puzzle solving algorithm is an interactive protocol between two users A (with inputs the signing key sk^A, the encryption key $\tilde{\mathsf{ek}}$, a message m_{AH}, and a puzzle τ) and H (with inputs the decryption key $\tilde{\mathsf{dk}}$, the verification key pk^A, and a message m_{AH}) and returns to both users either a signature σ^* (A additionally receives a secret s) or $\bot$.
- $\{\sigma, \bot\} \leftarrow \mathsf{Open}(\tau, s)$: The open algorithm takes as input a puzzle τ and a secret s and returns a signature σ or $\bot$.

Next, we define correctness.

Definition 4.2 (Correctness). A blind conditional signature Π_{BCS} is correct if for all $n \in \mathbb{N}$, all $(\tilde{\mathsf{ek}}, \tilde{\mathsf{dk}})$ in the support of $\mathsf{Setup}(1^n)$, all $(\mathsf{vk}^H, \mathsf{sk}^H)$ and $(\mathsf{vk}^A, \mathsf{sk}^A)$ in the support of $\Pi_{\mathsf{DS}}.\mathsf{KGen}(1^n)$, and all pairs of messages (m_{HB}, m_{AH}), it holds that

$$\Pr\left[\mathsf{Vf}(\mathsf{vk}^H, m_{HB}, \mathsf{Open}(\tau, s)) = 1\right] = 1$$

and

$$\Pr\left[\mathsf{Vf}(\mathsf{vk}^A, m_{AH}, \sigma^*) = 1\right] = 1$$

where

- $\tau \leftarrow \mathsf{PPromise}\left\langle \begin{matrix} H\left(\tilde{\mathsf{dk}}, \mathsf{sk}^H, m_{HB}\right) \\ B\left(\tilde{\mathsf{ek}}, \mathsf{vk}^H, m_{HB}\right) \end{matrix} \right\rangle$ and
- $((\sigma^*, s), \sigma^*) \leftarrow \mathsf{PSolver}\left\langle \begin{matrix} A\left(\mathsf{sk}^A, \tilde{\mathsf{ek}}, m_{AH}, \tau\right) \\ H\left(\tilde{\mathsf{dk}}, \mathsf{vk}^A, m_{AH}\right) \end{matrix} \right\rangle$.

We now present the security guarantees of BCS in the game-based setting. Our definition of blindness is akin to the strong blindness notion of standard blind signatures [19], in which the adversary picks the keys (as opposed to the weak version in which they are chosen by the experiment)[5]. Roughly speaking, it says that two promise/solve sessions cannot be linked together by the hub.[6]

Definition 4.3 (Blindness). A blind conditional signature Π_{BCS} is blind if there exists a negligible function $\mathsf{negl}(n)$ such that for all $n \in \mathbb{N}$ and all PPT adversaries $\mathcal{A}$, the following holds:

$$\Pr\left[\mathsf{ExpBlnd}_{\Pi_{\mathsf{puzzle}}}^{\mathcal{A}}(n) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(n)$$

where ExpBlnd is defined in Figure 5.[7]

Next, we define unlockability, which says that it should be hard for Hub to create a valid signature on Alice's message that does not allow Bob to unlock the full signature in the corresponding promise session.

[5] We opt for this stronger version since we want to provide anonymity even in the case of a fully malicious hub, which can pick its keys adversarially to try to link a sender/receiver pair.

[6] We do not consider the case in which Hub colludes with either Alice or Bob, since deanonymization is trivial (Alice (resp. Bob) simply reveals the identity of Bob (resp. Alice) to Hub); this is in line with [53].

[7] In previous works, descriptions of unlinkability assume an explicit step for blinding the puzzle τ between PPromise and PSolver. Here, we assume that PSolver performs this blinding functionality.

```
$\mathsf{ExpBlnd}^{\mathcal{A}}_{\Pi_{\mathsf{BCS}}}(n)$
$(\tilde{\mathsf{ek}}, \mathsf{vk}^H_0, \mathsf{vk}^H_1, (m_{HB,0}, m_{AH,0}), (m_{HB,1}, m_{AH,1})) \leftarrow \mathcal{A}(1^n)$
$(\mathsf{vk}^A_0, \mathsf{sk}^A_0) \leftarrow \mathsf{KGen}(1^n)$
$(\mathsf{vk}^A_1, \mathsf{sk}^A_1) \leftarrow \mathsf{KGen}(1^n)$
$\tau_0 \leftarrow \mathsf{PPromise}\left\langle \mathcal{A}(\mathsf{vk}^A_0, \mathsf{vk}^A_1), B(\tilde{\mathsf{ek}}, \mathsf{vk}^H_0, m_{HB,0})\right\rangle$
$\tau_1 \leftarrow \mathsf{PPromise}\left\langle \mathcal{A}(\mathsf{vk}^A_0, \mathsf{vk}^A_1), B(\tilde{\mathsf{ek}}, \mathsf{vk}^H_1, m_{HB,1})\right\rangle$
$b \leftarrow \{0, 1\}$
$(\sigma^*_0, s_0) \leftarrow \mathsf{PSolver}\left\langle A\left(\mathsf{sk}^A_0, \tilde{\mathsf{ek}}, m_{AH,0}, \tau_{0\oplus b}\right), \mathcal{A}\right\rangle$
$(\sigma^*_1, s_1) \leftarrow \mathsf{PSolver}\left\langle A\left(\mathsf{sk}^A_1, \tilde{\mathsf{ek}}, m_{AH,1}, \tau_{1\oplus b}\right), \mathcal{A}\right\rangle$
if $(\sigma^*_0 = \bot) \vee (\sigma^*_1 = \bot) \vee (\tau_0 = \bot) \vee (\tau_1 = \bot)$
    $\sigma_0 := \sigma_1 := \bot$
else
    $\sigma_{0\oplus b} \leftarrow \mathsf{Open}(\tau_{0\oplus b}, s_0)$
    $\sigma_{1\oplus b} \leftarrow \mathsf{Open}(\tau_{1\oplus b}, s_1)$
$b' \leftarrow \mathcal{A}(\sigma_0, \sigma_1)$
return $(b = b')$
```

Figure 5: Blindness experiment

```
$\mathsf{ExpUnlock}^{\mathcal{A}}_{\Pi_{\mathsf{BCS}}}(n)$
$(\tilde{\mathsf{ek}}, \mathsf{vk}^H, m_{HB}, m_{AH}) \leftarrow \mathcal{A}(1^n)$
$(\mathsf{vk}^A, \mathsf{sk}^A) \leftarrow \mathsf{KGen}(1^n)$
$\tau \leftarrow \mathsf{PPromise}\left\langle \mathcal{A}(\mathsf{vk}^A), B(\tilde{\mathsf{ek}}, \mathsf{vk}^H, m_{HB})\right\rangle$
if $\tau = \bot$
    $(\hat{\sigma}, \hat{m}) \leftarrow \mathcal{A}$
    $b_0 := (\mathsf{Vf}(\mathsf{vk}^A, \hat{\sigma}, \hat{m}) = 1)$
if $\tau \neq \bot$
    $(\sigma^*, s) \leftarrow \mathsf{PSolver}\left\langle A\left(\mathsf{sk}^A, \tilde{\mathsf{ek}}, m_{AH}, \tau\right), \mathcal{A}\right\rangle$
    $(\hat{\sigma}, \hat{m}) \leftarrow \mathcal{A}$
    $b_1 := (\mathsf{Vf}(\mathsf{vk}^A, \hat{\sigma}, \hat{m}) = 1) \wedge (\hat{m} \neq m_{AH})$
    $b_2 := (\mathsf{Vf}(\mathsf{vk}^A, \sigma^*, m_{AH}) = 1)$
    $b_3 := (\mathsf{Vf}(\mathsf{vk}^H, m_{HB}, \mathsf{Open}(\tau, s)) \neq 1)$
return $b_0 \vee b_1 \vee (b_2 \wedge b_3)$
```

Figure 6: Unlockability experiment

Definition 4.4 (Unlockability). A blind conditional signature Π_{BCS} is unlockable if there exists a negligible function $\mathsf{negl}(n)$ such that for all $n \in \mathbb{N}$ and all PPT adversaries $\mathcal{A}$, the following holds:

$$\Pr\left[\mathsf{ExpUnlock}^{\mathcal{A}}_{\Pi_{\mathsf{BCS}}}(n) = 1\right] \leq \mathsf{negl}(n)$$

where ExpUnlock is defined in Figure 6.

Our definition of unforgeability is inspired by the unforgeability of blind signatures [19]. We require that Alice and Bob cannot recover q signatures from Hub while successfully querying the solving oracle at most $q - 1$ times. Since each successful query reveals a signature from Alice's key (which in turn corresponds to a transaction from Alice to Hub), this requirement implicitly captures the fact that Alice and Bob cannot steal coins from Hub. The winning condition b_0 captures the scenario where the adversary forges a signature of the hub on a message previously not used in any promise oracle query. The remaining conditions b_1, b_2 and b_3 together capture the scenario in which the adversary outputs q valid distinct key-message-signature tuples while having queried for solve only $q - 1$ times. Hence, in the second condition, the attacker manages to *complete* q promise interactions with only $q - 1$ solve interactions, whereas in the first winning condition, the adversary computes a *fresh* signature that is not the completion of any promise interaction. These conditions are technically incomparable: an attacker that succeeds under one condition does not imply an attacker succeeding on the other. It is important to note that this is different from the unforgeability notion of blind signatures (where the attacker only has access to a single signing oracle), since in our case the hub is offering the attacker two oracles: promise and solve.

Definition 4.5 (Unforgeability). A blind conditional signature Π_{BCS} is *unforgeable* if there exists a negligible function $\mathsf{negl}(n)$ such that for all $n \in \mathbb{N}$ and all PPT adversaries $\mathcal{A}$, the following holds:

$$\Pr\left[\mathsf{ExpUnforg}^{\mathcal{A}}_{\Pi_{\mathsf{BCS}}}(n) = 1\right] \leq \mathsf{negl}(n)$$

where ExpUnforg is defined in Figure 7.

We define security as the collection of all properties.

Definition 4.6 (Security). A blind conditional signature Π_{BCS} is secure if it is blind, unlockable, and unforgeable.

4.2 The A^2L^+ Protocol

In the following we describe our A^2L^+ construction. Our scheme is a provable variant of A^2L (Section 3.1) and therefore we only describe the differences with respect to the original protocol. The concrete modifications are as follows:

- Augment the public key of Hub ek_H with a NIZK proof that certifies that $\mathsf{ek}_H \in \mathsf{Supp}(\Pi_{\mathsf{E}}.\mathsf{KGen}(1^n))$. All parties verify this proof during their first interaction with Hub.
- In PSolver (Figure 13), Hub additionally checks if vk^A_{AH} is in the support of $\Pi_{\mathsf{ADP}}.\mathsf{KGen}(1^n)$ before the decryption (line 6). Furthermore, we replace the condition (line 8) with

$$\Pi_{\mathsf{ADP}}.\mathsf{PreVf}(\mathsf{vk}^A_{AH}, m_{AH}, Y'', \tilde{\sigma}^A_{AH}) \neq 1 \vee\ g^{s''} \neq Y''.$$

4.3 Security Analysis

In this section we present our security results and defer the proofs to Appendix C. Before proving our main theorem, we define a property which is going to be useful for our analysis.

Definition 4.7 (OM-CCA-A2L). An encryption scheme Π_{E} is one-more CCA-A2L-secure (OM-CCA-A2L) if there exists a negligible function $\mathsf{negl}(n)$ such that for all $n \in \mathbb{N}$, all polynomials $q = q(n)$, and all PPT adversaries $\mathcal{A}$, the following holds:

$$\Pr\left[\text{OM-CCA-A2L}^{\mathcal{A}}_{\Pi_{\mathsf{E}},q}(n) = 1\right] \leq \mathsf{negl}(n),$$

```
$\mathsf{ExpUnforg}^{\mathcal{A}}_{\Pi_{\mathsf{BCS}}}(n)$
$\mathcal{L} := \emptyset, Q := 0$
$(\tilde{\mathsf{ek}}, \tilde{\mathsf{dk}}) \leftarrow \mathsf{Setup}(1^n)$
$(\mathsf{vk}^H_1, m_1, \sigma_1), \ldots, (\mathsf{vk}^H_q, m_q, \sigma_q) \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{PP}(\cdot),\mathcal{O}\mathsf{PS}(\cdot)}(\tilde{\mathsf{ek}})$
$b_0 := \exists i \in [q]$ s.t. $(\mathsf{vk}^H_i, \cdot) \in \mathcal{L} \wedge (\mathsf{vk}^H_i, m_i) \notin \mathcal{L}$
            $\wedge \mathsf{Vf}(\mathsf{vk}^H_i, m_i, \sigma_i) = 1$
$b_1 := \forall i \in [q], (\mathsf{vk}^H_i, m_i) \in \mathcal{L} \wedge \mathsf{Vf}(\mathsf{vk}^H_i, m_i, \sigma_i) = 1$
$b_2 := \bigwedge_{i,j \in [q], i \neq j} (\mathsf{vk}^H_i, m_i, \sigma_i) \neq (\mathsf{vk}^H_j, m_j, \sigma_j)$
$b_3 := (Q \leq q - 1)$
return $b_0 \vee (b_1 \wedge b_2 \wedge b_3)$

$\mathcal{O}\mathsf{PP}(m)$
$(\mathsf{vk}^H, \mathsf{sk}^H) \leftarrow \Pi_{\mathsf{ADP}}.\mathsf{KGen}(1^n)$
$\mathcal{L} := \mathcal{L} \cup \{(\mathsf{vk}^H, m)\}$
$\bot \leftarrow \mathsf{PPromise}\langle H(\tilde{\mathsf{dk}}, \mathsf{sk}^H, m), \mathcal{A}(\mathsf{vk}^H)\rangle$

$\mathcal{O}\mathsf{PS}(\mathsf{vk}^A, m')$
$\sigma^* \leftarrow \mathsf{PSolver}\langle \mathcal{A}, H(\tilde{\mathsf{dk}}, \mathsf{vk}^A, m')\rangle$
if $\sigma^* \neq \bot$ then $Q := Q + 1$
```

Figure 7: Unforgeability experiment

```
$\mathsf{OM\text{-}CCA\text{-}A2L}^{\mathcal{A}}_{\Pi_{\mathsf{E}},q}$
$Q := 0$
$(\mathsf{ek}, \mathsf{dk}) \leftarrow \Pi_{\mathsf{E}}.\mathsf{KGen}(1^n)$
$r_1, \ldots, r_{q+1} \leftarrow\$ \{0,1\}^n$
$c_i \leftarrow \Pi_{\mathsf{E}}.\mathsf{Enc}(\mathsf{ek}, r_i)$
$(r'_1, \ldots, r'_{q+1}) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{A^2L}}_{\mathsf{dk},\Pi_{\mathsf{E}},\Pi_{\mathsf{ADP}}}}(\mathsf{ek}, (c_1, g^{r_1}), \ldots, (c_{q+1}, g^{r_{q+1}}))$
if $r'_i = r_i \; \forall i \in 1, \ldots, q+1 \wedge Q \leq q$ then return $1$
else return $0$

$\mathcal{O}^{\mathsf{A^2L}}_{\mathsf{dk},\Pi_{\mathsf{E}},\Pi_{\mathsf{ADP}}}(\mathsf{vk}, m, h, c, \tilde{\sigma})$
check if $\mathsf{vk} \in \mathsf{Supp}(\Pi_{\mathsf{ADP}}.\mathsf{KGen}(1^n))$
$\tilde{x} \leftarrow \Pi_{\mathsf{E}}.\mathsf{Dec}(\mathsf{dk}, c)$
if $\Pi_{\mathsf{ADP}}.\mathsf{PreVf}(\mathsf{vk}, m, h, \tilde{\sigma}) = 1$ and $g^{\tilde{x}} = h$
    $Q := Q + 1$
    return $\sigma' \leftarrow \Pi_{\mathsf{ADP}}.\mathsf{Adapt}(\tilde{\sigma}, \tilde{x})$
else return $\bot$
```

Figure 8: OM-CCA-A2L game

where OM-CCA-A2L is defined in Figure 8.

The following technical lemma shows that an LOE scheme satisfies this property, assuming the hardness of the OMDL problem. The formal analysis of the below lemma is deferred to Appendix C.

Lemma 4.8. *Let Π_{E} be an LOE scheme. Assuming the hardness of OMDL, Π_{E} is OM-CCA-A2L secure.*

Main Theorem. We are now ready to give the main theorem of this section. The formal analysis is deferred to Appendix C.

Theorem 4.9. *Let Π_{E} be an LOE scheme, Π_{ADP} a secure adaptor signature scheme, and Π_{NIZK} a sound NIZK proof system. Assuming the hardness of OMDL, the A^2L^+ protocol is a secure blind conditional signature scheme.*

5 UC-SECURE BLIND CONDITIONAL SIGNATURES

We now model security in the *universal composability* framework from Canetti [16] extended to support a global setup [17] in order to capture concurrent executions. We refer the reader to [16] for a comprehensive discussion. We consider *static* corruptions, where the adversary announces at the beginning which parties it corrupts. We denote the environment by $\mathcal{E}$. For a real protocol Π and an adversary $\mathcal{A}$ we write $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{E}}$ to denote the ensemble corresponding to the protocol execution. For an ideal functionality $\mathcal{F}$ and an adversary $\mathcal{S}$ we write $\mathsf{EXEC}_{\mathcal{F},\mathcal{S},\mathcal{E}}$ to denote the distribution ensemble of the ideal world execution.

Definition 5.1 (Universal Composability). A protocol Π UC-realizes an ideal functionality $\mathcal{F}$ if for any PPT adversary $\mathcal{A}$ there exists a simulator $\mathcal{S}$ such that for any environment $\mathcal{E}$ the ensembles $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{E}} \approx \mathsf{EXEC}_{\mathcal{F},\mathcal{S},\mathcal{E}}$ are computationally indistinguishable.

In our protocol, we assume the existence of a general-purpose UC-secure 2-party computation (2PC) protocol [18, 33], where two parties interact with the ideal functionality to compute a function $f(x, y)$ over their private inputs x and y.

5.1 Ideal functionality

We describe the ideal functionality $\mathcal{F}_{\mathsf{BCS}}$ that captures the functionality and security of BCS in the UC framework. We refer the reader to Appendix D for the formal description of $\mathcal{F}_{\mathsf{BCS}}$. The ideal functionality has three routines, namely for puzzle promise, puzzle solver, and open, which intuitively capture the functionality of BCS as discussed in Section 4. On a high level, $\mathcal{F}_{\mathsf{BCS}}$ captures *blindness* by sampling the puzzle identifiers pid and pid′, which correspond to puzzle promise and puzzle solve interactions, locally together, but never revealing them together to the hub. $\mathcal{F}_{\mathsf{BCS}}$ captures *atomicity* by returning a successful message (not aborting) for pid during open if and only if it sent a successful solved message during the puzzle solve interaction for the puzzle identifier pid′ (where pid and pid′ correspond to each other). Note that the above atomicity guarantee implies the game-based definitions of unlockability and unforgeability.

Our functionality $\mathcal{F}_{\mathsf{BCS}}$ is taken verbatim from the $\mathcal{F}_{\mathsf{A^2L}}$ functionality in [53] except that we do not consider user registrations (as done in $\mathcal{F}_{\mathsf{A^2L}}$) to tackle *griefing attacks* [44] in the coin mixing layer. These attacks are mounted by Bob starting many puzzle promise operations, each of which requires Hub to lock coins, whereas the corresponding puzzle solver interactions are never carried out. As a consequence, all of Hub's coins are locked and no longer available, which results in a form of denial of service attack. We argue that the issue does not concern the functionality or security of BCS as a cryptographic tool, but only affects the coin mixing protocol at the transaction layer. We emphasize that griefing attacks can be thwarted at this layer in both the formal model and the construction using the same ideas as in [53].

5.2 Our Protocol: A^2L^{UC}

We now describe our protocol A^2L^{UC} that realizes the ideal functionality $\mathcal{F}_{\mathsf{BCS}}$. We assume the following cryptographic building blocks:

- An adaptor signature scheme Π_{ADP} defined with respect to Π_{DS} and a hard relation R_{DL}.
- A UC-secure NIZK proof system Π_{NIZK} for the language

$$\mathcal{L} := \{(\mathsf{ek}, Y, c) \mid \exists s, \text{ s.t. } c \leftarrow \Pi_{\mathsf{E}}.\mathsf{Enc}(\mathsf{ek}, s) \wedge Y = g^s\}.$$

- A UC-secure 2PC protocol.
- A CCA-secure [8] encryption scheme $\Pi_{\mathsf{E}} := (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ with unique decryption keys.

The property of unique decryption keys is formalized below.

Definition 5.2 (Unique Decryption Keys). An encryption scheme Π_{E} has unique decryption keys if the KGen algorithm is of the following form:

- Sample $\mathsf{dk} \leftarrow\!\!\$ \{0,1\}^n$.
- Run $\mathsf{ek} \leftarrow \mathsf{Gen}(\mathsf{dk})$.

Furthermore, for all ek output by KGen, there exists a unique dk such that $\mathsf{ek} = \mathsf{Gen}(\mathsf{dk})$. In other words, Gen is injective.

This property is already satisfied by most natural public-key encryption schemes, but it can be generically achieved by augmenting the encryption key with a perfectly binding commitment $\mathsf{com}(\mathsf{dk})$ to the decryption key dk.

Protocol Description. We assume Alice and Hub have a key pair for the signature scheme Π_{DS}. Specifically, we have the verification-signing key pairs $(\mathsf{vk}^H_{HB}, \mathsf{sk}^H_{HB})$ and $(\mathsf{vk}^A_{AH}, \mathsf{sk}^A_{AH})$, belonging to Hub and Alice, respectively. We then have two messages $m := m_{HB}$ and $m' := m_{AH}$ for which the users wish to generate blind conditional signatures. The setup and open algorithms are formally described in Figure 9. The puzzle promise and puzzle solver of A^2L^{UC} are formally described in Figure 10 and Figure 11, respectively. For ease of understanding, we briefly describe below our A^2L^{UC} protocol in terms of the differences with the A^2L protocol (Figures 12 and 13).

- The setup algorithm (Figure 9) of A^2L^{UC} generates the keys of Hub, which are the keys for the (CCA-secure) encryption scheme Π_{E}.
- In PPromise of A^2L^{UC} (Figure 10),
  - The NIZK proof system is UC-secure.
  - Bob no longer re-randomizes the instance or the ciphertext. Therefore, we drop the re-randomization steps (line 9 and 10) of PPromise in A^2L (Figure 12). Simply set the puzzle to $\tau := (m_{HB}, \tilde{\sigma}^H_{HB}, (Y, c))$.
- In PSolver of A^2L^{UC} (Figure 11),
  - Alice no longer sends the ciphertext to Hub (line 5 of Figure 13). We therefore remove the local decryption step (line 6 of Figure 13), and replace it with a 2PC protocol (line 6 of Figure 11).
  - At the end of the 2PC protocol, Alice receives $\perp$, while Hub receives the value z. Hub additionally checks if $Y' = g^z$ (line 7) and uses z to adapt the pre-signature $\tilde{\sigma}^A_{AH}$ to signature σ^A_{AH}.
  - We add a check for Alice (line 10) that σ^A_{AH} is a valid signature before extracting the witness z' in line 12.
- The Open algorithm (Figure 9) is the same as in Figure 14 of A^2L, except we skip removing the randomness factor. The algorithm in Figure 9 now simply adapts a pre-signature $\tilde{\sigma}$ to a valid signature σ which it returns as output.

```
Setup(1^n)
  $(\mathsf{ek}_H, \mathsf{dk}_H) \leftarrow \Pi_{\mathsf{E}}.\mathsf{KGen}(1^n)$
  set $\tilde{\mathsf{pk}} := \mathsf{ek}_H, \tilde{\mathsf{sk}} := \mathsf{dk}_H$
  return $(\tilde{\mathsf{pk}}, \tilde{\mathsf{sk}})$

Open(τ, s)
  parse $\tau := (\cdot, \tilde{\sigma}, \cdot)$
  $\sigma \leftarrow \Pi_{\mathsf{ADP}}.\mathsf{Adapt}(\tilde{\sigma}, s)$
  return $\sigma$
```

Figure 9: Setup and Open algorithms of our conditional puzzle construction

5.3 Security Analysis

We now show that A^2L^{UC} satisfies UC-security. In favor of a simpler analysis, we assume that the verification keys of all parties are honestly generated. In practice, this can be enforced by augmenting keys with NIZKs that certify their validity [12, 36]. We state here our security theorem and defer the formal proof to Appendix C.

Theorem 5.3. *Let* Π_{E} *be a CCA-secure encryption scheme,* Π_{ADP} *a secure adaptor signature scheme,* 2PC *a UC-secure two-party computation protocol, and* Π_{NIZK} *a UC-secure NIZK for the language* $\mathcal{L}$ *above. Then the* A^2L^{UC} *protocol UC-realizes* $\mathcal{F}_{\mathsf{BCS}}$.

6 EFFICIENCY

We now discuss the efficiency of our constructions A^2L^+ and A^2L^{UC} in terms of number of cryptographic operations.

6.1 A^2L^+

Recall that we use an encryption scheme Π_{E} in the LOE model. Below we present an instantiation of such a Π_{E}.

Instantiating Linear-Only Encryption. As shown in [11] it is not sufficient to instantiate this with any linearly homomorphic encryption (e.g., ElGamal). Though the scheme may not support homomorphic operations beyond linear, it may still have *obliviously sampleable ciphertexts*, i.e., the ability to sample a ciphertext without knowing the underlying plaintext. Note that this falls outside the LOE model, since there is no oracle that implements this functionality. Thus, as suggested in [11] we implement an additional safeguard needed to prevent oblivious sampling. Given a linearly homomorphic encryption scheme $\Pi^*_{\mathsf{E}} := (\mathsf{KGen}^*, \mathsf{Enc}^*, \mathsf{Dec}^*)$ over $\mathbb{Z}_p$, we define a candidate LOE $\Pi_{\mathsf{E}} := (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ as follows:

```
Public parameters: group description $(\mathbb{G}, g, q)$, message $m_{HB}$
$\mathsf{PPromise}\langle H(\mathsf{dk}_H, \mathsf{sk}^H_{HB}), \cdot\rangle$                          $\mathsf{PPromise}\langle \cdot, B(\mathsf{ek}_H, \mathsf{vk}^H_{HB})\rangle$
$s \leftarrow\$\, \mathbb{Z}_p, Y := g^s$
$c \leftarrow \Pi_{\mathsf{E}}.\mathsf{Enc}(\mathsf{ek}_H, s)$
$\pi_s \leftarrow \mathsf{NIZK.P}((\mathsf{ek}_H, Y, c), s)$
$\tilde{\sigma}^H_{HB} \leftarrow \Pi_{\mathsf{ADP}}.\mathsf{PreSig}(\mathsf{sk}^H_{HB}, m_{HB}, Y)$
                              $\xrightarrow{Y, c, \pi_s, \tilde{\sigma}^H_{HB}}$
                                                    If $\mathsf{NIZK.V}((\mathsf{ek}_H, Y, c), \pi_s) \neq 1$ then return $\bot$
                                                    If $\Pi_{\mathsf{ADP}}.\mathsf{PreVf}(\mathsf{vk}^H_{HB}, m_{HB}, Y, \tilde{\sigma}^H_{HB}) \neq 1$ then
                                                      return $\bot$
                                                    set $\tau := (m_{HB}, \tilde{\sigma}^H_{HB}, (Y, c))$
return $\bot$                                       return $\tau$
```

Figure 10: Puzzle promise protocol of $\mathrm{A^2L^{UC}}$

```
Public parameters: group description $(\mathbb{G}, g, q)$, message $m_{AH}$
$\mathsf{PSolver}\langle A(\mathsf{sk}^A_{AH}, \mathsf{ek}_H, \tau), \cdot\rangle$                          $\mathsf{PSolver}\langle \cdot, H(\mathsf{dk}_H, \mathsf{vk}^A_{AH})\rangle$
parse $\tau := (\cdot, \cdot, (Y, c))$
$r \leftarrow\$\, \mathbb{Z}_p, Y' := Y \cdot g^r$
$\tilde{\sigma}^A_{AH} \leftarrow \Pi_{\mathsf{ADP}}.\mathsf{PreSig}(\mathsf{sk}^A_{AH}, m_{AH}, Y')$
                              $\xrightarrow{Y', \tilde{\sigma}^A_{AH}}$
                                                    If $\Pi_{\mathsf{ADP}}.\mathsf{PreVf}(\mathsf{vk}^A_{AH}, m_{AH}, Y', \tilde{\sigma}^A_{AH}) \neq 1$ then
                                                      return $\bot$
                  $\mathsf{2PC}((r, c), (\mathsf{dk}_H))$
                    if $\mathsf{ek}_H \neq \Pi_{\mathsf{E}}.\mathsf{Gen}(\mathsf{dk}_H)$
                     then abort
                    $s^* \leftarrow \Pi_{\mathsf{E}}.\mathsf{Dec}(\mathsf{dk}_H, c)$
                    $z := s^* + r$
                    return $((\bot), (z))$
                                                    If $Y' \neq g^z$ then return $\bot$
                                                    $\sigma^A_{AH} \leftarrow \Pi_{\mathsf{ADP}}.\mathsf{Adapt}(\tilde{\sigma}^A_{AH}, z)$
                              $\xleftarrow{\sigma^A_{AH}}$
If $\Pi_{\mathsf{ADP}}.\mathsf{Vf}(\mathsf{vk}^A_{AH}, m_{AH}, \sigma^A_{AH}) \neq 1$ then
  return $\bot$
$z' \leftarrow \Pi_{\mathsf{ADP}}.\mathsf{Ext}(\tilde{\sigma}^A_{AH}, \sigma^A_{AH}, Y')$
$s := z' - r$
return $(\sigma^A_{AH}, s)$                          return $\sigma^A_{AH}$
```

Figure 11: Puzzle solver protocol of $\mathrm{A^2L^{UC}}$

Table 1: Operations in $\mathrm{A^2L}$ and $\mathrm{A^2L^+}$ when instantiated with Schnorr or ECDSA adaptor signatures [4]. We give the number group exponentiations (Exp) and group operations (Op) in both class groups (CL) and groups of prime order p ($\mathbb{G}$), where $\log p = n$. Group element inversions (Inv) only occur in class groups. Modular multiplications (×) and additions (+) are performed modulo q. We denote by #H the number of hash computations. Decryption of a CL ciphertext also involves solving a discrete logarithm in a class group, which we denote by DLog.

Protocol	Signature	Exp (CL)	Op (CL)	Inv (CL)	DLog (CL)	Exp ($\mathbb{G}$)	Op ($\mathbb{G}$)	× mod q	+ mod q	#H
$\mathrm{A^2L}$	Schnorr	18	12	1	1	13	8	4	9	6
(insecure)	ECDSA	18	12	1	1	27	8	17	10	11
$\mathrm{A^2L^+}$	Schnorr	28	20	2	2	14	9	5	9	6
	ECDSA	28	20	2	2	32	10	21	12	11

- $\mathsf{KGen}(1^n)$: Sample $(\mathsf{ek}^*, \mathsf{dk}^*) \leftarrow \mathsf{KGen}^*(1^n)$ and some $\alpha \leftarrow\!\!\$\, \mathbb{Z}_p$. Return $\mathsf{dk} := (\mathsf{dk}^*, \alpha)$ as the decryption key and $\mathsf{ek} := (\mathsf{ek}^*, \mathsf{Enc}^*(\mathsf{ek}^*, \alpha))$ as the encryption key.
- $\mathsf{Enc}(\mathsf{ek}^*, x)$: Compute c as $(\mathsf{Enc}^*(\mathsf{ek}^*, x), \mathsf{Enc}^*(\mathsf{ek}^*, \alpha \cdot x))$, where $\mathsf{Enc}^*(\mathsf{ek}^*, \alpha \cdot x)$ is computed homomorphically using ek.
- $\mathsf{Dec}(\mathsf{dk}^*, c)$: Parse c as (c_0, c_1) and compute $x_0 \leftarrow \mathsf{Dec}^*(\mathsf{dk}^*, c_0)$ and $x_1 \leftarrow \mathsf{Dec}^*(\mathsf{dk}^*, c_1)$. If $x_1 = \alpha \cdot x_0$ return x_0, else return $\perp$.

We note that the security of Π_E follows from the security of Π^*_E. Intuitively, we prevent oblivious ciphertext sampling, since it is infeasible for an adversary to sample a ciphertext component c_0 that is consistent with c_1 without knowing the underlying plaintext of c_0.

Added Costs. The new consistency check by the hub in PSolver adds 1 group operation and group exponentiation (Schnorr) or 5 group operations and 2 group exponentiations (ECDSA). The check on Alice's verification key vk^A_{AH} adds 3 modular multiplications and 2 modular additions in the ECDSA case. Furthermore, applying the LOE transformation described above to the CL encryption scheme results in a doubled ciphertext size and a corresponding increase in the operation count for decryption. We summarize the costs of $\mathrm{A^2L}$ and $\mathrm{A^2L^+}$ in Table 1.

6.2 $\mathrm{A^2L^{UC}}$

Compared to $\mathrm{A^2L^+}$, our $\mathrm{A^2L^{UC}}$ protocol removes the check on vk^A_{AH}, adds a signature verification, and moves the re-randomization and decryption into the 2PC. Additionally, Π_E is now required to be CCA-secure and the NIZK used must be UC-secure. The cost of the first two changes is minimal (net 1 group exponentiation, 1 group operation, and 1 hash computation); the most significant overhead is the result of the 2PC computation and the NIZK.

Assuming the CCA-secure Π_E in the 2PC is instantiated with the (prime-order-based) Cramer-Shoup cryptosystem [21] with SHA3-256 [41] as the hash function, this incurs an overhead of 11 exponentiations, 9 multiplications, and 1 division in a group of prime order p and $\lceil \frac{3n}{1088} \rceil \cdot 38400$ binary (AND) operations, where the security parameter n equals $\log p$. Because the 2PC requires a mix of arithmetic and binary operations, a mixed-circuit 2PC protocol as implemented e.g. in [34] could be used. Additionally, UC security of the NIZK can be achieved by replacing the use of the Fiat-Shamir transform in $\mathrm{A^2L}$ (and $\mathrm{A^2L^+}$) with the Fischlin transform, incurring a cost of roughly $O(\log(n))$ parallel repetitions of the base Fiat-Shamir NIZK. We stress that we view $\mathrm{A^2L^{UC}}$ as a proof-of-concept protocol showing the feasibility of achieving UC-secure blind conditional signatures and leave the problem of constructing an efficient UC-secure realization as an interesting direction for future work.

7 CONCLUSIONS

We investigate the notion of synchronization puzzles, the cryptographic building blocks at the core of hub-enabled coin mixing services. We find that the previous formalization of a synchronization puzzle in [53] is flawed. In fact, we identify several issues in its formal model which can be easily exploited to break the security of the resultant coin mixing protocol. We conclude that tighter formalization of the functionality and security of synchronization puzzles is necessary.

To fill this gap, we introduce the notion of blind conditional signatures (BCS). Additionally, we provide different security formalizations for BCS at varying levels of strength (game-based and in the UC framework) accompanied by a provably secure variant of $\mathrm{A^2L}$ called $\mathrm{A^2L^+}$ and a new provably UC-secure construction $\mathrm{A^2L^{UC}}$. Our performance evaluation results show an efficiency vs. security trade-off in the case of our constructions, yet show with $\mathrm{A^2L^+}$ that provably secure coin mixing services are deployable in practice.

ACKNOWLEDGMENTS

This work was partially supported by the European Research Council (ERC) under the European Union's Horizon 2020 research (grant agreement 771527-BROWSEC), by the Austrian Science Fund (FWF) through the projects PROFET (grant agreement P31621) and the project W1255-N23, by the Austrian Research Promotion Agency (FFG) through COMET K1 SBA and COMET K1 ABC, by the Vienna Business Agency through the project Vienna Cybersecurity and Privacy Research Center (VISP), by the Austrian Federal Ministry for Digital and Economic Affairs, the National Foundation for Research, Technology and Development and the Christian Doppler Research Association through the Christian Doppler Laboratory Blockchain Technologies for the Internet of Things (CDL-BOT). This work has been also partially supported by the Madrid regional government as part of the program S2018/TCS-4339 (BLOQUES-CM) co-funded by EIE Funds of the European Union, by grant IJC2020-043391-I/MCIN/AEI/10.13039/501100011033 and European Union NextGenerationEU/PRTR, by SCUM Project (RTI2018-102043-B-I00) MCIN/AEI/10.13039/501100011033/ERDF A way of making Europe, and by the project HACrypt. Additionally, this material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE 1840340. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. The work was also made possible through the support of the David and Lucille Packard Foundation (award #202071730), SRI International (award #53978) / Prime: Defense Advanced Research Projects Agency (award #HR00110C0086), and the National Science Foundation (award #2212746). G.M. was partially supported by the German Federal Ministry of Education and Research (BMBF) (grant 16K15K042, project 6GEM).

REFERENCES

[1] Accessed on May 2022. CoinJoin - Bitcoin Wiki. https://en.bitcoin.it/wiki/CoinJoin.
[2] Accessed on May 2022. Coinswap. https://coinswap.space.
[3] Lukas Aumayr, Oguzhan Ersoy, Andreas Erwig, Sebastian Faust, Kristina Hostakova, Matteo Maffei, Pedro Moreno-Sanchez, and Siavash Riahi. 2020. Generalized Bitcoin-Compatible Channels. Cryptology ePrint Archive, Report 2020/476. https://eprint.iacr.org/2020/476.
[4] Lukas Aumayr, Oguzhan Ersoy, Andreas Erwig, Sebastian Faust, Kristina Hostakova, Matteo Maffei, Pedro Moreno-Sanchez, and Siavash Riahi. 2020. Generalized Channels from Limited Blockchain Scripts and Adaptor Signatures. Cryptology ePrint Archive, Report 2020/476. https://ia.cr/2020/476.
[5] European Central Bank. 2021. Digital Euro. https://www.ecb.europa.eu/paym/digital_euro/html/index.en.html
[6] Simon Barber, Xavier Boyen, Elaine Shi, and Ersin Uzun. 2012. Bitter to Better — How to Make Bitcoin a Better Currency. *Advances in Water Resources - ADV WATER RESOUR* 7397. https://doi.org/10.1007/978-3-642-32946-3_29

[7] Balthazar Bauer, Georg Fuchsbauer, and Antoine Plouviez. 2021. The One-More Discrete Logarithm Assumption in the Generic Group Model. Cryptology ePrint Archive, Report 2021/866. https://ia.cr/2021/866.

[8] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. 1997. A concrete security treatment of symmetric encryption. In *Proceedings 38th Annual Symposium on Foundations of Computer Science*. 394–403. https://doi.org/10.1109/SFCS.1997.646128

[9] Mihir Bellare, Chanathip Namprempre, David Pointcheval, and Michael Semanko. 2003. The One-More-RSA-Inversion Problems and the Security of Chaum's Blind Signature Scheme. *Journal of Cryptology* 16, 3 (June 2003), 185–215. https://doi.org/10.1007/s00145-002-0120-1

[10] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. 2014. Zerocash: Decentralized Anonymous Payments from Bitcoin. In *2014 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, 459–474. https://doi.org/10.1109/SP.2014.36

[11] Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth. 2013. Succinct Non-interactive Arguments via Linear Interactive Proofs. In *TCC 2013 (LNCS, Vol. 7785)*, Amit Sahai (Ed.). Springer, Heidelberg, 315–333. https://doi.org/10.1007/978-3-642-36594-2_18

[12] Alexandra Boldyreva. 2003. Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme. In *PKC 2003 (LNCS, Vol. 2567)*, Yvo Desmedt (Ed.). Springer, Heidelberg, 31–46. https://doi.org/10.1007/3-540-36288-6_3

[13] Dan Boneh, Shai Halevi, Michael Hamburg, and Rafail Ostrovsky. 2008. Circular-Secure Encryption from Decision Diffie-Hellman. In *CRYPTO 2008 (LNCS, Vol. 5157)*, David Wagner (Ed.). Springer, Heidelberg, 108–125. https://doi.org/10.1007/978-3-540-85174-5_7

[14] Joseph Bonneau, Arvind Narayanan, Andrew Miller, Jeremy Clark, Joshua A. Kroll, and Edward W. Felten. 2014. *Mixcoin: Anonymity for Bitcoin with Accountable Mixes.* Technical Report 077.

[15] Jan Camenisch, Stephan Krenn, and Victor Shoup. 2011. A Framework for Practical Universally Composable Zero-Knowledge Protocols. In *ASIACRYPT 2011 (LNCS, Vol. 7073)*, Dong Hoon Lee and Xiaoyun Wang (Eds.). Springer, Heidelberg, 449–467. https://doi.org/10.1007/978-3-642-25385-0_24

[16] Ran Canetti. 2001. Universally composable security: A new paradigm for cryptographic protocols. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*. IEEE, 136–145.

[17] Ran Canetti, Yevgeniy Dodis, Rafael Pass, and Shabsi Walfish. 2007. Universally Composable Security with Global Setup. In *TCC 2007 (LNCS, Vol. 4392)*, Salil P. Vadhan (Ed.). Springer, Heidelberg, 61–85. https://doi.org/10.1007/978-3-540-70936-7_4

[18] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. 2002. Universally composable two-party and multi-party secure computation. In *34th ACM STOC*. ACM Press, 494–503. https://doi.org/10.1145/509907.509980

[19] David Chaum. 1982. Blind Signatures for Untraceable Payments. In *CRYPTO'82*, David Chaum, Ronald L. Rivest, and Alan T. Sherman (Eds.). Plenum Press, New York, USA, 199–203.

[20] Mihai Christodorescu, Erin English, Wanyun Catherine Gu, David Kreissman, Ranjit Kumaresan, Mohsen Minaei, Srinivasan Raghuraman, Cuy Sheffield, Arjuna Wijeyekoon, and Mahdi Zamani. 2021. Universal Payment Channels: An Interoperability Platform for Digital Currencies. *arXiv:2109.12194 [cs]* (Sept. 2021). arXiv:2109.12194 [cs]

[21] Ronald Cramer and Victor Shoup. 1998. A Practical Public Key Cryptosystem Provably Secure Against Adaptive Chosen Ciphertext Attack. In *CRYPTO'98 (LNCS, Vol. 1462)*, Hugo Krawczyk (Ed.). Springer, Heidelberg, 13–25. https://doi.org/10.1007/BFb0055717

[22] Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. 1987. Non-interactive zero-knowledge proof systems. In *Conference on the Theory and Application of Cryptographic Techniques*. Springer, 52–72.

[23] Stefan Dziembowski, Lisa Eckey, Sebastian Faust, and Daniel Malinowski. 2019. Perun: Virtual Payment Hubs over Cryptocurrencies. In *2019 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, 106–123. https://doi.org/10.1109/SP.2019.00020

[24] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. 2018. The Algebraic Group Model and its Applications. In *CRYPTO 2018, Part II (LNCS, Vol. 10992)*, Hovav Shacham and Alexandra Boldyreva (Eds.). Springer, Heidelberg, 33–62. https://doi.org/10.1007/978-3-319-96881-0_2

[25] Craig Gentry. 2009. Fully homomorphic encryption using ideal lattices. In *41st ACM STOC*, Michael Mitzenmacher (Ed.). ACM Press, 169–178. https://doi.org/10.1145/1536414.1536440

[26] Simin Ghesmati, Walid Fdhila, and Edgar Weippl. 2021. *SoK: How Private Is Bitcoin? Classification and Evaluation of Bitcoin Mixing Techniques.* Technical Report 629.

[27] Shafi Goldwasser and Silvio Micali. 1984. Probabilistic encryption. *J. Comput. System Sci.* 28, 2 (1984), 270–299. https://doi.org/10.1016/0022-0000(84)90070-9

[28] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. 1988. A Digital Signature Scheme Secure Against Adaptive Chosen-message Attacks. *SIAM J. Comput.* 17, 2 (April 1988), 281–308.

[29] Jens Groth. 2004. Rerandomizable and Replayable Adaptive Chosen Ciphertext Attack Secure Cryptosystems. In *TCC 2004 (LNCS, Vol. 2951)*, Moni Naor (Ed.). Springer, Heidelberg, 152–170. https://doi.org/10.1007/978-3-540-24638-1_9

[30] Bernhard Haslhofer, Rainer Stütz, Matteo Romiti, and Ross King. 2021. GraphSense: A General-Purpose Cryptoasset Analytics Platform. *CoRR* abs/2102.13613 (2021). arXiv:2102.13613

[31] Ethan Heilman, Leen Alshenibr, Foteini Baldimtsi, Alessandra Scafuro, and Sharon Goldberg. 2017. TumbleBit: An Untrusted Bitcoin-Compatible Anonymous Payment Hub. In *NDSS 2017*. The Internet Society.

[32] Ethan Heilman, Foteini Baldimtsi, and Sharon Goldberg. 2016. *Blindly Signed Contracts: Anonymous On-Blockchain and Off-Blockchain Bitcoin Transactions.* Technical Report 056.

[33] Omer Horvitz and Jonathan Katz. 2007. Universally-Composable Two-Party Computation in Two Rounds, See [40], 111–129. https://doi.org/10.1007/978-3-540-74143-5_7

[34] Marcel Keller. 2020. MP-SPDZ: A Versatile Framework for Multi-Party Computation. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. https://doi.org/10.1145/3372297.3417872

[35] Russell W. F. Lai, Viktoria Ronge, Tim Ruffing, Dominique Schröder, Sri Aravinda Krishnan Thyagarajan, and Jiafan Wang. 2019. Omniring: Scaling Private Payments Without Trusted Setup. In *ACM CCS 2019*, Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz (Eds.). ACM Press, 31–48. https://doi.org/10.1145/3319535.3345655

[36] Steve Lu, Rafail Ostrovsky, Amit Sahai, Hovav Shacham, and Brent Waters. 2006. Sequential Aggregate Signatures and Multisignatures Without Random Oracles. In *EUROCRYPT 2006 (LNCS, Vol. 4004)*, Serge Vaudenay (Ed.). Springer, Heidelberg, 465–485. https://doi.org/10.1007/11761679_28

[37] Ueli M. Maurer. 2005. Abstract Models of Computation in Cryptography (Invited Paper). In *10th IMA International Conference on Cryptography and Coding (LNCS, Vol. 3796)*, Nigel P. Smart (Ed.). Springer, Heidelberg, 1–12.

[38] Sarah Meiklejohn and Rebekah Mercer. 2017. Möbius: Trustless Tumbling for Transaction Privacy. *Proceedings on Privacy Enhancing Technologies* 2018 (2017), 105 – 121.

[39] Sarah Meiklejohn, Marjori Pomarole, Grant Jordan, Kirill Levchenko, Damon McCoy, Geoffrey M. Voelker, and Stefan Savage. 2016. A Fistful of Bitcoins: Characterizing Payments among Men with No Names. *Commun. ACM* 59, 4 (March 2016), 86–93. https://doi.org/10.1145/2896384

[40] Alfred Menezes (Ed.). 2007. *CRYPTO 2007*. LNCS, Vol. 4622. Springer, Heidelberg.

[41] National Institute of Standards and Technology. 2015. SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. http://dx.doi.org/10.6028/NIST.FIPS.202

[42] Rafail Ostrovsky, Anat Paskin-Cherniavsky, and Beni Paskin-Cherniavsky. 2014. Maliciously Circuit-Private FHE. In *CRYPTO 2014, Part I (LNCS, Vol. 8616)*, Juan A. Garay and Rosario Gennaro (Eds.). Springer, Heidelberg, 536–553. https://doi.org/10.1007/978-3-662-44371-2_30

[43] Manoj Prabhakaran and Mike Rosulek. 2007. Rerandomizable RCCA Encryption, See [40], 517–534. https://doi.org/10.1007/978-3-540-74143-5_29

[44] D. Robinson. 2019. HTLCs considered harmful. https://cbr.stanford.edu/sbc19/.

[45] Tim Ruffing and Pedro Moreno-Sanchez. 2017. ValueShuffle: Mixing Confidential Transactions for Comprehensive Transaction Privacy in Bitcoin. In *Financial Cryptography and Data Security - FC 2017 International Workshops, WAHC, BITCOIN, VOTING, WTSC, and TA, Sliema, Malta, April 7, 2017, Revised Selected Papers (Lecture Notes in Computer Science, Vol. 10323)*, Michael Brenner, Kurt Rohloff, Joseph Bonneau, Andrew Miller, Peter Y. A. Ryan, Vanessa Teague, Andrea Bracciali, Massimiliano Sala, Federico Pintore, and Markus Jakobsson (Eds.). Springer, 133–154. https://doi.org/10.1007/978-3-319-70278-0_8

[46] Tim Ruffing, Pedro Moreno-Sanchez, and Aniket Kate. 2014. CoinShuffle: Practical Decentralized Coin Mixing for Bitcoin. In *Computer Security - ESORICS 2014 - 19th European Symposium on Research in Computer Security, Wroclaw, Poland, September 7-11, 2014. Proceedings, Part II (Lecture Notes in Computer Science, Vol. 8713)*, Miroslaw Kutylowski and Jaideep Vaidya (Eds.). Springer, 345–364. https://doi.org/10.1007/978-3-319-11212-1_20

[47] Tim Ruffing, Pedro Moreno-Sanchez, and Aniket Kate. 2016. *P2P Mixing and Unlinkable Bitcoin Transactions.* Technical Report 824.

[48] Dominique Schröder and Dominique Unruh. 2017. Security of Blind Signatures Revisited. *Journal of Cryptology* 30, 2 (April 2017), 470–494. https://doi.org/10.1007/s00145-015-9225-1

[49] István András Seres, Dániel A. Nagy, Chris Buckland, and Péter Burcsi. 2019. MixEth: Efficient, Trustless Coin Mixing Service for Ethereum. In *International Conference on Blockchain Economics, Security and Protocols, Tokenomics*, Vol. 71. 13:1–13:20.

[50] Victor Shoup. 1997. Lower Bounds for Discrete Logarithms and Related Problems. In *EUROCRYPT'97 (LNCS, Vol. 1233)*, Walter Fumy (Ed.). Springer, Heidelberg, 256–266. https://doi.org/10.1007/3-540-69053-0_18

[51] Johann Stockinger, Bernhard Haslhofer, Pedro Moreno-Sanchez, and Matteo Maffei. 2021. Pinpointing and Measuring Wasabi and Samourai CoinJoins in the Bitcoin Ecosystem. arXiv:2109.10229 [cs.CR]

[52] Jeff Stone. Accessed on May 2022. Evolution Downfall: Insider 'Exit Scam' Blamed For Massive Drug Bazaar's Sudden Disappearance. https://www.ibtimes.com/evolution-downfall-insider-exit-scam-blamed-massive-drug-bazaars-sudden-disappearance-1856190.

[53] E. Tairi, P. Moreno-Sanchez, and M. Maffei. 2021. A2L: Anonymous Atomic Locks for Scalability in Payment Channel Hubs. In *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, Los Alamitos, CA, USA, 1834–1851. https://doi.org/10.1109/SP40001.2021.00111

[54] Luke Valenta and Brendan Rowan. 2015. Blindcoin: Blinded, Accountable Mixes for Bitcoin. In *Financial Cryptography and Data Security (Lecture Notes in Computer Science)*, Michael Brenner, Nicolas Christin, Benjamin Johnson, and Kurt Rohloff (Eds.). Springer, Berlin, Heidelberg, 112–126. https://doi.org/10.1007/978-3-662-48051-9_9

A RANDOMIZABLE PUZZLES AND HOMOMORPHIC ENCRYPTION

Here we recall the definitions of randomizable puzzles [53] and we show that they are trivially satisfied by a CPA-secure homomorphic encryption (over $\mathbb{Z}_p$), with statistical circuit privacy [42]. We recall the syntax as defined in [53].

Definition A.1 (Randomizable Puzzle). A randomizable puzzle scheme RP = (PSetup, PGen, PSolve, PRand) with a solution space $\mathcal{S}$ (and a function ϕ acting on $\mathcal{S}$) consists of four algorithms defined as:

$(\mathsf{pp}, \mathsf{td}) \leftarrow \mathsf{PSetup}(1^n)$**:** is a PPT algorithm that on input security parameter 1^n, outputs public parameters pp and a trapdoor td.

$Z \leftarrow \mathsf{PGen}(\mathsf{pp}, \zeta)$**:** is a PPT algorithm that on input public parameters pp and a puzzle solution ζ, outputs a puzzle Z.

$\zeta := \mathsf{PSolve}(\mathsf{td}, Z)$**:** is a *DPT* algorithm that on input a trapdoor td and puzzle Z, outputs a puzzle solution ζ.

$(Z', r) \leftarrow \mathsf{PRand}(\mathsf{pp}, Z)$**:** is a PPT algorithm that on input public parameters pp and a puzzle Z (which has a solution ζ), outputs a randomization factor r and a randomized puzzle Z' (which has a solution $\phi(\zeta, r)$).

It is not hard to see that a linearly homomorphic encryption scheme (KGen, Enc, Dec) matches the syntax of a randomizable puzzle, setting pp to the encryption key and td to be the decryption key. For the PRand algorithm, we can sample a random $r \leftarrow\!\$\, \mathbb{Z}_p$ and compute

$$\mathsf{Enc}(\mathsf{ek}, \zeta) \circ \mathsf{Enc}(\mathsf{ek}, r) = c$$

which is an encryption of $\phi(\zeta, r) = \zeta + r$. Next we recall the definition of security for randomizable puzzles.

Definition A.2 (Security). A randomizable puzzle scheme RP is secure, if there exists a negligible function negl, such that

$$\Pr\left[\zeta \leftarrow \mathcal{A}(\mathsf{pp}, Z) \;\middle|\; \begin{array}{l} (\mathsf{pp}, \mathsf{td}) \leftarrow \mathsf{PSetup}(1^n) \\ \zeta \leftarrow\!\$\, \mathcal{S}, Z \leftarrow \mathsf{PGen}(\mathsf{pp}, \zeta) \end{array}\right] \leq \mathsf{negl}(n).$$

This follows as an immediate application of CPA-security (in fact, even the weaker one-wayness suffices) of the encryption scheme. Finally we recall the notion of privacy.

Definition A.3 (Privacy). A randomizable puzzle scheme RP is private if for every PPT adversary $\mathcal{A}$ there exists a negligible function negl such that:

$$\Pr[\mathsf{RPRandSec}_{\mathcal{A},\mathsf{RP}}(n) = 1] \leq 1/2 + \mathsf{negl}(n)$$

where the experiment $\mathsf{RPRandSec}_{\mathcal{A},\mathsf{RP}}$ is defined as follows:

- $(\mathsf{pp}, \mathsf{td}) \leftarrow \mathsf{PSetup}(1^n)$
- $((Z_0, \zeta_0), (Z_1, \zeta_1)) \leftarrow \mathcal{A}(\mathsf{pp}, \mathsf{td})$
- $b \leftarrow\!\$\, \{0, 1\}$
- $(Z_0', r_0) \leftarrow \mathsf{PRand}(\mathsf{pp}, Z_0)$
- $(Z_1', r_1) \leftarrow \mathsf{PRand}(\mathsf{pp}, Z_1)$
- $b' \leftarrow \mathcal{A}(\mathsf{pp}, \mathsf{td}, Z_b')$
- Return $\mathsf{PSolve}(\mathsf{td}, Z_0) = \zeta_0 \wedge \mathsf{PSolve}(\mathsf{td}, Z_1) = \zeta_1$ $\wedge\, b = b'$

Recall that circuit privacy implies that the distribution induced by $\mathsf{Enc}(\mathsf{ek}, \zeta) \circ \mathsf{Enc}(\mathsf{ek}, r)$ is statistically close to that induced by a a fresh encryption $\mathsf{Enc}(\mathsf{ek}, \zeta + r)$. This implies that privacy is satisfies in a statistical sense. Thus we can state the following.

Lemma A.4. *Assuming that* (KGen, Enc, Dec) *is a linearly homomorphic encryption with statistical circuit privacy, the there exists a randomizable puzzle with statistical privacy.*

B A^{2}L PROTOCOL

We recall the formal description of Puzzle promise (Figure 12) and Puzzle solver (Figure 13) of A^{2}L. We report their protocol after translating the same to our syntax for consistency. The setup algorithm is the same as in Figure 9 and the open algorithm is given in Figure 14.

C SECURITY PROOFS

Proof of Lemma 4.8.

Proof. We give a proof by reduction. Let $\mathcal{A}$ be a PPT adversary with non-negligible advantage in the OM-CCA-A2L game. We now construct an adversary $\mathcal{R}$ which uses $\mathcal{A}$ to break the security of OMDL.

$\mathcal{R}$ is given $(h_1, \dots, h_{q+1}) = (g^{r_1}, \dots, g^{r_{q+1}})$ by the OMDL game. It will run $\mathcal{A}$ to attempt to obtain the $q + 1$ discrete logarithms to win the game. Crucially, $\mathcal{R}$ must simulate $\mathcal{A}$'s oracle access to $\mathcal{O}^{\mathsf{A^2L}}_{\mathsf{sk},\Pi_\mathsf{E},\Pi_\mathsf{ADP}}$, which consists of at most q successful queries (but unlimited $\perp$ queries), while making at most q queries (of *any* kind) to its oracle $\mathsf{DL}(\cdot)$.

$\mathcal{R}$ proceeds as follows. First, it samples $q+1$ uniform λ-bit strings $(c_1^*, \dots, c_{q+1}^*)$. Note that these are identically distributed to outputs of $\mathcal{O}^{\mathsf{Enc}}$. It enters $(X_1, c_1^*), \dots, (X_{q+1}, c_{q+1}^*)$ into a table M, where the X_i are random variables. Now it sends $(c_1^*, h_1), \dots, (c_{q+1}^*, h_{q+1})$ to the adversary $\mathcal{A}$.

Any queries $\mathcal{A}$ makes to the encryption scheme oracles ($\mathcal{O}^{\mathsf{Gen}}$, $\mathcal{O}^{\mathsf{Enc}}, \mathcal{O}^{\mathsf{Dec}}, \mathcal{O}^{\mathsf{Add}}$) and their corresponding responses are passed along unchanged by $\mathcal{R}$ but recorded in its table M. Whenever $\mathcal{A}$ makes some query $(\mathsf{vk}_i, m_i, k_i, c_i, \tilde{\sigma}_i)$ to $\mathcal{O}^{\mathsf{A^2L}}$, $\mathcal{R}$ first checks that vk_i is in the support of $\Pi_{\mathsf{ADP}}.\mathsf{KGen}(1^\lambda)$ (this is a publicly checkable predicate since the valid verification keys are defined to be all group elements). After this, it acts in one of four ways:

(1) If $c_i = c_j^*$ and $k_i = h_j$ for some j, it checks $\mathsf{PreVf}(\mathsf{vk}_i, m_i, k_i, c_i) = 1$. If not, it returns $\perp$; otherwise, it queries $\mathsf{DL}(h_j)$ to get x_j and returns $\Pi_{\mathsf{ADP}}.\mathsf{Adapt}(\tilde{\sigma}_i, x_j)$ to $\mathcal{A}$.

(2) If $c_i = c_j^*$ but $k_i \neq h_j$, $\mathcal{R}$ sends $\perp$ to $\mathcal{A}$.

(3) If $(\cdot, c_i) \notin M$, $\mathcal{R}$ sends $\perp$ to $\mathcal{A}$.

```
Public parameters: group description $(\mathbb{G}, g, q)$, message $m_{HB}$

PPromise$\langle H(\mathsf{dk}_H, \mathsf{sk}^H_{HB}), \cdot\rangle$
    $s \leftarrow\$ \mathbb{Z}_p, Y := g^s$
    $c \leftarrow \Pi_{\mathsf{E}}.\mathsf{Enc}(\mathsf{ek}_H, s)$
    $\pi_s \leftarrow \mathsf{NIZK.P}((\mathsf{ek}_H, Y, c), s)$
    $\tilde{\sigma}^H_{HB} \leftarrow \Pi_{\mathsf{ADP}}.\mathsf{PreSig}(\mathsf{sk}^H_{HB}, m_{HB}, Y)$
                         $\xrightarrow{Y, c, \pi_s, \tilde{\sigma}^H_{HB}}$
    return $\bot$

PPromise$\langle \cdot, B(\mathsf{ek}_H, \mathsf{vk}^H_{HB})\rangle$
    If $\mathsf{NIZK.V}((\mathsf{ek}_H, Y, c), \pi_s) \neq 1$ then return $\bot$
    If $\Pi_{\mathsf{ADP}}.\mathsf{PreVf}(\mathsf{vk}^H_{HB}, m_{HB}, Y, \tilde{\sigma}^H_{HB}) \neq 1$ then
        return $\bot$
    $r \leftarrow\$ \mathbb{Z}_q, Y' := Y \cdot g^r$
    $c' \leftarrow \Pi_{\mathsf{E}}.\mathsf{Rand}(c, r)$
    Set $\tau := (r, m_{HB}, \tilde{\sigma}^H_{HB}, (Y, c), (Y', c'))$
    return $\tau$
```

Figure 12: Puzzle promise protocol of A^{2}L

```
Public parameters: group description $(\mathbb{G}, g, q)$, message $m_{AH}$

PSolver$\langle A(\mathsf{sk}^A_{AH}, \mathsf{ek}_H, \tau), \cdot\rangle$
    Parse $\tau := (\cdot, \cdot, \cdot, \cdot, (Y', c'))$
    $r' \leftarrow\$ \mathbb{Z}_p, Y'' := Y' \cdot g^{r'}$
    $c'' \leftarrow \Pi_{\mathsf{E}}.\mathsf{Rand}(c', r')$
    $\tilde{\sigma}^A_{AH} \leftarrow \Pi_{\mathsf{ADP}}.\mathsf{PreSig}(\mathsf{sk}^A_{AH}, m_{AH}, Y'')$
                         $\xrightarrow{Y'', c'', \tilde{\sigma}^A_{AH}}$

PSolver$\langle \cdot, H(\mathsf{dk}_H, \mathsf{vk}^A_{AH})\rangle$
    $s'' \leftarrow \Pi_{\mathsf{E}}.\mathsf{Dec}(\mathsf{dk}_H, c'')$
    $\sigma^A_{AH} \leftarrow \Pi_{\mathsf{ADP}}.\mathsf{Adapt}(\tilde{\sigma}^A_{AH}, s'')$
    If $\Pi_{\mathsf{ADP}}.\mathsf{Vf}(\mathsf{vk}^A_{AH}, m_{AH}, \sigma^A_{AH}) \neq 1$ then
        return $\bot$
                         $\xleftarrow{\sigma^A_{AH}}$

PSolver$\langle A(\mathsf{sk}^A_{AH}, \mathsf{ek}_H, \tau), \cdot\rangle$ (continued)
    $s'' \leftarrow \Pi_{\mathsf{ADP}}.\mathsf{Ext}(\sigma^A_{AH}, \tilde{\sigma}^A_{AH}, Y'')$
    If $s'' = \bot$ then return $\bot$
    $s' := s'' - r'$
    return $(\sigma^A_{AH}, s')$

PSolver$\langle \cdot, H(\mathsf{dk}_H, \mathsf{vk}^A_{AH})\rangle$ (continued)
    return $\sigma^A_{AH}$
```

Figure 13: Puzzle solver protocol of A^{2}L

```
Open$(\tau, s')$
    Parse $\tau := (r, \cdot, \tilde{\sigma}, \cdot, \cdot)$
    $s := s' - r$
    $\sigma \leftarrow \Pi_{\mathsf{ADP}}.\mathsf{Adapt}(\tilde{\sigma}, s)$
    return $\sigma$
```

Figure 14: Open algorithm of A^{2}L

(4) Otherwise, let p_i be the plaintext entry corresponding to c_i in M. Notice that, by the linear-only property of the encryption scheme, p_i is a polynomial in $X_1, \ldots, X_{q+1}$ with $\deg(p_i) \leq 1$.
   (a) If $\deg(p_i) = 0$, p_i is some constant value x_j. In this case, $\mathcal{R}$ uses x_j to proceed as the normal $O^{\mathsf{A}^2\mathsf{L}}$ oracle does (checks if the pre-signature verifies and adapts it if so) and sends its output to $\mathcal{A}$.
   (b) If $\deg(p_i) = 1$, define $p_i := \alpha_0 + \alpha_1 X_1 + \ldots + \alpha_n X_{q+1}$. If $k_i = g^{\alpha_0} \prod_{k=1}^{q+1} h_k^{\alpha_k} = g^{p_i}$ and $\mathsf{PreVf}(\mathsf{vk}_i, m_i, k_i, c_i) = 1$, $\mathcal{R}$ uses a query $\mathsf{DL}(k_i)$ to get x_j and outputs $\Pi_{\mathsf{ADP}}.\mathsf{Adapt}(\tilde{\sigma}_i, x_j)$. Otherwise, it sends $\bot$ to $\mathcal{A}$.

Observe that $\mathcal{R}$ returns $\bot$ *without querying* $\mathsf{DL}(\cdot)$ for all $\bot$ A^{2}L-queries $\mathcal{A}$ makes. Thus it makes at most q queries to $\mathsf{DL}(\cdot)$. If $\mathcal{A}$ outputs winning values $(r_1, \ldots, r_{q+1})$, $\mathcal{R}$ outputs the same values, thereby winning the OMDL game. By assumption, $\mathcal{A}$ succeeds with non-negligible probability, and thus $\mathcal{R}$ also wins with non-negligible probability. This violates the OMDL assumption, implying that no such adversary $\mathcal{A}$ can exist. □

Proof of Theorem 4.9.

Proof. We argue about each property separately.

Lemma C.1 (Blindness). *Assuming Π_{NIZK} is sound, the A^{2}L$^+$ scheme is blind in the LOE model.*

Proof. This holds information-theoretically. Fix any two PPromise executions. We now show, via a series of hybrid experiments, that the cases of $b = 0$ and $b = 1$ are statistically close.

Hybrid $\mathcal{H}_0$: Run $\mathsf{ExpBlnd}$ with $b = 0$.

Hybrid $\mathcal{H}_1$: In both runs of $\mathsf{PSolver}$, sample $r \leftarrow_{\$} \mathbb{Z}_q$ and set $Y'' := g^r$ and $c'' \leftarrow \Pi_{\mathsf{E}}(\mathsf{pk}_H, r)$.

Hybrid $\mathcal{H}_2$: Compute c'' and Y'' honestly using τ_1 in the first run of $\mathsf{PSolver}$ and τ_0 in the second run of $\mathsf{PSolver}$.

Hybrid $\mathcal{H}_3$: Run $\mathsf{ExpBlnd}$ with $b = 1$.

Claim 1. *For all PPT adversaries $\mathcal{A}$,*

$$\mathsf{EXEC}_{\mathcal{H}_0,\mathcal{A}} \approx \mathsf{EXEC}_{\mathcal{H}_1,\mathcal{A}}$$

Proof. Y'' is g raised to a uniform element and c'' is an encryption of the same uniform element in both experiments, conditioned on the ciphertext provided by the Hub being well-formed. Thus, any distinguishing advantage necessarily corresponds to a violation of the soundness property of Π_{NIZK}. It follows that the executions are statistically indistinguishable. □

Claim 2. *For all PPT adversaries $\mathcal{A}$,*

$$\mathsf{EXEC}_{\mathcal{H}_1,\mathcal{A}} \approx \mathsf{EXEC}_{\mathcal{H}_2,\mathcal{A}}$$

Proof. This holds by the same logic as Claim 1. □

Claim 3. *For all PPT adversaries $\mathcal{A}$,*

$$\mathsf{EXEC}_{\mathcal{H}_2,\mathcal{A}} \equiv \mathsf{EXEC}_{\mathcal{H}_3,\mathcal{A}}$$

Proof. The change is only syntactical and the executions are identical. □

Hence, the cases of $b = 0$ and $b = 1$ are statistically indistinguishable. □

Lemma C.2 (Unlockability). *Assuming that Π_{ADP} is witness extractable, pre-signature adaptable, and unforgeable the A^2L^+ scheme is unlockable.*

Proof. We consider two cases separately.

$(b_2 \land b_3) = 1$: First, let us consider the case in which $\mathcal{A}$ outputs a valid signature σ^A_{AH} while at the same time $s'' \leftarrow \Pi_{\mathsf{ADP}}.\mathsf{Ext}(\mathsf{vk}^A_{AH}, \tilde{\sigma}^A_{AH}, \sigma^A_{AH}, Y'')$ is not a valid witness for Y''. Then we can give a reduction which breaks witness extractability with non-negligible probability. The reduction samples a uniform element $r \leftarrow_{\$} \mathbb{Z}_q$ and runs $\mathcal{A}$. It sets $Y'' := g^r$ and uses the encryption key $\tilde{\mathsf{ek}}$ output by $\mathcal{A}$ compute $c'' \leftarrow \Pi_{\mathsf{E}}.\mathsf{Enc}(\tilde{\mathsf{ek}}, r)$. In the puzzle solver phase, it sends Y'', c'' and the witness extractability challenge $\tilde{\sigma}$ to $\mathcal{A}$ and outputs the signature σ it receives in response (note that this is perfectly indistinguishable from an honest run of the protocol). Then $\Pi_{\mathsf{ADP}}.\mathsf{Ext}(\mathsf{vk}^A_{AH}, \tilde{\sigma}, \sigma, Y'')$ is not a valid witness for Y'', but this violates the witness extractability of Π_{ADP}, and therefore the probability of this case occurring is negligible.

The above argument establishes that s'' is a valid witness for Y'' with all but negligible probability. Since $Y'' = Y \cdot g^{r+r'} = g^y \cdot g^{r+r'}$, the only valid witness for Y'' is $y + (r + r')$, and therefore $s'' = y + (r + r')$. Hence $y = s'' - (r + r')$ is a valid witness for the statement Y and thus also for Bob's pre-signature $\tilde{\sigma}^H_{HB}$ (recall that in the protocol, Bob explicitly checks the pre-signature validity of $\tilde{\sigma}_{HB}$ with respect to Y). By pre-signature adaptability of Π_{ADP}, we have that $\Pi_{\mathsf{ADP}}.\mathsf{Vf}(\mathsf{vk}^H_{HB}, m, \Pi_{\mathsf{ADP}}.\mathsf{Adapt}(\tilde{\sigma}^H_{HB}, y)) = 1$ with probability 1. Therefore, the adversary succeeds in this case with negligible probability.

$(b_0 = 1) \lor (b_1 = 1)$: In this case, the adversary is able to produce a valid signature on a message without seeing any pre-signature on it. This only happens with negligible probability by the unforgeability of the adaptor signature scheme. □

Lemma C.3 (Unforgeability). *Assuming the hardness of OMDL and that Π_{ADP} is witness extractable and unforgeable, the A^2L^+ scheme is unforgeable in the LOE model.*

Proof. We give a series of hybrid experiments, show they are indistinguishable, and prove by reduction to OM-CCA-A2L that no adversary exists with non-negligible advantage against the final hybrid.

Hybrid $\mathcal{H}_0$: This is the normal game $\mathsf{ExpUnforg}$ (Figure 7).

Hybrid $\mathcal{H}_1$: Simulate all NIZK proofs using $\Pi_{\mathsf{NIZK}}.\mathsf{Sim}$.

Hybrid $\mathcal{H}_2$: If $\exists\, i \in [q]$ such that $\mathsf{Vf}(\mathsf{vk}^H_i, m_i, \sigma_i) = 1$ and $(\mathsf{vk}^H_i, \cdot) \in \mathcal{L}$ but $(\mathsf{vk}^H_i, m_i) \notin \mathcal{L}$, return 0.

Hybrid $\mathcal{H}_3$: If $\exists\, i \in [q]$ such that $\mathsf{Vf}(\mathsf{vk}^H_i, m_i, \sigma_i) = 1$ and $g^{\mathsf{Ext}(\tilde{\sigma}_i, \sigma_i)} \neq Y_i$, return 0.

Claim 4. *For all PPT adversaries $\mathcal{A}$,*

$$\mathsf{EXEC}_{\mathcal{H}_0,\mathcal{A}} \approx \mathsf{EXEC}_{\mathcal{H}_1,\mathcal{A}}$$

Proof. This follows directly from zero-knowledge of Π_{NIZK}. □

Claim 5. *For all PPT adversaries $\mathcal{A}$,*

$$\mathsf{EXEC}_{\mathcal{H}_1,\mathcal{A}} \approx \mathsf{EXEC}_{\mathcal{H}_2,\mathcal{A}}$$

Proof. The hybrids differ only in the case where the attacker returns a valid signature on a message that was not part of the transcript. By the unforgeability of the adaptor signature, this happens only with negligible probability. □

Claim 6. *For all PPT adversaries $\mathcal{A}$,*

$$\mathsf{EXEC}_{\mathcal{H}_2,\mathcal{A}} \approx \mathsf{EXEC}_{\mathcal{H}_3,\mathcal{A}}$$

Proof. Any distinguishing advantage corresponds to the case in which $\mathcal{A}$ outputs some tuple $(\mathsf{vk}^H_i, m_i, \sigma_i)$ such that, for corresponding $(Y_i, \tilde{\sigma}_i)$, $g^{\Pi_{\mathsf{ADP}}.\mathsf{Ext}(\tilde{\sigma}_i, \sigma_i)} \neq Y_i$. In this case, we can give a reduction to witness extractability of Π_{ADP}. The reduction runs the setup as in $\mathcal{H}_3$ and receives a verification key vk from the witness extractability game. It now picks some guess $i^* \leftarrow_{\$} \{1, \ldots q - 1\}$ (where $q - 1$ is the number of queries of the adversary) for the distinguishing index and starts $\mathcal{A}$ on $\tilde{\mathsf{ek}}$, behaving the same way as $\mathcal{H}_3$ for all oracle queries, except for the i^*-th interaction, in which it sets $\mathsf{vk}^H := \mathsf{vk}$. In the execution of $\mathsf{PPromise}$, it sends m_{i^*} to the witness extractability game and receives $\tilde{\sigma}$, which it gives to $\mathcal{A}$ instead of computing $\tilde{\sigma}^H_{HB}$ itself. Once $\mathcal{A}$ terminates and outputs $\{\mathsf{vk}^H_i, m_i, \sigma_i)\}^q_{i=1}$, the reduction sends σ_{i^*} to its game. If it guessed the distinguishing index i^* correctly, this is a winning signature. Suppose the distinguishing advantage is non-negligible. Since the guess is correct with probability $1/(q-1)$, the reduction violates witness extractability also with non-negligible advantage, which is a contradiction. Hence the two experiments must be computationally close. □

Now we give a reduction from hybrid $\mathcal{H}_3$ to OM-CCA-A2L. Suppose there exists an adversary $\mathcal{A}$ with non-negligible success probability in $\mathcal{H}_3$. We give a reduction that uses $\mathcal{A}$ to win the OM-CCA-A2L game. The reduction is given $(c_1, h_1), \ldots, (c_{q+1}, h_{q+1})$. It generates $(\tilde{\mathsf{ek}}, \tilde{\mathsf{dk}}) \leftarrow \Pi_{\mathsf{E}}.\mathsf{KGen}(1^n)$ and $(\mathsf{vk}^H, \mathsf{sk}^H)$ as in $\mathcal{H}_3$ and starts $\mathcal{A}$ on input $\tilde{\mathsf{ek}}$. For $\mathcal{O}$PPromise queries, the reduction follows the same steps as $\mathcal{H}_3$ except it uses a different challenge h_i each time it generates a pre-signature. When $\mathcal{A}$ queries $\mathcal{O}$PSolver, the reduction computes the completed signature σ^A_{AH} as the output of $\mathcal{O}^{\mathsf{A^2L}}$ run on $\mathcal{A}$'s inputs $(\mathsf{vk}^A_{AH}, m', Y'', c'', \sigma^A_{AH})$. Note that since $\mathcal{A}$ makes at most q non-$\perp$ queries to $\mathcal{O}$PSolver, the reduction also makes at most q non-$\perp$ queries to $\mathcal{O}^{\mathsf{A^2L}}$, as the oracles return $\perp$ in exactly the same cases.

Once $\mathcal{A}$ returns $q+1$ tuples $(\mathsf{vk}^H_j, m_j, \sigma_j)$, the reduction computes $r_i \leftarrow \Pi_{\mathsf{ADP}}.\mathsf{Ext}(\mathsf{vk}^H_j, \tilde{\sigma}_i, \sigma_j, h_i) \forall i, j \in [q+1]$ until it has $q+1$ non-$\perp$ values r_i (at most $(q+1)^2$ invocations of the algorithm) and outputs those values. Note that by the definition of $\mathcal{H}_3$, when $\mathcal{A}$ completes successfully, $g^{r_i} = h_i \ \forall i \in [q+1]$. By assumption, the reduction wins the OM-CCA-A2L game with non-negligible probability. This violates OM-CCA-A2L-security of Π_{E} (implied by Lemma 4.8), so no such adversary against $\mathcal{H}_3$ exists. Thus, no adversary with non-negligible success in $\mathsf{ExpUnforg}$ can exist either. □

The theorem follows directly from Lemmas C.1 to C.3. □

Proof of Theorem 5.3.

Proof. We proceed by describing the UC simulator and arguing about indistinguishability from the real execution of the protocol. We consider the cases where the adversary corrupts a different subset of parties separately. We describe the simulator for a single session and the security of the overall interaction is established via a standard hybrid argument.

H Corrupted. We first give a simulator $\mathcal{S}_H$, then give a series of hybrid experiments that gradually change the real experiment (i.e., the construction in Figures 10 and 11) into the ideal experiment given by the interaction of the corrupted H and the simulator $\mathcal{S}_H$, which has access to $\mathcal{F}_{\mathsf{BCS}}$.

Simulator $\mathcal{S}_H$: Upon receiving $(\mathsf{promise{-}req}, B)$ from $\mathcal{F}_{\mathsf{BCS}}$, $\mathcal{S}_H$ proceeds as follows:

(1) Ask the attacker to initiate a session and receive in return $(Y, c, \pi_s, \tilde{\sigma}^H_{HB})$. If $\Pi_{\mathsf{ADP}}.\mathsf{PreVf}(\mathsf{vk}^H_{HB}, m_{HB}, Y, \tilde{\sigma}^H_{HB}) = 1$ and $\mathsf{NIZK.V}((\mathsf{ek}_H, Y, c), \pi_s) = 1$, proceed as in the protocol and send $(\mathsf{promise{-}res}, \top)$ to $\mathcal{F}_{\mathsf{BCS}}$. Otherwise, abort and send $(\mathsf{promise{-}res}, \perp)$.
(2) Receive $(\mathsf{promise}, \mathsf{pid})$ from $\mathcal{F}_{\mathsf{BCS}}$.
(3) Upon receiving $(\mathsf{solve{-}req}, A, \mathsf{pid}')$ from $\mathcal{F}_{\mathsf{BCS}}$ at some later point, sample a uniform element $y' \leftarrow\$ \mathbb{Z}_q$ and generate keys $(\mathsf{vk}^A_{AH}, \mathsf{sk}^A_{AH}) \leftarrow \Pi_{\mathsf{ADP}}.\mathsf{KGen}(1^n)$. Compute $Y' \leftarrow g^{y'}, \tilde{\sigma}^A_{AH} \leftarrow \Pi_{\mathsf{ADP}}.\mathsf{PreSig}(\mathsf{sk}^A_{AH}, m_{AH}, Y')$ and send them to the attacker.
(4) When the attacker initiates the 2PC, run the 2PC simulator to recover its input dk_H. If $\mathsf{ek}_H \neq \Pi_{\mathsf{E}}.\mathsf{Gen}(\mathsf{dk}_H)$, program the output of the 2PC to $\perp$, otherwise to y'.
(5) Receive σ^A_{AH} in response from the attacker and check that $\Pi_{\mathsf{ADP}}.\mathsf{Vf}(\mathsf{vk}^A_{AH}, m_{AH}, \sigma^A_{AH}) = 1$. Additionally check if $\Pi_{\mathsf{ADP}}.\mathsf{Ext}(\tilde{\sigma}^A_{AH}, \sigma^A_{AH}, Y') = y'$. If both checks pass, send $(\mathsf{solve{-}res}, \top)$ to $\mathcal{F}_{\mathsf{BCS}}$, compute $s \leftarrow \Pi_{\mathsf{E}}.\mathsf{Dec}(\mathsf{dk}_H, c)$, and output (σ^A_{AH}, s) as in the protocol; otherwise, send $(\mathsf{solve{-}res}, \perp)$ and abort.
(6) If, at any point before the successful completion of step 4, the attacker produces a valid signature σ^A_{AH}, or at any point in the protocol (including after step 4), a valid signature on a message $m'_{AH} \neq m_{AH}$, send $(\mathsf{solve{-}res}, \perp)$ to $\mathcal{F}_{\mathsf{BCS}}$ and abort.

Hybrid $\mathcal{H}_0$: This corresponds to the real protocol (Figures 10 and 11).

Hybrid $\mathcal{H}_1$: Simulate the 2PC (Fig. 11, line 6) and send the output z to H.

Hybrid $\mathcal{H}_2$: Replace Y' with $Y'' := g^{y'}$ where $y' \leftarrow\$ \mathbb{Z}_q$ (Fig. 11, line 2). If $\Pi_{\mathsf{E}}.\mathsf{Gen}(\mathsf{dk}_H) = \mathsf{ek}_H$, send y' to H instead of z; otherwise, send $\perp$.

Hybrid $\mathcal{H}_3$: Abort if $z' \neq y'$ (after line 12 of Fig. 11).

Hybrid $\mathcal{H}_4$: Abort if any valid signature σ^A_{AH} is received on a different message $m'_{AH} \neq m_{AH}$ or on any message before the 2PC has successfully completed.

Claim 7. *For all PPT distinguishers $\mathcal{E}$,*

$$\mathsf{EXEC}_{\mathcal{H}_0,\mathcal{A},\mathcal{E}} \approx \mathsf{EXEC}_{\mathcal{H}_1,\mathcal{A},\mathcal{E}}$$

Proof. This follows directly from the security of the 2PC protocol. □

Claim 8. *For all PPT distinguishers $\mathcal{E}$,*

$$\mathsf{EXEC}_{\mathcal{H}_1,\mathcal{A},\mathcal{E}} \approx \mathsf{EXEC}_{\mathcal{H}_2,\mathcal{A},\mathcal{E}}$$

Proof. By the uniqueness of the decryption key and correctness of Π_{E}, $\mathsf{ek}_H = \Pi_{\mathsf{E}}.\mathsf{Gen}(\mathsf{dk}_H)$ implies $\Pi_{\mathsf{E}}.\mathsf{Dec}(\mathsf{dk}_H, \Pi_{\mathsf{Enc}}.\mathsf{Enc}(\mathsf{ek}_H, m)) = m$ for all m in the message space. Thus, the output of the 2PC z is necessarily $s + r$, where $s \in \mathbb{Z}_q$ such that $c = \Pi_{\mathsf{E}}.\mathsf{Enc}(\mathsf{ek}_H, s) \wedge Y = g^s$ (this is guaranteed by the NIZK). Since r is uniformly random, y' is identically distributed to $z = s + r$. The same holds for Y'' and $Y' = Y \cdot g^r$. Furthermore, it still holds that y' is the discrete logarithm of Y'' (cf. z and Y'). □

Claim 9. *For all PPT distinguishers $\mathcal{E}$,*

$$\mathsf{EXEC}_{\mathcal{H}_2,\mathcal{A},\mathcal{E}} \approx \mathsf{EXEC}_{\mathcal{H}_3,\mathcal{A},\mathcal{E}}$$

Proof. If $z' \neq y'$, by the uniqueness of dlog witnesses $g^{z'} \neq Y''$. By the witness extractability of Π_{ADP}, $\Pr[g^{z'} \neq Y'' \wedge \Pi_{\mathsf{ADP}}.\mathsf{Vf}(\mathsf{vk}^A_{AH}, m_{AH}, \sigma^A_{AH}) = 1]$ is negligible, so the abort only happens with negligible probability. □

Claim 10. *For all PPT distinguishers $\mathcal{E}$,*

$$\mathsf{EXEC}_{\mathcal{H}_3,\mathcal{A},\mathcal{E}} \approx \mathsf{EXEC}_{\mathcal{H}_4,\mathcal{A},\mathcal{E}}$$

Proof. Any distinguishing advantage implies a case in which $\mathcal{A}$ outputs some valid signature σ^A_{AH} for some message m'_{AH} for which it has potentially been given a presignature $\tilde{\sigma}^A_{AH}$ and corresponding statement Y. This signature is a winning instance in the unforgeability experiment for Π_{ADP}, but by assumption this only occurs with negligible probability, and so the distinguishing advantage must be negligible. Therefore the experiments are statistically close. □

Claim 11. *For all PPT distinguishers $\mathcal{E}$,*

$$\mathsf{EXEC}_{\mathcal{H}_4,\mathcal{A},\mathcal{E}} \equiv \mathsf{EXEC}_{\mathcal{F}_{\mathsf{BCS}},\mathcal{S},\mathcal{E}}$$

Proof. $\mathcal{H}_4$ is identical to the ideal world. □

A,B Corrupted. Again, we give a simulator $\mathcal{S}_{AB}$ that interacts with $\mathcal{F}_{\mathsf{BCS}}$ and show by a series of hybrids that our protocol is indistinguishable from ideal experiment in which the corrupted parties interact with the simulator $\mathcal{S}_{AB}$.

Simulator $\mathcal{S}_{AB}$: When a recipient Bob indicates he would like to initiate a transaction, $\mathcal{S}_{AB}$ proceeds as follows:

(1) Send $(\mathsf{PPromise}, A)$ to $\mathcal{F}_{\mathsf{BCS}}$.
(2) Upon receiving $(\mathsf{promise}, (\mathsf{pid}, \mathsf{pid}'))$ from $\mathcal{F}_{\mathsf{BCS}}$, sample a uniform value $s \leftarrow\$ \mathbb{Z}_q$ and compute $Y \leftarrow g^s$. Generate keys $(\mathsf{ek}_H, \mathsf{dk}_H) \leftarrow \Pi_{\mathsf{E}}.\mathsf{KGen}(1^n)$ and $(\mathsf{vk}^H_{HB}, \mathsf{sk}^H_{HB}) \leftarrow \Pi_{\mathsf{ADP}}.\mathsf{KGen}(1^n)$; let $c \leftarrow \Pi_{\mathsf{E}}.\mathsf{Enc}(\mathsf{ek}_H, 0)$ and $\tilde{\sigma}^H_{HB} \leftarrow \Pi_{\mathsf{ADP}}.\mathsf{PreSig}(\mathsf{sk}^H_{HB}, m_{HB}, Y)$. Simulate the NIZK $\pi_s \leftarrow \mathsf{NIZK.Sim}(\mathsf{td}, (\mathsf{ek}_H, Y, c))$. Finally, pre-compute $\sigma^H_{HB} \leftarrow \Pi_{\mathsf{ADP}}.\mathsf{Adapt}(\tilde{\sigma}^H_{HB}, s)$ and save $((\mathsf{pid}, \mathsf{pid}'), (Y, c, s, \sigma^H_{HB}), \bot)$ into a table $\mathcal{P}$. Send $(Y, c, \pi_s, \tilde{\sigma}^H_{HB})$ to the attacker (who is impersonating Bob).
(3) At a later point in time, the attacker sends $(Y', \tilde{\sigma}^A_{AH})$ on behalf of Alice. If $\Pi_{\mathsf{ADP}}.\mathsf{PreVf}(\mathsf{vk}^A_{AH}, m_{AH}, Y', \tilde{\sigma}^A_{AH}) \neq 1$, abort.
(4) When the attacker initiates the 2PC, run the 2PC simulator to recover its inputs (c^*, r^*); compute the result $(\bot)$ and return it to the attacker.
(5) Depending on whether or not $c^* \in \mathcal{P}$ do the following:
   (a) If $c^* \in \mathcal{P}$, retrieve the corresponding Y, s, and pid'. Check that $Y' = Y \cdot g^{r^*}$ (if not, abort); send $\Pi_{\mathsf{ADP}}.\mathsf{Adapt}(\tilde{\sigma}^A_{AH}, s + r^*)$ to the attacker masquerading as Alice and $(\mathsf{PSolver}, B, \mathsf{pid}')$ to $\mathcal{F}_{\mathsf{BCS}}$. Update the last element of the entry in $\mathcal{P}$ to $\top$.
   (b) If $c^* \notin \mathcal{P}$, compute $z' \leftarrow \Pi_{\mathsf{E}}.\mathsf{Dec}(\mathsf{dk}_H, c^*) + r^*$. Check that $Y' = g^{z'}$ (if not, abort) and send $\Pi_{\mathsf{ADP}}.\mathsf{Adapt}(\tilde{\sigma}^A_{AH}, z')$ to the attacker. Send nothing to $\mathcal{F}_{\mathsf{BCS}}$. (Note that this corresponds to the case where some party Alice is paying Hub without Bob initiating the interaction, which is something that she can do at any time.)
(6) When the attacker outputs some valid signature σ^H_{HB}, check that the following conditions hold: $\Pi_{\mathsf{ADP}}.\mathsf{Vf}(\mathsf{vk}^H_{HB}, m_{HB}, \sigma^H_{HB}) = 1$ and $((\mathsf{pid}, \cdot), (\cdot, \cdot, \cdot, \sigma^H_{HB}), \top) \in \mathcal{P}$. If so, send $(\mathsf{Open}, \mathsf{pid})$ to $\mathcal{F}_{\mathsf{BCS}}$; otherwise, abort.

Hybrid $\mathcal{H}_0$: This corresponds to the real protocol (Figures 10 and 11).

Hybrid $\mathcal{H}_1$: Replace the honestly-computed NIZK π_s (Figure 10, line 4) with a simulated proof.

Hybrid $\mathcal{H}_2$: Simulate the 2PC (Figure 11, line 6).

Hybrid $\mathcal{H}_3$: Add the list $\mathcal{P}$ and step 5 of the simulator (in particular, case 5a) to Figure 11, line 7-10.

Hybrid $\mathcal{H}_4$: Replace c (Figure 10, line 2) with an encryption of zero.

Hybrid $\mathcal{H}_5$: When Bob outputs a valid signature, abort if $(\cdot, (\cdot, \cdot, \cdot, \sigma^H_{HB}), b) \in \mathcal{P}$ and $b \neq \top$.

Claim 12. *For all PPT distinguishers $\mathcal{E}$,*

$$\mathsf{EXEC}_{\mathcal{H}_0,\mathcal{A},\mathcal{E}} \approx \mathsf{EXEC}_{\mathcal{H}_1,\mathcal{A},\mathcal{E}}$$

Proof. This follows directly from the zero-knowledge property of the NIZK. □

Claim 13. *For all PPT distinguishers $\mathcal{E}$,*

$$\mathsf{EXEC}_{\mathcal{H}_1,\mathcal{A},\mathcal{E}} \approx \mathsf{EXEC}_{\mathcal{H}_2,\mathcal{A},\mathcal{E}}$$

Proof. This follows directly from the UC-security of the 2PC protocol. □

Claim 14. *For all PPT distinguishers $\mathcal{E}$,*

$$\mathsf{EXEC}_{\mathcal{H}_2,\mathcal{A},\mathcal{E}} \equiv \mathsf{EXEC}_{\mathcal{H}_3,\mathcal{A},\mathcal{E}}$$

Proof. By definition, for $c^* \in \mathcal{P}$, the corresponding s and Y in $\mathcal{P}$ are $\Pi_{\mathsf{E}}.\mathsf{Dec}(\mathsf{dk}_H, c^*)$ and g^s, respectively. Therefore $z' = s + r^*$ and the case of $c^* \in \mathcal{P}$ is handled in the same way as all cases were in the previous hybrid experiment. □

Claim 15. *For all PPT distinguishers $\mathcal{E}$,*

$$\mathsf{EXEC}_{\mathcal{H}_3,\mathcal{A},\mathcal{E}} \approx \mathsf{EXEC}_{\mathcal{H}_4,\mathcal{A},\mathcal{E}}$$

Proof. Suppose towards a contradiction that $\mathcal{E}$ can distinguish the two executions with nonnegligible probability. We give a reduction to the CCA-security game of Π_{E}. The reduction sets $m_0 := s$ and $m_1 := 0$, sends them to the CCA game, and receives c. It then acts as hub in its interaction with $\mathcal{E}$, computing everything as in Hybrid 3, except for c, which it sets to the ciphertext it received from the game. When it needs to decrypt c^* it uses the CCA decryption oracle. At the end of the execution, based on $\mathcal{E}$'s guess, it outputs a bit to the CCA game (0 if $\mathcal{E}$ guesses $\mathcal{H}_3$, 1 otherwise), which will be correct with nonnegligible advantage. This violates the CCA-security of Π_{E}, so the two executions must be indistinguishable. □

Claim 16. *For all PPT distinguishers $\mathcal{E}$,*

$$\mathsf{EXEC}_{\mathcal{H}_4,\mathcal{A},\mathcal{E}} \approx \mathsf{EXEC}_{\mathcal{H}_5,\mathcal{A},\mathcal{E}}$$

Proof. If $b \neq \top$, Alice did not receive the completed signature σ^A_{AH} for that session and thus cannot recover the secret s to send to Bob. This means Bob's signature σ^H_{HB} was created without knowing the witness for the pre-signature $\tilde{\sigma}^H_{HB}$, which, by aEUF-CMA of Π_{ADP}, can only happen with negligible probability. Thus the abort also only happens with negligible probability and the two experiments are indistinguishable. □

Claim 17. *For all PPT distinguishers $\mathcal{E}$,*

$$\mathsf{EXEC}_{\mathcal{H}_5,\mathcal{A},\mathcal{E}} \equiv \mathsf{EXEC}_{\mathcal{F}_{\mathsf{BCS}},\mathcal{S},\mathcal{E}}$$

Proof. $\mathcal{H}_5$ is identical to the ideal world. □

A,H Corrupted. This case is trivial, as B has no secret information and the simulator therefore simply follows the protocol.

H,B Corrupted. The simulator in this case follows the protocol honestly. If hub publishes a valid signature σ^A_{AH} on a transaction m that is not in the simulator's (acting as Alice) transcript, the simulator aborts. This means that the adversary was able to forge a signature σ^A_{AH} on some transaction m for which it did not previously receive a pre-signature $\tilde{\sigma}^A_{AH}$. By EUF-CMA of the adaptor signature scheme, this case occurs with negligible probability and thus for all PPT distinguishers $\mathcal{E}$, the real world (an honest execution of the protocol) and the ideal world (an interaction with the simulator) are indistinguishable. □

Ideal Functionality $\mathcal{F}_{\mathsf{BCS}}$

Puzzle Promise: On input $(\mathsf{PPromise}, A)$ from B, $\mathcal{F}_{\mathsf{BCS}}$ proceeds as follows:

- Send $(\mathsf{promise{-}req}, B)$ to H and $\mathcal{S}$.
- Receive $(\mathsf{promise{-}res}, b)$ from H.
- If $b = \bot$ then abort.
- Sample $\mathsf{pid}, \mathsf{pid}' \leftarrow\$ \{0,1\}^n$.
- Store the tuple $(\mathsf{pid}, \mathsf{pid}', \bot)$ into $\mathcal{P}$.
- Send $(\mathsf{promise}, (\mathsf{pid}, \mathsf{pid}'))$ to B, $(\mathsf{promise}, \mathsf{pid})$ to H, $(\mathsf{promise}, \mathsf{pid}')$ to A, and inform $\mathcal{S}$.

Puzzle Solver: On input $(\mathsf{PSolver}, B, \mathsf{pid}')$ from A, $\mathcal{F}_{\mathsf{BCS}}$ proceeds as follows:

- If $(\cdot, \mathsf{pid}', \cdot) \notin \mathcal{P}$ then abort.
- Send $(\mathsf{solve{-}req}, A, \mathsf{pid}')$ to H and $\mathcal{S}$.
- Receive $(\mathsf{solve{-}res}, b)$ from H.
- If $b = \bot$ then abort.
- Update entry to $(\cdot, \mathsf{pid}', \top)$ in $\mathcal{P}$.
- Send $(\mathsf{solved}, \mathsf{pid}', \top)$ to A, B and $\mathcal{S}$.

Open: On input $(\mathsf{Open}, \mathsf{pid})$ from B, $\mathcal{F}_{\mathsf{BCS}}$ proceeds as follows:

- If $(\mathsf{pid}, \cdot, b) \notin \mathcal{P}$ or $b = \bot$ then send $(\mathsf{open}, \mathsf{pid}, \bot)$ to B and abort. Else send $(\mathsf{open}, \mathsf{pid}, \top)$ to B.

Figure 15: Ideal functionality $\mathcal{F}_{\mathsf{BCS}}$ (corresponds to $\mathcal{F}_{\mathsf{A^2L}}$ in [53]). Portions related to griefing protection (i.e., registration) have been removed.

D IDEAL FUNCTIONALITY $\mathcal{F}_{\mathsf{BCS}}$

In Appendix D, we describe the ideal functionality $\mathcal{F}_{\mathsf{BCS}}$ that captures the functionality and security of BCS in the UC framework.