Cryptography for Blockchain Applications

Noemi Glaeser

Abstract

Blockchains were introduced in 2008 by Satoshi Nakamoto as a way to implement a trusted but decentralized append-only ledger. This simple functionality has given rise to a plethora of decentralized applications utilizing the blockchain as a public bulleting board. In recent years, it has become clear that this basic functionality is not enough to prevent widespread attacks on both the privacy and security of blockchain users, as evidenced by the blockchain analytics industry and the billions of dollars stolen via cryptocurrency exploits to date. This work explores the role cryptography has to play in the blockchain ecosystem to both enhance user privacy and secure user funds.

Contents

1 Introduction

Bitcoin [Nak08] was the first digital currency to successfully implement a fully trustless and decentralized payment system. todo: ... Ethereum [But14] introduced programmability via *smart contracts* todo: ...

1.1 Model

1.2 Universal Composability (UC) Framework

1.3 Notation and preliminaries

We use $x \overset{\$}{\leftarrow} \mathcal{S}$ to denote sampling an element x uniformly at random from a set $\mathcal{S}$.

Definition 1 (NP Relation)**.** *An NP relation $\mathcal{R}$ is todo: ...*

The language corresponding to $\mathcal{R}$ is defined as $\mathcal{L}_\mathcal{R} := \{Y : \exists y$ such that $(Y, y) \in \mathcal{R}\}$.

Definition 2 (CPA-security)**.** *An encryption scheme Π_E is said to be CPA-secure if todo: ...*

Definition 3 (Rerandomizable encryption)**.** *An rerandomizable encryption scheme Π_E has an additional algorithm* Rand *todo: ...*

Definition 4 (existential unforgeability)**.** *A digital signature scheme Π_ADP is said to be existentially unforgeable (or EUF-CMA-secure) if todo: ...*

2 Privacy-Enhancing Building Blocks

Although cryptocurrencies are often treated as fully anonymous digital currencies, numerous works have shown how to link transactions or even fully deanonymize users in many popular cryptocurrencies [BKP14, BT19, KKM14, MSH$^+$18, KYMM18]. Numerous mitigations have been suggested, including cryptocurrency mixers (described below) and privacy-first cryptocurrencies like Zcash [zca] and Monero [mon].

2.1 Zero-Knowledge Proofs and their trust assumptions

Non-interactive zero-knowledge proofs (NIZKs) are ubiquitous building blocks in many blockchains. Zcash [zca], as indicated by the letter "Z" in its name, relies heavily on a type of NIZK called zkSNARK (zero-knowledge succinct argument of knowledge) to achieve private payments: zkSNARKs are used to prove a party has sufficient funds to make a payment without revealing anything more about those funds [BCG$^+$14]. On Ethereum, (zk-)rollups enhance scalability by leveraging the succinctness of (zk)SNARKs, though they may or may not offer the zero-knowledge property.

To achieve such a high level of succinctness, SNARKs rely on a trusted setup to generate a *common reference string (CRS)*. In keeping with the primary innovation of the blockchain, which is the elimination of a trusted third party (TTP), practitioners use various approaches to minimize the trust in the CRS generation. Zcash uses a multi-party computation ceremony [zca16] with many independent participants to distribute the trust among several parties. Another trust-minimizing approach consists of using SNARKs with universal and updatable CRS [GKM+18, MBKM19, CHM+20, GWC19]. A universal CRS can be reused across applications, avoiding a new complicated setup ceremony for every use. Updatable CRSs allow any participant in a system to contribute randomness to the CRS at any point, including once the CRS is in production use, to enable a "one-out-of-many" trust scenario in which the user must only trust themselves to contribute (and then delete) good randomness to the CRS in order for the whole system to be secure.

An orthogonal concern is maintaining the security of SNARKs when they are composed with other protocols in the complex blockchain ecosystem. Formally, this is modeled by universally composable security via the UC framework [Can01]. Unfortunately, most SNARKs in deployment today are not provably UC-secure. Although compilers to transform any SNARK or NIZK into a UC variant exist [KZM+15, GKO+23], these are not compatible with the aforementioned trust-minimizing properties like updatability. A generic compiler which adds UC-security while maintaining updatability would help ensure confidence in both the trusted setup and the operational security of deployed NIZKs.

2.1.1 Contribution: Circuit-succinct universally composable NIZKs with updatable CRS

In this section, we summarize the contributions and constructions of [AGRS24]. todo: ...

	UC		succinctness-preserving		
	SE	BBE	in $\lvert C\rvert$	in $\lvert w\rvert$	upd. CRS
C∅C∅ [KZM+15]	✓	✓	✓	✗	✗
DS [DS19]	✓	✗	✓	✓	✗
Lamassu [ARS20]	✓	✗	✓	✓	✓
This work [AGRS24]	✓	✓	✓	✗	✓
Concurr. work [GKO+23]	✓	✓	✓	✓	✗

Table 1: Comparison with concurrent and previous work.

3 Privacy-Enhancing Applications

3.1 Cryptocurrency Mixers

(Parts of this section are taken/adapted from [GMM+22].)

Noemi: How to position this? Our focus will be on *off-chain* mixers, whose focus is on enabling *scalability* and *interoperability* rather than privacy... todo: Talk about how these also solve scalability and interoperability?

todo: citations [RMK14, SNBB19, TLK+18, BNM+14], Bitcoin (CoinJoin, CoinShuffle), Bolt, Blindcoin, Mixcoin Cryptocurrency mixers add a measure of k-anonymity to cryptocurrency tokens by employing a central party, or *mixer*, to shuffle the tokens among users of the service. Users deposit their coins into the service, and later retrieve them again (using a different address, otherwise anonymity is trivially broken). Any particular token (retrieved from the mixer) cannot be tied to a particular source (user who deposited money from the mixer): each of the k users is equally likely to be the source of a given token. For security, a mixer must offer *atomicity*, i.e., a user pays $c + \epsilon$ coins if and only if the "recipient" (normally the same user, but under a new address) is paid c coins (ϵ is a parameter which represents the mixer and transaction fees). todo: exit scams

Mixers come in two flavors: on- and off-chain mixers. On-chain mixers todo: [] are simply accounts into which users can deposit coins and later retrieve them (or allow another party to retrieve them) by redeeming some token, with atomicty enforced via an on-chain script.Noemi: check this

Off-chain mixers normally require more complicated protocols to enforce the atomicity requirement without the scripting functionality offered by the underlying blockchain. One line of work, initiated by TumbleBit [HAB+17], uses a protocol paradigm which we refer to as a *synchronization puzzle*. A synchronization puzzle is a three-party protocol between a sender (Alice), a mixer (Hub), and a recipient (Bob) todo: insert figure. Synchronization puzzle protocols consist of four steps: (1) Hub and Bob execute *puzzle promise* phase with respect to some message m_{HB}, which outputs a puzzle τ containing a hidden signature s on m_{HB}. (2) Bob sends τ to Alice via a private channel, who (3) uses it to execute the *puzzle solver* phase with Hub with respect to another message m_{AH}. At the end of this phase, Alice obtains the signature s on m_{HB} and Hub learns s' on m_{AH}. To conclude, (4) Alice sends s to Bob. A synchronization puzzle protocol should satisfy the following properties:

- **Blindness**: In the puzzle solver phase, Hub *blindly* helps solve τ, i.e., the phase should not reveal anything about τ to Hub. (This keeps Alice and Bob unlinkable from the point of view of Hub.)
- **Unlockability**: If the puzzle solver phase completes successfully, s must be a valid secret for τ. (This ensures that the Hub cannot learn s' without revealing s.)
- **Unforgeability**: Bob cannot output a valid signature s on m_{HB} before

the puzzle solver phase completes. (This ensures Bob cannot learn s without Hub learning s'.)

To use a synchronization puzzle to realize an atomic payment, the parties set $m_{AH} : (A \xrightarrow{c+\epsilon} H)$ and $m_{HB} : (H \xrightarrow{c} B)$, where $(U_i \xrightarrow{c} U_j)$ denotes a payment of c coins from user U_i to U_j. Then s' and s are set to the signatures authorizing m_{AH} and m_{HB}, respectively. Thus, at the completion of the protocol Alice will have sent c coins to Bob (and paid a fee of ϵ to Hub).

Tumblebit realizes a synchronization puzzle via todo: ???.

3.1.1 Anonymous Atomic Locks (A^{2}L)

A follow-up work [TMM21] introduces Anonymous Atomic Locks (A^{2}L), a protocol for off-chain coin mixing which decreases the required capabilities of the underlying blockchain Noemi: improve wording?. Before we describe its approach to instantiating synchronization puzzles, we introduce the notion of *adaptor signatures*, which will be used as a building block:

Definition 5 (adaptor signature [AEE$^+$21])**.** *An adaptor signature scheme* $\Pi_{\mathsf{ADP}} := (\mathsf{KGen}, \mathsf{PreSig}, \mathsf{PreVrfy}, \mathsf{Adapt}, \mathsf{Vrfy}, \mathsf{Ext})$ *is defined with respect to a digital signature scheme* Π_{ADP} *and an NP relation* $\mathcal{R}$*:*

$\mathsf{KGen}(1^\lambda) \to (\mathsf{vk}, \mathsf{sk})$**:** *The key generation algorithm is the same as in the underlying digital signature scheme, i.e.,* $\Pi_{\mathsf{ADP}}.\mathsf{KGen}$*.*

$\mathsf{PreSig}(\mathsf{sk}, m, Y) \to \tilde{\sigma}$**:** *The pre-signing algorithm takes as input a signing key* sk*, message* m*, and instance* Y *of the relation* $\mathcal{R}$ *and returns a pre-signature* $\tilde{\sigma}$*.*

$\mathsf{PreVrfy}(\mathsf{vk}, m, Y, \tilde{\sigma}) \to \{0, 1\}$**:** *The pre-verification algorithm checks that a pre-signature is well-formed.*

$\mathsf{Adapt}(\tilde{\sigma}, y) \to \sigma$**:** *Given a witness* y *for the instance* Y*, this algorithm adapts the pre-signature* $\tilde{\sigma}$ *into a valid signature* σ*.*

$\mathsf{Vrfy}(\mathsf{vk}, m, \sigma) \to \{0, 1\}$**:** *The verification algorithm is the same as in the underlying digital signature scheme, i.e.,* $\Pi_{\mathsf{ADP}}.\mathsf{Vrfy}$*.*

$\mathsf{Ext}(\tilde{\sigma}, \sigma, Y) \to y$**:** *Given a pre-signature* $\widetilde{sigma}$ *and a signature* σ *generated with respect to some instance* Y*, the extract algorithm outputs the corresponding witness* y *such that* $(Y, y) \in \mathcal{R}$*.*

An adaptor signature scheme should satisfy the following properties: *pre-signature correctness*, which guarantees that for all instances $Y \in \mathcal{L}_Rel$ and honestly generated $\tilde{\sigma}, \sigma$, the pre-signature and signature pass (pre-)verification and the extracted witness $y' \leftarrow \mathsf{Ext}(\tilde{\sigma}, \sigma, Y)$ should satisfy $(Y, y') \in \mathcal{R}$; *unforgeability*, which is a straightforward extension of the standard existential unforgeability notion (EUF-CMA) for digital signatures; *pre-signature adaptability*, which states that for any $Y \in \mathcal{L}_{\mathcal{R}}$ and corresponding pre-signature $\tilde{\sigma}$, pre-verification implies that $\tilde{\sigma}$ can be adapted to a verifying signature σ; and *witness extractability*, which says that it is difficult for an adversary to adapt an honestly-generated pre-signature $\tilde{\sigma}$ into a signature σ which verifies but where $y' \leftarrow \mathsf{Ext}(\tilde{\sigma}, \sigma, Y)$ such that $(Y, y') \notin \mathcal{L}_{\mathcal{R}}$. We refer the reader to [GMM+22] for formal definitions.

Anonymous Atomic Locks (A^{2}L) [TMM21] uses a rerandomizable CPA-secure encryption scheme Π_{E} and an *adaptor signature* scheme Π_{ADP} to realize a synchronization puzzle which is compatible with a wider range of cryptocurrencies. Specifically, the puzzle promise phase outputs to Bob a ciphertext $c \leftarrow \Pi_{\mathsf{E}}.\mathsf{Enc}(\mathsf{ek}_H, y)$ and a pre-signature $\tilde{\sigma}_{HB}$ on m_{HB} with respect to a discrete-log instance $Y := g^y$. Bob sends (c, Y) to Alice, who rerandomizes them using fresh randomness r to $(c', Y') := (\Pi_{\mathsf{E}}.\mathsf{Enc}(\mathsf{ek}_H, y+r), g^y g^r)$. She can now use Y' to produce a pre-signature $\tilde{\sigma}_{AH}$ on m_{AH} and executes the puzzle solver phase with Hub using $c', \tilde{\sigma}_{AH}$. Hub can use sk_H to decrypt c' and obtain $y' := y + r$, which it uses to complete the presignature into a full signature σ_{AH} on m_{AH} (thus completing the left side of the payment). This allows Alice to extract y' via $\Pi_{\mathsf{ADP}}.\mathsf{Ext}(\tilde{\sigma}_{AH}, \sigma_{AH}, Y')$ and use her knowledge of r to recover y, which she sends to Bob. Now Bob uses y to adapt $\tilde{\sigma}_{HB}$ to a full signature σ_{HB} on m_{HB}, thereby completing the right side of the payment.

[TMM21] defines an ideal functionality, which is reproduced as $\mathcal{F}_{\mathsf{BCS}}$ in [GMM+22], and gives a UC proof of security for A2L:

Theorem 1 (Main theorem [TMM21] (paraphrased)[1])**.** *Let* COM *be a secure commitment scheme,* NIZK *be a non-interactive zero-knowledge proof,* Π_{ADP} *be EUF-CMA-secure signature schemes,* $\mathcal{R}$ *be an NP-relation,* Π_{ADP} *be a secure adaptor signature scheme with respect to* Π_{ADP} *and* $\mathcal{R}$*, and* Π_{E} *be a rerandomizable CPA-secure encryption scheme. Then A*^{2}*L UC-realizes the ideal functionality* $\mathcal{F}_{\mathsf{BCS}}$ *assuming anonymous guaranteed delivery channels and a synchronous network.*

3.1.2 Contribution: Insecurity of A^{2}L

In [GMM+22], we analyzed the A^{2}L protocol [TMM21] and found that, in contrast to its claims, it is not secure. Although A^{2}L was proven secure in the universal composability (UC) [Can01] framework, we show that a gap in their formal model allows two constructions which are completely insecure despite meeting their definitions: one admits a key recovery attack and the other allows a colluding sender and recipient to steal coins from the mixer. We will show how to close this gap in section 3.1.3.

In more detail, we show that there exist cryptographic primitives which satisfy the prerequisites of A^{2}L's main theorem, but allow *(a)* a *key recovery attack*, in which a malicious user is able to learn the long-term secret of the hub or *(b)* a *one-more signature attack*, in which a sender and recipient can collude to obtain $\lambda + 1$ tokens from the hub while only sending λ tokens. Both attacks run in polynomial time and succeed with overwhelming probability.

[1] As we will show, this theorem is incorrect.

These instantiations of A^{2}L are specifically crafted allow an attack and do not imply that all instantiations of A^{2}L are broken; however, we cannot prove the security of A^{2}L either. This tension is discussed further in section 3.1.3.

Below we give informal descriptions of both attacks. We refer the reader to [GMM$^+$22] for detailed descriptions and analysis. Both attacks rely on the fact that in the A^{2}L protocol, Hub offers a malicious Alice something very close to a decryption oracle. This oracle, which we refer to as $\mathcal{O}^{\mathsf{A^2L}}$, is programmed with a decryption key dk and takes as input a verification key vk, message m, group element h, ciphertext c, and pre-signature $\tilde{\sigma}$. It computes the plaintext $\tilde{x} \leftarrow \Pi_{\mathsf{E}}.\mathsf{Dec}(\mathsf{dk}, c)$ and attempts to use it to adapt $\tilde{\sigma}$, i.e., computes $\sigma' \leftarrow \Pi_{\mathsf{ADP}}.\mathsf{Adapt}(\tilde{\sigma}, \tilde{x})$. If it is successful, i.e., $\Pi_{\mathsf{ADP}}.\mathsf{Vrfy}(\mathsf{vk}, m, \sigma') = 1$ or equivalently $\tilde{x} = x$ Noemi: check this, it returns σ'; otherwise it returns $\perp$. Note that the sender (Alice) can easily generate inputs to query the oracle: vk is the querier's own verification key, m can be any arbitrary message in the message space, and generating $\tilde{\sigma}$ that is valid when adapted with the (unknown) value x requires only knowledge of the party's own signing key and a value $h = g^x$. The counterexamples below make use of the fact that σ' implicitly reveals $\tilde{x} = \Pi_{\mathsf{ADP}}.\mathsf{Ext}(\tilde{\sigma}, \sigma, h)$. This leakage is not addressed in A^{2}L's proof of security.

Key recovery attack. We show how to recover Hub's decryption key dk with λ queries to $\mathcal{O}^{\mathsf{A^2L}}$ when A^{2}L is instantiated with an encryption scheme Π_{E} which, in addition to being re-randomizable and CPA-secure (as required by Theorem 1) has the following properties:

- linearly homomorphic over $\mathbb{Z}_p$
- circular secure for bit encryption, i.e., CPA-secure even given the bitwise encryption of the decryption key
- the aforementioned ciphertexts $(c_1, \ldots, c_\lambda) := (\mathsf{Enc}(\mathsf{ek}, \mathsf{dk}_1), \ldots, \mathsf{Enc}(\mathsf{ek}, \mathsf{dk}_\lambda))$ are included in the encryption key ek

Note that even with these additional assumptions, Π_{E} still satisfies the conditions of Theorem 1, and yet A^{2}L instantiated with such a scheme is insecure. In particular, a malicious Alice can use λ queries to recover Hub's long-term decryption key dk. The intuition of the attack is as follows: Alice samples a witness $x \overset{\$}{\leftarrow} \mathbb{Z}_p$ and computes the ciphertext $c \leftarrow \Pi_{\mathsf{E}}.\mathsf{Enc}(\mathsf{ek}, x)$. Now she can use the provided bitwise encryption of dk to homomorphically compute $c_i' := \Pi_{\mathsf{E}}.\mathsf{Enc}(\mathsf{ek}, x + \mathsf{dk}_i)$ for each bit of the key. Getting the remaining inputs to the oracle is easy, since she can use her signing key to compute $h := g^x$ and $\tilde{\sigma} \leftarrow \Pi_{\mathsf{ADP}}.\mathsf{PreSig}(\mathsf{sk}, m, h)$ for an arbitrary m. At this point, she can query $\mathcal{O}^{\mathsf{A^2L}}(\mathsf{vk}_A, m, h, c_i, \tilde{\sigma})$ for $i = 1, \ldots, \lambda$. The oracle returns $\perp$ if and only if $\mathsf{dk}_i = 1$, since $g^{x+1} \neq h = g^x$; otherwise, the oracle outputs an adapted signature σ', and Alice learns that $\mathsf{dk}_i = 0$. This attack succeeds with probability 1. Recall that obtaining a non-$\perp$ response from the oracle is equivalent to authorizing a payment of c coins from Alice to Hub, so it costs $nc \leq n\lambda$ coins (where n is the number of 0 bits in dk).

One-more signature attack. Next, we show how to steal 1 coin from the hub for every λ successful payments, i.e., learn $\lambda + 1$ signature witnesses for every λ non-$\perp$ queries to $\mathcal{O}^{\mathsf{A^2L}}$. This attack works when A^{2}L is instantiated with an encryption scheme Π_E which, in addition to being re-randomizable and CPA-secure (as required by Theorem 1) has the following properties:

- linearly homomorphic over $\mathbb{Z}_p$
- supports homomorphic evaluation of the *conditional bit flip* (CFlip) function, defined as $\Pi_\mathsf{E}.\mathsf{CFlip}(\mathsf{ek}, i, \Pi_\mathsf{E}.\mathsf{Enc}(\mathsf{ek}, x)) := \Pi_\mathsf{E}.\mathsf{Enc}(\mathsf{ek}, y)$ where

  $$y = \begin{cases} x & \text{if } x_i = 0 \\ x \oplus e_i & \text{if } x_i = 1 \end{cases}$$

  and e_i is the i-th unit vector.

Again, even with these additional assumptions, Π_E still satisfies the conditions of Theorem 1, and yet A^{2}L instantiated with such a scheme is insecure. The intuition of the attack is as follows: given $\lambda + 1$ puzzle promise instances ($c_j := \Pi_\mathsf{E}.\mathsf{Enc}(\mathsf{ek}, x_j), h_j := g^{x_j}$) for $j = 1, \ldots, \lambda + 1$, Alice will attempt to learn the bits of x_1 by conditionally flipping them one at a time. Each query i consists of a ciphertext c' containing a random linear combination of the c_j's, where c_1's ith bit is also conditionally flipped, and h' which is the same random linear combination of h_j's *without* x_1's ith bit being conditionally flipped. If the oracle response is $\perp$, c' and h' were inconsistent which means the ith bit of x_1 was a 1. Otherwise, if the oracle responds with a non-$\perp$ value y_i, the ith bit of x_1 was a 0 and Alice has additionally learned the value of one random linear combination of the x_j's. By the time λ non-$\perp$ queries have been made, Alice has learned all the bits of x_1 and also has λ linear equations with λ unknowns. Since the coefficients are uniformly chosen, the equations are, with all but negligible probability, linearly independent; and since $\mathbb{Z}_p$ is a field, the solution is uniquely determined and can be found efficiently via Gaussian elimination.

3.1.3 Contribution: Formalizing Blind Conditional Signatures

In light of our attacks, the natural question is whether we can establish formally rigorous security guarantees for an (appropriately patched) A^{2}L protocol. 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.

To place the synchronization puzzle on firmer foundations, in [GMM+22] we introduce a new primitive called blind conditional signtaures (BCS)[2] which captures the core synchronization puzzle functionality. We give game-based security definitions for BCS and show how to modify A^2L to obtain a new protocol, A^2L^+, which meets these definitions. We also give a UC-secure construction of BCS, dubbed A^2L^{UC}, which requires much more complex machinery. In this section, we summarize the remaining contributions and constructions of [GMM+22].

Game-based definitions. Towards establishing a formal analysis of A^2L, we introduce the notion of blind conditional signatures (BCS) which captures the core synchronization puzzle functionality. We propose game-based definitions similar in spirit to the well-established security definitions of regular blind signatures [Cha82, SU17].

Definition 6 (Blind Conditional Signature)**.** *A blind conditional signature* Π_{BCS} $:= (\mathsf{Setup}, \mathsf{PPromise}, \mathsf{PSolver}, \mathsf{Open})$ *is defined with respect to a signature scheme* $\Pi_{\mathsf{ADP}} := (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrfy})$ *and consists of the following efficient algorithms.*

- $(\tilde{\mathsf{ek}}, \tilde{\mathsf{dk}}) \leftarrow \mathsf{Setup}(1^\lambda)$*: The setup algorithm takes as input the security parameter* 1^λ *and outputs a key pair* $(\tilde{\mathsf{ek}}, \tilde{\mathsf{dk}})$*.*

- $(\perp, \{\tau, \perp\}) \leftarrow \mathsf{PPromise}\left\langle \begin{matrix} H\left(\tilde{\mathsf{dk}}, \mathsf{sk}^H, m_{HB}\right) \\ B\left(\tilde{\mathsf{ek}}, \mathsf{vk}^H, m_{HB}\right) \end{matrix} \right\rangle$*: 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* $\perp$ *to* H *and either a puzzle* τ *or* $\perp$ *to* B*.*

- $(\{(\sigma^*, s), \perp\}, \{\sigma^*, \perp\}) \leftarrow \mathsf{PSolver}\left\langle \begin{matrix} A\left(\mathsf{sk}^A, \tilde{\mathsf{ek}}, m_{AH}, \tau\right) \\ H\left(\tilde{\mathsf{dk}}, \mathsf{vk}^A, m_{AH}\right) \end{matrix} \right\rangle$*: 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* vk^A*, and a message* m_{AH}*) and returns to both users either a signature* σ^* *(*A *additionally receives a secret* s*) or* $\perp$*.*

- $\{\sigma, \perp\} \leftarrow \mathsf{Open}(\tau, s)$*: The open algorithm takes as input a puzzle* τ *and a secret* s *and returns a signature* σ *or* $\perp$*.*

Informally, correctness holds if for all honest executions of the protocols, the resulting signatures σ and σ^* verify.

The game-based security guarantees of BCS consist of three properties:

[2]not to be confused with conditional blind signatures [ZGP17].

Blindness. Our definition of blindness is akin to the strong blindness notion of standard blind signatures [Cha82], in which the adversary picks the keys (as opposed to the weak version in which they are chosen by the experiment)[3]. Roughly speaking, it says that two promise/solve sessions cannot be linked together by the hub.[4][5]

Unlockability. This property 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.

Unforgeability. Our definition of unforgeability is inspired by the unforgeability of blind signatures [Cha82]. 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.

The security games are given in Figures 1 to 3. todo: combine figures We define security as the collection of all properties:

Definition 7 (Security)**.** *A blind conditional signature* Π_{BCS} *is secure if it is blind, unlockable, and unforgeable, i.e., if there exist negligible functions* $\mathsf{negl}_1, \mathsf{negl}_2, \mathsf{negl}_3$ *such that for all* $\lambda \in \mathbb{N}$ *and all PPT adversaries* $\mathcal{A}$*, all of*

[3] 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.

[4] 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 [TMM21].

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

```
$\mathsf{ExpBInd}^{\mathcal{A}}_{\Pi_{\mathsf{BCS}}}(\lambda)$

$(\tilde{\mathsf{ek}}, \mathsf{vk}_0^H, \mathsf{vk}_1^H, (m_{HB,0}, m_{AH,0}), (m_{HB,1}, m_{AH,1})) \leftarrow \mathcal{A}(1^\lambda)$
$(\mathsf{vk}_0^A, \mathsf{sk}_0^A) \leftarrow \mathsf{KGen}(1^\lambda)$
$(\mathsf{vk}_1^A, \mathsf{sk}_1^A) \leftarrow \mathsf{KGen}(1^\lambda)$
$\tau_0 \leftarrow \mathsf{PPromise}\left\langle \mathcal{A}(\mathsf{vk}_0^A, \mathsf{vk}_1^A), B(\tilde{\mathsf{ek}}, \mathsf{vk}_0^H, m_{HB,0}) \right\rangle$
$\tau_1 \leftarrow \mathsf{PPromise}\left\langle \mathcal{A}(\mathsf{vk}_0^A, \mathsf{vk}_1^A), B(\tilde{\mathsf{ek}}, \mathsf{vk}_1^H, m_{HB,1}) \right\rangle$
$b \leftarrow \{0, 1\}$
$(\sigma_0^*, s_0) \leftarrow \mathsf{PSolver}\left\langle A\left(\mathsf{sk}_0^A, \tilde{\mathsf{ek}}, m_{AH,0}, \tau_{0\oplus b}\right), \mathcal{A} \right\rangle$
$(\sigma_1^*, s_1) \leftarrow \mathsf{PSolver}\left\langle A\left(\mathsf{sk}_1^A, \tilde{\mathsf{ek}}, m_{AH,1}, \tau_{1\oplus b}\right), \mathcal{A} \right\rangle$
**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 1: Blindness experiment

```
$\mathsf{ExpUnlock}^{\mathcal{A}}_{\Pi_{\mathsf{BCS}}}(\lambda)$

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

Figure 2: Unlockability experiment

the following hold:

$$\Pr\left[\mathsf{ExpBInd}^{\mathcal{A}}_{\Pi_{\mathsf{puzzle}}}(\lambda) = 1\right] \leq \frac{1}{2} + \mathsf{negl}_1(\lambda)$$
$$\Pr\left[\mathsf{ExpUnlock}^{\mathcal{A}}_{\Pi_{\mathsf{BCS}}}(\lambda) = 1\right] \leq \mathsf{negl}_2(\lambda)$$
$$\Pr\left[\mathsf{ExpUnforg}^{\mathcal{A}}_{\Pi_{\mathsf{BCS}}}(\lambda) = 1\right] \leq \mathsf{negl}_3(\lambda)$$

The A^{2}L$^+$ protocol: a secure version of A^{2}L. todo: ... [copied] We now prove that A^{2}L$^+$, our appropriately modified version of A^{2}L, satisfies these definitions. 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 [Gro04] that only models adversaries that are restricted to perform "legal" operations on ciphertexts, similarly to the generic/algebraic group model. While this is far from a complete analysis, it increases our confidence in the security of the system.[6]

[6] 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.

```
$\mathsf{ExpUnforg}^{\mathcal{A}}_{\Pi_{\mathsf{BCS}}}(\lambda)$

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

$\mathcal{O}\mathsf{PP}(m)$

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

$\mathcal{O}\mathsf{PS}(\mathsf{vk}^A, m')$

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

Figure 3: Unforgeability experiment

UC-security. [copied] 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^2L (or A^2L^+) is secure under composition, which is why we prove A^2L^+ 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 [TMM21] is flawed due to this same reason. Thus, we develop a different protocol (called A^2L^{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.

todo: However, even if the encryption scheme could be replaced with a CCA-secure scheme (notwithstanding the well-known and unresolved tension between rerandomizability and CCA-security, which we will touch on later), another problem still stands in the way of UC-security.

todo: A^2L fails to realize the UC functionality: at its core, it relies on ciphertext rerandomizability, but because it also offers a decryption oracle it must at the same time use a CCA-secure encryption scheme. todo: Why do we need game-based defs? A^2L had an independent problem of linkability with the UC proof too. todo: refer reader to A2L and our paper for UC ideal functionality

Constructions. todo: Write both constructions?

3.2 On-chain private voting

3.2.1 Contribution: Cicada, a framework for private non-interactive on-chain auctions and voting

In this section, we summarize the contributions and constructions of [GSZB23].
todo: ...

4 Proposed Work

4.1 Registration-Based Encryption as a Web3 service

Noemi: Unclear if this can be included

4.2 Threshold cryptocurrency wallets in the hot-cold paradigm

5 Timeline

todo:

References

[AEE$^+$21] Lukas Aumayr, Oguzhan Ersoy, Andreas Erwig, Sebastian Faust, Kristina Hostáková, Matteo Maffei, Pedro Moreno-Sanchez, and Siavash Riahi. Generalized channels from limited blockchain scripts and adaptor signatures. In *Advances in Cryptology–ASIACRYPT 2021: 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6–10, 2021, Proceedings, Part II 27*, pages 635–664. Springer, 2021.

[AGRS24] Behzad Abdolmaleki, Noemi Glaeser, Sebastian Ramacher, and Daniel Slamanig. Circuit-succinct universally-composable NIZKs with updatable CRS. In *37th IEEE Computer Security Foundations Symposium*, 2024.

[ARS20] Behzad Abdolmaleki, Sebastian Ramacher, and Daniel Slamanig. Lift-and-shift: Obtaining simulation extractable subversion and updatable SNARKs generically. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 1987–2005. ACM Press, November 2020.

[BCG$^+$14] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474. IEEE Computer Society Press, May 2014.

[BKP14] Alex Biryukov, Dmitry Khovratovich, and Ivan Pustogarov. Deanonymisation of clients in bitcoin P2P network. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 2014*, pages 15–29. ACM Press, November 2014.

[BNM$^+$14] Joseph Bonneau, Arvind Narayanan, Andrew Miller, Jeremy Clark, Joshua A. Kroll, and Edward W. Felten. Mixcoin: Anonymity for bitcoin with accountable mixes. In Nicolas Christin and Reihaneh Safavi-Naini, editors, *FC 2014*, volume 8437 of *LNCS*, pages 486–504. Springer, Heidelberg, March 2014.

[BT19] Alex Biryukov and Sergei Tikhomirov. Deanonymization and linkability of cryptocurrency transactions based on network analysis. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 172–184, 2019.

[But14] Vitalik Buterin. A next-generation smart contract and decentralized application platform. *White paper*, 2014.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001.

[Cha82] David Chaum. Blind signatures for untraceable payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *CRYPTO'82*, pages 199–203. Plenum Press, New York, USA, 1982.

[CHM$^+$20] Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Psi Vesely, and Nicholas P. Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 738–768. Springer, Heidelberg, May 2020.

[DS19] David Derler and Daniel Slamanig. Key-homomorphic signatures: definitions and applications to multiparty signatures and non-interactive zero-knowledge. *Designs, Codes and Cryptography*, 87:1373–1413, 2019.

[GKM$^+$18] Jens Groth, Markulf Kohlweiss, Mary Maller, Sarah Meiklejohn, and Ian Miers. Updatable and universal common reference strings with applications to zk-SNARKs. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 698–728. Springer, Heidelberg, August 2018.

[GKO$^+$23] Chaya Ganesh, Yashvanth Kondi, Claudio Orlandi, Mahak Pancholi, Akira Takahashi, and Daniel Tschudi. Witness-succinct universally-composable SNARKs. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part II*, volume 14005 of *LNCS*, pages 315–346. Springer, Heidelberg, April 2023.

[GMM$^+$22] Noemi Glaeser, Matteo Maffei, Giulio Malavolta, Pedro Moreno-Sanchez, Erkan Tairi, and Sri Aravinda Krishnan Thyagarajan. Foundations of coin mixing services. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 1259–1273. ACM Press, November 2022.

[Gro04] Jens Groth. Rerandomizable and replayable adaptive chosen ciphertext attack secure cryptosystems. In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 152–170. Springer, Heidelberg, February 2004.

[GSZB23] Noemi Glaeser, István András Seres, Michael Zhu, and Joseph Bonneau. Cicada: A framework for private non-interactive on-chain auctions and voting. Cryptology ePrint Archive, Paper 2023/1473, 2023. https://eprint.iacr.org/2023/1473.

[GWC19] Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. PLONK: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Report 2019/953, 2019. https://eprint.iacr.org/2019/953.

[HAB$^+$17] Ethan Heilman, Leen Alshenibr, Foteini Baldimtsi, Alessandra Scafuro, and Sharon Goldberg. TumbleBit: An untrusted bitcoin-compatible anonymous payment hub. In *NDSS 2017*. The Internet Society, February / March 2017.

[KKM14] Philip Koshy, Diana Koshy, and Patrick McDaniel. An analysis of anonymity in bitcoin using P2P network traffic. In Nicolas Christin and Reihaneh Safavi-Naini, editors, *FC 2014*, volume 8437 of *LNCS*, pages 469–485. Springer, Heidelberg, March 2014.

[KYMM18] George Kappos, Haaroon Yousaf, Mary Maller, and Sarah Meiklejohn. An empirical analysis of anonymity in zcash. In William Enck and Adrienne Porter Felt, editors, *USENIX Security 2018*, pages 463–477. USENIX Association, August 2018.

[KZM$^+$15] Ahmed Kosba, Zhichao Zhao, Andrew Miller, Yi Qian, Hubert Chan, Charalampos Papamanthou, Rafael Pass, abhi shelat, and Elaine Shi. C∅c∅: A framework for building composable zero-knowledge proofs. Cryptology ePrint Archive, Report 2015/1093, 2015. https://eprint.iacr.org/2015/1093.

[MBKM19] Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 2111–2128. ACM Press, November 2019.

[mon] Monero. https://getmonero.org.

[MSH$^+$18] Malte Möser, Kyle Soska, Ethan Heilman, Kevin Lee, Henry Heffan, Shashvat Srivastava, Kyle Hogan, Jason Hennessey, Andrew Miller, Arvind Narayanan, and Nicolas Christin. An empirical analysis of traceability in the monero blockchain. *PoPETs*, 2018(3):143–163, July 2018.

[Nak08] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized business review*, 2008.

[RMK14] Tim Ruffing, Pedro Moreno-Sanchez, and Aniket Kate. CoinShuffle: Practical decentralized coin mixing for bitcoin. In Miroslaw Kutylowski and Jaideep Vaidya, editors, *ESORICS 2014, Part II*, volume 8713 of *LNCS*, pages 345–364. Springer, Heidelberg, September 2014.

[SNBB19] István András Seres, Dániel A. Nagy, Chris Buckland, and Péter Burcsi. MixEth: efficient, trustless coin mixing service for Ethereum. Cryptology ePrint Archive, Report 2019/341, 2019. https://eprint.iacr.org/2019/341.

[SU17] Dominique Schröder and Dominique Unruh. Security of blind signatures revisited. *Journal of Cryptology*, 30(2):470–494, April 2017.

[TLK$^+$18] Muoi Tran, Loi Luu, Min Suk Kang, Iddo Bentov, and Prateek Saxena. Obscuro: A bitcoin mixer using trusted execution environments. In *Proceedings of the 34th Annual Computer Security Applications Conference*, pages 692–701, 2018.

[TMM21] Erkan Tairi, Pedro Moreno-Sanchez, and Matteo Maffei. A^{2}L: Anonymous atomic locks for scalability in payment channel hubs. In *2021 IEEE Symposium on Security and Privacy*, pages 1834–1851. IEEE Computer Society Press, May 2021.

[zca] Zcash. https://z.cash.

[zca16] The design of the ceremony, 10 2016. https://electriccoin.co/blog/the-design-of-the-ceremony/.

[ZGP17] Alexandros Zacharakis, Panagiotis Grontas, and Aris Pagourtzis. Conditional blind signatures. Cryptology ePrint Archive, Report 2017/682, 2017. https://eprint.iacr.org/2017/682.