Cryptography for Private and Secure 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. I discuss how to generically add universally composable security to any non-interactive zero-knowledge proof in a way that is compatible with an updatable common reference string, thus showing how to strengthen security for any system relying on ZKPs while maintaining minimal trust assumptions; improve the security of a class of coin mixing protocols by giving a formal security treatment of this class and patching the security of an existing, insecure protocol; and show how to construct efficient, non-interactive, and private on-chain protocols for a large class of elections and auctions. Finally, I describe two proposed works: a new threshold wallet construction in a strong threat model and a comparison of on-chain key management approaches.

Contents

1 Introduction

Bitcoin [Nak08] was the first digital currency to successfully implement a fully trustless and decentralized payment system. Its underlying innovation was the *blockchain*, a distributed append-only ledger to record transactions. Its consistency is enforced via a "proof-of-work" (PoW) consensus mechanism in which participants solve difficult computational puzzles (hash preimages) to append the newest bundle of transactions (block) to the chain.

Ethereum [But14] introduced programmability via *smart contracts*, special applications which sit on top of the consensus layer and can maintain state and modify it programmatically. This has led to the emergence of a number of *decentralized applications* enabling more diverse functionalities in a trustless manner.

This proposal describes various applications of cryptography to blockchains. In Section 2, I discuss work on privacy-enhancing building blocks of blockchains, namely zero-knowledge proofs. In particular, I summarize my work adding universally composable security to zero-knowledge proofs with updatable common reference strings [AGRS24]. Section 3 moves on to privacy-enchancing blockchain applications, specifically my work analyzing the security of a class of coin mixing services [GMM$^+$22] and designing a framework for non-interactive and private on-chain elections and auctions [GSZB23]. Finally, in Section 4, I discuss my proposed work: a new construction for a threshold cryptocurrency wallet and a systematic analysis of approaches to key distribution on-chain.

1.1 Notation

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

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 on many popular blockchains [BKP14, BT19, KKM14, MSH$^+$18, KYMM18]. Numerous mitigations have been suggested, including adding privacy as an overlay on top of existing blockchains (e.g., via cryptocurrency mixers, described in section 3.1), or by building new privacy-first cryptocurrencies like Zcash [zca] and Monero [mon]. Many of these overlays and new blockchains rely on zero-knowledge proofs.

2.1 Zero-Knowledge Proofs and their trust assumptions

Non-interactive zero-knowledge proofs (NIZKs) are ubiquitous building blocks for achieving both privacy and scalability. Zcash [zca] relies heavily on a type of NIZK called a zk-SNARK (zero-knowledge succinct non-interactive argument of knowledge) to prove that a party has sufficient funds to make a payment without revealing anything more about those funds [BCG^{+}14]. On Ethereum, so-called (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, practitioners use various approaches to minimize 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 universal composability (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.

	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. SE = simulation extractability, BBE = black-box extractability.

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

(Parts of this section are taken/adapted from [AGRS24].)

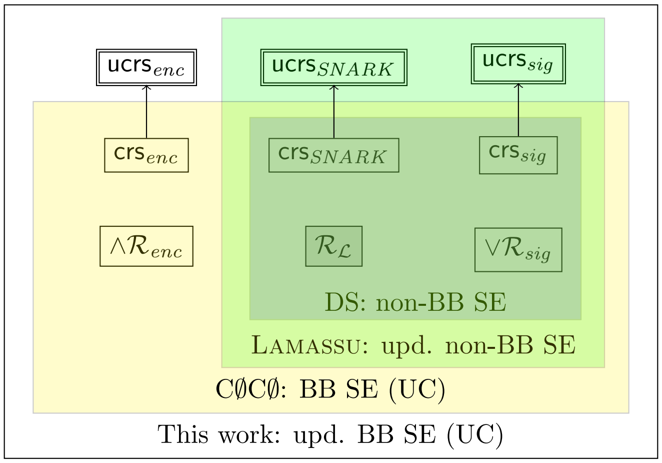

Figure 1: Overview of our approach including previous work from Table 1. ucrs denotes an updatable CRS.

In [AGRS24] we show how to build such a compiler by giving the first *fully black-box approach to generically build* circuit-succinct UC-secure NIZKs with updatable CRS from any zk-SNARK, circumventing the aforementioned problems. BB-Lamassu is a framework for black-box (BB) simulation-extractable (SE) (i.e., universally-composable) circuit-succinct NIZKs with updatable CRS. It can be seen as a hybrid of C∅C∅ [KZM+15] and Lamassu [ARS20], combining the BB extractability of the former with the updatable CRS compatilbility of the latter (see Table 1).

C∅C∅ (yellow box in Figure 1) combines two tricks from the literature to achieve BBE and SE. First, to avoid non-black-box extractors that rely on rewinding or knowledge assumptions, they extend the CRS by a public key and include an encryption of the witness in the proof. This also requires extending the original statement to show that the correct witness was encrypted ($\mathcal{R}_{enc}$) [DP92]. Now the extractor can extract the witness by simply decrypting the ciphertext in the CRS. Second, they use the classical OR trick (i.e., an alternate clause $\mathcal{R}_{sig}$, to be used only by the simulator, which checks for a valid signature) to enable unbounded simulation of proofs [DDO+01], i.e., SE with a straight-line extractor. Together, these additions augment the underlying scheme with SE and UC-security, but the new ciphertext in the CRS (crs_{enc}) adds a linear overhead of $|w|$, so it does not fully preserve succinctness and can only give UC SE *NIZKs*.

Lamassu (green box) revisits the C∅C∅ framework, tailoring it to updatable NIZKs and removing the $|w|$ overhead. This is achieved by using the non-black-box extractor of the underlying scheme instead of an encryption of the witness, thus preserving succinctness (modulo some small constant overhead). To make unbounded proof simulation compatible with an updatable

CRS, Lamassu adapts the simulation technique of [DS19] (blue box), which used the OR trick to combine the underlying SNARK's non-BB extractor with key-homomorphic signatures. To support an updatable CRS, Lamassu swaps the signature for an updatable signature (US). The result is a generic framework for CRS-updatable SE succinct NIZKs, but it sacrifices UC-security due to the non-black-box extractor.

BB-Lamassu. Our new framework BB-Lamassu (outside box) uses Lamassu as a starting point and adds back in an encryption of the witness ($\mathcal{R}_{enc}$) for BB-extractability. To be compatible with updatability, we instantiate this with a novel public-key encryption (PKE) primitive which we call *extractable key-updatable PKE (EKU-PKE)*, for which we show an efficient construction. We still have to overcome the hurdle of providing BB extraction for the US and the public key of the EKU-PKE in the CRS (crs_{enc}). In brief, this is done by using an efficient (but not necessarily succinct) BBE NIZK *without a CRS* to prove updates of the CRS elements, i.e., updates of crs_{SNARK}, the US public key crs_{sig}, and the EKU-PKE public key crs_{enc}.

UC security proof. Since BB-Lamassu is BB SE, it is also UC-secure and should therefore realize the NIZK ideal functionality $\mathcal{F}_{\mathsf{NIZK}}$ of [Gro06]. However, so far this ignores the updatable CRS aspect. To formally confirm this intuition, we introduce a new ideal functionality $\mathcal{F}_{\mathsf{up\text{-}CRS}}$ for the updatable CRS generation and then prove that BB-Lamassu realizes the functionality $\mathcal{F}_{\mathsf{NIZK}}$ in the $\mathcal{F}_{\mathsf{up\text{-}CRS}}$-hybrid model. Our analysis is carried out in the local ROM, which can be realized in practice by domain separation in the hash function. We note that the use of an RO arises from a building block (the proof of CRS update) and not from the construction of our compiler.

Performance evaluation. To demonstrate the applicability of BB-Lamassu, we provide a detailed analysis of the induced overheads. For concrete instantiations, we estimate overheads of 32 bytes for the CRS, 170 bytes for the CRS update, and 256 bytes plus the size of the witness for the proof. This is a reduction in both storage and runtime overheads compared to Lamassu [ARS20]. For witness sizes observed in practical applications such as Zcash, BB-Lamassu adds well below 10,000 additional constraints.

As a concrete example, we apply BB-Lamassu to Sonic [MBKM19], a zk-SNARK with updatable CRS.[1] We then experimentally evaluate the overhead introduced by BB-Lamassu. For a SHA-256 preimage, which is interesting for Merkle-tree membership proofs, the prover and verifier overhead, respectively, is $\approx 1.2\times$ and $1.07\times$. Our evaluation shows that as the circuits become larger and more complex, proving and verifying the original circuit dominates the overall performance costs and the overhead added by BB-Lamassu converges to the size of the witness (see Figure 2).

[1] https://github.com/nglaeser/sonic-ucse/

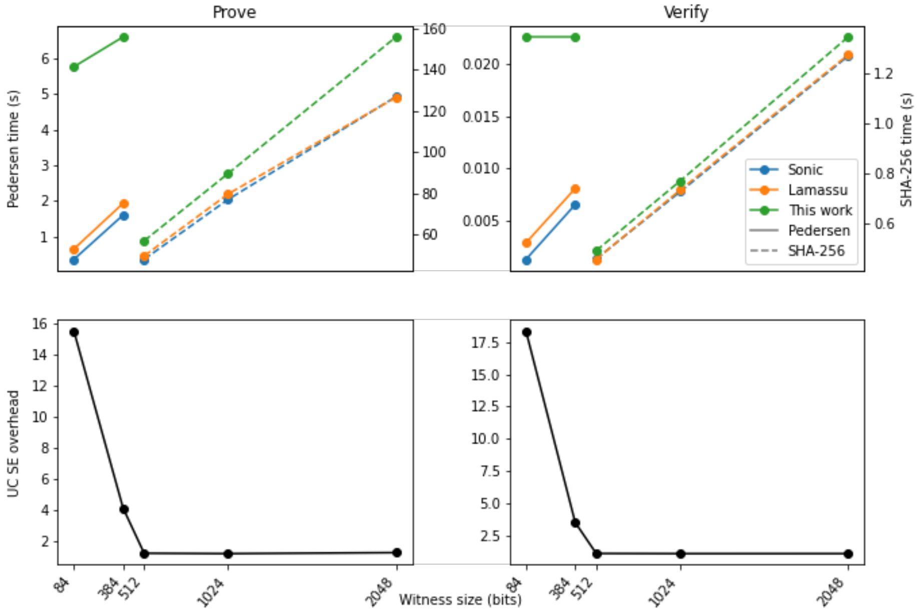

Figure 2: Runtimes of the Prove and Verify algorithms for Pedersen and SHA-256 preimages using our BB SE succinct NIZK compared to the non-BB SE zk-SNARK obtained via Lamassu [ARS20] and the base non-SE zk-SNARK Sonic [MBKM19]. In the lower part we plot the overhead of our transformation to add BB SE, which decreases as the witness size increases.

3 Privacy-Enhancing Applications

Because the blockchain itself does not guarantee the privacy of users, privacy is often added at the application layer. For example, many applications serve as a privacy overlay to the underlying payment layer. Other applications, such as private voting, enable a wholly new functionality while also adding privacy.

3.1 Cryptocurrency Mixers

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

Cryptocurrency mixers [RMK14, SNBB19, TLK+18, BNM+14, coi22, GM17] 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 into 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).

When the underlying blockchain offers scripting functionality, a simple mixer can be set up as an account into which users deposit coins and later retrieve them (or allow another party to retrieve them) by redeeming a token, where atomicty is enforced via an on-chain script. When the goal is scalability and/or interoperability between different blockchains, more complicated protocols are needed to enforce the atomicity requirement without relying on the on-chain scripting functionality.

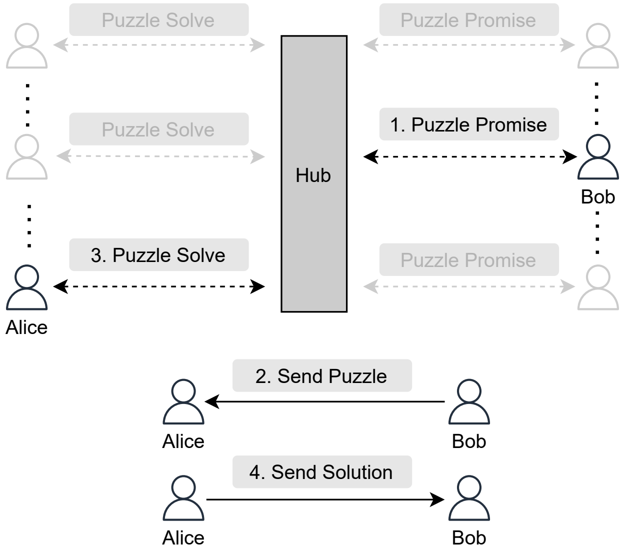

Figure 3: Protocol flow of a synchronization puzzle, the un- derlying cryptographic mechanism of Tumblebit and A^{2}L, which we formalize as Blind Conditional Signatures. Dotted double-edged arrows indicate 2-party protocols. Solid arrows indicate secure point-to-point communication.

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 (Figure 3). Synchronization puzzle protocols consist of four steps: (1) Hub and Bob execute the *puzzle promise* phase with respect to some message m_{HB}, which outputs a puzzle τ containing a hidden signature σ_{HB} 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 σ_{HB} on m_{HB} and Hub learns σ_{AH} on m_{AH}. To conclude, (4) Alice sends σ_{HB} 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 σ_{AH} without revealing σ_{HB}.)
- **Unforgeability**: Bob cannot output a valid signature σ_{HB} on m_{HB} before the puzzle solver phase completes. (This ensures Bob cannot learn σ_{HB} without Hub learning σ_{AH}.)

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 σ_{AH} and σ_{HB} 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).

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 requires only limited capabilities of the underlying blockchain, i.e., no scripting functionality. 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 1 (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* $\mathcal{R}$*:*

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

$\mathsf{PreSig}(\mathsf{sk}, m, Y) \rightarrow \tilde{\sigma}$**:** *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* $\tilde{\sigma}$*.*

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

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

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

$\mathsf{Ext}(\tilde{\sigma}, \sigma, Y) \rightarrow y$**:** *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 $Y \in \mathcal{L}_{\mathcal{R}}$ 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 A^{2}L:

Theorem 1 (Main theorem [TMM21] (paraphrased)[2]). *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.

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

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. 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 $\tilde{x} = x$, 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 $i = 1, \dots, \lambda$. 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 $q \in O(\lambda)$ successful payments, i.e., learn $q+1$ signature witnesses for every q 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

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

  Here e_i is the i-th unit vector and $x[i]$ is the ith bit of x.

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 $q+1$ puzzle promise instances $(c_j := \Pi_{\mathsf{E}}.\mathsf{Enc}(\mathsf{ek}, x_j), h_j := g^{x_j})$ for $j = 1, \dots, q+1$, 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 $q = |x_1|$ 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. Furthermore, the length of each x_i is λ, so the attack is efficient.

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 $\mathrm{A}^2\mathrm{L}$. 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)[3] which captures the core synchronization puzzle functionality. We give game-based security definitions for BCS and show how to modify $\mathrm{A}^2\mathrm{L}$ to obtain a new protocol, $\mathrm{A}^2\mathrm{L}^+$, which meets these definitions. We also give a UC-secure construction of BCS, dubbed $\mathrm{A}^2\mathrm{L}^{\mathrm{UC}}$, 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 $\mathrm{A}^2\mathrm{L}$, 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 2 (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* $(\tilde{\mathsf{ek}}, \tilde{\mathsf{dk}})$*.*

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

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

- $\{\sigma, \perp\} \leftarrow \mathsf{Open}(\tau, s)$*: 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:

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)[4]. Roughly speaking, it says that two promise/solve sessions cannot be linked together by the hub.[5][6]

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.

We refer the reader to [GMM$^+$22] for the security games. Security is defined as the collection of all three properties:

[4] 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.

[5] 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].

[6] 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.

Definition 3 (Security). *A blind conditional signature Π_{BCS} is secure if it is blind, unlockable, and unforgeable.*

The A^{2}L$^+$ protocol: a secure version of A^{2}L. With the BCS formalism in place, we can prove that an appropriately modified version of A^{2}L, which we call A^{2}L$^+$, satisfies the above 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.[7] We defer the details of the A^{2}L$^+$ protocol, its analysis, and a discussion of its overhead with respect to A^{2}L to [GMM$^+$22].

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 [TMM21] is flawed due to this same reason.

Thus, we develop a different protocol, called A^{2}L$^{\mathrm{UC}}$ that we can prove realizes the BCS UC functionality $\mathcal{F}_{\mathsf{BCS}}$ (essentially the same functionality introduced in [TMM21]) in the standard model. We assume the following cryptographic building blocks:

- An adaptor signature scheme Π_{ADP} defined with respect to Π_{ADP} 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 [BDJR97] encryption scheme $\Pi_{\mathsf{E}} := (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ with unique decryption keys, i.e., ek can be computed injectively from dk.

A^{2}L$^{\mathrm{UC}}$ uses the heavy hammer of multi-party computation to reconcile the need for both CCA-security and rerandomizability of the encryption scheme.

[7] 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.

As in $\mathrm{A}^2\mathrm{L}$ and $\mathrm{A}^2\mathrm{L}^+$, the puzzle sent in the puzzle promise phase consists of a (now CCA-secure) encryption c of the witness y and the instance $Y := g^y$ along with a pre-signature $\tilde{\sigma}_{HB}$; but in the puzzle solver phase, Alice can only rerandomize the instance on her own to obtain $Y' := Y \cdot g^r$, which she uses for the pre-signature $\tilde{\sigma}_{AH}$. To rerandomize the witness contained in c, Alice and Hub engage in a two-party computation (2PC) protocol in which Alice inputs r, c and Hub inputs its decryption key. The 2PC decrypts c and adds r to the result, which it outputs to Hub. The rest of the protocol continues as before: Hub uses the rerandomized witness to complete $\tilde{\sigma}_{AH}$ to σ_{AH}, Alice extracts the witness, derandomizes it, and sends it to Bob, and Bob uses it to complete $\tilde{\sigma}_{HB}$ to σ_{HB}. The details of the protocol, the ideal functionality $\mathcal{F}_{\mathsf{BCS}}$, and the UC security proof are left to [GMM+22].

Because the scheme relies on standard general-purpose cryptographic tools such as a UC-secure 2PC, it 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.

3.2 On-chain private voting and auctions

(Parts of this section are taken/adapted from [GSZB23].)

The application layer can provide more varied functionalities than simply adding anonymity to payments. This includes on-chain auctions and voting, which are are becoming increasingly requested web3 applications. Decentralized marketplaces run auctions to sell digital goods like non-fungible tokens (NFTs) [Ope23] or domain names [XWY+21], while decentralized autonomous organizations (DAOs) deploy voting schemes to enact decentralized governance [Opt23]. Most auction or voting schemes currently deployed on blockchains, e.g., NFT auctions on OpenSea or Uniswap governance [FMW22], lack bid/ballot privacy. This can negatively influence user behavior, for example, by vote herding or discouraging participation [EL04, GY19, SY03]. The lack of privacy can cause surges in congestion and transaction fees as users try to outbid each other to participate, a negative externality for the entire network.

Existing private voting protocols [PE23, GG22, PE] achieve privacy at the cost of introducing a trusted authority who is still able to view all submissions. Alternatively, the only private *and* trustless auction in deployment we are aware of [XWY+21] uses a two-round commit-reveal protocol: in the first round, every party commits to their bid, and in the second round they open the commitments and the winner can be determined. Other protocols relying on more heavyweight cryptographic building blocks have been proposed in the literature. We summarize the various approaches for private voting and auctions in Table 2. Unfortunately, all of them suffer from at least one of the following limitations, hindering widespread adoption:

Interactivity. Interactivity is a usability hurdle that often causes friction in

Approach	NI	No TTPs	Efficient	Tally privacy	Ev. ballot privacy
Commit-reveal [FOO93, GY19]	○	●	●	●	○
Zero-knowledge proofs [PE23]	○	○	●	●	●
Fully homomorphic TLPs [MT19]	●	●	○	●	○
FHE [Gen09, CGGI16, dLNS17]	●	○	○	●	●
Multi-party computation [BDJ+06, AOZZ15]	○	◐	●	●	●
TLPs + homomorphic encryption [CJSS21]	●	◐	●	●	○
HTLPs (our approach)	●	●*	●	●	◐

* when using class groups

Table 2: Qualitative comparison of major cryptographic approaches for designing private auction/voting schemes. NI = non-interactive, Ev. = everlasting. [AOZZ15, CJSS21] require a trusted setup but no TTP. Everlasting ballot privacy can be added to our approach via an extension discussed in [GSZB23].

the protocols' execution. Mandatory bid/ballot reveals are also a target for censorship. A malicious party can bribe the block proposers to exclude certain bids or ballots until the auction/voting ends [PRF23].

Trusted third party (TTP). Many schemes use a trusted coordinator to tally submissions during the voting/bidding phase. This introduces a strong assumption which is at odds with the trustless ethos of the blockchain ecosystem.

Inefficiency. Introducing more complicated cryptographic primitives such as fully-homomorphic encryption (FHE) introduces additional overheads which can incur extra gas fees or at least extra time for all participants.

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

We introduce Cicada, a general framework for practical, privacy-preserving, and trust-minimized protocols for both auctions and voting. Cicada uses time-lock puzzles (TLPs) [RSW96] to achieve *privacy and non-interactivity* in a trustless and efficient manner. A TLP allows a party to "encrypt" a message to the future. Specifically, to recover the solution, one needs to perform a computation that is believed to be inherently sequential, with a parameterizable number of steps. Intuitively, the TLPs play the role of commitments to bids/ballots that any party can open after a predefined time, avoiding the reliance on a second *reveal* round.

Since solving a TLP is computationally intensive, ideally, an efficient protocol would require solving only a sublinear number of TLPs (in the number of voters/bidders). Cicada achieves this via *homomorphic* TLPs (HTLPs): bids/ballots encoded as HTLPs can be "squashed" into a sublinear number of TLPs. Fully homomorphic TLPs are not practically efficient, but Malavolta and Thyagarajan [MT19] introduced efficient additively and multiplicatively homomorphic TLP constructions which we will describe below. This is clearly enough for simple constructions like first-past-the-post (FPTP) voting, but it remained an open problem how to apply HTLPs to realize more complicated auction and voting protocols, e.g., cumulative voting.

We show how to use HTLPs to build practically efficient, private, and non-interactive protocols for a special class of auction and voting schemes where the tallying procedure can be expressed as a linear function. We show that this limited class nonetheless includes many schemes of interest:

Majority, approval, range, and cumulative voting. In majority/FPTP voting, users can cast a 1 (support) vote for a single candidate (or cause) and give 0 (oppose) to the others. Approval voting is a slight generalization of binary voting, where users can submit several binary votes for multiple candidates, i.e., the cast ballot s can be seen as $s \in \{0,1\}^m$, where m is the number of causes. In a range voting scheme (or score voting), users can give each candidate some weight between 0 and w. A similar scheme is cumulative voting, where users can distribute w votes (points) among the candidates. In each case, each candidate's points are tallied and the candidate with the highest number is declared the winner.

Ranked-choice voting. In a ranked-choice voting scheme, voters can signal more fine-grained preferences among m candidates by listing them in order of preference. There are multiple approaches to determining the winner, including single transferrable vote (STV) and instant runoff voting (IRV). In this work, we focus on the simpler Borda count version [Eme13], where each voter can cast $m-1$ points to their first-choice candidate, $m-2$ points to their second-choice candidate, etc., and the candidate with the most points is the winner. Our protocols can be adapted to similar counting functions, such as the Dowdall system [FG14], via minor modifications.

Quadratic voting. In quadratic voting [LW18], each user's ballot is a vector $\vec{b} = (b_1, \ldots, b_m)$ such that $\langle \vec{b}, \vec{b} \rangle = \|\vec{b}\|_2^2 = \sum_i b_i^2 \leq w$. Once again, the winner is determined by summing all the ballots and determining the candidate with the most points.

Homomorphic time-lock puzzles. Malavolta and Thyagarajan [MT19] construct two HTLPs with, respectively, linear and multiplicative homomorphisms in groups of unknown order. For our purposes we are only interested in the former, which is based on the Paillier cryptosystem [Pai99]. It uses $N = pq$ a strong semiprime, $g \xleftarrow{\$} \mathbb{Z}_N^*$ and $h = g^{2^T}$, and has solution space $\mathbb{Z}_N$. A puzzle Z is constructed as

$$(g^r, h^{r \cdot N}(1+N)^s) \in \mathbb{J}_N \times \mathbb{Z}_{N^2}^* \qquad (1)$$

where $\mathbb{J}_N$ is the subgroup of $\subseteq \mathbb{Z}_N^*$ of elements with Jacobi symbol $+1$. To recover s, a solver must recompute $h^r = (g^r)^{2^T}$, which is believed to be an inherently sequential computation in a group of unknown order.

As an alternative, we introduce a novel linear HTLP based on the exponential ElGamal cryptosystem [CGS97] over a group of unknown order. This construction is more efficient for a small solution space $\mathcal{S} \subset \mathbb{Z}_N$, i.e., $\mathcal{S} = \{s : s \in \mathbb{J}_N \wedge\ s \ll N\}$. Here a puzzle Z is constructed as

$$(g^r, h^r y^s) \in (\mathbb{Z}_N^*)^2 \tag{2}$$

where $g, y \xleftarrow{\$} \mathbb{Z}_N^*$ and again $h = g^{2^T}$. This scheme is only practical for small $\mathcal{S}$ since, in addition to recomputing h^r, recovering s requires brute-forcing the discrete modulus of y^s. We discuss the efficiency trade-off between these two constructions in [GSZB23].

Introducing a homomorphism raises the issue of puzzle malleability, i.e., the possibility of "mauling" one puzzle (whose solution may be unknown) into a puzzle with a related solution. This could lead to issues when HTLPs are deployed in larger systems, prompting research into non-malleable TLPs [FKPS21]. In our case, we define and enforce non-malleability at the system level (see below).

We will use NIZKs to enforce well-formedness of user submissions while maintaining their secrecy. This prevents users from "poisoning" the aggregate HTLP maintained by the on-chain coordinator. For efficiency, we make use of custom NIZKs in groups of unknown order following the approach of [BBF19] (see [GSZB23]).

Syntax of Time-Locked Voting and Auction Protocols We now introduce a generic syntax for a time-locked voting/auction protocol. Any such protocol is defined with respect to a base *scoring function* $\Sigma : \mathcal{X}^n \to \mathcal{Y}$ (e.g., second-price auction, range voting), which takes as input n submissions (bids/ballots) $s_1, \ldots, s_n$ in the submission domain $\mathcal{X}$ and computes the election/auction result $\Sigma(s_1, \ldots, s_n) \in \mathcal{Y}$. It is useful to break down the scoring function into the "tally" or aggregation function $t : \mathcal{X}^n \to \mathcal{X}'$ and the finalization function $f : \mathcal{X}' \to \mathcal{Y}$, i.e., $\Sigma = f \circ t$. For example, in first-past-the-post voting, the tally function t is addition, and the finalization function f is $\arg\max$ over the final tally/bids.

Definition 4 (Time-locked voting/auction protocol). *A time-locked voting/auction protocol* $\Pi_\Sigma = (\mathsf{Setup}, \mathsf{Seal}, \mathsf{Aggr}, \mathsf{Open}, \mathsf{Finalize})$ *is defined with respect to a base voting/auction protocol* $\Sigma = f \circ t$*, where* $t : \mathcal{X}^n \to \mathcal{X}'$ *and* $f : \mathcal{X}' \to \mathcal{Y}$*.*

$\mathsf{Setup}(1^\lambda, T) \xrightarrow{\$} (\mathsf{pp}, \mathcal{Z})$**.** *Given a security parameter* λ *and a time parameter* T*, output public parameters* pp *and an initial list of HTLP(s)* $\mathcal{Z}$ *that corresponds to the running tally or bid computation.*

$\mathsf{Seal}(\mathsf{pp}, i, s) \xrightarrow{\$} (\mathcal{Z}_i, \pi_i)$**.** *User* $i \in [n]$ *wraps its submission* $s \in \mathcal{X}$ *in a (list of) HTLP(s)* $\mathcal{Z}_i$*. It also outputs a proof of well-formedness* π_i*.*

$\mathsf{Aggr}(\mathsf{pp}, \mathcal{Z}, i, \mathcal{Z}_i, \pi_i) \to \mathcal{Z}'$**.** *Given a list of (tally) HTLPs* $\mathcal{Z}$*, time-locked submission* $\mathcal{Z}_i$ *of user* i*, and proof* π_i*, the transparent contract potentially aggregates the sealed submission homomorphically into* $\mathcal{Z}$ *to get an updated (tally)* $\mathcal{Z}' = t(\mathcal{Z}, \mathcal{Z}_i)$*.*

$\mathsf{Open}(\mathsf{pp}, \mathcal{Z}) \to (\mathcal{S}, \pi_{\mathsf{open}})$. *Open $\mathcal{Z}$ to solution(s) $\mathcal{S}$, requiring T sequential steps, and compute a proof π_{open} to prove correctness of $\mathcal{S}$.*

$\mathsf{Finalize}(\mathsf{pp}, \mathcal{Z}, \mathcal{S}, \pi_{\mathsf{open}}) \to \{y, \perp\}$. *Given proposed solution(s) $\mathcal{S}$ to $\mathcal{Z}$ with proof π_{open}, the coordinator may reject $\mathcal{S}$ or compute the final result $y = f(\mathcal{S}) \in \mathcal{Y}$.*

We note that the $\mathsf{Setup}(\cdot)$ algorithm in our protocols may use private randomness. In particular, our constructions use cryptographic groups (RSA and Paillier groups) that cannot be efficiently instantiated without a trusted setup (an untrusted setup would require gigantic moduli [San99]). This trust can be minimized by generating the group via a distributed trusted setup, e.g., [BF01a, CHI+21, DM10]. Alternatively, the HTLPs in our protocols could be instantiated in class groups [TCLM21], which do not require a trusted setup; however, HTLPs in class groups are less efficient and verifying them on-chain would be prohibitively costly.

A time-locked voting/auction protocol Π_Σ must satisfy the following informal security properties, which we define formally in [GSZB23]:

Correctness. Π_Σ is *correct* if, assuming setup, submission of n puzzles, aggregation of all n submissions, and opening are all performed honestly, the finalization procedure outputs a winner consistent with the base protocol Σ.

Submission privacy. The scheme satisfies *submission privacy* if the adversary cannot distinguish between two submissions, i.e., bids or ballots. Note that this property is only ensured up to time T.

Non-malleability. Notice that submission privacy alone does not suffice for security: even without knowing the contents of other puzzles, an adversary could submit a value that depends on other participants' (sealed) submissions. For example, in an auction, one could be guaranteed to win by homomorphically computing an HTLP containing the sum of all the other participants' bids plus a small value ϵ. Therefore, we also require *non-malleability*, which requires that no participant can take another's submission and replay it or "maul" it into a valid submission under its own name.

A note on anonymity. We consider user anonymity an orthogonal problem. In the applications we have in mind, users can increase their anonymity by using zero-knowledge mixers or other privacy-enhancing overlays, e.g., zero-knowledge sets [Eth19]. Additionally, users can decouple their identities from their ballots by applying a verifiable shuffle [Nef01], although the on-chain verification of a shuffle proof might be prohibitively costly for larger elections.

Efficient vector encoding for HTLPs. In many voting schemes, a ballot consists of a vector indicating the voter's relative preferences or point allocations for all m candidates. To avoid solving many HTLPs, it is desirable to encode this vector into a single HTLP, which requires representing the vector as a single integer.

Definition 5 (Packing scheme)**.** *A setup algorithm* PSetup *and pair of efficiently computable bijective functions* $(\mathsf{Pack}, \mathsf{Unpack})$ *is called a* packing scheme *and has the following syntax:*

- $\mathsf{PSetup}(\ell, w) \to \mathsf{pp}$*. Given a vector dimension* ℓ *and maximum entry* w*, output public parameters* pp*.*
- $\mathsf{Pack}(\mathsf{pp}, \vec{a}) \to s$*. Encode* $\vec{a} \in (\mathbb{Z}^+)^\ell$ *as a positive integer* $s \in \mathbb{Z}^+$*.*
- $\mathsf{Unpack}(\mathsf{pp}, s) \to \vec{a}$*. Given* $s \in \mathbb{Z}^+$*, recover a vector* $\vec{a} \in (\mathbb{Z}^+)^\ell$*.*

For correctness *we require* $\mathsf{Unpack}(\mathsf{Pack}(\vec{a})) = \vec{a}$ *for all* $\vec{a} \in (\mathbb{Z}^+)^\ell$*.*

The classic approach to packing [Gro05, HS00] uses a *positional numeral system (PNS)* to encode a vector of entries bounded by w as a single integer in base $M := w$. Instead, we will set $M := nw + 1$ to accommodate the homomorphic addition of all n users' vectors: each voter submits a length-m vector with entries $\leq w$. Summing over n voters, the result is a length-m vector with a maximum entry value nw; to prevent overflow, we set $M = nw + 1$.

We also introduce an alternative approach in Construction 1 which is based on the *residue numeral system (RNS)*. The idea of the RNS packing is to interpret the entries of $\vec{a}$ as prime residues of a single unique integer s, which can be found efficiently using the Chinese Remainder Theorem (CRT). In other words, for all $j \in [\ell]$, s captures a_j as $s \bmod p_j$.

Construction 1 (Packing from Residue Numeral System)**.**

- $\mathsf{PSetup}(\ell, w) \to \vec{p}$ *: Let* $M := w + 1$ *and sample* ℓ *distinct primes* $p_1, \ldots, p_\ell$ *s.t.* $p_j \geq M \ \forall j \in [\ell]$*. Return* $\vec{p} := (p_1, \ldots, p_\ell)$*.*
- $\mathsf{Pack}(\vec{p}, \vec{a}) \to s$*: Given* $\vec{a} \in (\mathbb{Z}^+)^\ell$*, use the CRT to find the unique* $s \in \mathbb{Z}^+$ *s.t.* $s \equiv a_j \pmod{p_j} \ \forall j \in [\ell]$*.*
- $\mathsf{Unpack}(\vec{p}, s) \to \vec{a}$*: return* $(a_1, \ldots, a_\ell)$ *where* $a_j \equiv s \mod p_j \ \forall j \in [\ell]$*.*

A major advantage of this approach is that, in contrast to the PNS approach, which is only homomorphic for SIMD (single instruction, multiple data) addition, the RNS encoding is fully SIMD homomorphic: the sum of vector encodings $\sum_{i\in[n]} s_i$ encodes the vector $\vec{a}_+ = \sum_{i\in[n]} \vec{a}_i$, and the product $\prod_{i\in[n]} s_i$ encodes the vector $\vec{a}_\times = \prod_{i\in[n]} \vec{a}_i$. Note that as in the PNS approach, we set $M = nw + 1$ to accommodate homomorphic addition of submissions; homomorphic multiplication, however, would require $M = w^n + 1$, and the primes in $\vec{p}$ would therefore be larger as well. Although the RNS has found application in error correction [KPT+22, TC14], side-channel resistance [PFPB18], and parallelization of arithmetic computations [AHK17, BDM06, GTN11, VNL+20], to our knowledge it has not been applied to voting schemes. We show in [GSZB23] that RNS is in fact a natural fit for some voting schemes, in particular quadratic voting, where results in more efficient proofs of ballot correctness.

The Cicada Framework

Let $\Sigma : \mathcal{X}^n \to \mathcal{Y}$ be an linear voting/auction scheme where $\mathcal{X} = [0, w]^m$, HTLP a linear HTLP, $T \in \mathbb{N}$ be a time parameter representing the election/auction length, and a packing scheme $(\mathsf{PSetup}, \mathsf{Pack}, \mathsf{Unpack})$. Let NIZK be a NIZKPoK for submission correctness (the language depends on Σ and HTLP) and PoE a proof of exponentiation [Pie19, Wes19].

$\mathsf{Setup}(1^\lambda, T, \ell) \xrightarrow{\$} (\mathsf{pp}, \mathcal{Z})$**.** Set up the public parameters $\mathsf{pp}_{\mathsf{NIZK}} \xleftarrow{\$} \mathsf{NIZK.Setup}(1^\lambda)$, $\mathsf{pp}_{\mathsf{tlp}} \xleftarrow{\$} \mathsf{HTLP.Setup}(1^\lambda, T)$, and $\mathsf{pp}_{\mathsf{pack}} \leftarrow \mathsf{PSetup}(\ell, w)$. Let $\mathcal{Z} = \{Z_j\}_{j \in [m/\ell]}$ where $Z_j \xleftarrow{\$} \mathsf{HTLP.Gen}(0)$. Output $\mathsf{pp} := (\mathsf{pp}_{\mathsf{tlp}}, \mathsf{pp}_{\mathsf{pack}}, \mathsf{pp}_{\mathsf{NIZK}})$ and $\mathcal{Z}$.

$\mathsf{Seal}(\mathsf{pp}, i, \vec{v}_i) \xrightarrow{\$} (\mathcal{Z}_i, \pi_i)$**.** Parse $\vec{v}_i := \vec{v}_{i,1} || \dots || \vec{v}_{i,m/\ell}$. Compute $Z_{i,j} \leftarrow \mathsf{HTLP.Gen}(\mathsf{Pack}(\vec{v}_{i,j}))\ \forall j \in [m/\ell]$ and $\pi_i \leftarrow \mathsf{NIZK.P}((i, \mathcal{Z}_i), \vec{v}_i)$. Output $(\mathcal{Z}_i := \{Z_{i,j}\}_{j \in [m/\ell]}, \pi_i)$

$\mathsf{Aggr}(\mathsf{pp}, \mathcal{Z}, i, \mathcal{Z}_i, \pi_i) \to \mathcal{Z}'$**.** If $\mathsf{NIZK}.vrfy((i, \mathcal{Z}_i), \pi_i) = 1$, update $\mathcal{Z}$ to $\mathcal{Z} \boxplus \mathcal{Z}_i$.

$\mathsf{Open}(\mathsf{pp}, \mathcal{Z}) \to (\mathcal{S}, \pi_{\mathsf{open}})$**.** Parse $\mathcal{Z} := \{Z_j\}_{j \in [m/\ell]}$ and solve for the encoded tally $\mathcal{S} = \{s_j\}_{j \in [m/\ell]}$ where $s_j \leftarrow \mathsf{HTLP.Solve}(Z_j)$. Prove the correctness of the solution(s) as $\pi_{\mathsf{open}} \leftarrow \mathsf{PoE.Prove}(\mathcal{S}, \mathcal{Z}, 2^T)$ and output $(\mathcal{S}, \pi_{\mathsf{open}})$.

$\mathsf{Finalize}(\mathsf{pp}, \mathcal{Z}, \mathcal{S}, \pi_{\mathsf{open}}) \to \{y, \bot\}$**.** If $\mathsf{PoE.Verify}(\mathcal{S}, \mathcal{Z}, 2^T, \pi_{\mathsf{open}}) \neq 1$, return $\bot$. Otherwise, parse $S := \{s_j\}_{j \in [m/\ell]}$ and let $\vec{v} := \vec{v}_1 || \dots || \vec{v}_{m/\ell}$, where $\vec{v}_j \leftarrow \mathsf{Unpack}(s_j)\ \forall j \in [m/\ell]$. Output y such that $y = \Sigma(\vec{v})$.

Figure 4: The Cicada framework for non-interactive private auctions and elections.

The Cicada framework. We present Cicada, our framework for non-interactive private auctions/elections, in Figure 4. Cicada can be applied to voting and auction schemes where the scoring function Σ has a linear tally function t. This includes the following schemes:

Additive voting. FPTP, approval, range, and cumulative voting are all examples of schemes with a linear tally function: each ballot (a length-m vector) is simply added to the tally, and the finalization function f is applied to the tally after the voting phase has ended to determine the winner. Borda-count ranked-choice voting [Eme13] can also be expressed as a linear scheme, where each candidate is given a descending number of points based on preference and the vectors are again added. All of these schemes differ only in what qualifies as a "proper" ballot, which will be enforced by the NIZK.

Sealed-bid auctions.[8] The Cicada framework can also be used to implement a sealed-bid auction with a number of HTLPs which is independent of the number of participants n. Assuming bids are bounded by M, we use an HTLP with solution space $\mathcal{S}$ such that $|\mathcal{S}| > M^n$. Each user i submits $Z_i \leftarrow \mathsf{HTLP.Gen}(bid_i)$ and π_i, where π_i proves $0 \leq bid_i \leq M$. A packing of the bids is computed at aggregation time, with Aggr updating Z to $Z \boxplus (M^{i-1} \cdot Z_i)$. After the bidding phase, the final "tally" is opened to s^* and the bids are recovered as $\mathsf{Bids} := \{s^* \mod M^{i-1}\}_{i \in [n]}$. Any payment and allocation function can now be computed over the bids; in the simplest case, the winner is $\arg\max_i(\mathsf{Bids})$ and their payment is $\max_i(\mathsf{Bids})$. Notice that the full set of bids is revealed after the auction concludes. This cannot be avoided when using Cicada with linear HTLPs, since $\max_i$ is a nonlinear function, i.e., it cannot be computed it homomorphically.

Cicada is instantiated with a linear HTLP, vector packing scheme, and matching NIZK for membership in $\mathcal{X}$ to ensure the correctness of submissions. We note that Cicada introduces a crucial design choice via the packing parameter $\ell \in [m]$, which defines a storage-computation trade-off that we discuss in [GSZB23].

Theorem 2. *Given a linear scoring function* Σ*, a secure NIZKPoK* NIZK*, a secure* HTLP*, and a packing scheme* $(\mathsf{PSetup}, \mathsf{Pack}, \mathsf{Unpack})$*, the Cicada protocol* Π_Σ *(Figure 4) is a secure time-locked voting/auction protocol.*

Intuitively, submission privacy follows from the security of the HTLP and zero-knowledge of the NIZK used: the submission can't be opened before time T and none of the proofs leak any information about it. Non-malleability is enforced by requiring the NIZK to be a proof of knowledge and including the user's identity i in the instance to prove, e.g., including it in the hash input of the Fiat-Shamir transform. This prevents a malicious actor from replaying a different user's ballot correctness proof. We delegate the full proof to [GSZB23].

Implementation. We provide open-source, freely available implementations tailored to the popular Ethereum Virtual Machine (EVM) with word size 256 bits.[9] Our most efficient protocols work in $\mathbb{Z}_N^*$ for $N \approx 2^{1024}$, groups which are not natively supported by EVM. We implement several gas-efficient libraries to support modular arithmetic in such groups of unknown order. These protocols can be run today on Ethereum Layer 1. Our non-interactive protocols are particularly well-suited to the EVM since, unlike prior works, we do not need to keep bids, ballots, and proofs in persistent storage as they are not required for any subsequent rounds.

[8] Locking up collateral is necessary for every (private) auction scheme. We treat the problem of collateral lock-up as an important but orthogonal problem and refer to [TAF+23] for an extensive discussion.

[9] https://github.com/a16z/cicada

4 Proposed Work

4.1 Threshold cryptocurrency wallets in the hot-cold paradigm

One ongoing work to be finalized and submitted in the coming weeks is a threshold wallet construction. To achieve very strong security guarantees we assume that the wallet client C places trust in n institutional custodians who consist of both a *hot*, i.e., online, component, and a *cold*, i.e., offline, component. Our construction is secure even when all n hot storages and up to a threshold t of cold storages are compromised. We assume the cold storages are resource-constrained and minimize their computation and communication with the hot storages. In particular, for each message signed, the cold storages send a single message-specific "cold partial signature" to the corresponding hot storage. The hot storage also computes a partial signature, and the hot and cold partial signatures are combined into a partial threshold signature. Additionally, the signing key shares can be refreshed in a distributed manner to provide proactive security, i.e., prevent an adversary from incrementally compromising parties' signing key shares. The client can furthermore request "proofs of remembrance" from any of the hot and cold storages to ensure they are continuing to persistently store the secret key material to avoid loss of funds.

In more detail, we give a construction for (threshold) BLS signatures [BLS01]. Let $\mathbb{G}$ be a group of prime order p with generator g, $H : \{0,1\}^* \to \mathbb{G}$ be a cryptographic hash function, and $h : \mathbb{G} \to \mathbb{Z}_p$ be a universal hash function[10]. Given t-out-of-n Shamir shares $\mathsf{sk}_1, \ldots, \mathsf{sk}_n$ of a BLS signing keypair $(\mathsf{sk}, \mathsf{vk})$, each cold storage P_i^{cold} will store an encryption secret key dk_i and the corresponding hot storage P_i^{hot} stores an encrypted signing key share $\tilde{x}_i := \mathsf{sk}_i + h(\mathsf{ek}_i^{\mathsf{sk}})$. To sign a message $m \in \{0,1\}^*$, P_i^{cold} uses its decryption key to compute a cold partial signature $\sigma_i^{\mathsf{cold}} := H(m)^{h(\mathsf{vk}^{\mathsf{dk}_i})}$, which it sends (out of band) to P_i^{hot}. P_i^{hot} uses its encrypted key share to compute a hot partial signature $\sigma_i^{\mathsf{hot}} := H(m)^{\tilde{x}_i}$ and combines it with σ_i^{cold} to obtain a partial BLS signature $\sigma_i := \sigma_i^{\mathsf{hot}} / \sigma_i^{\mathsf{cold}} = H(m)^{\mathsf{sk}_i}$. Given t valid partial signatures, a full BLS signature on m can be computed in the standard manner (i.e., reconstructing the secret key in the exponent).

The homomorphic nature of the encryption scheme allows key share refreshes via addition of Shamir secret shares of zero. We how to use KZG polynomial commitments [KZG10] and their evaluation proofs to show well-formedness of the share updates. The hot storage proofs of remembrance are also achieved via KZG evaluation proofs, but blinded via a folklore technique [CHM+20, §2.5] so as not to publicly reveal the value of the hot storage key shares.

We are still working on distributed protocols for the initial key share generation and distribution as well as a hot storage recovery protocol in which other parties can help recover a lost encrypted key share $\tilde{x}_i$. The protocol will have a UC security proof and a proof of concept implementation, both of which are still in progress.

[10] In actuality, we use a construction for $h : \mathbb{G}^3 \to \mathbb{Z}_p$, but we use this simpler function for ease of exposition.

4.2 SoK: On-chain key management

In [GKMR23] we introduced the first efficient construction of registration-based encryption (RBE). RBE can be seen as a transparent version of identity-based encryption (IBE) [Sha84] in which the trusted third party (TTP) is replaced with an untrusted, transparent *key curator* (KC). This allows encryption directly to the identity (i.e., some unique identifying string such as a username or phone number) of any party registered in the system. The transparent nature of the KC, who simply accumulates the public keys of registering parties into a succinct and publicly available key registry "digest", makes RBE a natural fit to the blockchain. With an efficient construction now available, such a deployment is a realistic possibility.

An important problem in all secure systems is correctly linking identities to public keys. The usual approach in classic networks is some kind of public key infrastructure (PKI), but the trustless and decentralized nature of blockchains offers alternative trust-minimized approaches. I propose a detailed analysis of the tradeoffs between various approaches to key distribution that can be deployed on-chain in the form of a systematization of knowledge (SoK). These include a simple public-key registry (PKR) of identity-public key pairs (the existing Ethereum Name Service (ENS) [ens] is an example), (threshold) IBE, and RBE. For the case of sending encrypted messages, Table 3 gives an initial comparison of a simple public-key registry (where a list of identity-public key pairs are stored on-chain), the Boneh-Franklin IBE [BF01b] with a secret-shared master secret key, and our efficient RBE construction [GKMR23].

	Public-key registry	Threshold IBE	RBE
Succinct on-chain storage	○	●	◐
Non-interactive encryption	◐	●	◐
Non-interactive decryption	●	●	◐
Sender-anonymous	◐	●	●
Recipient-anonymous	●	○	◐
No TTP	●	◐	●
Arbitrary IDs	●	●	●*

Table 3: Properties of an on-chain public key registry, threshold IBE [BF01b], and RBE [GKMR23]. "Succint" storage means constant (●) or sublinear (◐) in the number of parties, non-interactive means only once (●) or infrequently (◐) in the lifetime of the system. Since the PKR keeps the full list of keys on-chain, it does not have succinct storage; furthermore, the sender occasionally has to retrieve the public key for a new recipient, so encryption and sender-anonymity are not fully achieved. Meanwhile threshold IBE ciphertexts do not normally hide the recipient [BLSV18] and rely on a set of TTPs. Finally, recipient-anonymity can be added to the RBE construction (◐) as an extension; the asterisk indicates that arbitrary identity strings for RBE are enabled by a follow-up work [FKP23].

A more in-depth evaluation would include various proposed alternatives to storing the full PKR on-chain while maintaining transparency [MBB+15, CDGM19, Bon16, TD17, MKKS+23]. Furthermore, our basic RBE construction of did not consider key updates or revocation, which are important in practice, but recent work [FKP23] has made progress in that direction and should be incorporated into the analysis. I am also working on a smart contract implementation of the RBE construction to get a more accurate comparison of the on-chain costs of the aforementioned approaches.

5 Timeline

January 2024: Preliminary exam (Jan. 22); submit Cicada (Section 3.2) and threshold hot-cold wallet (Section 4.1) to CCS (Jan. 28).

March 2024: Publish a16z blog post about on-chain RBE along with smart contract implementation.

Summer/Fall 2024: Submit SoK (Section 4.2) to IEEE S&P'25 or FC'25.

December 2024: Dissertation defense.

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.

[AHK17] Shahzad Asif, Md Selim Hossain, and Yinan Kong. High-throughput multi-key elliptic curve cryptosystem based on residue number system. *IET Computers & Digital Techniques*, 11(5):165–172, 2017.

[AOZZ15] Joël Alwen, Rafail Ostrovsky, Hong-Sheng Zhou, and Vassilis Zikas. Incoercible multi-party computation and universally composable receipt-free voting. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 763–780. Springer, Heidelberg, August 2015.

[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.

[BBF19] Dan Boneh, Benedikt Bünz, and Ben Fisch. Batching techniques for accumulators with applications to IOPs and stateless blockchains. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 561–586. Springer, Heidelberg, August 2019.

[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.

[BDJ$^+$06] Peter Bogetoft, Ivan Damgård, Thomas Jakobsen, Kurt Nielsen, Jakob Pagter, and Tomas Toft. A practical implementation of secure auctions based on multiparty integer computation. In Giovanni Di Crescenzo and Avi Rubin, editors, *FC 2006*, volume 4107 of *LNCS*, pages 142–147. Springer, Heidelberg, February / March 2006.

[BDJR97] Mihir Bellare, Anand Desai, Eric Jokipii, and Phillip Rogaway. A concrete security treatment of symmetric encryption. In *38th FOCS*, pages 394–403. IEEE Computer Society Press, October 1997.

[BDM06] Jean-Claude Bajard, Sylvain Duquesne, and Nicolas Méloni. *Combining Montgomery Ladder for Elliptic Curves Defined over $\mathbb{F}_p$ and RNS Representation*. PhD thesis, LIR, 2006.

[BF01a] Dan Boneh and Matthew Franklin. Efficient generation of shared RSA keys. *Journal of the ACM (JACM)*, 48(4):702–722, 2001.

[BF01b] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer, Heidelberg, August 2001.

[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.

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532. Springer, Heidelberg, December 2001.

[BLSV18] Zvika Brakerski, Alex Lombardi, Gil Segev, and Vinod Vaikuntanathan. Anonymous IBE, leakage resilience and circular security from new assumptions. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 535–564. Springer, Heidelberg, April / May 2018.

[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.

[Bon16] Joseph Bonneau. EthIKS: Using Ethereum to audit a CONIKS key transparency log. In Jeremy Clark, Sarah Meiklejohn, Peter Y. A. Ryan, Dan S. Wallach, Michael Brenner, and Kurt Rohloff, editors, *FC 2016 Workshops*, volume 9604 of *LNCS*, pages 95–105. Springer, Heidelberg, February 2016.

[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.

[CDGM19] Melissa Chase, Apoorvaa Deshpande, Esha Ghosh, and Harjasleen Malvai. SEEMless: Secure end-to-end encrypted messaging with less trust. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 1639–1656. ACM Press, November 2019.

[CGGI16] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. A homomorphic LWE based E-voting scheme. In Tsuyoshi Takagi, editor, *Post-Quantum Cryptography - 7th International Workshop, PQCrypto 2016*, pages 245–265. Springer, Heidelberg, 2016.

[CGS97] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A secure and optimally efficient multi-authority election scheme. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 103–118. Springer, Heidelberg, May 1997.

[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.

[CHI$^+$21] Megan Chen, Carmit Hazay, Yuval Ishai, Yuriy Kashnikov, Daniele Micciancio, Tarik Riviere, abhi shelat, Muthu Venkitasubramaniam, and Ruihan Wang. Diogenes: Lightweight scalable RSA modulus generation with a dishonest majority. In *2021 IEEE Symposium on Security and Privacy*, pages 590–607. IEEE Computer Society Press, May 2021.

[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.

[CJSS21] Peter Chvojka, Tibor Jager, Daniel Slamanig, and Christoph Striecks. Versatile and sustainable timed-release encryption and sequential time-lock puzzles (extended abstract). In Elisa Bertino, Haya Shulman, and Michael Waidner, editors, *ESORICS 2021, Part II*, volume 12973 of *LNCS*, pages 64–85. Springer, Heidelberg, October 2021.

[coi22] CoinJoin - Bitcoin Wiki, Accessed on May 2022. https://en.bitcoin.it/wiki/CoinJoin.

[DDO+01] Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. Robust non-interactive zero knowledge. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 566–598. Springer, Heidelberg, August 2001.

[dLNS17] Rafaël del Pino, Vadim Lyubashevsky, Gregory Neven, and Gregor Seiler. Practical quantum-safe voting from lattices. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 1565–1581. ACM Press, October / November 2017.

[DM10] Ivan Damgård and Gert Læssøe Mikkelsen. Efficient, robust and constant-round distributed RSA key generation. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 183–200. Springer, Heidelberg, February 2010.

[DP92] Alfredo De Santis and Giuseppe Persiano. Zero-knowledge proofs of knowledge without interaction (extended abstract). In *33rd FOCS*, pages 427–436. IEEE Computer Society Press, October 1992.

[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.

[EL04] Edith Elkind and Helger Lipmaa. Interleaving cryptography and mechanism design: The case of online auctions. In Ari Juels, editor, *FC 2004*, volume 3110 of *LNCS*, pages 117–131. Springer, Heidelberg, February 2004.

[Eme13] Peter Emerson. The original Borda count and partial voting. *Social Choice and Welfare*, 40:353–358, 2013.

[ens] Ethereum name service. https://ens.domains/.

[Eth19] Ethereum Foundation. Semaphore: a zero-knowledge set implementation for ethereum., May 2019.

[FG14] Jon Fraenkel and Bernard Grofman. The Borda Count and its real-world alternatives: Comparing scoring rules in Nauru and Slovenia. *Australian Journal of Political Science*, 49(2), 2014.

[FKP23] Dario Fiore, Dimitris Kolonelos, and Paola de Perthuis. Cuckoo commitments: Registration-based encryption and key-value map commitments for large spaces. In Jian Guo and Ron Steinfeld, editors, *Advances in Cryptology – ASIACRYPT 2023*, pages 166–200, Singapore, 2023. Springer Nature Singapore.

[FKPS21] Cody Freitag, Ilan Komargodski, Rafael Pass, and Naomi Sirkin. Non-malleable time-lock puzzles and applications. In Kobbi Nissim and Brent Waters, editors, *TCC 2021, Part III*, volume 13044 of *LNCS*, pages 447–479. Springer, Heidelberg, November 2021.

[FMW22] Robin Fritsch, Marino Müller, and Roger Wattenhofer. Analyzing voting power in decentralized governance: Who controls DAOs? *arXiv preprint arXiv:2204.01176*, 2022.

[FOO93] Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. A practical secret voting scheme for large scale elections. In *Advances in Cryptology—AUSCRYPT'92: Workshop on the Theory and Application of Cryptographic Techniques Gold Coast, Queensland, Australia, December 13–16, 1992 Proceedings 3*, pages 244–251. Springer, 1993.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 169–178. ACM Press, May / June 2009.

[GG22] Aayush Gupta and Kobi Gurkan. Plume: An ecdsa nullifier scheme for unique pseudonymity within zero knowledge proofs. Cryptology ePrint Archive, Paper 2022/1255, 2022. https://eprint.iacr.org/2022/1255.

[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.

[GKMR23] Noemi Glaeser, Dimitris Kolonelos, Giulio Malavolta, and Ahmadreza Rahimi. Efficient registration-based encryption. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pages 1065–1079, 2023.

[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.

[GM17] Matthew Green and Ian Miers. Bolt: Anonymous payment channels for decentralized currencies. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 473–489. ACM Press, October / November 2017.

[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.

[Gro05] Jens Groth. Non-interactive zero-knowledge arguments for voting. In John Ioannidis, Angelos Keromytis, and Moti Yung, editors, *ACNS 05*, volume 3531 of *LNCS*, pages 467–482. Springer, Heidelberg, June 2005.

[Gro06] Jens Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In Xuejia Lai and Kefei Chen, editors, *ASIACRYPT 2006*, volume 4284 of *LNCS*, pages 444–459. Springer, Heidelberg, December 2006.

[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.

[GTN11] Mahadevan Gomathisankaran, Akhilesh Tyagi, and Kamesh Namuduri. Horns: A homomorphic encryption scheme for cloud computing using residue number system. In *2011 45th Annual Conference on Information Sciences and Systems*, pages 1–5, 2011.

[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.

[GY19] Hisham S Galal and Amr M Youssef. Verifiable sealed-bid auction on the ethereum blockchain. In *2nd Workshop on Trusted Smart Contracts*, pages 265–278. Springer, 2019.

[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.

[HS00] Martin Hirt and Kazue Sako. Efficient receipt-free voting based on homomorphic encryption. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 539–556. Springer, Heidelberg, May 2000.

[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.

[KPT$^+$22] Igor Anatolyevich Kalmykov, Vladimir Petrovich Pashintsev, Kamil Talyatovich Tyncherov, Aleksandr Anatolyevich Olenev, and Nikita Konstantinovich Chistousov. Error-correction coding using polynomial residue number system. *Applied Sciences*, 12(7):3365, 2022.

[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.

[KZG10] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 177–194. Springer, Heidelberg, December 2010.

[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.

[LW18] Steven P Lalley and E Glen Weyl. Quadratic voting: How mechanism design can radicalize democracy. In *AEA Papers and Proceedings*, volume 108, 2018.

[MBB$^+$15] Marcela S. Melara, Aaron Blankstein, Joseph Bonneau, Edward W. Felten, and Michael J. Freedman. CONIKS: Bringing key transparency to end users. In Jaeyeon Jung and Thorsten Holz, editors, *USENIX Security 2015*, pages 383–398. USENIX Association, August 2015.

[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.

[MKKS$^+$23] Harjasleen Malvai, Lefteris Kokoris-Kogias, Alberto Sonnino, Esha Ghosh, Ercan Oztürk, Kevin Lewi, and Sean Lawlor. Parakeet: Practical key transparency for end-to-end encrypted messaging. Cryptology ePrint Archive, Report 2023/081, 2023. https://eprint.iacr.org/2023/081.

[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.

[MT19] Giulio Malavolta and Sri Aravinda Krishnan Thyagarajan. Homomorphic time-lock puzzles and applications. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 620–649. Springer, Heidelberg, August 2019.

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

[Nef01] C. Andrew Neff. A verifiable secret shuffle and its application to e-voting. In Michael K. Reiter and Pierangela Samarati, editors, *ACM CCS 2001*, pages 116–125. ACM Press, November 2001.

[Ope23] OpeanSea. How to sell an NFT, June 2023.

[Opt23] Optimism. What is the Optimism Collective?, February 2023.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 223–238. Springer, Heidelberg, May 1999.

[PE] Privacy and Scaling Explorations. Rate-limiting nullifier. https://rate-limiting-nullifier.github.io/rln-docs/.

[PE23] Privacy and Scaling Explorations. Maci: Minimal anti-collusion infrastructure. https://maci.pse.dev/, 2023.

[PFPB18] Louiza Papachristodoulou, Apostolos P. Fournaris, Kostas Papagiannopoulos, and Lejla Batina. Practical evaluation of protected residue number system scalar multiplication. *IACR TCHES*, 2019(1):259–282, 2018. https://tches.iacr.org/index.php/TCHES/article/view/7341.

[Pie19] Krzysztof Pietrzak. Simple verifiable delay functions. In Avrim Blum, editor, *ITCS 2019*, volume 124, pages 60:1–60:15. LIPIcs, January 2019.

[PRF23] Mallesh Pai, Max Resnick, and Elijah Fox. Censorship resistance in onchain auctions. *arXiv preprint arXiv:2301.13321*, 2023.

[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.

[RSW96] Ronald L Rivest, Adi Shamir, and David A Wagner. Time-lock puzzles and timed-release crypto. 1996.

[San99] Tomas Sander. Efficient accumulators without trapdoor extended abstracts. In Vijay Varadharajan and Yi Mu, editors, *ICICS 99*, volume 1726 of *LNCS*, pages 252–262. Springer, Heidelberg, November 1999.

[Sha84] Adi Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and David Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 47–53. Springer, Heidelberg, August 1984.

[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.

[SY03] Koutarou Suzuki and Makoto Yokoo. Secure generalized Vickrey auction using homomorphic encryption. In Rebecca Wright, editor, *FC 2003*, volume 2742 of *LNCS*, pages 239–249. Springer, Heidelberg, January 2003.

[TAF$^+$23] Nirvan Tyagi, Arasu Arun, Cody Freitag, Riad Wahby, Joseph Bonneau, and David Mazières. Riggs: Decentralized sealed-bid auctions. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pages 1227–1241, 2023.

[TC14] Thian Fatt Tay and Chip-Hong Chang. A new algorithm for single residue digit error correction in redundant residue number system. In *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1748–1751, 2014.

[TCLM21] Sri Aravinda Krishnan Thyagarajan, Guilhem Castagnos, Fabien Laguillaumie, and Giulio Malavolta. Efficient CCA timed commitments in class groups. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021*, pages 2663–2684. ACM Press, November 2021.

[TD17] Alin Tomescu and Srinivas Devadas. Catena: Efficient non-equivocation via bitcoin. In *2017 IEEE Symposium on Security and Privacy*, pages 393–409. IEEE Computer Society Press, May 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.

[VNL$^+$20] M.V. Valueva, N.N. Nagornov, P.A. Lyakhov, G.V. Valuev, and N.I. Chervyakov. Application of the residue number system to reduce hardware costs of the convolutional neural network implementation. *Mathematics and Computers in Simulation*, 177:232–243, 2020.

[Wes19] Benjamin Wesolowski. Efficient verifiable delay functions. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 379–407. Springer, Heidelberg, May 2019.

[XWY$^+$21] Pengcheng Xia, Haoyu Wang, Zhou Yu, Xinyu Liu, Xiapu Luo, and Guoai Xu. Ethereum name service: the good, the bad, and the ugly. *arXiv preprint arXiv:2104.05185*, 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.