Encryption to the Future

A Paradigm for Sending Secret Messages to Future (Anonymous) Committees

Matteo Campanelli[1⋆], Bernardo David[3], Hamidreza Khoshakhlagh[2], Anders Konring[3], and Jesper Buus Nielsen[2]

[1] Protocol Labs matteo@protocol.ai
[2] Aarhus University, Denmark {hamidreza,jbn}@cs.au.dk
[3] IT University of Copenhagen, Denmark {beda,konr}@itu.dk

Abstract. A number of recent works have constructed cryptographic protocols with flavors of adaptive security by having a randomly-chosen anonymous committee run at each round. Since most of these protocols are stateful, transferring secret states from past committees to future, but still unknown, committees is a crucial challenge. Previous works have tackled this problem with approaches tailor-made for their specific setting, which mostly rely on using a blockchain to orchestrate auxiliary committees that aid in the state hand-over process. In this work, we look at this challenge as an important problem on its own and initiate the study of Encryption to the Future (EtF) as a cryptographic primitive. First, we define a notion of an EtF scheme where time is determined with respect to an underlying blockchain and a lottery selects parties to receive a secret message at some point in the future. While this notion seems overly restrictive, we establish two important facts: 1. if used to encrypt towards parties selected in the "far future", EtF implies witness encryption for NP over a blockchain; 2. if used to encrypt only towards parties selected in the "near future", EtF is not only sufficient for transferring state among committees as required by previous works, but also captures previous tailor-made solutions. To corroborate these results, we provide a novel construction of EtF based on witness encryption over commitments (cWE), which we instantiate from a number of standard assumptions via a construction based on generic cryptographic primitives. Finally, we show how to use "near future" EtF to obtain "far future" EtF with a protocol based on an auxiliary committee whose communication complexity is *independent* of the length of plaintext messages being sent to the future.

1 Introduction

Most cryptographic protocols assume that parties' identities are publicly known. This is a natural requirement, since standard secure channels are identified by a sender and a receiver. However, this status quo also makes it easy for adaptive (or

⋆ Work done in part while the author was affiliated to Aarhus University.

proactive) adversaries to readily identify which parties are executing a protocol and decide on an optimal corruption strategy. In more practical terms, a party with a known identity (*e.g.* IP address) is at risk of being attacked.

A recent line of work [3,15,16] has investigated means for avoiding adaptive (or proactive) corruptions by having different randomly chosen committees of anonymous parties execute each round of a protocol. The rationale is that parties whose identities are unknown cannot be purposefully corrupted. Hence, having each round of a protocol executed by a fresh anonymous committee makes the protocol resilient to such powerful adversaries. However, this raises a new issue:

How can past committees efficiently transfer secret states to future yet-to-be-assigned anonymous committees?

1.1 Motivation: Role Assignment

The task of sending secret messages to a committee member that will be elected in the future can be abstracted as *role assignment*, a notion first introduced in [3] and further developed in [15]. This task consists of sending a message to an abstract role R at a given point in the future. A role is just a bit-string describing an abstract role, such as R ="party number j in round sl of the protocol Γ". Behind the scenes, there is a mechanism that samples the identity of a random party P_i and associates this machine to the role R. Such a mechanism allows anyone to send a message m to R and have m arrive at P_i chosen at some point in the future to act as R. A crucial point is: no one should know the identity of P_i even though P_i learns that it is chosen to act as R.

The approaches proposed in [3,15,16] for realizing role assignment all use an underlying Proof-of-Stake (PoS) blockchain (*e.g.* [9]). On a blockchain, a concrete way to implement role assignment is to sample a fresh key pair $(\mathsf{sk}_\mathsf{R}, \mathsf{pk}_\mathsf{R})$ for a public key encryption scheme, post $(\mathsf{R}, \mathsf{pk}_\mathsf{R})$ on the blockchain and somehow send sk_R to a random P_i without leaking the identity of this party to anyone. Once $(\mathsf{R}, \mathsf{pk}_\mathsf{R})$ is known, every party has a target-anonymous channel to P_i and is able to encrypt under pk_R and post the ciphertext on the blockchain. Notice that using time-lock puzzles (or similar notions) is not sufficient for achieving this notion, since only the party or parties elected for a role should receive a secret message encrypted for that role, while time-lock puzzles allow any party to recover the message if they invest enough computing time.

A shortcoming of the approaches of [3,15,16] is that, besides an underlying blockchain, they require an auxiliary committee to aid in generating $(\mathsf{sk}_\mathsf{R}, \mathsf{pk}_\mathsf{R})$ and selecting P_i. In the case of [3], the auxiliary committee performs cheap operations but can adversarially influence the probability distribution with which P_i is chosen. In the case of [15,16], the auxiliary committee cannot bias this probability distribution but must perform very expensive operations (using Mix-Nets or FHE; see also Section 1.3). Moreover, these approaches have another caveat: they can only be used to select P_i to act as R according to a probability distribution *already known* at the time the auxiliary committee outputs $(\mathsf{R}, \mathsf{pk}_\mathsf{R})$. Hence, they only allow sending messages to future committees that have been *recently*

elected. Later we explicitly consider this weaker setting—where we want to communicate with a "near-future" committee (i.e., whose distribution is known)—and dub it "Encryption to the Current Winner[4]" (ECW).

In this paper we further investigate solutions to the role-assignment problem[5]. Taking a step back from specific solutions to this problem, we strive to obtain non-interactive solutions to *encrypting* to a future role with IND-CPA security *without* the aid of an auxiliary committee. We improve on solutions relying on interaction with an auxiliary committee and shed light on the hardness of achieving a fully non-interactive solution. We also discuss how to extend our approach to IND-CCA2 security and how to allow winners of a role to authenticate themselves when sending a message, achieving both goals using standard assumptions.

1.2 Our Contributions

We look at the issue of sending messages to future roles as a problem on its own and introduce the Encryption to the Future (EtF) primitive as a central tool to solve it. Apart from defining this primitive and showing constructions based on previous works, we propose constructions based on new insights and investigate limits of EtF in different scenarios. Our general constructions for EtF work by lifting a weaker primitive, namely encryption for the aforementioned "near-future" setting, or ECW. Before providing further details, we summarize our contributions as follows:

– A definition for the notion of Encryption to the Future (EtF) in terms of an underlying blockchain and an associated lottery scheme that selects parties in the future to receive messages for a role. We study the strength of EtF as a primitive and prove that a non-interactive EtF scheme allowing for encryption towards parties selected at arbitrary points in the future implies a flavor of witness encryption for NP over a blockchain (referred to as BWE).

[4] The word "winner" here refers to the party who is selected to perform a role according to the underlying lottery of the PoS blockchain (see remainder of introduction).

[5] The family of protocols we consider actually has two role-related aspects to solve. The first—and the focus of this paper—is the aforementioned *role assignment (RA) which deals with the sending of messages to parties selected to perform future roles of a protocol while hiding the identities of such parties*. The other aspect is *role execution* (RX) which focuses on the execution of the specific protocol that runs on top of the RA mechanism, i.e., what messages are sent to which roles and what specification the protocol implements. In [15] the so-called *You Only Speak Once* (YOSO) model is introduced for studying RX. In the YOSO model the protocol execution is between abstract roles which can each speak only once. Later these can then be mapped to physical machines using an RA mechanism. The work of [15] shows that given RA in a synchronous model, any well-formed ideal functionality can be implemented in the YOSO model with security against malicious adaptive corruption of a minority of machines. Concretely, [15] gives an ideal functionality for RA and shows that a YOSO protocol for abstract roles can be compiled into the RA-hybrid model to give a protocol secure against adaptive attacks.

– A novel construction of Encryption to the Current Winner (ECW), *i.e.* EtF where the receiver of a message is determined by the *current* state of the blockchain, which can be instantiated *without auxiliary committees* from standard assumptions via a construction based on generic primitives.
– A transformation from ECW to EtF through an auxiliary committee holding a *small* state, i.e., with communication complexity *independent* of plaintext size $|m|$ (in contrast to [3,15,16] where a committee's state grows with $|m|$).
– An application of ECW as a central primitive for realizing role assignment in protocols that require it (*e.g.* [3,15,16]).

Our EtF notion arguably provides a useful abstraction for the problem of transferring secret states to secret committees. Our ECW construction is the first primitive to realize role assignment without the need for an auxiliary committee. Moreover, building on new insights from our EtF notion and constructions, we show the first protocol for obtaining role assignment with no constraints on when parties are chosen to act as the role. While our protocol uses auxiliary committees, it improves on previous work by only requiring a communication complexity *independent* of the plaintext length. We elaborate on our results, discussing the intuition behind the notion of EtF, its constructions and its fundamental limits. We also invite the reader to use Fig. 1 as reference for the discussion below.

Encryption to the Future (EtF)—Section 3. As in previous works [3,15,16], an EtF scheme is defined with respect to an underlying PoS blockchain. We naturally use core features of the PoS setting to define what "future" means. The vast majority of PoS blockchains (*e.g.* [9]) associates a *slot number* to each block and uses a lottery for selecting parties to generate blocks according to a stake distribution (*i.e.* the probability a party is selected is proportional to the stake the party controls). Thus, in EtF, we let a message be encrypted towards a party that is selected by the underlying blockchain's lottery scheme at a given future slot. We can generalize this and let the lottery select parties for multiple roles associated to each slot (so that committees consisting of multiple parties can be elected at a single point in time). We note that the goal of defining EtF with respect to an underlying blockchain is to construct it without having to assume very strong primitives such as (extractable) witness encryption for NP[6]. Moreover, it is necessary to provide a non-interactive EtF scheme with a means to publicly verify whether a given party has won the lottery to perform a certain role. Since this lottery predicate's output must hold for all parties, we need a consensus mechanism that allows for all parties to agree on lottery parameters/outputs while allowing for third parties to verify this result. An important point of our EtF definition is that it does not impose any constraints on the underlying blockchain's lottery scheme (*e.g.* it is not required to be anonymous) or on the slot when a party is supposed to be chosen to receive a message sent to a given role (*i.e.* party selection for a given role may happen w.r.t. a *future* stake distribution).

[6] While one *might* define EtF in more general settings, namely without a blockchain, it is unclear how to obtain *interesting* instantiations, that is from standard primitives.

Relation to "Blockchain Witness Encryption" (BWE)—Section 8. In order to study how hard it is to realize EtF, we show that EtF implies a version of witness encryption [14] over a blockchain (similar to that of [18] but without relying on committees). The crux of the proof: if we can encrypt a message towards a role assigned to a party only at an arbitrary point in the future, then we can easily construct a witness encryption scheme exploiting EtF and a smart contract on the EtF's underlying blockchain. We also prove the opposite direction (BWE implies EtF), showing that the notions are similar from a feasibility standpoint. This shows another crucial point: to implement non-interactive EtF, we would plausibly need strong assumptions (e.g., full-blown WE). This follows by observing that existing constructions of WE over blockchains (e.g., [18]) are interactive in the sense that they rely on a committee that holds all encrypted messages in secret shared form and periodically re-share them. On the other hand, in the interactive setting, we show a construction of EtF with improved communication complexity that is *independent* from the size (or amount) of EtF encrypted messages: the committee only needs to hold an IBE master secret key (secret shared) and compute secret keys for specific identities. We note that the goal of constructing BWE from EtF is not to provide a concrete instantiation based on existing blockchains but rather to provide evidence that EtF is hard to construct from standard assumptions. The underlying blockchain protocol and lottery we use are standard Proof-of-Stake based blockchains with a VRF-based lottery and smart contracts. The only non-realistic assumption we make is that the stake is distributed in arbitrarily (i.e. it is all locked inside one smart contract) which is an assumption on how the blockchain is operated rather than on how it is constructed or why it is secure.

Encryption to the Current Winner (ECW)—Section 3. By the previous result we know that, unless we turn to strong assumptions, we may not construct a fully non-interactive EtF (i.e., without auxiliary committees); therefore, we look for efficient ways to construct EtF under standard assumptions while minimizing interaction. As a first step towards such a construction, we define the notion of Encryption to a Current Winner (ECW), which is a restricted version of EtF where messages can only be encrypted towards parties selected for a role whose lottery parameters are available for the *current* slot, the one in which we encrypt (this is as in previous constructions [3,15,16]). Unrestricted EtF, on the other hand, allows for encrypting a message toward lottery winners that will be determined at any arbitrary point in the future, including parties who only join the protocol execution far in the future (after the ciphertext has been generated).

Constructing ECW (non-interactively)—Section 5. We show that it is possible to construct a fully non-interactive ECW scheme from standard assumptions. Our construction relies on a milder flavor of witness encryption, which we call Witness Encryption over Commitments (cWE) and define it in Section 4. This primitive is significantly more restricted than full-fledged WE (see also discussion in Remark 2), but still powerful enough: we show in Section 5.1 that ECW can be constructed in a black-box manner from cWE, which in turn can be constructed

from oblivious transfer and garbled circuits [7]. This construction improves over the previous results [3,15,16] since it does not rely on auxiliary committees.

Instantiating YOSO MPC using ECW—Section 6. The notion of ECW is more restricted than EtF, but it can still be useful in applications. We show how to use it as a building block for the YOSO MPC protocol of [15]. Here, each of the rounds in an MPC protocol is executed by a different committee. This same committee will simultaneously transfer its secret state to the next (near-future) committee, which in turn remains anonymous until it transfers its own secret state to the next committee, and so on. This setting clearly matches what ECW offers as a primitive, but it also introduces a few more requirements: 1. ECW ciphertexts must be non-malleable, *i.e.* we need an IND-CCA secure ECW scheme; 2. Only one party is selected for each role; 3. A party is selected for a role at random with probability proportional to its relative stake on the underlying PoS blockchain; 4. Parties selected for roles remain anonymous until they choose to reveal themselves; 5. A party selected for a role must be able to authenticate messages on behalf of the role, *i.e.* publicly proving that it was selected for a certain role and that it is the author of a message. We show that all of these properties can be obtained departing from an IND-CPA secure ECW scheme instantiated over a natural PoS blockchain (*e.g.* [9]). First, we observe that VRF-based lottery schemes implemented in many PoS blockchains are sufficient to achieve properties 1, 2 and 3. We then observe that natural block authentication mechanisms used in such PoS blockchains can be used to obtain property 4. Finally, we show that standard techniques can be used to obtain an IND-CCA secure ECW scheme from an IND-CPA secure ECW scheme.

Constructing EtF from ECW (interactively)—Section 7. Since we argued the implausibility of constructing EtF non-interactively from standard assumptions, we study how to transform an ECW scheme into an unrestricted EtF scheme when given access to an auxiliary committee but with "low communication" (and still from standard assumptions). We explain what we mean by "low communication" by an example of its opposite: in previous works ([3,15,16]) successive committees were required to store and reshare secret shares of every message to be sent to a party selected in the future. That is, their communication complexity grows both with the number and the amount and length of the encrypted messages. In contrast, our solution has communication complexity independent of the plaintext length. How our transformation from ECW to EtF works: we associate each role in the future to a unique identity of an Identity Based Encryption scheme (IBE); to encrypt a message towards a role we apply the encryption of the IBE scheme. When, at any point in the future, a party for that role is selected, a committee generates and delivers the corresponding secret key for that role/identity. To realize the latter step, we apply YOSO MPC instantiated from ECW as shown in Section 6. In contrast to previous schemes, our auxiliary committee only needs to hold shares of the IBE's master secret key and so it performs communication/computation dependent on the security parameter but not on the length/amount of messages encrypted to the future.

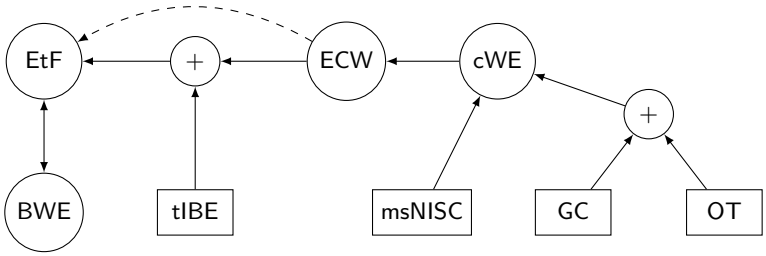

Fig. 1. Dependency diagram for primitives in this work. Legend: primitives wrapped in circles are introduced in this work; $A \rightarrow B$: *"We can construct B from A"*; $A \dashrightarrow B$: *"A is a special case of B"*.

1.3 Previous Works

We compare previous works related to our notions of EtF and ECW (encryption to future and current winner, respectively) in Fig. 2.

Type	Scheme	Communication	Committee?	Interaction?
ECW	CaBKaS [3]	$O(1)$	yes	yes
	RPIR [16]	$O(1)$	yes	yes
	cWE (MS-NISC) (Sec. 4.2)	$O(N)$	no	no*
	cWE (GC+OT) (Sec. 4.2)	$O(N)$	no	no*
EtF	IBE (Sec. 7)	$O(1)$	yes	yes
	WEB [18]	$O(M)$	yes	yes
	Full-fledged WE	$O(1)$	no	no

Fig. 2. The column "Committee?" indicates whether a committee is required. The column "Communication" refers to the communication complexity in terms of the number of all parties N, and the number of plaintexts (called deposited secrets in [18]) M of a given fixed length. We denote by an asterisk non-interactive solutions that require sending a first reusable message during the initial step.

Encryption to the Current Winner (ECW). We recall that ECW is an easier setting than EtF: both the stake distribution and the randomness extracted from the blockchain are static and known at the time of encryption. This means that all of the parameters except the secret key of the lottery winner are available to the encryption algorithm. We now survey works that solved this problem and compare them to our solutions:

– "Can a Blockchain Keep a Secret?" (CaBKaS) [3]. The work of [3] addresses the setting where a dynamically changing committee (over a public blockchain) maintains a secret. The main challenge in order for the committee to *securely* reshare its secret can be summarized as: how to select a

small committee from a large population of parties so that everyone can send secure messages to the committee members without knowing who they are? The solution of [3] is to select the "secret-holding" committee by having another committee, a "nominating committee", that nominates members of the former (while the members of the nominating committee are self-nominated). One can see the nominating committee as a tool providing the ECW functionality. A major caveat in such a solution, however, is that to guarantee an honest majority in the committees, [3] can only tolerate up to 1/4 as the fraction of corrupted parties. This is because corrupted nominators can always select corrupted parties, whereas honest nominators may select corrupted parties by chance. We can improve this through our non-interactive ECW: we can remove the nominating committee and just let the current committee ECW-encrypt their secret shares to the roles of the next committee.

– "Random-Index PIR" (RPIR) [16]. The recent work of [16] defines a new flavour of Private Information Retrieval (PIR) called Random-index PIR (or RPIR) that allows each committee to perform the nomination task by themselves. While RPIR improves on [3] (not requiring a nominating committee and tolerating up to 1/2 of corrupted parties), its constructions are inefficient, either based on Mix-Nets or Fully Homomorphic Encryption (FHE). The construction based on Mix-Nets uses k shufflers, where k is the security parameter, and has an impractical communication complexity of $O(nk^2)$, where n is the number of public keys that each shuffler broadcasts. The FHE-based construction gives a total communication complexity of $O(k^3)$ where $O(k)$ is the length of an FHE decryption share.

WE over commitments (cWE). Benhamouda and Lin [4] defined a type of witness encryption, called "Witness Encryption for NIZK of Commitments". In their setting, parties first commit to their private inputs once and for all. Later, an encryptor can produce a ciphertext so that any party with a committed input that satisfies the relation (specified at encryption time) can decrypt. More accurately, who can decrypt is any party *with a NIZK showing that the committed input satisfies the relation.* The authors construct this primitive based on standard assumptions in asymmetric bilinear groups.

In our work, we generalize the encryption notion in [4], formalize it as cWE and finally use it to construct ECW. While the original construction of [4] fits the definition of cWE, we observe it is an overkill for our application. Specifically our setting does not require NIZKs to be involved in encryption/decryption. We instead give more efficient instantiations based on two-party Multi-Sender Non-Interactive Secure Computation (MS-NISC) protocols and Oblivious Transfer plus Garbled Circuits.

Encryption to the Future (EtF). The general notion of EtF is significantly harder to realize than ECW (as we show in Section 8). Below we discuss natural ideas to obtain EtF. They can be seen as illustrating two extremes where our approach (Section 7) lies in the middle.

– Non-Interactive—Using Witness Encryption [14]: One trivial approach to realize EtF is to use full-fledged general Witness Encryption [14] (WE) for the arithmetic relation $\mathcal{R}$ being the lottery predicate such that the party who holds a winning secret key sk can decrypt the ciphertext. However, constructing a general witness encryption scheme [14] which we can instantiate reliably is still an open problem. Existing constructions rely on very strong assumptions such as multilinear maps, indistinguishability obfuscation or other complexity theoretical conjectures [2]. The challenges in applying this straightforward solution are not surprising given our result showing that EtF implies a flavor of WE.
– Interactive—Multiple Committees and Continuous Executions of ECW: A simple way to achieve an interactive version of EtF is to first encrypt secret shares of a message towards members of a committee that then re-share their secrets towards members of a future anonymous committee via an invocation of ECW (in our instantiations or those in [3] and [16]). This is essentially the solution proposed in CaBKaS [3] where committees interact in order to carry a secret (on the blockchain) into the future. Notice that, for a fixed security parameter and corruption ratio, the communication complexity of the protocol executed by the committee in this solution depends on the plaintext message length. On the other hand, for a fixed security parameter and corruption ratio, the communication complexity of our committee-based transformation from ECW to EtF is *constant*.

Other works. Using blockchains in order to construct non-interactive primitives with game-based security has been previously considered in [17]. Other approaches for transferring secret state to future committees have been proposed in [18], although anonymity is not a concern in this setting. On the other hand, using anonymity to overcome adaptive corruption has been proposed in [12], although this work considers anonymous channels among a fixed set of parties.

2 Preliminaries

Notation. For any positive integer n, $[n]$ denotes the set $\{1, \ldots, n\}$. We use λ to denote the security parameter. We write $a \xleftarrow{\$} S$ to denote that a is sampled according to distribution S, or uniformly randomly if S is a set. We write $A(x; r)$ to denote the output of algorithm A given an input x and a random tape r.

2.1 Proof-of-Stake (PoS) Blockchains

In this work we rely on PoS-based blockchain protocols. In such a protocol, each participant is associated with some stake in the system. A process called leader election encapsulates a lottery mechanism that ensures (of all eligible parties) each party succeeds in generating the next block with probability proportional to its stake in the system. In order to formally argue about executions of such protocols, we depart from the framework presented in [17] which, in turn, builds

on the analysis done in [13] and [21]. We invite the reader to re-visit the abstraction used in [17]. We present a summary of the framework in the full version [7] and discuss below the main properties we will use in the remainder of this paper. Moreover, we note that in [17] it is proven that there exist PoS blockchain protocols with the properties described below, *e.g.* Ouroboros Praos [9].

Blockchain Structure. A genesis block $B_0 = \{(\mathsf{Sig.pk}_1, \mathsf{aux}_1, \mathsf{stake}_1), \ldots, (\mathsf{Sig.pk}_n, \mathsf{aux}_n, \mathsf{stake}_n), \mathsf{aux}\}$ associates each party P_i to a signature scheme public key $\mathsf{Sig.pk}_i$, an amount of stake stake_i and auxiliary information aux_i (*i.e.* any other relevant information required by the blockchain protocol, such as verifiable random function public keys). A blockchain $\mathbf{B}$ relative to a genesis block B_0 is a sequence of blocks $B_1, \ldots, B_n$ associated with a strictly increasing sequence of slots $\mathsf{sl}_1, \ldots, \mathsf{sl}_m$ such that $B_i = (\mathsf{sl}_j, H(B_{i-1}), \mathsf{d}, \mathsf{aux}))$. Here, sl_j indicates the time slot that B_i occupies, $H(B_{i-1})$ is a collision resistant hash of the previous block, d is data and aux is auxiliary information required by the blockchain protocol (*e.g.* a proof that the block is valid for slot sl_j). We denote by $\mathbf{B}^{\lceil \ell}$ the chain (sequence of blocks) $\mathbf{B}$ where the last ℓ blocks have been removed and if $\ell \geq |\mathbf{B}|$ then $\mathbf{B}^{\lceil \ell} = \epsilon$. Also, if $\mathbf{B}_1$ is a prefix of $\mathbf{B}_2$ we write $\mathbf{B}_1 \preceq \mathbf{B}_2$. Each party participating in the protocol has public identity P_i and most messages will be a transaction of the following form: $m = (P_i, P_j, \mathsf{q}, \mathsf{aux})$ where P_i transfers q coins to P_j along with some optional, auxiliary information aux.

Blockchain Setup and Key Knowledge. As in [9], we assume that the genesis block is generated by an initialization functionality $\mathcal{F}_{\mathsf{INIT}}$ that registers all parties' keys. Moreover, we assume that primitives specified in separate functionalities in [9] as incorporated into $\mathcal{F}_{\mathsf{INIT}}$. $\mathcal{F}_{\mathsf{INIT}}$ is executed by the environment $\mathcal{Z}$ as defined below and is parameterized by a stake distribution associating each party P_i to an initial stake stake_i. Upon being activated by P_i for the first time, $\mathcal{F}_{\mathsf{INIT}}$ generates a signature key pair $\mathsf{Sig.sk}_i, \mathsf{Sig.pk}_i$, auxiliary information aux_i and a lottery witness $\mathsf{sk}_{L,i}$, which will be defined as part of the lottery predicate in Section 2.1, sending $(\mathsf{Sig.sk}_i, \mathsf{Sig.pk}_i, \mathsf{aux}_i, \mathsf{sk}_{L,i}, \mathsf{stake}_i)$ to P_i as response. After all parties have activated $\mathcal{F}_{\mathsf{INIT}}$, it responds to requests for a genesis block by providing $B_0 = \{(\mathsf{Sig.pk}_1, \mathsf{aux}_1, \mathsf{stake}_1), \ldots, (\mathsf{Sig.pk}_n, \mathsf{aux}_n, \mathsf{stake}_n), \mathsf{aux}\}$, where aux is generated according to the underlying blockchain consensus protocol.

Since $\mathcal{F}_{\mathsf{INIT}}$ generates keys for all parties, we capture the fact that even corrupted parties have registered public keys and auxiliary information such that they know the corresponding secret keys. Moreover, when our *EtF* constructions are used as part of more complex protocols, a simulator executing the *EtF* and its underlying blockchain with the adversary will be able to predict which ciphertexts can be decrypted by the adversary by simulating $\mathcal{F}_{\mathsf{INIT}}$ and learning these keys. This fact will be important when arguing the security of protocols that use our notion of EtF.

Evolving Blockchains. In order to define an EtF scheme, some concept of future needs to be established. In particular we want to make sure that the initial

chain $\mathbf{B}$ has "correctly" evolved into the final chain $\tilde{\mathbf{B}}$. Otherwise, the adversary can easily simulate a blockchain where it wins a future lottery and finds itself with the ability to decrypt. Fortunately, the *Distinguishable Forking* property provides just that (see full version [7] and [17] for more details). A sufficiently long chain in an honest execution can be distinguished from a fork generated by the adversary by looking at the combined amount of stake proven in such a sequence of blocks. We encapsulate this property in a predicate called $\mathsf{evolved}(\cdot,\cdot)$. First, let $\Gamma^V = (\mathsf{UpdateState}^V, \mathsf{GetRecords}, \mathsf{Broadcast})$ be a blockchain protocol with validity predicate V and where the $(\alpha, \beta, \ell_1, \ell_2)$-*distinguishable forking* property holds. And let $\mathbf{B} \leftarrow \mathsf{GetRecords}(1^\lambda, \mathsf{st})$ and $\tilde{\mathbf{B}} \leftarrow \mathsf{GetRecords}(1^\lambda, \tilde{\mathsf{st}})$.

Definition 1 (Evolved Predicate). *An evolved predicate is a polynomial time function* $\mathsf{evolved}$ *that takes as input blockchains* $\mathbf{B}$ *and* $\tilde{\mathbf{B}}$

$$\mathsf{evolved}(\mathbf{B}, \tilde{\mathbf{B}}) \in \{0, 1\}$$

It outputs 1 iff $\mathbf{B} = \tilde{\mathbf{B}}$ *or the following holds (i)* $V(\mathbf{B}) = V(\tilde{\mathbf{B}}) = 1$*; (ii)* $\mathbf{B}$ *and* $\tilde{\mathbf{B}}$ *are consistent i.e.* $\mathbf{B}^{\lceil\kappa} \preceq \tilde{\mathbf{B}}$ *where* κ *is the common prefix parameter; (iii) Let* $\ell' = |\tilde{\mathbf{B}}| - |\mathbf{B}|$ *then it holds that* $\ell' \geq \ell_1 + \ell_2$ *and* $\mathsf{u\text{-}stakefrac}(\tilde{\mathbf{B}}, \ell' - \ell_1) > \beta$.

Blockchain Lotteries. Earlier we mentioned the concept of leader election in PoS-based blockchain protocols. In this kind of lottery any party can win the right to become a slot leader with a probability proportional to its relative stake in the system. Usually, the lottery winner wins the right to propose a new block for the chain, introduce new randomness to the system or become a part of a committee that carries out some computation. In our encryption scheme we take advantage of this inherent lottery mechanism.

Independent Lotteries. In some applications it is useful to conduct multiple independent lotteries for the same slot sl. Therefore we associate each slot with a set of roles $\mathsf{R}_1, \ldots, \mathsf{R}_n$. Depending on the lottery mechanism, each pair $(\mathsf{sl}, \mathsf{R}_i)$ may yield zero, one or multiple winners. Often, a party can locally compute if it, in fact, is the lottery winner for a given role and the evaluation procedure may equip the party with a proof for others to verify. The below definition details what it means for a party to win a lottery.

Definition 2 (Lottery Predicate). *A lottery predicate is a polynomial time function* $\mathsf{lottery}$ *that takes as input a blockchain* $\mathbf{B}$*, a slot* sl*, a role* R *and a lottery witness* $\mathsf{sk}_{L,i}$ *and outputs 1 if and only if the party owning* $\mathsf{sk}_{L,i}$ *won the lottery for the role* R *in slot* sl *with respect to the blockchain* $\mathbf{B}$.
Formally, we write

$$\mathsf{lottery}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}_{L,i}) \in \{0, 1\}$$

It is natural to establish the set of lottery winning keys $\mathcal{W}_{\mathbf{B},\mathsf{sl},\mathsf{R}}$ for parameters $(\mathbf{B}, \mathsf{sl}, \mathsf{R})$. This is the set of eligible keys satisfying the $\mathsf{lottery}$ predicate.

2.2 Commitment Schemes

We recall the syntax for a commitment scheme $\mathsf{C} = (\mathsf{Setup}, \mathsf{Commit})$ below:

- $\mathsf{Setup}(1^\lambda) \to \mathsf{ck}$ outputs a commitment key. The commitment key ck defines a message space $\mathcal{S}_m$ and a randomizer space $\mathcal{S}_r$.
- $\mathsf{Commit}(\mathsf{ck}, \mathsf{s}; \rho) \to \mathsf{cm}$ outputs a commitment given as input a message $\mathsf{s} \in \mathcal{S}_m$ and randomness $\rho \in \mathcal{S}_r$.

We require a commitment scheme to satisfy the standard properties of *binding* and *hiding*. It is binding if no efficient adversary can come up with two pairs $(\mathsf{s}, \rho), (\mathsf{s}', \rho')$ such that $\mathsf{s} \neq \mathsf{s}'$ and $\mathsf{Commit}(\mathsf{ck}, \mathsf{s}; \rho) = \mathsf{Commit}(\mathsf{ck}, \mathsf{s}'; \rho')$ for $\mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda)$. The scheme is hiding if for any two $\mathsf{s}, \mathsf{s}' \in \mathcal{S}_m$, no efficient adversary can distinguish between a commitment of s and one of s'.

Extractability. In our construction of ECW from cWE (Section 5.1), we require our commitments to satisfy an additional property which allows to *extract* message and randomness of a commitment. In particular we assume that our setup outputs both a commitment key and a trapdoor td and that there exists an algorithm Ext such that $\mathsf{Ext}(\mathsf{td}, \mathsf{cm})$ outputs (s, ρ) such that $\mathsf{cm} = \mathsf{Commit}(\mathsf{ck}, \mathsf{s}; \rho)$. We remark we can generically obtain this property by attaching to the commitment a NIZK argument of knowledge that shows knowledge of opening, i.e., for the relation $\mathcal{R}^{\mathrm{opn}}(\mathsf{cm}_i; (\mathsf{s}, \rho)) \iff \mathsf{cm}_i = \mathsf{Commit}(\mathsf{ck}, \mathsf{s}; \rho)$.

2.3 (Threshold) Identity Based Encryption

In an IBE scheme, users can encrypt simply with respect to an *identity* (rather than a public key). Given a master secret key, an IBE can generate secret keys that allows to open to specific identities. In our construction of EtF (Section 7.1) we rely on a *threshold variant of IBE* (TIBE) where no single party in the system holds the master secret key. Instead, parties in a committee hold a partial master secret key msk_i. Like other threshold protocols, threshold IBE can be generically obtained by "lifting" an IBE through a secret sharing with homomorphic properties (see for example [20]).

Threshold IBE. A TIBE system consists of the following algorithms.

$\Pi_{\mathsf{TIBE}}.\mathsf{Setup}(1^\lambda, n, k) \to (\mathsf{sp}, \mathsf{vk}, \vec{\mathsf{msk}})$: It outputs some public system parameters sp (including mpk), verification key vk, and vector of master secret key shares $\vec{\mathsf{msk}} = (\mathsf{msk}_1, \ldots, \mathsf{msk}_n)$ for n with threshold k. We assume that all algorithms takes sp as input implicitly.

$\Pi_{\mathsf{TIBE}}.\mathsf{ShareKG}(i, \mathsf{msk}_i, \mathsf{ID}) \to \theta = (i, \hat{\theta})$: It outputs a private key share $\theta = (i, \hat{\theta})$ for ID given a share of the master secret key.

$\Pi_{\mathsf{TIBE}}.\mathsf{ShareVerify}(\mathsf{vk}, \mathsf{ID}, \theta) \to 0/1$: It takes as input the verification key vk, an identity ID, and a share of master secret key θ, and outputs 0 or 1.

$\Pi_{\mathsf{TIBE}}.\mathsf{Combine}(\mathsf{vk}, \mathsf{ID}, \vec{\theta}) \to \mathsf{sk}_{\mathsf{ID}}$: It combines the shares $\vec{\theta} = (\theta_1, \ldots, \theta_k)$ to produce a private key $\mathsf{sk}_{\mathsf{ID}}$ or $\perp$.

$\Pi_{\mathsf{TIBE}}.\mathsf{Enc}(\mathsf{ID}, m) \rightarrow \mathsf{ct}$: It encrypts message m for identity ID and outputs a ciphertext ct.

$\Pi_{\mathsf{TIBE}}.\mathsf{Dec}(\mathsf{ID}, \mathsf{sk}_{\mathsf{ID}}, \mathsf{ct}) \rightarrow m$: It decrypts the ciphertext ct given a private key $\mathsf{sk}_{\mathsf{ID}}$ for identity ID.

Correctness. A TIBE scheme Π_{TIBE} should satisfy two correctness properties:

1. For any identity ID, if $\theta = \Pi_{\mathsf{TIBE}}.\mathsf{ShareKG}(i, \mathsf{msk}_i, \mathsf{ID})$ for $\mathsf{msk}_i \in \vec{\mathsf{msk}}$, then $\Pi_{\mathsf{TIBE}}.\mathsf{ShareVerify}(\mathsf{vk}, \mathsf{ID}, \theta) = 1$.
2. For any ID, if $\vec{\theta} = \{\theta_1, \ldots, \theta_k\}$ where $\theta_i = \Pi_{\mathsf{TIBE}}.\mathsf{ShareKG}(i, \mathsf{msk}_i, \mathsf{ID})$, and $\mathsf{sk}_{\mathsf{ID}} = \Pi_{\mathsf{TIBE}}.\mathsf{Combine}(\mathsf{vk}, \mathsf{ID}, \vec{\theta})$, then for any $m \in \mathcal{M}$ and $\mathsf{ct} = \Pi_{\mathsf{TIBE}}.\mathsf{Enc}(\mathsf{ID}, m)$ we have $\Pi_{\mathsf{TIBE}}.\mathsf{Dec}(\mathsf{ID}, \mathsf{sk}_{\mathsf{ID}}, \mathsf{ct}) = m$.

Structural Property: TIBE as IBE + Secret Sharing. We model threshold IBE in a modular manner from IBE and assume it to have a certain structural property: that it can be described as an IBE "lifted" through a homomorphic secret-sharing [6,5,20]. TIBE constructions can often be described as such. We assume this structural property to present our proofs for EtF modularly, but we remark our results do not depend on it and they hold for an arbitrary TIBE. For lack of space we refer the reader to the full version for details.

Assume a secure IBE (the non-threshold variant of TIBE). We can transform it into a threshold IBE using homomorphic secret sharing algorithms $(\mathsf{Share}, \mathsf{EvalShare}, \mathsf{Combine})$. A homomorphic secret sharing scheme is a secret sharing scheme with an extra property: given a shared secret, it allows to compute a share of a function of the secret on it. The correctness of the homomorphic scheme requires that running $y_i \leftarrow \mathsf{EvalShare}(\mathsf{msk}_i, f)$ on msk_i output of Share and then running $\mathsf{Combine}$ on (a large enough set of) the y_i-s produces the same output as $f(\mathsf{msk})$. We also require that $\mathsf{Combine}$ can reconstruct msk from a large enough set of the msk_i-s. For security we assume we can simulate the shares not available to the adversaries (if the adversary holds at most $T = k$ shares). For the resulting TIBE's security we assume that, for an adversary holding at most T shares, we can simulate: master secret key shares not held by the adversary (*msk shares simulation*) and shares of the id-specific keys (*key-generation simulation*) for the same shares. We finally assume we can verify that each of the id-specific key shares are authenticated (*robustness*) and that shares of the master secret key can be reshared (*proactive resharing*).

3 Modelling EtF

In this section, we present a model for encryption to the future winner of a lottery. In order to argue about a notion of future, we use the blocks of an underlying blockchain ledger and their relative positions in the chain to specify points in time. Intuitively, our notion allows for creating ciphertexts that can only be decrypted by a party that is selected to perform a certain role R at a future slot sl according to a lottery scheme associated with a blockchain protocol. The winner of the lottery at a point in the future with respect to a blockchain

state $\tilde{\mathbf{B}}$ is determined by the lottery predicate defined in Section 2.1, *i.e.* the winner is the holder of a lottery secret key sk such that $\mathsf{lottery}(\tilde{\mathbf{B}}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}) = 1$. However, notice that the winner might only be determined by a blockchain state produced in the future as a result of the blockchain protocol execution. This makes it necessary for the ciphertext to encode an initial state $\mathbf{B}$ of the blockchain that allows for verifying that a future state $\tilde{\mathbf{B}}$ (presented at the time of decryption) has indeed been produced as a result of correct protocol execution. This requirement is captured by the evolving blockchain predicate defined in Section 2.1, *i.e.* $\mathsf{evolved}(\mathbf{B}, \tilde{\mathbf{B}}) = 1$ iff $\tilde{\mathbf{B}}$ is obtained as a future state of executing the blockchain protocol departing from $\mathbf{B}$.

Definition 3 (Encryption to the Future). *A pair of PPT algorithms* $\mathcal{E} = (\mathsf{Enc}, \mathsf{Dec})$ *in the context of a blockchain* Γ^V *is an EtF-scheme with evolved predicate* $\mathsf{evolved}$ *and a lottery predicate* $\mathsf{lottery}$*. The algorithms work as follows.*

Encryption. $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, m)$ *takes as input an initial blockchain* $\mathbf{B}$*, a slot* sl*, a role* R *and a message* m*. It outputs a ciphertext* ct *- an encryption to the future.*

Decryption. $m/\perp \leftarrow \mathsf{Dec}(\tilde{\mathbf{B}}, \mathsf{ct}, \mathsf{sk})$ *takes as input a blockchain state* $\tilde{\mathbf{B}}$*, a ciphertext* ct *and a secret key* sk *and outputs the original message* m *or* $\perp$*.*

An EtF must satisfy the following properties:

Correctness. An EtF-scheme is said to be correct if for honest parties i and j, there exists a negligible function μ such that for all $\mathsf{sk}, \mathsf{sl}, \mathsf{R}, m$:

$$\left| \Pr \left[\begin{array}{c} \mathsf{view} \leftarrow \mathsf{EXEC}^{\Gamma}(\mathcal{A}, \mathcal{Z}, 1^{\lambda}) \\ \mathbf{B} = \mathsf{GetRecords}(\mathsf{view}_i) \\ \tilde{\mathbf{B}} = \mathsf{GetRecords}(\mathsf{view}_j) \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, m) \\ \mathsf{evolved}(\mathbf{B}, \tilde{\mathbf{B}}) = 1 \end{array} : \begin{array}{c} \mathsf{lottery}(\tilde{\mathbf{B}}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}) = 0 \\ \vee\ \mathsf{Dec}(\tilde{\mathbf{B}}, \mathsf{ct}, \mathsf{sk}) = m \end{array} \right] - 1 \right| \leq \mu(\lambda)$$

Security. We establish a game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$. In Section 2.1 we describe how $\mathcal{A}$ and $\mathcal{Z}$ execute a blockchain protocol. In addition, we now let the adversary interact with the challenger in a game $\mathrm{Game}^{\mathsf{IND\text{-}CPA}}_{\Gamma,\mathcal{A},\mathcal{Z},\mathcal{E}}$ described in Algorithm 1. The game can be summarized as follows:

1. $\mathcal{A}$ executes the blockchain protocol Γ together with $\mathcal{Z}$ and at some round r chooses a blockchain $\mathbf{B}$, a role R for the slot sl and two messages m_0 and m_1 and sends it all to $\mathcal{C}$.
2. $\mathcal{C}$ chooses a random bit b and encrypts the message m_b with the parameters it received and sends ct to $\mathcal{A}$.
3. $\mathcal{A}$ continues to execute the blockchain until some round $\tilde{r}$ where the blockchain $\tilde{\mathbf{B}}$ is obtained and $\mathcal{A}$ outputs a bit b'.

If the adversary is a lottery winner for the challenge role R in slot sl, the game outputs a random bit. If the adversary is not a lottery winner for the challenge role R in slot sl, the game outputs $b \oplus b'$. The reason for outputting

a random guess in the game when the challenge role is corrupted is as follows. Normally the output of the IND-CPA game is $b \oplus b'$ and we require it to be 1 with probability 1/2. This models that the guess b' is independent of b. This, of course, cannot be the case when the challenge role is corrupted. We therefore output a random guess in these cases. After this, any bias of the output away from 1/2 still comes from b' being dependent on b.

Algorithm 1 $\text{Game}^{\mathsf{IND\text{-}CPA}}_{\Gamma,\mathcal{A},\mathcal{Z},\mathcal{E}}$

$\mathsf{view}^r \leftarrow \mathsf{EXEC}^{\Gamma}_r(\mathcal{A}, \mathcal{Z}, 1^\lambda)$ ▷ $\mathcal{A}$ executes Γ with $\mathcal{Z}$ until round r
$(\mathbf{B}, \mathsf{sl}, \mathsf{R}, m_0, m_1) \leftarrow \mathcal{A}(\mathsf{view}^r_{\mathcal{A}})$ ▷ $\mathcal{A}$ outputs challenge parameters
$b \xleftarrow{\$} \{0,1\}$
$\mathsf{ct} \leftarrow \mathsf{Enc}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, m_b)$
$\mathsf{st} \leftarrow \mathcal{A}(\mathsf{view}^r_{\mathcal{A}}, \mathsf{ct})$ ▷ $\mathcal{A}$ receives challenge ct
$\mathsf{view}^{\tilde{r}} \leftarrow \mathsf{EXEC}^{\Gamma}_{(\mathsf{view}^r, \tilde{r})}(\mathcal{A}, \mathcal{Z}, 1^\lambda)$ ▷ Execute from view^r until round $\tilde{r}$
$(\tilde{\mathbf{B}}, b') \leftarrow \mathcal{A}(\mathsf{view}^{\tilde{r}}_{\mathcal{A}}, \mathsf{st})$
if $\mathsf{evolved}(\mathbf{B}, \tilde{\mathbf{B}}) = 1$ **then** ▷ $\tilde{\mathbf{B}}$ is a valid evolution of $\mathbf{B}$
  if $sk^{\mathcal{A}}_{L,j} \notin \mathcal{W}_{\tilde{\mathbf{B}},\mathsf{sl},\mathsf{R}}$ **then** ▷ $\mathcal{A}$ does not win role R
    return $b \oplus b'$
  end if
end if
return $\hat{b} \xleftarrow{\$} \{0,1\}$

Definition 4 (IND-CPA Secure EtF). *An EtF-scheme $\mathcal{E} = (\mathsf{Enc}, \mathsf{Dec})$ in the context of a blockchain protocol Γ executed by PPT machines $\mathcal{A}$ and $\mathcal{Z}$ is said to be* IND-CPA *secure if, for any $\mathcal{A}$ and $\mathcal{Z}$, there exists a negligible function μ such that for $\lambda \in \mathbb{N}$:*

$$\left| 2 \cdot \Pr\left[\text{Game}^{\mathsf{IND\text{-}CPA}}_{\Gamma,\mathcal{A},\mathcal{Z},\mathcal{E}} = 1 \right] - 1 \right| \leq \mu(\lambda)$$

Remark 1 (On the requirement of Proof-of-Stake for EtF). The EtF notion requires the guarantee that an honest chain should be verifiable without interaction with the network (i.e. verified by the EtF ciphertext). While this is possible for Proof-of-Stake (PoS) blockchains, in a Proof-of-Work (PoW) blockchain the adversary can always simulate a chain where it generates all blocks. In general we require a blockchain in order to model time (via block height) for EtF.

3.1 ECW as a Special Case of EtF

In this section we focus on a special class of EtF. We call schemes in this class ECW schemes. ECW is particularly interesting since the underlying lottery is always conducted with respect to the current blockchain state. This has the following consequences

1. $\mathbf{B} = \tilde{\mathbf{B}}$ means that $\mathsf{evolved}(\mathbf{B}, \tilde{\mathbf{B}}) = 1$ is trivially true.
2. The winner of role R in slot sl is already defined in $\mathbf{B}$.

It is easy to see that this kind of EtF scheme is simpler to realize since there is no need for checking if the blockchain has "correctly" evolved. Furthermore, all lottery parameters like stake distribution and randomness extracted from the blockchain are static. Thus, an adversary has no way to move stake between accounts in order to increase its chance of winning the lottery.
Note that, when using an ECW scheme, the lottery winner is already decided at encryption time. In other words, there is no delay and the moment a ciphertext is produced the receiver is chosen.

4 Witness Encryption over Commitments (cWE)

Here, we describe witness encryption over commitments that is a relaxed notion of witness encryption. In witness encryption parties encrypt to a public input for some NP statement. In cWE we have two phases: first parties provide a (honestly generated) commitment cm of their private input s. Later, anybody can encrypt to a public input for an NP statement which *also* guarantees correct opening of the commitment. Importantly, in applications, the first message in our model can be reused for many different invocations.

Remark 2 (Comparing cWE and WE). We observe that cWE is weaker than standard WE because of its deterministic flavor. In standard WE we encrypt without having any "pointer" to an alleged witness, but in cWE it requires the witness to be implicitly *known* at encryption time through the commitment (to which it is bound). That is why—as for the weak flavors of witness encryption in [4]—we believe it would be misleading to just talk about WE. This is true in particular since we show cWE can be constructed from standard assumptions such as oblivious transfer and garbled circuits (see full version [7]), whereas constructions of WE from standard assumptions are still an open problem or require strong primitives like indistinguishability obfuscation. Finally we stress a difference with the trivial "interactive" WE proposed in [14] (Section 1.3): cWE is still non-interactive after producing a once-and-for-all reusable commitment.

4.1 Definition

The type of relations we consider are of the following form: a statement $\mathtt{x} = (\mathsf{cm}, C, y)$ and a witness $\mathtt{w} = (\mathsf{s}, \rho)$ are in the relation (i.e., $(\mathtt{x}, \mathtt{w}) \in \mathcal{R}$) iff "$\mathsf{cm}$ commits to some secret value s using randomness ρ, and $C(\mathsf{s}) = y$". Here, C is a circuit in some circuit class $\mathcal{C}$ and y is the expected output of the function.
Formally, we define witness encryption over commitments as follows:

Definition 5 (Witness encryption over commitments). *Let $\mathsf{C} = (\mathsf{Setup}, \mathsf{Commit})$ be a non-interactive commitment scheme. A cWE-scheme for witness encryption over commitments with circuit class $\mathcal{C}$ and commitment scheme C consists of a pair of algorithms $\Pi_{\mathsf{cWE}} = (\mathsf{Enc}, \mathsf{Dec})$:*

Encryption phase. $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{ck}, \mathtt{x}, m)$ *on input a commitment key* ck*, a statement* $\mathtt{x} = (\mathsf{cm}, C, y)$ *such that* $C \in \mathcal{C}$*, and a message* $m \in \{0,1\}^*$*, generates a ciphertext* ct*.*

Decryption phase. $m/\perp \leftarrow \mathsf{Dec}(\mathsf{ck}, \mathsf{ct}, \mathtt{w})$ *on input a commitment key* ck*, a ciphertext* ct*, and a witness* $\mathtt{w}$*, returns a message* m *or* $\perp$*.*

A cWE should satisfy *correctness* and *semantic security* as defined below.

(Perfect) Correctness. An honest prover with a statement $\mathtt{x} = (\mathsf{cm}, C, y)$ and witness $\mathtt{w} = (\mathsf{s}, \rho)$ such that $\mathsf{cm} = \mathsf{Commit}(\mathsf{ck}, \mathsf{s}; \rho)$ and $C(\mathsf{s}) = y$ can always decrypt with overwhelming probability. More precisely, a cWE with circuit class $\mathcal{C}$ and commitment scheme C has perfect correctness if for all $\lambda \in \mathbb{N}$, $C \in \mathcal{C}$, $\mathsf{ck} \in \mathsf{Range}(\mathsf{C.Setup})$, $\mathsf{s} \in \mathcal{S}_m$, randomness $\rho \in \mathcal{S}_r$, commitment $\mathsf{cm} \leftarrow \mathsf{C.Commit}(\mathsf{ck}, \mathsf{s}; \rho)$, and bit message $m \in \{0,1\}^*$, it holds that

$$\Pr\left[\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{ck}, (\mathsf{cm}, C, C(\mathsf{s})), m); m' \leftarrow \mathsf{Dec}(\mathsf{ck}, \mathsf{ct}, (\mathsf{s}, \rho)) \ : \ m = m'\right] = 1$$

(Weak) Semantic Security. Intuitively, encrypting with respect to a false statement (with honest commitment) produces indistinguishable ciphertexts. Formally, there exists a negligible function μ such that for all $\lambda \in \mathbb{N}$, all auxiliary strings aux and all PPT adversaries $\mathcal{A}$:

$$\left| 2 \cdot \Pr\left[\begin{array}{r} \mathsf{ck} \leftarrow \mathsf{C.Setup}(1^\lambda) \\ (\mathsf{st}, \mathsf{s}, \rho, C, y, m_0, m_1) \leftarrow \mathcal{A}(\mathsf{ck}, \mathsf{aux}) \\ \mathsf{cm} \leftarrow \mathsf{C.Commit}(\mathsf{ck}, \mathsf{s}; \rho); b \xleftarrow{\$} \{0,1\} \ : \ \mathcal{A}(\mathsf{st}, \mathsf{ct}) = b \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{ck}, (\mathsf{cm}, C, y), m_b) \\ \mathsf{ct} := \perp \text{ if } C(\mathsf{s}) = y,\ C \notin \mathcal{C} \text{ or } |m_0| \neq |m_1| \end{array}\right] - 1 \right| \leq \mu(\lambda)$$

Later, to show the construction of ECW from cWE, we need a stronger notion of semantic security where the adversary additionally gets to see ciphertexts of the challenge message under true statements with unknown to $\mathcal{A}$ witnesses. We formalize this property inthe full version [7] and show that weak semantic security together with hiding of the commitment imply strong semantic security.

4.2 Constructions of cWE

From Multi-Sender 2P-NISC [1]. A cWE scheme can be constructed from protocols for Multi-Sender (reusable) Non-Interactive Secure Computation (MS-NISC) [1]. In such protocols, there is a receiver R with input x who first broadcasts an encoding of its input, and then later every sender S_i with input y_i can send a single message to R that conveys only $f(x, y_i)$. This is done while preserving privacy of inputs and correctness of output.

In the full version [7] we provide a detailed explanation of how to construct cWE using MS-NISC as in [1]. We here state the main points of the construction. Let f be the function that on input $y = (\mathtt{x}, k)$ and $x = \mathtt{w}$ outputs k if and only if $(\mathtt{x}, \mathtt{w}) \in \mathcal{R}$. This will be the underlying function for the MS-NISC protocol. We then obtain a cWE scheme over the relation $\mathcal{R}$ in the following way:

1. First, the receiver commits to its witness $\mathtt{w}$ by providing an encoding of it as its first message in the MS-NISC protocol.
2. Secondly, to encrypt m under statement $\mathtt{x}$, a sender samples a key k of size $|m|$ and provides an encoding of $(\mathtt{x}, k)$ as the second message in the MS-NISC protocol and sends the ciphertext $\mathsf{ct} = m \oplus k$ to the receiver.
3. Finally, the receiver obtains the key as the output of $f(x = \mathtt{w}, y = (\mathtt{x}, k)) = k$ iff $\mathtt{w}$ is a valid witness for the statement $\mathtt{x}$ encoded in the second message. And it decrypts the ciphertext $m = \mathsf{ct} \oplus k$.

We observe that the above construction actually yields a stronger notion of cWE where the statement $\mathtt{x}$ is private which is not a requirement in our setting. This asymmetry between sender and receiver privacy was also observed by others [19] and it opens the door for efficient constructions using oblivious transfer (OT) and privacy-free garbled circuits as described in [23]. More details on the more efficient construction of cWE using OT and garbled circuits are provided inthe full version [7].

5 Construction of ECW

Here we show a novel construction of ECW from cWE. We then show alternative constructions through instantiations from previous work.

5.1 ECW from cWE

In this section we realize the notion of ECW from cWE. We define our scheme with respect to a set of parties $\mathcal{P} = \{P_1, \ldots, P_n\}$ executing a blockchain protocol Γ as described in Section 2.1, *i.e.* each party P_i has access to the blockchain ledger and is associated to a tuple $(\mathsf{Sig.pk}_i, \mathsf{aux}_i, \mathsf{st}_i)$ registered in the genesis block for which it has corresponding secret keys $(\mathsf{Sig.sk}_i, \mathsf{sk}_{L,i})$. Our construction uses as a main building block a witness encryption scheme over commitments $\Pi_{\mathsf{cWE}} = (\mathsf{Enc}_{\mathsf{cWE}}, \mathsf{Dec}_{\mathsf{cWE}})$; we assume the commitments to be extractable. The class of circuits $\mathcal{C}$ of Π_{cWE} includes the lottery predicate $\mathsf{lottery}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}_{L,i})$. We let each party publish an initial commitment of its witness. This way we can do without any interaction for encryption/decryption through a one-time setup where parties publish the commitments over which all following encryptions are done. We construct our ECW scheme Π_{ECW} as follows:

System Parameters: We assume that a commitment key $\mathsf{Setup}(1^\lambda) \to \mathsf{ck}$ is contained in the genesis block B_0 of the underlying blockchain.

Setup Phase: All parties $P_i \in \mathcal{P}$ proceed as follows:

1. Compute a commitment $\mathsf{cm}_i \leftarrow \mathsf{Commit}(\mathsf{ck}, \mathsf{sk}_{L,i}; \rho_i)$ to $\mathsf{sk}_{L,i}$ using randomness ρ_i. We abuse the notation and define P_i's secret key as $\mathsf{sk}_{L,i}||\rho_i$.
2. Compute a signature $\sigma_i \leftarrow \mathsf{Sig}_{\mathsf{Sig.sk}_i}(\mathsf{cm}_i)$.
3. Publish $(\mathsf{cm}_i, \sigma_i)$ on the blockchain by executing $\mathsf{Broadcast}(1^\lambda, (\mathsf{cm}_i, \sigma_i))$.

Encryption $\mathsf{Enc}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, m)$**:** Construct a circuit C that encodes the predicate $\mathsf{lottery}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}_{L,i})$, where $\mathbf{B}$, sl and R are hardcoded and $\mathsf{sk}_{L,i}$ is the witness. Let $\mathcal{P}_{Setup}$ be the set of parties with non-zero relative stake and a valid setup message $(\mathsf{cm}_i, \sigma_i)$ published in the common prefix $\mathbf{B}^{\lceil\kappa}$ (if P_i has published more than one valid $(\mathsf{cm}_i, \sigma_i)$, only the latest one is considered). For every $P_i \in \mathcal{P}_{Setup}$, compute $\mathsf{ct}_i \leftarrow \mathsf{Enc}_{\mathsf{cWE}}(\mathsf{ck}, \mathsf{x}_i = (\mathsf{cm}_i, C, 1), m)$. Output $\mathsf{ct} = \big(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \{\mathsf{ct}_i\}_{P_i \in \mathcal{P}_{Setup}}\big)$.

Decryption $\mathsf{Dec}(\mathbf{B}, \mathsf{ct}, \mathsf{sk})$**:** Given $\mathsf{sk} := \mathsf{sk}_{L,i} || \rho_i$ such that $\mathsf{cm}_i = \mathsf{Commit}(\mathsf{ck}, \mathsf{sk}_{L,i}; \rho_i)$ and $\mathsf{lottery}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}_{L,i}) = 1$ for parameters $\mathbf{B}, \mathsf{sl}, \mathsf{R}$ from ct, output $m \leftarrow \mathsf{Dec}_{\mathsf{cWE}}(\mathsf{ck}, \mathsf{ct}_i, (\mathsf{sk}_{L,i}, \rho_i))$. Otherwise, output $\perp$.

Theorem 1. *Let* $\mathsf{C} = (\mathsf{Setup}, \mathsf{Commit})$ *be a non-interactive extractable commitment scheme and* $\Pi_{\mathsf{cWE}} = (\mathsf{Enc}_{\mathsf{cWE}}, \mathsf{Dec}_{\mathsf{cWE}})$ *be a strong semantically secure cWE over* C *for a circuit class* $\mathcal{C}$ *encoding the lottery predicate* $\mathsf{lottery}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}_{L,i})$ *as defined in Section 4. Let* Γ *be a blockchain protocol as defined in Section 2.1.* Π_{ECW} *is an* IND-CPA*-secure ECW scheme as per Definition 4.*

The proof is provided inthe full version [7].

5.2 Other Instantiations

ECW from target anonymous channels [16,3]. As mentioned before, another approach to construct ECW can be based on a recent line of work that aims to design secure-MPC protocols where parties should remain anonymous until they speak [16,3,15]. The baseline of these results is to establish a communication channel to random parties, while preserving their anonymity. It is quite clear that such anonymous channels can be used to realize our definition of ECW for the underlying lottery predicate that defines to whom the anonymous channel is established. Namely, to encrypt m to a role R at a slot sl with respect to a blockchain state $\mathbf{B}$, create a target anonymous channel to $(\mathsf{R}, \mathsf{sl})$ over $\mathbf{B}$ by using the above approaches and send m via this channel. Depending on the lottery predicate that specifies which random party the channel is created for, a recipient with the secret key who wins this lottery can retrieve m. To include some concrete examples, the work of Benhamouda et al. [3] proposed the idea of using a "nomination" process, where a nominating committee chooses a number of random parties $\mathcal{P}$, look up their public keys, and publish a re-randomization of their key. This allows everyone to send messages to $\mathcal{P}$ while keeping their anonymity. The work of [3] answered this question differently by delegating the nomination task to the previous committees without requiring a nominating committee. That is, the previous committee runs a secure-MPC protocol to choose a random subset of public keys, and broadcasts the rerandomization of the keys. To have a MPC protocol that scales well with the total number of parties, they define a new flavour of private information retrieval (PIR) called random-index PIR (or RPIR) and show how each committee—playing the role of the RPIR client—can select the next committee with the complexity only proportional to the size of the committee. There are two constructions of RPIR

proposed in [16], one based on Mix-Nets and the other based on FHE. Since the purpose of the constructions described is to establish a target-anonymous channel to a random party, one can consider them as examples of a stronger notion of ECW with anonymity and a specific lottery predicate that selects *a single* random party from the entire population as the winner.

ECW from [10]. Derler and Slamanig [10] (DS) constructed a variant of WE for a restricted class of algebraic languages. In particular, a user can conduct a Groth-Sahai (GS) proof for the satisfiability of some pairing-product equations (PPEs). Such a proof contains commitments to the witness using randomness only known by this user. The proof can be used by anyone to encrypt a message resulting in a ciphertext which can only be decrypted by knowing this randomness. More formally, they consider a type of WE associated with a proof system $\Pi = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ consisting of two rounds. In the first round, a recipient computes and broadcasts $\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, \mathtt{x}, \mathtt{w})$. Later, a user can verify the proof and encrypt a message m under $(\mathtt{x}, \pi)$ if $\mathsf{Verify}(\mathsf{crs}, \mathtt{x}, \pi) = 1$. We note that the proof π does not betray the user conducting the proof and therefore it can use an anonymous broadcast channel to communicate the proof to the encrypting party in order to obtain anonymous ECW. Moreover, although GS proofs may look to support only a restricted class of statements based on PPEs, they are expressive enough to cover all the statements arising in pairing-based cryptography. This indicates the applicability of this construction for any VRF-based lottery where the VRF is algebraic and encodable as a set of PPEs. Further details are provided inthe full version [7]. This interactive ECW just described yields an improvement in communication complexity at the cost of having an extra round of interaction.

From Signatures of Knowledge. Besides the above instantiations, we point out a (potentially more inefficient) abstract construction from zero-knowledge signatures of knowledge (SoK) [8] (roughly, a non-malleable non-interactive zero-knowledge proof). This is similar in spirit to the previous instantiation and can be seen as a generalization. Assume each party has a (potentially ephemeral) public key. At the time the lottery winner has been decided, the winners can post a SoK showing knowledge of the secret key corresponding to their pk *and* that their key is a winner of the lottery. To encrypt, one would first verify the SoK and then encrypt with respect to the corresponding public key.

6 YOSO Multiparty Computation from ECW

In this section we show how ECW can be used as the crucial ingredient in setting up a YOSO MPC. So far we have only focused on IND-CPA secure ECW, which falls short of role assignment in the sense of [15]. In general role assignment requires the following properties which are not provided by ECW (or EtF):

1. Multiple parties must be able to send messages to the same role (in most applications this requires IND-CCA).

2. Parties must authenticate messages on behalf of a role they executed in the past (authentication from the past)
3. A party assigned to a given role must stay covert until the role is executed.

We will define a number of properties needed for EtF to realize applications such as role assignment. We start by looking at CCA security for an EtF scheme. We then introduce the notion of Authentication from the Past (AfP) and definition of unforgeability and privacy guarantees. Finally, we introduce the notion of YOSO-friendly blockchains that have inbuilt lotteries with properties that are needed to conduct YOSO MPC and corresponding EtF and AfP schemes.

6.1 IND-CCA EtF

In this section we define what it means for an EtF to be IND-CCA secure. This security property is useful in many applications where more encryptions are done towards the same slot and role. As in the definition of IND-CPA, we establish a game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$. We introduce a decryption oracle, $\mathcal{O}_{\mathsf{EtF}}$, which on input ct returns the decryption of ciphertext. Furthermore, the $\mathcal{O}_{\mathsf{EtF}}$ maintains a list of ciphertext queries $\mathcal{Q}_{\mathsf{EtF}}$. Algorithm 2 shows the details of the game.

Algorithm 2 $\text{Game}^{\mathsf{IND\text{-}CCA2}}_{\Gamma,\mathcal{A},\mathcal{Z},\mathsf{E}}$

$\mathsf{view}^r \leftarrow \mathsf{EXEC}^{\Gamma}_r(\mathcal{A}^{\mathcal{O}_{\mathsf{EtF}}}, \mathcal{Z}, 1^\lambda)$ ▷ $\mathcal{A}$ executes Γ with $\mathcal{Z}$ until round r
$(\mathbf{B}, \mathsf{sl}, \mathsf{R}, m_0, m_1) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{EtF}}}(\mathsf{view}^r_{\mathcal{A}})$ ▷ $\mathcal{A}$ outputs challenge parameters
$b \xleftarrow{\$} \{0,1\}$
$\mathsf{ct} \leftarrow \mathsf{Enc}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, m_b)$
$\mathsf{st} \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{EtF}}}(\mathsf{view}^r_{\mathcal{A}}, \mathsf{ct})$ ▷ $\mathcal{A}$ receives challenge ct
$\mathsf{view}^{\tilde{r}} \leftarrow \mathsf{EXEC}^{\Gamma}_{(\mathsf{view}^r, \tilde{r})}(\mathcal{A}^{\mathcal{O}_{\mathsf{EtF}}}, \mathcal{Z}, 1^\lambda)$ ▷ Execute from view^r until round $\tilde{r}$
$(\tilde{\mathbf{B}}, b') \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{EtF}}}(\mathsf{view}^{\tilde{r}}_{\mathcal{A}}, \mathsf{st})$
if $\mathsf{evolved}(\mathbf{B}, \tilde{\mathbf{B}}) = 1$ **then** ▷ $\tilde{\mathbf{B}}$ is a valid evolution of $\mathbf{B}$
  if $\mathsf{sk}^{\mathcal{A}}_{L,j} \notin \mathcal{W}_{\tilde{\mathbf{B}},\mathsf{R},\mathsf{sl}} \wedge \mathsf{ct} \notin \mathcal{Q}_{\mathsf{EtF}}$ **then** ▷ $\mathcal{A}$ does not win role R
    return $b \oplus b'$
  end if
end if
return $g \xleftarrow{\$} \{0,1\}$

Definition 6 (IND-CCA2 Secure EtF). *Formally, an EtF-scheme $\mathcal{E}$ is said to be IND-CCA2 secure in the context of a blockchain protocol Γ executed by PPT machines $\mathcal{A}$ and $\mathcal{Z}$ if there exists a negligible function μ such that for $\lambda \in \mathbb{N}$:*

$$\left| 2 \cdot \Pr\left[\text{Game}^{\mathsf{IND\text{-}CCA2}}_{\Gamma,\mathcal{A},\mathcal{Z},\mathcal{E}} = 1 \right] - 1 \right| \leq \mu(\lambda)$$

To add IND-CCA2 security to an IND-CPA secure EtF scheme (as defined in Definition 4) we can use standard transformations such as [11,22]. In the

transformation based on [22] we could add to the setup of the blockchain a CRS for a simulation-sound extractable NIZK. When encrypting m to a role R the sender will send along a proof of knowledge of the plaintext m. We get the challenge ciphertext from the IND-CPA game and use the ZK property to simulate the NIZK proof. We can use the extraction trapdoor of the proof system to simulate the CCA decryption oracles by simulation soundness. When the IND-CCA2 adversary makes a guess, we make the same guess. The details of the construction and proof follow using standard techniques and are omitted. On the other hand, the popular transformation of [11] allows for simulating CCA decryption oracles by observing the adversary's queries to a random oracle, which should not be an issue since an EtF scheme is likely already running on top of a blockchain which is secure in the random oracle model. We leave the construction of concretely efficient IND-CCA2 EtF as future work.

6.2 Authentication from the Past (AfP)

When the winner of a role R_1 sends a message m to a future role R_2 then it is typically also needed that R_2 can be sure that the message m came from a party P which, indeed, won the role R_1. Most PoS blockchains deployed in practice have a lottery where a certificate can be released proving that P won the role R_1. In order to formalize this concept, we introduce an AfP scheme with a corresponding EUF-CMA game representing the authentication property.

Definition 7 (Authentication from the Past). *A pair of PPT algorithms $\mathcal{U} = (\mathsf{Sign}, \mathsf{Verify})$ is a scheme for authenticating messages as a winner of a lottery in the past in the context of blockchain Γ with lottery predicate* $\mathsf{lottery}$.

Authenticate. $\sigma \leftarrow \mathsf{AfP.Sign}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}, m)$ *takes as input a blockchain* $\mathbf{B}$*, a slot* sl*, a role* R*, a secret key* sk*, and a message* m*. It outputs a signature* σ *that authenticates the message* m*.*

Verify. $\{0,1\} \leftarrow \mathsf{AfP.Verify}(\tilde{\mathbf{B}}, \mathsf{sl}, \mathsf{R}, \sigma, m)$ *uses the blockchain* $\tilde{\mathbf{B}}$ *to ensure that* σ *is a signature on* m *produced by the secret key winning the lottery for slot* sl *and role* R*.*

Furthermore, an AfP-scheme has the following properties:

Correctness. *An AfP-scheme is said to be correct if for honest parties i and j, there exists a negligible function μ such that for all $\mathsf{sk}, \mathsf{sl}, \mathsf{R}, m$:*

$$\left| \Pr \left[\begin{array}{c} \mathsf{view} \leftarrow \mathit{EXEC}^{\Gamma}(\mathcal{A}, \mathcal{Z}, 1^{\lambda}) \\ \mathbf{B} = \mathsf{GetRecords}(\mathsf{view}_i) \\ \tilde{\mathbf{B}} = \mathsf{GetRecords}(\mathsf{view}_j) \\ \sigma \leftarrow \mathsf{AfP.Sign}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}, m) \end{array} : \begin{array}{c} \mathsf{lottery}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}) = 0 \\ \vee\ \mathsf{lottery}(\tilde{\mathbf{B}}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}) = 0 \\ \vee\ \mathsf{AfP.Verify}(\tilde{\mathbf{B}}, \mathsf{sl}, \mathsf{R}, \sigma, m) = 1 \end{array} \right] - 1 \right| \leq \mu(\lambda)$$

In other words, an AfP on a message from an honest party with a view of the blockchain $\mathbf{B}$ *can attest to the fact that the sender won the role* R *in slot* sl*. If another party, with blockchain* $\tilde{\mathbf{B}}$ *agrees, then the verification algorithm will output 1.*

Security. *We here describe the game detailed in Algorithm 3 representing the security of an AfP scheme. The algorithm represents a standard EUF-CMA game where the adversary has access to a signing oracle $\mathcal{O}_{\mathsf{AfP}}$ which it can query with a slot sl, a role R and a message m_i and obtain AfP signatures $\sigma_i = \mathsf{AfP.Sign}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}_j, m_i)$ where $\mathsf{sk}_j \in \mathcal{W}_{\mathbf{B},\mathsf{sl},\mathsf{R}}$ i.e. $\mathsf{lottery}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}_j) = 1$. The oracle maintains the list of queries $\mathcal{Q}_{\mathsf{AfP}}$.*
Formally, an AfP-scheme $\mathcal{U}$ is said to be EUF-CMA secure in the context of a blockchain protocol Γ executed by PPT machines $\mathcal{A}$ and $\mathcal{Z}$ if there exists a negligible function μ such that for $\lambda \in \mathbb{N}$:

$$\Pr\left[\mathrm{Game}^{\mathsf{EUF\text{-}CMA}}_{\Gamma,\mathcal{A},\mathcal{Z},\mathcal{U}} = 1\right] \leq \mu(\lambda)$$

Algorithm 3 $\mathrm{Game}^{\mathsf{EUF\text{-}CMA}}_{\Gamma,\mathcal{A},\mathcal{Z},\mathcal{U}}$

$\mathsf{view} \leftarrow \mathsf{EXEC}^{\Gamma}(\mathcal{A}, \mathcal{Z}, 1^{\lambda})$ ▷ $\mathcal{A}$ executes Γ with $\mathcal{Z}$
$(\mathbf{B}, \mathsf{sl}, \mathsf{R}, m', \sigma') \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{AfP}}}(\mathsf{view}_{\mathcal{A}})$
if $(m' \in \mathcal{Q}_{\mathsf{AfP}}) \vee (\mathsf{sk}^{\mathcal{A}}_{L,j} \in \mathcal{W}_{\mathbf{B},\mathsf{sl},\mathsf{R}})$ **then** ▷ $\mathcal{A}^{\mathcal{O}_{\mathsf{AfP}}}$ won or queried illegal m'
  return 0
end if
$\mathsf{view}^{\tilde{r}} \leftarrow \mathsf{EXEC}^{\Gamma}_{(\mathsf{view}^{r},\tilde{r})}(\mathcal{A}, \mathcal{Z}, 1^{\lambda})$ ▷ Execute from view^{r} until round $\tilde{r}$
$\tilde{\mathbf{B}} \leftarrow \mathsf{GetRecords}(\mathsf{view}^{\tilde{r}}_{i})$
if $\mathsf{evolved}(\mathbf{B}, \tilde{\mathbf{B}}) = 1$ **then**
  if $\mathsf{AfP.Verify}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \sigma', m') = 1$ **then** ▷ $\mathcal{A}$ successfully forged an AfP
    return 1
  end if
end if
return 0

General AfP. In general we can add authentication to a message as follows. Recall that P_i wins R if $\mathsf{lottery}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}_{L,i}) = 1$. Here, $\mathcal{R}(\mathtt{x} = (\mathbf{B}, \mathsf{sl}, \mathsf{R}), \mathtt{w}) = \mathsf{lottery}(\mathtt{x}, \mathtt{w})$ is an NP relation where all parties know $\mathtt{x}$ but only the winner knows a witness $\mathtt{w}$ such that $\mathcal{R}(\mathtt{x}, \mathtt{w}) = 1$. We can therefore use a signature of knowledge (SoK) [8] to sign m under the knowledge of $\mathsf{sk}_{L,i}$ such that $\mathsf{lottery}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}_{L,i}) = 1$. This will attest that the message m was sent by a winner of the lottery for R. In [7], we show more efficient construction of AfP by exploring the structure of PoS-based blockchains with VRF lotteries.

6.3 AfP Privacy

Just EUF-CMA security is not sufficient for an AfP mechanism to be YOSO friendly. It must also preserve the privacy guarantees of the lottery predicate, guaranteeing that the adversary does not gain any undue advantage in predicting when a party is selected to perform a role after it uses AfP to authenticate a

message. To appreciate this fact, we consider the case where instead of creating a signature of knowledge of $\mathsf{sk}_{L,i}$ on message m we simply use a regular EUF-CMA secure signature scheme to sign the message concatenated with $\mathsf{sk}_{L,i}$, revealing the signature public key, the resulting signature and $\mathsf{sk}_{L,i}$ itself as a means of authentication. By definition, this will still constitute an existentially unforgeable AfP but will also reveal whether the party who owns $\mathsf{sk}_{L,i}$ is the winner when future lotteries are conducted. The specific privacy property we seek is that an adversary, observing AfP tags from honest parties, cannot use this information to enhance its chances in predicting the winners of lotteries for roles for which an AfP tag has not been published. On the other hand, the identity of a party who won the lottery for a given role is not kept private when it publishes an AfP tag on behalf of this role, which is not an issue in a YOSO-setting since corruption after-the-fact is futile. Specifically, we allow an AfP tag to be linked to the identity of the party who generated it. Note, that this kind of privacy is different from notions like k-anonymity since the success of the adversary in guessing lottery winners with high accuracy depends on the stake distribution. The stake distribution is public in most PoS-settings and, thus, a privacy definition must take into account this inherent leakage.

Definition 8 (AfP Privacy.). *An AfP scheme $\mathcal{U}$ with corresponding lottery predicate* $\mathsf{lottery}$ *is private if a PPT adversary $\mathcal{A}$ is unable to distinguish between the scenarios defined in Algorithm 4 and Algorithm 5 with more than negligible probability in the security parameter.*

Scenario 0 ($b = 0$)**.** *In this scenario (Algorithm 4), $\mathcal{A}$ is first running the blockchain Γ together with the environment $\mathcal{Z}$. At round r, $\mathcal{A}$ is allowed to interact with the oracle $\mathcal{O}_{\mathsf{AfP}}$ (see Definition 7). The adversary then continues the execution until round $\tilde{r}$ where it outputs a bit b'.*

Scenario 1 ($b = 1$)**.** *This scenario (Algorithm 5) is identical to scenario 0 but instead of interacting with $\mathcal{O}_{\mathsf{AfP}}$, the adversary interacts with a simulator* Sim.

Algorithm 4 $b = 0$

$\mathsf{view}^r \leftarrow \mathsf{EXEC}^{\Gamma}_r(\mathcal{A}, \mathcal{Z}, 1^\lambda)$
$\mathcal{A}^{\mathcal{O}_{\mathsf{AfP}}}(\mathsf{view}^r_{\mathcal{A}})$
$\mathsf{view}^{\tilde{r}} \leftarrow \mathsf{EXEC}^{\Gamma}_{(\mathsf{view}^r, \tilde{r})}(\mathcal{A}, \mathcal{Z}, 1^\lambda)$
return $b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{AfP}}}(\mathsf{view}^{\tilde{r}}_{\mathcal{A}})$

Algorithm 5 $b = 1$

$\mathsf{view}^r \leftarrow \mathsf{EXEC}^{\Gamma}_r(\mathcal{A}, \mathcal{Z}, 1^\lambda)$
$\mathcal{A}^{\mathsf{Sim}}(\mathsf{view}^r_{\mathcal{A}})$
$\mathsf{view}^{\tilde{r}} \leftarrow \mathsf{EXEC}^{\Gamma}_{(\mathsf{view}^r, \tilde{r})}(\mathcal{A}, \mathcal{Z}, 1^\lambda)$
return $b' \leftarrow \mathcal{A}^{\mathsf{Sim}}(\mathsf{view}^{\tilde{r}}_{\mathcal{A}})$

We let $\mathrm{Game}^{\mathsf{AfP\text{-}PRIV}}_{\Gamma,\mathcal{A},\mathcal{Z},\mathcal{U}}$ *denote the game where a coin-flip decides whether the adversary is executed in scenario 0 or scenario 1. We say that the adversary wins the game (i.e.* $\mathrm{Game}^{\mathsf{AfP\text{-}PRIV}}_{\Gamma,\mathcal{A},\mathcal{Z},\mathcal{U}} = 1$*) iff $b' = b$. Finally, an AfP scheme $\mathcal{U}$ is called private in the context of the blockchain Γ executed together with environment $\mathcal{Z}$ if the following holds for a negligible function μ.*

$$\left| 2 \cdot \Pr\left[\mathrm{Game}^{\mathsf{AfP\text{-}PRIV}}_{\Gamma,\mathcal{A},\mathcal{Z},\mathcal{U}} = 1 \right] - 1 \right| \leq \mu(\lambda)$$

6.4 Round and Committee Based YOSO Protocols

Having IND-CCA2 ECW and an EUF-CMA secure and Private AfP, we can establish a round-based YOSO model, where there is a number of rounds $r = 1, 2, \ldots$ and where for each round there are n roles $\mathsf{R}_{r,i}$. We call the role $\mathsf{R}_{r,i}$ "party i in round r". We fix a round length L and associate role $\mathsf{R}_{r,i}$ to slot $\mathsf{sl} = L \cdot r$. This L has to be long enough that in each round the parties executing the roles can decrypt ciphertexts sent to them, execute the steps of the role, compute encryptions to the roles in the next round and post these to the blockchain in time for these to be available to the committee of round $r+1$ before slot $(r+1)\cdot L$. Picking such an L depends crucially on the underlying blockchain and network, and we will here simply assume that it can be done for the blockchain at hand.

Using this setup, the roles $\mathsf{R}_{r,i}$ of round r can use ECW and AfP with the aforementioned properties to send secret authenticated messages to the roles $\mathsf{R}_{r+1,i}$ in round $r+1$. They find their ciphertexts on the blockchain before slot $r \cdot L$, decrypt using ECW, compute their outgoing messages, encrypt using ECW, authenticate using AfP, and post the ciphertexts and AfP tags on the blockchain.

Honest Majority. In round based YOSO MPC it is critical that we can assume some fraction of honesty in each committee $\mathsf{R}_{r,1}, \ldots, \mathsf{R}_{r,n}$. We discuss here assumptions needed on the lottery for this to hold and how to guarantee it. Assume an adversary that can corrupt parties identified by sk and a lottery assigning parties to roles $\mathsf{R}_{r,i}$. We map the corruption status of parties to roles as follows:

1. If a role $\mathsf{R}_{r,i}$ is won by a corrupted party or by several parties, call the role Malicious. Even if $\mathsf{R}_{r,i}$ is won by two honest parties, they will both execute the role and send outgoing messages, which might violate security.
2. If a role $\mathsf{R}_{r,i}$ is won by exactly one honest party, call it Honest.
3. If a role $\mathsf{R}_{r,i}$ is not won by any party, call it Crashed. These roles will not be executed and are therefore equivalent to a crashed party.

Note that because we assume corrupted parties know their lottery witness $\mathsf{sk}_{L,i}$ in our model, we can, in poly-time, extract those witnesses and compute the corruption status of roles. This will be crucial in our reductions. Imagine that a role could be won by an honest party but also by a corrupted party which stays completely silent but decrypts messages sent to the role. If we are not aware of the corrupted party winning the role, we might send a simulated ciphertext to the apparently honest role. The corrupted party also having won the role would be able to detect this. Since any role won by an honest party could also be corrupted by a silent malicious party, simulation would become impossible.

In order to realize YOSO MPC, we will need committees where a majority of the roles are honest according to the description above. We capture this requirement in the definition below and argue how it can be achieved in the full version [7].

Definition 9 (Honest Committee Friendly). *We call a blockchain Γ honest committee friendly if there exist n and H and T such that $H > T$ s.t. we can define a sequence of roles $\mathsf{R}_{r,i}$ for $r = 1, \ldots, \mathsf{poly}(\lambda)$ and $i = 1, \ldots, n$ for a slot sl_r and that for all r it holds that except with negligible probability there are at least H honest roles in $\mathsf{R}_{r,1}, \ldots, \mathsf{R}_{r,n}$ and at most T malicious roles. Furthermore, if an honest party executing $\mathsf{R}_{r,1}, \ldots, \mathsf{R}_{r,n}$ sends a message at sl_r, it is guaranteed to appear on the blockchain before slot sl_{r+1}.*

We are now ready to capture the above discussion using a definition.

Definition 10 (YOSO Friendly Blockchain). *Let Γ be a blockchain with a lottery predicate $\mathsf{lottery}(\mathbf{B}, \mathsf{sl}_r, \mathsf{R}_{r,i}, \mathsf{sk}_{L,i})$ and let $\mathcal{E} = (\mathsf{Enc}, \mathsf{Dec})$ and $\mathcal{U} = (\mathsf{Sign}, \mathsf{Verify})$ be an EtF and AfP for $\mathsf{lottery}(\mathbf{B}, \mathsf{sl}_r, \mathsf{R}_{r,i}, \mathsf{sk}_{L,i})$, respectively. We call $(\Gamma, \mathcal{E}, \mathcal{U})$ YOSO MPC friendly if the following holds:*

1. *$\mathcal{E}$ is an IND-CCA2 secure EtF (Definition 6).*
2. *$\mathcal{U}$ is a secure and private AfP (Definition 7 and Definition 8).*
3. *Γ is honest committee friendly (Definition 9).*

We will later assume a YOSO friendly blockchain, and we argued above that the existence of a YOSO friendly blockchain is a plausible assumption without having given formal proofs of this. It is interesting future work to prove that a concrete blockchain is a YOSO friendly blockchain in a given communication model. We omit this as our focus is on constructing flavours of EtF.

7 Construction of EtF from ECW and Threshold-IBE

The key intuition about our construction is as follows: we use IBE to encrypt messages to an arbitrary future $(\mathsf{R}, \mathsf{sl}_{\text{fut}})$ pair. When the winners of the role in slot sl_{fut} are assigned, we let them obtain an ID-specific key for $(\mathsf{R}, \mathsf{sl}_{\text{fut}})$ from the IBE key-generation algorithm using ECW as a channel. Notice that this key-generation happens in the *present* while the encryption could have happened at any earlier time. We generate the key for $(\mathsf{R}, \mathsf{sl}_{\text{fut}})$ in a threshold manner by assuming that, throughout the blockchain execution, a set of committee members each holds a share of the master secret key msk_i.

7.1 Construction

We now describe our construction. We assume an encryption to the current winner $\Pi_{\mathsf{ECW}} = (\mathsf{Enc}_{\mathsf{ECW}}, \mathsf{Dec}_{\mathsf{ECW}})$ and a threshold IBE scheme Π_{TIBE}. In the setup stage we assume a dealer acting honestly by which we can assign master secret key shares of the TIBE.

Parameters: We assume that the genesis block B_0 of the underlying blockchain contains all the parameters for Π_{ECW}.

Setup Phase: Parties run the setup stage for the Π_{ECW}. The dealer produces $(\mathsf{mpk}, \vec{\mathsf{msk}} = (\mathsf{msk}_1, \ldots, \mathsf{msk}_n))$ from TIBE setup with threshold k. Then it chooses n random parties and gives a distinct msk_i to each. All learn mpk.

Blockchain Execution: The blockchain execution we assume is as in Section 3. We additionally require that party i holding a master secret key share msk_i broadcasts $\mathsf{ct}^{\mathsf{sk},i}_{(\mathsf{sl},\mathsf{R})} \leftarrow \mathsf{Enc}_{\mathsf{ECW}}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}^i_{(\mathsf{sl},\mathsf{R})})$, whenever the winner of role R in slot sl is defined in the blockchain $\mathbf{B}$, where $\mathsf{sk}^i_{(\mathsf{sl},\mathsf{R})} \leftarrow \Pi_{\mathsf{TIBE}}.\mathsf{IDKeygen}(\mathsf{msk}_i, (\mathsf{sl}, \mathsf{R}))$.

Encryption $\mathsf{Enc}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, m)$**:** Each party generates $\mathsf{ct}_i \leftarrow \Pi_{\mathsf{TIBE}}.\mathsf{Enc}(\mathsf{mpk}, \mathsf{ID} = (\mathsf{sl}, \mathsf{R}), m)$. Output $\mathsf{ct} = (\mathbf{B}, \mathsf{sl}, \mathsf{R}, \{\mathsf{ct}_i\}_{P_i})$.

Decryption $\mathsf{Dec}(\mathbf{B}, \mathsf{ct}, \mathsf{sk})$**:** Party i outputs $\perp$ if it does not have $\mathsf{sk}_{L,i}$ such that $\mathsf{lottery}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}_{L,i}) = 1$ for parameters $\mathbf{B}, \mathsf{sl}, \mathsf{R}$ from ct. Otherwise, it retrieves enough (above threshold) valid ciphertexts $\mathsf{ct}^{\mathsf{sk},j}_{(\mathsf{sl},\mathsf{R})}$ from the current state of the blockchain and decrypts each through Π_{ECW} obtaining $\mathsf{sk}^j_{(\mathsf{sl},\mathsf{R})}$. It then computes $\mathsf{sk}_{(\mathsf{sl},\mathsf{R})} \leftarrow \Pi_{\mathsf{TIBE}}.\mathsf{Combine}(\mathsf{mpk}, (\mathsf{sk}^j_{(\mathsf{sl},\mathsf{R})})_j)$. It finally outputs $m \leftarrow \Pi_{\mathsf{TIBE}}.\mathsf{Dec}(\mathsf{sk}_{(\mathsf{sl},\mathsf{R})}, \mathsf{ct})$.

Resharing. We can ensure that the master secret key is proactively reshared by modifying each party so that msk_i-s are reshared and reconstructed in the evolution of the blockchain.

Correctness. Correctness of the construction follows from the correctness of the underlying IBE and the fact that a winning role will be able to decrypt the id-specific key by the correctness of the ECW scheme.

In the following we assume some of the extensions discussed in Section 6.

Theorem 2 (informal). *Let Γ^V be a YOSO MPC friendly blockchain, Π_{TIBE} be a robust secure threshold IBE as in Section 2.3 with threshold $n/2$, and Π_{ECW} a secure* IND-CCA2 *ECW. The construction in Section 7.1 is a secure EtF.*

We refer the reader to the full version for a proof of security [7].

8 Blockchain WE versus EtF

In this section we show that an account-based PoS blockchain with sufficiently expressive smart contracts and an EtF scheme for this blockchain implies a notion of witness encryption on blockchains, and *vice versa*. The construction of EtF from BWE is completely straightforward and natural: encrypt to the witness which is the secret key winning the lottery. The construction of BWE from EtF is also straightforward but slightly contrived: it requires that we can restrict the lottery such that only some accounts can win a given role and that the decryptor has access to a constant fraction of the stake on the blockchain and are willing to bind them for the decryption operation. The reason why we still prove the result is that it establishes a connection at the feasibility level. For sufficiently expressive blockchains the techniques allowing to construct EtF and BWE are the same. To get EtF from simpler techniques than those we need for BWE we need to do it in the context of very simple blockchains. In addition, the techniques allowing to get EtF without getting BWE should be such that they prevent the

blockchain from having an expressive smart contract layer added. This seems like a very small loophole, so we believe that the result shows that there is essentially no assumptions or techniques which allow to construct EtF which do not also allow to construct BWE. Since BWE superficially looks stronger than EtF the equivalence helps better justify the strong assumptions for constructing EtF.

Definition 11 (Blockchain Witness Encryption). *Consider PPT algorithms* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *in the context of a blockchain* Γ^V *is an BWE-scheme with evolved predicate* $\mathsf{evolved}$ *and a lottery predicate* $\mathsf{lottery}$ *working as follows:*

Setup. $(\mathsf{pv}, \mathsf{td}) \leftarrow \mathsf{Gen}()$ *generates a public value* pv *and an extraction trapdoor* td*. Initially* pv *is put on* $\mathbf{B}$*.*

Encryption. $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathbf{B}, W, m)$ *takes as input a blockchain* $\mathbf{B}$*, including the public value, a PPT function* W*, the witness recogniser, and a message* m*. It outputs a ciphertext* ct*, a blockchain witness encryption.*

Decryption. $m/\perp \leftarrow \mathsf{Dec}(\tilde{\mathbf{B}}, \mathsf{ct}, \mathtt{w})$ *in input a blockchain state* $\tilde{\mathbf{B}}$*, including the a public value* pv*, a ciphertext* ct *a witness* $\mathtt{w}$*, it outputs a message* m *or* $\perp$*.*

Correctness. An BWE-scheme is correct if for honest parties i and j, PPT function W, and witness $\mathtt{w}$ such that $W(\mathtt{w}) = 1$ the following holds with overwhelming probability: if party i runs $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathbf{B}, W, m)$ and party j starts running $\mathsf{Dec}(\tilde{\mathbf{B}}, \mathsf{ct}, \mathtt{w})$ in $\tilde{\mathbf{B}}$ evolved from $\mathbf{B}$, then eventually $\mathsf{Dec}(\tilde{\mathbf{B}}, \mathsf{ct}, \mathtt{w})$ outputs m.

Security. We establish a game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$. In section 2.1 we described how $\mathcal{A}$ and $\mathcal{Z}$ execute a blockchain protocol. In addition, we now let the adversary interact with the challenger in a game $\mathrm{Game}^{\mathsf{IND\text{-}CPA}}_{\Gamma,\mathcal{A},\mathcal{Z},\mathcal{E}}$ which can be summarized as follows.

1. $(\mathsf{pv}, \mathsf{td}) \leftarrow \mathsf{Gen}()$ and put pv on the blockchain.
2. $\mathcal{A}$ executes the blockchain protocol Γ together with $\mathcal{Z}$ and at some round r chooses a blockchain $\mathbf{B}$, a function W and two messages m_0 and m_1 and sends it all to $\mathcal{C}$.
3. $\mathcal{C}$ chooses a random bit b and encrypts the message m_b with the parameters it received and sends ct to $\mathcal{A}$.
4. $\mathcal{A}$ continues to execute the blockchain until some round $\tilde{r}$ where the blockchain $\tilde{\mathbf{B}}$ is obtained and the $\mathcal{A}$ outputs a bit b'.

The adversary wins the game if it succeeds in guessing b with probability notably greater than one half without $W(\mathsf{Extract}(\mathsf{td}, \tilde{\mathbf{B}}, \mathsf{ct}, W)) = 1$.

EtF from BWE. We first show the trivial direction of getting EtF from BWE. Let $\Pi_{\mathsf{BWE}} = (\mathsf{Gen}_{\mathsf{BWE}}, \mathsf{Enc}_{\mathsf{BWE}}, \mathsf{Dec}_{\mathsf{BWE}})$ be an BWE scheme. Recall that one wins the lottery if $\mathsf{lottery}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}) = 1$. We construct a EtF scheme. To encrypt, let W be the function $W(\mathtt{w}) = \mathsf{lottery}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathtt{w})$ and output $\mathsf{Enc}_{\mathsf{BWE}}(\mathbf{B}, W, m)$. If winning the lottery for $(\mathsf{sl}, \mathsf{R})$ then let $\mathtt{w}$ be the secret key winning the lottery and output $\mathsf{Dec}(\tilde{\mathbf{B}}, \mathsf{ct}, \mathtt{w})$. The proof is straightforward.

BWE from EtF. We describe a proof of this direction in the full version [7].

Acknowledgements Bernardo David is supported by the Concordium Foundation and by the Independent Research Fund Denmark (IRFD) grants number 9040-00399B (TrA^{2}C), 9131-00075B (PUMA) and 0165-00079B. Hamidreza Khoshakhlagh has been funded by the Concordium Foundation under Concordium Blockchain Research Center, Aarhus. Anders Konring is supported by the IRFD grant number 9040-00399B (TrA^{2}C). Jesper Buus Nielsen is partially funded by the Concordium Foundation; The Danish Independent Research Council under Grant-ID DFF-8021-00366B (BETHE); The Carlsberg Foundation under the Semper Ardens Research Project CF18-112 (BCM).

References

1. A. Afshar, P. Mohassel, B. Pinkas, and B. Riva. Non-interactive secure computation based on cut-and-choose. In P. Q. Nguyen and E. Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 387–404. Springer, Heidelberg, May 2014.
2. O. Barta, Y. Ishai, R. Ostrovsky, and D. J. Wu. On succinct arguments and witness encryption from groups. In D. Micciancio and T. Ristenpart, editors, *CRYPTO 2020, Part I*, volume 12170 of *LNCS*, pages 776–806. Springer, Heidelberg, Aug. 2020.
3. F. Benhamouda, C. Gentry, S. Gorbunov, S. Halevi, H. Krawczyk, C. Lin, T. Rabin, and L. Reyzin. Can a public blockchain keep a secret? In R. Pass and K. Pietrzak, editors, *TCC 2020, Part I*, volume 12550 of *LNCS*, pages 260–290. Springer, Heidelberg, Nov. 2020.
4. F. Benhamouda and H. Lin. Mr NISC: Multiparty reusable non-interactive secure computation. In R. Pass and K. Pietrzak, editors, *TCC 2020, Part II*, volume 12551 of *LNCS*, pages 349–378. Springer, Heidelberg, Nov. 2020.
5. D. Boneh, X. Boyen, and S. Halevi. Chosen ciphertext secure public key threshold encryption without random oracles. In D. Pointcheval, editor, *CT-RSA 2006*, volume 3860 of *LNCS*, pages 226–243. Springer, Heidelberg, Feb. 2006.
6. E. Boyle, N. Gilboa, Y. Ishai, H. Lin, and S. Tessaro. Foundations of homomorphic secret sharing. In A. R. Karlin, editor, *ITCS 2018*, volume 94, pages 21:1–21:21. LIPIcs, Jan. 2018.
7. M. Campanelli, B. David, H. Khoshakhlagh, A. Konring, and J. B. Nielsen. Encryption to the future: A paradigm for sending secret messages to future (anonymous) committees. Cryptology ePrint Archive, Paper 2021/1423, 2021. https://eprint.iacr.org/2021/1423.
8. M. Chase and A. Lysyanskaya. On signatures of knowledge. In C. Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 78–96. Springer, Heidelberg, Aug. 2006.
9. B. David, P. Gazi, A. Kiayias, and A. Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In J. B. Nielsen and V. Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 66–98. Springer, Heidelberg, Apr. / May 2018.
10. D. Derler and D. Slamanig. Practical witness encryption for algebraic languages and how to reply an unknown whistleblower. Cryptology ePrint Archive, Report 2015/1073, 2015. https://eprint.iacr.org/2015/1073.

11. E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In M. J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 537–554. Springer, Heidelberg, Aug. 1999.
12. J. A. Garay, R. Gelles, D. S. Johnson, A. Kiayias, and M. Yung. A little honesty goes a long way - the two-tier model for secure multiparty computation. In Y. Dodis and J. B. Nielsen, editors, *TCC 2015, Part I*, volume 9014 of *LNCS*, pages 134–158. Springer, Heidelberg, Mar. 2015.
13. J. A. Garay, A. Kiayias, and N. Leonardos. The bitcoin backbone protocol: Analysis and applications. In E. Oswald and M. Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 281–310. Springer, Heidelberg, Apr. 2015.
14. S. Garg, C. Gentry, A. Sahai, and B. Waters. Witness encryption and its applications. In D. Boneh, T. Roughgarden, and J. Feigenbaum, editors, *45th ACM STOC*, pages 467–476. ACM Press, June 2013.
15. C. Gentry, S. Halevi, H. Krawczyk, B. Magri, J. B. Nielsen, T. Rabin, and S. Yakoubov. YOSO: You only speak once - secure MPC with stateless ephemeral roles. In T. Malkin and C. Peikert, editors, *CRYPTO 2021, Part II*, volume 12826 of *LNCS*, pages 64–93, Virtual Event, Aug. 2021. Springer, Heidelberg.
16. C. Gentry, S. Halevi, B. Magri, J. B. Nielsen, and S. Yakoubov. Random-index pir and applications. In *Theory of Cryptography Conference*, pages 32–61. Springer, 2021.
17. R. Goyal and V. Goyal. Overcoming cryptographic impossibility results using blockchains. In Y. Kalai and L. Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 529–561. Springer, Heidelberg, Nov. 2017.
18. V. Goyal, A. Kothapalli, E. Masserova, B. Parno, and Y. Song. Storing and retrieving secrets on a blockchain. Cryptology ePrint Archive, Report 2020/504, 2020. https://eprint.iacr.org/2020/504.
19. M. Jawurek, F. Kerschbaum, and C. Orlandi. Zero-knowledge using garbled circuits: how to prove non-algebraic statements efficiently. In A.-R. Sadeghi, V. D. Gligor, and M. Yung, editors, *ACM CCS 2013*, pages 955–966. ACM Press, Nov. 2013.
20. J. B. Nielsen. *On protocol security in the cryptographic model*. Citeseer, 2003.
21. R. Pass, L. Seeman, and a. shelat. Analysis of the blockchain protocol in asynchronous networks. In J.-S. Coron and J. B. Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 643–673. Springer, Heidelberg, Apr. / May 2017.
22. A. Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th FOCS*, pages 543–553. IEEE Computer Society Press, Oct. 1999.
23. S. Zahur, M. Rosulek, and D. Evans. Two halves make a whole - reducing data transfer in garbled circuits using half gates. In E. Oswald and M. Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 220–250. Springer, Heidelberg, Apr. 2015.