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.

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

Table of Contents

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 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 [BGG+20,GHK+21,GHM+21] 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 [BGG+20] and further developed in [GHK+21]. 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 [BGG+20,GHK+21,GHM+21] for realizing role assignment all use an underlying Proof-of-Stake (PoS) blockchain (*e.g.* [DGKR18]). 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 [BGG+20,GHK+21,GHM+21] 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 [BGG+20], the auxiliary

committee performs cheap operations but can adversarially influence the probability distribution with which P_i is chosen. In the case of [GHK+21,GHM+21], 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

[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 [GHK+21] 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 [GHK+21] 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, [GHK+21] 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.

"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).
- 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 [BGG+20,GHK+21,GHM+21] 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.* [BGG+20,GHK+21,GHM+21]).

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 [BGG+20,GHK+21,GHM+21], 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.* [DGKR18]) 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

[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.

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).

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 [GGSW13] over a blockchain (similar to that of [GKM+20] 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., [GKM+20]) 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 [BGG+20,GHK+21,GHM+21]). 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 (Section 4.2). This construction improves over the previous results [BGG+20,GHK+21,GHM+21] 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 [GHK+21]. 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.* [DGKR18]). 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 ([BGG+20,GHK+21,GHM+21]) successive committees were required to store and reshare secret shares of every

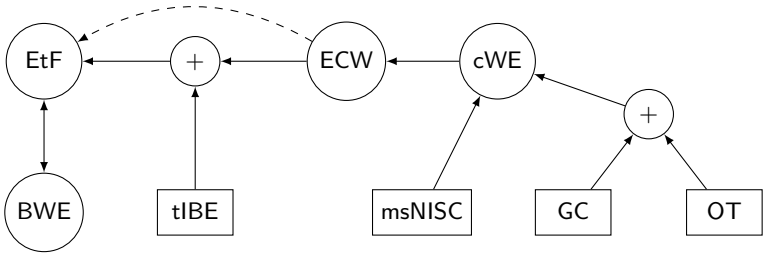

Fig. 1. Dependency diagram for primitives in this work. Legend: primitives wrapped in circles are introduced in this work; $A \to B$: *"We can construct B from A"*; $A \dashrightarrow B$: *"A is a special case of B"*.

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.

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.

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) [BGG+20]. The work of [BGG+20] 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 [BGG+20] is to select the "secret-holding" committee by having another committee, a "nominating committee", that

Type	Scheme	Communication	Committee?	Interaction?
ECW	CaBKaS [BGG+20]	$O(1)$	yes	yes
	RPIR [GHM+21]	$O(1)$	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 [GKM+20]	$O(M)$	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 [GKM+20]) 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.

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, [BGG+20] 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) [GHM+21]. The recent work of [GHM+21] 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 [BGG+20] (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 [BL20] 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 [BL20], formalize it as cWE and finally use it to construct ECW. While the original construction of [BL20] 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 [GGSW13]: One trivial approach to realize EtF is to use full-fledged general Witness Encryption [GGSW13] (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 [GGSW13] 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 [BIOW20]. 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 [BGG+20] and [GHM+21]). This is essentially the solution proposed in CaBKaS [BGG+20] 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 [GG17]. Other approaches for transferring secret state to future committees have been proposed in [GKM+20], although anonymity is not a concern in this setting. On the other hand, using anonymity to overcome adaptive corruption has been proposed in [GGJ+15], 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 [GG17] which, in turn, builds on the analysis done in [GKL15] and [PSs17]. We invite the reader to revisit the abstraction used in [GG17]. We present a summary of the framework in Appendix A.1 and discuss below the main properties we will use in the remainder of this paper. Moreover, we note that in [GG17] it is proven that there exist PoS blockchain protocols with the properties described below, *e.g.* Ouroboros Praos [DGKR18].

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 [DGKR18], 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 [DGKR18] 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 Appendix A.1 and [GG17] 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 Oblivious Transfer

A 2-round oblivious transfer (OT) protocol between a receiver R and a sender S consists of three polynomial-time algorithms $\Pi_{\mathsf{OT}} = (\Pi^R_{\mathsf{OT}}, \Pi^S_{\mathsf{OT}}, \Pi^O_{\mathsf{OT}})$:

Choose. On input $(\texttt{receive}, \mathsf{sid}, b)$ from R , where $b \in \{0,1\}$, if no messages of the form $(\texttt{receive}, \mathsf{sid}, b)$ is stored, store $(\texttt{receive}, \mathsf{sid}, b)$ and send $(\texttt{receive}, \mathsf{sid})$ to S.

Transfer. On input $(\texttt{send}, \mathsf{sid}, x^0, x^1)$ from S, with $x^0, x^1 \in \{0,1\}^k$, if no messages of the form $(\texttt{send}, \mathsf{sid}, x^0, x^1)$ is stored and a message of the form $(\texttt{receive}, \mathsf{sid}, b)$ is present, send $(\texttt{sent}, \mathsf{sid}, x^b)$ to R.

Fig. 3. The ideal functionality $\mathcal{F}_{\mathsf{OT}}$ for oblivious transfer

$m^R \leftarrow \Pi^R_{\mathsf{OT}}(b; r^R)$**.** In the first round, the receiver R on input $b \in \{0,1\}$ and random tape $r^R \in \{0,1\}^{\mathrm{poly}(\lambda)}$ generates the OT first message m^R.

$m^S \leftarrow \Pi^S_{\mathsf{OT}}(m^R, (x^0, x^1); r^S)$**.** In the second round, the sender S on input (x^0, x^1), where $x^l \in \{0,1\}^{\mathrm{poly}(\lambda)}$ for $l \in \{0,1\}$, generates the second message m^S using random tape $r^S \in \{0,1\}^{\mathrm{poly}(\lambda)}$.

$x \leftarrow \Pi^O_{\mathsf{OT}}(m^S, b, r^R)$**.** R computes the output $x = \Pi^O_{\mathsf{OT}}(m^S, b, r^R)$.

We require an OT protocol to securely implement the ideal functionality $\mathcal{F}_{\mathsf{OT}}$ given in Fig. 3 in the presence of malicious adversaries.

2.4 Garbled Circuit

We recall the definition of *garbling schemes* formalized by Bellare et al. in [BHR12].

Definition 3 (Garbling Scheme). *Let $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ be a polynomial-size circuit class. A garbled circuit scheme* GC *for $\mathcal{C}$ consists of four polynomial-time algorithms* $\mathsf{GC} = (\mathsf{Garble}, \mathsf{Encode}, \mathsf{Eval}, \mathsf{Decode})$*:*

$(\mathbf{C}, e, d) \leftarrow \mathsf{Garble}(1^\lambda, C)$**:** *On input a boolean circuit $C \in \mathcal{C}_\lambda$, outputs $(\mathbf{C}, e, d)$, where $\mathbf{C}$ is a garbled circuit, e is encoding information, and d is decoding information.*

$X \leftarrow \mathsf{Encode}(e, x)$**:** *On input e and x, where x is a suitable input for C, outputs a garbled input X.*

$Y = \mathsf{Eval}(\mathbf{C}, X)$**:** *On input $(\mathbf{C}, X)$ as above, outputs a garbled output Y.*

$y \leftarrow \mathsf{Decode}(d, Y)$**:** *On input (d, Y) as above, outputs a plain output y.*

For our construction, we are interested in garbling schemes with the following properties.

Correctness. For any security parameter $\lambda \in \mathbb{N}$, for any circuit $C \in \mathcal{C}_\lambda$, for $(\mathbf{C}, e, d) \leftarrow \mathsf{Garble}(1^\lambda, C)$, and for all suitable input x:

$$\mathsf{Decode}(d, \mathsf{Eval}(\mathbf{C}, \mathsf{Encode}(e, x))) = C(x)$$

Authenticity. For all circuits $C : \{0,1\}^n \to \{0,1\}$, inputs $x \in \{0,1\}^n$, where $n = \text{poly}(\lambda)$, and for all PPT adversaries $\mathcal{A}$,

$$\Pr\left[\begin{matrix}\hat{Y} \neq \mathsf{Eval}(\mathbf{C}, X) \wedge \\ \mathsf{Decode}(d, \hat{Y}) \neq \perp\end{matrix} : \begin{matrix}(\mathbf{C}, e, d) \leftarrow \mathsf{Garble}(1^\lambda, C) \\ X = \mathsf{Encode}(e, x);\ \hat{Y} \leftarrow \mathcal{A}(C, x, \mathbf{C}, X)\end{matrix}\right] \approx_\lambda 0$$

2.5 (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 [Nie03]).

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 $\perp$.

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

$\Pi_{\mathsf{TIBE}}.\mathsf{Dec}(\mathsf{ID}, \mathsf{sk}_{\mathsf{ID}}, \mathsf{ct}) \to m$: 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 [BGI+18,BBH06,Nie03]. 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 defer the reader to Appendix A.2 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 4 (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.

Definition 5 (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* $\lambda \in \mathbb{N}$*:*

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

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 $\hat{b} \xleftarrow{\$} \{0,1\}$

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 [BL20]—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 (Section 4.2), 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 [GGSW13] (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 6 (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. $m/\bot \leftarrow \mathsf{Dec}(\mathsf{ck}, \mathsf{ct}, \mathtt{w})$ *on input a commitment key ck, a ciphertext ct, and a witness $\mathtt{w}$, returns a message m or $\bot$.*

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 $\mathcal{A}$:

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

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. Below we formalize this property and show that weak semantic security together with hiding of the commitment imply strong semantic security.

Strong Semantic Security. Informally, this property states that encrypting a message m with respect to a false statement $\mathsf{x} = (\mathsf{cm}, C, y)$ produces indistinguishable ciphertexts to an adversary $\mathcal{A}$ who knows the commitment opening, even if $\mathcal{A}$ gets to see encryptions of m under other (possibly true) statements $\mathsf{x}_i = (\mathsf{cm}_i, C, y)$ but with unknown commitment opening. Formally, there exists a negligible function μ such that for all $\lambda \in \mathbb{N}$, all auxiliary strings aux and all PPT adversaries $\mathcal{A}$:

$$\left| 2 \cdot \Pr \left[\begin{array}{r} \mathsf{ck} \leftarrow \mathsf{C.Setup}(1^\lambda) \\ (\mathsf{st}, \mathsf{s}, \rho, C, y, m_0, m_1) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathrm{com}}(\cdot)}(\mathsf{ck}, \mathsf{aux}) \\ \mathsf{cm} \leftarrow \mathsf{C.Commit}(\mathsf{ck}, \mathsf{s}; \rho); b \xleftarrow{\$} \{0,1\} \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{ck}, \mathsf{x} = (\mathsf{cm}, C, y), m_b) \\ \forall \mathsf{cm}_i \in \mathcal{Q} : \mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{ck}, \mathsf{x}_i = (\mathsf{cm}_i, C, y), m_b) \\ \mathbf{ct} := \{\mathsf{ct}\} \cup \{\mathsf{ct}_i\}_{i \in [|\mathcal{Q}|]} \\ \mathbf{ct} := \bot \text{ if } C(\mathsf{s}) = y \text{ or } C \notin \mathcal{C} \end{array} : \mathcal{A}(\mathsf{st}, \mathbf{ct}) = b \right] - 1 \right| \leq \mu(\lambda)$$

where $\mathcal{O}_{\mathrm{com}}(\cdot)$ is a commitment oracle parametrized by ck and defined as follows: on input s_i, computes and returns $\mathsf{cm}_i \leftarrow \mathsf{C.Commit}(\mathsf{ck}, \mathsf{s}_i; \rho_i)$ for some randomness ρ_i, and stores cm_i in $\mathcal{Q}$.

Lemma 1. *Let* $\mathsf{C} = (\mathsf{Setup}, \mathsf{Commit})$ *be a non-interactive commitment scheme. Let* Π_{cWE} *be a witness encryption over commitments for some circuit class* $\mathcal{C}$ *over commitment scheme* C*. If* Π_{cWE} *has weak semantic security, and* C *has hiding property, then* Π_{cWE} *has strong semantic security.*

Proof. Assume $\mathcal{A}$ is a PPT adversary against strong semantic security. We construct an efficient adversary $\mathcal{B}$ that breaks weak semantic security of Π_{cWE} with non-negligible advantage.

First, $\mathcal{B}$ runs $\mathcal{A}$ with the commitment key ck received from the challenger. $\mathcal{B}$ then simulates the oracle $\mathcal{O}_{\mathrm{com}}$ for $\mathcal{A}$ in the natural way. Namely, for any input s_i, it outputs $\mathsf{cm}_i \leftarrow \mathsf{C.Commit}(\mathsf{ck}, \mathsf{s}_i; \rho_i)$ for some randomness $\rho_i \in \mathcal{S}_r$, and stores cm_i in $\mathcal{Q}$. Upon receiving a tuple $(\mathsf{st}, \mathsf{s}, \rho, C, y, m_0, m_1)$ from $\mathcal{A}$, $\mathcal{B}$ forwards the tuple to the challenger. Upon receiving the challenge ciphertext ct (for the encryption of m_b) from the challenger, $\mathcal{B}$ generates a ciphertext ct_i for each commitment $\mathsf{cm}_i \in \mathcal{Q}$. To do so, $\mathcal{B}$ selects $c \xleftarrow{\$} \{0,1\}$ and computes $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{ck}, (\mathsf{cm}_i, C, y), m_c)$ for any $\mathsf{cm}_i \in \mathcal{Q}$. Next, $\mathcal{B}$ checks whether $C(\mathsf{s}) \neq y$, and if so forwards $\mathbf{ct} := \{\mathsf{ct}\} \cup \{\mathsf{ct}_i\}_{i \in [|\mathcal{Q}|]}$ to $\mathcal{A}$. Otherwise, $\mathcal{B}$ outputs a random guess $b' \xleftarrow{\$} \{0,1\}$ for the bit b. Finally, upon receiving a guess b' from $\mathcal{A}$, $\mathcal{B}$ forwards b' to the challenger. It is easy to see that if $c = b$, then $\mathcal{B}$ is perfectly simulating strong semantic security game for $\mathcal{A}$.

To prove the lemma, we define $|\mathcal{Q}| + 2$ hybrid distributions such that the first hybrid corresponds to the strong semantic security and the last hybrid corresponds to the above game simulated by $\mathcal{B}$. We conclude the proof by showing that an adversary with non-negligible advantage in the first hybrid implies the existence of an efficient adversary with non-negligible advantage in the last hybrid.

Hybrid 0. This is the strong semantic security game. Namely,

1. The adversary $\mathcal{A}$ receives the commitment key ck, where $\mathsf{ck} \leftarrow \mathsf{C.Setup}(1^\lambda)$.
2. $\mathcal{A}$ adaptively makes commitment queries for messages s_i, and for each receives $\mathsf{cm}_i \leftarrow \mathsf{C.Commit}(\mathsf{ck}, \mathsf{s}_i; \rho_i)$.
3. After some number of queries listed in $\mathcal{Q}$, $\mathcal{A}$ outputs a tuple $(\mathsf{st}, \mathsf{s}, \rho, C, y, m_0, m_1)$ for which $C(\mathsf{s}) \neq y$. The challenger samples a random bit $b \xleftarrow{\$} \{0,1\}$, generates encryptions of m_b via $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{ck}, (\mathsf{cm}, C, y), m_b)$, and $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{ck}, (\mathsf{cm}_i, C, y), m_b)$ for all $\mathsf{cm}_i \in \mathcal{Q}$, and sends $\mathbf{ct} := \{\mathsf{ct}\} \cup \{\mathsf{ct}_i\}_{i \in [|\mathcal{Q}|]}$ to $\mathcal{A}$ as the challenge ciphertext.
4. Eventually, $\mathcal{A}$ outputs a guess b' for the bit b.

Hybrids $k = 1, \ldots, |\mathcal{Q}|$. Same as the previous hybrid, except the first k ciphertexts $\{\mathsf{ct}_i\}_{i \in [k]}$ are computed with respect to cm_i being a commitment of s. Namely,

1. *Identical to Hybrid 0.*
2. *Identical to Hybrid 0.*
3. The challenger samples a random bit $b \xleftarrow{\$} \{0,1\}$, generates encryptions of m_b via $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{ck}, (\mathsf{cm}, C, y), m_b)$, and $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{ck}, (\mathsf{cm}_i, C, y), m_b)$ $(i = 1, \ldots, |\mathcal{Q}|)$ computed as before, except in the first k ciphertexts $\{\mathsf{ct}_i\}_{i \in [k]}$, the commitment cm_i is computed as $\mathsf{cm}_i \leftarrow \mathsf{C.Commit}(\mathsf{ck}, \mathsf{s}; \rho_i)$ for some randomness $\rho_i \in \mathcal{S}_r$.
4. *Identical to Hybrid 0.*

Hybrid $k = |\mathcal{Q}| + 1$. Same as the previous hybrid, except the ciphertexts $\{\mathsf{ct}_i\}_{i \in [|\mathcal{Q}|]}$ are encryptions of m_c for a uniformly random $c \xleftarrow{\$} \{0,1\}$. Namely,

1. *Identical to Hybrid $|\mathcal{Q}|$.*

2. *Identical to Hybrid* $|\mathcal{Q}|$.
3. The challenger samples random bits $b \xleftarrow{\$} \{0,1\}$ and $c \xleftarrow{\$} \{0,1\}$, and generates encryptions of m_b and m_c respectively via $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{ck}, (\mathsf{cm}, C, y), m_b)$, and $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{ck}, (\mathsf{cm}_i, C, y), m_c)$ $(i = 1, \ldots, |\mathcal{Q}|)$.
4. *Identical to Hybrid* $|\mathcal{Q}|$.

For $k = 0, \ldots, |\mathcal{Q}| + 1$, denote by adv_i the advantage of $\mathcal{A}$ in guessing the bit b in Hybrid i. It is easy to see that for any $0 \leq i < |\mathcal{Q}|$, we have $|\mathsf{adv}_i - \mathsf{adv}_{i+1}| \leq \mathsf{negl}(\lambda)$, where $\mathsf{negl}(\lambda)$ is some negligible function. This is because the distributions $\mathcal{D}_i$ and $\mathcal{D}_{i+1}$ respectively defined by the hybrids i and $i+1$ only differ in their $(i+1)$-th ciphertext, which is the encryption of m_b under a commitment of s_{i+1} for $\mathcal{D}_i$ and encryption of m_b under a commitment of s for $\mathcal{D}_{i+1}$. Thus, the difference $|\mathsf{adv}_i - \mathsf{adv}_{i+1}|$ is exactly equal to the adversary's advantage in the hiding experiment. By the hiding property of the commitment scheme, it thus follows that this difference is negligible.
Furthermore, observe that in the last hybrid, $c = b$ with probability $1/2$ and hence we have that with probability at least $1/2$, the two distributions $\mathcal{D}_{|\mathcal{Q}|}$ and $\mathcal{D}_{|\mathcal{Q}|+1}$ are identical. This, together with the fact that $\mathcal{D}_0$ and $\mathcal{D}_{|\mathcal{Q}|}$ have a negligible difference imply that having an efficient adversary with non-negligible advantage ε against hybrid 0 results in a non-negligible advantage $\varepsilon/2$ against hybrid $|\mathcal{Q}| + 1$. This completes the proof of the lemma. □

4.2 Constructions of cWE

From Multi-Sender 2P-NISC [AMPR14]. A cWE scheme can be constructed from protocols for Multi-Sender (reusable) Non-Interactive Secure Computation (MS-NISC) [AMPR14]. 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. The ideal functionality of MS-NISC as presented in [AMPR14] is depicted in Fig. 4.

In Fig. 5, we show how to construct cWE by having black-box access to $\mathcal{F}_{\mathsf{MS\text{-}NISC}}$. The main idea is that a party acts as a receiver and sends the first message in MS-NISC containing its witness $\mathtt{w}$ in order to provide a "commitment" to that witness. Later on, any other party can use this "commitment" to create a cWE ciphertext by sending an encryption of the message and acting as the sender of the MS-NISC to provide a second message that allows for evaluating a function $f(\mathtt{w}, y)$ that outputs a decryption key iff the witness $\mathtt{w}$ satisfies a given relation. Note that the ideal functionalities used in the construction are stated for clarity and is not compatible with our game-based notion of security for cWE. By assuming a concrete secure realization of the above functionalities, one can argue about security using the corresponding simulator and use that to extract witnesses from commitments and make the proof go through.

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

Assume $f(\bot, \cdot) = f(\cdot, \bot) = \bot$.

- Initialize a list, L, of pairs of strings.
- Upon receiving a message $(\texttt{input}, x)$ from R, store x and continue.
  1. Upon receiving message $(\texttt{input}, y)$ from S_i, insert the pair (S_i, y) into L. If R is corrupted send $(S_i, f(x, y))$ to the adversary. Otherwise, send $(\texttt{messageReceived}, S_i)$ to R.
  2. Upon receiving a message $\texttt{getOutputs}$ from R, send $\{(S_i, f(x, y))\}_{(S_i,y)\in L}$ to R.

Fig. 4. MS-NISC Functionality $\mathcal{F}_{\mathsf{MS\text{-}NISC}}$

Initialization: Initialize $\mathcal{F}_{\mathsf{MS\text{-}NISC}}$ by instantiating a list L of pairs of strings.
Commit: R proceeds as follows:
- Commits to its witness $\mathtt{w}$ by calling $\mathcal{F}_{\mathsf{MS\text{-}NISC}}$ on input $(\texttt{input}, \mathtt{w})$.

Encryption: S proceeds as follows:
- Generates a key k of length $|m|$ and encrypts the message m as $\mathsf{ct} \leftarrow k \oplus m$.
- Calls $\mathcal{F}_{\mathsf{MS\text{-}NISC}}$ on input $(\mathtt{x}, k)$ and sends ct directly to R.

Decryption: R receives $(\texttt{messageReceived}, S)$ from $\mathcal{F}_{\mathsf{MS\text{-}NISC}}$ and ct from S and proceeds as follows:
- Calls $\mathcal{F}_{\mathsf{MS\text{-}NISC}}$ on input $\texttt{getOutputs}$.
- Upon receiving k from $\mathcal{F}_{\mathsf{MS\text{-}NISC}}$, outputs $m \leftarrow k \oplus \mathsf{ct}$.

Fig. 5. Construction of cWE based on MS-NISC

setting. This asymmetry between sender and receiver privacy was also observed by others [JKO13] and it opens the door for efficient constructions using oblivious transfer (OT) and privacy-free garbled circuits as described in [ZRE15].

cWE using Garbled Circuits and Oblivious Transfer. Instead of relying on the full MS-NISC functionality in a black-box way, we now do a careful analysis resulting in a protocol which uses only the properties of MS-NISC needed to obtain a protocol that satisfies the definition of cWE.

We observe that the correctness property in the definition of cWE only requires that a correctly generated ciphertext can be decrypted by the decryption algorithm. Thus, we expect the second message of MS-NISC functionality to be generated correctly. In particular, when looking into the internals of the protocol in [AMPR14], we observe that we can construct cWE from a MS-NISC protocol without the precautions against a malicious sender S. However, we still want to make sure that we preserve authenticity of the underlying garbled circuit scheme. This property guarantees that no garbled output can be constructed different

from what is dictated by the function and its inputs. In other words, the only thing a malicious receiver can do with the garbled circuit is Evaluate it on the committed input. Finally, we observe that privacy of input is not a requirement for the sender. Thus, we can consider variants of garbled circuit schemes without privacy guarantees.

Privacy-free Garbled Circuits. One of the most efficient GC schemes in terms of communication is the scheme by [ZRE15] based on a technique called half-gates. Using their technique in the privacy-free setting results in garbled circuits containing one ciphertext for each AND gate and no ciphertexts for XOR gates.

cWE from privacy-free GC and OT. We now present an efficient construction of cWE using only a privacy-free garbled circuit and oblivious transfer.

Let $\mathsf{GC} = (\mathsf{Garble}, \mathsf{Encode}, \mathsf{Eval}, \mathsf{Decode}, \mathsf{Verify})$ be a garbled circuit with correctness and authenticity, and $\Pi_{\mathsf{OT}} = (\Pi^R_{\mathsf{OT}}, \Pi^S_{\mathsf{OT}}, \Pi^O_{\mathsf{OT}})$ be an oblivious transfer protocol that realizes $\mathcal{F}_{\mathsf{OT}}$. We consider two parties E and D that respectively play the role of the encryptor and the decryptor in an execution of the cWE scheme. The construction of $\Pi_{\mathsf{cWE}} = (\mathsf{Enc}, \mathsf{Dec})$ with commitment Π^R_{OT} for circuit class $\mathcal{C}$ is given in Fig. 6.

Theorem 1. *Let $\mathcal{C}$ be a class of circuits. Let Π_{OT} be an OT protocol that realizes $\mathcal{F}_{\mathsf{OT}}$ and GC be a correct and authentic garbling scheme. The cWE scheme Π_{cWE} for $\mathcal{C}$ in Fig. 6 is correct and semantically secure as defined in Definition 6.*

Proof. **(Correctness).** Follows directly from the correctness property of the Π_{OT} and GC.

(Semantic Security). Assume that $\mathcal{A}$ is a PPT adversary against semantic security of Π_{cWE} such that, for adversarially chosen values $(\mathsf{s}, \rho, C, y, m_0, m_1)$, given an encryption of m_b under statement $\mathsf{x} = (\mathsf{cm}, C, y)$, where $\mathsf{cm} = \mathsf{Commit}(\mathsf{s}; \rho)$ and $y \neq C(\mathsf{s})$, $\mathcal{A}$ can guess the bit b with non-negligible advantage. We first observe that by the construction of Π_{cWE}, $\mathcal{A}$ can guess b correctly only by distinguishing the correct label k^1 from random. Informally, given that $C(\mathsf{s}) \neq y$, there are only two possible cases in which $\mathcal{A}$ can distinguish k^1 from random: either by the ability to gain knowledge about invalid labels $k_j^{1-\mathsf{s}_j}$ that do not correspond to $\mathcal{A}$'s committed value, or by the ability to gain knowledge about k^1 directly. We show that a successful adversary in the first case can be used to break the sender security of Π_{OT} whereas a successful adversary in the second case can be exploited to break the authenticity of GC.

In order to formally prove semantic security, we first define the experiment $\mathsf{Exp}^{\mathsf{SS\text{-}b}}_{\mathcal{A},\lambda}$ in Algorithm 2. We define b' as the output of $\mathsf{Exp}^{\mathsf{SS\text{-}b}}_{\mathcal{A},\lambda}$. Note that $\mathsf{Exp}^{\mathsf{SS\text{-}b}}_{\mathcal{A},\lambda}$ corresponds to the semantic security experiment of Π_{cWE} in Definition 6, except that b is fixed.

To prove the theorem, let us assume by contradiction that there is an adversary $\mathcal{A}$ that breaks the semantic security of Π_{cWE}. That is, for a non-negligible function ϵ, we have

$$\left|\Pr[1 \leftarrow \mathsf{Exp}^{\mathsf{SS\text{-}0}}_{\mathcal{A},\lambda}] - \Pr[1 \leftarrow \mathsf{Exp}^{\mathsf{SS\text{-}1}}_{\mathcal{A},\lambda}]\right| \geq \epsilon(\lambda)$$

Primitives: A correct and authentic garbling scheme $\mathsf{GC} = (\mathsf{Garble}, \mathsf{Encode}, \mathsf{Eval}, \mathsf{Decode})$, and a 2-round OT $\Pi_{\mathsf{OT}} = (\Pi^R_{\mathsf{OT}}, \Pi^S_{\mathsf{OT}}, \Pi^O_{\mathsf{OT}})$.
Commit: D with secret $\mathtt{s} \in \{0,1\}^n$ plays the role of the receiver in n instances of Π_{OT} and computes $(\mathsf{cm}, \mathtt{w})$ as follows:

- Select $\rho^R_j \xleftarrow{\$} \{0,1\}^\lambda$, and compute $m^R_j \leftarrow \Pi^R_{\mathsf{OT}}(\mathsf{s}_j; \rho^R_j)$ for $j \in [n]$.
- Define $\mathsf{cm} = \{m^R_j\}_{j\in[n]}$, and $\mathtt{w} = (\mathsf{s}, \{\rho^R_j\}_{j\in[n]})$. Note that $\mathtt{w}$ can be seen as an opening of cm.

Common inputs: A security parameter λ, a circuit $C \in \mathcal{C}$, a commitment key ck, and a statement $\mathtt{x} = (\mathsf{cm}, C, y)$.
Encryption.: E plays the role of the sender in n instances of Π_{OT} and computes a ciphertext $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2)$ as follows:

1. Let $C_{\mathtt{x}}$ be a circuit that realizes the following relation $\mathcal{R}$ on $\mathtt{x}$: $\mathcal{R}(\mathtt{x} = (\mathsf{cm}, C, y), (\mathsf{s}, \vec{\mathsf{d}})) = 1$ iff $(\mathsf{s}, \vec{\mathsf{d}})$ opens cm and $C(\mathsf{s}) = y$. Compute $(\mathbf{C}, e, d) \leftarrow \mathsf{Garble}(1^\lambda, C_{\mathtt{x}})$, where $e := \{k^0_j, k^1_j\}_{j\in[n]}$, and $d := (k^0, k^1) \in \{0,1\}^{2|m|}$.
2. For $j \in [n]$, select $\rho^S_j \xleftarrow{\$} \{0,1\}^\lambda$, and compute $m^S_j = \Pi^S_{\mathsf{OT}}(k^0_j, k^1_j, m^R_j; \rho^S_j)$.
3. Compute $\mathsf{ct}_1 = k^1 \oplus m$ and $\mathsf{ct}_2 = (\mathbf{C}, \{m^S_j\}_{j\in[n]})$.
4. Send $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2)$ to D.

Decryption: Given $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2)$ and $\mathtt{w} = (\mathsf{s}, \{\rho^R_j\}_{j\in[n]})$, D proceeds as follows:

1. Parse ct_2 as $(\mathbf{C}, \{m^S_j\}_{j\in[n]})$.
2. Execute $k^{\mathsf{s}_j}_j = \Pi^O_{\mathsf{OT}}(m^S_j, \mathsf{s}_j, \rho^R_j)$ for $j \in [n]$, and $Y = \mathsf{Eval}(\mathbf{C}, \{k^{\mathsf{s}_j}_j\}_{j\in[n]})$.
3. Compute $m = Y \oplus \mathsf{ct}_1$.

Fig. 6. cWE based on GC and OT

We now use a standard hybrid argument and define several games, where the first is $\mathsf{Exp}^{\mathsf{SS\text{-}0}}_{\mathcal{A},\lambda}$, the last is $\mathsf{Exp}^{\mathsf{SS\text{-}1}}_{\mathcal{A},\lambda}$, and the intermediate hybrids are defined as follows:
Hybrid 0 is defined as $\mathsf{Exp}^{\mathsf{SS\text{-}0}}_{\mathcal{A},\lambda}$.
Hybrid 1 is the same as **Hybrid 0**, except that the messages $\{m^S_j\}_{j\in[n]}$ are computed by the OT simulator i.e., as $\{m^S_j\}_{j\in[n]} \leftarrow \mathsf{Sim}(1^\lambda, \{m^R_j\}_{j\in[n]})$.
Hybrid 2 is the same as **Hybrid 1**, except that ct_1 is defined as $\mathsf{ct}_1 := k^1 \oplus m_1$.
Hybrid 3 is defined as $\mathsf{Exp}^{\mathsf{SS\text{-}1}}_{\mathcal{A},\lambda}$.

By assumption, $\mathcal{A}$ must distinguish some pair of adjacent intermediate hybrids. That is, for some $i \in \{0,1,2\}$, we must have

$$\left|\Pr[1 \leftarrow \mathsf{Hybrid}^i_{\mathcal{A},\lambda} - \Pr[1 \leftarrow \mathsf{Hybrid}^{i+1}_{\mathcal{A},\lambda}]\right| \geq \frac{1}{3}\epsilon(\lambda)$$

We now analyze all three cases:

- $i = 0$. Notice that the only difference between **Hybrid 0** and **Hybrid 1** is that in the former, the sender's message $\{m^S_j\}_{j\in[n]}$ is computed by a real

Algorithm 2 $\mathsf{Exp}^{\mathsf{SS\text{-}b}}_{\mathcal{A},\lambda}$

$(\mathsf{st}, \mathsf{s}, \rho, C, y, m_0, m_1) \leftarrow \mathcal{A}(1^\lambda)$
$\rho = \rho_1 || \ldots || \rho_n$; $\mathsf{cm} = \{m_j^R\}_{j\in[n]}$, where $m_j^R \leftarrow \Pi_{\mathsf{OT}}^R(\mathsf{s}_j; \rho_j)\ \forall j \in [n]$.
$\mathtt{x} := (\mathsf{cm}, C, y)$; $(\mathbf{C}, e, d) \leftarrow \mathsf{Garble}(1^\lambda, C_{\mathtt{x}})$ where $e := \{k_j^0, k_j^1\}_{j\in[n]}$, and $d := (k^0, k^1)$.
$\rho_j^S \xleftarrow{\$} \{0,1\}^\lambda$; $m_j^S = \Pi_{\mathsf{OT}}^S(k_j^0, k_j^1, m_j^R; \rho_j^S)$ $(\forall j \in [n])$.
$\mathsf{ct}_1 = k^1 \oplus m_b$ and $\mathsf{ct}_2 = (\mathbf{C}, \{m_j^S\}_{j\in[n]})$; $\mathsf{ct} := (\mathsf{ct}_1, \mathsf{ct}_2)$.
$\mathsf{ct} := \perp$ if $C(\mathsf{s}) = y$ or $C \notin \mathcal{C}$ or $|m_0| \neq |m_1|$
$b' \leftarrow \mathcal{A}(\mathsf{st}, \mathsf{ct})$

sender (World 0), whereas in the latter, it is computed by the simulator (World 1). Assuming that $\mathcal{A}$ can distinguish hybrids 0 and 1, we construct an adversary $\mathcal{B}$ against sender security of Π_{OT} that distinguishes the two worlds with the same probability. $\mathcal{B}$ works as follows:

1. $\mathcal{B}$ invokes $\mathcal{A}(1^\lambda)$ and obtains $(\mathsf{s}, \rho, C, y, m_0, m_1)$.
2. If $|m_0| \neq |m_1|$ or $C(\mathsf{s}) = y$, $\mathcal{B}$ aborts; otherwise, it parses $\rho = \rho_1 || \ldots || \rho_n$ and defines $\mathsf{cm} = \{m_j^R\}_{j\in[n]}$, where $m_j^R \leftarrow \Pi_{\mathsf{OT}}^R(\mathsf{s}_j; \rho_j)$ for $j \in [n]$. Let $\mathtt{x} = (\mathsf{cm}, C, y)$. As an environment controlling the OT execution, $\mathcal{B}$ provides the input of sender and receiver to the OT challenger as follows:
   - computes a garbling of circuit $C_{\mathtt{x}}$ (as defined in Fig. 6) by $(\mathbf{C}, e, d) \leftarrow \mathsf{Garble}(1^\lambda, C_{\mathtt{x}})$ and sends the input keys to the OT challenger as the sender's input.
   - sends s to the OT challenger as the receiver's choice bits.
3. The OT challenger computes the sender's message $\{m_j^S\}_{j\in[n]}$ either by invoking a real sender (World 0), or by invoking the simulator (World 1), and sends it to $\mathcal{B}$.
4. $\mathcal{B}$ parses $d = (k^0, k^1)$, and forwards $\mathsf{ct} := (\mathsf{ct}_1, \mathsf{ct}_2)$ to the cWE adversary $\mathcal{A}$, where $\mathsf{ct}_1 = k^1 \oplus m_0$ and $\mathsf{ct}_2 = (\mathbf{C}, \{m_j^S\}_{j\in[n]})$.

It is clear that $\mathcal{B}$ has the same advantage in breaking sender security of Π_{OT} as $\mathcal{A}$ in distinguishing the two hybrids.

– $i = 1$. The only difference in **Hybrid 1** and **Hybrid 2** is in how we generate ct_1 (that is $\mathsf{ct}_1 := k^1 \oplus m_{i-1}$ in **Hybrid i**). To argue indistinguishability of the two hybrids, it suffices to show that k^1 is indistinguishable from random. To achieve this, we observe that because in both hybrids, the sender's message $\{m_j^S\}_{j\in[n]}$ is computed by the simulator i.e., as $\{m_j^S\}_{j\in[n]} \leftarrow \mathsf{Sim}(1^\lambda, \{m_j^R\}_{j\in[n]})$, $\mathcal{A}$ cannot distinguish k^1 from random by the ability of knowing invalid labels. Thus, the only way $\mathcal{A}$ can distinguish k^1 from random should be by directly forging an output key k^1 for the garbled circuit $\mathbf{C}$. It is therefore straightforward to use a successful adversary that distinguishes the two hybrids with non-negligible advantage to break the authenticity of the underlying garbling scheme.

– $i = 2$. This is handled identically to $i = 0$, except that in this case $\mathsf{ct}_1 := k^1 \oplus m_1$ encrypts m_1 instead of m_0.

We conclude the proof by this observation that in any of the three cases, we reach a contradiction and thus our assumption of the existence of $\mathcal{A}$ against the semantic security of Π_{cWE} cannot be true. □

Remark 3. The commitment scheme in Π_{cWE} is the receiver's algorithm of Π_{OT} and therefore by UC-security of Π_{OT}, it satisfies both extractability and hiding property. Using Lemma 1 and weak semantic security shown in Theorem 1, one can then conclude that Π_{cWE} also achieves strong semantic security.

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 2. *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 5.*

Proof. Assume by contradiction that there exists an adversary $\mathcal{A}_{\mathrm{ECW}}$ with non-negligible advantage in $\mathrm{Game}^{\mathsf{IND\text{-}CPA}}_{\Gamma,\mathcal{A},\mathcal{Z},\mathcal{E}}$ in the ECW setting as described in Section 3.1. We construct an adversary $\mathcal{A}_{\mathsf{cWE}}$ with black-box access to $\mathcal{A}_{\mathrm{ECW}}$ that has non-negligible advantage in breaking strong semantic security of Π_{cWE} as defined above. We assume (w.l.o.g.) that $\mathcal{A}_{\mathrm{ECW}}$ only corrupts one party P_a[7]. $\mathcal{A}_{\mathsf{cWE}}$ proceeds as follows:

1. Upon receiving the commitment key ck from the challenge, $\mathcal{A}_{\mathsf{cWE}}$ proceeds as follows:
   (a) $\mathcal{A}_{\mathsf{cWE}}$ acts as the environment $\mathcal{Z}$ orchestrating the execution of the blockchain protocol Γ towards $\mathcal{A}_{\mathrm{ECW}}$, placing the commitment key ck in the genesis block. $\mathcal{A}_{\mathsf{cWE}}$ acts exactly as $\mathcal{Z}$ in $\mathrm{Game}^{\mathsf{IND\text{-}CPA}}_{\Gamma,\mathcal{A},\mathcal{Z},\mathcal{E}}$.
   (b) $\mathcal{A}_{\mathsf{cWE}}$ simulates honest parties P_h executing the setup phase and publishing a valid $(\mathsf{cm}_h, \sigma_h)$ on the blockchain. To simulate cm_h for each honest party, $\mathcal{A}_{\mathsf{cWE}}$ calls the oracle $\mathcal{O}_{\mathrm{com}}$ on some random input $\mathsf{sk}_{L,h}$ and sets cm_h to be $\mathcal{O}_{\mathrm{com}}$'s output.
   (c) At some point, $\mathcal{A}_{\mathrm{ECW}}$ outputs challenge parameters $\mathbf{B}, \mathsf{sl}, \mathsf{R}, m_0, m_1$ from its view of the blockchain. $\mathcal{A}_{\mathsf{cWE}}$ constructs 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.
   (d) Finally, if there exists a valid setup message $(\mathsf{cm}_a, \sigma_a)$ published in the common prefix $\mathbf{B}^{\lceil\kappa}$ by P_a (*i.e.* the corrupted party P_a is in $\mathcal{P}_{Setup}$), $\mathcal{A}_{\mathsf{cWE}}$ extracts $\mathsf{sk}_{L,a}, \rho_a$ from cm_a using the extractability of the commitment scheme C and outputs $(\mathsf{st}, \mathsf{sk}_{L,a}, \rho_a, C, 1, m_0, m_1)$ to the challenger. Otherwise, $\mathcal{A}_{\mathsf{cWE}}$ outputs $(\mathsf{st}, \mathsf{sk}_{L,k}, \rho_k, C, 1, m_0, m_1)$ to the challenger, where $\mathsf{sk}_{L,k}, \rho_k$ are chosen at random and such that $C(\mathsf{sk}_{L,k}) \neq 1$.

2. Upon receiving ciphertexts $\mathbf{ct} = \{\mathsf{ct}\} \cup \{\mathsf{ct}_h\}_{P_h \in \mathcal{P}_{Setup}}$ from the challenger, if $P_a \in \mathcal{P}_{Setup}$, then $\mathsf{ct} = \mathsf{ct}_a$ was computed w.r.t. P_a's commitment cm_a and ct_h computed w.r.t. the honest party's commitment cm_h. Otherwise, if only honest parties are in $\mathcal{P}_{Setup}$, $\mathcal{A}_{\mathsf{cWE}}$ forwards the ECW ciphertexts $\mathbf{ct} = \{\mathsf{ct}_h\}_{P_h \in \mathcal{P}_{Setup}}$ to $\mathcal{A}_{\mathrm{ECW}}$. $\mathcal{A}_{\mathsf{cWE}}$ continues the execution of Γ with $\mathcal{A}_{\mathrm{ECW}}$ from the round where it stopped when $\mathcal{A}_{\mathrm{ECW}}$ outputted challenge parameters $\mathbf{B}, \mathsf{sl}, \mathsf{R}, m_0, m_1$.
3. Upon receiving a guess b' from $\mathcal{A}_{\mathrm{ECW}}$, $\mathcal{A}_{\mathsf{cWE}}$ forwards b' to the challenger.

First, notice that $\mathcal{A}_{\mathrm{ECW}}$ has the same access to the underlying blockchain protocol Γ (and to the system parameters in the genesis block) as in $\mathrm{Game}^{\mathsf{IND\text{-}CPA}}_{\Gamma,\mathcal{A},\mathcal{Z},\mathcal{E}}$.

[7] In reality there will be more than one corrupted party; the main argument underlying our proof holds regardless.

In case $\mathcal{A}_{\text{ECW}}$ provided a valid setup message, it receives $\mathbf{ct}$ containing a cWE ciphertext ct_a generated with respect to its commitment cm_a and the circuit encoding the lottery predicate $\mathsf{lottery}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}_{L,i})$, where $\mathbf{B}$, sl and R provided by $\mathcal{A}_{\text{ECW}}$ are hardwired. Moreover, $\mathbf{ct}$ contains ciphertexts ct_h for each cm_h, encrypting the same m_b as in ct_a. Hence, $\mathbf{ct}$ is distributed exactly as in $\text{Game}^{\mathsf{IND\text{-}CPA}}_{\Gamma,\mathcal{A},\mathcal{Z},\mathcal{E}}$. If $\mathcal{A}_{\text{ECW}}$ has non-negligible advantage in $\text{Game}^{\mathsf{IND\text{-}CPA}}_{\Gamma,\mathcal{A},\mathcal{Z},\mathcal{E}}$, it is able to distinguish whether ct_a contains m_0 or m_1 with non-negligible advantage even though it does not have $\mathsf{sk}_{L,a}$ and $\mathsf{cm}_a \leftarrow \mathsf{Commit}(\mathsf{ck}, \mathsf{sk}_{L,a}; \rho_a)$ such that $\mathsf{lottery}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}_{L,a}) = 1$, *i.e.* it does not have $\mathsf{sk}_{L,a}$ such that $C(\mathsf{sk}_{L,a}) = 1$. This means that, by forwarding guess b' from $\mathcal{A}_{\text{ECW}}$, $\mathcal{A}_{\mathsf{cWE}}$ in the cWE semantic security game has the same advantage as $\mathcal{A}_{\text{ECW}}$ in $\text{Game}^{\mathsf{IND\text{-}CPA}}_{\Gamma,\mathcal{A},\mathcal{Z},\mathcal{E}}$. In case it did not provide a valid setup message, $\mathcal{A}_{\text{ECW}}$ only sees $\mathbf{ct} = \{\mathsf{ct}_h\}_{P_h \in \mathcal{P}_{Setup}}$ with ct_h being an encryption of m_b with respect to the commitments cm_h for which it does not know the opening. Hence, $\mathbf{ct}$ is again distributed exactly as in $\text{Game}^{\mathsf{IND\text{-}CPA}}_{\Gamma,\mathcal{A},\mathcal{Z},\mathcal{E}}$ with probability 1. In this case, by an analogous argument as before, the advantage of the adversary $\mathcal{A}_{\mathsf{cWE}}$ must be the same as the advantage of the adversary $\mathcal{A}_{\text{ECW}}$ in $\text{Game}^{\mathsf{IND\text{-}CPA}}_{\Gamma,\mathcal{A},\mathcal{Z},\mathcal{E}}$.

Since we assume that $\mathcal{A}_{\text{ECW}}$ has a non-negligible advantage, $\mathcal{A}_{\mathsf{cWE}}$ will also obtain a non-negligible advantage and thus break the cWE scheme we assume is secure. Hence, Π_{ECW} is an $\mathsf{IND\text{-}CPA}$-secure ECW scheme. □

5.2 Other Instantiations

ECW from target anonymous channels [GHM+21,BGG+20]. 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 [GHM+21,BGG+20,GHK+21]. 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. [BGG+20] 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 [BGG+20] 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 [GHM+21], 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 [DS15]. Derler and Slamanig [DS15] (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 in Appendix B. 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) [CL06] (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 [GHK+21]. 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 3 shows the details of the game.

Algorithm 3 $\mathrm{Game}^{\mathsf{IND\text{-}CCA2}}_{\Gamma,\mathcal{A},\mathcal{Z},\mathsf{E}}$

$\mathsf{view}^r \leftarrow \mathsf{EXEC}^{\Gamma}_r(\mathcal{A}^{\mathcal{O}_{\mathsf{EtF}}}, \mathcal{Z}, 1^\lambda)$ ▷ $\mathcal{A}$ 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 7 (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 $\lambda \in \mathbb{N}$:*

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

To add IND-CCA2 security to an IND-CPA secure EtF scheme (as defined in Definition 5) we can use standard transformations such as [FO99,Sah99]. In the transformation based on [Sah99] 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 [FO99] 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 8 (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 4 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 4 $\mathrm{Game}^{\mathsf{EUF\text{-}CMA}}_{\Gamma,\mathcal{A},\mathcal{Z},\mathcal{U}}$

$\mathsf{view} \leftarrow \mathsf{EXEC}^{\Gamma}(\mathcal{A}, \mathcal{Z}, 1^{\lambda})$ ▷ $\mathcal{A}$ 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) [CL06] 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 Section 6.4, 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 9 (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 5 and Algorithm 6 with more than negligible probability in the security parameter.*

Scenario 0 ($b = 0$). *In this scenario (Algorithm 5), $\mathcal{A}$ 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 8). The adversary then continues the execution until round $\tilde{r}$ where it outputs a bit b'.*

Scenario 1 ($b = 1$). *This scenario (Algorithm 6) is identical to scenario 0 but instead of interacting with $\mathcal{O}_{\mathsf{AfP}}$, the adversary interacts with a simulator* Sim.

Algorithm 5 $b = 0$

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

Algorithm 6 $b = 1$

$\mathsf{view}^r \leftarrow \mathsf{EXEC}^{\Gamma}_{r}(\mathcal{A}, \mathcal{Z}, 1^\lambda)$
$\mathcal{A}^{\mathsf{Sim}}(\mathsf{view}^r_{\mathcal{A}})$
$\mathsf{view}^{\tilde{r}} \leftarrow \mathsf{EXEC}^{\Gamma}_{(\mathsf{view}^r, \tilde{r})}(\mathcal{A}, \mathcal{Z}, 1^\lambda)$
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 More Efficient AfP based on VRF

VRF-based Lottery. This section introduces a specific lottery mechanism which will be the underlying lottery predicate for the AfP described in the next section. The backbone of the lottery is a VRF scheme VRF as described in [DGKR18]. This VRF has the properties of simulatability and unpredictability under malicious key generation which will become useful when arguing about security of the AfP. The VRF scheme is a tuple $(\mathsf{VRF.Gen}, \mathsf{VRF.Prove}, \mathsf{VRF.Verify})$ where $\mathsf{VRF.Gen}(1^\kappa)$ outputs a pair of keys $(\mathsf{VRF.pk}, \mathsf{VRF.sk})$. The $\mathsf{VRF.Prove}$ takes as input a value x and outputs a pair $(y, \pi) \leftarrow \mathsf{VRF.Prove}_{\mathsf{VRF.sk}}(x)$ which is the output value y and the correctness certificate π. The verification is then done by evaluating $\mathsf{VRF.Verify}_{\mathsf{VRF.pk}}(x, y, \pi)$ which outputs 1 iff π attests to the correctness of y as the output of the VRF evaluated on x with key $\mathsf{VRF.sk}$.
We recall the blockchain setup described in Section 2.1 where each party P_i is represented by a pair $(\mathsf{Sig.sk}_i, \mathsf{sk}_{L,i})$ associated with public data $(\mathsf{Sig.pk}_i, \mathsf{aux}_i, \mathsf{stake}_i)$. Let aux_i contain a VRF public key $\mathsf{VRF.pk}_i$ as described above and let the lottery secret key be $\mathsf{sk}_{L,i} = (\mathsf{Sig.pk}_i, \mathsf{VRF.sk}_i)$.
Finally, we introduce a function $\mathsf{param}(\mathbf{B}, \mathsf{sl})$. This function outputs a tuple $(\{\mathsf{Sig.pk}_i, \mathsf{VRF.pk}_i, \mathsf{stake}_i\}_{i\in[n]}, \eta, \phi)$ associated with the specific blockchain $\mathbf{B}$ and slot sl. Beyond obtaining the public information $(\mathsf{Sig.pk}_i, \mathsf{VRF.pk}_i, \mathsf{stake}_i)$ the function also returns a nonce, η, as well as a public function $\phi(\cdot)$ which on input stake_i computes the threshold for winning the lottery.
The lottery predicate based on the VRF is described in Algorithm 7.

Algorithm 7 $\mathsf{lottery}_{\mathsf{VRF}}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}_{L,j})$

$(\{\mathsf{Sig.pk}_i, \mathsf{VRF}_{L,i}, \mathsf{stake}_i\}_{i\in[n]}, \eta, \phi) \leftarrow \mathsf{param}(\mathbf{B}, \mathsf{sl})$
$(\mathsf{Sig.pk}_j, \mathsf{VRF.sk}_j) \leftarrow \mathsf{sk}_{L,j}$
$(y, \pi) \leftarrow \mathsf{VRF.Prove}_{\mathsf{sk}_{L,j}}(\mathsf{sl}||\mathsf{R}||\eta)$
if $y < \phi(\mathsf{stake}_j)$ **then**
  if $\mathsf{VRF.Verify}_{\mathsf{vk}_{L,j}}(\mathsf{sl}||\mathsf{R}||\eta, y, \pi) =$ **then**
    return 1
  end if
end if
return 0

VRF-based AfP. With the VRF-based lottery $\mathsf{lottery}_{\mathsf{VRF}}$ in place, we are now ready to introduce the VRF-based AfP. We first note that our general approach of applying a SoK for the knowledge of a secret key still applies. However, using the structure of the lottery, and in particular the VRF, allows for a much more efficient AfP which has applications in most PoS settings as well.
The AfP scheme uses a $\mathsf{NIZKPoK}$ which has a setup executed as a part of the blockchain setup such that the CRS is in the genesis block. The algorithms for the scheme are $\pi \leftarrow \mathsf{NIZKPoK.Prove}(\mathsf{crs}, \mathtt{x}, \mathtt{w})$ and

$\{0,1\} \leftarrow \mathsf{NIZKPoK.Verify}(\mathsf{crs}, \mathsf{x}, \pi)$.

Protocol Π_{AfP} The VRF-based AfP protocol Π_{AfP} is described below.

Authenticate. $\sigma \leftarrow \Pi_{\mathsf{AfP}}.\mathsf{Sign}(\mathbf{B}, \mathsf{sl}, \mathsf{S}, \mathsf{sk}_{L,j}, m)$ To authenticate a message, m, a party first checks that $\mathsf{lottery}_{\mathsf{VRF}}(\mathbf{B}, \mathsf{sl}, \mathsf{S}, \mathsf{sk}_{L,j}) = 1$. It then obtains the output and certificate $(y, \pi_{\mathsf{VRF}}) \leftarrow \mathsf{VRFrf.Prove}_{\mathsf{VRF.sk}_{L,j}}(\mathsf{sl}||\mathsf{R}||\eta)$. Finally, it produces $\pi_{\mathsf{NIZKPoK}} \leftarrow \mathsf{NIZKPoK.Prove}\{\sigma_{\mathsf{SIG}} \mid \mathsf{Sig.Verify}_{\mathsf{Sig.pk}_j}(\sigma_{\mathsf{SIG}}, m) = 1\}$ which is a NIZK-PoK of a signature produced under $\mathsf{Sig.sk}_j$.
It then outputs a tuple $\sigma_{\mathsf{AfP}} \leftarrow (\mathsf{Sig.pk}_j, y, \pi_{\mathsf{VRF}}, \pi_{\mathsf{NIZKPoK}})$

Verify. $\{0,1\} \leftarrow \Pi_{\mathsf{AfP}}.\mathsf{Verify}(\tilde{\mathbf{B}}, \mathsf{sl}, \mathsf{S}, \sigma, m)$ To verify an AfP tag the verifier obtains parameters from the blockchain $(\{\mathsf{Sig.pk}_i, \mathsf{VRF.pk}_i, \mathsf{stake}_i\}_{i\in[n]}, \eta, \phi) \leftarrow \mathsf{param}(\mathbf{B}, \mathsf{sl})$. It then parses the tag as $\sigma_{\mathsf{AfP}} \leftarrow (\mathsf{Sig.pk}_j, y, \pi_{\mathsf{VRF}}, \pi_{\mathsf{NIZKPoK}})$ and gets the VRF verification key $\mathsf{VRF.pk}_j$ for the party that the AfP points to. It then checks the following

1. Makes sure that $\mathsf{VRF.Verify}_{\mathsf{VRF.pk}_j}(\mathsf{sl}||\mathsf{R}||\eta, y, \pi_{\mathsf{VRF}}) = 1$ *i.e.* the VRF output was correctly generated under lottery key of party P_j.
2. Checks that $\mathsf{NIZKPoK.Verify}(\pi_{\mathsf{NIZKPoK}}, (\mathsf{Sig.pk}_j, m)) = 1$ which verifies the proof of signature knowledge.
3. And $y < \phi(\mathsf{stake}_j)$ which makes sure that the lottery was conducted correctly with the stake of P_j.

If all checks go through, the algorithm outputs 1. Otherwise, it outputs 0.

Theorem 3. *Let* VRF *be the VRF scheme described in [DGKR18] with a secure NIZK-PoK scheme* NIZKPoK*. The protocol* Π_{AfP} *(described above) running in the context of a blockchain protocol* Γ *with underlying lottery* $\mathsf{lottery}_{\mathsf{VRF}}$ *(Algorithm 7) is an AfP scheme according to Definition 8.*

Proof. (Sketch) Assume that an adversary $\mathcal{A}$ obtains a non-negligible advantage in $\mathrm{Game}^{\mathsf{EUF\text{-}CMA}}_{\Gamma,\mathcal{A},\mathcal{Z},\mathcal{U}}$. In other words, $\mathcal{A}$ is able to forge an AfP tag with noticeable probability. We claim that such an adversary can do at least one of two things:

1. It can forge a signature under ($\mathsf{Sig.sk}_i$, thus violating the EUf-CMA security of the signature scheme.
2. It can produce a convincing proof π_{NIZKPoK} of knowledge of a signature produced with a signature secret key where the corresponding lottery secret key did not win the lottery. Since we assume that only P_i knows the pair $(\mathsf{Sig.sk}_i, \mathsf{sk}_{L,i})$, such a convincing proof must violate the soundness of the NIZKPoK scheme.
3. It can forge a VRF certificate such that the VRF.Verify algorithm accepts a certificate π_{VRF} under a different $y' \neq y$ when evaluated with the VRF public key VRF.pk of the adversary and thus convinces the authenticator. This violates the simulator of the simulatable VRF introduced in [DGKR18].

Since we assume that NIZKPoK is a secure NIZK-PoK scheme and VRF a secure scheme based on the functionality in [DGKR18], we conclude that Π_{AfP} is secure with respect to Definition 8.

AfP Privacy. This simple AfP mechanism for a VRF-based lottery predicate does not only satisfy the existential unforgeability definition of an AfP. It has AfP privacy.

Theorem 4. *The AfP protocol Π_{AfP} described above with underlying lottery predicate $\mathsf{lottery}_{\mathsf{VRF}}$ running in the context of blockchain Γ has AfP privacy.*

Proof (Sketch).
We use the notation $\mathcal{D}_0$ and $\mathcal{D}_1$ for the distribution of outputs when the adversary is put in scenario 0 and scenario 1, respectively. Our aim is to show the existence of a simulator such that $\mathcal{D}_0$ and $\mathcal{D}_1$ are computational indistinguishable.
We introduce 4 hybrids $\{\mathsf{H}_i\}_{i=1,\ldots,4}$ where $\mathsf{H}_1 = \mathcal{D}_0$ and $\mathsf{H}_4 = \mathcal{D}_1$.

- H_2 This hybrid is identical to H_1 but we use the simulator of the $\mathsf{NIZKPoK}$ scheme to simulate the proof of signature knowledge that convinces. Due to the security of the $\mathsf{NIZKPoK}$ scheme H_2 and H_1 are indistinguishable to the PPT adversary $\mathcal{A}$.
- H_3 The difference from H_2 is that instead of invoking the VRF scheme VRF we are using the simulatability of the construction to output valid proofs.
- H_4 This hybrid does not need access to any lottery winning secret keys and thus can be completely simulated by Sim. It is still necessary to observe the distribution of the stake to correctly simulate the output of the oracle $\mathcal{O}_{\mathsf{AfP}}$.

Assume that an adversary can distinguish $\mathcal{D}_0$ and $\mathcal{D}_1$ with non-negligible probability ρ. It implies that there exists an $i \in \{1, 2, 3\}$ such that H_i and H_{i+1} can be distinguished with non-negligible probability at least $\rho/3$. This contradicts the indistinguishability of hybrids. Thus, we conclude that the distributions $\mathcal{D}_0$ and $\mathcal{D}_1$ are computationally indistinguishable due to the simulator Sim obtained through the sequence of hybrids above.

6.5 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 Appendix C.1.

Definition 10 (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 11 (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 7).*
2. *$\mathcal{U}$ is a secure and private AfP (Definition 8 and Definition 9).*
3. *Γ is honest committee friendly (Definition 10).*

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.

7.2 Security and Proof Intuition

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

Theorem 5 (informal). *Let Γ^V be a YOSO MPC friendly blockchain, Π_{TIBE} be a robust secure threshold IBE as in Section 2.5 with threshold $n/2$, and Π_{ECW} a secure* IND-CCA2 *ECW. The construction in Section 7.1 is a secure EtF.*

At the high level we show security in two steps. We first show the security of our construction for a simplified non-threshold setting with a standard IBE instead of a threshold one with key-sharing. In other words we do not temporarily consider the real case where there is a committee of parties holding a share of the master secret key, but we assume the execution uses a "key provider" oracle holding the master secret key of the IBE scheme. In particular, we define the behavior of oracle $\mathcal{O}^{\text{k-provider}}_{\mathsf{msk}}$ as follows: given in input a blockchain $\mathbf{B}$ and a slot sl (such that the latest slot of $\mathbf{B}$ is sl), it broadcasts a ciphertext for the winner[8] of the slot computed as $\mathsf{ct}^{\mathsf{sk}}_{\mathsf{sl}} \leftarrow \mathsf{Enc}_{\mathsf{ECW}}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}_{\mathsf{sl}})$ where $\mathsf{sk}_{\mathsf{sl}} \leftarrow \mathsf{IBE.Keygen}(\mathsf{msk}, (\mathsf{sl}, \mathsf{R}))$.

As a second step in the proof we show that, in the threshold-setting (where the master secret key is actually shared), one can obtain an adversary with a comparable advantage in the threshold-setting from an adversary in the non-threshold setting. Intuitively, we can do this because of the low amount of stake the adversary is controlling and the security of threshold-IBE.

Finally, our proof considers the case of an adversary with static corruptions, but we point out it can be straightforwardly compiled to a full round and committee YOSO setting as described in Section 6.

Proof. We proceed in two steps: first we consider an idealized case where there is no threshold committee; we then show we can prove security of our threshold construction from this setting.

1. The non-threshold case. The simplified setting we will now show security for is in Fig. 7. A point on the view of the adversary: we recall that, at any given point in time, a valid blockchain execution contains ciphertexts $\mathsf{ct}^{\mathsf{sk}}_{\mathsf{sl}}$, encrypting slot-specific secret keys for the winner of the slot sl in the chain. In the non-threshold setting, they correspond to the output of the key-provider oracle (in the actual construction, there are more ciphertexts, each containing a share of the key).

Now assume an adversary $\mathcal{A}^{\text{no-thresh}}_{EtF}$ for the EtF security experiment controlling at most an α fraction of the stake with non-negligible success probability in the EtF security experiment. We first to construct an adversary $\mathcal{A}_{\text{IBE}}$ for IBE security using $\mathcal{A}^{\text{no-thresh}}_{EtF}$. Adversary $\mathcal{A}_{\text{IBE}}$ works as follows:

– On receiving the IBE public parameters from the IBE challenger, it injects into blockchain genesis block the IBE's master public. The adversary $\mathcal{A}^{\text{no-thresh}}_{EtF}$ declares a corrupted set of parties S_{corr} and then $\mathcal{A}_{\text{IBE}}$ runs an

[8] This is actually a vector, one for each winner in the slot. For clarity of discussion we just consider the case for one winner. The general case follows straightforwardly.

The non-threshold setting we consider is the same as that in Section 7.1 with the following exceptions:

- At the beginning of the run of the blockchain, there is no sharing of the master secret key of the IBE scheme.
- We let the honest parties run exactly as in the other construction, with the exception that they validate and messages related to the shares of the master secret keys, as well as of the secret keys for specific slots.
- We change the way we encapsulate the secret-key for a certain slot. While in Section 7.1 we require committee members to each broadcast a ciphertext containing a share of the secret-key for slot sl, here we instead replace that stage with the execution of the following oracle $\mathcal{O}_{\mathsf{msk}}^{\text{k-provider}}$.

  $\mathcal{O}_{\mathsf{msk}}^{\text{k-provider}}(\mathbf{B}, \mathsf{sl})$:
  - $\mathsf{sk}_{\mathsf{sl}} \leftarrow \mathsf{IBE.Keygen}(\mathsf{msk}, (\mathsf{sl}, \mathsf{R}))$
  - $\mathsf{ct}_{\mathsf{sl}}^{\mathsf{sk}} \leftarrow \mathsf{Enc}_{\mathsf{ECW}}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}_{\mathsf{sl}})$
  - Broadcast $\mathsf{ct}_{\mathsf{sl}}^{\mathsf{sk}}$

Fig. 7. Hybrid non-threshold setting for proof of security

execution of the blockchain with $\mathcal{A}_{EtF}^{\text{no-thresh}}$ where $\mathcal{A}_{\text{IBE}}$ simulates the honest parties. In this execution $\mathcal{A}_{\text{IBE}}$ acts as key-provider oracle, which it emulates as follows. We distinguish two cases depending on whether the winner of the slot is a corrupted party or an honest one[9]. On query $(\mathbf{B}, \mathsf{sl})$:

- *if a corrupted party has won* the role for slot sl (i.e. $\mathsf{winners}(\mathbf{B}, \mathsf{sl}, \mathsf{R}) \cap S_{\text{corr}} \neq \emptyset$) then invoke the IBE challenger oracle on identity sl obtaining $\mathsf{sk}_{\mathsf{sl}}$ and broadcast $\mathsf{ct}_{\mathsf{sl}}^{\mathsf{sk}} \leftarrow \mathsf{Enc}_{\mathsf{ECW}}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}_{\mathsf{sl}})$.
- *if a corrupted party has not won* the role for slot sl then broadcast the encryption of a dummy plaintext $\mathsf{ct}_{\mathsf{sl}}^{\mathsf{sk}} \leftarrow \mathsf{Enc}_{\mathsf{ECW}}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \vec{0})$ where $\vec{0}$ is a string of zeros of the appropriate length.

The intuitive reason for separating the two cases is that we want to query the same slots that $\mathcal{A}_{EtF}^{\text{no-thresh}}$ wins and no more. In particular we do not want to query the challenge slot sl^* (defined next). Notice, in fact, that only the slots for which the adversary has a corrupted winner will be asked to the IBE key-generation oracle. At the end of this stage, $\mathcal{A}_{EtF}^{\text{no-thresh}}$ will return $(\mathbf{B}, \mathsf{sl}^*, \mathsf{R}, m_0, m_1)$ and $\mathcal{A}_{\text{IBE}}$ will forward $((\mathsf{sl}^*, \mathsf{R}), m_0, m_1)$ to the IBE challenger.

- After receiving a ciphertext ct^* from the IBE challenger, $\mathcal{A}_{\text{IBE}}$ forwards it to $\mathcal{A}_{EtF}^{\text{no-thresh}}$. Then $\mathcal{A}_{\text{IBE}}$ simulates the execution of the blockchain as described above. At the end of the execution $\mathcal{A}_{EtF}^{\text{no-thresh}}$ outputs a guessing bit b^* which $\mathcal{A}_{\text{IBE}}$ forwards to the IBE challenger.

[9] Notice that we can check this for both types of parties as discussed in Section 2.1.

We claim that the advantage of $\mathcal{A}_{\text{IBE}}$ in the IBE experiment is negligibly close to that of $\mathcal{A}_{EtF}^{\text{no-thresh}}$ in the EtF non-threshold experiment (the one without threshold sharing). With that goal in mind, we first show that the inputs we feed to $\mathcal{A}_{EtF}^{\text{no-thresh}}$ and the blockchain execution emulated by $\mathcal{A}_{\text{IBE}}$ is indistinguishable from that in the EtF experiment. Notice that the only difference in the distributions is in the ciphertexts for the non-corrupted winners. If we could distinguish between the two cases, then we could break security of the ECW scheme. Therefore the views of $\mathcal{A}_{EtF}^{\text{no-thresh}}$ in the two cases is indistinguishable. Finally, we lower-bound the success probability of $\mathcal{A}_{\text{IBE}}$. Intuitively, we can observe that two adversaries return the same experiment bit. The only aspect that could impair $\mathcal{A}_{\text{IBE}}$'s success probability compared to $\mathcal{A}_{EtF}^{\text{no-thresh}}$'s is the possibility of having asked the IBE key-generation oracle for the challenge slot sl^*. We observe this does not affect the success probability of $\mathcal{A}_{\text{IBE}}$. Formal details are in Section 7.2.

2. Security of threshold construction from non-threshold case. The argument above had a simplified setting where we abstracted out all the threshold aspects of the protocol. This includes the committee holding shares of the master secret key and dealing shares of the slot-specific secret key. We now prove security for the actual threshold scenario (Section 7.1) building an adversary for our actual (threshold) construction using the adversary for the non-threshold construction (Fig. 7).

The threshold adversary $\mathcal{A}_{EtF}^{\text{thresh}}$ needs to emulate the setting for the other adversary where there is a single ciphertexts containing the slot-specific secret key (instead of several containing their shares). It works as follows. First, it corrupts the same parties as $\mathcal{A}_{EtF}^{\text{no-thresh}}$ and executes a blockchain as $\mathcal{A}_{EtF}^{\text{no-thresh}}$ does and broadcasting the same messages it does, with one exception which we now describe. The views of two adversaries (threshold vs non-threshold) differ in only one respect—and so do the two respective blockchains executions. The view of the threshold execution contains ciphertexts of this type for each winning slot sl (we use bracket notation for shares for readability): $\left(\left(\mathsf{ct}_{\mathsf{sl}}^{\text{hon}}[j]\right)_{j\notin S_{\text{corr}}}, \left(\mathsf{ct}_{\mathsf{sl}}^{\text{cor}}[j]\right)_{j\in S_{\text{corr}}}\right)$ These contain the shares for the slot-specific slot sl. The view for the non-threshold execution instead contains a single ciphertext with slot-specific secret key. For a honest slot not corrupted by the adversary, we denote it by $\hat{\mathsf{ct}}_{\mathsf{sl}}^{\text{hon}}$, otherwise we denote it by $\hat{\mathsf{ct}}_{\mathsf{sl}}^{\text{cor}}$. During the blockchain execution $\mathcal{A}_{EtF}^{\text{no-thresh}}$ will expect to see some ciphertext ($\hat{\mathsf{ct}}_{\mathsf{sl}}^{\text{hon}}/\hat{\mathsf{ct}}_{\mathsf{sl}}^{\text{cor}}$) whenever a slot is won, which corresponds to a query of $\mathcal{O}_{\mathsf{msk}}^{\text{k-provider}}$. The threshold adversary $\mathcal{A}_{EtF}^{\text{thresh}}$ can emulate this as follows. For every query to $\mathcal{O}_{\mathsf{msk}}^{\text{k-provider}}$:

- *if the slot is won by a honest party*, then broadcast $\hat{\mathsf{ct}}_{\mathsf{sl}}^{\text{hon}} \leftarrow \mathsf{Enc}_{\mathsf{ECW}}(\mathbf{B}, \mathsf{sl}, \vec{0})$ for a vector of zeros of the appropriate length.
- *if the slot is won by a corrupted party*, then its view will contain $\left(\mathsf{ct}_{\mathsf{sl}}^{\text{cor}}[j]\right)_{j\in[n]}$. It can then decrypt them, combine the obtained shares into a slot key $\mathsf{sk}_{\mathsf{sl}}$ and broadcast $\hat{\mathsf{ct}}_{\mathsf{sl}}^{\text{cor}} \leftarrow \mathsf{Enc}_{\mathsf{ECW}}(\mathbf{B}, \mathsf{sl}, \mathsf{sk}_{\mathsf{sl}})$

After receiving challenge messages from $\mathcal{A}_{EtF}^{\text{no-thresh}}$, adversary $\mathcal{A}_{EtF}^{\text{thresh}}$ simply forwards them to its challenger, then continues the execution as above. Finally it outputs the same output guess as $\mathcal{A}_{EtF}^{\text{no-thresh}}$.

We now claim that a successful non-threshold adversary $\mathcal{A}_{EtF}^{\text{no-thresh}}$ for the construction in *Fig.* 7 would allow $\mathcal{A}_{EtF}^{\text{thresh}}$ to have a similar advantage (up to negligible additive factors). We proceed by a standard hybrid argument. We define the first hybrid H_0 as the output of running the $\mathcal{A}_{EtF}^{\text{thresh}}$ adversary as just described. The "terminal" hybrid H_6 is defined as the output of running the $\mathcal{A}_{EtF}^{\text{no-thresh}}$ adversary. The intermediate hybrids are as follows.

- H_1: like H_0 except that we change one step in how $\mathcal{A}_{EtF}^{\text{thresh}}$ emulates $\mathcal{O}_{\mathsf{msk}}^{\text{k-provider}}$. Specifically, for the case of the honest parties, we now run $(\mathsf{sk}_i^{\mathsf{ID}})_{i\in[n]} \leftarrow \mathsf{Sim}_{\text{kg}}(\mathsf{mpk}, (\mathsf{msk}_i)_{i\in S_{\text{corr}}}, \mathsf{sl})$ to simulate the shares of the honest parties. This simulator exists by key-generation simulation of the threshold IBE scheme. We can then combine all shares to obtain a slot-specific key, encrypt it through ECW and then broadcast the encryption $\hat{\mathsf{ct}}_{\mathsf{sl}}^{\text{hon}}$. We have that $H_0 \approx H_1$ because of the security of ECW, since otherwise we would be able to distinguish encryptions of zeros from encryptions of the (combination of) the simulated slot-specific key shares.
- H_2: as previous item but now, instead of the actual secret shares, we give $\mathcal{A}_{EtF}^{\text{thresh}}$ produced by $\mathsf{Sim}_{\mathsf{msk}}$, the simulator from master secret key shares simulation of the threshold IBE scheme. $H_1 \approx H_2$ follows by the same property.
- H_3: like the previous hybrid, but now we replace the blockchain execution from H_2 with one where we do not use the shares to produce $\hat{\mathsf{ct}}_{\mathsf{sl}}^{\text{hon}}$ and $\hat{\mathsf{ct}}_{\mathsf{sl}}^{\text{cor}}$. Instead we move to a blockchain execution as in Fig. 7 with the difference that $\mathcal{O}_{\mathsf{msk}}^{\text{k-provider}}$ has a master secret key computed as follows. Let msk be the master secret key obtained by combining the (simulated) shares msk_i. Then we just run $\mathcal{O}_{\mathsf{msk}}^{\text{k-provider}}$ with this master secret key every time we need to provide a ciphertext for a new winning slot. We have $H_2 \approx H_3$ by definition of $\mathcal{O}_{\mathsf{msk}}^{\text{k-provider}}$, by correctness of the underlying homomorphic secret sharing scheme and the simulation of key-generation evaluations of IBE.
- H_4: as before but we now define msk not as the combination of the shares, but as the output of $\mathsf{Sim}_{\mathsf{msk}}$ on the master public key and the corruption set. $H_3 \approx H_4$ follows by simulation of the master secret-key property of the threshold IBE.
- H_5: Like previous item but now we do not use the key-generation simulator and instead apply the key-generation of the IBE before providing a ciphertext. $H_4 \approx H_5$ again follows by the key-generation simulation of the threshold IBE scheme. Also this is the same as H_6 by construction.

Bounding the Advantage of $\mathcal{A}_{\text{IBE}}$ in Proof of Theorem 5. Here we formally claim that the advantage of $\mathcal{A}_{\text{IBE}}$ in the IBE experiment is negligibly close to that of $\mathcal{A}_{EtF}^{\text{no-thresh}}$ in the EtF non-threshold experiment:

$$\begin{aligned}\Pr[\mathsf{WinIBE}] &\geq \Pr[\neg\mathsf{QryClgSlot} \ \wedge \ \mathsf{WinEtFHyb}] \\ &= (1 - \Pr[\mathsf{QryClgSlot} \mid \mathsf{WinEtFHyb}]) \cdot \Pr[\mathsf{WinEtFHyb}] \\ &\approx (1 - \Pr[S_{\mathrm{corr}} \cap \mathsf{winners}(\mathsf{sl}^*) \neq \emptyset \mid \mathsf{WinEtFHyb}]) \cdot \Pr[\mathsf{WinEtFHyb}] \\ &= \Pr[\mathsf{WinEtFHyb}]\end{aligned}$$

Above the $\mathsf{QryClgSlot}$ is the event where $\mathcal{A}_{\mathrm{IBE}}$ queries the challenge slot in the IBE experiment; WinIBE is the event where $\mathcal{A}_{\mathrm{IBE}}$ wins in the IBE experiment; $\mathsf{WinEtFHyb}$ is the event where $\mathcal{A}_{EtF}^{\text{no-thresh}}$ wins in the EtF experiment against the non-threshold hybrid model (Fig. 7). The first inequality follows by construction of $\mathcal{A}_{\mathrm{IBE}}$. The following ones follow from elementary probability theory and from observing that $\mathcal{A}_{\mathrm{IBE}}$ could query the challenge slot only if that was among the corrupted set (but this does not occur condition on the success of $\mathcal{A}_{EtF}^{\text{no-thresh}}$ by the definition of EtF security). □

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 12 (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 now show how to construct BWE from EtF. Let $(\mathsf{Enc}_{\mathsf{EtF}}, \mathsf{Dec}_{\mathsf{EtF}})$ be an EtF scheme. Assume a blockchain with Turing complete smart contracts which can be programmed to send, receive, and reject stake. Assume furthermore that if a constant fraction of the stake is moved to an account then within a polynomial number of slots it will begin winning the lottery with constant probability.

We assume that the contract C of an account is hardcoded into the account when created and cannot be changed. We also need to assume that the blockchain reaches all slot numbers such that there is an independent chance to win at all slot numbers. We also need that only polynomially many slot numbers are reached in polynomial time. We need that the lottery can be filtered such that only certain accounts can win a given role. We need that the filtering can depend on the smart contract put on the account when the account was created.

The construction needs a notion of labelled simulation-sound NIZK proof of knowledge. For such a scheme there is a label connected to a proof and a proof

of instance $\mathtt{x}$ and label L cannot be mauled into a proof of instance $\mathtt{x}$ and label $L' \neq L$. This can generically be constructed from an unlabelled scheme simply by letting the label be part of the instance. Let pv of the BWE scheme be the CRS of the NIZK and let td be the extraction trapdoor of the BWE scheme.

To encrypt proceed as follows.

1. Create a fresh account vk with a smart contract E and with no stake on it. Program E with W hard-coded and such that E is willing to receive calls of the form $(\textsc{Transfer}, \pi, f, F)$ from any other smart contract D. If D has f stake available and π is a proof of knowledge of $\mathtt{w}$ such that $W(\mathtt{w}) = 1$ and with label F, then accept a transfer of f stake from D and send them to F.
2. Let filter be the filter which only accepts accounts which have no stake initially and which have smart contracts C of the form that it will only accept stake from the account vk created by the encryptor above.
3. Use $\mathsf{Enc}_{\mathsf{EtF}}$ to encrypt to roles E at slots $2^i + j$ for $i = 1, \ldots, \kappa$ and $j = 1, \ldots, \kappa$. Use the filter filter.

To decrypt create a new account F with a contract accepted by filter. Then use $\mathtt{w}$ to transfer stake to F via E. Note that F is allowed to win the lotteries used in the EtF encryptions. No matter when the decryption is performed, the slots of the blockchain will eventually reach the next slot of the form 2^i as at most polynomially many slots were reached already. After this comes κ slots in a row to which the encryptor encrypted using EtF. Each of these is won with a constant probability. Therefore the probability of not decrypting is negligible.

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

AMPR14. Arash Afshar, Payman Mohassel, Benny Pinkas, and Ben Riva. Non-interactive secure computation based on cut-and-choose. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 387–404. Springer, Heidelberg, May 2014.

BBH06. Dan Boneh, Xavier Boyen, and Shai Halevi. Chosen ciphertext secure public key threshold encryption without random oracles. In David Pointcheval, editor, *CT-RSA 2006*, volume 3860 of *LNCS*, pages 226–243. Springer, Heidelberg, February 2006.

BF01. Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer, Heidelberg, August 2001.

BGG+20. Fabrice Benhamouda, Craig Gentry, Sergey Gorbunov, Shai Halevi, Hugo Krawczyk, Chengyu Lin, Tal Rabin, and Leonid Reyzin. Can a public blockchain keep a secret? In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part I*, volume 12550 of *LNCS*, pages 260–290. Springer, Heidelberg, November 2020.

BGI+18. Elette Boyle, Niv Gilboa, Yuval Ishai, Huijia Lin, and Stefano Tessaro. Foundations of homomorphic secret sharing. In Anna R. Karlin, editor, *ITCS 2018*, volume 94, pages 21:1–21:21. LIPIcs, January 2018.

BHR12. Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *ACM CCS 2012*, pages 784–796. ACM Press, October 2012.

BIOW20. Ohad Barta, Yuval Ishai, Rafail Ostrovsky, and David J. Wu. On succinct arguments and witness encryption from groups. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part I*, volume 12170 of *LNCS*, pages 776–806. Springer, Heidelberg, August 2020.

BL20. Fabrice Benhamouda and Huijia Lin. Mr NISC: Multiparty reusable non-interactive secure computation. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part II*, volume 12551 of *LNCS*, pages 349–378. Springer, Heidelberg, November 2020.

CL06. Melissa Chase and Anna Lysyanskaya. On signatures of knowledge. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 78–96. Springer, Heidelberg, August 2006.

CS02. Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 45–64. Springer, Heidelberg, April / May 2002.

DGKR18. Bernardo David, Peter Gazi, Aggelos Kiayias, and Alexander Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 66–98. Springer, Heidelberg, April / May 2018.

DS15. David Derler and Daniel 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.

FO99. Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael J. Wiener, editor,

CRYPTO'99, volume 1666 of *LNCS*, pages 537–554. Springer, Heidelberg, August 1999.

GG17. Rishab Goyal and Vipul Goyal. Overcoming cryptographic impossibility results using blockchains. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 529–561. Springer, Heidelberg, November 2017.

GGH13a. Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 1–17. Springer, Heidelberg, May 2013.

GGH+13b. Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th FOCS*, pages 40–49. IEEE Computer Society Press, October 2013.

GGJ+15. Juan A. Garay, Ran Gelles, David S. Johnson, Aggelos Kiayias, and Moti Yung. A little honesty goes a long way - the two-tier model for secure multiparty computation. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part I*, volume 9014 of *LNCS*, pages 134–158. Springer, Heidelberg, March 2015.

GGSW13. Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 467–476. ACM Press, June 2013.

GHK+21. Craig Gentry, Shai Halevi, Hugo Krawczyk, Bernardo Magri, Jesper Buus Nielsen, Tal Rabin, and Sophia Yakoubov. YOSO: You only speak once - secure MPC with stateless ephemeral roles. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part II*, volume 12826 of *LNCS*, pages 64–93, Virtual Event, August 2021. Springer, Heidelberg.

GHM+21. Craig Gentry, Shai Halevi, Bernardo Magri, Jesper Buus Nielsen, and Sophia Yakoubov. Random-index pir and applications. In *Theory of Cryptography Conference*, pages 32–61. Springer, 2021.

GKL15. Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 281–310. Springer, Heidelberg, April 2015.

GKM+20. Vipul Goyal, Abhiram Kothapalli, Elisaweta Masserova, Bryan Parno, and Yifan Song. Storing and retrieving secrets on a blockchain. Cryptology ePrint Archive, Report 2020/504, 2020. https://eprint.iacr.org/2020/504.

GLW14. Craig Gentry, Allison B. Lewko, and Brent Waters. Witness encryption from instance independent assumptions. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 426–443. Springer, Heidelberg, August 2014.

JKO13. Marek Jawurek, Florian Kerschbaum, and Claudio Orlandi. Zero-knowledge using garbled circuits: how to prove non-algebraic statements efficiently. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013*, pages 955–966. ACM Press, November 2013.

Nie03. Jesper Buus Nielsen. *On protocol security in the cryptographic model*. Citeseer, 2003.

PSs17. Rafael Pass, Lior Seeman, and abhi shelat. Analysis of the blockchain protocol in asynchronous networks. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 643–673. Springer, Heidelberg, April / May 2017.

Sah99. Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th FOCS*, pages 543–553. IEEE Computer Society Press, October 1999.

ZRE15. Samee Zahur, Mike Rosulek, and David Evans. Two halves make a whole - reducing data transfer in garbled circuits using half gates. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 220–250. Springer, Heidelberg, April 2015.

Supplementary Material

A Further Preliminaries

A.1 Proof-of-Stake (PoS) Blockchains

In this section, we give an overview of the framework from [GG17] for arguing about PoS blockchain protocol security.

Blockchain Protocol Execution. Let the blockchain protocol $\Gamma^V = (\mathsf{UpdateState}^V, \mathsf{GetRecords}, \mathsf{Broadcast})$ be guarded by a validity predicate V. The algorithms can be described as follows:

- $\mathsf{UpdateState}(1^\lambda) \to \mathsf{bst}$ where bst is the local state of the blockchain along with metadata.
- $\mathsf{GetRecords}(1^\lambda, \mathsf{bst}) \to \mathbf{B}$ outputs the longest sequence $\mathbf{B}$ of valid blocks (wrt. V).
- $\mathsf{Broadcast}(1^\lambda, m)$ Broadcast the message m over the network to all parties executing the blockchain protocol.

An execution of a blockchain protocol Γ^V proceeds by participants running the algorithm $\mathsf{UpdateState}^V$ to get the latest blockchain state, $\mathsf{GetRecords}$ to extract the ledger data structure from a state and $\mathsf{Broadcast}$ to distribute messages which are added to the blockchain if accepted by V. An execution is orchestrated by an environment $\mathcal{Z}$ which classifies parties as either honest or corrupt. All honest parties executes $\Gamma^V(1^\lambda)$ with empty local state bst and all corrupted parties are controlled by the adversary $\mathcal{A}$ who also controls network including delivery of messages between all parties.

- In each round all honest parties receive a message m from $\mathcal{Z}$ and potentially receive incoming network messages delivered by $\mathcal{A}$. The honest parties may do computation, broadcast messages and/or update their local states.
- $\mathcal{A}$ is responsible for delivering all messages sent by honest parties to all other parties. $\mathcal{A}$ cannot modify messages from honest parties but may delay and reorder messages on the network.
- At any point $\mathcal{Z}$ can communicate with adversary $\mathcal{A}$ or use $\mathsf{GetRecords}$ to retrieve a view of the local state of any party participating in the protocol.

The result is a random variable $\mathsf{EXEC}^{\Gamma^V}(\mathcal{A}, \mathcal{Z}, 1^\lambda)$ denoting the joint view of all parties (i.e. all inputs, random coins and messages received) in the above execution. Note that the joint view of all parties fully determines the execution. We define the view of the adversary as $\mathsf{view}_{\mathcal{A}}(\mathsf{EXEC}^{\Gamma^V}(\mathcal{A}, \mathcal{Z}, 1^\lambda))$ and the view of the party P_i as $\mathsf{view}_{P_i}(\mathsf{EXEC}^{\Gamma^V}(\mathcal{A}, \mathcal{Z}, 1^\lambda))$. If it is clear from the context which execution the argument is referring to, then we just write view_i. We assume that it is possible to take a snapshot i.e. a view of the protocol after the first

r rounds have been executed. We denote that by $\mathsf{view}^r \leftarrow \mathsf{EXEC}_r^{\Gamma^V}(\mathcal{A}, \mathcal{Z}, 1^\lambda)$. Furthermore, we can resume the execution departing from this view and continue until round $\tilde{r}$ resulting in the full view including round $\tilde{r}$ denoted by $\mathsf{view}^{\tilde{r}} \leftarrow \mathsf{EXEC}^{\Gamma^V}_{(\mathsf{view}^r, \tilde{r})}(\mathcal{A}, \mathcal{Z}, 1^\lambda)$.

We let the function $\mathsf{stake}_i = \mathsf{stake}(\mathbf{B}, i)$ take as input a local blockchain $\mathbf{B}$ and a party P_i and output a number representing the stake of party P_i wrt. to blockchain $\mathbf{B}$. Let the sum of stake controlled by the adversary be $\mathsf{stake}_{\mathcal{A}}(\mathbf{B})$, the total stake held by all parties $\mathsf{stake}_{\mathsf{total}}(\mathbf{B})$ and the adversaries relative stake is $\mathsf{stake\text{-}ratio}_{\mathcal{A}}(\mathbf{B})$. We also consider the PoS-fraction $\mathsf{u\text{-}stakefrac}(\mathbf{B}, \ell)$ as the amount of unique stake whose proof is provided in the last ℓ mined blocks. More precisely, let $\mathcal{M}$ be the index i corresponding to miners P_i of the last ℓ blocks in $\mathbf{B}$ then

$$\mathsf{u\text{-}stakefrac}(\mathbf{B}, \ell) = \frac{\sum_{i \in \mathcal{M}} \mathsf{stake}(\mathbf{B}, i)}{\mathsf{stake}_{\mathsf{total}}}$$

A note on corruption. For simplicity in the above execution we restrict the environment to only allow static corruption while the execution described in [PSs17] supports adaptive corruption with erasures.

A note on admissible environments. [PSs17] specifies a set of restrictions on $\mathcal{A}$ and $\mathcal{Z}$ such that only compliant executions are considered and argues that certain security properties holds with overwhelming probability for these executions. An example of such a restriction is that $\mathcal{A}$ should deliver network messages to honest parties within Δ rounds.

Blockchain Properties. We recall that running a protocol Γ^V with appropriate restrictions on $\mathcal{A}$ and $\mathcal{Z}$ will yield certain compliant executions $\mathsf{EXEC}^{\Gamma^V}(\mathcal{A}, \mathcal{Z}, 1^\lambda)$ where some security properties will hold with overwhelming probability. An array of prior works, including [GKL15,PSs17], have converged towards a few security properties that characterizes blockchain protocols. These include *Common Prefix* or *Chain Consistency*, *Chain Quality* and *Chain Growth*. From these basic properties, a number of stronger properties were derived in [GG17]. Among them, is the *Distinguishable Forking* property which will be the main requirement when introducing the EtF scheme.

Definition 13 (Common Prefix). *Let $\kappa \in \mathbb{N}$ be the common prefix parameter. The chains $\mathbf{B}_1$, $\mathbf{B}_2$ possessed by two honest parties P_1 and P_2 in slots $\mathsf{sl}_1 < \mathsf{sl}_2$ satisfy $\mathbf{B}_1^{\lceil \kappa} \preceq \mathbf{B}_2$.*

Definition 14 (Chain Growth). *Let $\tau \in (0, 1]$, $s \in \mathbb{N}$ and let $\mathbf{B}_1$, $\mathbf{B}_2$ be as above with the additional restriction that $\mathsf{sl}_1 + s \leq \mathsf{sl}_2$. Then $\mathsf{len}(\mathbf{B}_2) - \mathsf{len}(\mathbf{B}_1) \geq \tau s$ where τ is the speed coefficient.*

Definition 15 (Chain Quality). *Let $\mu \in (0, 1]$ and $\kappa \in \mathbb{N}$. Consider any set of consecutive blocks of length at least κ from an honest party's chain $\mathbf{B}_1$. The ratio of adversarial blocks in the set is $1 - \mu$ where μ is the quality coefficient.*

Stake Contribution Property. At a high level, the sufficient stake contribution property states that after sufficiently many rounds, the total amount of proof-of-stake in mining the ℓ most recent blocks is at least β fraction of the total stake in the system.

Definition 16 (Sufficient Stake Contribution). *Let* $\mathsf{suf\text{-}stake\text{-}contr}$ *be the predicate such that* $\mathsf{suf\text{-}stake\text{-}contr}^{\ell}(\mathsf{view}, \beta) = 1$ *iff for any round* $r \geq \ell$*, and any party* i *in* view *that is honest at round* r *with blockchain* $\mathbf{B}$*, we have* $\mathsf{u\text{-}stakefrac}(\mathbf{B}, \ell) > \beta$*. A blockchain protocol* Γ *has* $(\beta(\cdot), \ell_0(\cdot))$*-sufficient stake contribution property with adversary* $\mathcal{A}$ *in environment* $\mathcal{Z}$*, if there is a negligible function* $\mathrm{negl}(\cdot)$ *such that for any* $\lambda \in \mathbb{N}$*,* $\ell \geq \ell_0$*, it holds that*

$$\Pr\left[\mathsf{suf\text{-}stake\text{-}contr}^{\ell}(\mathsf{view}, \beta(\lambda)) = 1 \,\middle|\, \mathsf{view} \leftarrow \mathsf{EXEC}^{\Gamma}(\mathcal{A}, \mathcal{Z}, 1^{\lambda})\right] \geq 1 - \mathrm{negl}(\lambda)$$

Bounded Forking Property. Roughly speaking, the bounded forking property requires that no efficient adversary can create a sufficiently long fork so that its total amount of proof of stake is higher than a certain threshold. In more detail, it states that for property parameters α, ℓ_1, ℓ_2, the proof-of-stake fraction in the last ℓ_2 blocks in any adversarially created fork of length at least $\ell_1 + \ell_2$ should not be more than α.

Definition 17 (Bounded Stake Forking). *Let* $\mathsf{bd\text{-}stake\text{-}fork}$ *be the predicate such that* $\mathsf{bd\text{-}stake\text{-}fork}^{(\ell_1,\ell_2)}(\mathsf{view}, \alpha) = 1$ *iff for any round* $r \geq \tilde{r}$*, and any pair of parties* i, j *in* view *such that* i *is honest at round* r *with blockchain* $\mathbf{B}$ *and* j *is corrupt in round* $\tilde{r}$ *with blockchain* $\tilde{\mathbf{B}}$*, if there exists* $\ell' \geq \ell_1 + \ell_2$ *such that* $\tilde{\mathbf{B}}^{\lceil \ell'} \preceq \mathbf{B}$ *and for all* $\tilde{\ell} < \ell'$*,* $\tilde{\mathbf{B}}^{\lceil \tilde{\ell}} \not\preceq \mathbf{B}$*, then* $\mathsf{u\text{-}stakefrac}(\tilde{\mathbf{B}}, \ell' - \ell_1) \leq \alpha$*.*
A blockchain protocol Γ *has* $(\alpha(\cdot), \ell_1(\cdot), \ell_2(\cdot))$*-bounded stake forking property with adversary* $\mathcal{A}$ *in environment* $\mathcal{Z}$*, if there exists negligible functions* $\mathrm{negl}(\cdot)$ *and* $\delta(\cdot)$ *such that for any* $\lambda \in \mathbb{N}$*,* $\ell \geq \ell_1(\lambda)$*,* $\tilde{\ell} \geq \ell_2(\lambda)$*, it holds that*

$$\Pr\left[\mathsf{bd\text{-}stake\text{-}fork}^{(\ell,\tilde{\ell})}(\mathsf{view}, \alpha(\lambda) + \delta(\lambda)) = 1 \,\middle|\, \mathsf{view} \leftarrow \mathsf{EXEC}^{\Gamma}(\mathcal{A}, \mathcal{Z}, 1^{\lambda})\right] \geq 1 - \mathrm{negl}(\lambda)$$

Definition 18 (Distinguishable Forking). *A blockchain protocol* Γ *satisfies* $(\alpha(\cdot), \beta(\cdot), \ell_1(\cdot), \ell_2(\cdot))$*-distinguishable forking property with adversary* $\mathcal{A}$ *in environment* $\mathcal{Z}$*, if there exists negligible functions* $\mathrm{negl}(\cdot)$*,* $\delta(\cdot)$ *such that for every* $\lambda \in \mathbb{N}, \ell \geq \ell_1(\lambda), \tilde{\ell} \geq \ell_2(\lambda)$ *it holds that*

$$\Pr\left[\begin{array}{c} \alpha(\lambda) + \delta(\lambda) < \beta(\lambda) \wedge \\ \mathsf{suf\text{-}stake\text{-}contr}^{\tilde{\ell}}(\mathsf{view}, \beta(\lambda)) = 1 \wedge \\ \mathsf{bd\text{-}stake\text{-}fork}^{(\ell,\tilde{\ell})}(\mathsf{view}, \alpha(\lambda) + \delta(\lambda)) = 1 \end{array} \,\middle|\, \mathsf{view} \leftarrow \mathsf{EXEC}^{\Gamma}(\mathcal{A}, \mathcal{Z}, 1^{\lambda})\right] \geq 1 - \mathrm{negl}(\lambda)$$

A.2 (Threshold) Identity Based Encryption

We recall the definition of an identity-based encryption (IBE) scheme [BF01].

IBE An IBE scheme Π_{IBE} consists of the following algorithms:

$\mathsf{Setup}(1^\lambda)$**.** The setup algorithm takes as input a security parameter λ and returns a master key msk together with some publicly known system parameters sp including a master public key mpk, message space $\mathcal{M}$ and ciphertext space $\mathcal{C}$. We assume that all algorithms takes sp as input implicitly.

$\mathsf{IDKeygen}(\mathsf{msk}, \mathsf{ID})$**.** The identity key-generation algorithm takes as input msk and an identity $\mathsf{ID} \in \{0,1\}^*$, and returns a decryption key $\mathsf{sk}_{\mathsf{ID}}$ for ID.

$\mathsf{Enc}(\mathsf{ID}, m)$**.** The encryption algorithm takes as input an identity string $\mathsf{ID} \in \{0,1\}^*$ and $m \in \mathcal{M}$. It returns a ciphertext $ct \in \mathcal{C}$.

$\mathsf{Dec}(ct, \mathsf{sk}_{\mathsf{ID}})$**.** The decryption algorithm takes as input $ct \in \mathcal{C}$ and a decryption key $\mathsf{sk}_{\mathsf{ID}}$. It returns $m \in \mathcal{M}$.

Correctness. An IBE scheme Π_{IBE} should satisfy the standard correctness property, namely for $\mathsf{sk}_{\mathsf{ID}} \leftarrow \mathsf{IDKeygen}(\mathsf{msk}, \mathsf{ID})$ and for any $m \in \mathcal{M}$, we must have:

$$\mathsf{Dec}(\mathsf{Enc}(\mathsf{ID}, m), \mathsf{sk}_{\mathsf{ID}}) = m.$$

where $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$

Security. We use adaptive-identity security [BF01]. After the challenger runs the setup algorithm, the adversary has access to an oracle $\mathcal{O}_{\mathsf{msk}}$ that on input any id, returns sk_{id}. $\mathcal{A}$ may query the oracle on arbitrary identities of its choice even before selecting the messages m_0, m_1. More formally, we say that Π_{IBE} is secure if any PPT adversary $\mathcal{A}$ has only negligibly greater than $1/2$ probability of correctly guessing the bit b in the following game:

1. The challenger runs Setup and outputs sp to $\mathcal{A}$.
2. $\mathcal{A}$ may query the oracle $\mathcal{O}_{\mathsf{msk}}$ that on any input id returns sk_{id}.
3. $\mathcal{A}$ outputs a target identity id^* and two equal-size messages $m_0, m_1 \in \mathcal{M}$.
4. The challenger selects a random bit b and outputs $c^* \leftarrow \mathsf{Enc}(id^*, m_b)$ to $\mathcal{A}$.
5. $\mathcal{A}$ may continue to query $\mathcal{O}_{\mathsf{msk}}$ on any input $id \neq id^*$.
6. $\mathcal{A}$ outputs b'.

where $\mathcal{O}_{\mathsf{msk}}(\mathsf{ID})$ outputs $\mathsf{IDKeygen}(\mathsf{msk}, \mathsf{ID})$.

Constructing TIBE from IBE and Homomorphic Secret Sharing. Assume a secure $\mathsf{IBE} = (\mathsf{Setup}, \mathsf{IDKeygen}, \mathsf{Enc}, \mathsf{Dec})$. 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. It has the following syntax (which we specialize for the IBE setting):

- $\mathsf{Share}(\mathsf{msk}, k, n) \to (\mathsf{msk}_1, \ldots, \mathsf{msk}_n)$ shares the secret.
- $\mathsf{EvalShare}(\mathsf{msk}_i, f) \to y_i$ obtains a share for $f(\mathsf{msk})$ where f is a function.
- $\mathsf{Combine}((y_i)_{i \in T}) \to y^*$ where T is a set with size above threshold.

We assume all the algorithms above take as input the master public-key for simplicity. 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.

The construction for threshold IBE is now straightforward:

- at setup time, we produce shares $\mathsf{msk}_1, \ldots, \mathsf{msk}_n$ of the master secret key using the Share algorithm on the master secret key output of Setup.
- encryption is syntactically and functionally the same in both cases.
- to produce a partial secret-key for a certain id, we just run $\mathsf{sk}_i^{\mathsf{ID}} \leftarrow \mathsf{EvalShare}(\mathsf{msk}_i, \mathsf{IBE.IDKeygen}(\mathsf{mpk}, \cdot, \mathsf{ID}))$.
- for decryption, given enough shares for an ID ID, we run on them algorithm $\mathsf{Combine}$ to obtain $\mathsf{sk}_{\mathsf{ID}}$; we then simply run $\mathsf{IBE.Dec}$.

Threshold IBE security. If the homomorphic secret sharing supports up to a threshold k, then we obtain analogous properties for the threshold IBE construction. In particular the threshold IBE satisfies the following simulation properties for any n and threshold k supported by the homomorphic secret sharing scheme[10].

Master secret-key share simulation. For any PPT adversary $\mathcal{A}$ there exists a simulator $\mathsf{Sim}_{\mathsf{msk}}$ such that the following two distributions are indistinguishable.

$$\begin{aligned}
&\{(\mathsf{mpk}, (\mathsf{msk}_i)_{i\in S_{\mathrm{corr}}}) : (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{IBE.Setup}(1^\lambda);\\
&\qquad\qquad S_{\mathrm{corr}} \leftarrow \mathcal{A}(\mathsf{mpk}); (\mathsf{msk}_i)_{i\in n} \leftarrow \mathsf{Share}(\mathsf{msk}, n, k)\} \approx\\
&\{(\mathsf{mpk}, (\mathsf{msk}_i)_{i\in S_{\mathrm{corr}}}) : (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{IBE.Setup}(1^\lambda);\\
&\qquad\qquad S_{\mathrm{corr}} \leftarrow \mathcal{A}(\mathsf{mpk}); (\mathsf{msk}_i)_{i\in S_{\mathrm{corr}}} \leftarrow \mathsf{Sim}_{\mathsf{msk}}(\mathsf{mpk}, S_{\mathrm{corr}}, n, k)\}
\end{aligned}$$

Key-generation simulation. For any PPT adversary there exists a simulator $\mathsf{Sim}_{\mathrm{kg}}$ such that the following two distributions are indistinguishable.

$$\begin{aligned}
&\{(\mathsf{mpk}, (\mathsf{sk}_i^{\mathsf{ID}})_{i\in[n]}) : (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{IBE.Setup}(1^\lambda);\\
&\qquad\qquad S_{\mathrm{corr}} \leftarrow \mathcal{A}(\mathsf{mpk}); (\mathsf{msk}_i)_{i\in n} \leftarrow \mathsf{Share}(\mathsf{msk}, n, k);\\
&\qquad\qquad \mathsf{ID} \leftarrow \mathcal{A}(\mathsf{mpk}, (\mathsf{msk}_i)_{i\in S_{\mathrm{corr}}});\\
&\qquad\qquad \mathsf{sk}_i^{\mathsf{ID}} \leftarrow \mathsf{EvalShare}(\mathsf{msk}_i, \mathsf{ID}) \text{ for } i \in [n]\} \approx\\
&\{(\mathsf{mpk}, (\mathsf{sk}_i^{\mathsf{ID}})_{i\in[n]}) : (\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{IBE.Setup}(1^\lambda);\\
&\qquad\qquad S_{\mathrm{corr}} \leftarrow \mathcal{A}(\mathsf{mpk}); (\mathsf{msk}_i)_{i\in n} \leftarrow \mathsf{Share}(\mathsf{msk}, n, k);\\
&\qquad\qquad \mathsf{ID} \leftarrow \mathcal{A}(\mathsf{mpk}, (\mathsf{msk}_i)_{i\in S_{\mathrm{corr}}});\\
&\qquad\qquad (\mathsf{sk}_i^{\mathsf{ID}})_{i\in[n]} \leftarrow \mathsf{Sim}_{\mathrm{kg}}(\mathsf{mpk}, (\mathsf{msk}_i)_{i\in S_{\mathrm{corr}}}, \mathsf{ID})\}
\end{aligned}$$

[10] The security of this type of construction is proven for example in [Nie03] to which we defer the reader for details.

Robustness of TIBE. We assume a *robust* threshold IBE scheme, where we can verify that each of the ID-specific shares are authenticated, i.e. they have been produced by a party with the related master secret key share. This property can be obtained by assuming an underlying secret sharing scheme which is itself robust. This in turn can be obtained by attaching a NIZK or a homomorphic signature to the share.

TIBE with Proactive Secret Sharing. We assume our TIBE to allow for the shares of the master secret keys to be reshared among the committee members which evolve through time. With this goal in mind we can consider a proactive secret sharing scheme which includes a *handover* (each committee member can reshare its share) and *reconstruction* stage (committee members in a new epoch can reconstruct their secret from the output of the handover). We can directly extend a TIBE with such syntax. The resulting scheme should provide the same simulation properties as the ones described above for the non proactive case.

A.3 Smooth Projective Hash Function (SPHF)

Let $\mathcal{L}_{\mathtt{lpar}}$ be a NP language, parametrized by a language parameter $\mathtt{lpar}$, and $\mathcal{R}_{\mathtt{lpar}} \subseteq \mathcal{X}_{\mathtt{lpar}}$ be its corresponding relation. A Smooth projective hash functions (SPHFs, [CS02]) for $\mathcal{L}_{\mathtt{lpar}}$ is a cryptographic primitive with this property that given $\mathtt{lpar}$ and a statement $\mathtt{x}$, one can compute a hash of $\mathtt{x}$ in two different ways: either by using a projection key hp and $(\mathtt{x}, \mathtt{w}) \in \mathcal{R}_{\mathtt{lpar}}$ as $\mathsf{pH} \leftarrow \mathsf{projhash}(\mathtt{lpar}; \mathsf{hp}, \mathtt{x}, \mathtt{w})$, or by using a hashing key hk and $\mathtt{x} \in \mathcal{X}_{\mathtt{lpar}}$ as $\mathsf{H} \leftarrow \mathsf{hash}(\mathtt{lpar}; \mathsf{hk}, \mathtt{x})$.

Definition 19. *A SPHF for* $\{\mathcal{L}_{\mathtt{lpar}}\}_{\mathtt{lpar}}$ *is a tuple of PPT algorithms* $(\mathsf{setup}, \mathsf{hashkg}, \mathsf{projkg}, \mathsf{hash}, \mathsf{projhash})$*, which are defined as follows:*

$\mathsf{setup}(1^\lambda)$**:** *Takes in a security parameter* λ *and generates the global parameters* pp *together with the language parameters* $\mathtt{lpar}$*. We assume that all algorithms have access to* pp*.*

$\mathsf{hashkg}(\mathtt{lpar})$**:** *Takes in a language parameter* $\mathtt{lpar}$ *and outputs a hashing key* hk*.*

$\mathsf{projkg}(\mathtt{lpar}; \mathsf{hk}, \mathtt{x})$**:** *Takes in a hashing key* hk*,* $\mathtt{lpar}$*, and a statement* $\mathtt{x}$ *and outputs a projection key* hp*, possibly depending on* $\mathtt{x}$*.*

$\mathsf{hash}(\mathtt{lpar}; \mathsf{hk}, \mathtt{x})$**:** *Takes in a hashing key* hk*,* $\mathtt{lpar}$*, and a statement* $\mathtt{x}$ *and outputs a hash value* H*.*

$\mathsf{projhash}(\mathtt{lpar}; \mathsf{hp}, \mathtt{x}, \mathtt{w})$**:** *Takes in a projection key* hp*,* $\mathtt{lpar}$*, a statement* $\mathtt{x}$*, and a witness* $\mathtt{w}$ *for* $\mathtt{x} \in \mathcal{L}_{\mathtt{lpar}}$ *and outputs a hash value* pH*.*

A SPHF has to fulfill two properties:

Correctness. For all $\mathtt{x} \in \mathcal{L}_{\mathtt{lpar}}$ and their corresponding witnesses $\mathtt{w}$, we have that $\mathsf{hash}(\mathtt{lpar}; \mathsf{hk}, \mathtt{x}) = \mathsf{projhash}(\mathtt{lpar}; \mathsf{hp}, \mathtt{x}, \mathtt{w})$.

Smoothness. For any $\mathtt{lpar}$ and any $\mathtt{x} \notin \mathcal{L}_{\mathtt{lpar}}$, the hash value $\mathsf{hash}(\mathtt{lpar}; \mathsf{hk}, \mathtt{x})$ is indistinguishable from a random element in the set of hash values.

B ECW from [DS15]

While general constructions of witness encryption (WE) [GGH13a,GGH+13b,GLW14] are impractical, Derler and Slamanig [DS15] puts forward a notion inspired by the standard definition of WE, but weakened by having one extra round. A standard WE scheme consists of two algorithms Enc and Dec (ignoring the setup phase), wherein a user, in a single flow, can encrypt a message m under a specific statement $\mathtt{x}$ and produce a ciphertext ct. A recipient of ct is then able to recover the message if they know a witness $\mathtt{w}$ which certifies the truth of $\mathtt{x}$. The weakened variant of WE in [DS15] is associated with a proof system $\Pi = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ and consists 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 encrypts a message m under $(\mathtt{x}, \pi)$ if $\mathsf{Verify}(\mathsf{crs}, \mathtt{x}, \pi) = 1$.

In this section, we show how to realize ECW from [DS15] and provide additional details on their constructions for the sake of completeness.

ECW with respect to Groth-Sahai NIZK Proofs. Groth-Sahai (GS) proofs work by using dual-mode commitments that are homomorphic with respect to group operations and consist of two setup algorithms. If the commitment parameters are generated by the first algorithm, one obtains perfectly binding commitments. In contrast, the second algorithm generates the parameters in a way that leads to perfectly hiding commitments. The GS protocol aims at convincing a verifier that a set of equations are satisfied by the values inside the commitments. The prover gives to the verifier a proof containing commitments to the witness together with some additional group elements. Given such a proof and based on the linearity of GS commitments, one can encrypt a message with respect to the commitments used within the proof using a *smooth projective hash function* (SPHF) for linear languages (see Appendix A.3).

Let us give more details. The commitment parameters in the perfectly binding mode [11] are group elements $[U_1], [U_2], [U_3] \in \mathbb{G}^3 \times \mathbb{G}^3 \times \mathbb{G}^3$ defined as follows:

$$[U_1] = ([\tau_1], [0], [1]) \quad ; \quad [U_2] = ([0], [\tau_2], [1])$$
$$[U_3] = \tau_3 \cdot [U_1] + \tau_4 \cdot [U_2] = ([\tau_1\tau_3], [\tau_2\tau_4], [\tau_3 + \tau_4])$$

where $\tau_1, \tau_2, \tau_3, \tau_4 \xleftarrow{\$} \mathbb{Z}_q$. Setting these parameters, a commitment to a message $m \in \mathbb{G}$ is by choosing $r_1, r_2, r_3 \xleftarrow{\$} \mathbb{Z}_q$ and computing

$$\begin{aligned}\mathsf{cm}_m &= ([0], [0], [m]) + r_1U_1 + r_2U_2 + r_3U_3 \\ &= ([\tau_1(r_1 + \tau_3 r_3)], [\tau_2(r_2 + \tau_4 r_3)], [m + r_1 + r_2 + r_3(\tau_3 + \tau_4)]).\end{aligned}$$

Now let $\mathtt{lpar} = ([U_1], [U_2], [U_3])$ and define

$$\mathcal{L}_{\mathtt{lpar}} = \{([m], \mathsf{cm}_m) | \exists (r_1, r_2, r_3) \in \mathbb{Z}_q^3 : \mathsf{cm}_m = ([0], [0], [m]) + r_1U_1 + r_2U_2 + r_3U_3\}$$

[11] Here we only focus on the perfectly binding mode. All the explanations can also be applied to the perfectly hiding mode in a straightforward manner.

It is not hard to see that $\mathcal{L}_{\mathtt{lpar}}$ is a linear language and therefore a SPHF for it can be constructed as follows:

$\mathsf{setup}(1^\lambda)$**:** Run the bilinear group generation algorithm and let pp be the bilinear group description. Also, set the language parameters $\mathtt{lpar} = ([U_1], [U_2], [U_3])$ as defined above.

$\mathsf{hashkg}(\mathtt{lpar})$**:** Choose $\alpha_1, \alpha_2, \alpha_3 \xleftarrow{\$} \mathbb{Z}_q^3$ and return $\mathsf{hk} = (\alpha_1, \alpha_2, \alpha_3)$.

$\mathsf{projkg}(\mathtt{lpar}; \mathsf{hk}, \mathtt{x})$**:** Parse $\mathtt{lpar}$ as $([U_1], [U_2], [U_3])$ and hk as $(\alpha_1, \alpha_2, \alpha_3)$. Return $\mathsf{hp} = (\gamma_1, \gamma_2, \gamma_3) \in \mathbb{G}^3$, where

$$\gamma_1 = \alpha_1[\tau_1] + [\alpha_3];\ \gamma_2 = \alpha_2[\tau_2] + [\alpha_3]$$
$$\gamma_3 = \alpha_1[\tau_1\tau_3] + \alpha_2[\tau_2\tau_4] + \alpha_3[\tau_3 + \tau_4]$$

$\mathsf{hash}(\mathtt{lpar}; \mathsf{hk}, \mathtt{x})$**:** Parse the statement $\mathtt{x}$ as $([m], \mathsf{cm}_m = ([u], [v], [e]))$, and hk as $(\alpha_1, \alpha_2, \alpha_3)$. Return H computed as $\mathsf{H} = [u] \cdot \alpha_1 + [v] \cdot \alpha_2 + ([e] - [m]) \cdot \alpha_3$.

$\mathsf{projhash}(\mathtt{lpar}; \mathsf{hp}, \mathtt{x}, \mathtt{w})$**:** Parse hp as $(\gamma_1, \gamma_2, \gamma_3)$ and $\mathtt{w}$ as (r_1, r_2, r_3). Return pH computed as $\mathsf{pH} = \gamma_1 r_1 + \gamma_2 r_2 + \gamma_3 r_3$.

We refer the reader to [DS15] for the security proof of the above SPHF.

Equipped with this construction, one can now construct an ECW by encoding the lottery statement into vectors of commitments satisfying pairing-product equations (PPEs). In more details, the construction works as follows. All receivers who have $\mathsf{sk}_{L,i}$ such that $\mathsf{lottery}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}_{L,i}) = 1$ publish a GS proof π_i that they have such secret key $\mathsf{sk}_{L,i}$. The encrypting party encodes the lottery predicate into a set of pairing-product equations (PPEs) and encrypts the message under each of these GS proofs using the SPHF scheme described above. We note that this construction can be used for any type of algebraic lottery that can be represented as a set of pairing product equation (e.g., algebraic VRF-based lotteries). Moreover, while this can be seen as a weaker variant of ECW where the (claimed) winners are required to send a proof of winning the lottery in advance and thus requires an extra round of communication, it results in a construction with significant improvement on the ciphertext size published by the encrypting party, i.e., only linear in the number of winners receiving the message.

C Further Details on YOSO MPC

C.1 Extended Lotteries

So far the probability of winning a role can depend on the slot sl, the role R, and the lottery witness $\mathsf{sk}_{L,i}$. The dependence can be arbitrarily complex. We sometimes need to assume that the lottery shows some level of structure to be able to ensure that a given set of roles has enough honest machines winning them.

Smooth Lotteries. First we define that a lottery has individual winning probabilities if for a given sl and a given account sk there exist a probability p such that it holds for all R that $\Pr[\mathsf{lottery}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}_{L,i}) = 1] \approx p$. This is the case for most PoS lotteries as the probability of winning depends only on the stake that $\mathsf{sk}_{L,i}$ has in a given slot.

We will also need to assume independence of winning events. It is useless for the sake of using, e.g., the law of large numbers if in a given slot either all honest parties win a role or none win a role. We typically needed that except with negligible probability some large enough fraction wins roles.

We require that for all sl and all $(\mathsf{R}_i, \mathsf{sk}_{L,i})$ and $(\mathsf{R}_j, \mathsf{sk}_{L,j}) \neq (\mathsf{R}, \mathsf{sk}_{L,i}$ it holds that $\Pr[\mathsf{lottery}(\mathbf{B}, \mathsf{sl}, \mathsf{R}_i, \mathsf{sk}_{L,i}) = 1 \,|\, \mathsf{lottery}(\mathbf{B}, \mathsf{sl}, \mathsf{R}_j, \mathsf{sk}_{L,j})] \approx \Pr[\mathsf{lottery}(\mathbf{B}, \mathsf{sl}, \mathsf{R}_i, \mathsf{sk}_{L,i}) = 1]$. We extend this to n-independence where the probability of an account winning a role does not depend on the outcome of $n-1$ other lotteries in the same slot.

We call a lottery with individual winning probabilities and n-independence an n-smooth lottery. For an n-smooth lottery we can compute the probability that a set of up to n roles are won by honest parties directly from the individual winning probabilities of the slot.

Hardness Adjustment. We will sometimes need to assume that the hardness of the lottery can be adjusted. This can in principle be captured in the current formalism $\mathsf{lottery}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}_{L,i})$ as we could have some roles be harder to win. This would however ruin individual winning probabilities so we prefer an explicit notation for it. We assume a new parameter $\mathsf{hard} \in [0,1]$ which can be used to control hardness of the lottery. For simplicity assume that $\mathsf{hard} \in [0,1]$. We require that

$$\Pr[\exists \mathsf{sk}\,(\mathsf{lottery}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}_{L,i}, \mathsf{hard}) = 1)] \approx \mathsf{hard}\ .$$

A more realistic model would have to assume that the probability can be controlled to be in some interval, e.g., $[\mathsf{hard}/2, 2\mathsf{hard}]$, but nothing essential is lost in assuming the simplistic model in this work where the focus is on EtF and not intricacies of the lotteries themselves. Scaling of hardness is typically easy to construct as most PoS lotteries give each party a pseudo-random number and say that the party won if the number is below some threshold. One can use hard to adjust the threshold. We assume that adjusting the hardness maintains n-smoothness.

Filtering. Another possible extension is having an extra parameter filter which is a PPT predicate filtering the lottery. Given an account $\mathsf{sk}_{L,i}$ we assume it can be computed in PPT from the information on $\mathbf{B}$ and the public parameters $\mathsf{pk}_{L,i}$ associated to $\mathsf{sk}_{L,i}$. In particular, filter does not need $\mathsf{sk}_{L,i}$ to be efficiently computable. We require that $\mathsf{filter}(\mathbf{B}, \mathsf{sk}_{L,i})$ outputs $\top$ or $\bot$. We require from the lottery that if $\mathsf{filter}(\mathbf{B}, \mathsf{sk}_{L,i}) = \bot$ then $\mathsf{lottery}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}_{L,i}, \mathsf{filter}) = 0$. If $\mathsf{filter}(\mathbf{B}, \mathsf{sk}_{L,i}) = \top$, then we require that $\mathsf{lottery}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}_{L,i}, \mathsf{filter}) =$

$\mathsf{lottery}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}_{L,i})$. Since the filter can be computed in PPT given the blockchain it is typically trivial to augment existing lotteries with a filter. We can simply let the lottery predicate include a check of the filter. We could again capture this in the existing formalism simply by letting $\mathsf{lottery}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}_{L,i})$ ignore the filtered winners, but this would again ruin individual winning probabilities of the underlying mechanism.

Honest Majority. When realizing YOSO MPC, we must show that a lottery selects parties for roles in such a way that achieve honest majority among roles, according to the corruption statuses defined in Section 6.5. For simplicity of the discussion we assume an n-smooth lottery, where we can control the hardness. Assume that we set the hardness such that a given role is won with probability ϕ. It is easy to see that when we have n-smoothness then if there is probability ϕ that a role is won, then there is about probability ϕ^2 that it is won twice, giving a MALICIOUS role. Clearly there is probability $1-\phi$ that it is not won, giving a CRASHED role. The expression $\phi^2 + 1 - \phi$ has a minimum of 75%, so 75% of all roles will be malicious or crashed even if we start with perfect honesty. We can therefore never expect to get honest majority. The trick is to design protocols which can tolerate many crashed parties as long as there are more honest parties than corrupted parties among the non-crashed parties.

Assume a lottery where a unique winner is honest with probability $\frac{1}{2} + \epsilon$. In that case the probability that a role is won by a single honest winner is $\phi(\frac{1}{2} + \epsilon)$. The probability that the role is won more than once is about $\phi/(1-\phi)$ by an application of a geometric series. The probability that it is won by a single corrupted party is $\phi(\frac{1}{2} - \epsilon)$. To have more honest parties than corrupted parties in expectation we therefore need that $\phi(\frac{1}{2} + \epsilon) > \phi/(1-\phi) + \phi(\frac{1}{2} - \epsilon)$, which solves to $\phi > 1 - 2\epsilon$. By picking $\phi = 1 - 2\epsilon + \delta$ for a positive constant we get that the expected number of honest parties is $h = \phi(\frac{1}{2} + \epsilon)n$ and that the expected number of corrupted parties is $t = (\phi/(1-\phi) + \phi(\frac{1}{2} - \epsilon))n$ and that $h - t > 2\gamma n$ for a positive constant γ.

By setting $H = h - \gamma n$ and using a Chernoff bound and n-smoothness we can pick n large enough to ensure that there are more than H honest parties except with negligible probability. By setting $T = t + \gamma n$ and picking n large enough we can similarly ensure that there are less that T corrupted parties except with negligible probability. Note that $H > T$. Hence, we can satisfy Definition 10 of Section 6.5.