Encryption to the Future

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

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

[1] Aarhus University, Denmark
{matteo,hamidreza,jbn}@cs.au.dk
[2] 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 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 a *non-interactive* 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. Inspired by 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 lift "near future" EtF to "far future" EtF with a protocol based on an auxiliary committee whose communication complexity is *independent* from the length of plaintext messages being sent to the future.

1 Introduction

Most cryptographic protocols assume that parties' identities are publicly known. This is a natural requirement, since standard secure channels are identified by a sender and a receiver. However, this status quo also makes it easy for adaptive (or 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 [2,15,17] 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 to transfer secret states from past committees to new anonymous committees freshly chosen in the future?

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 [2] and further developed in [15]. This task consists of sending a message to an abstract role R at a given point in the future. A role is just a bit-string describing an abstract role, such as $\mathsf{R} =$"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 that no one should know the identity of P_i even though P_i learns that it is chosen to act as R.

The approaches to role assignment proposed in [2,15,17] all use an underlying Proof-of-Stake (PoS) blockchain (*e.g.* [10]). 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 time-locked commitments/encryption) is not sufficient for achieving this notion, since only the party(ies) elected for a role should receive a secret message encrypted for that role, while time-lock puzzles allow every party who invests enough computing time to recover the message.

A shortcoming of the approaches of [2,15,17] 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 [2], the auxiliary committee performs cheap operations but can adversarially influence the probability distribution with which P_i is chosen. In the case of [15,17], the auxiliary committee cannot bias this probability distribution but must perform very expensive operations. Moreover, these approaches have another big caveat; they can only be used to select P_i to act as R according to a probability distribution known at the time the auxiliary committee outputs $(\mathsf{R}, \mathsf{pk}_\mathsf{R})$. Hence, they only allow for sending messages to future committees that have been recently elected.

Note that there are two distinct aspects to such a protocol. One aspect is role assignment (RA) which deals with the sending of messages to future roles of a protocol while hiding the physical machine executing the role. The other aspect is the *role execution* (RX) aspect which focuses on the execution of the specific protocol that runs on top of the RA mechanism, i.e., what messages are sent to which roles and what specification the protocol implements. In [15] the so-called *You Only Speak Once* (YOSO) model is introduced for studying RX. In the YOSO model the protocol execution is between abstract roles which can each speak only once. Later these can then be mapped to physical machines using an RA mechanism. It was shown in [15] that once you can do RA in a synchronous model, then any well-formed ideal functionality can be implemented in the YOSO model with security against malicious, adaptive corruption of a minority of machines. Concretely, [15] gives an ideal functionality for RA and shows that a YOSO protocol for abstract roles can be compiled into the RA-hybrid model to give a protocol secure against adaptive attacks.

In this paper we further investigate the RA aspect. Taking a step back from specific solutions for role assignment, we will focus on how to non-interactively *encrypt* to a future role with IND-CPA security *without* the aid of an auxiliary committee. We will 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. In particular, once our main schemes are in place, these other aspects can be added using standard techniques.

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. Before describing them in more detail, we summarize our contributions as follows:

- A definition for the notion of *non-interactive* 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.
- A proof that an EtF scheme, which allows for encryption towards parties selected at arbitrary points in the future, implies a flavor of witness encryption for NP.
- 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 that requires an auxiliary committee but enjoys a limited amount of communication complexity *independent* from plaintext message length.

– An application of ECW as a central primitive for realizing role assignment in protocols that require it (*e.g.* [2,15,17]).

Our EtF notion provides an arguably useful abstraction for the problem of transferring secret states to secret committees. In particular, 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. The protocol uses auxiliary committee's but with communication complexity which is independent of the plaintext length. We now elaborate on our results, discussing the intuition behind the notion of EtF, its constructions and its fundamental limits.

Encryption to the Future (EtF) - Section 3. As in previous works [2,15,17], an EtF scheme is defined with respect to an underlying PoS blockchain. Notice that the vast majority of PoS blockchains (*e.g.* [10]) associate a slot number to each block and have an intrinsic lottery that selects parties to generate a block according to a stake distribution (*i.e.* the probability a party is selected is proportional to the stake the party controls). These natural features are leveraged to define EtF in such a way that a message is encrypted towards a party that is selected by the underlying blockchain's lottery scheme at a given future slot. More generally, we can leverage this lottery mechanism to select parties for multiple roles associated to each slot (so that committees consisting of multiple parties can be elected for a single point in time). 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) EtF - Section 8. We show that EtF implies a version of witness encryption [14] over a blockchain (similar to that of [19]). The crux of our proof is being able to encrypt a message towards a role for which a party will only be chosen at an arbitrary point in the future. If this is possible, we show how to construct a witness encryption scheme relying on an EtF scheme and a smart contract on the EtF's underlying blockchain. We also show that BWE implies EtF. This shows that at the level of feasibility the notions are similar. It also shows that if we want to implement non-interactive EtF, we would have to use strong assumptions.

Encryption to the Current Winner (ECW) - Section 3. Given that our general notion of EtF would require very strong assumptions to achieve without interaction (use of auxilliary committees), we look towards efficient ways to construct EtF under standard assumptions while minimizing interaction. As a first step towards such a constrcution, 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 w.r.t. the lottery parameters available at the current slot when encryption is performed (as in previous results [2,15,17]).

Constructing ECW (non-interactively) - Section 5. We show that it is possible to construct a non-interactive ECW scheme (*i.e.* without using auxiliary committees which was required in previous results [2,15,17]) with security under standard assumptions. Our construction relies on a restricted flavor of witness encryption which we formalize as Witness Encryption over Commitments (cWE) in Section 4. We show in Section 3 that ECW can be constructed in a black-box manner from cWE, which in turn can be constructed from oblivious transfer and garbled circuits (as shown in Appendix C). This construction improves over previous results [2,15,17] in the sense that it does not rely on auxiliary committees.

Instantiating YOSO MPC using ECW- Section 6. We show an application of our ECW notion as a building block for the YOSO MPC protocol of [15]. In this protocol, each round of an MPC protocol is executed by a different committee of parties, who transfer secret state to a future committee that remains anonymous until it transfers its own secret state to the next committee. This application has the following main specific 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.* [10]). 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. Given the implausibility of constructing EtF non-interactively from standard assumptions, we show that we can transform an ECW scheme into an unrestricted EtF scheme when given limited access to an auxiliary committee. Namely, our solution requires communication complexity independent from plaintext length. Notice that previous works [2,15,17] require successive committees to store and reshare secret shares of every message to be sent to a party selected in the future, *i.e.* communication complexity is dependent on both the amount and length of plaintext messages. In our transformation from ECW to EtF, each role in the future is associated to a unique identity of an Identity Based Encryption scheme (IBE) and messages towards this role are encrypted using IBE. The secret key corresponding to a given role is later generated and given to the party who is selected for that role at any point in the future. In order to generate such secret keys, we use YOSO MPC instantiated from ECW as shown in Section 6. In contrast to previous schemes, the auxiliary committee executing this instance of YOSO MPC only needs to hold shares of the IBE's master secret key and perform 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 (future winner) and ECW (current winner) in Fig. 1.

Type	Scheme	Communication	Committee?	Interactive?
ECW	CABKAS	$O(1)$	yes	yes
	RPIR	$O(1)$	yes	yes
	cWE (MS-NISC)	$O(N)$	no	no*
	cWE (GC+OT)	$O(N)$	no	no*
EtF	IBE	$O(1)$	yes	yes
	Full-fledged WE	$O(1)$	no	no

Fig. 1. The column "Committee?" indicates whether a committee is required. The column "Communication" refers to whether the communication complexity grows or not with N, the number of all parties. We denote by an asterisk non-interactive solutions that require sending a first reusable message during the initial step.

Constructions of ECW. In ECW, both the stake distribution and the randomness extracted from the blockchain are static and known at the time of encryption and thus the encryption algorithm has all the parameters known except the secret key of the lottery winner. This determinism of the lottery predicate (although with unknown parameter sk) at the time of encryption makes it possible to realize ECW with several constructions.

- CABKAS [2]: In this work, they propose a scalable evolving-committee proactive secret-sharing (ECPSS) scheme that allows committees to maintain a secret over a public blockchain. This committee is dynamically changing and thus needs to reshare its secret shares to the next committee. The main challenge is how to select a small committee from a large population of parties such that everyone can send secure messages to the committee members without knowing who they are. The solution of [2] is to select the committee by having another committee called "nominating committee" who nominate members of the new committee. Using ECW, we can get rid of the nominating committee and ask the current committee to ECW encrypt their secret shares to the roles of the next committee. One can also see the nominating committee as a tool which provides ECW functionality. I.e., the main challenge in encryption to future is how to encrypt without knowing the recipient and this is exactly what the nominating committee is providing. A major caveat is that in order to guarantee an honest majority in the committees, [2] can tolerate up to a fraction of 1/4 corruption only. This is because corrupted nominators can always select corrupted parties, whereas honest nominators may select corrupted parties by chance.

- RPIR [17]: the main contribution of this paper is to solve the above challenge of selecting committees with supporting a better fraction of corrupted parties, that is $< 1/2$. To this end, the authors define a new flavour of Private Information Retrieval (PIR) called Random-index PIR (or RPIR) that allows each committee to do the nomination task by themselves and select the next committee without requiring any nominating committee. Differently from [2], we don't need a nominating committee here, but the main disadvantage of RPIR is that the constructions are quite inefficient. More specifically, it is 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 a huge 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 $O(k^3)$ with the $O(k)$ being the length of an FHE decryption share.
- MrNISC [3]: [3] defines a non-generic version of witness encryption, called "Witness Encryption for NIZK of Commitments". In their setting, parties first commit to their private inputs once and for all and, later, different players acting as the encryptor can send the ciphertext in one flow such that any party with a committed input that satisfies the relation can decrypt. The authors show how to construct this primitive based on standard assumptions in asymmetric bilinear groups. We generalize and formalize this notion as cWE and show how it can be used to construct ECW. While the original construction of MrNISC fits the definition of cWE (through which we build ECW), we observe it is an overkill for our application. 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.

Constructions of EtF. As indicated by our results, the general notion of EtF is significantly harder to realize. Here we discuss natural ideas for realizing EtF that illustrate two extremes where our approach lies in the middle:

- Non-Interactive—Using Witness Encryption [14]: One trivial approach to realize EtF is to use Witness Encryption [14] (WE) for the arithmetic relation $\mathcal{R}$ being the lottery predicate such that the party who possess a winning secret key sk can decrypt the ciphertext. However, existing witness encryption schemes [14] are impractical. In particular, all existing constructions rely on very strong assumptions such as multilinear maps and indistinguishability obfuscation. Hence, this "trivial" solution comes with a heavy cost, which is not surprising given our result showing that EtF implies a flavor of WE.
- Interactive—Multiple Committees and Successive Rounds of RPIR/CABKAS/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 reshare their shares towards members of a future anonymous committee via an invocation of RPIR/CABKAS/ECW. This is essentially the solution proposed in CABKAS [2] 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*.

Using blockchains in order to construct non-interactive primitives with game-based security has been previously considered in [18]. Other approaches for transferring secret state to future committees have been proposed in [19], although anonymity is not a concern in this setting. On the other hand, using anonymity as a means for overcoming corruption have been proposed in [12], although this work considers anonymous channels among a fixed set of parties.

2 Preliminaries

PPT stands for probabilistic polynomial time and we use λ to denote the security parameter. We write $a \overset{\$}{\leftarrow} A$ to denote that a is sampled uniformly from a set A.

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 [18] which, in turn, builds on the analysis done in [13] and [23]. We invite the reader to re-visit the abstraction used in [18]. 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 [18] it is proven that there exist PoS blockchain protocols with the properties described below, *e.g.* Ouroboros Praos [10].

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 $\mathbf{B}^{\lceil \ell}$ be the chain (sequence of blocks) $\mathbf{B}$ where the last ℓ blocks have been removed and if $\ell \leq |\mathbf{B}|$ 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 [10], we assume that the genesis block is generated by an initialization functionality $\mathcal{F}_{INIT}$ that registers all parties' keys. Moreover, we assume that primitives specified in separate functionalities in [10] as incorporated into $\mathcal{F}_{INIT}$. $\mathcal{F}_{INIT}$ 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, $\mathcal{F}_{INIT}$ 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 $\mathcal{F}_{INIT}$, it responds to requests for a genesis block by providing $B_0 = (\mathsf{Sig.pk}_0, \mathsf{aux}_0, \mathsf{stake}_0), \ldots, (\mathsf{Sig.pk}_n, \mathsf{aux}_n, \mathsf{stake}_n), \mathsf{aux}$, where aux is generated according to the underlying blockchain consensus protocol.

Since $\mathcal{F}_{INIT}$ 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 $\mathcal{F}_{INIT}$ 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 [18] 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 $\mathit{\Gamma}^V = (\mathsf{UpdateState}^V, \mathsf{GetRecords}, \mathsf{Broadcast})$ 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 if and only if $\mathbf{B} = \tilde{\mathbf{B}}$ *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 $(\mathsf{sl}, \mathsf{R}_i)$-pair may yield zero, one or multiple winners. In most cases, 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 winners $\mathcal{W}$ for a given role $(\mathsf{sl}, \mathsf{R})$. This is the set of lottery witnesses satisfying the $\mathsf{lottery}$ predicate. Therefore, we write the shorthand $\{\mathsf{sk}_{L,i}\}_{P_i \in \mathcal{W}} \leftarrow \mathsf{winners}(\mathbf{B}, \mathsf{sl}, \mathsf{R})$.

2.2 Commitment Scheme

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

- $\mathsf{Setup}(1^\lambda) \rightarrow \mathsf{ck}$ outputs a commitment key.
- $\mathsf{Commit}(\mathsf{ck}, m; \rho) \rightarrow \mathsf{cm}$ outputs a commitment given as input a message m and randomness ρ.

We require a commitment scheme to satisfy the standard properties of *binding* and *hiding*. It is (computationally) binding if it is not possible for an efficient adversary to come up with two pairs $(m, \rho), (m', \rho')$ such that $m \neq m'$ and such that $\mathsf{Commit}(\mathsf{ck}, m; \rho) = \mathsf{cm} = \mathsf{Commit}(\mathsf{ck}, m'; \rho')$ for $\mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda)$. The scheme is hiding if for any two m, m' an efficient adversary cannot distinguish between a commitment of m and one of m'.

Extractability In our construction 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 $\mathcal{E}$ such that $\mathcal{E}(\mathsf{td}, \mathsf{cm})$ outputs (m, ρ) such that $\mathsf{cm} = \mathsf{Commit}(\mathsf{ck}, m; \rho)$. We remark we can generically obtain this property by attaching to the commitment a non-interactive zero-knowledge argument of knowledge that shows knowledge of opening, i.e., for the relation $\mathcal{R}^{\mathrm{opn}}(\mathsf{cm}_i; (m, \rho)) \iff \mathsf{cm}_i = \mathsf{Commit}(\mathsf{ck}, m; \rho)$.

2.3 (Threshold) Identity Based Encryption

In an IBE scheme, users can encrypt simply with respect to an *identity* (rather than a public key). Given a master secret key, an IBE can generate secret keys that allows to open to specific identities. In our construction of EtF (Section 7.1) we rely on a *threshold variant of IBE*(TIBE) where no single party in the system holds the master secret key. Instead parties in a committee hold a partial master secret key msk_i. Like other threshold protocols, threshold IBE can be generically obtained by "lifting" an IBE through a secret sharing with homomorphic properties (see for example [22]).

Threshold IBE This is a threshold variant of IBE with the following syntax:

$\mathit{\Pi}_{\mathsf{TIBE}}.\mathsf{Setup}(1^\lambda, n, k) \to (\mathsf{sp}, \mathsf{vk}, \vec{\mathsf{msk}})$: 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.

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

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

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

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

Correctness A TIBE scheme $\mathit{\Pi}_{\mathsf{TIBE}}$ should satisfy two correctness properties:

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

Structural Property: TIBE as IBE + Secret Sharing We model threshold IBE in a modular manner from IBE and assume it to have a certain structural property: that it can be described as an IBE "lifted" through a homomorphic secret-sharing [6,4,22]. 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 where we can verify that each of the id-specific key shares are authenticated (*robustness*) and that shares of the master secret key can be reshared (*proactive resharing*).

3 Modelling EtF

In this section, we present a model for encryption to the future winner of a lottery. In order to argue about a notion of future, we use the blocks of an underlying blockchain ledger and their relative positions in the chain to specify points in time. Intuitively, our notion allows for creating ciphertexts that can only be decrypted by a party that is selected to perform a certain role R at a future slot sl according to a lottery scheme associated with a blockchain protocol. The winner of the lottery at a point in the future with respect to a blockchain state $\tilde{\mathbf{B}}$ is determined by the lottery predicate defined in Section 2.1, *i.e.* the winner is the holder of a lottery secret key sk such that $\mathsf{lottery}(\tilde{\mathbf{B}}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}) = 1$. However, notice that the winner might only be determined by a blockchain state produced in the future as a result of the blockchain protocol execution. This makes it necessary for the ciphertext to encode an initial state $\mathbf{B}$ of the blockchain that allows for verifying that a future state $\tilde{\mathbf{B}}$ (presented at the time of decryption) has indeed been produced as a result of correct protocol execution. This requirement is captured by the evolving blockchain predicate defined in Section 2.1, *i.e.* $\mathsf{evolved}(\mathbf{B}, \tilde{\mathbf{B}}) = 1$ iff $\tilde{\mathbf{B}}$ is obtained as a future state of executing the blockchain protocol departing from $\mathbf{B}$.

Definition 3 (Encryption to the Future). *A pair of PPT algorithms* $\mathcal{E} = (\mathsf{Enc}, \mathsf{Dec})$ *in the 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* $\perp$*.*

Correctness An EtF-scheme is said to be correct if for honest parties i and j, there exists a negligible function μ such that

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

Security We establish a game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$. In Section 2.1 we 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 $\mathrm{Game}^{\mathsf{IND\text{-}CPA}}_{\Gamma,\mathcal{A},\mathcal{Z},\mathcal{E}}$ 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 form b' being dependent on b.

Definition 4 ($\mathsf{IND\text{-}CPA}$ 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* $\mathsf{IND\text{-}CPA}$ *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 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 $\Gamma$ 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 $\mathsf{ct}$
$\mathsf{view}^{\tilde{r}} \leftarrow \mathsf{EXEC}^{\Gamma,\mathsf{view}^r}_{\tilde{r}}(\mathcal{A},\mathcal{Z},1^\lambda)$    ▷ Execute from $\mathsf{view}^r$ 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 $\{\mathsf{sk}^{\mathcal{A}}_{L,0},\ldots,\mathsf{sk}^{\mathcal{A}}_{L,j}\} \cap \mathsf{winners}(\tilde{\mathbf{B}},\mathsf{sl},\mathsf{R}) = \emptyset$ then    ▷ $\mathcal{A}$ does not win role $\mathsf{R}$
        return $b \oplus b'$
    end if
end if
return $\hat{b} \xleftarrow{\$} \{0,1\}$
```

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 committed inputs. This is a relaxed notion of witness encryption. In witness encryption we allow to 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 $\mathtt{w}$. 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.

We observe that this type of WE is morally weaker than standard WE because of its deterministic flavor. In cWE we require to encrypt with respect to a commitment, which intuitively binds to a witness; in standard WE we encrypt without having any "pointer" to an alleged witness.

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

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

Encryption phase. $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathtt{x}, m)$ *on input a statement* $\mathtt{x} = (\mathsf{ck}, \mathsf{cm}, C, y)$ *such that* $C \in \mathcal{C}$ *and a message* $m \in \{0,1\}^*$*, it generates a ciphertext* ct.
Decryption phase. $m/\bot \leftarrow \mathsf{Dec}(\mathsf{ct}, \mathtt{w}, \mathsf{d})$ *on input a ciphertext* ct*, a witness* $\mathtt{w}$*, and a decommitment* d*, returns a message* m *or* $\bot$.

A cWE should satisfy *completeness* and *semantic security* as defined below:

(Perfect) Completeness. An honest prover with a statement $\mathtt{x} = (\mathsf{ck}, \mathsf{cm}, C, y)$ and witness $\mathtt{w}$ such that $\mathsf{cm} = \mathsf{Commit}(\mathsf{ck}, \mathtt{w})$ and $C(\mathtt{w}) = y$ can always decrypt with overwhelming probability. More precisely, a cWE has perfect completeness if for all $\lambda \in \mathbb{N}$, for all $C \in \mathcal{C}$, and for all $\mathsf{ck} \in \mathsf{Range}(\mathsf{C}.\mathsf{Setup})$, strings $\rho, y, \mathtt{w}$ and bit messages $m \in \{0,1\}^*$ it holds that

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

where $\mathsf{cm} = \mathsf{C}.\mathsf{Commit}(\mathsf{ck}, \mathtt{w}; \rho)$ and $C(\mathtt{w}) = y$.

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 $\mathcal{A}$:

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

where $\mathcal{O}_{\mathrm{enc}}(\cdot)$ is an encryption oracle defined as follows. On input $(\mathtt{w}, \rho, C, y, m)$, it computes $\mathsf{cm} \leftarrow \mathsf{C}.\mathsf{Commit}(\mathsf{ck}, \mathtt{w}; \rho)$ and returns $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathtt{x} = (\mathsf{ck}, \mathsf{cm}, C, y), m)$[3].

[3] While such oracle is not necessary for the model (the adversary is obtaining information it could compute by itself), we explicitly model it to clarify that the adversary can have access to additional ciphertexts in the experiment.

Instantiations of cWE

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

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

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

We observe that the above construction actually yields a stronger notion of cWE where the statement $\mathtt{x}$ is private which is not a requirement in our setting. This asymmetry between sender and receiver privacy was also observed by others [21] and it opens the door for efficient constructions using oblivious transfer (OT) and privacy-free garbled circuits as described in [24]. More details on the more efficient construction of cWE using OT and garbled circuits are provided in C.2.

5 Construction of ECW

Here we show a novel construction of ECW from cWE (Section 5.1). 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 $\mathsf{cWE} = (\mathsf{Enc}_{\mathsf{cWE}}, \mathsf{Dec}_{\mathsf{cWE}})$; we assume the commitments to be extractable. The class of circuits $\mathcal{C}$ of 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 $\mathsf{Setup}(1^\lambda) \rightarrow \mathsf{ck}$ 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 $\mathsf{cm}_i \leftarrow \mathsf{Commit}(\mathsf{ck}, \mathsf{sk}_{L,i}; \rho)$ to $\mathsf{sk}_{L,i}$ using randomness ρ; 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 $\mathsf{ct}_i \leftarrow \Pi_{\mathsf{cWE}}.\mathsf{Enc}_{\mathsf{cWE}}(\mathtt{x}_i = (\mathsf{ck}, \mathsf{cm}_i, C, 1), m)$. Output $\mathsf{ct} = \big(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \{\mathsf{ct}_i\}_{P_i \in \mathcal{P}_{Setup}}\big)$.

Decryption $\mathsf{Dec}(\mathbf{B}, \mathsf{ct}, \mathsf{sk})$: If P_i has $\mathsf{sk}_{L,i}$ 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, output $m \leftarrow \Pi_{\mathsf{cWE}}.\mathsf{Dec}_{\mathsf{cWE}}(\mathsf{ct}_i, \mathsf{sk}_{L,i}, \rho_i)$. Otherwise, output $\perp$.

Theorem 1. *Let* $\mathsf{C} = (\mathsf{Setup}, \mathsf{Commit})$ *be a non-interactive extractable commitment scheme and* $\mathsf{cWE} = (\mathsf{Enc}_{\mathsf{cWE}}, \mathsf{Dec}_{\mathsf{cWE}})$ *be a witness encryption scheme 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* $\mathsf{IND\text{-}CPA}$*-secure ECW scheme as per Definition 4.*

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 the semantic security of cWE as defined in Section 4. We assume (w.l.o.g.) that $\mathcal{A}_{\mathrm{ECW}}$ only corrupts one party P_a and that there exists only one honest party P_h[4]. $\mathcal{A}_{\mathsf{cWE}}$ 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 party P_h executing the setup phase and publishing a valid $(\mathsf{cm}_h, \sigma_h)$ on the blockchain and saving $\mathsf{sk}_{L,h}, \rho_h$.
   (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.

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

(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 $(\mathsf{st}, \mathsf{sk}_{L,h}, \rho_h, C, 1, m_0, m_1)$ to the challenger, where $\mathsf{sk}_{L,h}, \rho_h$ were used to generate the commitment by the simulated honest party P_h.

2. Upon receiving ciphertext ct_i from the challenger, if $P_a \in \mathcal{P}_{Setup}$, then $\mathsf{ct}_i = \mathsf{ct}_a$ was computed w.r.t. P_a's commitment cm_a and $\mathcal{A}_{\mathsf{cWE}}$ computes a cWE ciphertext $\mathsf{ct}_h \leftarrow \mathit{\Pi}_{\mathsf{cWE}}.\mathsf{Enc}_{\mathsf{cWE}}(\mathsf{x}_h = (\mathsf{ck}, \mathsf{cm}_h, C, 1), m_0)$ w.r.t. the honest party's commitment cm_h (which is possible since $\mathcal{A}_{\mathsf{cWE}}$ knows $\mathsf{cm}_h, \mathsf{sk}_{L_h}, \rho_h$). Otherwise, only P_h is in $\mathcal{P}_{Setup}$ and $\mathsf{ct}_i = \mathsf{ct}_h$. $\mathcal{A}_{\mathrm{ECW}}$ creates the ECW ciphertext $\mathsf{ct} = \big(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \{\mathsf{ct}_i\}_{P_i \in \mathcal{P}_{Setup}}\big)$ and forwards it to $\mathcal{A}_{\mathrm{ECW}}$. $\mathcal{A}_{\mathsf{cWE}}$ continues the execution of $\mathit{\Gamma}$ 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 $\mathit{\Gamma}$ (and to the system parameters in the genesis block) as in $\mathrm{Game}^{\mathsf{IND\text{-}CPA}}_{\mathit{\Gamma},\mathcal{A},\mathcal{Z},\mathcal{E}}$.

In case $\mathcal{A}_{\mathrm{ECW}}$ provided a valid setup message, it receives 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 $\mathcal{A}_{\mathrm{ECW}}$ are hardwired. Moreover, ct contains a ciphertext ct_h containing the same m_b as in ct_a with probability 1/2. Hence, ct is distributed exactly as in $\mathrm{Game}^{\mathsf{IND\text{-}CPA}}_{\mathit{\Gamma},\mathcal{A},\mathcal{Z},\mathcal{E}}$ with probability 1/2. If $\mathcal{A}_{\mathrm{ECW}}$ has non-negligible advantage in $\mathrm{Game}^{\mathsf{IND\text{-}CPA}}_{\mathit{\Gamma},\mathcal{A},\mathcal{Z},\mathcal{E}}$, 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 $\mathcal{A}_{\mathrm{ECW}}$, $\mathcal{A}_{\mathsf{cWE}}$ in the cWE semantic security game has the same advantage as $\mathcal{A}_{\mathrm{ECW}}$ in $\mathrm{Game}^{\mathsf{IND\text{-}CPA}}_{\mathit{\Gamma},\mathcal{A},\mathcal{Z},\mathcal{E}}$ as long as ct was distributed as in $\mathrm{Game}^{\mathsf{IND\text{-}CPA}}_{\mathit{\Gamma},\mathcal{A},\mathcal{Z},\mathcal{E}}$, which happens with probability 1/2. Hence, in the cWE semantic security game, $\mathcal{A}_{\mathsf{cWE}}$ has half of the advantage of $\mathcal{A}_{\mathrm{ECW}}$ in $\mathrm{Game}^{\mathsf{IND\text{-}CPA}}_{\mathit{\Gamma},\mathcal{A},\mathcal{Z},\mathcal{E}}$.

In case it did not provide a valid setup message, $\mathcal{A}_{\mathrm{ECW}}$ only sees ct with ct_h containing m_b(in this case with probability 1) generated with respect to commitment cm_h for which it does not know the opening (and the same circuit C). Hence, ct is distributed exactly as in $\mathrm{Game}^{\mathsf{IND\text{-}CPA}}_{\mathit{\Gamma},\mathcal{A},\mathcal{Z},\mathcal{E}}$ 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 $\mathcal{A}_{\mathrm{ECW}}$ in $\mathrm{Game}^{\mathsf{IND\text{-}CPA}}_{\mathit{\Gamma},\mathcal{A},\mathcal{Z},\mathcal{E}}$.

Since we assume that $\mathcal{A}_{\mathrm{ECW}}$ has a non-negligible advantage and $\mathcal{A}_{\mathsf{cWE}}$ has a constant fraction of $\mathcal{A}_{\mathrm{ECW}}$'s advantage, it will also obtain a non-negligible advantage and thus break the *cWE* scheme we assume is secure. Hence, $\mathit{\Pi}_{\mathrm{ECW}}$ is an $\mathsf{IND\text{-}CPA}$-secure ECW scheme.

5.2 Other Instantiations

ECW from target anonymous channels [17,2] 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 [17,2,16]. 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. [2] 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 [2] 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 [17], 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 WE of NIZK proofs [11]. Derler and Slamanig [11] (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) [7] (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 signature of knowledge 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 model. So far we have only focused on *encryption* to the future and we focused on the least meaningful notion of secrecy, namely IND-CPA. This falls short of role assignment in the sense of [16], where a message can also be authenticated to come from a given role in the past and where multiple parties can send messages to the same role, for which we in most applications need IND-CCA. We here note that these properties can be added using standard techniques. This is one of several advantages of having a game-based definition where we have white-box access to cryptographic objects like ciphertexts and lottery predicates. We also look at some extensions of the lottery, allowing to set the hardness adaptively and allowing to restrict which accounts can win a given role.

6.1 Extended Lotteries

We first discuss how to extend the lottery with additional features. So far the probability of winning a role can depend on the sl, the role R, and the account sk. 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 machine 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 $\Pr[\mathsf{lottery}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}) = 1] \approx p$. This is the case for most PoS lotteries as the probability of winning depends only on the stake that sk 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 $(\mathsf{R}, \mathsf{sk})$ and $(\mathsf{R}', \mathsf{sk}') \neq (\mathsf{R}, \mathsf{sk})$ it holds that $\Pr[\mathsf{lottery}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}) = 1 \mid \mathsf{lottery}(\mathbf{B}, \mathsf{sl}, \mathsf{R}', \mathsf{sk}')] \approx \Pr[\mathsf{lottery}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}) = 1]$. 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. We use this below.

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 $\mathsf{lottery}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk})$ 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

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

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 sk we assume it can be computed in PPT from the information on $\mathbf{B}$ and the public key pk associated to sk. In particular, filter does not need sk to be efficiently computable. We require that $\mathsf{filter}(\mathbf{B}, \mathsf{sk})$ outputs $\top$ or $\bot$. We require from the lottery that if $\mathsf{filter}(\mathbf{B}, \mathsf{sk}) = \bot$ then $\mathsf{lottery}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}, \mathsf{filter}) = 0$. If $\mathsf{filter}(\mathbf{B}, \mathsf{sk}) = \top$, then we require that $\mathsf{lottery}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}, \mathsf{filter}) = \mathsf{lottery}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk})$. 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 $\mathsf{lottery}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk})$ ignore the filtered winners, but this would again ruin individual winning probabilities of the underlying mechanism.

6.2 IND-CCA ECW

We sometimes require that the EtF is IND-CCA secure. We define this as usual. We establish a game between a challenger C and an adversary $\mathcal{A}$. The game proceeds as the IND-CPA game except that we give the adversary the usual decryption oracles. Let O be an oracle which on input (pk, c) finds the secret key sk corresponding to pk and return the decryption of c. Let O_{ct} be the same oracle but which ignores the input ct. We give the adversary access to the first oracle before giving the challenge ciphertext ct and access to the second oracle afterwards.

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

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

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

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

In the following we will assume that we have an IND-CCA2 ECW. To add IND-CCA2 security to an EtF which already has IND-CPA security we could add to the setup of the network a CRS for a simulation-sound extractable NIZK [20]. When encrypting m to a role S 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. We leave to future work to construct concretely efficient IND-CCA2 EtF.

6.3 Authentication from the Past (AFP)

When the winner of a role S sends a message m to a future role R then it is typically also needed that R can be sure that the message m came from a party P which won the role S. Most PoS blockchains deployed in practice have a lottery where a certificate can be released proving that P won the role S. At the same time parties, identified by their accounts on the blockchain, are able to sign in the name of the account. They each have a public key for an EUF-CMA signature scheme as part of the account and they know the secret key. This is used to sign transactions to ensure that only the owner of the account can transfer funds out of the account. For a blockchain with the above properties AFP can be implemented as follows: the winning certificate can be released to prove that P won the role S and then P can sign the message m.

In general we can add authentication as follows. Recall that P wins S if $\mathsf{lottery}(\mathbf{B}, \mathsf{sl}, \mathsf{S}, \mathsf{sk}_\mathsf{P}) = 1$. Here $\mathcal{R}(\mathtt{x} = (\mathbf{B}, \mathsf{sl}, \mathsf{S}), \mathtt{w}) = \mathsf{lottery}(\mathtt{x}, \mathtt{w})$ is a PPT function and all parties know $\mathtt{x}$ and (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 [8] to sign m under the knowledge of sk_P such that $\mathsf{lottery}(\mathbf{B}, \mathsf{sl}, \mathsf{S}, \mathsf{sk}_\mathsf{P}) = 1$. This will exactly attest that the message m was sent by a winner of the lottery for S. The details of the construction and proof follow using standard techniques and are omitted.

In the following we will assume that we have authentication from the past. We assume that the AFP mechanism is EUF-CMA in the following sense. Parties can send messages via the blockchain. When receiving a message a party outputs (S, m) to indicate that the role S sent the message. Correctness says that if you won a role you can send messages on behalf of that role. We define an EUF-CMA game as follows. The adversary participates as adversary in an execution of the blockchain. At some point it outputs $(\mathsf{S}, m, \mathsf{P})$, where S is a role which was not won by a corrupted party, P is an honest party, and where m is a message never sent by the honest party winning S. The adversary wins the game if P at some point outputted (S, m). We require that any efficient adversary playing the blockchain game wins the game with negligible probability. The definition and construction from EUF-CMA signatures or signatures of knowledge follows using standard techniques and we omit them here.

6.4 Round and Committee Based YOSO Protocols

Having IND-CCA ECW and 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 machines executing the roles can perform the required decryptions of ciphertexts sent to them, compute the code of the role, compute encryptions to the roles in the next round, have time to post these to the blockchain and 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.

With this setup in place, the roles $\mathsf{R}_{r,i}$ of round r can use IND-CCA2 secure ECW and AFP to send secret authenticated messages to the roles $\mathsf{R}_{r+1,i}$ in round $r+1$. They pick up their own messages from the blockchain before slot $r \cdot L$, authenticate using AFP, decrypt using ECW, compute their outgoing messages, encrypt using ECW and post the ciphertexts 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 the assumptions needed on the lottery for this to hold and some details of how to guarantee it.

Assume an adversary which can corrupt parties, identified by sk, and a lottery assigning parties to roles $\mathsf{R}_{r,i}$. We first discuss how to map the corruption status of parties to roles.

1. If a role $\mathsf{R}_{r,i}$ is won by a corrupted party, call the role MALICIOUS.
2. If a role $\mathsf{R}_{r,i}$ is won by several parties, call the role MALICIOUS. This is even if it is won only by honest parties. If $\mathsf{R}_{r,i}$ is won by two honest parties they will both execute the role and send outgoing messages. This might violate the security of the role.
3. If a role $\mathsf{R}_{r,i}$ is won by exactly one honest party, call it HONEST.
4. 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 have assumed the corrupted parties know their secret keys sk in our model, the security experiment can compute in poly-time the corruption status of roles. We can also use these statuses in security reductions, which we will do crucially later. Contrast this to the situation where the security experiment, and thus the reduction, would not known sk of corrupted parties. In that case it could be that a role was won by an honest party but also by a corrupted party which stays completely silent. It sends no messages, it only decrypt messages sent to the role. If the reduction is not aware of the corrupted party winning the role it 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 in principle also be corrupted by a silent malicious party, simulation would become impossible.

We return to the discussion of obtaining honest majority among roles. 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 $H > T$.

Definition 7 (honest committee friendly). *We call a blockchain Γ^V honest committee friendly if there exists n and H and T such that $H > T$ s.t. we can define a sequence of roles $\mathsf{R}_{r,i}$ for $r = 1, \ldots$ and $i = 1, \ldots, n$ such that $\mathsf{R}_{r,1}, \ldots, \mathsf{R}_{r,n}$ are associated to the same slot sl_r and such 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, honest parties in $\mathsf{R}_{r,1}, \ldots, \mathsf{R}_{r,n}$ can if they begin when the blockchain is at sl_r send EtF encrypted and AFP authenticated messages which appears on the blockchain before slot sl_{r+1}.*

We capture the above discussion using a definition.

Definition 8 (YOSO MPC friendly). *Let $\Gamma^V = (\mathsf{UpdateState}^V, \mathsf{GetRecords}, \mathsf{Broadcast})$ be a blockchain and let $\mathsf{E} = (\mathsf{Enc}, \mathsf{Dec})$ be an EtF scheme. We call (Γ^V, E) YOSO MPC friendly if the following holds. The EtF scheme is IND-CCA2 secure. The blockchain Γ^V is EUF-CMA AFP secure and honest committee friendly.*

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 happened at any earlier time. We generate the key for $(\mathsf{R}, \mathsf{sl}_{\mathrm{fut}})$ 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 Π_{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 $\mathsf{ct}^{\mathsf{sk},i}_{(\mathsf{sl},\mathsf{R})} \leftarrow \Pi_{\mathrm{ECW}}.\mathsf{Enc}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}^i_{(\mathsf{sl},\mathsf{R})})$, 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 2 (informal). *Let Γ^V be a YOSO MPC friendly blockchain. Let Π_{TIBE} be a robust secure threshold IBE as in Section 2.3 with threshold $n/2$, and let Π_{ECW} be a secure* IND-CCA2 *ECW, then the construction in Section 7.1 is a secure EtF.*

We describe our proof in Appendix D. 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 $\mathcal{O}_{\mathsf{msk}}^{\text{k-provider}}$ 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[5] of the slot computed as $\mathsf{ct}_{\mathsf{sl}}^{\mathsf{sk}} \leftarrow \text{ECW}.\mathsf{Enc}(\mathbf{B}, \mathsf{sl}, , \mathsf{R}, \mathsf{sk}_{\mathsf{sl}})$ 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.

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 9 (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:*

[5] 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.

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. $m/\bot \leftarrow \mathsf{Dec}(\tilde{\mathbf{B}}, \mathsf{ct}, \mathtt{w})$ *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* $\bot$*.*

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 $(\mathsf{BWE.Gen}, \mathsf{BWE.Enc}, \mathsf{BWE.Dec})$ be an BWE scheme. Recall that one wins the lottery if $\mathsf{lottery}(\mathbf{B}, \mathsf{sl}, \mathsf{R}) = 1$. 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 $\mathsf{Enc}(\mathbf{B}, W, m)$. 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 $(\mathit{EtF}.\mathsf{Enc}, \mathit{EtF}.\mathsf{Dec})$ 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 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.

References

1. A. Afshar, P. Mohassel, B. Pinkas, and B. Riva. Non-interactive secure computation based on cut-and-choose. In P. Q. Nguyen and E. Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 387–404. Springer, Heidelberg, May 2014.
2. F. Benhamouda, C. Gentry, S. Gorbunov, S. Halevi, H. Krawczyk, C. Lin, T. Rabin, and L. Reyzin. Can a public blockchain keep a secret? In R. Pass and K. Pietrzak, editors, *TCC 2020, Part I*, volume 12550 of *LNCS*, pages 260–290. Springer, Heidelberg, Nov. 2020.
3. F. Benhamouda and H. Lin. Mr NISC: Multiparty reusable non-interactive secure computation. In R. Pass and K. Pietrzak, editors, *TCC 2020, Part II*, volume 12551 of *LNCS*, pages 349–378. Springer, Heidelberg, Nov. 2020.
4. D. Boneh, X. Boyen, and S. Halevi. Chosen ciphertext secure public key threshold encryption without random oracles. In D. Pointcheval, editor, *CT-RSA 2006*, volume 3860 of *LNCS*, pages 226–243. Springer, Heidelberg, Feb. 2006.

5. D. Boneh and M. K. Franklin. Identity-based encryption from the Weil pairing. In J. Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer, Heidelberg, Aug. 2001.
6. E. Boyle, N. Gilboa, Y. Ishai, H. Lin, and S. Tessaro. Foundations of homomorphic secret sharing. In A. R. Karlin, editor, *ITCS 2018*, volume 94, pages 21:1–21:21. LIPIcs, Jan. 2018.
7. M. Chase and A. Lysyanskaya. On signatures of knowledge. In C. Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 78–96. Springer, Heidelberg, Aug. 2006.
8. M. Chase and A. Lysyanskaya. On signatures of knowledge. In C. Dwork, editor, *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings*, volume 4117 of *Lecture Notes in Computer Science*, pages 78–96. Springer, 2006.
9. R. Cramer and V. Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In L. R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 45–64. Springer, Heidelberg, Apr. / May 2002.
10. B. David, P. Gazi, A. Kiayias, and A. Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In J. B. Nielsen and V. Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 66–98. Springer, Heidelberg, Apr. / May 2018.
11. D. Derler and D. Slamanig. Practical witness encryption for algebraic languages and how to reply an unknown whistleblower. Cryptology ePrint Archive, Report 2015/1073, 2015. https://eprint.iacr.org/2015/1073.
12. J. A. Garay, R. Gelles, D. S. Johnson, A. Kiayias, and M. Yung. A little honesty goes a long way - the two-tier model for secure multiparty computation. In Y. Dodis and J. B. Nielsen, editors, *TCC 2015, Part I*, volume 9014 of *LNCS*, pages 134–158. Springer, Heidelberg, Mar. 2015.
13. J. A. Garay, A. Kiayias, and N. Leonardos. The bitcoin backbone protocol: Analysis and applications. In E. Oswald and M. Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 281–310. Springer, Heidelberg, Apr. 2015.
14. S. Garg, C. Gentry, A. Sahai, and B. Waters. Witness encryption and its applications. In D. Boneh, T. Roughgarden, and J. Feigenbaum, editors, *45th ACM STOC*, pages 467–476. ACM Press, June 2013.
15. C. Gentry, S. Halevi, H. Krawczyk, B. Magri, J. B. Nielsen, T. Rabin, and S. Yakoubov. YOSO: You only speak once - secure MPC with stateless ephemeral roles. In T. Malkin and C. Peikert, editors, *CRYPTO 2021, Part II*, volume 12826 of *LNCS*, pages 64–93, Virtual Event, Aug. 2021. Springer, Heidelberg.
16. C. Gentry, S. Halevi, H. Krawczyk, B. Magri, J. B. Nielsen, T. Rabin, and S. Yakoubov. YOSO: you only speak once / secure MPC with stateless ephemeral roles. *IACR Cryptol. ePrint Arch.*, 2021:210, 2021.
17. C. Gentry, S. Halevi, B. Magri, J. B. Nielsen, and S. Yakoubov. Random-index PIR with applications to large-scale secure MPC. Cryptology ePrint Archive, Report 2020/1248, 2020. https://eprint.iacr.org/2020/1248.
18. R. Goyal and V. Goyal. Overcoming cryptographic impossibility results using blockchains. In Y. Kalai and L. Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 529–561. Springer, Heidelberg, Nov. 2017.
19. V. Goyal, A. Kothapalli, E. Masserova, B. Parno, and Y. Song. Storing and retrieving secrets on a blockchain. Cryptology ePrint Archive, Report 2020/504, 2020. https://eprint.iacr.org/2020/504.

20. J. Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In X. Lai and K. Chen, editors, *Advances in Cryptology - ASIACRYPT 2006, 12th International Conference on the Theory and Application of Cryptology and Information Security, Shanghai, China, December 3-7, 2006, Proceedings*, volume 4284 of *Lecture Notes in Computer Science*, pages 444–459. Springer, 2006.
21. M. Jawurek, F. Kerschbaum, and C. Orlandi. Zero-knowledge using garbled circuits: how to prove non-algebraic statements efficiently. In A.-R. Sadeghi, V. D. Gligor, and M. Yung, editors, *ACM CCS 2013*, pages 955–966. ACM Press, Nov. 2013.
22. J. B. Nielsen. *On protocol security in the cryptographic model.* Citeseer, 2003.
23. R. Pass, L. Seeman, and a. shelat. Analysis of the blockchain protocol in asynchronous networks. In J.-S. Coron and J. B. Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 643–673. Springer, Heidelberg, Apr. / May 2017.
24. S. Zahur, M. Rosulek, and D. Evans. Two halves make a whole - reducing data transfer in garbled circuits using half gates. In E. Oswald and M. Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 220–250. Springer, Heidelberg, Apr. 2015.

Supplementary Material

A Further Preliminaries

In this appendix, we introduce extra definitions and concepts used in the paper.

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

In this section, we give an overview of the framework from [18] for arguing about PoS blockchain protocol security.

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

- $\mathsf{UpdateState}(1^\lambda) \rightarrow \mathsf{bst}$ where bst is the local state of the blockchain along with metadata.
- $\mathsf{GetRecords}(1^\lambda, \mathsf{st}) \rightarrow \mathbf{B}$ 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 $\varGamma^V$ 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 $\varGamma^V(1^\lambda)$ with empty local state st 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 receives a message m from $\mathcal{Z}$ and potentially receives 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 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 $\mathsf{EXEC}^{\varGamma^V}(\mathcal{A}, \mathcal{Z}, 1^\lambda)$ 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 $\mathsf{view}_{\mathcal{A}}(\mathsf{EXEC}^{\varGamma^V}(\mathcal{A}, \mathcal{Z}, 1^\lambda))$ and the view of the party P_i as $\mathsf{view}_{P_i}(\mathsf{EXEC}^{\varGamma^V}(\mathcal{A}, \mathcal{Z}, 1^\lambda))$. 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 the the full view including round $\tilde{r}$ denoted by $\mathsf{view}^{\tilde{r}} \leftarrow \mathsf{EXEC}_{\mathsf{view},\tilde{r}}^{\Gamma^V}(\mathcal{A}, \mathcal{Z}, 1^\lambda)$.

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 [23] supports adaptive corruption with erasures.

A note on admissible environments [23] 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 In coming sections we will define what it means to encrypt to a future state of the blockchain. First, we need to ensure what it means for a blockchain execution to have evolved from one state to another. 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 [13,23], 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 [18]. Among them, is the *Distinguishable Forking* property which will be the main requirement when introducing the EtF scheme.

Definition 10 (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 11 (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 12 (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.*

Definition 13 (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, 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 [5].

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 [5]. 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 IBE $\mathsf{IBE} = (\mathsf{Setup}, \mathsf{IDKeygen}, \mathsf{Enc}, \mathsf{Dec})$. 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):

- $\mathsf{Share}(\mathsf{msk}, k, n) \rightarrow (\mathsf{msk}_1, \ldots, \mathsf{msk}_n)$ shares the secret.
- $\mathsf{EvalShare}(\mathsf{msk}_i, f) \rightarrow y_i$ obtains a share for $f(\mathsf{msk})$ where f is a function.
- $\mathsf{Combine}((y_i)_{i \in T}) \rightarrow y^*$ 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[6].

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

[6] The security of this type of construction is proven for example in [22] to which we defer the reader for details.

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

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

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

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.

B ECW from [11]

Witness Encryption (WE) was introduced by Garg et al. [14] with a candidate construction based on multilinear maps to create WE for any NP language. Realizing that establishing this kind of WE requires some heavy machinery, others have started looking at more practical ways to realize a similar notion but for restricted languages. A recurring strategy to achieve this is using Smooth Projective Hash Functions (SPHF) which we know how to construct for a specific class of languages called *algebraic*. Using this approach, [11] puts forward a notion inspired by the standard definition of WE, but weakened by having one extra round. While 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 that $\mathtt{x}$ is in the language. [11] considered a new type of WE that is associated with a proof system $\mathit{\Pi} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ 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 [11] and provide additional details on their constructions for sake of completeness.

Construction of ECW. We encode the lottery statement into vectors of commitments satisfying pairing-product equations (PPEs). Subsequently, as shown in [11], one can extend the construction we detail below into the more general case of PPEs. Such an extension directly implies a SPHF construction for statements in the GS proof framework. Equipped with this construction, one can now construct an ECW 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. We now provide details underlying the construction in [11].

Groth-Sahai NIZK Proofs Groth-Sahai proofs work by using commitments that are homomorphic both with respect to group operations and a bilinear map. The 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 commitments to each element of the witness and some additional information, the proof. Commitments and proof satisfy some related set of equations computable by the verifier because of their algebraic properties and will be a convincing proof.

Pairing product equations (PPEs) in the Groth-Sahai (GS) framework can be written as

$$\prod_{q=1}^{N} e(a_q \prod_{i=1}^{m} x_i^{\alpha_{qi}}, b_q \prod_{j=1}^{n} y_j^{\beta_{qj}}) = t$$

where $a_q \in G_1, b_q \in G_2, \alpha_{qi}, \beta_{qj} \in R$ and $t \in G_T$ are constants, while $x_i \in G_1$ and $y_j \in G_2$ are variables that we wish to prove they satisfy the equation. In the GS framework we commit to values x and y by $\mathsf{cm}_x = \mathsf{Commit}(x)$ and $\mathsf{cm}_y = \mathsf{Commit}(y)$ and subsequently prove that the openings to these commitments satisfy the equations.

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, [9]) 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 $\mathsf{H} \leftarrow \mathsf{hash}(\mathtt{lpar}; \mathsf{hk}, \mathtt{x})$. The formal definition of SPHF follows.

Definition 14. *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* $\mathtt{x} \in \mathcal{L}$ *and outputs a hash value* pH*.*

A SPHF needs to satisfy the following properties:

Correctness. It is required that $\mathsf{hash}(\mathtt{lpar}; \mathsf{hk}, \mathtt{x}) = \mathsf{projhash}(\mathtt{lpar}; \mathsf{hp}, \mathtt{x}, \mathtt{w})$ for all $\mathtt{x} \in \mathcal{L}$ and their corresponding witnesses $\mathtt{w}$.

Smoothness. It is required that for any $\mathtt{lpar}$ and any $\mathtt{x} \notin \mathcal{L}$, the following distributions are statistically indistinguishable:

$$\begin{gathered}\{(\mathsf{hp}, \mathsf{H}) : \mathsf{hk} \leftarrow \mathsf{hashkg}(\mathtt{lpar}), \mathsf{hp} \leftarrow \mathsf{projkg}(\mathtt{lpar}; \mathsf{hk}, \mathtt{x}), \mathsf{H} \leftarrow \mathsf{hash}(\mathtt{lpar}; \mathsf{hk}, \mathtt{x})\} \\ \left\{(\mathsf{hp}, \mathsf{H}) : \mathsf{hk} \leftarrow \mathsf{hashkg}(\mathtt{lpar}), \mathsf{hp} \leftarrow \mathsf{projkg}(\mathtt{lpar}; \mathsf{hk}, \mathtt{x}), \mathsf{H} \xleftarrow{\$} \Omega\right\} \quad .\end{gathered}$$

where Ω is the set of hash values.

Pseudo-randomness. This property states that the hash value $\mathsf{H}(\mathtt{lpar}; \mathsf{hk}, \mathtt{x})$ for a randomly chosen statement $\mathtt{x} \xleftarrow{\$} \mathcal{L}_{\mathtt{lpar}}$ should look computationally random (i.e., indistinguishable from $\mathsf{H} \xleftarrow{\$} \Omega$).

SPHF for Linear Groth-Sahai Commitments We recall the construction of SPHF for Groth-Sahai (GS) commitments from [11]. A GS commitment is a dual-mode commitment consisting 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. Here we only focus on the perfectly binding mode. This is due to the computationally indistinguishability of the two setups (under the DLIN assumption) and all the following explanations can also be applied to the perfectly hiding mode in a straightforward manner.

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

$$\begin{aligned}
[U_1] &= ([\tau_1], [0], [1]) \\
[U_2] &= ([0], [\tau_2], [1]) \\
[U_3] &= \tau_3 \cdot [U_1] + \tau_4 \cdot [U_2] = ([\tau_1 \tau_3], [\tau_2 \tau_4], [\tau_3 + \tau_4])
\end{aligned}$$

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

$$\mathsf{cm}_m = ([0], [0], [m]) + r_1 U_1 + r_2 U_2 + r_3 U_3 = ([\tau_1(r_1 + \tau_3 r_3)], [\tau_2(r_2 + \tau_4 r_3)], [m + r_1 + r_2 + r_3(\tau_3 + \tau_4)]).$$

Note that cm_m can be seen as an encryption of $[m]$ with respect to randomness $(r_1 + \tau_3 r_3, r_2 + \tau_4 r_3)$.

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

$$\mathcal{L}_{\mathtt{lpar}} = \{\mathtt{x} = ([m], \mathsf{cm}_m) | \exists \mathtt{w} = (r_1, r_2, r_3) \in \mathbb{Z}_q^3 : \mathsf{cm}_m = ([0], [0], [m]) + r_1 U_1 + r_2 U_2 + r_3 U_3\}$$

A SPHF for such linear language $\mathcal{L}_{\mathtt{lpar}}$ 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

$$\begin{aligned}
\gamma_1 &= \alpha_1 [\tau_1] + [\alpha_3] \\
\gamma_2 &= \alpha_2 [\tau_2] + [\alpha_3] \\
\gamma_3 &= \alpha_1 [\tau_1 \tau_3] + \alpha_2 [\tau_2 \tau_4] + \alpha_3 [\tau_3 + \tau_4]
\end{aligned}$$

$\mathsf{hash}(\mathtt{lpar}; \mathsf{hk}, \mathtt{x})$**:** 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 [11] for the security proof of the above SPHF.

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

- Initialize a list, L, of pairs of strings.
- Upon receiving a message $(\texttt{input}, x)$ from P_1, store x and continue
  1. Upon receiving message $(\texttt{input}, y)$ from P_i, insert the pair (P_i, y) into L. If P_1 is corrupted send $(P_i, f(x, y))$ to the adversary. Otherwise, send $(\texttt{messageReceived}, P_i)$ to P_1.
  2. Upon receiving a message $\texttt{getOutputs}$ from P_1, send $\{(P_i, f(x, y))\}_{(P_i,y)\in L}$ to P_1.

Fig. 2. MS-NISC Functionality $\mathcal{F}_{\mathsf{MS\text{-}NISC}}$

C Alternative Constructions of cWE

C.1 cWE from MS-NISC

Two-Party Non-Interactive Secure Computation (2P-NISC) is a type of protocol where a sender S (with input y) and a receiver R (with input x) want to jointly compute a function $f(x, y)$. In this setting, R first publishes a first message. Then, any sender S holding an input y can send a message to the receiver revealing only $f(x, y)$ to R. As usual in secure computation, the protocol must provide privacy of the inputs and correctness of the output.

Afshar et al. [1] introduced another flavour of NISC called *Multi-Sender NISC* (MS-NISC) where the first message can be reused to run secure computation with many different senders. That is, R, with input x, publishes a first message as before, but now any party who wants to participate in secure computation with R can send back a message to S who can then output the result of the computation. The ideal functionality of MS-NISC as presented in [1] is depicted in Fig. 2.

In Fig. 3, 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 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 $f(w, y)$ that outputs a decryption key iff the witness w satisfies a given relation.

Note, 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.

Initialization: Initialize $\mathcal{F}_{\mathsf{MS\text{-}NISC}}$ by instantiating a list L of pairs of strings.
Commit: P_1 proceeds as follows:

- Commits to its witness $\mathtt{w}$ by calling $\mathcal{F}_{\mathsf{MS\text{-}NISC}}$ on input $(\mathtt{input}, \mathtt{w})$.

Encryption: P_2 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 P_1.

Decryption: P_1 receives $(\mathtt{messageReceived}, P_2)$ from $\mathcal{F}_{\mathsf{MS\text{-}NISC}}$ and ct from P_2 and proceeds as follows:

- Calls $\mathcal{F}_{\mathsf{MS\text{-}NISC}}$ on input $\mathtt{getOutputs}$.
- Upon receiving k from $\mathcal{F}_{\mathsf{MS\text{-}NISC}}$, outputs $m \leftarrow k \oplus \mathsf{ct}$.

Fig. 3. Construction of cWE based on MS-NISC

C.2 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 [1], we observe that we can construct cWE from a MS-NISC protocol without the precautions against a malicious sender P_2. 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 [24] based on a technique called half-gates. Using their technique in the privacy-free setting results in garbles circuits containing one ciphertext for each AND gate and no ciphertexts for XOR gates.

cWE from privacy-free GC and OT We conclude this section by presenting an efficient construction of cWE using only a privacy-free garbled circuit and oblivious transfer. The protocol is shown in Fig. 4.

Initialization: We let $\mathsf{COT}_1()$ and $\mathsf{COT}_2()$ be first and second message of a committed-OT (as in [1]).
Let $C_{\mathtt{x}}$ be the circuit that encodes the statement $\mathtt{x}$ such that on input a witness $\mathtt{w}$ it will output $C_{\mathtt{x}}(\mathtt{w}) = 1$ iff $(\mathtt{x}, \mathtt{w}) \in \mathcal{R}$.
We use the simple notation Garble to represent an algorithm that on input a circuit C will output a garbled version of that circuit $gc \leftarrow \mathsf{Garble}(C)$.

Commit: P_1 proceeds as follows:

- commits to its witness $\mathtt{w}$ by inputting the witness as the first message in a committed-OT $\mathsf{COT}_1(\mathtt{w})$

Encryption: P_2 does the following:

- generates a key k of length $|m|$ and computes the ciphertext $\mathsf{ct} \leftarrow k \oplus m$.
- creates a circuit $C_{\mathtt{x}}$ as described and garbles the circuit according to above by $gc \leftarrow \mathsf{Garble}(C_{\mathtt{x}})$. Furthermore, P_2 construct the output label of gc such that $K^1_{out} = k$.
- sends ct and gc and corresponding input labels $\mathsf{COT}_2(\{K^0_l, K^1_l\}_{l \in [0,..,|\mathtt{w}|]})$ to P_1

Decryption: Upon receiving ct, gc and input labels, P_1 proceeds as follows:

- evaluates the circuit gc on the input labels corresponding to $\mathtt{w}$, obtain the output label k (assuming $(\mathtt{x}, \mathtt{w}) \in \mathcal{R}$) and decrypt by $m \leftarrow \mathsf{ct} \oplus k$.

Fig. 4. Construction of cWE based on GC and OT

D Proof of Security for Our EtF Construction

Here we prove security of Theorem 2. Recall 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.

CCA and resharing. Our proof below ignores decryption oracles for the IBE scheme and considers the case where the master secret key is shared only at the beginning (instead of at each slot). We observe these can be accounted for with additional hybrids in our proof below in a straightforward manner and we defer these details to the full version of the paper.

1. The non-threshold case. The simplified setting we will now show security for is in Fig. 5.

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 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 $\mathcal{O}^{\text{k-provider}}_{\mathsf{msk}}$.

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

Fig. 5. Hybrid non-threshold setting for proof of security

the EtF security experiment. We first to construct an adversary $\mathcal{A}_{\mathrm{IBE}}$ for IBE security using $\mathcal{A}^{\text{no-thresh}}_{EtF}$. Adversary $\mathcal{A}_{\mathrm{IBE}}$ 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 $\mathcal{A}_{\mathrm{IBE}}$ runs an execution of the blockchain with $\mathcal{A}^{\text{no-thresh}}_{EtF}$ where $\mathcal{A}_{\mathrm{IBE}}$ simulates the honest parties. In this execution $\mathcal{A}_{\mathrm{IBE}}$ 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[7]. On query $(\mathbf{B}, \mathsf{sl})$:
  - *if a corrupted party has won* the role for slot sl (i.e. $\mathsf{winners}(\mathbf{B}, \mathsf{sl}, \mathsf{R}) \cap S_{\mathrm{corr}} \neq \emptyset$) then invoke the IBE challenger oracle on identity sl obtaining $\mathsf{sk}_{\mathsf{sl}}$ and broadcast $\mathsf{ct}^{\mathsf{sk}}_{\mathsf{sl}} \leftarrow \mathrm{ECW}.\mathsf{Enc}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \mathsf{sk}_{\mathsf{sl}})$.
  - *if a corrupted party has not won* the role for slot sl then broadcast the encryption of a dummy plaintext $\mathsf{ct}^{\mathsf{sk}}_{\mathsf{sl}} \leftarrow \mathrm{ECW}.\mathsf{Enc}(\mathbf{B}, \mathsf{sl}, \mathsf{R}, \vec{0})$ 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 $\mathcal{A}^{\text{no-thresh}}_{EtF}$ 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, $\mathcal{A}^{\text{no-thresh}}_{EtF}$ will return $(\mathbf{B}, \mathsf{sl}^*, \mathsf{R}, m_0, m_1)$ and $\mathcal{A}_{\mathrm{IBE}}$ will forward $((\mathsf{sl}^*, \mathsf{R}), m_0, m_1)$ to the IBE challenger.

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

- 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.

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 Appendix D.1.

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. 5).

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

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.* 5 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 $(\mathsf{sk}_i^{\mathsf{ID}})_{i\in[n]} \leftarrow \mathcal{S}_{\text{kg}}(\mathsf{mpk}, (\mathsf{msk}_i)_{i\in S_{\text{corr}}}, \mathsf{sl})$ 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 $\mathcal{S}_{\mathsf{msk}}$, 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. 5 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 $\mathcal{S}_{\mathsf{msk}}$ 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.

D.1 Bounding the Advantage of $\mathcal{A}_{\mathbf{IBE}}$ in Proof of Theorem 2

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

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

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. 5). 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).