Can a Blockchain Keep a Secret?

Fabrice Benhamouda[1], Craig Gentry[1], Sergey Gorbunov[2,5], Shai Halevi[1], Hugo Krawczyk[1], Chengyu Lin[3], Tal Rabin[1], and Leo Reyzin[2,4]

[1]Algorand Foundation
[2]Algorand Inc.
[3]Columbia University
[4]Boston University
[5]University of Waterloo

April 21, 2020

Abstract

Blockchains are gaining traction and acceptance, not just for cryptocurrencies but increasingly as a general-purpose architecture for distributed computing. In this work we seek solutions that allow a blockchain to act as a trusted long-term repository of secret information: Our goal is to deposit a secret with the blockchain and specify how to use it (e.g., the conditions under which it is released), and have the blockchain keep this information secret and use it only in the requested manner (e.g., only release it once the conditions are met). This simple functionality would be an enabler for many powerful applications, including signing statements on behalf of the blockchain, using blockchain as the control plane for a storage system, performing decentralized program-obfuscation-as-a-service, and many more.

We present a scalable solution for implementing this functionality on a public proof-of-stake blockchain, in the presence of a mobile adversary controlling a small minority of the stake, using proactive secret sharing techniques. The main challenge is that, on the one hand, scalability requires that we use small committees to represent the entire stake, but, on the other hand, a mobile adversary may be able to corrupt the entire committee if it is small. For this reason, prior proactive secret sharing solutions are either non-scalable or insecure in our setting.

We solve this issue using "player replaceability", where the committee is anonymous until after it performs its actions, as in the Algorand blockchain. (Algorand uses player replaceability to defend against DDoS attacks.) Our main technical contribution is a system that allows sharing and re-sharing of secrets among the members of small dynamic committees, without knowing who they are until after they perform their actions. Our solution handles a fully mobile adversary corrupting less than 25% of the stake at any time, and is scalable in terms of both the number of parties on the blockchain and the number of time intervals.

Blockchain, Mobile Adversary, Player Replacability, Proactive Secret Sharing

1 Introduction

Imagine posting an encrypted will to a blockchain to be opened only when the blockchain reaches consensus that the subject died. More generally, consider using the blockchain as a secure storage solution, allowing applications and clients to deposit potentially secret data and specify the permissible use of that data. Some example applications include keeping a signature key that can sign on behalf of the blockchain (or on behalf of a specific application), providing a root of trust for key-management and certification solutions, allowing users and programs to enforce policies specifying how their private data can be used, enabling program obfuscation and encrypted computation as a service (using homomorphic encryption and consensus-enforced conditional decryption), and many others. This basic functionality is an enabler for solutions that treat the blockchain as a trusted party that performs complex tasks for its clients protected by the integrity properties of the blockchain.

In this work we investigate the functionality of keeping a secret on the blockchain, in the context of public proof-of-stake blockchains. In this context, to achieve a scalable solution we need the "will of the blockchain" to be represented by a small committee. At the same time, we need our solution to remain secure even against a mobile adversary that can corrupt different participants at different times, as long as it corrupts no more than a set fraction of the total stake at any single time interval. But this presents us with a challenge: If we use small committees, the adversary would have enough "corruption budget" to corrupt them all.

One beautiful approach for addressing this challenge is using the *player replacability* property of systems such as Algorand [CM19]. In such a system, committees are selected to reach consensus on blocks. and each selected member is charged with sending a *single* message. Moreover, the member remains completely anonymous before it sends that message. The attacker, not knowing the identities of the selected members, cannot target them for corruption until after the committee finishes its job. This idea is implemented using "cryptographic sortition" based on verifiable random functions (VRFs) [MRV99].

Using this same approach for the purpose of keeping a secret is far from simple. How can one share a secret among the members of a committee without knowing who they are? At first, this may seem "obviously impossible." It appears that to distribute a secret among the members of a small *unknown* committee would require that *any small committee* be able to recover the secret. But then also the committee of all corrupted players will be able to recover it. Luckily this naïve thinking is incorrect, as there are other aspects of a blockchain that let us solve this problem.

We note that a "solution" to the above problem can be devised [GG17] using the cryptographic sledgehammers of witness encryption [GGSW13] and/or obfuscation [BGI+12, GGH+16]: The idea is that "the committee votes to open the secret" can be made into an NP-statement, so we can use witness-encryption relative to that statement. While this shows that polynomial-time solutions may exist, we are interested in solutions that can be used in practice.

1.1 Using Proactive Secret Sharing

In this work we address the above challenges using proactive secret sharing techniques [OY91, CH94, HJKY95]. In this approach, the secret is shared among members of a dynamic committee which is refreshed every few rounds. Whenever the committee changes, a handover protocol is executed in which members of the current committee re-share the secret among the members of the next one, in a way that prevents the adversary from combining shares from different committees. Crucially,

our solution uses the "anonymous committee" technique mentioned before, so the attacker does not know the identity of committee members until after they hand over fresh shares to a new committee and erase their own. So by the time the attacker learns the identities, there is no point in compromising these parties.

Early work on proactive secret sharing assumed a fixed committee (say of size N), where parties are occasionally corrupted by the adversary and later recover and re-join the honest set. A drawback of these protocols in our context is that they require all the members to participate in every handover protocol. In the case of public proof-of-stake blockchains, this solution is not scalable to a huge number of users since every stakeholder has to participate all the time.

The case of dynamic committees was addressed in a number of previous works (e.g., [SLL08, BDLO15, MZW+19]), but these protocols do not ensure the anonymity of new committees. In our setting we need a method of selecting the members in the next committee and sending messages to them, without the senders knowing who the recipients are. Moreover, communication in our model must be strictly one way, since the adversary learns a node's identity once it sends any message. In fact committee members are not even allowed to know the identities of their peers (since some of them may be adversarial), so interactive protocols among the current members are also not allowed. Designing a solution in this challenging context is the main contribution of this work.

1.2 Overview of Our Solution

Our solutions consists of time intervals (or periods) with a handover protocol at the beginning of each one. In each period i, the secret is shared among members of a period-i committee, and the committee changes from one period to the next. The committee at every period is small, consisting of only c_i members out of the entire universe of N users. This lets us reduce the complexity of the handover protocol from $\Omega(n)$ to $O(c_i)$ broadcast messages, while retaining comparable security. Our proactive solution is based on the Shamir secret sharing scheme [Sha79], and uses the following components:

- We use the blockchain itself in several roles. First, it gives us a public-key infrastructure. Namely, we assume that in each period i we have a set of N_i parties and every party knows the long-term public keys of the other parties. In addition, the blockchain provides us with a broadcast mechanism. Namely, every party can broadcast a message to all other parties, which would reach all of them in at most δ rounds (where δ is a known bound).

- We use a cryptographic sortition process for choosing random but verifiable committees. See section 2.4.

- We use two PKE schemes, one for long-term keys and the other for ephemeral committee-specific keys. The long-term PKE needs to be anonymous [BBDP01], namely, ciphertexts under that scheme do not disclose the public key that was used to generate them, see section 2.6. We also need a specific combination of these schemes (see below) to be secure against receiver-selected-opening attack, see section 2.5.

- We use non-interactive zero-knowledge (NIZK) proofs for statements about encrypted values lying on a low-degree polynomial (under the ephemeral scheme). The number of values in each one of these statements is small, essentially the size c_i of the committees from above.

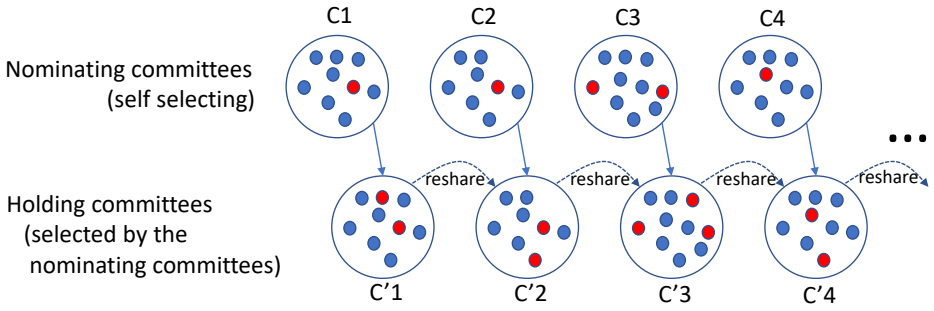


Figure 1: Nominating and holding committees, the red dots represent corrupted parties.

The crux of our solution is a method for selecting the committee to hold the secret, allowing the secret to be passed from one committee to the next while keeping members of the committee anonymous. We use two different committees for that purpose:

- A *holding committee* that holds shares of the secret.
- A *nominating committee* whose role is to choose the holding committee (the nominating committee does not hold shares).

Crucially, the nominating committee can be self-selected using cryptographic (verifiable) sortition, since its members need not know any global secrets. Once self-selected, each member of the nominating committee chooses one member of the future holding committee and publishes on the blockchain information that allows the members of the current holding committee to re-share the secret among the members of the next holding committee. Nothing in the information published by the nominating committee or in the resharing actions should reveal the identity of any member of the new holding committee to anyone other than the member itself. See fig. 1 for a pictorial illustration of this process.

In more detail, every member of the nominating committee chooses one member of the holding committee at random, then chooses and posts to the blockchain a new ephemeral public-key for that party, along with an encryption of the corresponding ephemeral secret key under the party's long-term public key. We use anonymous encryption to ensure that the ephemeral keys and ciphertexts do not betray the identities (or long-term keys) of the holding committee members. Only the members selected by some nominator can learn which keys were encrypted under their long-term public keys. Note that the ephemeral keys themselves may use a different encryption scheme, which need not be anonymous. Only the ciphertext under the parties' long-term keys must be anonymous.[1]

Once the ephemeral keys of the next committee are posted, everyone knows the size of that committee (call it c_i). Each member of the current holding committee re-shares its share using a t-of-c_i Shamir scheme (for some agreed-upon threshold t), then uses the j'th ephemeral key to encrypt the j'th share and broadcast all these encrypted shares along with a proof that the sharing was done properly.

[1]However, the mobile adversary in our setting can see the public keys and ciphertexts and then decide who to corrupt. Hence the combination of long-term and ephemeral schemes needs to be secure against selective opening attacks.

Members of the next holding committee recover their ephemeral secret keys by decrypting the posted ciphertexts using their long-term keys. Each member then collects all the shares that were encrypted under its ephemeral key and uses them to compute its share of the global secret in the new committee. Note that all these ciphertexts are publicly known, so they can serve also as a commitment to the share, enabling the holding committee member to prove correct re-sharing in the next iteration of the protocol. (If the ephemeral PKE scheme is also linearly-homomorphic, then it may be possible to compress this commitment to a single ciphertext encrypting the share of that party.)

An important feature of this solution is that *it does not require the nominating committee to prove anything* about how they chose their nominees or how the ephemeral keys were generated. (Note that proving the selection would be of limited value since even if we force corrupted members of the nominating committee to abide by the protocol, they can corrupt the nominees as soon as they are chosen.) Dispensing of any proofs in this step simplifies the protocol a great deal. Asking the nominating committee to prove anything about their choice while maintaining anonymity would have required the use of expensive SNARKs. In contrast, the statements being proven in our solution (and their witnesses) are all short: Their size size depends only on the committee size, and does not grow with the total stake or the history of the blockchain. Hence the NIZK complexity in our solution is just polynomial in the security parameter, even if we were to use the most naive ones.

On the other hand, the lack of proofs by the nominating committee allows the adversary to double dip: An adversary controlling an f fraction of the stake will have roughly an f fraction of the nominating committee members (all of which can choose to nominate corrupted parties to the holding committee), and another f fraction of the holding committee members nominated by honest parties. Hence, our solution can only tolerate adversaries that control less than 25% of the total stake. In the appendix we discuss a variant of the protocol that does require SNARKs and is resilient to a higher percentage of adversarial stake, but in a weaker adversary model.

We also comment that members of the holding committee must replace the secret key for their long-term keys (and erase the old secret key) before they post their message in the protocol. Otherwise the adversary can corrupt them and use the old secret key to decrypt everything that was sent to them (in particular the shares that they received). This means that these "long-term keys" are either forward-secure, or they are not really long-term but are used once and then discarded.

1.2.1 Aside: anonymous PKE and selective-opening

Our solution relies on the use of anonymous PKE in a setting with dynamic corruptions. This brings up the question of whether anonymous PKE remains anonymous under selective-opening attacks. Note that for secrecy, it is known that semantic-security *does not* implies secrecy against selective-opening. For anonymity, however, we provide in appendix B.1 strong evidence that the answer is yes. Namely anonymity in the static case implies also anonymity against selective opening. However we do not fully resolve this question, it remains an interesting problem for future work.

1.3 Related Work

Secret sharing was introduced in the works of Shamir [Sha79] and Blakley [Bla79]. The proactive setting stems from the mobile adversary model of Ostrovsky and Yung [OY91] followed by works of Canetti-Herzberg and Herzberg et al. in the static setting [CH94, HJKY95, HJJ+97]. The

dynamic setting where the set of shareholders changes over time was contemplated in several works, cf., [DJ97, WWW02, ZSvR05, STY05, SLL10, DGGK10, BDLO15]. We refer the reader to Maram et al. [MZW+19] for a detailed comparison of these works (in particular, see their Section 8 and Table 4).

Several works also deal with dynamic shareholder sets in the context of blockchain. The Ekiden design [CZK+19] provides privacy in smart contracts using a trusted execution environment (TEE). They also use threshold PRFs to derive periodic contract-specific symmetric keys for encrypting smart-contracts. Their scheme is described using a static committee but they suggest the use of proactive secret sharing and rotating committees for increased security. Calypso [KAS+18] uses blockchain and threshold encryption to build an auditable access control system for the management of keys and confidential data. That work contemplates the possibility of shareholder committees changing periodically to accommodate the dynamic nature of participants in a blockchain setting.

Helix [ACG+18] selects per-block committees who agree on the next block in the chain using a PBFT protocol. A main characteristic of Helix, originating with the Honey Badger BFT work [MXC+16], is that selection of transactions for inclusion in a block is done while the transactions are encrypted (and routed anonymously) hence avoiding biases in the selection. Only after the transactions for inclusion are decided, Helix uses the shareholders committee to decrypt them using threshold decryption. In addition, Helix uses the threshold decryption scheme to implement a verifiable source of randomness that seeds the selection of the next committee (committee members are selected with probability proportional to their reputation rank). However, in strong contrast to our use of threshold schemes, in Helix the set of shareholders is static. Dfinity [HMW18] also uses threshold cryptography (signatures in their case) and dynamic shareholder committees for implementing a randomness beacon but the shared secret changes with each new committee.

Closest to our work is CHURP [MZW+19] that builds proactive secret sharing over dynamic groups in a blockchain environment. Their goal is to minimize communication in the "optimistic runs" of the protocol. The crucial difference between their work and ours is that they simply *assume a bound of t corrupted committee members*, without regard to how to ensure that such a bound holds. In fact their techniques are inapplicable in our setting, as they crucially build on active participation of the receiving committee in the handover protocol. As a result, in the mobile adversary model that we consider their protocol is either non-scalable (requiring participation of all the stakeholders) or insecure (if using small committees). In contrast, our main goal is to maintain absolute secrecy of the new committee members during handover, to enable the use of small committees.

2 Background and Definitions

2.1 Synchrony, Broadcast, and PKI

We use the blockchain as both a synchronization mechanism and a broadcast channel. For synchrony, we assume that all parties know what is the current block number on the blockchain. For communication, any party can broadcast a message to the blockchain at round i, and be assured that everyone will receive it no later than round $i + \delta$ (where δ is a known bound). Moreover, a party that received a message on the blockchain in round i is ensured that all other parties received the same message at the same round.

An important feature of our solution is that it uses broadcast as the only communication

mechanism. This solution is built around the premise that parties remain anonymous until they post a message. This means in particular that a party can only receive a message if everybody else also received it, else a network adversary could notice that party and target it for corruption.

Finally, we use the broadcast mechanism to implement the PKI that we need. Each party in our system can simply broadcast a public key on the blockchain, hence letting everyone else know about this key.

2.2 The Mobile Adversary Model

We use essentially the Ostrovsky-Yung adversary model from [OY91], with parties that are occasionally corrupted by the adversary and can later recover and re-join the honest set. In our setting of a public proof-of-stake blockchain, it takes some care to define what we mean by a party. While it is possible to consider each token as a party, we find it more natural to keep the traditional view of parties represented by cryptographic keys, where each key controls some number of tokens (which are the stake of that key).

In our model the "corruption budget" of the adversary is expressed in tokens rather than in parties. That is, we assume that the adversary can control up to some fraction of the total stake, rather than some fraction of the keys. This can be formulated using a UC-like environment that provides parties with tokens and move those tokens between them at will. As normal with honest-majority protocols, our protocol will only be secure against environment/adversary pairs where the adversary never corrupts parties controlling more than f-fraction of the overall tokens, with f a parameter of the protocol (in our case $f < 0.25$).

The stake of compromised parties. Assuming that parties can recover from compromise may seem questionable in our context. After all, if the adversary corrupts a party holding some stake, can't it just "take the money and run"? That is, can't the adversary simply transfer all the stake of a corrupted party into the adversary's own coffers, thereafter forever controlling it?

The answer lies in the distinction between keys that control tokens (called *spending/withdrawal* keys) and keys that are used in the consensus (called *participation/validation* keys): PoS blockchain usually assume that stake-controlling keys are kept highly secure (e.g., offline, in a hardware device, or using some secret-sharing mechanism), and are only accessed infrequently. The cryptographic keys used for the consensus, on the other hand, must be accessed frequently and kept online. In our model we therefore assume that the token-controlling keys are (almost) never compromised, but the consensus keys are easier to corrupt. A corrupted party is one whose consensus key was compromised, and it can later recover by (cleaning up the node and) using the token-controlling key to choose and broadcast a new consensus key.

Epochs and periods. The speed in which the adversary can move between parties is captured (for the most part) by breaking the lifetime of the system into epochs, and assuming that the set of all the parties that are compromised at any time during the epoch controls at most some f fraction of the total stake. (To simplify notations we assume that the total stake is held fixed during each epoch. While not strictly needed, it makes is easier to talk about the total stake N_i in the i'th epoch.)

As is common in the proactive security literature, parties may be unaware of the fact that they were compromised in the previous epoch, hence they must run some recovery procedure in every

epoch, just in case they were compromised. In our case, the only operation that all parties must do every epoch is to choose a new consensus key pair and broadcast their new public key signed by their token-controlling key. In addition, parties must also refresh their consensus keys each time they participate in the periodic re-sharing protocol (and also if they participate in reconstructing a shared secret, producing a threshold signature, etc.).

While epochs may be long (e.g., weeks or more), our solution enables a much more rapid refresh of the shares (e.g. every few minutes). Importantly, this refresh operation is cheap, involving only broadcast messages from the handful committee members. For example, rapid refresh of the secret may be needed when that secret is used by a higher level application (e.g., to decrypt or sign something). Every time the secret is used, the adversary can identify the current holding committee and target them for corruption, hence the committee must rotate and the shares must be refreshed. We thus further partition each epoch into periods, and the sharing of the secret is refreshed every period.

Three different corruption modes. It is instructive to consider the type of corruptions we are likely to confront in a PoS blockchain and their characteristics.

- *Mostly static adversarial base.* There may be a set of token keys that are held by the adversary, and hence their consensus keys remain adversarial throughout. While that set (and the stake that it holds) is not completely static, it may be reasonable to assume that it changes slowly.
- *Somewhat dynamic node corruptions.* A second type of adversarial parties represent nodes where the stake key is held by honest participants but the consensus keys are subject to compromise due to security breaches. These tend to be more dynamic from the first set, but corruptions still require significant effort on the part of the attacker. It may be reasonable to assume that corruption of new nodes usually takes significant time, possibly in the order of full epochs.
- *Fully dynamic fail-stop.* A third set of "adversarial" nodes are fail-stop nodes, that are just knocked off due to denial-of-service (DoS) attacks. It seems reasonable to assume that the adversary can mount a DoS attack almost instantaneously and keep it going for a while.

Our main solution in section 3 only assumes a bound on the fraction of total stake held by the adversary at any epoch, but it is only resilient to a compromise of less than 25% of the stake. In contrast, in appendix A we sketch a variant of the solution that remains secure even against adversaries that control higher stake (more or less 35%), but only if the adversary is slow-moving and full corruption takes more than one epoch to inflict. (That protocol remains secure even when the adversary can inflict fail-stop corruptions quickly.)

2.3 Proactive Secret Sharing

Recall that in Shamir secret sharing [Sha79], a secret σ is shared among n parties with recovery threshold t (denoted t-of-n sharing) by choosing a random degree-$(t-1)$ polynomial whose free term is σ (over some field $\mathcal{F}$ of size at least $n+1$), associating publicly with each party i a distinct point $\alpha_i \in \mathcal{F}$, then giving that party the value $\sigma_i = F(\alpha_i)$. A collection of t parties or more can interpolate and recover the free term of F.

Proactive secret sharing [OY91, CH94, HJKY95] is a method of maintaining a shared secret in the presence of a mobile adversary. In a proactive secret-sharing protocol, the secret is re-shared in

every time period in such a way that shares from different periods cannot be combined to recover the secret. The only way to recover the secret is to obtain enough shares *from the same period*, a task which is assumed to be beyond the adversary's grasp. We included in section 1.3 some details on different solutions to this problem in different setting.

2.4 Verifiable Random Functions and Cryptographic Sortition

A verifiable random function (VRF) [MRV99] is a pseudorandom function that enables the key holder to prove (input, output) pairs. Technically it consists of key-generation, evaluation, and verification: The key generation chooses public and secret keys, evaluation takes the secret key and an input and returns the function value and a proof, and verification takes the public key, input, value, and proof, and outputs accept or reject.

The security properties of a VRF are (a) *pseudorandomness*: the function value (sans proofs) are pseudorandom, even given the public key; (b) *completeness*: the (value, proof) pairs that are output by evaluation are accepted; and (c) *uniqueness*: it is infeasible to generate a public key, an input, and two different (value, proof) pairs, which are both accepted by the verifier (wrt these public key and input). Constructions of VRFs are known under various number theoretic assumptions (such as RSA, DDH, or hardness in paring groups), with or without the random-oracle heuristic.

VRFs can be used to implement *cryptographic sortition*, which is essentially a verifiable lottery [CM19]. In its simplest form, there is a prescribed random process outputting yes with some probability p or no with probability $1 - p$. The parties are supposed to run this process in order to self-select themselves to a committee. Each party has a VRF key pair, the parties all know each other's public keys, and there is a publicly known input value that they all agree on. Each party computes the VRF on the public input using its secret key, thereby obtaining a random seed that it can use to run the prescribed random process, and this random process tells the party whether or not it was selected to the committee. Moreover the party can prove to everyone that its self-selection was done properly, by exhibiting the seed value along with the VRF proof.

Ideally, we would like to model cryptographic sortition as a "perfect" lottery functionality that chooses for each party whether or not it is included. (The functionality also tells everyone else about that choice, once the party in question authorizes the disclosure.) However, there are many setting (including ours) where the VRF implementation sketched above falls short of implementing this "perfect" functionality. The reason is that the adversary has some influence over the public input, making it possible for it to try many inputs until it finds one that it likes.

We therefore add to the model an initial phase where the adversary can reset the lottery arbitrarily many times, each time getting the choices corresponding to the parties that it controls. Eventually the adversary decides that it is happy with its choices, and then the lottery functionality is activated for everyone. This functionality is described in fig. 2.

2.5 Selective-Opening Security for PKE

In a selective opening attack on an encryption scheme [DNRS03, CFGN96], the adversary is given a set of n ciphertexts, encrypting n possibly related messages, and then chooses a t-size subset of the ciphertexts to be opened. Ciphertexts are opened either by corrupting senders (revealing their messages and encryption randomness) or by corrupting receivers (revealing their decryption keys). The scheme is SOA-secure if the adversary learns nothing about the unopened messages beyond

Cryptographic Sortition

Parameters are probability $p \in (0,1)$ and a set of N parties $P_1, \ldots, P_N$.

1. Initialization. For each $i = 1, \ldots, N$ choose a random independent bit b_i with $\Pr[b_i = 1] = p$. The adversary can repeatedly request to see all the bits for the corrupted parties, and can ask that all the bits will be chosen afresh. Once it is happy with its bits, the adversary can end this phase and move to Phase 2.

2a. Lottery. Once initialization ends, every party P_i can ask for its state, getting the bit b_i.

2b. Verification. All parties begin in *private mode*, and any party can ask at any time for its mode to be changed to *public mode*. A party P_i can ask for the state of any other party P_j, getting $\perp$ if P_j is still in private mode or the bit b_j if P_j is in public mode.

Figure 2: The cryptographic sortition functionality.

what's implied by the opened ones. This requirement can be formulated using either simulation or indistinguishability [BHY09, BDWY12, HPW15]. Unfortunately, neither formulation follows from semantic security [GM84, BHY09, BDWY12, HRW16].

Our protocol uses an encryption scheme that has selective opening security for receiver corruptions, and the weaker indistinguishability-based notion suffices for our purposes.[2] We follow Hazay et al.'s definitions of indistinguishability-based receiver-selective-opening security (RIND-SO) [HPW15], which build on [DNRS03, BDWY12].

In the RIND-SO security game, the adversary sees a vector of ciphertexts, encrypting messages that are drawn from some distribution $\mathcal{D}$. It obtains the opening of a selected subset of them (by obtaining secret keys), then receives from the challenger either the actual remaining plaintexts, or fake remaining plaintexts that are drawn afresh from $\mathcal{D}$ *conditioned on the opened plaintexts*. (This game requires that $\mathcal{D}$ be efficiently resamplable [BHK12], namely it should be feasible to draw from $\mathcal{D}$ conditioned on the opened plaintexts.)

Definition 2.1 (Efficiently Resamplable Distribution)**.** *Let $k, n > 0$ and let $\mathcal{D}$ be a distribution over $(\{0,1\}^k)^n$. We say that $\mathcal{D}$ is efficiently resamplable if there is a PPT algorithm $\mathsf{Resamp}_{\mathcal{D}}$ such that for any $\mathcal{I} \subset [n]$ and any partial vector $\mathbf{m}'_{\mathcal{I}}$ consisting of $|\mathcal{I}|$ k-bit strings, $\mathsf{Resamp}_{\mathcal{D}}(\mathbf{m}'_{\mathcal{I}})$ returns a vector $\mathbf{m}$ sampled from $\mathcal{D}|\mathbf{m}'_{\mathcal{I}}$ – i.e., $\mathbf{m}$ is sampled from $\mathcal{D}$ conditioned on $\mathbf{m}_{\mathcal{I}} = \mathbf{m}'_{\mathcal{I}}$.*

Definition 2.2 (RIND-SO Security)**.** *For a PKE scheme $\mathrm{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, security parameter λ, and a stateful PPT adversary $\mathcal{A}$, the RIND-SO game $\mathrm{Exp}^{\mathrm{rind-so}}_{\mathrm{PKE}}(\mathcal{A}, \lambda)$ is as follows.*

[2] Note that dealing with sender corruptions is trivial in our setting, as we anyway need to assume secure erasure because proactive security is impossible without it.

1. $(\mathbf{pk}, \mathbf{sk}) = (pk_i, sk_i)_{i\in[n]} \leftarrow (\mathsf{Gen}(1^\lambda))_{i\in[n]}$
2. $(\mathcal{D}, \mathsf{Resamp}_\mathcal{D}, \text{state}_1) \leftarrow \mathcal{A}(\mathbf{pk})$
3. $\mathbf{m} = (m_i)_{i\in[n]} \leftarrow \mathcal{D}$
4. $\mathbf{c} = (c_i)_{i\in[n]} \leftarrow (\mathsf{Enc}_{pk_i}(m_i; \$))_{i\in[n]}$
5. $(\mathcal{I}, \text{state}_2) \leftarrow \mathcal{A}(\mathbf{c}, \text{state}_1)$
6. $\mathbf{m}' \leftarrow \mathsf{Resamp}_\mathcal{D}(\mathbf{m}_\mathcal{I})$
7. $b \leftarrow \{0,1\}, \mathbf{m}^* \leftarrow \begin{cases} \mathbf{m}' & \text{if } b = 0 \\ \mathbf{m} & \text{if } b = 1 \end{cases}$
8. $b' \leftarrow \mathcal{A}(\mathbf{sk}_\mathcal{I}, \mathbf{m}^*, \text{state}_2)$

The advantage of the adversary $\mathcal{A}$ is $2 \cdot |\Pr[b = b'] - \frac{1}{2}|$. We say that the scheme is RIND-SO secure if every PPT $\mathcal{A}$ only has advantage negligible in λ.

Constructions using complexity leveraging. While not following from standard semantic security (against CPA), selective-opening security can be obtained from exponentially CPA-secure encryption via complexity leveraging. Starting with an encryption scheme that has (at least) subexponential security, one sets the security parameter λ relative to t so that the IND-CPA adversary's advantage is $\exp(-\omega(t \log \lambda))$. With comparatively large probability $\exp(-O(t \log \lambda))$, the simulator will correctly guess which subset of t ciphertexts the adversary will open, and will have no trouble opening them. Accordingly, the adversary's advantage must also be $\exp(-\omega(t \log \lambda))$.

Constructions from non-committing encryption. Encryption schemes with selective-opening security also can be built from receiver-non-committing encryption (RNCE) [CFGN96]. In a receiver-non-committing encryption scheme, there is a simulator that can generate fake public keys and ciphertexts, computationally indistinguishable from real ones, such that later the simulator can open the ciphertext to any given message (by providing an appropriate secret key). Nielsen [Nie02] used a counting argument to show that an RNCE scheme must have secret-key at least as long as the total size of plainetxts that are encrypted to it. However, Hazay et al. [HPW15] showed that RIND-SO security can be obtained from a weaker "tweaked" notion of RNCE, and that a construction due to Canetti et al. [CHK05] achieves the desired notion under the Decision-Composite-residuosity (DCR) assumption.

2.6 Anonymous Public Key Encryption

For our solution we need to use anonymous public-key encryption [BBDP01] to hide the link between parties and their (ephemeral) keys. Namely, a ciphertext should not betray the public key that was used to generate it. An anonymous PKE has the same key-generation, encryption, and decryption routines as a standard PKE, and we need the same semantic-security guarantees (i.e. CPA-security). In addition we need the following anonymity property:

Definition 2.3 (Anonymity [BBDP01])**.** *A PKE scheme $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is said to be anonymous if ciphertexts cannot be tied to the public key that was used to generate them. This is formulated by the following game, between a challenger and the adversary:*

1. *The challenger runs the key generation twice to get* $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Gen}(1^\lambda, \$)$ *for* $i = 0, 1$*, and sends* $\mathsf{pk}_0, \mathsf{pk}_1$ *to the adversary.*

2. *The adversary responds with a plaintext message* m*.* [3]

3. *The challenger chooses a secret bit* b*, encrypts* m *relative to* pk_b *to get* $\mathsf{ct} \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m)$*, and sends* ct *to the adversary.*

4. *The adversary outputs a guess* b' *for the bit* b*.*

The advantage of the adversary is $2 \cdot |\Pr[b = b'] - \frac{1}{2}|$*, and the scheme is anonymous if polynomial-time adversaries only have negligible advantage (in the security parameter).*

It is well known that most DL-based schemes and most LWE-based schemes are anonymous, and there are many variations of factoring-based schemes that are also anonymous. As we mentioned in the introduction, in our setting we use the anonymous PKE in a setting with selective opening, raising the question of whether Definition 2.3 implies anonymity also in that setting. Neither defining the question nor answering it is straightforward, we provide partial treatment in appendix B.1, giving "strong evidence" that Definition 2.3 is indeed sufficient even against dynamic corruptions.

2.7 Non-Interactive Zero-Knowledge Proofs

Let L be a language defined by the polynomial-time-computable relation $\mathcal{R}$. That is, $\mathcal{R}$ is a subset of $\{0,1\}^* \times \{0,1\}^*$ such that membership of (x, w) in $\mathcal{R}$ can be decided in time polynomial in $|x|$, and $L = \{x | \exists w : (x, w) \in \mathcal{R}\}$.

Definition 2.4 (Non-Interactive Zero-Knowledge Argument System)**.** *A non-interactive zero-knowledge argument system for an NP-language* L *with relation* R *consists of PPT algorithms* $(\mathcal{CRS}, \mathcal{P}, \mathcal{V})$ *with the following properties:*

- *Completeness: For every* $(x, w) \in \mathcal{R}$*, it holds that:*

$$\Pr\left[\sigma \leftarrow \mathcal{CRS}(1^\lambda); \mathcal{V}(\sigma, x, \mathcal{P}(\sigma, x, w)) = 1\right] = 1.$$

- *Soundness: For every PPT function* $f : \{0,1\}^{poly(\lambda)} \to \{0,1\}^\lambda \setminus L$ *and all PPT algorithms* $\mathcal{P}^*$*, there exists a negligible function* ν *such that for all* λ*:*

$$\Pr\left[\sigma \leftarrow \mathcal{CRS}(1^\lambda); \mathcal{V}^{\mathcal{O}}(\sigma, f(\sigma), {\mathcal{P}^*}^{\mathcal{O}}(\sigma)) = 1\right] < \nu(\lambda)$$

  where $\mathcal{O} : \{0,1\}^* \to \{0,1\}^\lambda$ *is a random function.*

- *Zero-Knowledge: For all PPT adversaries* $\mathcal{A}$*, there exists a PPT simulator* $\mathcal{S}$ *and a negligible function* ν *such that for all* λ*:*

$$\left|\Pr\left[\sigma \leftarrow \mathcal{CRS}(1^\lambda); \mathcal{A}^{\mathcal{P}(\sigma,x,w)}(1^\lambda, \sigma) = 1\right] - \Pr\left[\sigma \leftarrow \mathcal{CRS}(1^\lambda); \mathcal{A}^{\mathcal{S}(\sigma,x)}(1^\lambda, \sigma) = 1\right]\right| < \nu(\lambda).$$

[3] This message need not be in the plaintext space relative to these keys. Note that in that case the anonymity property implies that the scheme could also "encrypt" things outside of its plaintext space (although the result may not be decryptable).

2.8 Instantiating the Building Blocks for Our Solution

As we sketched in the introduction, our solution uses two PKE schemes, external one for the long-term keys and internal one for the ephemeral keys. Denote these schemes by $\mathcal{E}_1$ (external) and $\mathcal{E}_2$ (internal), and denote their combination by $\mathcal{E}_3$. Namely, $\mathcal{E}_3$ uses long-term keys from $\mathcal{E}_1$, and encrypts a message by choosing an ephemeral key pair for $\mathcal{E}_2$, encrypting the ephemeral secret key by the long-term public key, and encrypting the message by the ephemeral public key. The properties of these schemes that we need (in addition to semantic security) are:

- $\mathcal{E}_1$ should be anonymous as per Definition 2.3.
- The combination $\mathcal{E}_3$ should be receiver-selective-opening secure (RIND-SO) as per Definition 2.2.

In addition we would like the internal scheme $\mathcal{E}_2$ to be "secret-sharing friendly", in the sense that it allow efficient NIZK proofs that multiple values encrypted under multiple keys lie on a low-degree polynomial.[4] Below we sketch some possible instantiations, more details are deferred to future versions of this manuscript.

DCR-based instantiation. One method of achieving the security properties is to use for $\mathcal{E}_1$ a scheme which is both anonymous and receiver-non-committing (RNCE) and for $\mathcal{E}_2$ a scheme which is just receiver-non-committing. This means that the combination $\mathcal{E}_3$ is also RNCE, and therefore RIND-SO.

Since we only need the indistinguishability notion RIND-SO, we can use the weaker notion of "tweaked" RNCE from [HPW15]. This method can be instantiated based on the decision-composite-residuosity (DCR) assumption. We begin with the DCR-based RNCE scheme of Canetti et al. [CHK05], and apply the usual anonymization methods for factoring-based scheme to make it also anonymous (e.g., add a random multiple of n, see [HT07]).

This instantiation is also reasonably sharing-friendly, we can have a secret holder provide a Pedersen commitment to its secret, and prove that the encrypted shares are consistent with the commitment. A detailed description of such a scheme including the necessary zero-knowledge proofs can be found in [LNR18, Sec. 6.2.4], and can be made non-interactive using the Fiat-Shamir heuristic.

DDH-based instantiation. A variation of the above can also be instantiated under DDH. In this variant, we roughly replace Shamir secret sharing with a Shamir-in-the-exponent sharing (hence the secret is a random group element g^s). This means that the share holders can recover g^s, but not s itself. This supports applications that recover an individual secret but may not suffice for more complex threshold functions. We can then use the DDH-based RCNE scheme from [CHK05], and since we do not expect to recover s itself then we do not have the limitation from [CHK05] of only encrypting short messages. This DDH-based scheme can be easily made anonymous, and also allow simple NIZK proofs via the Fiat-Shamir heuristic.

Unfortunately, the above approach does not work as described due to the external-internal structure of our PKE encryption. Indeed, while we don't need to recover s in the internal PKE, we do need to recover the secret key which is encrypted under the external scheme. We therefore

[4] The witness for such proof consists of the secret key for one of the keys and the encryption randomness for all the others.

replace the external-internal structure with a single anonymous PKE with randomizable keys, as described in appendix A. Such a scheme can be constructed from the DDH-based RCNE scheme in [CHK05] and it has all the properties that we need (with the limitation that share holders recover the secret g^s rather than the exponent s).

Lattice-based instantiation. A plausible direction for implementing our schemes with lattice-based cryptography is to use a variant of Regev encryption [Reg09]. This cryptosystem is clearly anonymous, and has reasonable (if somewhat long) zero-knowledge proofs. In that case, we can even pack many ephemeral keys and the ciphertext that are encrypted under them, using a PVW-like scheme [PVW08], and take advantage of the packing to make the proofs more efficient, as described by Lyubashevsky and Neven in [LN17, Sec. 3]. (While we are not aware of LWE-based constructions of RNCE, it seems very plausible that such constructions can be achived by adapting leakage-resilience techniques that are used for in this domain.)

Heuristic instantiations. To achieve a fully practical scheme, one may need to resort to some heuristic assumptions. One would start with external and internal schemes that have good NIZK proofs (with an anonymous external scheme), and assume that the composed scheme is RIND-SO secure. Such an assumption seems quite reasonable, as the only counter-example of a scheme which is semantically secure but insecure for encrypting Shamir shares is very contrived (and uses obfuscation) [HRW16]. Moreover the anonymity of the external scheme suggests that selective-opening attacks on the internal scheme are much harder to mount.[5]

3 Our Proactive Secret Sharing Solution

The core of our design is the handover protocol, a proactive secret-sharing protocol in which the committee holding a shares of a secret at period i transfers this secret in shared form to a new holding committee of period $i+1$. If we knew the identities of the period $(i+1)$ holding committee, then they could use existing proactive secret-sharing mechanisms to transfer the secret. However, in our setting the new committee members must remain anonymous, so prior solutions cannot be used. To overcome this problem we introduce an intermediary committee called the nominating committee, as described below.

We assume that each party in the network has a long-term public key for an anonymous PKE, which is known to all other parties. The period i holding committee holds shares for a t-of-c_i Shamir secret sharing of global secret σ, where c_i is the size of the committee. In addition, there are commitments to these shares that are publicly known by all. The protocol consists of two phases, a *nomination phase* and a *proactive phase*

The first step is a nomination phase, where a nominating committee nominates the period $i+1$ holding committee. The members of the nominating committee are self-selected by cryptographic sortition. For each seat on the nominating committee, the party holding that seat nominates one seat for the holding committee. The period $(i+1)$ holding committee therefore comprises of c_{i+1} seats, and each seat will hold one new share in the end of the protocol. Each nominator creates an ephemeral key pair for its nominee, and encrypts the ephemeral secret key under the long-term public key of that nominee. The ephemeral key pairs will be used to communicate with the

[5] Drawing an analogy, the above assumption seems at least as reasonable as making circular-security assumptions in the context of homomorphic encryption, which is a rather common practice.

new committee without knowing who its members are, thereby transferring the new shares to the period $(i+1)$ holding committee (and creating commitments for them). While the nominating committee knows the identity of the new members, most of nominators are honest and therefore most of the new holding committee members remain anonymous for the adversary. After all the ephemeral keys are broadcast, the protocol enters the proactive phase.

Recall that the ephemeral secret keys are encrypted under the long-term public keys of members of period $(i+1)$ holding committee. Hence only these members can decrypt ciphertexts that were encrypted under the corresponding ephemeral public key. Moreover since we use anonymous PKE, them committee members remains anonymous to all other parties.

It the end of this phase, the period $(i+1)$ holding committee will hold shares for a t-of-c_{i+1} Shamir secret sharing of global secret σ. Moreover, we will use ephemeral public keys and ciphertexts as commitments to the plaintext values that are encrypted in them, which determine the shares of the period $(i+1)$ holding committee member. This will later enable that member to use NIZK proofs to prove that it re-shared its shares correctly. We now provide more details of this protocol.

3.1 Nomination Phase

Self-selection. This phase begins with self-selection of a nominating committee using cryptographic sortition. For a party p with stake $N_{i+1,p}$ at the beginning of period $(i+1)$, the sortition tosses $N_{i+1,p}$ coins, each with HEAD probability C/N_{i+1} — where N_{i+1} is the total stake in this period and C is a system parameter. Party p gets as many seats on the nominating committee as the number of HEADs among its coins (hence the expected stake held by the entire nominating committee is C).

Nominating next holding committee. For each seat on the nominating committee that party p has, it chooses at random one of the N_{i+1} tokens and assigns to (the owner of) that token a seat on the period $(i+1)$ holding committee. Suppose that party q is assigned a seat. Let pk_q be its long-term public key for the anonymous PKE. For this seat, the nominating party p generates a new ephemeral key pair $(\mathsf{esk}, \mathsf{epk})$, uses the long-term key pk_q to encrypt esk, $\mathsf{ct} \leftarrow \mathsf{Enc}_{\mathsf{pk}_q}(\mathsf{esk})$, and posts $(\mathsf{epk}, \mathsf{ct})$ to the blockchain. Then party p posts a sortition proof that it indeed has that many seats on the nominating committee.

Once δ rounds have passed in period $(i+1)$, all the ephemeral keys and ciphertexts, along with the sortition proofs, will be visible on the blockchain. Let $(\mathsf{epk}_1, \mathsf{ct}_1), \ldots, (\mathsf{epk}_{c_{i+1}}, \mathsf{ct}_{c_{i+1}})$ be the pairs whose sortition proofs are valid, ordered lexicographically by public keys. We number the seats in period $(i+1)$ holding committee from 1 to c_{i+1}, such that the corresponding encrypted key pair for seat k is $(\mathsf{epk}_k, \mathsf{ct}_k)$. Note that the all honest parties have a consistent view of this list.

3.2 Proactive Phase

Previous-period committee members. We use a technique similar to [GRR98] to re-share the secret among the c_{i+1} seats on the period $(i+1)$ holding committee. Recall that the shares of period i holding committee define a degree-$(t-1)$ polynomial F_i with $F_i(0) = \sigma$. WLOG each seat j holds a share $F_i(j)$.

A party holding seat j on the period-i holding committee, with share $F_i(j)$, uses a t-of-c_{i+1} Shamir secret-sharing to reshare that value. Namely it chooses a random degree-$(t-1)$ polynomial

G_j such that $G_j(0) = F_i(j)$, and sets $\sigma_{j,k} = G_j(k)$ for each $k = 1, \ldots, c_{i+1}$. The party then encrypts each $G_j(k)$ under the k'th ephemeral public key, obtaining $\mathsf{ct}_{j,k} = \mathsf{Enc}_{\mathsf{epk}_k}(\sigma_{j,k})$. It will later post that ciphertext on the blockchain, for the k'th seat on the holding committee of period $i+1$ to use in computing its share of the global secret.

To ensure the integrity of this process we need a *publicly verifiable* secret sharing scheme [Sta96] to re-share the shares, and moreover it has to be non-interactive. To that end, we use the commitment that we have from the previous period to the shares $F_i(j)$, and a NIZK proof of the statement that the sub-shares are generated properly. Specifically, let us denote by com_j the commitment from the previous round to the share $F_i(j)$, and let $\mathsf{ct}_{j,1}, \ldots, \mathsf{ct}_{j,c_{i+1}}$ be the ciphertexts that the party holding seat j encrypted above. The party generates a NIZK proof for the statement that $(\mathsf{com}_j, \mathsf{ct}_{j,1}, \ldots, \mathsf{ct}_{j,c_{i+1}})$ are commitment/encryptions of values that lie on a degree-$(t-1)$ polynomial w.r.t evaluation points $(0, 1, \ldots, c_{i+1})$, respectively. Denote this proof by $\pi_{i,j}$.

In addition, if there is some higher-level application that needs the use of the global secret (e.g. to decrypt or sign anything), then party j also prepares the relevant messages for that application, and we denote them by $\mathsf{aux}_{i,j}$.

Importantly, before sending any message, party j on the period i holding committee chooses a new long-term key-pair (denoted $(\mathsf{sk}'_j, \mathsf{pk}'_j)$), and erases all the protocol secrets (including shares of the global secret and all the secret keys).

Then party j broadcasts a single message, consisting of its evaluation point j, all the ciphertexts $\mathsf{ct}_{j,k}$ that it encrypted, the NIZK proof $\pi_{i,j}$, its new long-term public key pk'_j, and whatever auxiliary messages $\mathsf{aux}_{i,j}$ it prepared for higher-level applications.

Every party in the system. Each party p with long-term key pair $(\mathsf{sk}_p, \mathsf{pk}_p)$ must iterate through all broadcast pairs $(\mathsf{epk}_k, \mathsf{ct}_k)$ with valid sortition proofs. For every pair $(\mathsf{epk}_k, \mathsf{ct}_k)$, party p tries to decrypt $\mathsf{esk}_k \leftarrow \mathsf{Dec}_{\mathsf{sk}_p}(\mathsf{ct}_k)$, and test whether it is the corresponding secret key for epk_k (e.g., by encrypting and decrypting several random messages). If this is successful, the party p owns seat k on the holding committee of period $i+1$.

Next-period committee members. If party p has any seat on the holding committee of period $i+1$, it waits for another δ rounds and collects all messages that are broadcast by period i holding committee members. It orders those messages that include a valid NIZK proof lexicographically, and chooses the first t of them, corresponding to evaluation points $j_1, \ldots, j_t$. (Once again we note that honest members of the period-$(i+1)$ holding committee will agree on these messages and their evaluation points.)

For each seat k that it holds, Party p collects all the ciphertexts in those t messages that were encrypted under the k'th ephemeral key, namely $\mathsf{ct}_{j_1,k}, \ldots, \mathsf{ct}_{j_t,k}$. It uses the ephemeral secret key that it recovered above to decrypt them, obtaining the values $\sigma_{j_1,k} = G_{j_1}(k)$ through $\sigma_{j_t,k} = G_{j_t}(k)$.

Let $\lambda_{j_1}, \ldots, \lambda_{j_t}$ be the Lagrange coefficients for evaluation points $j_1, \ldots, j_t$. (Namely, the λ's are coefficients such that for every degree-$(t-1)$ polynomial F it holds that $F(0) = \sum_{k=1}^{t} \lambda_{j_k} \cdot F(j_k)$.) The share of the global secret corresponding to seat k is then computed as

$$\sum_{j \in \{j_1, \ldots, j_t\}} \lambda_j \cdot \sigma_{j,k}.$$

Moreover, the ciphertexts $\mathsf{ct}_{j_1,k}, \ldots, \mathsf{ct}_{j_t,k}$ are kept and used as the commitment value to this share (with the decommitment information being the ephemeral secret key esk_k).

We note that if the ephemeral PKE is linearly-homomorphic, then we may be able to use a shorter commitment, namely $\mathsf{ct}^*_k = \sum_j \lambda_j \cdot \mathsf{ct}_{j,k}$, rather than all the individual $\mathsf{ct}_{j,k}$'s.

Crucially, we were able to transfer shares of the global secret to the members of the holding committee of period $i+1$, without having them send any messages. Hence most of them remain anonymous (from the point of view of the adversary), and cannot be targeted for corruption.

Correctness. To see that the values computed by the holding committee members are indeed shares of the global secret, let us define the degree-$(t-1)$ polynomial

$$F_{i+1} = \sum_{j \in \{j_1,\ldots,j_t\}} \lambda_j \cdot G_j,$$

where G_j is the polynomial chosen by the (holder of) the j'th seat on the period-i committee. Then on one hand we have

$$F_{i+1}(0) = \sum_{j \in \{j_1,\ldots,j_t\}} \lambda_j \cdot G_j(0) = \sum_{j \in \{j_1,\ldots,j_t\}} \lambda_j \cdot F_i(j) = F_i(0) = \sigma.$$

On the other hand, for each seat k on the holding committee of period $(i+1)$, we have

$$\sum_{j \in \{j_1,\ldots,j_t\}} \lambda_j \cdot \sigma_{j,k} = \sum_{j \in \{j_1,\ldots,j_t\}} \lambda_j \cdot G_j(k) = F_{i+1}(k).$$

3.3 Security Analysis

To prove the security of this protocol, we need to show that the holding committees hold shares of the global secret σ, while σ remains (pseudo)random from the adversary's point of view throughout the handover protocols in the various periods. Namely, we consider a game in which the adversary is given either the global secret σ or another random secret σ', and we need to show that it only has a negligible advantage in guessing which is the case. As usual, the proof involves a game between the adversary and a challenger, and a sequence of hybrids that are proven indistinguishable via reductions to the hiding and anonymity properties of the encryption schemes, or to the soundness and zero-knowledge properties of the NIZK proofs.

H_0**: The real protocol.** This is a game where the challenger plays the role of all the honest parties, and in particular knows the global secret and all the shares. The adversary gets the secret σ in this game.

H_1**: Soundness of NIZK proofs.** In the next hybrid, the challenger aborts if at any point the honest parties accept a proof from the adversary even though the encrypted quantities in question do not lie on a degree-t polynomial. The challenger can detect this because it knows all the shares and it sees everything that the honest parties see. Using the soundness of the NIZK proofs, we argue that the challenger only aborts with negligible probability.

H_2**: Zero-knowledge proofs.** Next the challenger uses the NIZK simulator to generate the honest-party proofs. Since it is zero-knowledge, the adversary cannot detect the difference.

H_3: **Anonymous PKE.** In this hybrid the challenger aborts if the holding committee contains t or more corrupted seats, or fewer than t honest seats. We use the anonymity property of the long-term PKE to argue that this happens only with a negligible probability. This argument has two steps, first we show that the adversary does not corrupt too many parties on the nominating committee, this follows directly from Chernoff bound since it has no information about honest members of the nominating committee. Then we consider the set A of parties that were nominated by honest members of the nominating committee. Since the nominating committee is mostly honest, then this set A is rather large, roughly of size $(1-f)c$ of the total c nominees.

We then argue that while the adversary's view contains information about this set (since the ephemeral keys are encrypted under their long-term public keys), the adversary still cannot target members of A for corruption due to the anonymity of the long-term PKE scheme. This argument, explored in appendix B.1, is far from simple. In essence we need to show that an anonymous PKE remains anonymous even against a selective-opening attack. The same does not hold for *secrecy* properties of the PKE, in that context it is known that semantically-secure PKE may fail to maintain secrecy against selective-opening attacks (see below).

In appendix B.1 we formulate the property of a PKE remaining anonymous under selective-opening together with a *conjecture* that any anonymous PKE is also anonymous under selective-opening. While we were not able to prove it yet, we do provide strong evidence in support of this conjecture. Specifically we consider *semi-adaptive* adversaries, that see all the public keys and ciphertexts and then decide on a set of parties to open "in one shot." This class already include all the problematic aspects of selective-opening for secrecy, as well as all the negative examples, yet we prove that any anonymous PKE is also anonymous against adversaries in this class. The analysis for the general case boils down to analyzing a somewhat complex combinatorial game. We believe that such a proof is possible, but leave it to future work.

H_4: **Secure encryption of shares.** In this hybrid honest parties switch to encrypting the other secret σ' rather than σ (but the adversary still gets σ). (Equivalently, we can have the adversary getting σ' and the parties encrypting σ.) We argue that the adversary cannot distinguish these hybrids by reduction to the hiding property of the long-term and the ephemeral PKE.

The heart of the argument here is in showing that given n public keys epk_j, and n ciphertexts ct_j that encrypt shares of a secret σ under these keys, an adversary corrupting less than t keys cannot distinguish σ from a random and independent σ'.

Here too we need to consider an adaptive adversary that can decide on the keys to corrupt after seeing the ciphertexts, and the hiding property *does not follow* from semantic security of the underlying scheme alone (see, e.g., [HRW16]). Instead, we need to assume that the encryption enjoys *selective-opening security* for receivers [HPW15], as discussed in section 2.5. See more details in Lemma B.2 in the appendix.

Finally we can undo the changes in hybrids H_3, H_2, arriving at a game where the adversary gets σ' rather than σ. The adversary's inability to distinguish these hybrids imply that the secret is (pseudo)random to it. Moreover, due to H_3 we also know that there are enough honest members in the holding committee to recover the secret when needed.

3.4 Keeping the Committees Mostly Honest

Below we analyze the parameters of our scheme, specifically the fraction of corrupted stake that it can withstand. Our analysis uses tail bounds for the binomial distribution, so we begin by stating some properties of these bounds in the regime of interest. Let $p \in (0,1)$ and let k, n be integers with $pn < k \leq n$, Our analysis is concerned with a setting where $p = o(1) \approx C/n$ for some parameter C, and we use following Chernoff bounds:

$$\begin{aligned} \Pr\left[\mathrm{Bin}(n,p) > pn(1+\epsilon)\right] &< \exp(-np\epsilon^2/(2+\epsilon)), \text{ and} \\ \Pr\left[\mathrm{Bin}(n,p) < pn(1-\epsilon)\right] &< \exp(-np\epsilon^2/2). \end{aligned} \tag{1}$$

Below we argue that the holding committee contains less than t corrupted seats and at least t honest seats (except with insignificant probability). Let N denote the total stake, and let f be the fraction of stake controlled by the adversary. That is, we assume that the adversary controls parties that hold no more than fN tokens.

In this analysis we ignore computational issues such as anonymity against selective opening that was mentioned above. We assume for simplicity that the adversary selects the keys to open without any information about membership in the nominating- and holding-committees. In this information-theoretic analysis we can make the following simplifying assumptions:

- The adversary is computationally unbounded, but still can only reset the sortition a bounded number of times. Also it still needs to explicitly tell the challenger which parties are corrupted, and they are subject to the corruption budget of fN tokens.
- Corrupted members of the nominating committee choose only corrupted members for the holding committee, and
- The adversary corrupts all the $f \cdot N$ tokens at the beginning of the handover protocol and these remain unchanged throughout.

To see why we can make the last assumption (in this information-theoretic setting), observe that any change in the number of corrupted seats that happens because the adversary make later choice of who to corrupt, implies in particular that the adversary gained information about the not-yet-corrupted members of the holding committee.

If we let c denote the number of seats on the holding committee, ϕ denote the number of corrupted seats, and t denote the threshold, then we need $\phi < t$ (for secrecy) and $c - \phi \geq t$ (for liveness). We show below how to set the parameter C (that determines the expected committee size) and the threshold t so as to get secrecy and liveness with high probability.

Recalling that our model of sortition allows the adversary to reset its choice many times, the process that we want to analyze is as follows:

1. The adversary corrupts $f \cdot N$ tokens;
2. The adversary repeatedly resets the sortition until it is happy that enough of its corrupted tokens are selected to the nominating committee;
3. With the sortition so chosen, the honest (and corrupt) tokens are selected to the nominating committee;

4. Each member of the nominating committee selects a holding-committee token, with the honest ones selecting at random (and corrupted members always selecting other corrupted members).

Let k_1, k_2, k_3 be three security parameters for the analysis, as follows. We will assume the adversary can the adversary can reset the sortition functionality in the process above at most 2^{k_1} times.[6] We want to ensure secrecy except with probability 2^{-k_2} and liveness except with probability 2^{-k_3}. We will use parameters $\epsilon_1, \epsilon_2, \epsilon_3$, whose values we will fix later.

Let $B_1 = fC(1+\epsilon_1)$; B_1 represents the maximum tolerable number of corrupted tokens in the nominating committee. Let $B_2 = f(1-f)C(1+\epsilon_2)$; B_2 represents the number of additional corrupted tokens in the holding committee. We will set the threshold at $t = B_1 + B_2 + 1$. We then set $C, \epsilon_1, \epsilon_2, \epsilon_3$ to satisfy the following two conditions:

- Secrecy: $\Pr[\phi \geq t] \leq 2^{-k_2}$;
- Liveness: $\Pr[c - \phi < t] \leq 2^{-k_3}$.

The parameter ϵ_1. As described above, the adversary corrupts fN tokens, and then resets the sortition functionality at most 2^{k_1} times to try to get as many of these tokens selected to the nominating committee as it can. The number of corrupted tokens in the nominating committee *for each of these* 2^{k_1} *tries* is a binomial random variable $\mathrm{Bin}(n = fN, p = \frac{C}{N})$. We can set the parameters C and ϵ_1 large enough so as to ensure that

$$\Pr\left[\mathrm{Bin}(fN, \tfrac{C}{N}) > fC(1+\epsilon_1)\right] < 2^{-k_1-k_2-1},$$

in which case the union bound implies that

$$\Pr\left[\exists \text{ try with more than } fC(1+\epsilon_1) \text{ corrupted tokens selected}\right] < 2^{-k_2-1}.$$

Using Equation 1, a sufficient condition for ensuring the bound above is to set ϵ_1 and C large enough so as to get $\exp\left(-fN \cdot \frac{C}{N} \cdot \frac{{\epsilon_1}^2}{2+\epsilon_1}\right) < 2^{-k_1-k_2-1}$, or equivalently

$$C > \frac{(k_1+k_2+1)(2+\epsilon_1)\ln 2}{f{\epsilon_1}^2}. \tag{2}$$

The parameter ϵ_2. We next bound the number of additional corrupted tokens in the holding committee due to Step 4 above. Here we have a total of $(1-f)N$ honest tokens, each one is selected to the nominating committee with probability C/N and then each selected honest token chooses a corrupted token to the holding committee with probability f. Hence the number of additional corrupted tokens is a binomial random variable with $n = (1-f)N$ and $p = fC/N$ (and, unlike in the analysis of ϵ_1, this time the adversary gets only on attempt—there is no resetting, because the adversary cannot predict how sortition will select honest parties). The expected number of additional corrupted tokens is therefore $f(1-f)C$, and we get a high-probability bound on it by setting C and ϵ_2 large enough so as

$$\Pr\left[\mathrm{Bin}((1-f)N, \tfrac{fC}{N}) > f(1-f)C(1+\epsilon_2)\right] < 2^{-k_2-1}.$$

[6] Since in practice the adversary has very limited time in which to reset the sortition (e.g. less than 5 seconds in the Algorand network), it may be sufficient to use $k_1 = 64$.

Here too, we get a sufficient condition by applying Equation 1. For this we need to set ϵ_2 and C large enough to get $\exp\left(-(1-f)N \cdot \frac{fC}{N}) \cdot \frac{{\epsilon_2}^2}{2+\epsilon_2}\right) < 2^{-k_2-1}$, or equivalently

$$C > \frac{(k_2+1)(2+\epsilon_2)\ln 2}{f(1-f){\epsilon_2}^2}. \tag{3}$$

The parameter ϵ_3 and the liveness condition. The conditions from eqs. (2) and (3) ensure the secrecy condition except with probability 2^{-k_2}. It remains to set ϵ_3 and C to ensure liveness. Recall that the liveness condition holds as long as the number of honest tokens $(c-\phi)$ on the holding committee is at least t. Honest tokens come to the holding committee as follows: an honest token (out of $(1-f)N$ total) gets chosen to the nominating committee (with probability C/N), and then chooses an honest token (with probability $1-f$) to the holding committee. Thus, the number of honest tokens is a binomial random variable with $n=(1-f)N$ and $p=(1-f)C/N$. (Again, the adversary gets only one attempt, because the adversary cannot predict how sortition will select honest parties, so resetting doesn't help.) Since the expected value of this random variable is $(1-f)^2C$, it is sufficient to ensure that $t \le (1-f)^2C(1-\epsilon_3)$ for some $\epsilon_3 > 0$ such that

$$\Pr[\mathrm{Bin}((1-f)N, (1-f)C/N) < (1-f)^2C(1-\epsilon_3)] < 2^{-k_3}\,.$$

By Equation 1, this holds when $\exp\big(-(1-f)N \cdot (1-f)C/N \cdot {\epsilon_3}^2/2\big) < 2^{-k3}$, i.e.,

$$C > \frac{2k_3 \ln 2}{(\epsilon_3(1-f))^2}\,. \tag{4}$$

Recalling that our threshold was set to

$$t = B_1 + B_2 + 1 = fC(1+\epsilon_1) + f(1-f)C(1+\epsilon_2) + 1 \;=\; C\cdot\big((2+\epsilon_1+\epsilon_2)f - (1+\epsilon_2)f^2\big) + 1\,,$$

the condition $t \le (1-f)^2C(1-\epsilon_3)$ is equivalent to:

$$\epsilon_3 \le \frac{1-(4+\epsilon_1+\epsilon_2)f + (2+\epsilon_2)f^2 - \frac{1}{C}}{(1-f)^2} \tag{5}$$

Putting it all together. Given the fraction f of corrupted parties and the security parameters k_1, k_2, k_3, we need to find some setting of the other parameters $C, \epsilon_1, \epsilon_2, \epsilon_3$ that satisfies the bounds in eqs. (2) to (5). Clearly this is not possible when $f \ge 0.25$ (since in that case there is no $\epsilon_3 > 0$ that satisfies the condition from Equation 4). But for f which is bounded below 0.25, we can set the ϵ's to something like $(1-4f)/c$ for some moderate constant c, and get $C = O((k_1+k_2+k_3)/(1-4f)^2)$.

For example, let us choose security parameters $k_1 = 64$, and $k_2 = k_3 = 128$. The following table demonstrates the values of C that work different assumptions about the fraction f of corrupted parties (we omit the values of $\epsilon_1, \epsilon_2, \epsilon_3$ for conciseness; they can be easily obtained by a numerical search):

f	5%	10%	15%	20%	25%	30%
C	595	1040	2050	7759	26091	impossible

4 Applications

The solutions presented in this paper are broadly applicable, both in blockchain-specific contexts and for traditional uses of threshold cryptography. We describe a few example applications in this section. Perhaps the most natural application is for signing global blockchain state, such as accounts state or the content of particular blocks, as describe in the first two examples below. While this type of information can be validated by inspecting the blockchain itself, a threshold signature backed by blockchain consensus provides compact validation that saves the need to traverse the blockchain.[7] Later in this section we show how our techniques can turn a public blockchain into a "distributed trusted entity" that can be used as a service for general secure computation.

Blockchain Checkpointing. Blockchain "checkpoints" that validate the state of the blockchain at some points in time can be used to improve efficiency and security, particularly for initialization of new nodes joining the network. For example, Leung et al. [LSGZ19] described a "vault" system that essentially creates a sub-chain of checkpoints over an existing blockchain (e.g. every 1000 blocks), resulting in a dramatic savings in storage and computation. That solution, however, still has per-checkpoint storage cost proportional to the size of the committees that created that checkpoints, and moreover the blockchain verification time is still asymptotically linear.

Our technique enables a simpler solution, where the blockchain maintains a secret signature key and the holding committees use it to sign the blocks. This way, the checkpoints can be compressed to essentially a single signature that can be validated by anyone with the public key. In more details the blockchain is associated with a signature key-pair $(\mathsf{pk}_B, \mathsf{sk}_B)$, where pk_B is included in the genesis block and sk_B is maintained via our proactive secret sharing protocol. To generate a checkpoint at an agreed-upon round i, each member j of the current holding committee (that holds a share σ_j of a t-of-n Shamir sharing of sk_B) uses σ_j to produce a signature share $s_{i,j}$ on the current blockchain state, and propagates $s_{i,j}$ to the network. Subsequently, any blockchain user can combine t shares $\{s_{i,j}\}$ non-interactively and obtain a signature s_i on the state. Anyone can then validate the blockchain state just by verifying that one signature. (In particular, a user joining the network does not need to traverse the blockchain.)

Cross-Blockchain Token Bridge. Another attractive application in the blockchain context is cross-blockchain validation of transactions or other blockchain state (e.g., for cryptocurrency conversions, smart contracts that depend on two or more platforms, etc.). Naively, such token bridges require trusted parties that vouch for the state of one blockchain on the other. Using our technique, we can have the blockchain vouch for its own state, using the same signature mechanism from above.

As an example, suppose a user wants to transfer an asset C (e.g., a stablecoin) from a blockchain $\mathcal{A}$ to a blockchain $\mathcal{B}$. Blockchain $\mathcal{A}$ would have an account associated with a signature public key pk_A, and the secret key sk_A would be distributively managed by our secret sharing protocol. To transfer an asset C, a user would send it to an account managed by pk_A. The asset would be locked, and the user can obtain a short signature s under pk_A that indeed she locked the asset. The user can present the asset and signature to a smart contract running on blockchain $\mathcal{B}$. That contract

[7] In essence, our technique can turn statements about the state of the blockchain into NP statements with short witnesses, by having the holding committee sign them.

has pk_A hard-wired in it, it can verify that the asset is indeed locked on $\mathcal{A}$ just by checking the signature, and then mint the asset C to the user on $\mathcal{B}$.

Cryptography as a Service. The protocols in this paper can be used more generally to have the blockchain provide cryptographic services such as the storage of secrets [Sha79], (proactive) threshold signatures and decryption [DF89, HJJ+97, Rab98, Bol03], threshold PRFs/VPRFs/OPRFs, [NPR99, JKKX17, BLMR13], and more. We refer to this as "threshold cryptography as a service". While such services can be provided by more conventional systems of a few servers, here the guarantees are backed by the scale and security of blockchains and the post-compromise security provided by proactive re-sharing.

Perhaps the simplest example is for storage of secrets, such as the will example from the introduction. More generally the secret can be a symmetric key to encrypt/authenticate data, or a private key for a signature scheme, or a PRF key to support a threshold verifiable PRF. Threshold signature schemes can be deployed for purposes such as certification authorities, authentication of credentials, notarized services, etc. Another application is a verifiable randomness beacon, e.g., as used in [ACG+18, HMW18]. Yet another versatile primitive is threshold Oblivious PRFs which can be used to implement secure storage systems ranging from password-authenticated secrets (e.g., custodial services) [JKKX17] to cloud key management [JKR19], private information retrieval and search on encrypted data [FIPR05], oblivious pseudonyms [Leh19], password managers [SJKS17], and more.

MPC/obfuscation-as-a-service. An additional area that can make crucial use of threshold systems is multi-party computation (MPC). As it happens, our handover protocol is similar in many ways to the information-theoretic multiplication protocol from [GRR98]. In the full version of this work we show how to use this observation to implement generic secure computation, letting the current committee pass to the next one the sum/product of two secrets (as opposed to just passing the individual secrets themselves). Hence the blockchain can carry out arbitrary computation on behalf of its clients, without leaking anything but the end result. In effect, it lets the blockchain act as a trusted party.

A particularly powerful form of MPC-as-a-service is using threshold decryption of homomorphic encryption [BGG+18], which would enable applications akin to program obfuscation: Clients can encrypt their programs, anyone could apply these encrypted programs to arbitrary inputs, and the blockchain could decrypt the result (when accompanied by appropriate proofs). More limited in scope but with more practical implementations, threshold decryption of linearly-homomorphic encryption enable varied applications such as private set intersection [FNP04], asset management and fraud prevention [GKB+19], and many more.

References

[ACG+18] Avi Asayag, Gad Cohen, Ido Grayevsky, Maya Leshkowitz, Ori Rottenstreich, Ronen Tamari, and David Yakira. Helix: A scalable and fair consensus algorithm resistant to ordering manipulation. *IACR Cryptology ePrint Archive*, 2018.

[BBDP01] Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval. Key-privacy in public-key encryption. In *ASIACRYPT*, 2001.

[BDLO15] Joshua Baron, Karim El Defrawy, Joshua Lampkins, and Rafail Ostrovsky. Communication-optimal proactive secret sharing for dynamic groups. In *ACNS*, 2015. https://eprint.iacr.org/2015/304.

[BDWY12] Mihir Bellare, Rafael Dowsley, Brent Waters, and Scott Yilek. Standard security does not imply security against selective-opening. In *EUROCRYPT*, 2012.

[BGG+18] Dan Boneh, Rosario Gennaro, Steven Goldfeder, Aayush Jain, Sam Kim, Peter MR Rasmussen, and Amit Sahai. Threshold cryptosystems from threshold fully homomorphic encryption. In *CRYPTO*, 2018. https://eprint.iacr.org/2017/956.pdf.

[BGI+12] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 2012.

[BHK12] Florian Böhl, Dennis Hofheinz, and Daniel Kraschewski. On definitions of selective opening security. In *International Workshop on Public Key Cryptography*, 2012.

[BHY09] Mihir Bellare, Dennis Hofheinz, and Scott Yilek. Possibility and impossibility results for encryption and commitment secure under selective opening. In *EUROCRYPT*, 2009.

[Bla79] G. Blakley. Safeguarding cryptographic keys. In *Proceedings of the AFIPS Conference*, 1979.

[BLMR13] Dan Boneh, Kevin Lewi, Hart William Montgomery, and Ananth Raghunathan. Key homomorphic prfs and their applications. In *CRYPTO*, 2013. IACR Cryptology ePrint Archive 2015: 220 (2015).

[Bol03] Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In *PKC*, 2003.

[CFGN96] Ran Canetti, Uri Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In *STOC*, 1996.

[CH94] Ran Canetti and Amir Herzberg. Maintaining security in the presence of transient faults. In *CRYPTO*, 1994.

[CHK05] Ran Canetti, Shai Halevi, and Jonathan Katz. Adaptively-secure, non-interactive public-key encryption. In *TCC*, 2005.

[CM19] Jing Chen and Silvio Micali. Algorand: A secure and efficient distributed ledger. *Theor. Comput. Sci.*, 2019.

[CZK+19] Raymond Cheng, Fan Zhang, Jernej Kos, Warren He, Nicholas Hynes, Noah M. Johnson, Ari Juels, Andrew Miller, and Dawn Song. Ekiden: A platform for confidentiality-preserving, trustworthy, and performant smart contracts. In *IEEE European Symposium on Security and Privacy, EuroS&P*, 2019.

[DF89] Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In *CRYPTO*, 1989.

[DGGK10] Shlomi Dolev, Juan A. Garay, Niv Gilboa, and Vladimir Kolesnikov. Brief announcement: swarming secrets. In *PODC*, 2010.

[DJ97] Yvo Desmedt and Sushil Jajodia. Redistributing secret shares to new access structures and its applications. In *Technical Report*, 1997.

[DNRS03] Cynthia Dwork, Moni Naor, Omer Reingold, and Larry Stockmeyer. Magic functions: In memoriam: Bernard m. dwork 1923–1998. *Journal of the ACM (JACM)*, 2003.

[FIPR05] Michael J. Freedman, Yuval Ishai, Benny Pinkas, and Omer Reingold. Keyword search and oblivious pseudorandom functions. In *TCC*, 2005.

[FNP04] Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In *EUROCRYPT*, 2004.

[GG17] Rishab Goyal and Vipul Goyal. Overcoming cryptographic impossibility results using blockchains. In *TCC*, 2017.

[GGH+16] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. *SIAM J. Comput.*, 2016.

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In *STOC*, 2013.

[GKB+19] Hasini Gunasinghe, Ashish Kundu, Elisa Bertino, Hugo Krawczyk, Suresh Chari, Kapil Singh, and Dong Su. Prividex: Privacy preserving and secure exchange of digital identity assets. In *WWW*, 2019.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of computer and system sciences*, 1984.

[GM14] Spencer Greenberg and Mehryar Mohri. Tight lower bound on the probability of a binomial exceeding its expectation. *Statistics & Probability Letters*, 86:91 – 98, 2014.

[GRR98] Rosario Gennaro, Michael O. Rabin, and Tal Rabin. Simplified VSS and fast-track multiparty computations with applications to threshold cryptography. In Brian A. Coan and Yehuda Afek, editors, *Proceedings of the Seventeenth Annual ACM Symposium on Principles of Distributed Computing, PODC '98, Puerto Vallarta, Mexico, June 28 - July 2, 1998*, pages 101–111. ACM, 1998.

[HJJ+97] Amir Herzberg, Markus Jakobsson, Stanislaw Jarecki, Hugo Krawczyk, and Moti Yung. Proactive public key and signature systems. In *CCS*, 1997.

[HJKY95] Amir Herzberg, Stanislaw Jarecki, Hugo Krawczyk, and Moti Yung. Proactive secret sharing or: How to cope with perpetual leakage. In *CRYPTO*, 1995.

[HMW18] Timo Hanke, Mahnush Movahedi, and Dominic Williams. DFINITY technology overview series, consensus system. *CoRR*, 2018.

[HPW15] Carmit Hazay, Arpita Patra, and Bogdan Warinschi. Selective opening security for receivers. In *International Conference on the Theory and Application of Cryptology and Information Security*, 2015.

[HRW16] Dennis Hofheinz, Vanishree Rao, and Daniel Wichs. Standard security does not imply indistinguishability under selective opening. In *TCC (B2)*, 2016.

[HT07] Ryotaro Hayashi and Keisuke Tanaka. Anonymity on paillier's trap-door permutation. In *ACISP*, 2007.

[JKKX17] Stanislaw Jarecki, Aggelos Kiayias, Hugo Krawczyk, and Jiayu Xu. TOPPSS: cost-minimal password-protected secret sharing based on threshold OPRF. In *ACNS*, 2017.

[JKR19] Stanislaw Jarecki, Hugo Krawczyk, and Jason K. Resch. Updatable oblivious key management for storage systems. In *CCS*, 2019.

[KAS+18] Eleftherios Kokoris-Kogias, Enis Ceyhun Alp, Sandra Deepthy Siby, Nicolas Gailly, Philipp Jovanovic, Linus Gasser, and Bryan Ford. Verifiable management of private data under byzantine failures. *IACR Cryptology ePrint Archive*, 2018.

[Leh19] Anja Lehmann. Scrambledb: Oblivious (chameleon) pseudonymization-as-a-service. *PoPETs*, 2019.

[LN17] Vadim Lyubashevsky and Gregory Neven. One-shot verifiable encryption from lattices. In *EUROCRYPT*, 2017.

[LNR18] Yehuda Lindell, Ariel Nof, and Samuel Ranellucci. Fast secure multiparty ECDSA with practical distributed key generation and applications to cryptocurrency custody. *IACR Cryptology ePrint Archive*, 2018.

[LSGZ19] Derek Leung, Adam Suhl, Yossi Gilad, and Nickolai Zeldovich. Vault: Fast bootstrapping for the algorand cryptocurrency. In *NDSS*, 2019.

[MRV99] Silvio Micali, Michael O. Rabin, and Salil P. Vadhan. Verifiable random functions. In *FOCS*, 1999.

[MXC+16] Andrew Miller, Yu Xia, Kyle Croman, Elaine Shi, and Dawn Song. The honey badger of BFT protocols. In *CCS*, 2016.

[MZW+19] Sai Krishna Deepak Maram, Fan Zhang, Lun Wang, Andrew Low, Yupeng Zhang, Ari Juels, and Dawn Song. CHURP: dynamic-committee proactive secret sharing, 2019. https://eprint.iacr.org/2019/017.

[Nie02] Jesper Buus Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In *CRYPTO*, 2002.

[NPR99] Moni Naor, Benny Pinkas, and Omer Reingold. Distributed pseudo-random functions and kdcs. In *EUROCRYPT*, 1999.

[OY91] Rafail Ostrovsky and Moti Yung. How to withstand mobile virus attacks (extended abstract). In *PODC*, 1991.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In *CRYPTO*, 2008.

[Rab98] Tal Rabin. A simplified approach to threshold and proactive RSA. In *CRYPTO*, 1998.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 2009.

[Sha79] Adi Shamir. How to share a secret. *Commun. ACM*, 1979.

[SJKS17] Maliheh Shirvanian, Stanislaw Jarecki, Hugo Krawczyk, and Nitesh Saxena. SPHINX: A password store that perfectly hides passwords from itself. In *ICDCS*, 2017.

[SLL08] David A. Schultz, Barbara Liskov, and Moses D. Liskov. Mobile proactive secret sharing. In *PODC*, 2008.

[SLL10] David A. Schultz, Barbara Liskov, and Moses D. Liskov. MPSS: mobile proactive secret sharing. *ACM Trans. Inf. Syst. Secur.*, 2010.

[Sta96] Markus Stadler. Publicly verifiable secret sharing. In *International Conference on the Theory and Applications of Cryptographic Techniques*, 1996.

[STY05] Nitesh Saxena, Gene Tsudik, and Jeong Hyun Yi. Efficient node admission for short-lived mobile ad hoc networks. In *ICNP*, 2005.

[WWW02] Theodore M. Wong, Chenxi Wang, and Jeannette M. Wing. Verifiable secret redistribution for archive system. In *SISW*, 2002.

[ZSvR05] Lidong Zhou, Fred B. Schneider, and Robbert van Renesse. APSS: proactive secret sharing in asynchronous systems. *ACM Trans. Inf. Syst. Secur.*, 2005.

A A Variant with Higher Resilience in a Weaker Adversary Model

As we explained in section 3, an adversary controlling f fraction of the stake can control roughly $2f$ fraction of the holding committee, so we need the threshold value to be $t_i > 2f \cdot c_i$, and we need need at least t_i honest parties to recover the secret. It follows that our scheme can only handle corruption ratio of $f \leq 1/4$.

One could hope to get a solution with higher resilient by having the nomination committee use their VRF keys to select their nominees, and prove that they did so properly. In that case, a corrupted member of the nominating committee is no longer free to select anyone it wants to the holding committee, so we can hope that it will be forced to select an honest member. As we explained in the introduction, however, in our mobile adversary model this extra restriction does not help: A corrupted member of the nominating committee can use its VRF to select a member of the holding committee, and if that member happens to be honest then the adversary can simply corrupt it right then and there.

To make use of this additional restriction on the nomination process, we must therefore assume that the adversary, while mobile, cannot move too quickly. Specifically, we must assume that a attempted corruption that begins during the i'th period cannot be completed until after the end of the handover protocol in period $i + 1$.

But for such a slow-moving adversary, we can use even a simpler protocol: The holding committee can simply self-select, then announce themselves publicly, and then use any proactive secret sharing protocol from the literature (for dynamic committees) to pass the secret from one holding committee to the next. Indeed, such a slow-moving adversary will not be able to corrupt the next holding committee until after they already passed the secret to the committee after them.

But is this slow-moving adversary model a realistic one? We claim that it is not: While fully corrupting a target may be a slow process, the adversary can quickly mount DDoS attacks on members of the holding committee if it knows who they are. A more realistic model will allow the adversary to cause a fail-stop failures instantly, while limiting the speed at which it can fully corrupt a party. In this model, it makes sense to require that the nominating committee use their VRFs to select their nominees, and then prove that they did it properly.

Observe that since we want to maintain the anonymity of the holding committee, then the nominating committee proofs cannot simply reveal their VRF values (as those will let anyone compute the identity of their nominee). Instead, each nominator will choose its ephemeral keys as in our protocol from section 3, then prove that "there exists a member j such that the ciphertext contains an encryption of the ephemeral secret key under the public key of member j, and moreover j is the member selected by my VRF". While this is an NP statement, it is not a short one anymore, indeed its length is linear in the number of parties in the system. Hence to get a scalable solution we need the nominating committee to use SNARKs to prove these statements.

Anonymous PKE with randomizable keys. But this is still not enough, in this variant so far the nominator *knows the ephemeral secret key of its nominee*, so in particular it can recover its share of the global secret just like the nominee itself can. We therefore must replace our use of anonymous encryption by a stronger primitive, that we call *anonymous PKE with key randomization*. Namely, we assume that given the long-term public key pk of some party, anyone can generate a derivative public-key pk' such that:

1. no one can recognize that pk' was generated for pk,
2. Given pk', anyone can encrypt a message which would be decryptable by the secret key of pk and we have the usual semantically security, *even against the party that generated* pk'.

It is not hard to see that anonymous PKE with key randomization can be constructed from DDH and LWE: Roughly the derivative key will be an encryption of zero under the long-term key, which can be utilized for encryption using the homomorphism of these cryptosystems.

Specifically, for the DDH-based variant we can use Elgamal encryption where the long-term public key is a pair $g, h = g^x$) and the corresponding secret key is x. To randomize this key, one chooses another random integer y and output the derivative key $(a, b) = (g^y, h^y)$. Clearly, under DDH (a, b) are pseudorandom even given (g, h), yielding the anonymity property that we need. On the other hand (a, b) is itself an Elgamal public key relative to secret key x, so it can be used for encryption.

For the LWE-based variant weuse Regev encryption, where the public key is a pseudorandom matrix A (with many more columns than rows), and the corresponding secret key is a vector $\vec{s} = (\vec{s}' | -1)$ such that $\vec{s}A = \vec{e} \pmod q$ with $\|\vec{e}\| \ll q$. To randomize the key, one chooses a low-norm quare matrix R (say over $\{0, \pm 1\}$) and output $A^* = A \times R \pmod q$.

Using the fact that A is pseudorandom and the leftover hash lemma, it is easy to see that A^* is pseudorandom even given A, yielding the anonymity property that we need. On the other hand

it still a valid public key relative to the same secret key $\vec{s}$, since

$$\vec{s}A^* = \vec{s}AR = \vec{e}R \text{ with } \|\vec{e}R\| \leq \|\vec{e}\| \cdot \|R\| \ll q.$$

Hence A^* can be used for encryption.

A.1 Reworking the Parameters

Assuming the protocol from above and a slow-moving adversary, what fraction of the holding committee can an adversary corrupt if it controls an f fraction of the overall stake? In this model, a corrupted member of the nominating committee cannot freely choose its nominee but must apply its VRF to see who it is, and also cannot corrupt it is it is not already corrupted. However, if the would-be nominee is honest then the corrupted nominator can just refrain from nominating altogether.

Under this attack scenario, if the overall fraction of corrupted parties is f and we have a nominating committee with c members of which $f \cdot c$ are corrupted, then the expected membership of the resulting holding committee would be $c \cdot (1 - f + f^2)$ (i.e., all the nominees of honest parties and only f fraction of the would-be nominees of corrupted parties). Of these members, we expect the number of corrupted parties to be $c \cdot f$ (since each nominator has an f fraction chance of hitting a corrupted nominee). Hence the fraction of corrupted parties in the holding committee would be about $f/(1 - f + f^2)$. Since we need this fraction to be strictly smaller than $1/2$, we get the constraint $\frac{f}{1-f+f^2} < \frac{1}{2}$, i.e., $f < \frac{3-\sqrt{5}}{2} \approx 0.38$, which is slightly better than the 0.25 resilience that we have with our main protocol. (Of course this is an inaccurate estimate since all these numbers are just expectations, but replacing expectations by high-probability bounds does not make a significant difference.)

B Proofs

B.1 Anonymous PKE and Corruptions on the Holding Committee

Recall that in the analysis of hybrid H_3, we considered an alternative game H_3' in which honest nominators choose at random two candidates for each seat on the holding committee, and then toss a coin to decide which of these candidates to actually nominate. This process defines two holding committees, the actual committee consisting of the nominees, and a "ghost" committee consisting of the candidates that were not nominated. (The seats chosen by corrupted nominators are included in both committees.)

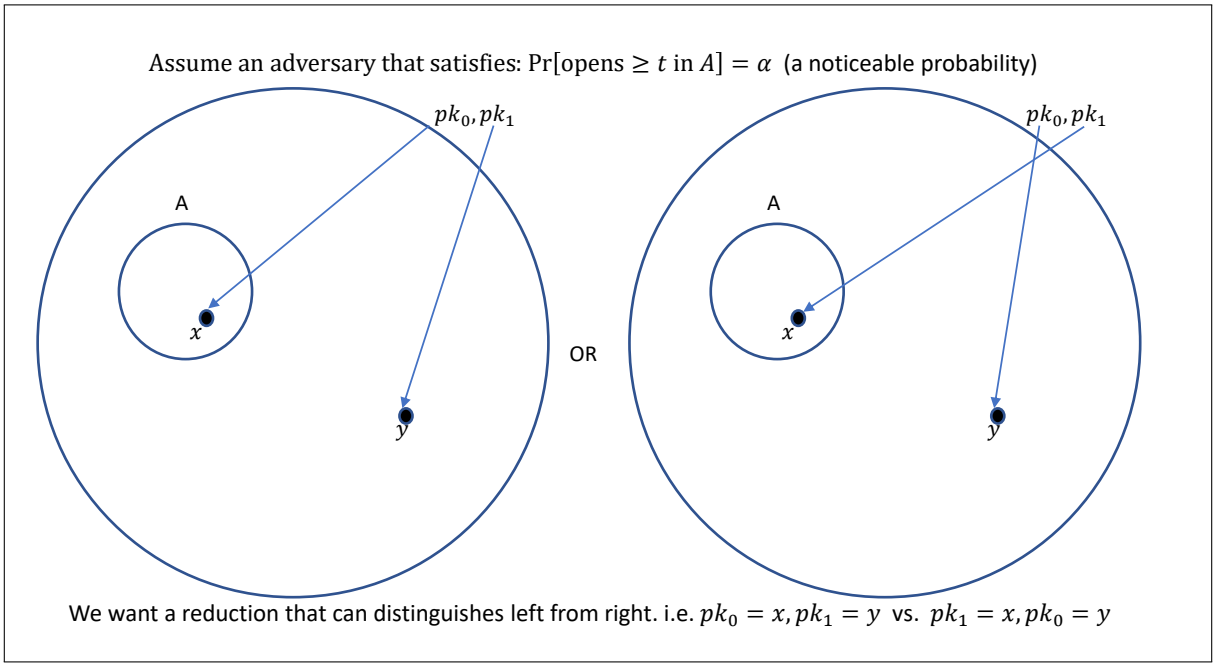
In section 3.4 we proved that with overwhelming probability, the ghost committee in H_3' contains at least t honest seats and fewer than t corrupted ones. Here we argue that the same holds also for the actual committee in H_3'(and therefore in H_3, because it is chosen identically), due to the anonymity property of the PKE that we use (for long-term keys).

The heart of the argument is showing that anonymous PKE remains anonymous also in the face of selective opening. Specifically, we consider that an adversary seeing n keys and $m \ll n$ ciphertexts encrypted under m of these keys, and then "opening" $\ell = fn$ of these keys by corrupting their owners and learning the secret keys (for some $f < 1/2$). We argue that for any constant $\epsilon > 0$, such an adversary only has a negligible (in m) probability of opening more than $(f + \epsilon)m$ of the keys under which ciphertext were encrypted. (In our setting, we would apply this to the set of

keys held by uncorrupted parties that were not selected by currupted members of the nominating committee.)

The setting that we care about is a challenger preparing n key-pairs, encrypting messages under m of them ($m \ll n$), with an anonymous PKE. Then the challenger is giving the adversary all the ciphertexts and keys, and the adversary can ask to open upto $\ell = f \cdot n$ of them. By "opening" a public key we mean the adversary asking for (and receiving) the corresponding secret key. (We can assume wlog that the adversary always opens exactly ℓ of them.)

Let A denote the set of public keys under which messages were encrypted, and we want to bound the probability that the adversary opens much more than $f \cdot m$ of the keys in A. Let $t^* = (1+\epsilon)fm$ be the bound that we use. Using Chernoff bound you can show that when the adversary opens an arbitrary subset D independent of A then it has only exponentially small probability of opening t keys in A. We want to use the anonymity property of the PKE to say that the same holds for any adversary strategy.



See the attached picture, showing a schematic view of the reduction. It gets two public keys $\mathsf{pk}_0, \mathsf{pk}_1$ and an encryption of some message under one of them, and it needs to guess which one. The problem is with the adaptive nature of the adversary: it can choose which keys to open after it sees all the keys and ciphertexts. When we try to reduce to anonymity, the reduction has to decide ahead of time where to put the public keys that it is challenged on, and will have to abort (or rewind the adversary) if the adversary asks to open these keys.

Conjecture 1. *If there is an efficient adversary that opens $\geq t^*$ keys in A with a noticeable probability α then the PKE in use is not anonymous.*

Here we prove the above conjecture for a restricted adversary that "opens" all the keys at once. That is, given the n public keys and m ciphertexts, the adversary outputs the set D of ℓ keys and gets all the secret keys for it at once. Note that this "semi-adaptive" adversary already exhibits all the problems with selective opening, in particular the negative examples (showing that semantic security does not imply security under selective opening) apply also to these restricted adversaries.

Lemma B.1. *If there is an efficient semi-adaptive adversary that opens at least $t^* = (1+\epsilon)fm$ keys in A with a noticeable probability α, then the PKE in use is not anonymous.*

Proof. Fix an adversary $\mathcal{A}$, denote by p_i the probability of $|D \cap A| = i$ for that adversary (for all $i = 0, 1, \ldots, m$), and assume that $\sum_{i \geq t^*} p_i = \alpha = 1/\mathsf{poly}(m)$.

We describe a reduction that uses this adversary to win the anonymous-PKE game with noticeable probability. The reduction has a parameter $\tau \leq m-1$, and it gets two keys $\mathsf{pk}_0, \mathsf{pk}_1$ and a ciphertext ct encrypted under one of them. It chooses $n-2$ more keys, selects a random subset $A' \subset [n]$ of size $m-1$, and encrypts messages under the keys in A'. The reduction then gives the adversary the n keys and m ciphertexts (in random order), and gets from the adversary the set D of ℓ keys to open. If $|A' \cap D| \geq \tau$ and in addition pk_1 is opened but pk_0 is not, then the reduction outputs 1. Otherwise the reduction outputs 0.

Let x denote the key under which the message is encrypted (pk_0 or pk_1), and y denote the other key (pk_0 or pk_1, respectively). The crux of the proof is showing that when the probability distribution $(p_0, p_1, \ldots, p_m)$ is far from an (n, m, ℓ)-hypergeometric distribution, there must exist some τ for which

$$\delta_\tau \stackrel{\text{def}}{=} \Pr[\text{reduction}_\tau \text{ outputs } 1 | x = \mathsf{pk}_1] - \Pr[\text{reduction}_\tau \text{ outputs } 1 | x = \mathsf{pk}_0]$$

is non-negligible (in m). Recall that the (n, m, ℓ)-hypergeometric distribution is $(p_0^*, p_1^*, \ldots, p_m^*)$ such that

$$p_i^* \stackrel{\text{def}}{=} \binom{i}{m}\binom{\ell - i}{n - m} / \binom{n}{\ell}.$$

Observe that when $x = \mathsf{pk}_1$, the reduction with τ outputs 1 if $|D \cap A| \geq \tau + 1$ (i.e., $\geq \tau$ for A' and one more for pk_1), and in addition $x = \mathsf{pk}_1 \in D$ and $y = \mathsf{pk}_0 \notin D$. Hence

$$\Pr[\text{reduction}_\tau \text{ outputs } 1 | x = \mathsf{pk}_1] = \sum_{i=\tau+1}^{m} p_i \cdot \frac{i}{m} \cdot \big(1 - \frac{\ell - i}{n - m}\big). \tag{6}$$

On the other hand when $x = \mathsf{pk}_0$, the reduction with τ outputs 1 if $|D \cap A| \geq \tau$, and in addition $y = \mathsf{pk}_1 \in D$ and $x = \mathsf{pk}_0 \notin D$. Hence

$$\Pr[\text{reduction}_\tau \text{ outputs } 1 | x = \mathsf{pk}_0] = \sum_{i=\tau}^{m} p_i \cdot \big(1 - \frac{i}{m}\big) \cdot \frac{\ell - i}{n - m}. \tag{7}$$

Let us denote $u_i = \frac{i}{m} \cdot (1 - \frac{\ell - i}{n-m})$ and $v_i = (1 - \frac{i}{m}) \cdot \frac{\ell - i}{n-m}$. From Eqs. 6 and 7 we have

$$\delta_\tau \;=\; -p_\tau v_\tau + \sum_{i=\tau+1}^{m} p_i(u_i - v_i) \;=\; \left(-p_\tau\big(1 - \frac{\tau}{m}\big) + \sum_{i=\tau+1}^{m} p_i\big(\frac{i}{m} - \frac{\ell}{n}\big)\right) \cdot \frac{m}{n-m}, \tag{8}$$

where the last equality follows because

$$u_i - v_i \;=\; \frac{i}{m} \cdot \frac{n - m - \ell + i}{n - m} - \frac{m - i}{m} \cdot \frac{\ell - i}{n - m} \;=\; \big(\frac{i}{m} - \frac{\ell}{n}\big) \cdot \frac{m}{n - m}.$$

Equation 8 yields a set of linear equations for expressing $\vec{\delta} = (\delta_0, \delta_1, \ldots, \delta_{m-1})$ in terms of $\vec{p} = (p_0, p_1, \ldots p_m)$. Let B be the $m \times (m+1)$ matrix representing these equations, namely $\vec{\delta} = \vec{p} \cdot B$.

We observe that the (n, m, ℓ)-hypergeometric distribution is the only probability distribution yielding $\vec{p}B = \vec{0}$. To see this, note that an adversary that chooses the set D at random among the n keys induces the (n, m, ℓ)-hypergeometric distribution on $|D \cap A|$, and for that adversary we have $\delta_\tau = 0$ for all τ. Moreover, it is easy to see that the matrix B above has full rank m. Hence the solution space for $\vec{p}B = \vec{0}$ is of the form $\rho \cdot \vec{p}^*$, where $\vec{p}^*$ is the (n, m, ℓ)-hypergeometric distribution and ρ is a scalar. Clearly, the only vector in this space whose entries sum up to 1 is $\vec{p}^*$ itself.

As the distribution $\vec{p}$ of the adversary $\mathcal{A}$ differs from $\vec{p}^*$ (since α is noticeable), we thus have $\vec{\delta} \neq 0$. We still need to prove, however, that $\vec{\delta}$ is *noticeably* far from zero. To that end, we look again at Equation 8 and give a name to the sum at the right-hand side. For every τ we denote:

$$\gamma_\tau \stackrel{\text{def}}{=} \sum_{i=\tau}^{m} p_i\left(\frac{i}{m} - \frac{\ell}{n}\right) = \sum_{i=\tau}^{m} p_i\left(\frac{i}{m} - f\right) \text{ and similarly } \gamma^*_\tau \stackrel{\text{def}}{=} \sum_{i=\tau}^{m} p^*_i\left(\frac{i}{m} - f\right).$$

Equation 8 can now be written as $\delta_\tau = \frac{m}{n-m}(\gamma_{\tau+1} - p_\tau(1 - \frac{\tau}{m}))$, and of course by definition we have $\gamma_\tau = p_\tau(\frac{\tau}{m} - f) + \gamma_{\tau+1}$. We similarly have $\gamma^*_\tau = p^*_\tau(\frac{\tau}{m} - f) + \gamma^*_{\tau+1}$, but here $\gamma^*_{\tau+1} - p^*_\tau(1 - \frac{\tau}{m}) = 0$. Note also that for $\tau \geq fm$ the term $\frac{\tau}{m} - f$ is non-negative. We next use the following two facts:

- By Chernoff bound, we have that $\gamma^*_{t^*} < \sum_{i \geq t^*} p^*_i$ is exponentially small in $\epsilon f \cdot m = \Theta(m)$.
- By our assumption on the adversary we have that

$$\gamma_{t^*} = \sum_{i \geq t^*} p_i\left(\frac{i}{m} - f\right) \geq \sum_{i \geq t^*} p_i\left(\frac{t^*}{m} - f\right) = \epsilon f \sum_{i \geq t^*} p_i = \epsilon f \alpha,$$

  which is non-negligible in m.

This means that γ_{t^*} is exponentially (in m) larger than $\gamma^*_{t^*}$, i.e. there exists some constant $\eta > 0$ such that $\gamma_{t^*} \geq (1 + \eta)^m \gamma^*_{t^*}$.

By the Claim B.1.1 below, we either have $p_{t^*-1} \geq (1+\eta)^m(1-\frac{\eta}{2})p^*_{t^*-1}$, or else $\delta_{t^*-1} > \frac{\eta m}{2(n-m)}\gamma_{t^*}$, which is non-negligible (in m). In the former case (of large p_{t^*-1}) we get

$$\begin{aligned}
\gamma_{t^*-1} &= p_{t^*-1}\left(\frac{t^*-1}{m} - f\right) + \gamma_{t^*} \geq (1+\eta)^m(1-\frac{\eta}{2})p^*_{t^*-1}\Big(\underbrace{\frac{t^*-1}{m} - f}_{>0}\Big) + (1+\eta)^m\gamma^*_{t^*} \\
&> (1+\eta)^m(1-\frac{\eta}{2})(p^*_{t^*-1}\left(\frac{t^*-1}{m} - f\right) + \gamma^*_{t^*}) \\
&= (1+\eta)^m(1-\frac{\eta}{2})\gamma^*_{t^*-1}.
\end{aligned}$$

In that case we can apply Claim B.1.1 again to conclude that either $p_{t^*-2} > (1+\eta)^m(1-\frac{\eta}{2})^2 p^*_{t^*-2}$ or else δ_{t^*-2} is non-negligible.

Repeating this process, we show by induction that either at least one of $\delta_{t^*-1}, \delta_{t^*-2}, \ldots, \delta_{fm}$ is non-negligible (in m), or else we have

$$\forall i \in [fm, t^* - 1],\ p_i > (1+\eta)^m(1-\frac{\eta}{2})^{t^*-i}.$$

But the last case cannot happen, since it means that the p_i's sum up to more than one. That is so because the hypergeometric distribution has probability at least 1/4 of exceeding the expected

value [GM14],[8] i.e., $\sum_{i \geq fm} p^*_i \geq 1/4$, and so

$$\begin{aligned}\sum_{i=0}^{m} p_i \;&\geq\; \sum_{i=fm}^{t^*-1} p_i + \sum_{i=t^*}^{m} p_i \;\geq\; \sum_{i=fm}^{t^*} (1+\eta)^m (1-\eta/2)^{t^*-i} p^*_i \;+\; (1+\eta)^m \sum_{i=t^*}^{m} p^*_i \\ &>\; (1+\eta)^m (1-\eta/2)^m \sum_{i \geq fm} p^*_i \;>\; \big(1+\eta/4\big)^m \cdot \frac{1}{4} > 1.\end{aligned}$$

This concludes the proof. □

Claim B.1.1. *For any $\tau \geq fm$, denote the ratio $R_{\tau+1} \stackrel{\text{def}}{=} \gamma_{\tau+1}/\gamma^*_{\tau+1}$ and let $\eta > 0$ be an arbitrary constant. Then either $p_\tau > R_{\tau+1}(1-\frac{\eta}{2})p^*_\tau$, or else $\delta_\tau \geq \frac{\eta m}{2(n-m)}\gamma_{\tau+1}$.*

Proof. Recall that for the hypergeometric distribution we have $\gamma^*_{\tau+1} = p^*_\tau(1-\frac{\tau}{m})$, and by definition of $R_{\tau+1}$'s we have $\gamma_{\tau+1} = R_{\tau+1}\gamma^*_{\tau+1}$. Assume that $p_\tau \leq R_{\tau+1}(1-\frac{\eta}{2})p^*_\tau$, and we need to show that $\delta_\tau \geq \frac{\eta m}{2(n-m)}\gamma_{\tau+1}$. By Equation 8 we have

$$\begin{aligned}\delta_\tau \cdot \frac{n-m}{m} \;&=\; \gamma_{\tau+1} - p_\tau(1-\frac{\tau}{m}) \;\geq\; R_{\tau+1}\gamma^*_{\tau+1} - R_{\tau+1}(1-\frac{\eta}{2})p^*_\tau(1-\frac{\tau}{m}) \\ &=\; R_{\tau+1}\big(\underbrace{\gamma^*_{\tau+1} - p^*_\tau(1-\frac{\tau}{m})}_{=0}\big) + \frac{\eta}{2}\cdot R_{\tau+1}\cdot p^*_\tau(1-\frac{\tau}{m}) \;=\; \frac{\eta}{2}\cdot R_{\tau+1}\gamma^*_{\tau+1} \;=\; \frac{\eta}{2}\cdot\gamma_{\tau+1}.\end{aligned}$$

Hence $\delta_\tau \geq \frac{\eta m}{2(n-m)}\gamma_{\tau+1}$, as needed. □

B.1.1 The fully-adaptive setting

In the fully adaptive setting, the adversary can open the keys one at a time, rather than all at once. Trying to apply the same reduction as above, the reduction no longer knows the full set of opened keys, since it has to abort once the adversary asks to open one of the two challenge ciphertexts (because it cannot answer that query). The reduction in this setting needs to decide on its output based on which of the two challenge keys was opened (if any), and how many of the keys in A' and outsidse of A' were opened at the time that the challenge key was requested. To prove the conjecture, we would have to show that for any adversary that has significant probability of opening more than t^* keys in A, there is a decision rule that yields significant advantage for the reduction.

As opposed to Lemma B.1 where the adversary's strategy can be characterized by the probabilities p_t of opening t keys in A, here the adversary has much more freedom in choosing not just if *but when* to open certain keys. Specifically, in the event that exactly t keys from A are opened, we can consider the keys outside of A as residing in $t+2$ buckets: Buckets $B_{t,0}, B_{t,1}, \ldots, B_{t,t}$ are the non-A keys that are opened after exactly $0, 1, \ldots, t$ keys from A, respectively, and $B_{t,\infty}$ are the non-A keys that are never opened. An adversary strategy is now characterized by the same p_t's as above, but in addition also by the expected sizes of the buckets $B_{t,j}$. (Hence we need $O(m^2)$ variables to describe the adversary, as opposed to $O(m)$ in the analysis in Lemma B.1.)

A plausible approach is to adopt the decision rule from Lemma B.1, where the reduction outputs 1 if pk_1 is opened (and pk_0 is not) and there are more than some threshold τ of other

[8] The proof in [GM14] is for the binomial distribution, but for our case of $m \ll n$ we get the same result upto a factor of $1 \pm o(1)$.

keys in A' that are opened before pk_1. Trying to analyze the advantage of δ_τ of this reduction (with parameter τ) we can express it in terms of the $O(m^2)$ variables from above. We get terms similar to above that correspond to the probability that only x or only y is opened, but also other terms corresponding to the case where both are opened, either x before y or vise versa. So far we were not able to analyze the resulting system of equations.

B.2 Selective-Opening Security

We now use the receiver-selective-opening of the PKE that we use to prove that the secret indeed remains secret. We note that we need *the composite scheme with both long-term and ephemeral keys* to be receiver-selective-opening-secure. Namely, if $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ and $\mathcal{E}' = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$ are the schemes that we use for long-term keys and ephemeral keys, respectively, then the scheme that we need to be receiver-selective-opening secure is the following combination $\mathcal{E}'' = (\mathsf{Gen}'', \mathsf{Enc}'', \mathsf{Dec}'')$:

$\mathsf{Gen}''(\$)$**.** This is just the long-term key-generation, $\mathsf{Gen}'' = \mathsf{Gen}$.

$\mathsf{Enc}''(\mathsf{pk}, m)$**.** To encrypt m under pk), run the ephemeral key generation to get $(\mathsf{sk}', \mathsf{pk}') \leftarrow \mathsf{Gen}'(\$)$. Then encrypt sk' under pk to get $\mathsf{ct}' \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{sk}')$, and encrypt m under pk' to get $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}', m)$. The cipehrtext is $(\mathsf{pk}', \mathsf{ct}', \mathsf{ct})$.

$\mathsf{Dec}''(\mathsf{sk}, (\mathsf{pk}', \mathsf{ct}', \mathsf{ct}))$**.** To decrypt we set $\mathsf{sk}' \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{ct}')$ and then outpur $m \leftarrow \mathsf{Dec}'(\mathsf{sk}', \mathsf{ct})$.

Assuming that the composite scheme $\mathcal{E}''$ is receiver-selected-opening secure as per Definition 2.2, we prove that hybrid H_3 and H_4 are indistinguishable to the avdersary. The proof is inductive, defining hybrids $H_3 = H_4^0, H_4^1, \ldots, H_4^\tau = H_4$, as follows: Recall that we have two global secrets σ_0, σ_1, and the adversary gets σ_0 and interacts with the protocol as in hybrid H_3. In H_4^i the first i handover protocols include encryptions of σ_1, and all the others include encryptions of σ_0.

Lemma B.2. *Assuming that the composite scheme is receiver-selected-opening secure as per Definition 2.2, hybrids $i-1$ and i are indistinguishable.*

Proof. (sketch) With all the periods but i fixed (and in particular the secrets σ_0, σ_1 determined by them), we let the input distribution $\mathcal{D}$ of the selective-opening attacker be as follows: First a bit σ is chosen at random, and the i'th handover protocol is run using secret σ_σ. (Recall that the NIZK proofs are simulated, hance we can run them even if the statements proved there are false.)

The attack proceeds as follows: the attacker chooses random σ_0, σ_1 and runs the H_4^i hybrid game with these two secrets, playing the challenger all the way to period $i-1$. At period i it uses the selective-opening game with input distribution $\mathcal{D}$ to get the keys and cipehrtexts, which it passes to the H_4^i adversary. Similarly it asks to corrupt the same keys that the H_4^i adversary does, and at the end of the game it gets the either the messages that were not compromised or freshly chosen messages from $\mathcal{D}$ conditioned on the compromized keys. Then it again runs the H_4^i hybrid game with these two secrets, playing the challenger from period $i+1$ and on, and gets the adversary's output bit b. If the messages that it saw at the end of the selective-opening attack correspond to the secret σ_1 then the attacker outputs the bit b, and otherwise it outputs the bit $1-b$. To analyze this attack, consider the following six possibilities:

E_1 is the event where the selective-opening attack game chooses the bit $\sigma = 0$ for the input distribution (so shares of σ_0 are encrypted), and the attacker gets the messages that were actually encrypted in the uncompromized cipehrtexts.

E_2 is the event where the selective-opening attack game chooses the bit $\sigma = 0$ for the input distribution (so shares of σ_0 are encrypted), and the attacker gets randomly chosen messages that happen to also correspond to a sharing of σ_0.

E_3 is the event where the selective-opening attack game chooses the bit $\sigma = 0$ for the input distribution (so shares of σ_0 are encrypted), and the attacker gets randomly chosen messages that happens to correspond to a sharing of the other secret σ_1.

E_4 is the event where the selective-opening attack game chooses the bit $\sigma = 1$ for the input distribution (so shares of σ_1 are encrypted), and the attacker gets randomly chosen messages that happens to correspond to a sharing of the other secret σ_0.

E_5 is the event where the selective-opening attack game chooses the bit $\sigma = 1$ for the input distribution (so shares of σ_1 are encrypted), and the attacker gets randomly chosen messages that happen to also correspond to a sharing of σ_1.

E_6 is the event where the selective-opening attack game chooses the bit $\sigma = 1$ for the input distribution (so shares of σ_1 are encrypted), and the attacker gets the messages that were actually encrypted in the uncompromized cipehrtexts.

Note that the view of the H_4^i-adversary is identical in cases E_1-E_3, and similarly in cases E_4-E_6. We denote by ϵ_1 the probability of the H_4^i-adversary outputting 1 in cases E_1-E_3, and the probability that it outputs 1 in cases E_4-E_6 is denoted ϵ_2. Hence the advantage of the H_4^i-adversary in distinguishing H_4^{i-1} from H_4^i is $a = \epsilon_1 - \epsilon_2$.

By construction, the probability that the selective-opening attacker outputs 1 conditioned on cases E_1, E_6 where it is given the messages that are actually encrypted in the uncompromised ciphertexts is $(\epsilon_1 + (1 - \epsilon_2))/2$. At the same time, the probability that the attacker outputs 1 conditioned on cases E_2-E_5 where it given instead freshly chosen messages is $\big(\epsilon_1 + (1 - \epsilon_1) + \epsilon_2 + (1 - \epsilon_2)\big)/4 = 1/2$. Hence, the advantage of the selective-opening attacker is exactly

$$(\epsilon_1 + 1 - \epsilon_2)/2 - 1/2 \;=\; (\epsilon_1 - \epsilon_2)/2 \;=\; a/2.$$

□