Proactive Secret Sharing without Erasures

Alexandru Cojocaru
University of Edinburgh
a.cojocaru@ed.ac.uk

Aggelos Kiayias
University of Edinburgh and IOG
aggelos.kiayias@ed.ac.uk

Yu Shen
The Hong Kong Polytechnic University
shenyu.tcv@gmail.com

Petros Wallden
University of Edinburgh
petros.wallden@ed.ac.uk

Abstract

Proactive secret-sharing (PSS) offers security for shared secrets in a setting of a *mobile* adversary which, over time, may corrupt the whole shareholder set. This remarkable property is achieved by having parties proactively and in a coordinated manner refresh their shares on a regular basis, while it assumes that the adversary never manages to corrupt more than a threshold number of parties between two consecutive share refresh operations.

A common assumption for achieving PSS is the ability of parties to securely erase their private state once they have performed the refresh operation. Motivated by the difficulty in the real world to ensure secure erasure, we investigate whether it is possible to achieve PSS without erasures. As in the classic model of computation it can be easily shown that PSS without erasures is impossible, we hence ask whether it is possible to achieve PSS via quantum computation, while still requiring only classical communication.

We answer the question in the affirmative by utilizing one-shot signatures and post-quantum classical extractable witness encryption. In the process of developing our result, we define and construct threshold one-shot decryption and make connections to quantum money with classical communication both of which may be of independent interest. Finally, we show how, by combining post-quantum secure functional witness encryption with our PSS, it is possible for the secret to be used without explicitly being reconstructed, something that paves the way towards proactively secure threshold cryptography without erasures.

An abridged version of this paper appears in *Proc. ASIACRYPT 2026*.

Contents

1 Introduction **3**
- 1.1 Technical Overview . . . 4
- 1.2 Related Work . . . 7

2 Preliminaries **8**
- 2.1 The Mobile Adversary Model . . . 9
- 2.2 Proactive Secret Sharing . . . 9
- 2.3 One-Shot Signatures . . . 12
- 2.4 Post-Quantum Cryptographic Tools . . . 13

3 On Necessity of Secure Erasures in Classical Setting **15**

4 One-Shot Decryption **15**
- 4.1 Two-Party One-shot Decryption . . . 15
- 4.2 Multi-Party One-shot Decryption . . . 17
- 4.3 Construction from One-shot Signatures . . . 19

5 Proactive Secret Sharing **21**
- 5.1 Constructions Against Passive Dynamic Mobile Adversary . . . 21
- 5.2 Constructions Against Active Mobile Adversary . . . 24

6 Secret Usability **27**

A Single-Decryptor Encryption **33**

B Preliminaries (Cont'd) **33**

1 Introduction

Secret sharing [Sha79, Bla79] is the fundamental cryptographic task of maintaining a secret distributed across a set of distinct parties so that (i) the secret can be reconstructed if all, or a sufficient number of, parties agree, and (ii) if some parties go "rogue," the confidentiality of the secret is still not compromised in any way. Secret sharing is the cornerstone of threshold cryptography [DF90] where additional protocols are utilized to obviate the need for reconstruction and allowing the shareholders to utilize their shares towards specific secret dependent computations without requiring ever the secret to be revealed.

When a secret is maintained in a shared state over an extended period of time, a natural consideration is that some of the shareholders may become compromised. It is the case moreover that, subsequently, they may detect this and manage to recover by reinstalling their software and erasing any injected malicious code returning to their system to the uncompromised state. The relevant question that arises in this context, is whether there is a way to still maintain the secret's confidentiality when, over time, adversarial corruption may even extend across the whole shareholder set. This important consideration was studied and resolved for the first time in the form of Proactive Secret Sharing (PSS) in [OY91, HJKY95]. In PSS, shareholders take proactive steps to refresh their private shares so that old compromised private state cannot be utilized effectively in the present to improve the adversary's position. In this way the timeline of the system is broken into epochs and the secret's confidentiality can be maintained as long as the number of compromises within an epoch remain below a threshold, while over time the number of compromises may exceed the threshold and in fact even extend across the whole set of shareholders.

A precondition to the ability to perform proactive secret sharing is the ability of parties to erase private state. Intuitively, this is needed as during the course of refreshing the secret, shareholders are required to erase the obsolete portion of their private state. In this way, a compromise in a current epoch cannot reveal information that can be combined with information obtained from past epoch compromises.

However, erasures are difficult in general to achieve: in fact, information in a computer system is typically replicated extensively due to fault tolerance considerations, so erasing a particular sequence of bits in a storage device does not guarantee that the same sequence has not being copied elsewhere during a prior computation that received that sequence as an input. Furthermore, even a deleted file in a storage device may be recovered in some circumstances depending on the low level physical properties of the device, cf. [Gut96], something that suggests that the act of *securely* deleting information from a computer system is in its own right a very challenging task.

The importance of erasures is further highlighted in more recent applications of PSS in the context of "You Only Speak Once" (YOSO) multiparty computation (MPC) [GHK+21], where a large population of participants wishes to perform an MPC with sub-quadratic communication so that only a small subset of the active participants at any time carries the private computation forward, despite the adversary's corruption threshold far exceeding the size of the subset. In such protocols, parties only speak once (because once they do, their identities leak to the adversary who may compromise them) and hence they have to immediately erase past secret state for fear of compromise. PSS has been found to be an invaluable building block for YOSO protocols [BGG+20, GHK+21]. Note that in this setting, it is crucial to consider an attacker who can bribe past committee members to reveal their keys. It follows that erasures are disincentivized: the rational choice for participants would be to maintain old key material so they can be sold for profit later.[1]

Given the above, it is natural to ask the following question.

[1] With foresight, our PSS construction would be the only way to provably thwart this attack vector in practice.

Is it possible to obviate the need for erasures altogether and achieve PSS in a setting where participants are never required to perform an erasure operation of their private state?

1.1 Technical Overview

We address comprehensively the question of proactive secret sharing without erasures. First we prove formally that in the classic model of computation, realizing PSS without erasures is impossible. Since we are not interested in a partial result that will still require erasures of some form or an idealized model of computation where some components are assumed to be capable of deletion, we look into adopting quantum computation techniques, while at the same time keeping the quantum capabilities of the parties involved at minimum. For this reason, in this work we show it is possible to achieve *proactive secret sharing without erasures* in the quantum setting, while *relying solely on classical communication.*

The motivation behind using quantum computation is a natural one. Erasure is conceptually tied to irreversibility. In an adversarial setting, we seek erasure mechanisms that (i) cannot be undone and (ii) can be certifiably verified by external parties, relying on physical laws rather than assumptions about honest behavior. Classical information does not offer such guarantees: deletion is inherently reversible from an information-theoretic perspective and can only be argued under behavioral assumptions.

In contrast, measurement of a quantum system is fundamentally irreversible. Moreover, quantum information theory provides a collection of techniques for certifying physical behavior, including proofs of quantumness and certifiable randomness [BCM+18], public-key quantum money [Aar09] and one-shot signatures [AGKZ20]. These developments combine the unclonability of quantum information with the destructive nature of measurement, making quantum systems a natural and well-motivated foundation for cryptographic notions of certified deletion or erasures.

Indeed, the problem of secure deletion has been previously studied in the quantum setting. Encryption with certified deletion has been studied in a long line of works [BI20, HMNY21, BK23, Por23] that introduce cryptographic primitives supporting certified deletion, where a party holding a quantum ciphertext can produce a classical certificate certifying that the plaintext cannot be recovered, even with unlimited computational power. Subsequently, the works of Bartusek and Raizes [BR24] and of Katz and Sela [KS24] extended these ideas to the setting of secret sharing, where a secret is split into quantum shares that can be verifiably destroyed, preventing reconstruction. Certified deletion primitives for secret sharing therefore enable parties to prove that quantum secret shares has been irreversibly erased.

While some techniques used in certified deletion — such as the no-cloning theorem, or irreversible transformations of quantum states — may show how information can be made unusable, certified deletion alone does not provide the core mechanism needed for proactive security.

In particular, in secret sharing with certified deletion, the dealer generates quantum shares; and by measuring using different bases, a shareholder can either acquire the classical share, or generate a proof that the share has been deleted — an "all-or-nothing" type of operation. In contrast, proactive security requires an additional mechanism beyond removing the current set of shares—parties must periodically acquire a new set of shares that maintains the same secret after reconstruction.

In the classical setting, this is achieved by a "handover" sub-protocol where share-holders first disperse their old shares using a fresh round of secret sharing, erase the old shares, and then reconstruct a new set of shares by exploiting some specific algebraic structure e.g., polynomial-based reconstruction in Shamir secret sharing [BGG+20]. In other words, parties have to generate

the new set of shares in a distributed manner. Now considering its quantum counterpart, emulating such procedure would require parties with quantum shares of a secret to generate in a distributed fashion a new set of quantum shares while destroying the old ones without leaving a trace.

Due to its all-or-nothing nature, certified deletion does not support in a straightforward manner such re-randomization of an existing encoding of the secret, nor the creation of new independent shares between different epochs—even more so, in our setting, where we are restricted to only use *classical communication*.

In the remainder of the section we describe our strategy to achieve proactive secret sharing without erasures while solely relying on classical communication.

Multi-party one-shot decryption. As a warmup, we first examine a primitive that we call *one-shot decryption* (OSD), which can also be of independent interest. In an OSD, a party publishes a classical public key and subsequently may receive multiple messages encrypted under that key over time. The task is to have *exactly* one of those ciphertexts decrypted with a *deterministic* decryption algorithm, while the rest of the ciphertexts cannot be decrypted even if the state of the receiver is revealed afterwards (as in a posterior corruption). Without erasures, it is easy to see the primitive is impossible to realize classically. It is also easy to see that the primitive is also infeasible in the quantum setting: indeed, the party could evaluate the quantum decryption circuit on a ciphertext, measure the register for the plaintext, and subsequently, due to the determinism of decryption, rewind the quantum computation so that it is capable of applying the quantum decryption operation again on a fresh ciphertext. What we show is that in the *threshold* setting we can achieve a threshold one-shot decryption that can withstand total posterior corruption, assuming that the number of corruptions, initially, are in the minority of the participant set.

The key idea of the construction is to use a post-quantum secure extractable witness encryption over a relation that requires a sufficient number (equal to the threshold) of *one-shot* signatures [AGKZ20] to be triggered. The public-key of the OSD will then consist of the collection of one-shot public-keys. Observe that the one-shot property of the OSS provides the posterior corruption security we are after.

Theorem (Informal, restated from Theorem 2). *Assuming one-shot signatures and post-quantum extractable witness encryption, there is a threshold public-key encryption scheme with one-shot decryption.*

Proactive secret sharing without secure erasures. We put forward three new proactive secret sharing protocols in different security models.

The first protocol, a conceptually simple construction inspired by threshold one-shot decryption described above, implements dynamic-committee PSS against a passive adversary who can access the internal states of corrupted parties, but has to otherwise follow the protocol (i.e., it learns all classical internal states and can perform any QPT quantum channel to its state and auxiliary registers, provided that before every prescribed protocol action there exists an efficient recovery map restoring the party's prescribed honest state).

At a very high level, in our construction, the dealer publishes a secret using (post-quantum) extractable witness encryption; and in each epoch, shares are a set of *one-shot-signature* (OSS) signing keys. The secret can be decrypted given multiple chains-of-signatures — a chain-of-signature is a list of one-shot signatures, where each one is verified under the previously-signed OSS public key in the chain (the first signature is verified under the first-epoch OSS public key, which is part of the witness-encryption statement). Each signature additionally signs a message indicating if parties wish to reconstruct the secret or not. During handover, parties broadcast their OSS signatures, thus destroying their old share in a publicly verifiable way, and transforming the OSS private key

of that signed public key to a new share of the secret. The secret can be retrieved when eventually all parties sign on the "open secret" message.

An alternative way to interpret the construction technique we put forward is to parallel it to a form of *quantum money* that uses only *classical communication* (cf. [AGKZ20]): the dealer creates a number of coins and distributes them to the shareholders. In every epoch, each shareholder creates a new account and receives their own coin from the previous epoch. Finally, the extractable witness encryption reveals the secret as long as a sufficient number of coins are "burned" and the evidence of that action is presented as the witness for the decryption.

This scheme tolerates all-but-one corruptions in every epoch, as the adversary is limited to extracting the secret from corrupted internal states.

Theorem (Informal, restated from Theorem 3). *Assuming one-shot signatures and post-quantum extractable witness encryption, there is a secure proactive secret sharing protocol that does* not *need secure erasures, against a passive adversary corrupting majority of the parties in every epoch.*

Observing that our first protocol fails to retain security if corrupted parties can actively destroy quantum states (in which case the protocol will prevent reconstruction if too many shares are not used as supposed), our next step is to modify the protocol so as to withstand active corruptions, resulting in a fully-fledged maliciously-secure PSS protocol tolerating minority corruptions in every epoch.

We make the following adjustments. The dealer still publishes her secret using extractable witness encryption, yet to a more elaborate relation where the witness is n vectors of OSS signatures, and the i-th vector certifies that parties in the i-th epoch unanimously sign on the public keys of the next committee in the $(i+1)$-th epoch, and sufficiently many OSS signatures in the last epoch sign on the "open secret" message. In other words, in order to decrypt the secret, one has to present a history of updates on the committee membership through one-shot-signatures. Specifically, during handover, parties first reach agreement over the next committee and their OSS public keys, and then all shareholders sign the set of next-epoch public keys (as well as indicating they are willing to reconstruct or not). Hence, for each epoch we get a vector of OSS signatures signing on the same next-epoch committee.

Theorem (Informal, restated from restated from Theorem 4). *Assuming one-shot signatures and post-quantum extractable witness encryption, there is a secure proactive secret sharing protocol that does* not *need secure erasures, against an active adversary that does not corrupt a majority of the parties in every epoch.*

On how to use the secret without directly reconstructing it. In most applications of threshold cryptography, the shared secret is not supposed to be reconstructed but instead used to compute some cryptographic function based on it, e.g., a digital signature algorithm or a decryption algorithm. It might be tempting to consider using post-quantum MPC on a classical functionality [ABG+21] on top of our PSS construction in order to meet this requirement. However one has to be careful, as a straightforward application of an MPC to our construction will create a "security hole" that a posterior corruption attacker can exploit: corrupting parties during the course of the MPC will enable the adversary to target the shared secret and extract it via posterior corruption by observing the old state of the parties during the MPC (since erasures are not allowed). Instead, in order to circumvent this problem, we utilize post-quantum functional witness encryption and arrive at the following result.

Theorem (Informal, restated from Theorem 5). *Assuming one-shot signatures and post-quantum functional witness encryption, there is a secure proactive secret sharing protocol that does* not *need secure erasures and additionally parties can evaluate any function F on the secret, without revealing the secret, against an active adversary that does not corrupt a majority of the parties in each epoch.*

We present the above theorem in the context of a lightweight security abstraction we term "secret usability," which captures the ability to evaluate functionalities on a proactively shared secret without reconstruction in the non-erasure setting and with resistance to posterior corruptions. Note that we opt to present our result in this way for illustratory purposes; we anticipate that focusing to specific threshold primitives, such as signatures or threshold encryption, may enable more efficient constructions or using weaker assumptions. While out of scope for the present work, this is an interesting direction for future work.

On the underlying cryptographic assumptions. Our constructions rely on strong cryptographic assumptions, notably post-quantum extractable witness encryption. Our primary goal is to demonstrate for the first time that proactive security without erasures is achievable, and to identify quantum cryptographic mechanisms that make this possible. In particular, our work reveals a conceptual connection between proactive security without erasures and quantum "one-shot" primitives, such as one-shot signatures, together with extractable witness encryption.

Currently, post-quantum extractable witness encryption is only understood from strong knowledge-type assumptions, and obtaining satisfactory instantiations remains an important open problem. An important direction left open by our work is whether the extractability requirements underlying our constructions can be weakened or eliminated altogether, for example by relying on more structured or restricted knowledge assumptions, or by developing alternative realizations of one-shot decryption without extractability. Even weakening extractability to more structured assumptions would already constitute meaningful progress. For example,[FHAS24] constructs (non-post-quantum-secure) extractable witness encryption from the algebraic group model, demonstrating that extractability-type guarantees can sometimes be obtained from more structured assumptions than general knowledge soundness. More generally, lattice-based knowledge assumptions, such as those in [ACL+22], could serve as potential inspiration.

1.2 Related Work

Mobile adversary and proactive secret sharing. The mobile adversary model [OY91], proposed by Ostrovsky and Yung, considers an adversary $\mathcal{A}$ that can occasionally corrupt parties subject to a corruption budget, and may eventually corrupt all parties by releasing (uncorrupting) old parties and corrupting new ones. Protocols secure against such a mobile adversary are said to satisfy proactive security.

In [OY91], secure MPC protocols are proposed against a mobile adversary. Later, this threat model is re-considered in the context of secret sharing [Sha79, Bla79], yielding proactive secret sharing (PSS) schemes—both in the static committee setting [CH94, HJKY95] and dynamic committee setting [BELO15, MZW+19]. At a high level, PSS protocols are secret sharing protocols with an additional handover sub-protocol that is executed periodically, where parties update their shares and erase old private state.

Recently, Benhamouda *et al.* [BGG+20] proposed the *evolving committee* proactive secret sharing (ECPSS) scheme, where the next secret-holding committee is of size independent of the total number of parties, and is selected as an output of a role-assignment protocol. This approach yields

good scalability as the communication complexity of the handover protocol is independent of the actual number of parties, and has found applications in many high-level protocols, e.g., most notably YOSO MPC [GHK+21].

Remark 1. We highlight that, all existing proactive protocols assume that parties can securely erase their internal state. This holds even if the mobile adversary is *passive* that can only read corrupted parties' internal state but not modify their behavior.

One-shot signatures. One-shot signature is a quantum cryptographic primitive that can be understood as a special type of signature where any secret key allows signing only a single message. One-shot signatures have been introduced in [AGKZ20] and the first construction was given in the work of [SZ25] assuming the existence of post-quantum indistinguishability obfuscation and the hardness of LWE, while recently more efficient constructions have been proposed in [HV25].

Single-decryptor encryption. One scheme that is closely related to our one-shot decryption primitive (in Section 4) is single-decryptor encryption (SDE, or, unclonable decryption) [GZ20, CLLZ21, KY25]. The single decryptor property requires that a (quantum) decryption key cannot be used by two devices. Different notions of such key unclonability have been proposed, and we refer to [KY25] for an overview and comparison.

The SDE constructions employ either a signature-token scheme, or a one-shot signature scheme. To decrypt multiple ciphertexts, the decryption algorithm runs in superposition the signing procedure, and then uses the signature to decrypt the ciphertext. After partial measurement yields the classical plaintext, the computation is rewound so that the quantum secret key is recovered so it can be reused. With foresight, SDE suggests also an attack against a naïve construction of one-shot decryption; for completeness, we also describe the SDE scheme of [GZ20] based on OSS in Appendix A.

2 Preliminaries

Throughout the paper, we will use λ to denote the security parameter. We write $\mathsf{negl}(\cdot)$ to denote a negligible function. For a probabilistic algorithm A, we denote $\mathsf{A}(x;r)$ as running on input x with randomness r. For ensembles of random variable $\mathcal{X} = \{X_i\}_{\lambda\in\mathbb{N},i\in I_\lambda}$ and $\mathcal{Y} = \{Y_i\}_{\lambda\in\mathbb{N},i\in I_\lambda}$ over the same set of indices $I = \bigcup_{\lambda\in\mathbb{N}} I_\lambda$, we write $\mathcal{X} \overset{c}{\approx} \mathcal{Y}$ if they are computationally indistinguishable. I.e., for any non-uniform QPT algorithm $\mathcal{A}$, there exists a negligible function negl such that for all $\lambda \in \mathbb{N}$, it holds that $|\mathbf{Pr}[\mathcal{A}(X_i)] - \mathbf{Pr}[\mathcal{A}(Y_i)]| \leq \mathsf{negl}(\lambda)$.

We assume that a global clock is available to all parties, and the network is synchronous with known upper bound on message delivery delay. We consider an open peer-to-peer network[2], where messages are authenticated using a PKI via a key registration service. I.e., parties can post their public-keys to the service, and anyone can query the set of keys that has so far been posted. This also includes a key-update mechanism which parties can ask to update her current public keys (more details in Section 2.1).

Regarding the computing model, we assume that parties can perform quantum computations locally; yet, all messages transmitted through the network are classical (a.k.a. the Quantum-Computation Classical-Communication [QCCC] model).

[2] As opposed to the point-to-point setup where parties are pairwise connected by secure (physical) channels.

2.1 The Mobile Adversary Model

The adversary $\mathcal{A}$ that we consider is a *mobile* adversary [OY91], who can occasionally corrupt parties up to a "corruption budget," and may eventually corrupt all protocol participants during some (transient) time periods. To model such adversarial capability, we divide the protocol execution into "epochs" (cf. Figure 1), where in each epoch ep the protocol is executed by a committee of parties $\mathcal{P}^{\mathsf{ep}}$, with a subset of parties $\mathcal{I}^{\mathsf{ep}}$ being corrupted. To withstand a mobile adversary, it is necessary to periodically refresh parties' internal states. This is reflected by a "Handover" stage at the onset of each epoch, where old and new committees engage in a handover protocol. E.g., during the handover stage of epoch ep, $\mathcal{P}^{\mathsf{ep}-1}$ can transmit a distributedly maintained secret to $\mathcal{P}^{\mathsf{ep}}$.

The mobile adversary can adaptively corrupt parties at any time during each epoch, taking control over them. The specific types of corruptions and adversarial behaviors are discussed in the sequel. If, after corruption, the adversary decides to "release" a party — i.e., to *uncorrupt* a party — then that party is left alone and resumes executing the original protocol. The "corruption budget" of a mobile adversary is thereby defined as the largest fraction of parties under its control among any committee. For example, consider the stationary case where a committee of n parties run the protocol throughout the execution. Given corruption budget f, the adversary can corrupt up to fn parties in each epoch.

We highlight that, during the handover stage of epoch ep, two committees $\mathcal{P}^{\mathsf{ep}-1}, \mathcal{P}^{\mathsf{ep}}$ engage in this procedure and for each committee, at most f fraction of them can be corrupted. If a party P stays in both committees, then corrupting P consumes budget in both committees. I.e., in the stationary case at most fn parties can be corrupted. With foresight, in the dynamic-committee setting (presented in the sequel), when the two committees do not intersect, at most $2fn$ corrupted parties can exist during the handover stage.

Passive/Active mobile adversary. When a party is corrupted, its entire internal state is exposed to the adversary. Yet, we distinguish two types of corruptions — namely passive and active corruption — where the adversary still follows the protocol but is curious in learning the secret in the former, and completely deviates from the protocol and can send any arbitrary messages in the latter.

Note that, while such mechanism is well defined with classical information, we have to be more careful as parties now possess quantum states. For an active adversary, he can perform arbitrary operations on the corrupted parties' quantum states, or even destroy them completely. In case of a passive adversary, we allow him to perform arbitrary operations on the quantum state that a corrupted party possesses, as long as after these operations the party can still complete the prescribed protocol.

Mobile adversary with stationary/dynamic committee. Regarding the committees that runs the protocol, we distinguish two settings. In the stationary setting, the same committee of n parties run the protocol throughout the entire execution; the set of corrupted parties keeps changing from epoch to epoch, subject to the corruption budget.

In the dynamic-committee setting, committees can change periodically. Further, there is no restrictions on the fraction of parties that can leave/join at once, and two neighboring committees can be two completely different set of parties.

2.2 Proactive Secret Sharing

We divide protocol time into consecutive, non-overlapping epochs of the same duration indexed by ep. Let $\mathcal{P}^{\mathsf{ep}} = (\mathsf{P}_1, \ldots, \mathsf{P}_n)$ denote the set of share-holders at epoch ep (we call this set of parties the "committee"), where a subset of them can be corrupted which we denote as $\mathcal{I}^{\mathsf{ep}}$. We write

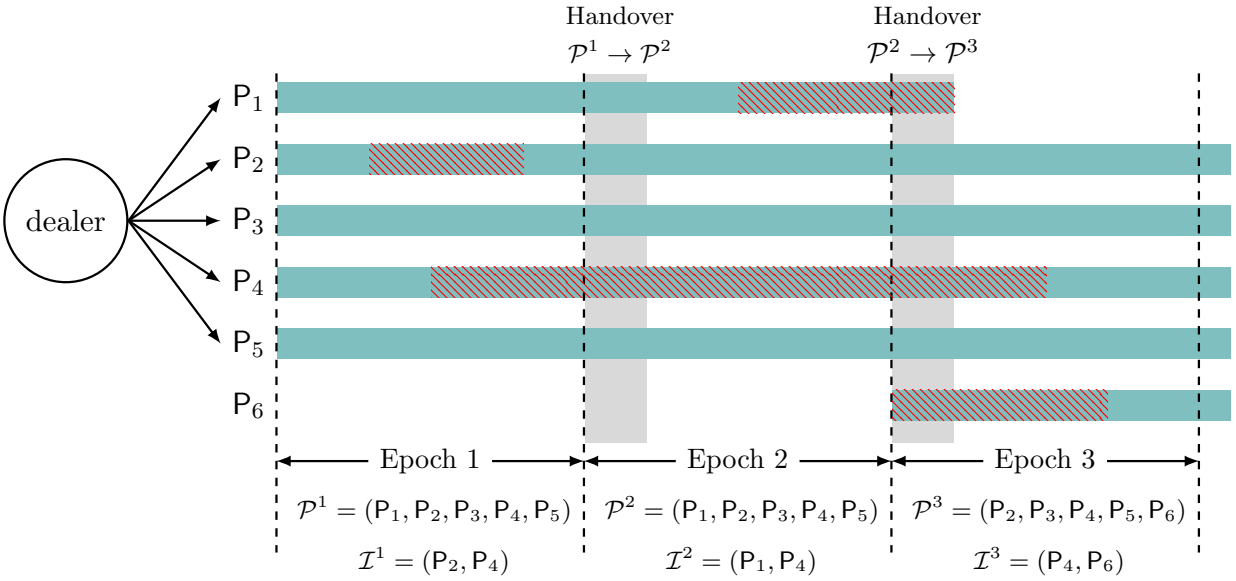

Figure 1: *An illustration of the mobile adversary corruption schedule in a proactive secret sharing protocol. The time interval that a party is a committee member and engages in the handover protocol is marked in teal; the time interval that a party gets corrupted is marked with red slant lines. Each epoch i (except the first one) starts with its handover stage; and then the secret is held by a set of parties $\mathcal{P}^i$, among which a subset of them $\mathcal{I}^i$ are corrupted. Note that, epoch 1 and 2 are run by the same committee, and during the handover stage, P_3 is considered corrupted for both committees. For epoch 2 and 3, P_1 leaves the committee but P_6 joins; they are both corrupted but only accounts for the corruption budget for epoch 2 and 3 respectively.*

$\mathbf{P}^{\mathsf{ep}} = (\mathcal{P}^1, \ldots, \mathcal{P}^{\mathsf{ep}})$ as the array of committees up to epoch ep.[3] We write $\mathbf{I}^{\mathsf{ep}} = (\mathcal{I}^1, \ldots, \mathcal{I}^{\mathsf{ep}})$ as the array of corrupted parties in each epoch up to ep (when it is clear in the context we drop the superscript and simply write $\mathbf{I}$).

A proactive secret sharing protocol (Definition 1) is a secret sharing protocol with an additional "handover" sub-protocol, where parties update their shares in the previous epoch to a fresh new share to be used in the next epoch.

Definition 1 (Proactive Secret Sharing). *A* proactive secret sharing *scheme is a tuple of protocols* $(\mathsf{Setup}, \mathsf{Share}, \mathsf{Handover}, \mathsf{Reconstruct})$[4]*:*

- $\mathsf{Setup}(1^\lambda) \to \mathsf{CRS}$*: A protocol that generates the common-reference string* CRS*.*
- $\mathsf{Share}(\mathsf{CRS}, s) \to (\mathsf{share}_1, \ldots, \mathsf{share}_n)$*: A protocol run by the dealer, which takes as input a common-reference string* CRS *and a secret s, and outputs n shares* $(\mathsf{share}_1, \ldots, \mathsf{share}_n)$ *for the first epoch.*
- $\mathsf{Handover}(\mathcal{P}^{\mathsf{ep}}, \mathcal{P}^{\mathsf{ep}+1}, (\mathsf{share}_i^{\mathsf{ep}})_{i \in [|\mathcal{P}^{\mathsf{ep}}|]}) \to (\mathsf{share}^{\mathsf{ep}+1})_{i \in [|\mathcal{P}^{\mathsf{ep}+1}|]}$*: A protocol run by two committees* $\mathcal{P}^{\mathsf{ep}}$ *and* $\mathcal{P}^{\mathsf{ep}+1}$*, where each party in* $\mathcal{P}^{\mathsf{ep}}$ *takes its own share* $\mathsf{share}_i^{\mathsf{ep}}$ *as input, and each party in* $\mathcal{P}^{\mathsf{ep}+1}$ *outputs a new share* $\mathsf{share}_i^{\mathsf{ep}+1}$ *at the end.*
- $\mathsf{Reconstruct}(\{\mathsf{share}_j\}_{j \in J, |J| > fn}) \to s/\perp$*: An algorithm takes $> fn$ shares (of the same epoch with n parties) as input, and outputs the secret s or $\perp$ (when reconstruction fails).*

[3] In the stationary case $\mathcal{P}^i = \mathcal{P}^j$ for any i, j. In the dynamic setting, two committees can be completely different set of parties.

[4] Note that, shares can be quantum states; and protocols can be interactive rather than algorithms that directly output shares.

An execution of the PSS protocol, denoted $\mathsf{Exec}^{\mathsf{ep}}_{\mathbf{P},\mathbf{I},\mathcal{A}}(\lambda, s, \mathsf{aux}_\lambda)$, refers to the execution where a CRS is generated by running Setup. Then the dealer, with secret s as input, secret shares s among the initial committee using Share. The adversary with auxiliary input aux_λ corrupts parties scheduled due to $\mathbf{I}$. At the onset of each epoch, a handover protocol is invoked where the new committee learns a new set of shares of s. The corruption and handover procedure iterates until the end of epoch ep.

Regarding the view of the adversary, let $\mathsf{View}^{\mathsf{ep}}_{\mathbf{P},\mathbf{I},\mathcal{A}}(\lambda, s_0, \mathsf{aux}_\lambda)$ denote the adversarial view at the end of execution $\mathsf{Exec}^{\mathsf{ep}}_{\mathbf{P},\mathbf{I},\mathcal{A}}(\lambda, s, \mathsf{aux}_\lambda)$, which contains all public information in the network, as well as the internal states of all corrupted parties in $\mathbf{I}$ in each epoch.

A PSS scheme satisfies the following two properties. First, secrecy which requires that, when corrupting less than f fraction of parties in each epoch, the adversary cannot distinguish two executions on two secrets, even if they are adversarially selected.

Definition 2 (Secrecy). *For any QPT adversary $\mathcal{A}$, for any epoch $\mathsf{ep} \in \mathbb{N}$, and for every s_0 and s_1, if $|\mathcal{I}^{\mathsf{ep}'}| \leq f \cdot |\mathcal{P}^{\mathsf{ep}'}|$ for every $\mathsf{ep}' \leq \mathsf{ep}$, then it holds that*

$$\left\{\mathsf{View}^{\mathsf{ep}}_{\mathbf{P},\mathbf{I},\mathcal{A}}(\lambda, s_0, \mathsf{aux}_\lambda)\right\}_{\lambda\in\mathbb{N}} \overset{c}{\approx} \left\{\mathsf{View}^{\mathsf{ep}}_{\mathbf{P},\mathbf{I},\mathcal{A}}(\lambda, s_1, \mathsf{aux}_\lambda)\right\}_{\lambda\in\mathbb{N}}.$$

In addition, when gathering sufficiently many (w.r.t. corruption budget F) valid shares, reconstruction should yield the original secret.

Definition 3 (Robustness). *For any QPT adversary $\mathcal{A}$, for any epoch $\mathsf{ep} \in \mathbb{N}$ and secret s, if $|\mathcal{I}^{\mathsf{ep}'}| \leq f \cdot |\mathcal{P}^{\mathsf{ep}'}|$ for every $\mathsf{ep}' \leq \mathsf{ep}$, then it holds that*

$$\mathsf{Reconstruct}(\{\mathsf{share}_j\}_{j\in J}) = s,$$

where share_j is the share of an honest party P_j at the end of $\mathsf{Exec}^{\mathsf{ep}}_{\mathbf{P},\mathbf{I},\mathcal{A}}(\lambda, s, \mathsf{aux}_\lambda)$ and $|J| > f \cdot |\mathcal{P}^{\mathsf{ep}}|$.

Using the secret without reconstruction (Threshold cryptography). Note that, in many applications, the secret s is used to evaluate a function $\mathsf{F}(s, \mathsf{inp})$ on some input inp, rather than reconstructing its original form (actually, it is often required that s remains hidden throughout the protocol execution). For example, the shared secret is a private key and parties want to jointly decrypt a ciphertext; or, the secret is a signing key and parties want to use it to sign a message.

Though the definition of threshold cryptographic schemes requires adapting to the respective use-case, we can still define an indicative property which we term *secret usability*. This property can be seen, at a high level, as expressing the fact that any information about the secret that does not leak through the function of interest F, should not be disclosed due to the protocol that realizes F in a threshold manner. Towards this goal, let us consider an additional sub-protocol:

- $\mathsf{Threshold}^{\mathsf{F}}(\mathcal{P}, (\mathsf{share}_i)_{i\in[|\mathcal{P}|]}, \mathsf{inp}) \to \mathsf{F}(s, \mathsf{inp})$: A protocol run by $\mathcal{P}$, where each party use its own share share_i and a public input inp as input, and outputs the evaluation result $\mathsf{F}(s, \mathsf{inp})$ on a function F, secret s and input inp.

Let $\mathsf{Exec}^{\mathsf{ep}}_{\mathbf{P},\mathbf{I},\mathcal{A},\mathsf{F},\mathbf{Inp}}(\lambda, s, \mathsf{aux}_\lambda)$ denote the execution the same as $\mathsf{Exec}^{\mathsf{ep}}_{\mathbf{P},\mathbf{I},\mathcal{A}}(\lambda, s, \mathsf{aux}_\lambda)$ except that in each epoch ep', the corresponding committee invokes the sub-protocol $\mathsf{Threshold}^{\mathsf{F}}$ on input $\mathsf{inp}^{\mathsf{ep}'}$ ($\mathbf{Inp} = (\mathsf{inp}^1, \ldots, \mathsf{inp}^{\mathsf{ep}})$). Then, the random variable $\mathsf{View}^{\mathsf{ep}}_{\mathbf{P},\mathbf{I},\mathcal{A},\mathsf{F},\mathbf{Inp}}(\lambda, s, \mathsf{aux}_\lambda)$ is the adversarial view at the end of epoch ep.

Definition 4 (Secret Usability). *(i) (Correctness) Let the execution with input* $(\mathsf{inp}^i)_{i\in[\mathsf{ep}]}$ *be denoted by* $\mathsf{Exec}^{\mathsf{ep}}_{\mathbf{P},\mathbf{I},\mathcal{A},\mathsf{F},\mathbf{Inp}}(\lambda, s, \mathsf{aux}_\lambda)$ *and let* $(\mathsf{output}^i)_{i\in[\mathsf{ep}]}$ *be the evaluation output of honest parties in epoch* i *respectively. It holds that* $\forall i \in [\mathsf{ep}], \mathsf{F}(s, \mathsf{inp}^i) = \mathsf{output}^i$.

(ii) (Indistinguishability) For any QPT adversary $\mathcal{A}$*, for any epoch* $\mathsf{ep} \in \mathbb{N}$ *and for every secret pair* (s_0, s_1)*, if* $|\mathcal{I}^{\mathsf{ep}'}| \leq f \cdot |\mathcal{P}^{\mathsf{ep}'}|$ *for every* $\mathsf{ep}' \leq \mathsf{ep}$*, then for any function* F *and inputs* $\mathbf{Inp} = (\mathsf{inp}^1, \ldots, \mathsf{inp}^{\mathsf{ep}})$ *satisfying that* $\mathsf{F}(s_0, \mathsf{inp}^i) = \mathsf{F}(s_1, \mathsf{inp}^i), \forall i \in [\mathsf{ep}]$*, it must hold that:*

$$\left\{\mathsf{View}^{\mathsf{ep}}_{\mathbf{P},\mathbf{I},\mathcal{A},\mathsf{F},\mathbf{Inp}}(\lambda, s_0, \mathsf{aux}_\lambda)\right\}_{\lambda\in\mathbb{N}} \overset{c}{\approx} \left\{\mathsf{View}^{\mathsf{ep}}_{\mathbf{P},\mathbf{I},\mathcal{A},\mathsf{F},\mathbf{Inp}}(\lambda, s_1, \mathsf{aux}_\lambda)\right\}_{\lambda\in\mathbb{N}}.$$

While secrecy and robustness are two defining properties of proactive secret sharing in the literature, we highlight secret usability as a stepping stone towards more fit for purpose threshold cryptographic operations that are proactively secure without erasures. Importantly, our construction highlights that more sophisticated techniques may be needed to achieve secret usability and thus also threshold cryptography. For details, refer to Section 6.

2.3 One-Shot Signatures

One-shot signatures [AGKZ20, SZ25] are fundamental building blocks of our PSS protocols. At a high level, one-shot-ness guarantees that the quantum secret key self-destructs after signing and no one, including the key holder, can forge a second signature. We present the the formal definition of one-shot signatures in Definition 5.

For our construction, we make black-box use of one-shot signatures. Note that, it has been observed by [AGKZ20] that, one-shot signatures are implied by collision-resistant always-non-collapsing hash functions. In Appendix B, we briefly discuss such type of hash functions by presenting a formal definition in Definition 11 and a construction in Algorithm 3 due to [SZ25] (in the oracle model). Regarding the OSS construction, refer to Algorithm 4.

Definition 5 (One-Shot Signatures). *A one-shot signature (OSS) scheme with message-space* $\mathcal{M}$ *is a tuple of algorithms* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$*:*

- $\mathsf{Setup}(1^\lambda) \to \mathsf{CRS}$*: takes as input a security parameter* λ *in unary, and outputs a common-reference string* CRS*.*
- $\mathsf{KeyGen}(\mathsf{CRS}) \to (\mathsf{pk}, |\mathsf{sk}\rangle)$*: takes as input a common-reference string* CRS*, and outputs a classical public key* pk *and a quantum signing key* $|\mathsf{sk}\rangle$*.*
- $\mathsf{Sign}(\mathsf{CRS}, |\mathsf{sk}\rangle, m) \to \sigma$*: takes as input a common-reference string* CRS*, a quantum signing key* $|\mathsf{sk}\rangle$*, and a message* m*, and outputs a signature* σ*.*
- $\mathsf{Verify}(\mathsf{CRS}, \mathsf{pk}, m, \sigma) \to 0/1$*: takes as input a common-reference string* CRS*, a public key* pk*, a message* m*, and a signature* σ*, and outputs* 1 *if verifies and* 0 *otherwise.*

The scheme satisfies the following properties:

- **Correctness:** *There exists a negligible function* negl *such that, for any* λ *and any* $m \in \mathcal{M}$ *it holds that*

$$\mathbf{Pr}\left[\mathsf{Verify}(\mathsf{CRS}, \mathsf{pk}, m, \sigma_m) = 1 \;\middle|\; \begin{array}{c} \mathsf{CRS} \leftarrow \mathsf{Setup}(1^\lambda), \\ (\mathsf{pk}, |\mathsf{sk}\rangle) \leftarrow \mathsf{KeyGen}(\mathsf{CRS}), \\ \sigma_m \leftarrow \mathsf{Sign}(\mathsf{CRS}, |\mathsf{sk}\rangle, m) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

- **Security:** *For any QPT algorithm* $\mathcal{A}$*, there exists a negligible function* negl *such that for all* λ *the following probability is negligible:*

$$\mathbf{Pr}\left[\begin{array}{l} \mathsf{Verify}(\mathsf{CRS}, \mathsf{pk}, m_0, \sigma_0) = 1 \\ \mathsf{Verify}(\mathsf{CRS}, \mathsf{pk}, m_1, \sigma_1) = 1 \end{array}\;\middle|\; \begin{array}{c} \mathsf{CRS} \leftarrow \mathsf{Setup}(1^\lambda), \\ (\mathsf{pk}, m_0, m_1, \sigma_0, \sigma_1) \leftarrow \mathcal{A}(\mathsf{CRS}) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

2.4 Post-Quantum Cryptographic Tools

Witness encryption. A witness encryption scheme [GGSW13, GKP+13] allows a party to encrypt a message to any NP relation R. I.e., a statement x is chosen at encryption; any witness w such that $(x, w) \in \mathsf{R}$ suffices to successful decryption. Note that, for our construction we need post-quantum witness encryption with extractable security.

Definition 6 (Witness Encryption). *An extractable witness encryption scheme for an* NP *relation* R *and message space* $\mathcal{M}$ *is a pair of algorithms* $(\mathsf{Enc}, \mathsf{Dec})$*:*

- $\mathsf{Enc}(1^\lambda, \mathsf{stmt}, m) \to \mathsf{ct}$*: takes as input a security parameter* λ *in unary, a statement* stmt *and a message* m*, and outputs a ciphertext* ct*.*
- $\mathsf{Dec}(\mathsf{ct}, \mathsf{w}) \to m/\bot$*: takes as input a ciphertext* ct *and a witness* w*, and outputs a message* m *or* $\bot$ *(when decryption fails).*

The scheme satisfies the following properties:

- **Correctness:** *For any* $m \in \mathcal{M}$*, for any* x *and* w *s.t.* $(\mathsf{stmt}, \mathsf{w}) \in \mathsf{R}$*, it holds that*

$$\mathbf{Pr}\big[\mathsf{Dec}(\mathsf{Enc}(1^\lambda, \mathsf{stmt}, m), \mathsf{w}) = m\big] = 1.$$

- **Extractable security:** *For any QPT adversary* $\mathcal{A}$*, polynomial* p *and messages* m_0, m_1*, there is a QPT extractor* $\mathcal{E}$ *and polynomial* q *such that for any mixed state* aux *potentially entangled with an external register, it holds that*

$$\Big|\mathbf{Pr}\big[\mathcal{A}(\mathsf{aux}, \mathsf{Enc}(1^\lambda, \mathsf{stmt}, m_0)) = 1\big] - \mathbf{Pr}\big[\mathcal{A}(\mathsf{aux}, \mathsf{Enc}(1^\lambda, \mathsf{stmt}, m_1)) = 1\big]\Big| \geq \frac{1}{p(\lambda)}$$
$$\implies \mathbf{Pr}\big[\mathcal{E}(1^\lambda, \mathsf{stmt}, \mathsf{aux}) = w \text{ s.t. } (\mathsf{stmt}, \mathsf{w}) \in \mathsf{R}\big] \geq \frac{1}{q(\lambda)}.$$

Extractable witness encryption can be constructed from AGM [FHAS24]; while for post-quantum candidates, there already exist lattice-based knowledge assumptions, see [ACL+22], that could be an inspiration.

Succinct non-interactive arguments of knowledge (SNARKs). For efficiency considerations, we need SNARKs to compress the size of witness used in the witness encryption scheme[5].

Definition 7 (Succinct Non-Interactive Arguments of Knowledge). *A Succinct Non-Interactive Arguments of Knowledge (SNARKs) scheme of a family of relations* $\mathbf{R}$ *is a tuple of algorithms* $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$*:*

- $\mathsf{Setup}(\mathsf{R}) \to \mathsf{CRS}$*: takes as input a relation* $\mathsf{R} \in \mathbf{R}$*, and outputs a common reference string* CRS*.*
- $\mathsf{Prove}(\mathsf{CRS}, \mathsf{stmt}, \mathsf{w}) \to \pi$*: takes as input a common reference string* CRS*, a statement* stmt *and the corresponding witness* w*, and outputs a proof* π*.*
- $\mathsf{Verify}(\mathsf{CRS}, \mathsf{stmt}, \pi) \to 0/1$*: takes as input a common reference string* CRS*, a statement* stmt *and a proof* π*, and outputs* 1 *if verifies and* 0 *otherwise.*

The scheme satisfies the following properties:

[5] For our application, we also require the SNARK scheme to be recursively composable. Note that it is common to conjecture that SNARKs can be recursively composed for a polynomial number of times. See, e.g., [BCTV14] and discussions in [AGKZ20].

– **Correctness:** *For any relation* $\mathsf{R} \in \mathbf{R}$*, and any statement-witness pair* $(\mathsf{stmt}, \mathsf{w}) \in \mathsf{R}$*, it holds that*

$$\mathbf{Pr}\big[\mathsf{Verify}(\mathsf{CRS}, \mathsf{stmt}, \pi) \;\big|\; \mathsf{CRS} \leftarrow \mathsf{Setup}(\mathsf{R}), \pi \leftarrow \mathsf{Prove}(\mathsf{CRS}, \mathsf{stmt}, \mathsf{w})\big] = 1.$$

– **(Knowledge-)Soundness:** *For any QPT adversarial prover* $\mathcal{A}$*, there exists a QPT extractor* $\mathcal{E}$ *such that for any mixed state* aux *potentially entangled with an external register, and for any relation* $\mathsf{R} \in \mathbf{R}$*, it holds that*

$$\mathbf{Pr}\left[\begin{array}{c}\mathsf{Verify}(\mathsf{CRS}, \mathsf{stmt}, \pi) = 1 \\ \wedge (\mathsf{stmt}, \mathsf{w}) \in \mathsf{R}\end{array}\middle|\begin{array}{c}\mathsf{CRS} \leftarrow \mathsf{Setup}(\mathsf{R}), \\ (\mathsf{stmt}, \pi) \leftarrow \mathcal{A}(\mathsf{CRS}, \mathsf{aux}), \\ \mathsf{w} \leftarrow \mathcal{E}(\mathsf{CRS}, \mathsf{aux})\end{array}\right] \leq \mathsf{negl}(\lambda)$$

– **Succinctness:** *There exists a polynomial function* poly *such that for all* λ *and any relation* $\mathsf{R} \in \mathbf{R}$ *and every statement-witness pair* $(\mathsf{stmt}, \mathsf{w}) \in \mathsf{R}$*, it holds that:*
  – *Every honestly generated proof* π *is of size* $\mathsf{poly}(\lambda + \log|\mathsf{w}|)$*.*
  – *The verifier runs in time* $\mathsf{poly}(\lambda + |\mathsf{stmt}| + \log|\mathsf{w}|)$*.*

For our constructions we need post-quantum SNARKs [COS20].

Functional witness encryption. Some of our constructions further require functional witness encryption [BCP14].

Definition 8 (Functional Witness Encryption). *A* functional witness encryption *scheme for an* NP *relation* R*, message space* $\mathcal{M}$*, and class of Turing machines* $\{\mathsf{M}_\lambda\}$ *is a pair of algorithms* $(\mathsf{Enc}, \mathsf{Dec})$*:*

– $\mathsf{Enc}(1^\lambda, \mathsf{stmt}, m, M) \to \mathsf{ct}$*: takes as input a security parameter* λ *in unary, a statement* stmt*, a message* m*, and a Turing machine description* $M \in \{\mathsf{M}_\lambda\}$*, and outputs a ciphertext* ct*.*
– $\mathsf{Dec}(\mathsf{ct}, \mathsf{w}) \to m/\perp$*: takes as input a ciphertext* ct *and a witness* w*, and outputs an evaluation* m *or* $\perp$ *(when decryption fails).*

The scheme satisfies the following properties:

– **Correctness:** *There exists a negligible function* negl *such that, for any* λ*, for any* $m \in \mathcal{M}$*, for any* x *and* w *s.t.* $(\mathsf{stmt}, \mathsf{w}) \in \mathsf{R}$*, and for any Turing machine* $M \in \{\mathsf{M}_\lambda\}$*, it holds that*

$$\mathbf{Pr}\big[\mathsf{Dec}(\mathsf{Enc}(1^\lambda, x, m, M), w) = M(m, w)\big] \geq 1 - \mathsf{negl}(\lambda).$$

– **Extractable security:** *For any QPT adversary* $\mathcal{A}$*, polynomial* p*, messages* m_0, m_1*, and Turing machine* $M \in \{\mathsf{M}_\lambda\}$*, there is a QPT extractor* $\mathcal{E}$ *and polynomial* q *such that for any mixed state* aux *potentially entangled with an external register, it holds that*

$$\Big|\mathbf{Pr}\big[\mathcal{A}(\mathsf{aux}, \mathsf{Enc}(1^\lambda, \mathsf{stmt}, m_0, M)) = 1\big] - \mathbf{Pr}\big[\mathcal{A}(\mathsf{aux}, \mathsf{Enc}(1^\lambda, \mathsf{stmt}, m_1, M)) = 1\big]\Big| \geq \frac{1}{p(\lambda)}$$

$$\implies \mathbf{Pr}\big[\mathcal{E}(1^\lambda, \mathsf{stmt}, m_0, m_1, M, \mathsf{aux}) = \mathsf{w} \text{ s.t. } (\mathsf{stmt}, \mathsf{w}) \in \mathsf{R} \wedge M(m_0, \mathsf{w}) \neq M(m_1, \mathsf{w})\big] \geq \frac{1}{q(\lambda)}.$$

Functional witness encryption is known to be equivalent to extractability obfuscation [BCP14]. Therefore, existence of post-quantum functional witness encryption follows from the existence of post-quantum extractability obfuscation.

3 On Necessity of Secure Erasures in Classical Setting

While secure erasures appear as an implicit assumption in all existing proactive secret sharing schemes, to better motivate our work, we here formally prove that, in the classical setting it is impossible to implement proactive security.

Theorem 1. *Consider a stationary set of n parties, and assume that none of them can possess quantum internal states nor perform secure erasures. Then, there is no protocol that achieves proactive secret sharing as defined in Definitions 1 to 3 in Section 2.2 for any corruption threshold $f > 0$.*

Proof. Suppose—towards a contradiction—that there is a protocol Π that realizes proactive security for $f = 1/n$, i.e., the adversary is allowed to corrupt only one party per epoch. We consider a passive adversary that can only observe the internal state of corrupted parties but has to otherwise follow the protocol; an active adversary can mimic a passive one so the proof applies as well.

Due to the fact that parties cannot perform secure erasures, whenever a party is corrupted in epoch ep, the adversary learns all its internal states in any epoch $\mathsf{ep}' \leq \mathsf{ep}$.

Consider two protocol executions with dealer invoked to share secret s and s' respectively ($s \neq s'$) running for n epochs. The adversary schedules corruption as $\mathbf{I} = ((\mathsf{P}_1), (\mathsf{P}_2), \ldots, (\mathsf{P}_n))$ — i.e., it corrupts the i-th party in the i-th epoch. Due to secrecy, we learn that

$$\left\{\mathsf{View}^n_{\mathcal{P},\mathbf{I},\mathcal{A}}(\lambda, s, \mathsf{aux}_\lambda)\right\}_{\lambda\in\mathbb{N}} \overset{c}{\approx} \left\{\mathsf{View}^n_{\mathcal{P},\mathbf{I},\mathcal{A}}(\lambda, s', \mathsf{aux}_\lambda)\right\}_{\lambda\in\mathbb{N}}.$$

Since no secure erasures are allowed, this implies that, for executions with $\mathbf{I}' = ((\mathsf{P}_1, \ldots, \mathsf{P}_n))$,

$$\left\{\mathsf{View}^1_{\mathcal{P},\mathbf{I}',\mathcal{A}}(\lambda, s, \mathsf{aux}_\lambda)\right\}_{\lambda\in\mathbb{N}} \overset{c}{\approx} \left\{\mathsf{View}^1_{\mathcal{P},\mathbf{I}',\mathcal{A}}(\lambda, s', \mathsf{aux}_\lambda)\right\}_{\lambda\in\mathbb{N}}.$$

In other words, $\mathcal{A}$ cannot distinguish the two executions with s, s' even if he manages to corrupt all parties during the first epoch.

Let $\{\mathsf{share}_i\}_{i\in[n]}$ ($\{\mathsf{share}'_i\}_{i\in[n]}$ respectively) denote the set of shares that dealer creates for secret s (s' respectively). The above indistinguishability implies that:

$$\left|\mathbf{Pr}\left[\mathcal{A}(\{\mathsf{share}_i\}_{i\in[n]})\right] - \mathbf{Pr}\left[\mathcal{A}(\{\mathsf{share}'_i\}_{i\in[n]})\right]\right| \leq \mathsf{negl}(\lambda).$$

This implies that $\mathsf{Reconstruct}(\{\mathsf{share}_i\}_{i\in[n]}) = \mathsf{Reconstruct}(\{\mathsf{share}'_i\}_{i\in[n]})$. However, this contradicts robustness which requires that $\mathsf{Reconstruct}(\{\mathsf{share}_i\}_{i\in[n]}) = s \neq s' = \mathsf{Reconstruct}(\{\mathsf{share}'_i\}_{i\in[n]})$. □

4 One-Shot Decryption

We present in this section the main building block of our PSS protocol— a public-key encryption scheme featuring *one-shot decryption* (a property which is obviously impossible in the classical setting). We start with the two-party case (Section 4.1), showing challenges from concrete attacks. Then, we move to the threshold setting and elaborate on our new definitions (Section 4.2) and construction from one-shot signatures (Section 4.3).

4.1 Two-Party One-shot Decryption

Our first attempt is to define one-shot decryption in the standard two-party setting with a single sender and receiver. Note that, we explicitly define a Setup procedure which generates the CRS; and to facilitate one-shot decryption, our secret key must be quantum and will be denoted by $|\mathsf{sk}\rangle$.

We present the formal definition in Definition 9. Correctness and IND-CPA security are defined in the standard way. Then, the third property — *one-shot decryption* — is defined using the experiment where the challenger first generates the CRS, and then the adversary hands out a key pair $(\mathsf{pk}, |\mathsf{sk}\rangle)$ with additionally four messages $(m_{0,0}, m_{0,1}, m_{1,0}, m_{1,1})$. The challenger then randomly samples two bits (b_0, b_1) and provides the adversary two ciphertexts under pk: c_0 as encryption of m_{0,b_0} and c_1 as encryption of m_{0,b_1}. The adversary wins the game if he comes up with (b'_0, b'_1) such that $b_0 = b'_0$ and $b_1 = b'_1$. A PKE scheme is said to have one-shot decryption if the adversary wins with only negligible advantage.

Definition 9 (PKE with One-shot Decryption). *A public-key encryption with* one-shot-decryption *scheme is a tuple of algorithms* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$*:*

- $\mathsf{Setup}(1^\lambda) \to \mathsf{CRS}$*: takes as input the security parameter* λ *in unary, and outputs the common reference string* CRS*.*
- $\mathsf{KeyGen}(\mathsf{CRS}; r) \to (\mathsf{pk}, |\mathsf{sk}\rangle)$*: takes as input the common reference string* CRS *and random coins* r*, and outputs a public key* pk *and a secret key* $|\mathsf{sk}\rangle$*.*
- $\mathsf{Enc}(\mathsf{CRS}, \mathsf{pk}, m; u) \to \mathsf{ct}$*: takes as input the common reference string* CRS*, a public keys* pk*, a message* m*, and random coins* u*, and outputs a ciphertext* ct*.*
- $\mathsf{Dec}(|\mathsf{sk}\rangle, \mathsf{ct}) \to m/\perp$*: takes as input a secret key* $|\mathsf{sk}\rangle$ *and ciphertext* ct*, and outputs a message* m *or symbol* $\perp$ *(when decryption fails).*

The scheme satisfies the following properties:

- **Correctness:** *There exists a negligible function* negl *such that, for any* λ*, any* $m \in \mathcal{M}$*, it holds that*

$$\mathbf{Pr}\left[\mathsf{Dec}(|\mathsf{sk}\rangle, \mathsf{ct}) \neq m \;\middle|\; \begin{array}{c}\mathsf{CRS} \leftarrow \mathsf{Setup}(1^\lambda), (\mathsf{pk}, |\mathsf{sk}\rangle) \leftarrow \mathsf{KeyGen}(\mathsf{CRS}),\\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{CRS}, \mathsf{pk}, m)\end{array}\right] \leq \mathsf{negl}(\lambda).$$

- **Security:** *For any QPT algorithm* $\mathcal{A}$*, there exists a negligible function* negl *such that for any* λ *it holds that*

$$\mathbf{Pr}\left[\mathcal{A}(\mathsf{ct}) = b \;\middle|\; \begin{array}{c}\mathsf{CRS} \leftarrow \mathsf{Setup}(1^\lambda), (\mathsf{pk}, |\mathsf{sk}\rangle) \leftarrow \mathsf{KeyGen}(\mathsf{CRS}),\\ (m_0, m_1) \leftarrow \mathcal{A}(\mathsf{pk}), b \xleftarrow{\$} \{0,1\}, \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{CRS}, \mathsf{pk}, m_b)\end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

- **One-Shot Decryption:** *For any QPT algorithm* $\mathcal{A}$*, there exists a negligible function* negl *such that for any* λ *it holds that*[6]

$$\mathbf{Pr}\left[(b_0, b_1) = (b'_0, b'_1) \;\middle|\; \begin{array}{c}\mathsf{CRS} \leftarrow \mathsf{Setup}(1^\lambda), (\mathsf{pk}, |\mathsf{sk}\rangle, m_{0,0}, m_{0,1}, m_{1,0}, m_{1,1}) \leftarrow \mathcal{A}(\mathsf{CRS}),\\ (b_0, b_1) \xleftarrow{\$} \{0,1\}, \mathsf{ct}_0 \leftarrow \mathsf{Enc}(\mathsf{CRS}, \mathsf{pk}, m_{0,b_0}),\\ \mathsf{ct}_1 \leftarrow \mathsf{Enc}(\mathsf{CRS}, \mathsf{pk}, m_{1,b_1}), (b'_0, b'_1) \leftarrow \mathcal{A}(\mathsf{CRS}, \mathsf{ct}_0, \mathsf{ct}_1)\end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

We highlight that, so far there is no construction implementing Definition 9. The reason is that, in order to facilitate one-shot decryption, the quantum secret key must be destroyed during decryption. However, since decryption is always deterministic, by exploiting the Gentle Measurement Lemma [Win99, Aar09], the adversary can perform decrypting operations in superposition, measure the output register and then rewind the computations. This leaves the secret key (the OSS signing key) undisturbed; therefore the receiver may re-use it in the future. Interestingly, such attack is indeed the construction for the single-decryptor encryption scheme [GZ20, CLLZ21, KY25]. See Algorithm 2 in Appendix A for more details.

[6] In the game, it is required that $|m_{0,0}| = |m_{0,1}| \wedge |m_{1,0}| = |m_{1,1}|$; otherwise it aborts.

4.2 Multi-Party One-shot Decryption

Given challenges raised from the fact that a deterministic decryption procedure is always "rewindable" with Gentle Measurement Lemma, in this section we explore new directions that can circumvent such "undesired"[7] scenario. Our new technique, at a high level, employs threshold decryption and relies on honest parties to indeed consume sufficiently many quantum private keys to withstand the aforementioned attack.

We term this new primitive *threshold PKE with one-shot decryption* (Definition 10). In this scheme, there is a committee of (quantum) private-key holders, to which any sender can encrypt their message. An additional algorithm $\mathsf{PartialDec}$ enables key holders to locally generate a classical decryption share of a message; and Dec decrypts a message when gathering sufficiently many shares of its ciphertext. Regarding the threshold, a set of at least $(1-f)n$ shares is necessary when considering a n-party committee where the adversary can corrupt up to f fraction of keys. Looking ahead, we set $f < 1/2$ — i.e., a minority of the parties can be corrupted and we need a majority of the shares to decrypt.

Correctness requires that any qualified subset with size no less than $(1-f)n$ of the committee should be able to jointly decrypt a message. For security and one-shot decryption, they are formalized through extending the corresponding properties in Definition 9 to the threshold setting against an *adaptive* adversary $\mathcal{A}$. I.e., $\mathcal{A}$ can corrupt parties at any time during the experiment as long as it controls at most fn private keys.

More specifically, in the security experiment, the adversary, when controlling no more than a f fraction of the quantum private keys (either maliciously generated at the onset or acquired adaptively from formerly honest parties), has to guess a bit b with the ciphertext ct encrypting a message m_b. Refer to Experiment 1 for more details. Security holds if $\mathcal{A}$ cannot win the game with non-negligible advantage.

Regarding one-shot decryption, it is required to guess b_0 and b_1 given two ciphertexts ct_0 and ct_1 encrypting m_{0,b_0} and m_{0,b_1} respectively. The adversary can adaptively corrupt parties in the same way as that in the game of IND-CPA security. Additionally, $\mathcal{A}$ can instruct honest parties to partially decrypt one ciphertext through a partial decryption oracle $\mathcal{O}^{\mathsf{Dec}}$, subject to the restriction that honest keys should unanimously decrypt either ct_0 or ct_1. Furthermore, we highlight that if $\mathcal{A}$ adaptively corrupts an honest party which has consumed her private key to perform partial decryption, then such posterior corruption is not counted in the fn corruption budget (see Line 13-18 in Experiment 2). In other words, the adversary can corrupt all parties after decryption. Looking ahead, in Theorem 2 we can show that by exploiting the security of one-shot signatures, one can have a construction tolerating such posterior corruption. One-shot decryption holds if the adversary can win the game (Experiment 2) with only negligible advantage.

Remark 2. If honest parties do not unanimously decrypt the same ciphertext, for instance half of them decrypt ct_0 and the rest decrypt ct_1, then the adversary can run the Dec algorithm on superposition and decrypt both ciphertexts, thus breaking one-shot decryption. This is a multi-party variant of the attack discussed in Section 4.1. To avoid such scenario, we need honest parties coordinating before partial decryption.

Definition 10 (Threshold PKE with One-shot Decryption). *A threshold public-key encryption with* one-shot-decryption *scheme, parameterized with corruption threshold f, is a tuple of algorithms* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{PartialDec}, \mathsf{Dec})$*:*

- $\mathsf{Setup}(1^\lambda) \to \mathsf{CRS}$*: takes as input the security parameter λ in unary, and outputs the common reference string* CRS*.*

[7] Undesired in terms of our applications.

- $\mathsf{KeyGen}(\mathsf{CRS}; r) \to (\mathsf{pk}, |\mathsf{sk}\rangle)$*: takes as input the common reference string* CRS *and random coins* r*, and outputs a public key* pk *and a secret key* $|\mathsf{sk}\rangle$*.*
- $\mathsf{Enc}(\mathsf{CRS}, \{\mathsf{pk}_i\}_{i\in[n]}, m; u) \to \mathsf{ct}$*: takes as input the common reference string* CRS*, a set of* n *public keys* $\{\mathsf{pk}_i\}_{i\in[n]}$*, a message* m*, and random coins* u*, and outputs a ciphertext* ct*.*
- $\mathsf{PartialDec}(\mathsf{CRS}, |\mathsf{sk}\rangle, \mathsf{ct}) \to \sigma/\perp$*: takes as input the common reference string* CRS*, the secret key* $|\mathsf{sk}\rangle$ *and ciphertext* ct*, and outputs a partial decryption* σ *or symbol* $\perp$ *(when partial decryption fails).*
- $\mathsf{Dec}(\{\sigma_j\}_{j\in J}, \mathsf{ct}) \to m/\perp$*: takes as input a set of partial decryptions with* $|J| > fn$ *and ciphertext* ct*, and outputs a message* m *or symbol* $\perp$ *(when decryption fails).*

The scheme satisfies the following properties:

- **Correctness:** *There exists a negligible function* negl *such that, for any* λ*, any* $m \in \mathcal{M}$ *and any set* $J \in [n]$ *with* $|J| \geq (1-f)n$*, it holds that*

$$\mathbf{Pr}\left[\mathsf{Dec}(\{\sigma_i\}_{i\in J}, \mathsf{ct}) \neq m \;\middle|\; \begin{array}{c} \mathsf{CRS} \leftarrow \mathsf{Setup}(1^\lambda), \{(\mathsf{pk}_i, |\mathsf{sk}_i\rangle) \leftarrow \mathsf{KeyGen}(\mathsf{CRS})\}_{i\in[n]} \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{CRS}, \{\mathsf{pk}_i\}_{i\in[n]}, m), \\ \{\sigma_j \leftarrow \mathsf{PartialDec}(\mathsf{CRS}, |\mathsf{sk}_j\rangle, \mathsf{ct})\}_{j\in[J]} \end{array}\right] \leq \mathsf{negl}(\lambda).$$

- **Security:** *For any QPT algorithm* $\mathcal{A}$*, there exists a negligible function* negl *such that for any* λ*, it holds that*

$$\mathbf{Pr}[\mathbf{Exp}^{f,n}_{\mathsf{PKE\text{-}OSTD\text{-}SEC},\mathcal{A}}(\lambda) = 1] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

  where $\mathbf{Exp}^{f,n}_{\mathsf{PKE\text{-}OSTD\text{-}SEC},\mathcal{A}}$ *is defined in Experiment 1.*
- **One-Shot Decryption:** *For any QPT algorithm* $\mathcal{A}$*, there exists a negligible function* negl *such that for any* λ*, it holds that*

$$\mathbf{Pr}[\mathbf{Exp}^{f,n}_{\mathsf{PKE\text{-}OSTD\text{-}OSD},\mathcal{A}}(\lambda) = 1] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

  where $\mathbf{Exp}^{f,n}_{\mathsf{PKE\text{-}OSTD\text{-}OSD},\mathcal{A}}$ *is defined in Experiment 2.*

Experiment PKE-OSTD-SEC

$\underline{\mathbf{Exp}^{f,n}_{\mathsf{PKE\text{-}OSTD\text{-}SEC},\mathcal{A}}(\lambda)}$

$\mathsf{CRS} \leftarrow \mathsf{Setup}(1^\lambda)$
$\{\mathsf{pk}_j\}_{j\in J} \leftarrow \mathcal{A}(\mathsf{CRS})$ ◁ Abort if $|J| > fn$
$\{(\mathsf{pk}_i, |\mathsf{sk}_i\rangle) \leftarrow \mathsf{KeyGen}(\mathsf{CRS})\}_{i\in[n]\setminus J}$
Invoke sub-procedure AdaptiveCorrupt
$(m_0, m_1) \leftarrow \mathcal{A}(\mathsf{CRS}, (\mathsf{pk}_i)_{i\in[n]})$ ◁ Abort if $|m_{0,0}| \neq |m_{0,1}|$ or $|m_{1,0}| \neq |m_{1,1}|$
$b \xleftarrow{\$} \{0,1\}$, $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{CRS}, \mathsf{pk}, m_b)$
Invoke sub-procedure AdaptiveCorrupt
$b' \leftarrow \mathcal{A}(\mathsf{CRS}, (\mathsf{pk}_i)_{i\in[n]})$
if $b = b'$ **then return** 1 **else return** 0

$\underline{\mathsf{AdaptiveCorrupt}}$

$J' \leftarrow \mathcal{A}(\mathsf{CRS}, \{\mathsf{pk}_i\}_{i\in[n]\setminus J})$
if $|J'| + |J| > fn$ **then abort**
for each $j \in J'$ **do**
    send $(\mathsf{pk}_j, |sk_j\rangle)$ to $\mathcal{A}$
end for
Set $J \leftarrow J \cup J'$

Experiment 1: *The security experiment for Definition 10.*

Experiment PKE-OSTD-OSD

$\underline{\mathbf{Exp}^{f,n}_{\mathsf{PKE\text{-}OSTD\text{-}OSD},\mathcal{A}}(\lambda)}$

$\mathsf{CRS} \leftarrow \mathsf{Setup}(1^\lambda)$
$\{\mathsf{pk}_j\}_{j\in J} \leftarrow \mathcal{A}(\mathsf{CRS})$ ◁ Abort if $|J| > fn$
$\{(\mathsf{pk}_i, |\mathsf{sk}_i\rangle) \leftarrow \mathsf{KeyGen}(\mathsf{CRS})\}_{i\in[n]\setminus J}$
Initialize $\mathcal{O}_{\mathsf{Dec}}$ with $\mathsf{keys} = \{(\mathsf{pk}_i, |\mathsf{sk}_i\rangle)\}_{i\in[n]\setminus J}$ and $\mathsf{ct} = \varepsilon$
Invoke sub-procedure AdaptiveCorrupt
$(m_{0,0}, m_{0,1}, m_{1,0}, m_{1,1}) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Dec}}}(\mathsf{CRS}, (\mathsf{pk}_i)_{i\in[n]})$ ◁ Abort if $|m_{0,0}| \neq |m_{0,1}|$ or $|m_{1,0}| \neq |m_{1,1}|$
$(b_0, b_1) \xleftarrow{\$} \{0,1\}$, $\mathsf{ct}_0 \leftarrow \mathsf{Enc}(\mathsf{CRS}, \mathsf{pk}, m_{0,b_0})$, $\mathsf{ct}_1 \leftarrow \mathsf{Enc}(\mathsf{CRS}, \mathsf{pk}, m_{1,b_1})$
Invoke sub-procedure AdaptiveCorrupt
$(b'_0, b'_1) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Dec}}}(\mathsf{CRS}, (\mathsf{pk}_i)_{i\in[n]})$
if $(b_0, b_1) = (b'_0, b'_1)$ **then return** 1 **else return** 0

$\underline{\mathsf{AdaptiveCorrupt}}$

$J' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Dec}}}(\mathsf{CRS}, \{\mathsf{pk}_i\}_{i\in[n]\setminus J})$
for each $j \in J'$ **do**
  if pk_j has not been queried in $\mathcal{O}_{\mathsf{Dec}}$ **then**
    if $|J'| + 1 > fn$ **then abort**
    Set $J \leftarrow J \cup \{j\}$
    Remove $(\mathsf{pk}_j, |\mathsf{sk}_j\rangle)$ from $\mathcal{O}_{\mathsf{Dec}}.\mathsf{keys}$
    Send $(\mathsf{pk}_j, |sk_j\rangle)$ to $\mathcal{A}$
  end if
end for

$\underline{\mathcal{O}_{\mathsf{Dec}}(\mathsf{pk}', \mathsf{ct}')}$

if pk' is queried before **then abort**
if $\mathsf{ct} = \varepsilon$ **then** $\mathsf{ct} = \mathsf{ct}'$
if $\mathsf{ct} \neq \mathsf{ct}'$ **then abort**
if $\mathsf{pk}' = \mathsf{pk}_i$ in keys **then**
  return $\mathsf{PartialDec}(\mathsf{CRS}, |\mathsf{sk}_i\rangle, \mathsf{ct})$
end if

Experiment 2: *The one-shot decryption experiment for Definition 10.*

4.3 Construction from One-shot Signatures

We present how to construct a threshold PKE scheme with one-shot decryption, using one-shot signatures and post-quantum witness encryption. Specifically, during setup stage a CRS of the underlying OSS scheme is generated; then, each party in the decryption committee locally generates an OSS key pair $(\mathsf{pk}, |\mathsf{sk}\rangle)$ and publishes pk. To encrypt a message m, the sender first picks a random string tag and then generates a ciphertext of m using post-quantum witness encryption under the statement that, informally speaking, requires more than $(1-f)n$ valid signatures on tag (see Equation (1)) as witness. Note that in witness encryption, by looking at the ciphertext, parties can learn the statement—i.e., the set of public keys this ciphertext encrypts to, as well as the string tag to be signed.

$$\mathsf{R}_{\mathsf{OSTD}} \triangleq \left\{ \begin{matrix} \mathsf{stmt} = (\mathsf{CRS}, (\mathsf{pk}_i)_{i\in[n]}, \mathsf{tag}), \\ \mathsf{w} = (\{\sigma_j\}_{j\in J}) \end{matrix} \;\middle|\; \begin{matrix} \forall j \in J, \mathsf{OSS.Verify}(\mathsf{CRS}, \mathsf{pk}_j, \mathsf{tag}, \sigma_j) = 1 \\ \wedge |J| \geq (1-f)n \end{matrix} \right\}. \tag{1}$$

To jointly decrypt a ciphertext ct, parties first locally generate partial decryptions which are exactly their signatures on string x in the statement. Then, after collecting $\geq (1-f)n$ partial decryptions anyone can decrypt ct and learn the message. To facilitate one-shot decryption, we set $f < 1/2$ – as decrypting two ciphertexts would thus need $2(1-f)n > n$ different one-shot signatures. Also, as a side remark, when given multiple ciphertexts, it can be the case that different parties sign on completely different tags hence none of the messages can be decrypted. In other words,

parties have to agree on the ciphertext before signing, as they cannot rewind the computation after acquiring the signature via measurement.[8]

A detailed code specification of our construction is presented in Algorithm 1; and we show that this implements Definition 10 in Theorem 2.

Algorithm PKE-OSTD

Let $\mathsf{OSS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ be a one-shot signature scheme as Definition 5, and $\mathsf{WE} = (\mathsf{Enc}, \mathsf{Dec})$ be a witness-encryption scheme as Definition 6 for relation $\mathsf{R}_{\mathsf{OSTD}}$ (cf. Equation (1)).

$\underline{\mathsf{Setup}(1^\lambda)}$
1: Set $\mathsf{CRS} \leftarrow \mathsf{OSS.Setup}(1^\lambda)$
2: Return CRS

$\underline{\mathsf{KeyGen}(\mathsf{CRS})}$
3: Set $(\mathsf{pk}, |\mathsf{sk}\rangle) \leftarrow \mathsf{OSS.KeyGen}(\mathsf{CRS})$
4: Return $(\mathsf{pk}, |\mathsf{sk}\rangle)$

$\underline{\mathsf{Enc}(\mathsf{CRS}, (\mathsf{pk}_i)_{i\in[n]}, m)}$
5: Pick $\mathsf{tag} \xleftarrow{\$} \{0,1\}^\lambda$
6: Compute $\mathsf{ct} \leftarrow \mathsf{WE.Enc}((\mathsf{CRS}, (\mathsf{pk}_i)_{i\in[n]}, \mathsf{tag}), m)$
7: Return ct

$\underline{\mathsf{PartialDec}(\mathsf{CRS}, |\mathsf{sk}_i\rangle, \mathsf{ct})}$
8: Parse ct statement as $(\mathsf{CRS}, (\mathsf{pk}_j)_{j\in[n]}, \mathsf{tag})$
9: $\sigma_i \leftarrow \mathsf{OSS.Sign}(\mathsf{CRS}, |\mathsf{sk}_i\rangle, \mathsf{tag})$
10: **return** σ_i

$\underline{\mathsf{Dec}(\mathsf{ct}, \{\sigma_j\}_{j\in J})}$
11: **if** $|J| < (1-f)n$ **then** abort
12: Compute $m \leftarrow \mathsf{WE.Dec}(\mathsf{ct}, (\{\sigma_j\}_{j\in J}))$
13: **return** m

Algorithm 1: *A threshold public-key encryption scheme with one-shot decryption.*

Theorem 2. *Assuming one-shot signatures as defined in Definition 5 and post-quantum extractable witness encryption as defined in Definition 6, there is a threshold public-key encryption with one-shot decryption algorithm satisfying correctness, security, one-shot decryption as defined in Definition 10 for any $f < 1/2$.*

Proof. Correctness. This follows the correctness properties of the underlying witness encryption and one-shot signature schemes.

Security. Note that the extractable security of witness encryption implies that, if an adversary $\mathcal{A}$ has non-negligible distinguishing advantage, then given the statement, there is an extractor $\mathcal{E}$ that is able to extract the witness—i.e., at least $(1-f)n$ one-shot signatures—with non-negligible probability. Thus, consider the following reduction. Let $\mathcal{A}_{\mathsf{OSS}}$ be the adversary in the security game of the OSS scheme (cf. Definition 5). $\mathcal{A}_{\mathsf{OSS}}$ runs Experiment 1 as the challenger with $(\mathcal{A}, \mathcal{E})$, where $\mathcal{A}$ is the adversary in the security game of the PKE with one-shot decryption scheme, and $\mathcal{E}$ is the extractor of the underlying witness encryption scheme. Let $\{\mathsf{pk}_j\}_{j\in J}$ be the set of public keys generated/corrupted by $\mathcal{A}$ at the end of the experiment, and let $\{\mathsf{pk}_i, |\mathsf{sk}_i\rangle\}_{i\in[n]\setminus J}$ be the set of key pairs controlled by $\mathcal{A}_{\mathsf{OSS}}$ at the end ($|J| \leq fn$). Since $\mathcal{A}$ wins with non-negligible probability, by a simple reduction to the extractable security of witness encryption we learn that $\mathcal{E}$ extracts no less than $(1-f)n$ one-shot signatures given the statement as defined in Equation (1). I.e., $\mathcal{E}$ produces at least one one-shot signature σ that verifies tag under $\mathsf{pk}^* \in \{\mathsf{pk}_i\}_{i\in[n]\setminus J}$. By signing on a different $\mathsf{tag}' \neq \mathsf{tag}$ using $|\mathsf{sk}\rangle^*$ which yields σ', $\mathcal{A}_{\mathsf{OSS}}$ wins the OSS security game with the tuple $(\mathsf{pk}^*, \mathsf{tag}, \mathsf{tag}', \sigma, \sigma')$.

[8] We assume out-of-the-scope the procedure that honest parties agree on a selected ciphertext to decrypt.

One-shot decryption. Again we prove this via a reduction to the security game of the underlying OSS scheme against $\mathcal{A}_{\mathsf{OSS}}$. Consider that $\mathcal{A}_{\mathsf{OSS}}$ runs Experiment 2 as the challenger with $(\mathcal{A}, \mathcal{E})$, where $\mathcal{A}$ is the adversary in the one-shot decryption game of the PKE with one-shot decryption scheme, and $\mathcal{E}$ is the extractor of the underlying witness encryption scheme. Given that $\mathcal{A}$ wins the game—i.e., $\mathcal{A}$ comes up with b'_0, b'_1 such that $(b_0, b_1) = (b'_0, b'_1)$—with non-negligible probability, it implies that $\mathcal{A}$ guesses simultaneously b_0 with $(m_{0,0}, m_{0,1}), \mathsf{ct}_0$ and b_1 with $(m_{1,0}, m_{1,1}), \mathsf{ct}_1$ correctly with non-negligible probability.

Let tag_0 and tag_1 denote two distinct random string used in the statement of ct_0 and ct_1 respectively. Following the same argument in proving security, we learn that by running $\mathcal{E}$ it yields $\geq (1-f)n$ valid signatures on tag_0 and $\geq (1-f)n$ valid signatures on tag_1. Recall that, by interacting with $\mathcal{O}^{\mathsf{Dec}}$, $\mathcal{A}$ acquires at most $n - |J|$ valid signatures on either tag_0 or tag_1. W.l.o.g., let us assume that honest parties sign on tag_0. We learn that $\mathcal{A}$, when controlling the set of public keys $\{\mathsf{pk}_j\}_{j \in J}$, successfully generates at least one σ_1 that verifies tag_1 under $\mathsf{pk}^* \in [n] \setminus J$. Let σ_0 be the signature produced by interacting $\mathcal{O}^{\mathsf{Dec}}$ with pk^*, $\mathcal{A}_{\mathsf{OSS}}$ wins the OSS security game with the tuple $(\mathsf{pk}^*, \mathsf{tag}_0, \mathsf{tag}_1, \sigma_0, \sigma_1)$. □

5 Proactive Secret Sharing

We present our proactive secret sharing protocols assuming no secure erasures in this section. The mobile adversary we consider spans from the weakest passive stationary adversary (for which we present a straightforward construction following Section 4.3), to the passive dynamic adversary (in Section 5.1), and finally we consider the active mobile adversary (in Section 5.2).

Constructions against passive stationary mobile adversary. As a starting point, we consider PSS with a stationary committee against a passive adversary $\mathcal{A}$. Note that, in the classical setting, without secure erasures, it is impossible to implement proactive security against such a (relatively) weak adversary, as the secret is unavoidably exposed to $\mathcal{A}$ after he passively corrupts all parties (cf. Section 3).

In the quantum world, we present a simple scheme that can tolerate passive corruptions. The main idea is to publish the secret in an encrypted form using the threshold PKE with one-shot decryption as introduced in Section 4.2; the decryption threshold is set the same as the threshold required by reconstruction. Thus, the shares of the proactive secret sharing will correspond to the decryption keys of the underlying threshold PKE with one-shot decryption. To reconstruct, parties locally generate partial decryptions and the secret can be retrieved after gathering sufficiently many shares. Regarding proactive security, we note that in the stationary setting, there is no need for parties to perform any operations during the handover stage. The intuitive reason is because a passive adversary can only perform deterministic operations on the internal quantum state (recall that $\mathcal{A}$ cannot do anything that leads to an unusable quantum state), which would not be helpful to acquire any valid one-shot signature.

5.1 Constructions Against Passive Dynamic Mobile Adversary

We have shown at the beginning of this section that proactive secret sharing among a stationary committee against a passive mobile adversary does not need any handover operations. Nonetheless, it also prohibits parties from updating the share-holding committee. Considering real-world applications, e.g., long-term distributed private information storage on a public blockchain, such limitation is undesirable.

In this section, we show a new proactive secret sharing protocol extending the base protocol, which enables parties to dynamically transfer their committee membership to others. I.e., a current committee member can nominate one committee member in the next epoch.

Our new construction again makes use of one-shot signatures and post-quantum witness encryption. The setup procedure remains the same: After generating the CRS of the underlying OSS scheme, each of the committee members of the first epoch locally generates a key pair $(\mathsf{pk}, |\mathsf{sk}\rangle)$ and publishes pk. Then, the dealer encrypts its secret to ct using witness encryption, but to a more elaborated relation $\mathsf{R}_{\mathsf{PSS\text{-}PD}}$ (Equation (3)). The statement consists of the common reference string CRS and the list of public keys in the first epoch $(\mathsf{pk}_i^1)_{i\in[n]}$; and the witness is represented by the n one-shot signature chains (Equation (2)).[9]

In more detail, a valid signature chain $(\sigma^{\mathsf{ep}})_{\mathsf{ep}\in[m]}$ of public key pk^1 is a chain of signatures where each $\mathsf{pk}^{\mathsf{ep}}, \mathsf{ep} \in [m]$ verifies the ep-th signature σ^{ep} on $\mathsf{pk}^{\mathsf{ep}+1}$. Alongside with the next public key, parties also sign on epoch index and either $\texttt{OPEN}$ or $\texttt{UPDATE}$ indicating whether they would like to reconstruct the secret or not. Finally, it is required that the last signature on chain must sign on $\texttt{OPEN}$. We define the predicate $\mathsf{ValidSigChain}$ in Equation (2).

$$\begin{aligned}&\mathsf{VaidSigChain}\big(\mathsf{CRS}, \{\mathsf{pk}^{\mathsf{ep}}\}_{\mathsf{ep}\in[m+1]}, (\sigma^{\mathsf{ep}})_{\mathsf{ep}\in[m]}\big)\\ &\triangleq \left\{\begin{array}{c}\forall \mathsf{ep}\in[m-1], \left(\begin{array}{c}\mathsf{OSS.Verify}(\mathsf{CRS}, \mathsf{pk}^{\mathsf{ep}}, \texttt{UPDATE} \,\|\, \mathsf{ep} \,\|\, \mathsf{pk}^{\mathsf{ep}+1}, \sigma^{\mathsf{ep}}) = 1\\ \vee \mathsf{OSS.Verify}(\mathsf{CRS}, \mathsf{pk}^{\mathsf{ep}}, \texttt{OPEN} \,\|\, \mathsf{ep} \,\|\, \mathsf{pk}^{\mathsf{ep}+1}, \sigma^{\mathsf{ep}}) = 1\end{array}\right)\\ \wedge \mathsf{OSS.Verify}(\mathsf{CRS}, \mathsf{pk}^m, \texttt{OPEN} \,\|\, m \,\|\, \mathsf{pk}^{m+1}, \sigma^m) = 1\end{array}\right\}. \end{aligned} \tag{2}$$

Given the signature chain validation predicate, the relation $\mathsf{R}_{\mathsf{PSS\text{-}PD}}$ used in witness encryption is defined as:

$$\mathsf{R}_{\mathsf{PSS\text{-}PD}} \triangleq \left\{ \begin{array}{c} \mathsf{stmt} = (\mathsf{CRS}, (\mathsf{pk}_i^1)_{i\in[n]}),\\ \mathsf{w} = (((\mathsf{pk}_j^{\mathsf{ep}+1}, \sigma_j^{\mathsf{ep}})_{\mathsf{ep}\in[m]})_{j\in[n]}) \end{array} \middle| \begin{array}{c} \forall j\in[n], \mathsf{validSigChain}(\mathsf{CRS},\\ \{\mathsf{pk}_j^{\mathsf{ep}}\}_{\mathsf{ep}\in[m+1]}, (\sigma_j^{\mathsf{ep}})_{\mathsf{ep}\in[m]}) = 1 \end{array} \right\}. \tag{3}$$

When entering a handover stage, committee members in epoch ep with public keys $(\mathsf{pk}_i^{\mathsf{ep}})_{i\in[n]}$ transfer the shares to the next committee $(\mathsf{pk}_i^{\mathsf{ep}+1})_{i\in[n]}$ as follows. For each party, they may choose to stay in the committee, by locally generating a new key pair $(\mathsf{pk}^{\mathsf{ep}+1}, |\mathsf{sk}^{\mathsf{ep}+1}\rangle)$ and use $|\mathsf{sk}^{\mathsf{ep}}\rangle$ to sign on $\texttt{UPDATE} \,\|\, \mathsf{ep} \,\|\, \mathsf{pk}^{\mathsf{ep}+1}$. Otherwise, they can transfer their committee membership to a new public key $\mathsf{pk}^{\mathsf{ep}+1}$ by signing on $\mathsf{pk}^{\mathsf{ep}+1}$ in the same way.

During handover, parties can alternatively sign on $\texttt{OPEN} \,\|\, \mathsf{ep} \,\|\, \mathsf{pk}^{\mathsf{ep}+1}$, indicating that they would like to open the secret. If, after the handover stage on epoch ep, all parties publish signatures with $\texttt{OPEN}$, then anyone can reconstruct the secret by decrypting ct.

We present the full protocol in Protocol 1 and claim that it is a secure proactive secret sharing scheme in Theorem 3 (note that secret usability is treated separately in Section 6).

Remark 3. We note that our presented protocol focus on the feasibility result—i.e., to construct PSS protocols using chains of one-shot signatures. The protocol is tweak-able to implement more complicated functionalities, by encrypting the secret to a more crafted relation. For example, it is possible to provide alternative ways to transfer committee memberships, or allow parties to split (merge resp.) their committee membership to (from resp.) multiple ones.

[9] Since we consider a passive mobile adversary, for brevity we require presenting all signature chains. Actually, any $f > 1/2$ suffices to work.

Protocol PSS-Passive-Dynamic

Let $\mathsf{OSS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ be a one-shot signature scheme as Definition 5, and $\mathsf{WE} = (\mathsf{Enc}, \mathsf{Dec})$ be a witness-encryption scheme as Definition 6. The relation $\mathsf{R}_{\mathsf{PSS\text{-}PD}}$ used for encryption is defined as in Equation (3).

$\underline{\mathsf{Setup}(1^\lambda)}$
$\mathsf{CRS} \leftarrow \mathsf{OSS.Setup}(1^\lambda)$
return CRS

$\underline{\mathsf{KeyGen}(\mathsf{CRS}, \mathsf{ep}, i)}$
$(\mathsf{pk}_i^{\mathsf{ep}}, |\mathsf{sk}_i^{\mathsf{ep}}\rangle) \leftarrow \mathsf{OSS.KeyGen}(\mathsf{CRS})$
broadcast $\mathsf{pk}_i^{\mathsf{ep}}$

$\underline{\mathsf{Share}(\mathsf{secret}, \mathsf{CRS}, (\mathsf{pk}_i^1)_{i\in[n]})}$
$\mathsf{ct} \leftarrow \mathsf{WE.Enc}((\mathsf{CRS}, (\mathsf{pk}_i^1)_{i\in[n]}), \mathsf{secret})$
broadcast ct

$\underline{\mathsf{Handover}(\mathsf{CRS}, \mathsf{ep}, \mathsf{pk}_i^{\mathsf{ep}+1}, \mathsf{choice})}$
$\sigma_i^{\mathsf{ep}} \leftarrow \mathsf{OSS.Sign}(\mathsf{CRS}, |\mathsf{sk}_i^{\mathsf{ep}}\rangle, \mathsf{choice} \,\|\, \mathsf{ep} \,\|\, \mathsf{pk}_i^{\mathsf{ep}+1})$
broadcast $(\mathsf{pk}_i^{\mathsf{ep}+1}, \sigma_i^{\mathsf{ep}})$

$\underline{\mathsf{Reconstruct}(\mathsf{CRS}, \mathsf{ep}, \mathsf{ct}, \{(m_j^{\mathsf{ep}}, \sigma_j^{\mathsf{ep}})_{\mathsf{ep}\in[m]}\}_{j\in[n]})}$
$\mathsf{secret} \leftarrow \mathsf{WE.Dec}(\mathsf{ct}, \{(m_j^{\mathsf{ep}}, \sigma_j^{\mathsf{ep}})_{\mathsf{ep}\in[m]}\}_{j\in[n]})$
return secret

$\underline{\mathsf{MainProtocol}(\mathsf{CRS}, \mathsf{P}_i, \mathsf{ep})}$
if $\mathsf{ep} = 1$ **then**
    Invoke $\mathsf{KeyGen}(\mathsf{CRS}, 1, i)$
    Store ct as received from the dealer's Share
    return
end if
Invoke $\mathsf{KeyGen}(\mathsf{CRS}, \mathsf{ep}, i)$
Let $\mathsf{choice} \in \{\texttt{UPDATE}, \texttt{OPEN}\}$ be P_i's choice on reconstruction or not
Invoke $\mathsf{Handover}(\mathsf{CRS}, \mathsf{ep}-1, \mathsf{pk}_i^{\mathsf{ep}}, \mathsf{choice})$
Record $(\sigma_i^{\mathsf{ep}})_{i\in[n]}$ the signatures received in the current epoch
if $\mathsf{choice} = \texttt{OPEN}$ **then**
    Let $\mathsf{secret} = \mathsf{Reconstruct}(\mathsf{CRS}, \mathsf{ep}, \mathsf{ct}, ((m_i^{\mathsf{ep}'}, \sigma_i^{\mathsf{ep}'})_{\mathsf{ep}\in[m]})_{i\in[n]})$
    if $\mathsf{secret} \neq \perp$ **then return** secret
end if

Protocol 1: *Dynamic-committee proactive secret sharing against passive mobile adversary.*

Theorem 3. *Assuming one-shot signatures, post-quantum extractable witness encryption, and* no *secure erasures, Protocol 1 is a dynamic-committee proactive secret sharing protocol satisfying secrecy and robust construction (as defined in Definitions 1 to 3) against a f-bounded passive mobile adversary for any $f < 1$.*

Proof. In each epoch ep, ct and shares $(|\mathsf{sk}_i^{\mathsf{ep}}\rangle)_{i\in[n]}$ forms an instance of the threshold PKE with one-shot decryption scheme (Definition 10). Thus, robust reconstruction directly follows the correctness property of the threshold PKE with one-shot decryption scheme.

Regarding secrecy, consider two distinct secrets $s_0 \neq s_1$. Let $\mathsf{Exec}^{\mathsf{ep}}_{\mathbf{P},\mathbf{I},\mathcal{A}}(\lambda, s_0, \mathsf{aux}_\lambda)$ be an execution up to epoch ep, where dealer shares s_0 (and publishes ct_0) and there does not exist any epoch where all parties sign on OPEN. Next, we consider another execution $\mathsf{Exec}^{\mathsf{ep}}_{\mathbf{P},\mathbf{I},\mathcal{A}}(\lambda, s_1, \mathsf{aux}_\lambda)$, where no epoch yields a reconstruction. The two executions share the same CRS and the set of public keys in the first epoch, yet in the second execution the dealer shares s_1 instead and publishes ct_1.

Suppose — towards a contradiction — that $\mathcal{A}$ can distinguish two views $\{\mathsf{View}^{\mathsf{ep}}_{\mathbf{P},\mathbf{I},\mathcal{A}}(\lambda, s_0, \mathsf{aux}_\lambda)\}_{\lambda\in\mathbb{N}}$ and $\{\mathsf{View}^{\mathsf{ep}}_{\mathbf{P},\mathbf{I},\mathcal{A}}(\lambda, s_1, \mathsf{aux}_\lambda)\}_{\lambda\in\mathbb{N}}$ with non-negligible advantage. W.l.o.g. assume such distinguishing advantage holds for $\mathsf{Exec}^{\mathsf{ep}}_{\mathbf{P},\mathbf{I},\mathcal{A}}(\lambda, s_0, \mathsf{aux}_\lambda)$ and $\mathsf{Exec}^{\mathsf{ep}}_{\mathbf{P},\mathbf{I},\mathcal{A}}(\lambda, s_1, \mathsf{aux}_\lambda)$. I.e., for $\mathsf{ct}_0 =$

$\mathsf{WE.Enc}((\mathsf{CRS}, (\mathsf{pk}_i^1)_{i\in[n]}), s_0)$ and $\mathsf{ct_1} = \mathsf{WE.Enc}((\mathsf{CRS}, (\mathsf{pk}_i^1)_{i\in[n]}), s_1)$, the adversary distinguishes with non-negligible advantage. By extractable security of the witness encryption scheme, we learn that for an execution E there is an extractor that can output n valid signature chains, i.e., n one-shot signatures $(\sigma_i)_{i\in[n]}$ signing on a string that begins with `OPEN` in the same epoch. Consider the following two cases:

- These one-shot signature are with respect to the same committee in an epoch $\mathsf{ep}' \leq \mathsf{ep}$ in E. Then, there is at least one public key pk^* that we come up with a tuple $(\mathsf{pk}^*, \texttt{UPDATE}, \texttt{OEPN}, \sigma, \sigma^*)$ that breaks the OSS security game, where $\sigma \in (\sigma_i)_{i\in[n]}$ is the signature in the execution signing on `UPDATE` and σ^* is generated by the extractor.
- The last epoch ep' of signature chains does not correspond to any epoch in the execution. This implies that there exist another epoch $\mathsf{ep}'' \leq \mathsf{ep}$ such that, in execution E, $(\mathsf{pk}_i^{\mathsf{ep}''})_{i\in[n]}$ produces $(\sigma_i^{\mathsf{ep}''})_{i\in[n]}$ on $(\mathsf{pk}_i^{\mathsf{ep}''+1})_{i\in[n]}$ respectively, while $\mathcal{A}$ manages to come up with at least one signature σ_j^* signing on $\mathsf{pk}_j^* \notin (\mathsf{pk}_i^{\mathsf{ep}''+1})_{i\in[n]}$ that verifies under $\mathsf{pk}_j^{\mathsf{ep}''}$. Thus, $\mathcal{A}$ manages to generates a tuple $(\mathsf{pk}_j^{\mathsf{ep}''}, \mathsf{pk}_j^{\mathsf{ep}''+1}, \mathsf{pk}_j^*, \sigma_j^{\mathsf{ep}''}, \sigma_j^*)$.

Both cases contradict the security property of the underlying OSS scheme, thus secrecy holds for Protocol 1. □

Succinct chains of signatures. In order to reconstruct the secret, Protocol 1 requires parties providing chains of all signatures from the onset of the protocol execution. In other words, witness size scales linearly with the protocol running time. Using simple techniques of post-quantum SNARKs (recall Definition 7), the size of the witness can be reduced to sub-linear with respect to protocol running time, thus getting rid of such undesired blow-up.[10]

We detail how we can augment Protocol 1. The NP-relation $\mathsf{R_{SNARK}}$ for SNARKs is defined as follows:

$$\mathsf{R_{SNARK}} \triangleq \left\{ \begin{array}{c} \mathsf{stmt} = (\mathsf{CRS}, \mathsf{pk}^1, \sigma), \\ \mathsf{w} = ((\sigma^{\mathsf{ep}})_{\mathsf{ep}\in[m]}) \end{array} \;\middle|\; \begin{array}{c} \mathsf{validSigChain}(\mathsf{CRS}, \mathsf{pk}^1, (\sigma^{\mathsf{ep}})_{\mathsf{ep}\in[m]}) = 1 \\ \wedge \sigma = \sigma^m \end{array} \right\}.$$

Let Π_{SNARK} be a SNARK scheme for NP-relation $\mathsf{R_{SNARK}}$, the dealer can encrypt her secret using witness encryption to the NP-relation $\mathsf{R_{PSS\text{-}PD\text{-}SNARK}}$ defined as follows:

$$\mathsf{R_{PSS\text{-}PD\text{-}SNARK}} \triangleq \left\{ \begin{array}{c} \mathsf{stmt} = (\mathsf{CRS}, (\mathsf{pk}_i^1)_{i\in[n]}), \\ \mathsf{w} = (\{\sigma_j, \pi_j\}_{j\in[n]}) \end{array} \;\middle|\; \forall j \in [n], \Pi_{\mathsf{SNARK}}.\mathsf{Verify}((\mathsf{CRS}, \mathsf{pk}_j^1, \sigma_j), \pi_j) = 1 \right\}.$$

Corollary 1. *Assuming one-shot signatures, post-quantum extractable witness encryption, post-quantum SNARK, and* no *secure erasures, there is a proactive secret sharing protocol satisfying secrecy and robust construction as defined in Definitions 1 to 3 against a f-bounded passive mobile adversary for any $f < 1$.*

5.2 Constructions Against Active Mobile Adversary

In this section we focus on active mobile adversary that can perform arbitrary operations over the corrupted internal states. The protocol introduced in Section 5.1 (which reconstructs the secret from signature chains) does not work in this setting, as an active adversary can simply destroy the internal quantum state of corrupted parties, thus preventing honest ones from eventually reconstructing the secret.

[10] Note that SNARKs are not necessarily zero-knowledge, thus it might leak information about on-chain signatures. Yet, for our applications we put aside such privacy concerns.

We present a new PSS protocol that is maliciously secure. At a high level, our new protocol still follows the similar paradigm as that in the previous section; with the main difference being that instead of parties locally handover their committee membership to a new party, we require that all parties sign on the next committee (i.e., a list of OSS verification keys).

The protocol setup stage remains the same as in Protocol 1; the dealer still encrypts her secret using witness encryption, yet to a more elaborated relation $\mathsf{R}_{\mathsf{PSS\text{-}Active}}$. Specifically, in $\mathsf{R}_{\mathsf{PSS\text{-}Active}}$, the statement is a common-reference string CRS and an array of OSS public keys in the first epoch $(\mathsf{pk}_i^1)_{i\in S^1}$. Note that, our protocol allows dynamic updates on the size of committees in each epoch, and we denote set of committee identifiers in epoch ep as S^{ep}.[11] Regarding the witness, it is an m-ary vector, where each element further consists two parts — a list of valid OSS signatures and a list of OSS public keys.

The relation $\mathsf{R}_{\mathsf{PSS\text{-}Active}}$ that we will use in witness encryption is defined as follows. The ep-th element $(\{\sigma_j^{\mathsf{ep}}\}_{j\in J^{\mathsf{ep}}}, (\mathsf{pk}_k^{\mathsf{ep}+1})_{k\in S^{\mathsf{ep}+1}})$ should form a valid update on the $(\mathsf{ep}-1)$-th element (for the first element, it should be a valid update on the statement $(\mathsf{pk}_i^1)_{i\in S^1}$). A pair of signatures and public keys $(\{\sigma_j^{\mathsf{ep}}\}_{j\in J^{\mathsf{ep}}}, (\mathsf{pk}_k^{\mathsf{ep}+1})_{k\in S^{\mathsf{ep}+1}})$ is a valid update on $(\mathsf{pk}_k^{\mathsf{ep}})_{k\in S^{\mathsf{ep}}}$ if: (i) J^{ep} is a subset of S^{ep} of size $|J| \geq (1-f)\cdot|S^{\mathsf{ep}}|$; and (ii) each signature σ_j signs on $\texttt{UPDATE}$, epoch index ep and next committee $(\mathsf{pk}_k^{\mathsf{ep}+1})_{k\in S^{\mathsf{ep}+1}}$, and verifies with $\mathsf{pk}_j^{\mathsf{ep}}$. Refer to Equation (4) for a formal specification of the $\mathsf{validUpdate}$ predicate.

$$\begin{aligned}&\mathsf{validUpdate}\Big(\mathsf{CRS}, (\mathsf{pk}_i^{\mathsf{ep}})_{i\in S^{\mathsf{ep}}}, \{\sigma_j\}_{j\in J}, (\mathsf{pk}_k^{\mathsf{ep}+1})_{k\in S^{\mathsf{ep}+1}}\Big)\\ &\qquad\triangleq \left\{\begin{array}{c}\forall j\in J, \mathsf{OSS.Verify}(\mathsf{CRS}, \mathsf{pk}_j^{\mathsf{ep}}, \texttt{UPDATE} \,\|\, \mathsf{ep} \,\|\, (\mathsf{pk}_k^{\mathsf{ep}+1})_{k\in S^{\mathsf{ep}+1}}, \sigma_j) = 1\\ \wedge J\subseteq S^{\mathsf{ep}} \wedge |J| \geq (1-f)\cdot|S^{\mathsf{ep}}|\end{array}\right\}.\end{aligned} \tag{4}$$

In addition, we define another predicate $\mathsf{validOpen}$ (Equation (5)) which is exactly the same as $\mathsf{validUpdate}$ except that all signatures sign on $\texttt{OPEN}$ instead of $\texttt{Update}$.

$$\begin{aligned}&\mathsf{validOpen}\Big(\mathsf{CRS}, (\mathsf{pk}_i^{\mathsf{ep}})_{i\in S^{\mathsf{ep}}}, \{\sigma_j\}_{j\in J}, (\mathsf{pk}_k^{\mathsf{ep}+1})_{k\in S^{\mathsf{ep}+1}}\Big)\\ &\qquad\triangleq \left\{\begin{array}{c}\forall j\in J, \mathsf{OSS.Verify}(\mathsf{CRS}, \mathsf{pk}_j^{\mathsf{ep}}, \texttt{OPEN} \,\|\, \mathsf{ep} \,\|\, (\mathsf{pk}_k^{\mathsf{ep}+1})_{k\in S^{\mathsf{ep}+1}}, \sigma_j) = 1\\ \wedge J\subseteq S^{\mathsf{ep}} \wedge |J| \geq (1-f)\cdot|S^{\mathsf{ep}}|\end{array}\right\}.\end{aligned} \tag{5}$$

Given the two predicates, the relation $\mathsf{R}_{\mathsf{PSS\text{-}Active}}$ that the dealer uses in witness encryption is $(m-1)$ valid updates followed by a valid open. Formally,

$$\mathsf{R}_{\mathsf{PSS\text{-}Active}} \triangleq \left\{\begin{array}{c}\mathsf{stmt} = (\mathsf{CRS}, \{\mathsf{pk}_i^1\}_{i\in S^1}),\\ \mathsf{w} = \left(\left(\begin{array}{c}\{\sigma_j^{\mathsf{ep}}\}_{j\in J^{\mathsf{ep}}},\\ (\mathsf{pk}_k^{\mathsf{ep}+1})_{k\in S^{\mathsf{ep}+1}}\end{array}\right)_{\mathsf{ep}\in[m]}\right)\end{array}\;\middle|\;\begin{array}{c}\forall \mathsf{ep}\in[m-1], \mathsf{validUpdate}(\mathsf{CRS},\\ (\mathsf{pk}_i^{\mathsf{ep}})_{i\in S^{\mathsf{ep}}}, \{\sigma_j\}_{j\in J^{\mathsf{ep}}}, (\mathsf{pk}_k^{\mathsf{ep}+1})_{k\in S^{\mathsf{ep}+1}}) = 1\\ \wedge \mathsf{validOpen}(\mathsf{CRS}, (\mathsf{pk}_i^m)_{i\in S^m},\\ \{\sigma_j\}_{j\in J^m}, (\mathsf{pk}_k^{m+1})_{k\in S^{m+1}}) = 1\end{array}\right\}. \tag{6}$$

At the onset of each epoch, parties engage into the handover protocol by first publishing their fresh OSS public keys. After reaching agreement over the next secret-holding committee (i.e., a set of public keys) and whether they prefer to retrieve the secret or not, parties sign accordingly and publish their one-shot signatures. Parties also book-keep all valid signatures that they have so far received—we shall show that with overwhelming probability, new signatures and the signed list of new keys always form a valid update on the current committee under the $\mathsf{validUpdate}$ predicate. In

[11] We remark that the approach that parties reach agreement on the next committee is out of the scope of this paper. Hence, in our protocol description we simply assume that such agreement is given; see, e.g., Line 17 in Protocol 2.

case that parties decide to reconstruct, they instead sign on `OPEN`; after collecting sufficiently many signatures, any party can retrieve the secret by decrypting ct published by the dealer.

We present the full protocol in Protocol 2. Note that, for brevity we describe the protocol with a minimum set of sub-components. Yet, we highlight that by employing the similar technique as we apply post-quantum SNARKs on Protocol 1, the protocol against active mobile adversary can be optimized to compressing the witness size to sub-linear with respect to the protocol execution time.

Protocol PSS-Active

Let $\mathsf{OSS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ be a one-shot signature scheme as Definition 5, and $\mathsf{WE} = (\mathsf{Enc}, \mathsf{Dec})$ be a witness-encryption scheme as Definition 6. The relation $\mathsf{R}_{\mathsf{PSS\text{-}Active}}$ used for encryption is defined as in Equation (6).

$\underline{\mathsf{Setup}(1^\lambda)}$
1: $\mathsf{CRS} \leftarrow \mathsf{OSS.Setup}(1^\lambda)$
2: **return** CRS

$\underline{\mathsf{KeyGen}(\mathsf{CRS}, \mathsf{ep}, i)}$
3: $(\mathsf{pk}_i^{\mathsf{ep}}, |\mathsf{sk}_i^{\mathsf{ep}}\rangle) \leftarrow \mathsf{OSS.KeyGen}(\mathsf{CRS})$
4: **broadcast** $\mathsf{pk}_i^{\mathsf{ep}}$

$\underline{\mathsf{Share}(\mathsf{secret}, \mathsf{CRS}, (\mathsf{pk}_i^1)_{i\in[n]})}$
5: $\mathsf{ct} \leftarrow \mathsf{WE.Enc}((\mathsf{CRS}, (\mathsf{pk}_i^1)_{i\in[n]}), \mathsf{secret})$
6: **broadcast** ct

$\underline{\mathsf{Handover}(\mathsf{CRS}, \mathsf{ep}, (\mathsf{pk}_i^{\mathsf{ep}+1})_{i\in S^{\mathsf{ep}+1}}, \mathsf{choice})}$
7: $\sigma_i \leftarrow \mathsf{OSS.Sign}(\mathsf{CRS}, |\mathsf{sk}_i^{\mathsf{ep}}\rangle, \mathsf{choice} \parallel \mathsf{ep} \parallel (\mathsf{pk}_i^{\mathsf{ep}+1})_{i\in S^{\mathsf{ep}+1}})$
8: **broadcast** σ_i

$\underline{\mathsf{Reconstruct}(\mathsf{CRS}, \mathsf{ep}, \mathsf{ct}, (\{\sigma_j^{\mathsf{ep}'}\}_{j\in J^{\mathsf{ep}'}}, (\mathsf{pk}_k^{\mathsf{ep}'+1})_{k\in S^{\mathsf{ep}'+1}})_{\mathsf{ep}'\in[\mathsf{ep}]})}$
9: $\mathsf{secret} \leftarrow \mathsf{WE.Dec}(\mathsf{ct}, (\{\sigma_j^{\mathsf{ep}'}\}_{j\in J^{\mathsf{ep}'}}, (\mathsf{pk}_k^{\mathsf{ep}'+1})_{k\in S^{\mathsf{ep}'+1}})_{\mathsf{ep}'\in[\mathsf{ep}]})$
10: **return** secret

$\underline{\mathsf{MainProtocol}(\mathsf{CRS}, \mathsf{P}_i, \mathsf{ep})}$
11: **if** $\mathsf{ep} = 1$ **then**
12: Invoke $\mathsf{KeyGen}(\mathsf{CRS}, 1, i)$
13: Store ct as received from the dealer's Share
14: **return**
15: **end if**
16: Invoke $\mathsf{KeyGen}(\mathsf{CRS}, \mathsf{ep}, i)$
17: Let $(\mathsf{pk}_i^{\mathsf{ep}})_{i\in S^{\mathsf{ep}}}$ be the agreed next committee list
18: Let $\mathsf{choice} \in \{\mathtt{UPDATE}, \mathtt{OPEN}\}$ be the agreed operation on the secret
19: Invoke $\mathsf{Handover}(\mathsf{CRS}, \mathsf{ep}-1, \mathsf{pk}_i^{\mathsf{ep}}, \mathsf{choice})$
20: Record $(\sigma_i^{\mathsf{ep}})_{i\in[n]}$ the signatures received in the current epoch
21: **if** $\mathsf{choice} = \mathtt{OPEN}$ **then**
22: $\mathsf{secret} = \mathsf{Reconstruct}(\mathsf{CRS}, \mathsf{ep}, \mathsf{ct}, (\{\sigma_j^{\mathsf{ep}'}\}_{j\in J^{\mathsf{ep}'}}, (\mathsf{pk}_k^{\mathsf{ep}'+1})_{k\in S^{\mathsf{ep}'+1}})_{\mathsf{ep}'\in[\mathsf{ep}]})$
23: **if** $\mathsf{secret} \neq \perp$ **then return** secret
24: **end if**

Protocol 2: *Dynamic-committee proactive secret sharing against active mobile adversary.*

Theorem 4. *Assuming one-shot signatures, post-quantum extractable witness encryption, and* no *secure erasures, Protocol 2 is a dynamic-committee proactive secret sharing protocol satisfying secrecy and robust construction (as defined in Definitions 1 to 3) against a f-bounded* active *mobile adversary for any $f < 1/2$.*

Proof. Robust reconstruction directly follows the correctness and security property of the threshold PKE with one-shot decryption scheme, as in each epoch, the encrypted secret and shares in that

epoch forms an instance of the threshold PKE with one-shot decryption scheme (Definition 10).

For secrecy, consider two distinct secrets $s_0 \neq s_1$. Let $\mathsf{Exec}^{\mathsf{ep}}_{\mathbf{P},\mathbf{I},\mathcal{A}}(\lambda, s_0, \mathsf{aux}_\lambda)$ and $\mathsf{Exec}^{\mathsf{ep}}_{\mathbf{P},\mathbf{I},\mathcal{A}}(\lambda, s_1, \mathsf{aux}_\lambda)$ be two executions defined in the same way as those in the proof of Theorem 3. Towards a contradiction, it suffices to show that an adversary can distinguish $\mathsf{Exec}^{\mathsf{ep}}_{\mathbf{P},\mathbf{I},\mathcal{A}}(\lambda, s_0, \mathsf{aux}_\lambda)$ and $\mathsf{Exec}^{\mathsf{ep}}_{\mathbf{P},\mathbf{I},\mathcal{A}}(\lambda, s_1, \mathsf{aux}_\lambda)$, and thus $\mathsf{ct}_0 = \mathsf{WE.Enc}((\mathsf{CRS}, (\mathsf{pk}_i^1)_{i\in[n]}), s_0)$ and $\mathsf{ct}_1 = \mathsf{WE.Enc}((\mathsf{CRS}, (\mathsf{pk}_i^1)_{i\in[n]}), s_1)$, with non-negligible advantage.

By extractable security of the witness encryption scheme, we learn that for an execution E there is an extractor that can output $m-1$ valid updates followed by one valid open as defined by Equations (4) and (5). We further consider the following two cases.

- In the valid open, all one-shot signature are with respect to the same committee in an epoch $\mathsf{ep}' \leq \mathsf{ep}$ in E. Then, there is at least one public key pk^* that we come up with a tuple $(\mathsf{pk}^*, \texttt{UPDATE}, \texttt{OEPN}, \sigma, \sigma^*)$ that breaks the OSS security game, where $\sigma \in (\sigma_i)_{i\in[n]}$ is the signature in the execution signing on `UPDATE` and σ^* is generated by the extractor.
- The valid open, consisted of signatures in epoch ep', does not correspond to any set of public keys in any epoch in the execution. This implies that there exist another epoch $\mathsf{ep}'' \leq \mathsf{ep}$ such that, in execution E, $(\mathsf{pk}_i^{\mathsf{ep}''})_{i\in[n]}$ produces $(\sigma_i^{\mathsf{ep}''})_{i\in[n]}$ on the same set of next-epoch signatures $(\mathsf{pk}_i^{\mathsf{ep}''+1})_{i\in[n]}$, while $\mathcal{A}$ manages to come up with at least one signature σ_j^* signing on $(\mathsf{pk}_i^*)_{i\in[n]} \neq (\mathsf{pk}_i^{\mathsf{ep}''+1})_{i\in[n]}$ that verifies under $\mathsf{pk}_j^{\mathsf{ep}''}$. Thus, $\mathcal{A}$ manages to generates a tuple $(\mathsf{pk}_j^{\mathsf{ep}''}, (\mathsf{pk}_i^*)_{i\in[n]}, (\mathsf{pk}_i^{\mathsf{ep}''+1})_{i\in[n]}, \sigma_j^{\mathsf{ep}''}, \sigma_j^*)$.

In either case, it contradict the security property of the underlying OSS scheme, thus secrecy holds for Protocol 2. □

6 Secret Usability

Proactive secret sharing protocols in Section 5 allows parties to reconstruct the original secret after collecting sufficiently many one-shot signatures. Nonetheless, in many applications one may wish to evaluate inputs using the secret without directly revealing it – recall the secret usability property (Definition 4).

In this section, we address this problem by employing additionally functional witness encryption (Definition 8) to enable evaluating the secret with inputs. We consider active mobile adversary, and the approach naturally applies to the passive case as well.

At the onset of the protocol, in addition to encrypting the secret with witness encryption as in Protocol 2, the dealer chooses a function F (we also write it as a Turing machine M) under which the secret is allowed to be evaluated, and publishes a functional witness encryption of the secret to relation $\mathsf{R}_{\mathsf{PSS\text{-}Eval}}$ and a Turning machine M. The relation that is used in the functional witness encryption consists of the same statement as the one used in Protocol 2, and the witness which consists of $m-1$ valid updates/evaluations followed by a valid evaluation. The $\mathsf{validEval}$ predicate is given in Equation (7), which is the same as $\mathsf{validUpdate}$ except that sufficiently many parties unanimously sign on `EVAL` and an additional input input.

$$\begin{aligned}&\mathsf{validEval}\Big(\mathsf{CRS}, (\mathsf{pk}_i^{\mathsf{ep}})_{i\in S^{\mathsf{ep}}}, \{\sigma_j\}_{j\in J}, (\mathsf{pk}_k^{\mathsf{ep}+1})_{k\in S^{\mathsf{ep}+1}}, \mathsf{input}\Big)\\ &\quad\triangleq \left\{\begin{array}{c}\forall j \in J, \mathsf{OSS.Verify}(\mathsf{CRS}, \mathsf{pk}_j^{\mathsf{ep}}, \texttt{EVAL} \parallel \mathsf{input} \parallel \mathsf{ep} \parallel (\mathsf{pk}_k^{\mathsf{ep}+1})_{k\in S^{\mathsf{ep}+1}}, \sigma_j) = 1\\ \wedge J \subseteq S^{\mathsf{ep}} \wedge |J| \geq (1-f)\cdot|S^{\mathsf{ep}}|\end{array}\right\}.\end{aligned} \tag{7}$$

The following predicate $\mathsf{validUpdateOrEval}$ checks if the given input is either a valid update or

evaluation.

$$\mathsf{validUpdateOrEval}(\mathsf{CRS}, (\mathsf{pk}_i^{\mathsf{ep}})_{i\in S^{\mathsf{ep}}}, \{\sigma_j\}_{j\in J}, (\mathsf{pk}_k^{\mathsf{ep}+1})_{k\in S^{\mathsf{ep}+1}}, \mathsf{input}) \triangleq \begin{pmatrix} \mathsf{validUpdate}(\mathsf{CRS}, (\mathsf{pk}_i^{\mathsf{ep}})_{i\in S^{\mathsf{ep}}}, \{\sigma_j\}_{j\in J}, (\mathsf{pk}_k^{\mathsf{ep}+1})_{k\in S^{\mathsf{ep}+1}}) \\ \vee \mathsf{validEval}(\mathsf{CRS}, (\mathsf{pk}_i^{\mathsf{ep}})_{i\in S^{\mathsf{ep}}}, \{\sigma_j\}_{j\in J}, (\mathsf{pk}_k^{\mathsf{ep}+1})_{k\in S^{\mathsf{ep}+1}}, \mathsf{input}) \end{pmatrix}.$$

Given these new predicates, the relation $\mathsf{R}_{\mathsf{PSS\text{-}Eval}}$ that is used in the functional witness encryption is defined formally as:

$$\mathsf{R}_{\mathsf{PSS\text{-}Eval}} \triangleq \left\{ \begin{array}{c} \mathsf{stmt} = (\mathsf{CRS}, \{\mathsf{pk}_i^1\}_{i\in S^1}), \\ \mathsf{w} = \left(\begin{pmatrix} \{\sigma_j^{\mathsf{ep}}\}_{j\in J^{\mathsf{ep}}}, \\ (\mathsf{pk}_k^{\mathsf{ep}+1})_{k\in S^{\mathsf{ep}+1}}, \\ \mathsf{input}^{\mathsf{ep}} \end{pmatrix}_{\mathsf{ep}\in[m]} \right) \end{array} \middle| \begin{array}{c} \forall \mathsf{ep} \in [m-1], \mathsf{validUpdateOrEval}(\mathsf{CRS}, \\ (\mathsf{pk}_i^{\mathsf{ep}})_{i\in S^{\mathsf{ep}}}, \{\sigma_j\}_{j\in J^{\mathsf{ep}}}, (\mathsf{pk}_k^{\mathsf{ep}+1})_{k\in S^{\mathsf{ep}+1}}) = 1 \\ \wedge \mathsf{validEval}(\mathsf{CRS}, (\mathsf{pk}_i^m)_{i\in S^m}, \\ \{\sigma_j\}_{j\in J^m}, (\mathsf{pk}_k^{m+1})_{k\in S^{m+1}}, \mathsf{input}^{\mathsf{ep}}) = 1 \end{array} \right\}. \tag{8}$$

Note that, after introducing the function evaluation operation into our protocol, the relation $\mathsf{R}_{\mathsf{PSS\text{-}Active}}$ that is used accordingly should be amended as well to admit valid evaluations before the final valid open. We omit the formal definition of this updated relation. Refer to Protocol 3 for a full specification of the protocol.

Protocol PSS-SecretUsability

Let $\mathsf{OSS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ be a one-shot signature scheme as Algorithm 4, $\mathsf{WE} = (\mathsf{Enc}, \mathsf{Dec})$ be a witness encryption scheme as Definition 6, and $\mathsf{FWE} = (\mathsf{Enc}, \mathsf{Dec})$ be a functional witness encryption scheme as Definition 8. The witness encryption scheme is for relation $\mathsf{R}_{\mathsf{PSS\text{-}Active}}$; and the relation $\mathsf{R}_{\mathsf{PSS\text{-}Eval}}$ used for encryption is defined as in Equation (8).

$\underline{\mathsf{Setup}(1^\lambda)}$
1: $\mathsf{CRS} \leftarrow \mathsf{OSS.Setup}(1^\lambda)$
2: **return** CRS

$\underline{\mathsf{KeyGen}(\mathsf{CRS}, \mathsf{ep}, i)}$
3: $(\mathsf{pk}_i^{\mathsf{ep}}, |\mathsf{sk}_i^{\mathsf{ep}}\rangle) \leftarrow \mathsf{OSS.KeyGen}(\mathsf{CRS})$
4: **broadcast** $\mathsf{pk}_i^{\mathsf{ep}}$

$\underline{\mathsf{Handover}(\mathsf{CRS}, \mathsf{ep}, (\mathsf{pk}_i^{\mathsf{ep}+1})_{i\in S^{\mathsf{ep}+1}}, \mathsf{choice}, \mathsf{input})}$
8: **if** $\mathsf{choice} \in \{\texttt{UPDATE}, \texttt{OPEN}\}$ **then**
9: $\quad \sigma_i \leftarrow \mathsf{OSS.Sign}(\mathsf{CRS}, |\mathsf{sk}_i^{\mathsf{ep}}\rangle, \mathsf{choice} \| \mathsf{ep} \| (\mathsf{pk}_i^{\mathsf{ep}+1})_{i\in S^{\mathsf{ep}+1}})$
10: **else**
11: $\quad \sigma_i \leftarrow \mathsf{OSS.Sign}(\mathsf{CRS}, |\mathsf{sk}_i^{\mathsf{ep}}\rangle, \mathsf{choice} \| \mathsf{ep} \| (\mathsf{pk}_i^{\mathsf{ep}+1})_{i\in S^{\mathsf{ep}+1}} \| \mathsf{input})$
12: **end if**
13: **broadcast** σ_i

$\underline{\mathsf{Reconstruct}(\mathsf{CRS}, \mathsf{ep}, \mathsf{ct}, (\{\sigma_j^{\mathsf{ep}'}\}_{j\in J^{\mathsf{ep}'}}, (\mathsf{pk}_k^{\mathsf{ep}'+1})_{k\in S^{\mathsf{ep}'+1}})_{\mathsf{ep}'\in[\mathsf{ep}]})}$
14: $\mathsf{secret} \leftarrow \mathsf{WE.Dec}(\mathsf{ct}, (\{\sigma_j^{\mathsf{ep}'}\}_{j\in J^{\mathsf{ep}'}}, (\mathsf{pk}_k^{\mathsf{ep}'+1})_{k\in S^{\mathsf{ep}'+1}})_{\mathsf{ep}'\in[\mathsf{ep}]})$
15: **return** secret

$\underline{\mathsf{Eval}(\mathsf{CRS}, \mathsf{ep}, \mathsf{ct}_{\mathsf{F}}, (\{\sigma_j^{\mathsf{ep}'}\}_{j\in J^{\mathsf{ep}'}}, (\mathsf{pk}_k^{\mathsf{ep}'+1})_{k\in S^{\mathsf{ep}'+1}}, \mathsf{input}^{\mathsf{ep}'})_{\mathsf{ep}'\in[\mathsf{ep}]})}$
16: $\mathsf{output} \leftarrow \mathsf{FWE.Dec}(\mathsf{ct}_{\mathsf{F}}, (\{\sigma_j^{\mathsf{ep}'}\}_{j\in J^{\mathsf{ep}'}}, (\mathsf{pk}_k^{\mathsf{ep}'+1})_{k\in S^{\mathsf{ep}'+1}}, \mathsf{input}^{\mathsf{ep}'})_{\mathsf{ep}'\in[\mathsf{ep}]})$
17: **return** output

$\underline{\mathsf{Share}(\mathsf{secret}, \mathsf{CRS}, (\mathsf{pk}_i^1)_{i\in[n]})}$
5: $\mathsf{ct} \leftarrow \mathsf{WE.Enc}((\mathsf{CRS}, (\mathsf{pk}_i^1)_{i\in[n]}), \mathsf{secret})$
6: $\mathsf{ct}_{\mathsf{F}} \leftarrow \mathsf{FWE.Enc}((\mathsf{CRS}, (\mathsf{pk}_i^1)_{i\in[n]}), \mathsf{secret}, \mathsf{M})$
7: **broadcast** $(\mathsf{ct}, \mathsf{ct}_{\mathsf{F}})$

$\underline{\mathsf{MainProtocol}(\mathsf{CRS}, \mathsf{P}_i, \mathsf{ep})}$
if $\mathsf{ep} = 1$ **then**
  Invoke $\mathsf{KeyGen}(\mathsf{CRS}, 1, i)$
  Store ct as received from the dealer's Share
  return
end if
Invoke $\mathsf{KeyGen}(\mathsf{CRS}, \mathsf{ep}, i)$
Let $(\mathsf{pk}_i^{\mathsf{ep}})_{i \in S^{\mathsf{ep}}}$ be the agreed next committee list
Let $\mathsf{choice} \in \{\texttt{UPDATE}, \texttt{OPEN}, \texttt{EVAL}\}$ be the agreed operation on the secret
Let input be the agreed evaluation input ($\mathsf{input} = \perp$ if $\mathsf{choice} \in \{\texttt{UPDATE}, \texttt{OPEN}\}$)
Invoke $\mathsf{Handover}(\mathsf{CRS}, \mathsf{ep}-1, \mathsf{pk}_i^{\mathsf{ep}}, \mathsf{choice}, \mathsf{input})$
Record $(\sigma_i^{\mathsf{ep}})_{i \in [n]}$ the signatures received in the current epoch
if $\mathsf{choice} = \texttt{OPEN}$ **then**
  $\mathsf{secret} = \mathsf{Reconstruct}(\mathsf{CRS}, \mathsf{ep}, \mathsf{ct}, (\{\sigma_j^{\mathsf{ep}'}\}_{j \in J^{\mathsf{ep}'}}, (\mathsf{pk}_k^{\mathsf{ep}'+1})_{k \in S^{\mathsf{ep}'+1}})_{\mathsf{ep}' \in [\mathsf{ep}]})$
  if $\mathsf{secret} \neq \perp$ **then return** secret
else if $\mathsf{choice} = \texttt{EVAL}$ **then**
  Let $\mathsf{output} = \mathsf{Eval}(\mathsf{CRS}, \mathsf{ep}, \mathsf{ct}_{\mathsf{F}}, (\{\sigma_j^{\mathsf{ep}'}\}_{j \in J^{\mathsf{ep}'}}, (\mathsf{pk}_k^{\mathsf{ep}'+1})_{k \in S^{\mathsf{ep}'+1}})_{\mathsf{ep}' \in [\mathsf{ep}]})$
  if $\mathsf{output} \neq \perp$ **then return** output
end if

Protocol 3: *Proactive secret sharing with secret usability.*

Theorem 5. *Assuming one-shot signatures, post-quantum extractable witness encryption, and* no *secure erasures, Protocol 3 is a dynamic-committee proactive secret sharing protocol satisfying secrecy, robust construction and secret usability (cf. Definitions 1 to 4) against a f-bounded* active *mobile adversary for any $f < 1/2$.*

Proof. The proof of this theorem follows generally the same structure as that for Theorem 4, thus we only discuss their difference.

Robust reconstruction is proved in the same way as the previous theorem. Regarding secrecy, the argument is exactly the same as arguing for secrecy in Theorem 4, except that we additionally consider ct_{F}—using the same technique we prove that if an adversary manages to distinguish the two secrets, then there is an extractor of the underlying functional witness encryption scheme that outputs a set of $(1-f)n$ one-shot-signatures signing on EVAL, yet honest parties never sign on evaluation in the execution, thus contradicting the security property of the underlying OSS scheme.

Finally, we consider secret usability. First, the correctness property of the underlying functional witness encryption scheme ensures that all evaluated outcomes are indeed the same as evaluating F over the secret and the input.

Next, if the adversary manages to distinguish with non-negligible advantage two executions, then it implies that an extractor outputs another sets of OSS signatures signing on inp^* such that $\forall i \in [\mathsf{ep}], \mathsf{inp}^* \neq \mathsf{inp}^i$. However, by considering that in the execution honest parties produce signatures on inp^i, following the similar argument for secrecy, it contradicts the security of the underlying OSS scheme. □

Finally we recall here that secret usability property is an indicative property meant to illustrate our technique. Depending on the use-case, it is easy to adapt the construction we presented above to capture various threshold cryptographic operations such as threshold signatures or threshold decryption.

Acknowledgments

AC and PW acknowledge support from the Quantum Advantage Turbo Charger grant (UKRI4257) and the Quantum Advantage Pathfinder (EP/X026167/1) both funded by UKRI's EPSRC. This work was also supported by Input Output (iog.io) through their funding of the Edinburgh Blockchain Technology Lab.

References

[Aar09] Scott Aaronson. Quantum copy-protection and quantum money. In *2009 24th Annual IEEE Conference on Computational Complexity*, pages 229–242, 2009. https://doi.org/10.1109/CCC.2009.42. (Cited on pages 4 and 16.)

[ABG+21] Amit Agarwal, James Bartusek, Vipul Goyal, Dakshita Khurana, and Giulio Malavolta. Post-quantum multi-party computation. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021, Part I*, volume 12696 of *Lecture Notes in Computer Science*, pages 435–464, Zagreb, Croatia, October 17–21, 2021. Springer, Cham, Switzerland. https://doi.org/10.1007/978-3-030-77870-5_16. (Cited on page 6.)

[ACL+22] Martin R. Albrecht, Valerio Cini, Russell W. F. Lai, Giulio Malavolta, and Sri Aravinda Krishnan Thyagarajan. Lattice-based SNARKs: Publicly verifiable, preprocessing, and recursively composable - (extended abstract). In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022, Part II*, volume 13508 of *Lecture Notes in Computer Science*, pages 102–132, Santa Barbara, CA, USA, August 15–18, 2022. Springer, Cham, Switzerland. https://doi.org/10.1007/978-3-031-15979-4_4. (Cited on pages 7 and 13.)

[AGKZ20] Ryan Amos, Marios Georgiou, Aggelos Kiayias, and Mark Zhandry. One-shot signatures and applications to hybrid quantum/classical authentication. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *52nd Annual ACM Symposium on Theory of Computing*, pages 255–268, Chicago, IL, USA, June 22–26, 2020. ACM Press. https://doi.org/10.1145/3357713.3384304. (Cited on pages 4, 5, 6, 8, 12, 13, and 35.)

[BCM+18] Zvika Brakerski, Paul Christiano, Urmila Mahadev, Umesh V. Vazirani, and Thomas Vidick. A cryptographic test of quantumness and certifiable randomness from a single quantum device. In Mikkel Thorup, editor, *59th Annual Symposium on Foundations of Computer Science*, pages 320–331, Paris, France, October 7–9, 2018. IEEE Computer Society Press. https://doi.org/10.1109/FOCS.2018.00038. (Cited on page 4.)

[BCP14] Elette Boyle, Kai-Min Chung, and Rafael Pass. On extractability obfuscation. In Yehuda Lindell, editor, *TCC 2014: 11th Theory of Cryptography Conference*, volume 8349 of *Lecture Notes in Computer Science*, pages 52–73, San Diego, CA, USA, February 24–26, 2014. Springer Berlin Heidelberg, Germany. https://doi.org/10.1007/978-3-642-54242-8_3. (Cited on page 14.)

[BCTV14] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Scalable zero knowledge via cycles of elliptic curves. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part II*, volume 8617 of *Lecture Notes in Computer Science*, pages 276–294, Santa Barbara, CA, USA, August 17–21, 2014. Springer Berlin Heidelberg, Germany. https://doi.org/10.1007/978-3-662-44381-1_16. (Cited on page 13.)

[BELO15] Joshua Baron, Karim El Defrawy, Joshua Lampkins, and Rafail Ostrovsky. Communication-optimal proactive secret sharing for dynamic groups. In Tal Malkin, Vladimir Kolesnikov, Allison Bishop Lewko, and Michalis Polychronakis, editors, *ACNS 2015: 13th International Conference on Applied Cryptography and Network Security*, volume 9092 of *Lecture Notes in Computer Science*, pages 23–41, New York, NY, USA, June 2–5, 2015. Springer, Cham, Switzerland. https://doi.org/10.1007/978-3-319-28166-7_2. (Cited on page 7.)

[BGG+20] Fabrice Benhamouda, Craig Gentry, Sergey Gorbunov, Shai Halevi, Hugo Krawczyk, Chengyu Lin, Tal Rabin, and Leonid Reyzin. Can a public blockchain keep a secret? In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020: 18th Theory of Cryptography Conference, Part I*, volume 12550 of *Lecture Notes in Computer Science*, pages 260–290, Durham, NC, USA, November 16–19, 2020. Springer, Cham, Switzerland. https://doi.org/10.1007/978-3-030-64375-1_10. (Cited on pages 3, 4, and 7.)

[BI20] Anne Broadbent and Rabib Islam. Quantum encryption with certified deletion. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020: 18th Theory of Cryptography Conference, Part III*, volume 12552 of *Lecture Notes in Computer Science*, pages 92–122, Durham, NC, USA, November 16–19, 2020. Springer, Cham, Switzerland. https://doi.org/10.1007/978-3-030-64381-2_4. (Cited on page 4.)

[BK23] James Bartusek and Dakshita Khurana. Cryptography with certified deletion. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023, Part V*, volume 14085 of *Lecture Notes in Computer Science*, pages 192–223, Santa Barbara, CA, USA, August 20–24, 2023. Springer, Cham, Switzerland. https://doi.org/10.1007/978-3-031-38554-4_7. (Cited on page 4.)

[Bla79] G. R. Blakley. Safeguarding cryptographic keys. *Proceedings of AFIPS 1979 National Computer Conference*, 48:313–317, 1979. (Cited on pages 3 and 7.)

[BR24] James Bartusek and Justin Raizes. Secret sharing with certified deletion. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology – CRYPTO 2024, Part VII*, volume 14926 of *Lecture Notes in Computer Science*, pages 184–214, Santa Barbara, CA, USA, August 18–22, 2024. Springer, Cham, Switzerland. https://doi.org/10.1007/978-3-031-68394-7_7. (Cited on page 4.)

[CH94] Ran Canetti and Amir Herzberg. Maintaining security in the presence of transient faults. In Yvo Desmedt, editor, *Advances in Cryptology – CRYPTO'94*, volume 839 of *Lecture Notes in Computer Science*, pages 425–438, Santa Barbara, CA, USA, August 21–25, 1994. Springer Berlin Heidelberg, Germany. https://doi.org/10.1007/3-540-48658-5_38. (Cited on page 7.)

[CLLZ21] Andrea Coladangelo, Jiahui Liu, Qipeng Liu, and Mark Zhandry. Hidden cosets and applications to unclonable cryptography. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part I*, volume 12825 of *Lecture Notes in Computer Science*, pages 556–584, Virtual Event, August 16–20, 2021. Springer, Cham, Switzerland. https://doi.org/10.1007/978-3-030-84242-0_20. (Cited on pages 8 and 16.)

[COS20] Alessandro Chiesa, Dev Ojha, and Nicholas Spooner. Fractal: Post-quantum and transparent recursive proofs from holography. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020, Part I*, volume 12105 of *Lecture Notes in Computer Science*, pages 769–793, Zagreb, Croatia, May 10–14, 2020. Springer, Cham, Switzerland. https://doi.org/10.1007/978-3-030-45721-1_27. (Cited on page 14.)

[DF90] Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 307–315, Santa Barbara, CA, USA, August 20–24, 1990. Springer, New York, USA. https://doi.org/10.1007/0-387-34805-0_28. (Cited on page 3.)

[FHAS24] Nils Fleischhacker, Mathias Hall-Andersen, and Mark Simkin. Extractable witness encryption for KZG commitments and efficient laconic OT. In Kai-Min Chung and Yu Sasaki, editors, *Advances in Cryptology – ASIACRYPT 2024, Part II*, volume 15485 of *Lecture Notes in Computer Science*, pages 423–453, Kolkata, India, December 9–13, 2024. Springer, Singapore, Singapore. https://doi.org/10.1007/978-981-96-0888-1_14. (Cited on pages 7 and 13.)

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th Annual ACM Symposium on Theory of Computing*, pages 467–476, Palo Alto, CA, USA, June 1–4, 2013. ACM Press. https://doi.org/10.1145/2488608.2488667. (Cited on page 13.)

[GHK+21] Craig Gentry, Shai Halevi, Hugo Krawczyk, Bernardo Magri, Jesper Buus Nielsen, Tal Rabin, and Sophia Yakoubov. YOSO: You only speak once - secure MPC with stateless ephemeral roles. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part II*, volume 12826 of *Lecture Notes in Computer Science*, pages 64–93, Virtual Event, August 16–20, 2021. Springer, Cham, Switzerland. https://doi.org/10.1007/978-3-030-84245-1_3. (Cited on pages 3 and 8.)

[GKP+13] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. How to run Turing machines on encrypted data. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 536–553, Santa Barbara, CA, USA, August 18–22, 2013. Springer Berlin Heidelberg, Germany. https://doi.org/10.1007/978-3-642-40084-1_30. (Cited on page 13.)

[Gut96] Peter Gutmann. Secure deletion of data from magnetic and Solid-State memory. In *6th USENIX Security Symposium (USENIX Security 96)*, San Jose, CA, July 1996. USENIX Association. URL: https://www.usenix.org/conference/6th-usenix-security-symposium/secure-deletion-data-magnetic-and-solid-state-memory. (Cited on page 3.)

[GZ20] Marios Georgiou and Mark Zhandry. Unclonable decryption keys. Cryptology ePrint Archive, Report 2020/877, 2020. URL: https://eprint.iacr.org/2020/877. (Cited on pages 8, 16, and 33.)

[HJKY95] Amir Herzberg, Stanislaw Jarecki, Hugo Krawczyk, and Moti Yung. Proactive secret sharing or: How to cope with perpetual leakage. In Don Coppersmith, editor, *Advances in Cryptology – CRYPTO'95*, volume 963 of *Lecture Notes in Computer Science*, pages 339–352, Santa Barbara, CA, USA, August 27–31, 1995. Springer Berlin Heidelberg, Germany. https://doi.org/10.1007/3-540-44750-4_27. (Cited on pages 3 and 7.)

[HMNY21] Taiga Hiroka, Tomoyuki Morimae, Ryo Nishimaki, and Takashi Yamakawa. Quantum encryption with certified deletion, revisited: Public key, attribute-based, and classical communication. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2021, Part I*, volume 13090 of *Lecture Notes in Computer Science*, pages 606–636, Singapore, December 6–10, 2021. Springer, Cham, Switzerland. https://doi.org/10.1007/978-3-030-92062-3_21. (Cited on page 4.)

[HV25] Andrew Huang and Vinod Vaikuntanathan. A simple and efficient one-shot signature scheme. Cryptology ePrint Archive, Report 2025/1906, 2025. URL: https://eprint.iacr.org/2025/1906. (Cited on page 8.)

[KS24] Jonathan Katz and Ben Sela. Secret sharing with publicly verifiable deletion. Cryptology ePrint Archive, Report 2024/1596, 2024. URL: https://eprint.iacr.org/2024/1596. (Cited on page 4.)

[KY25] Fuyuki Kitagawa and Takashi Yamakawa. Foundations of single-decryptor encryption. Cryptology ePrint Archive, Report 2025/1219, 2025. URL: https://eprint.iacr.org/2025/1219. (Cited on pages 8 and 16.)

[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. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019: 26th Conference on Computer and Communications Security*, pages 2369–2386, London, UK, November 11–15, 2019. ACM Press. https://doi.org/10.1145/3319535.3363203. (Cited on page 7.)

[OY91] Rafail Ostrovsky and Moti Yung. How to withstand mobile virus attacks (extended abstract). In Luigi Logrippo, editor, *10th ACM Symposium Annual on Principles of Distributed Computing*, pages 51–59, Montreal, QC, Canada, August 19–21, 1991. Association for Computing Machinery. https://doi.org/10.1145/112600.112605. (Cited on pages 3, 7, and 9.)

[Por23] Alexander Poremba. Quantum proofs of deletion for learning with errors. In Yael Tauman Kalai, editor, *ITCS 2023: 14th Innovations in Theoretical Computer Science Conference*, volume 251, pages 90:1–90:14, Cambridge, MA, USA, January 10–13, 2023. Leibniz International Proceedings in Informatics (LIPIcs). https://doi.org/10.4230/LIPIcs.ITCS.2023.90. (Cited on page 4.)

[Sha79] Adi Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, November 1979. https://doi.org/10.1145/359168.359176. (Cited on pages 3 and 7.)

[SZ25] Omri Shmueli and Mark Zhandry. On one-shot signatures, quantum vs. classical binding, and obfuscating permutations. In Yael Tauman Kalai and Seny F. Kamara, editors, *Advances in Cryptology – CRYPTO 2025, Part II*, volume 16001 of *Lecture Notes in Computer Science*, pages 350–383, Santa Barbara, CA, USA, August 17–21, 2025. Springer, Cham, Switzerland. https://doi.org/10.1007/978-3-032-01878-6_12. (Cited on pages 8, 12, 34, and 35.)

[Win99] A. Winter. Coding theorem and strong converse for quantum channels. *IEEE Transactions on Information Theory*, 45(7):2481–2485, 1999. https://doi.org/10.1109/18.796385. (Cited on page 16.)

A Single-Decryptor Encryption

For comparison, we describe the single-decryptor encryption scheme based on one-shot signatures due to Georgiou and Zhandry [GZ20].

Algorithm SingleDecryptorEncryption

Let $\mathsf{OSS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ be a one-shot signature scheme as defined in Algorithm 4.
Let $\mathsf{WE} = (\mathsf{Enc}, \mathsf{Dec})$ be a witness-encryption scheme for relation $\mathsf{R}_{\mathsf{SDE}}$:

$$\mathsf{R}_{\mathsf{SDE}} \triangleq \left\{ (\mathsf{CRS}, \mathsf{pk}, x), (\sigma) \;\middle|\; \mathsf{OSS.Verify}(\mathsf{CRS}, \mathsf{pk}, x, \sigma) = 1 \right\}.$$

$\underline{\mathsf{Setup}(1^\lambda)}$
Set $\mathsf{CRS} \leftarrow \mathsf{OSS.Setup}(1^\lambda)$
return CRS

$\underline{\mathsf{KeyGen}(\mathsf{CRS})}$
Set $(\mathsf{pk}, |\mathsf{sk}\rangle) \leftarrow \mathsf{OSS.KeyGen}(\mathsf{CRS})$
return $(\mathsf{pk}, |\mathsf{sk}\rangle)$

$\underline{\mathsf{Enc}(\mathsf{CRS}, \mathsf{pk}, m)}$
Pick $x \xleftarrow{\$} \{0,1\}^\lambda$
Compute $\mathsf{ct} \leftarrow \mathsf{WE.Enc}((\mathsf{CRS}, \mathsf{pk}, x), m)$
return ct

$\underline{\mathsf{Dec}(\mathsf{CRS}, |\mathsf{sk}\rangle, \mathsf{ct})}$
Parse the statement of ct as $(\mathsf{CRS}, \mathsf{pk}, x)$
Run on superposition the following:
- $\sigma \leftarrow \mathsf{OSS.Sign}(\mathsf{CRS}, |\mathsf{sk}\rangle, x)$
- $m \leftarrow \mathsf{WE.Dec}(c, \sigma)$

Measure and get m as a classical value
Rewind the computation and retrieve $|\mathsf{sk}\rangle$
return m

Algorithm 2: *The single-decryptor scheme using one-shot signatures [GZ20].*

B Preliminaries (Cont'd)

Definition 11. *A* collision-resistant always-non-collapsing hash function *is a pair of PPT algorithms* $(\mathsf{Setup}, \mathsf{Hash})$ *such that*

– **Collision-resistance:** *For any QPT algorithm $\mathcal{A}$, there exists a negligible function* negl *such that for all* λ*,*

$$\mathbf{Pr}\left[\mathsf{Hash}(\mathsf{CRS},x_0)=\mathsf{Hash}(\mathsf{CRS},x_1)\;\middle|\;\begin{matrix}\mathsf{CRS}\leftarrow\mathsf{Setup}(1^\lambda)\\(x_0,x_1)\leftarrow\mathcal{A}(\mathsf{CRS})\end{matrix}\right]\leq\mathsf{negl}(\lambda).$$

– **Always non-collapsing:** *There exists a pair of QPT algorithms $(\mathcal{S},\mathcal{D})$ and a negligible function* negl *such that for all* λ*,*

$$\left|\mathbf{Pr}\left[\mathcal{D}(x,\mathsf{aux})=1\;\middle|\;\begin{matrix}\mathsf{CRS}\leftarrow\mathsf{Setup}(1^\lambda),\\(|\psi\rangle,\mathsf{aux})\leftarrow\mathcal{S}(\mathsf{CRS}),\\x\leftarrow\mathsf{Measure}(|\psi\rangle)\end{matrix}\right]\right.$$
$$\left.-\mathbf{Pr}\left[\mathcal{D}(|\psi_y\rangle,\mathsf{aux})=1\;\middle|\;\begin{matrix}\mathsf{CRS}\leftarrow\mathsf{Setup}(1^\lambda),\\(|\psi\rangle,\mathsf{aux})\leftarrow\mathcal{S}(\mathsf{CRS}),\\|\psi_y\rangle\leftarrow\mathsf{PartialMeasure}_{\mathsf{Hash}(\mathsf{CRS},\cdot)}(|\psi\rangle)\end{matrix}\right]\right|\geq 1-\mathsf{negl}(\lambda).$$

Algorithm CR-NC-Hash

Let $\lambda\in\mathbb{N}$ be the security parameter. Define $s\triangleq 16\cdot\lambda$, $r\triangleq s\cdot(\lambda-1)$, $n\triangleq r+\frac{3}{2}\cdot s$, $k\triangleq n$.

$\underline{\mathsf{Setup}(1^\lambda)}$

1: Sample a random permutation $\Pi:\{0,1\}^n\to\{0,1\}^n$.
2: Sample a random function $F:\{0,1\}^r\to\{0,1\}^{k\cdot(n-r+1)}$.
3: Let $H(x)$ denote the first r bits of $\Pi(x)$, and $J(x)$ denote the last $n-r$ bits of $\Pi(x)$.
4: Let $\mathbf{A}(y)\in\mathbb{Z}_2^{k\times(n-r)}$ denote a random matrix with full column-rank, and $\mathbf{b}(y)\in\mathbb{Z}_2^k$ denote uniformly random, both genreated using $F(y)$.
5: Set $\mathcal{O}^P:\{0,1\}^n\to(\{0,1\}^r\times\mathbb{Z}_2^k)$, $\mathcal{O}^{P^{-1}}:(\{0,1\}^r\times\mathbb{Z}_2^k)\to\{0,1\}^n$ and $\mathcal{O}^\perp:(\{0,1\}^r\times\mathbb{Z}_2^k)\to\{0,1\}$ as follows:

$$\mathcal{O}^P(x)\triangleq(y,\mathbf{A}(y)\cdot J(x)+\mathbf{b}(y))\ \textit{where}\ y=H(x)$$

$$\mathcal{O}^{P^{-1}}(y,\mathbf{u})\triangleq\begin{cases}\Pi^{-1}(y,\mathbf{z}) & \exists\mathbf{z}\in\mathbb{Z}_2^{n-r}\ s.t.\ \mathbf{A}(y)\cdot\mathbf{z}+\mathbf{b}(y)=\mathbf{u}\\ \perp & else\end{cases}$$

$$\mathcal{O}^{\perp}(y,\mathbf{v})\triangleq\begin{cases}1 & if\ \mathbf{v}^T\cdot\mathbf{A}(y)=\mathbf{0}\\ 0 & otherwise\end{cases}$$

6: **return** $\mathsf{CRS}=(\mathcal{O}^P,\mathcal{O}^{P^{-1}},\mathcal{O}^\perp)$

$\underline{\mathsf{Hash}(\mathsf{CRS},x)}$

7: Parse CRS as $(\mathcal{O}^P,\cdot,\cdot)$ and compute $(y,\mathbf{u})\leftarrow\mathcal{O}^P(x)$
8: **return** y

Algorithm 3: *The collision-resistant always-non-collapsing hash function due to [SZ25].*

Algorithm OSS

$\underline{\mathsf{Setup}(1^\lambda)}$

1: Set $\mathsf{CRS}\leftarrow\mathsf{CR\text{-}NC\text{-}Hash.Setup}(1^\lambda)$ and **return** CRS

$\underline{\mathsf{KeyGen}(\mathsf{CRS})}$

Set $|\psi\rangle \leftarrow \sum_{x\in\mathbb{Z}_2^n} |x\rangle$

Execute $\mathsf{CR\text{-}NC\text{-}Hash.Hash}(\mathsf{CRS},\cdot)$ in superposition to obtain $\sum_{x\in\mathbb{Z}_2^n} |x\rangle\,|y_x\rangle$

Measure the second register and get y and $|\psi_y\rangle = \sum_{x\in\mathbb{Z}_2^n \wedge \mathsf{Hash}(\mathsf{CRS},x)=y} |x\rangle$

Set $\mathsf{pk} = y$ and $|\mathsf{sk}\rangle = |\psi_y\rangle$ and **return** $(\mathsf{pk}, |\mathsf{sk}\rangle)$

$\underline{\mathsf{Sign}(\mathsf{CRS}, |\mathsf{sk}\rangle, m)}$

Parse CRS as $(\mathcal{O}^P, \mathcal{O}^{P^{-1}}, \mathcal{O}^{\perp})$ and parse $|\mathsf{sk}\rangle$ as $\sum_{x\in\mathbb{Z}_2^n} \alpha_x |x\rangle$

Execute $\mathcal{O}^P$ in superposition to obtain $\sum_{x\in\mathbb{Z}_2^n} \alpha_x |x\rangle\,|y_x\rangle\,|\mathbf{u}_x\rangle$

Execute $\mathcal{O}^{P^{-1}}$ in superposition to uncompute the register holding x to obtain $\sum_{x\in\mathbb{Z}_2^n} \alpha_x |y_x\rangle\,|\mathbf{u}_x\rangle$

Execute a k-qubit Quantum Fourier Transform over $\mathbb{Z}_2$ on the rightmost k-qubit register (holding the vectors $\mathbf{u}_x$). This boils down to the execution of parallel Hadamard gates $H^{\otimes k}$ and obtains $\sum_{x\in\mathbb{Z}_2^n} \alpha_x |y_x\rangle\,(H^{\otimes k}\cdot|\mathbf{u}_x\rangle)$

Execute $\mathcal{O}^{\perp}(\cdot,\cdot)$ in superposition on the state and measure the output bit register b.

return b

$\underline{\mathsf{Verify}(\mathsf{CRS}, \mathsf{pk}, m, \sigma)}$

Set $y \leftarrow \mathsf{CR\text{-}NC\text{-}Hash.Hash}(\mathsf{CRS}, \sigma)$

if $y = \mathsf{pk} \wedge \sigma_0 = m$ **then return** true **else return** false

Algorithm 4: *The one-shot-signature scheme [AGKZ20, SZ25].*