Making Classical (Threshold) Signatures Post-Quantum for Single Use on a Public Ledger

Laurane Marco, Abdullah Talayhan, and Serge Vaudenay

EPFL, Switzerland
{laurane.marco, abdullah.talayhan, serge.vaudenay}@epfl.ch

Abstract. The Bitcoin architecture heavily relies on the ECDSA signature scheme which is broken by quantum adversaries as the secret key can be computed from the public key in quantum polynomial time. To mitigate this attack, bitcoins can be paid to the hash of a public key (P2PKH). However, the first payment reveals the public key so all bitcoins attached to it must be spent at the same time (i.e. the remaining amount must be transferred to a new wallet). Some problems remain with this approach: the owners are vulnerable against *rushing adversaries* between the time the signature is made public and the time it is committed to the blockchain. Additionally, there is no equivalent mechanism for threshold signatures. Finally, no formal analysis of P2PKH has been done. In this paper, we formalize the security notion of a *digital signature with a hidden public key* and we propose and prove the security of a generic transformation that converts a classical signature to a post-quantum one that can be used only once. We compare it with P2PKH. Namely, our proposal relies on pre-image resistance instead of collision resistance as for P2PKH, so allows for shorter hashes. Additionally, we propose the notion of a *delay signature* to address the problem of the rushing adversary when used with a public ledger and discuss the advantages and disadvantages of our approach. We further extend our results to threshold signatures.

1 Introduction

The recent progress made on the development of quantum computers poses a threat to public key cryptography schemes that are based on the discrete logarithm and factorization problems. Hence, it has become essential to find post-quantum alternatives to classical public key primitives. In this work, we focus on digital signatures, which are heavily used in cryptocurrencies. Despite a standardization effort made by the NIST, current post-quantum signatures are not yet mature enough, and struggle to compare with their classical counterparts, as shown by the new call for post-quantum signature proposals [15].

An alternative way to achieve post-quantum security of digital signatures, by using a simpler mechanism, is to hide the public key and release it along with the signature for verification. This however restricts key pairs to one-time use only, since in a quantum setting any key pair is trivially broken after its release. Bitcoin P2PKH (Pay To Public Key Hash) transactions [7] currently use a similar technique. More precisely, an account address consists of the hash of a public key and as long as a user does not spend the funds available on their account, only their address is revealed. The actual public key is released only when a transaction is made from this account. If a signer fears a quantum attack, they should collect all their unspent assets linked to the public key and make a new transaction to redistribute them to a new address (public key hash). The key pair is indeed broken after the transaction, but the wallet (related to the public key) does not contain any bitcoins left to be stolen.

However, the P2PKH construction has not been formalized nor proven secure, and it is open to attacks by a rushing quantum adversary between the release of the signature and the time the transaction is published to the ledger. Indeed, at this time any adversary who verifies the transaction is able to recover the secret key from the public one, and forge a new signature. If that signature is published on the ledger before the previous one, the attack is successful, and funds can be stolen. Furthermore, to the best of our knowledge, there is no threshold variant of this construction, and the rushing problem becomes worse in the threshold setting. Indeed, any signing participant which outputs the signature can proceed with such an attack, since the public key can be recovered from the signature. Additionally, without guaranteed output delivery, it might be the case that the honest signers do not even learn the original signature by the end of the protocol, while the adversary can forge a new one, since they get access to the common public key.

Our goal is to find a pragmatic solution to these problems that is efficient and secure against quantum adversaries whilst still being compatible with classical algorithms that are already in production. We stress that post-quantum signatures are the obvious solutions, but they are not fully standardized yet, are less efficient and rely on less studied assumptions. We thus propose our construction as a solution for a transitional phase, whilst waiting for efficient, standardized post-quantum signatures. In this work, we propose a generic transform that takes any classically secure signature scheme and turns it into a signature scheme with classical (with unlimited usage) and quantum (limited to one signature) security under minimal assumptions by hiding the public key (Section 2). We also formalize the existing construction used in Bitcoin P2PKH, analyze its security and compare it with our approach. Furthermore, we provide a framework for a delayed signature verification that protects the public key against rushing adversaries until the corresponding signature is published on a public ledger (Section 3). Finally, since threshold signatures are being used frequently in cryptocurrencies we extend the transform to the threshold setting (Section 4). Our extension of the generic transform to the threshold setting does not require a multi-party hash function evaluation (contrary to the post-quantum candidates), at the cost of an increase in the public key and signature size which is linear in the number of participants. In Appendix G, we introduce a non-generic threshold variant that only induces a small overhead of a single multi-party hash function evaluation with constant public key and signature size.

1.1 Related work

In this section, we discuss current classical signatures and post-quantum signature proposals, and their potential for constructing threshold signatures. We also introduce hash-based signatures as they share similar ideas to our construction.

Cozzo and Smart [6] estimate the performance of threshold versions of the NIST round 3 candidate schemes using generic MPC techniques. The most promising candidates are the multivariate schemes (Rainbow and LUOV), now broken ([3],[8]), as they only require linear secret sharing (LSSS). The other round 3 and 4 candidates (lattice-based, hash-based) are less friendly to generic MPC techniques, and require a tailored approach to be thresholdized. On the other hand, classical signatures,such as Schnorr and ECDSA signatures, already have practical and efficient threshold versions .
Hash-based signatures were introduced by Lamport. They are one-time signatures that rely on hash-functions, and signing requires to reveal the secret key. Since the unforgeability only relies on the security of the underlying one-way function it makes them a good candidate for post-quantum security. In fact, SPHINCS+ [2] is a NIST 4th round candidate and an instance of hash-based signature. The idea of post-quantum signatures designed for blockchain has already been explored in [5],[13]. They construct a post-quantum signature scheme from an optimised hash-based signature scheme with the objective of using it for blockchain applications. However, one common problem about constructing threshold variants of the hash based schemes is the need for multiple hash function evaluations over the secret shares. This problem motivated our approach that does not require a multi-party hash function evaluation. Finally the concept of hiding a public key to guarantee post-quantum security for blockchain transactions was introduced in [4], but it is not formalised nor proven secure, and no extension to the threshold case is considered.

1.2 Overview of the Construction

We now give an overview of our transformation and the methodology for releasing and verifying the signatures in order to give a complete picture of our contribution. We can simply explain our construction by putting together Figure 2, Figure 7, and Figure 9 as follows:
A key pair $(\mathsf{sk}, \mathsf{pk})$, e.g. for ECDSA, is transformed to

- $\mathsf{sk}' = (\mathsf{sk}, \rho, t_1, t_2)$
- $\mathsf{pk}' = (H(\rho), \mathsf{pk} + \rho, H(t_1), H(t_2))$

where ρ, t_1 and t_2 are sampled randomly. Intuitively, the public key is encrypted with a one-time pad key ρ, committed by $H(\rho)$. Two secret tokens t_1 and t_2 are also committed in pk'. Hence, the original public key pk is hidden using a committed secret key.

To sign a message msg, we compute a signature, e.g. ECDSA, on message msg with sk and obtain σ. Then we set:

- $\sigma_1 = H(\sigma, \rho, \mathsf{msg}, \mathsf{open})$
- $\sigma_2 = (\sigma, \rho, \mathsf{open})$

where open is a randomly sampled opening value. In order to publish the signature, the signer posts $(t_1, \sigma_1, \mathsf{msg})$ on the public ledger as a commitment for the actual signature σ (Stage 1). Once the signer observes that this commitment has been safely posted on the ledger, the signer posts t_2 on the ledger and further (t_1, σ_1) pairs are invalid afterwards (Stage 2). Finally, the signer posts σ_2 after confirming that t_2 is safely posted (Stage 3). This prevents forgeries by rushing adversaries. The signature (σ_1, σ_2) is verified for msg with public key $\mathsf{pk}' = (H(\rho), \mathsf{pk} + \rho, H(t_1), H(t_2))$ if the following conditions are satisfied:

1. σ_1 appears in the ledger between a preimage of $H(t_1)$ and a preimage of $H(t_2)$.
2. $(\sigma, \rho, \mathsf{open})$ parsed from σ_2 allows to verify $\sigma_1 = H(\sigma, \rho, \mathsf{msg}, \mathsf{open})$.
3. ρ is a preimage of $H(\rho)$.
4. After recovering pk using ρ, the original signature σ verifies the message msg.

Here, condition 1. checks that the order of commitments correctly appeared on the ledger. Condition 2. checks that the commitments are consistent with each other. Condition 3. verifies that the opening of the key commitment is correct and condition 4. checks if the original signature is actually a valid signature. The key idea is that a commitment on a signature must appear on the ledger between the publication of two tokens and the second token allows to safely publish pk.

For our extension to threshold signatures, the transformation is similar, except that the one-time pad key ρ and the public key pk are secretly shared. More concretely, we assume that each participant with a key pair $(\mathsf{pk}_i, \mathsf{sk}_i)$ will publish $\mathsf{pk}_i + \rho_i$ and keep ρ_i secret. The threshold signature will be augmented with the release of the ρ_i which will be used to uncover pk_i.

We summarize the overhead induced by $\mathsf{P2PKH}$ and our transformations in Table 1. Finally, one can find a reminder on the definitions concerning threshold signatures and their security, secret sharing and hash functions' security required for the rest of this work in Section A of the Appendix.

2 Hiding a Public Key

In this section, we give a generic transform that takes any classical signature scheme and hides the public key by adding a uniform masking value in order to obtain a post-quantum secure signature scheme with the single use of a public/private key pair. We then compare this transform with the Bitcoin P2PKH construction where the public key is hidden by using a hash function.

2.1 A Generic Transform

We present a transform that takes any signature scheme and allows one to sign while only disclosing the public key at signing time.
More precisely, it takes as input $\Sigma = (\mathsf{KeyGen}_\Sigma, \mathsf{Sign}_\Sigma, \mathsf{Verify}_\Sigma)$ with security parameter λ, and a hash function H. We assume that the public key domain $\mathcal{D}$ has a finite group structure with law $+$. The transform is illustrated in Figure 2.

	Key Generation	Signing	Verification	Signature Size	Assumption
P2PKH (Fig. 5)	1H	-	1H	$\|G_{\mathsf{el}}\|$	CR
HiddenPK (Fig. 2)	$1\mathsf{H} + 1\mathsf{G}_{\mathsf{add}}$	-	$1\mathsf{H} + 1\mathsf{G}_{\mathsf{add}}$	$\|G_{\mathsf{el}}\|$	2PreIm
DelayL (Fig. 9)	$3\mathsf{H} + 1\mathsf{G}_{\mathsf{add}}$	1H	$2\mathsf{LU} + 4\mathsf{H} + 1\mathsf{G}_{\mathsf{add}}$	$\|G_{\mathsf{el}}\|+\|\mathsf{H}\|+\lambda$	RO
Thresh-HiddenPK (Fig. 10)	$n\mathsf{H} + n\mathsf{G}_{\mathsf{add}}$	-	$\|S\|\mathsf{H} + \|S\|\mathsf{G}_{\mathsf{add}} + \mathcal{O}(\mathsf{Rec})$	$\|S\| \cdot \|G_{\mathsf{el}}\|$	2PreIm
Compact Thresh-HiddenPK (Fig. 23)	$\mathcal{O}(\mathsf{Shr}) + 1\mathsf{H}$	$\mathcal{O}(\mathsf{Rec})$	$1\mathsf{H} + 1\mathsf{G}_{\mathsf{add}}$	$\|G_{\mathsf{el}}\|$	2PreIm

Fig. 1: Complexity summary of the added overhead of the transformations - H denotes a hash function evaluation, $\mathsf{G}_{\mathsf{add}}$ a group operation, LU a table lookup of tokens, $\mathcal{O}(\mathsf{Shr})$ and $\mathcal{O}(\mathsf{Rec})$ the complexity of secret sharing and reconstruction, $|G_{\mathsf{el}}|$ the size of a group element, n is the number of parties involved in a threshold setting and S is the set of signers. CR stands for collision resistance, 2PreIm for 2nd pre-image resistance and RO for random oracle. λ is the security parameter.

```
HiddenPK-KeyGen($\lambda$)
1 : $(\mathsf{sk}, \mathsf{pk}) \leftarrow\$ \mathsf{KeyGen}_{\Sigma}(\lambda)$
2 : $\rho \leftarrow\$ \mathcal{D}$
3 : $\tilde{\mathsf{sk}} \leftarrow (\mathsf{sk}, \rho)$
4 : $\tilde{\mathsf{pk}} \leftarrow (H(\rho), \mathsf{pk} + \rho)$
5 : return $(\tilde{\mathsf{sk}}, \tilde{\mathsf{pk}})$

HiddenPK-Sign($\tilde{\mathsf{sk}}$, msg)
1 : $(\mathsf{sk}, \rho) \leftarrow \tilde{\mathsf{sk}}$
2 : $\sigma \leftarrow \mathsf{Sign}_{\Sigma}(\mathsf{sk}, \mathsf{msg})$
3 : $\tilde{\sigma} \leftarrow (\sigma, \rho)$
4 : return $\tilde{\sigma}$

HiddenPK-Verify($\tilde{\sigma}$, $\tilde{\mathsf{pk}}$, msg)
1 : $(\sigma, \rho) \leftarrow \tilde{\sigma}$
2 : $(\tilde{\mathsf{pk}}_1, \tilde{\mathsf{pk}}_2) \leftarrow \tilde{\mathsf{pk}}$
3 : if $H(\rho) = \tilde{\mathsf{pk}}_1$
4 :     $\mathsf{pk} = \tilde{\mathsf{pk}}_2 - \rho$
5 :     return $\mathsf{Verify}_{\Sigma}(\sigma, \mathsf{pk}, \mathsf{msg})$
6 : return 0
```

Fig. 2: HiddenPK(Σ, λ, H) generic transform.

We now define and prove the security of our transform in both the classical and the quantum setting.

Theorem 1 (Classical security of HiddenPK). *The HiddenPK(Σ, λ, H) transform is secure against existential forgery under chosen-message attack if the underlying signature scheme Σ is and if H is 2nd pre-image resistant.*

Proof. This is a classic proof by reduction, and we defer the full proof to Appendix B.

We now define a similar security notion for a quantum adversary and prove the security of HiddenPK. We want to formalize the fact that a quantum adversary should not be able to recover the private key (otherwise it can trivially forge signatures), and that the key pairs are being used only once for each signature. Contrarily to one-time signatures which remain secure after the signature is released, the same notion does not apply to the case where the key pair is broken by a quantum adversary. So, we rather call it a *zero-time signature*, and we consider key-only security, since each key-pair is discarded after a signature is issued.

Definition 1 (QEUF-KOA security). *A digital signature scheme* (KeyGen, Sign, Verify) *with associated security parameter λ is quantum secure against existential forgery under key-only attack if for any **quantum** polynomial time adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ in the* QEUF-KOA *game is negligible, i.e.*

$$\mathrm{Adv}_{\mathcal{A}}^{\mathsf{QEUF\text{-}KOA}} = \Pr[\mathsf{QEUF\text{-}KOA}(\mathcal{A}) \to 1] \leq \mathsf{negl}(\lambda)$$

where the QEUF-KOA *game is defined in Figure 3.*

```
QEUF-KOA($\mathcal{A}$)
1 : KeyGen $\to$ (sk, pk)
2 : $\mathcal{A}$(pk) $\to (m^*, \sigma^*)$
3 : return Verify($\sigma^*$, pk, $m^*$)
```

Fig. 3: The QEUF-KOA game for zero-time quantum security.

Theorem 2 (Quantum security of HiddenPK). *Consider a digital signature scheme Σ with security parameter λ, if the hash function H is quantum one-way over $\mathcal{D}$ and the statistical distance between the distribution of* pk *in the public key domain $(\mathcal{D}, +)$ and the uniform distribution in $\mathcal{D}$ is negligible, then* HiddenPK(Σ, λ, H) *is zero-time quantum unforgeable (QEUF-KOA).*

Note that this result requires no security assumption related to the underlying signature scheme Σ. It is true even though the underlying scheme is broken by quantum (or classical) adversaries. It is still true for a trivial scheme Σ for which Verify_{Σ} always returns true. The only security assumption is about the quantum one-wayness of H. The proof consists in saying that breaking HiddenPK implies breaking H.

Proof. In the QEUF-KOA game, the signing key sk is unused. We define $D(\mathsf{pk})$ by the execution of the game after the first step. $D(\mathsf{pk})$ can be seen as a distinguisher between the distribution of pk and the uniform distribution. The advantage is negligible, by assumption on the statistical distance. We can now define $\mathcal{B}$ as on Figure 4 playing the QOW game. Since ρ and pk are independent, $(H(\rho), \mathsf{pk} + \rho)$ and $(H(\rho), x)$ have the same distribution when pk is uniform. Hence, QOW returns 1 with the same probability as D. □

Note that the HiddenPK transform is generic, and can be further optimized to fit specific signature schemes to obtain shorter signatures. We give an example of HiddenPK tailored to ECDSA in Appendix E which does not store the masking value in the signature, by making use of the fact that we can recover public keys from the signature directly.

```
QOW($\mathcal{B}$)
1 : $\rho \leftarrow\$ \mathcal{D}$
2 : $y = H(\rho)$
3 : $\mathcal{B}(y) \rightarrow \rho^*$
4 : return $\mathbb{1}_{H(\rho)=H(\rho^*)}$
```

```
$\mathcal{B}(y)$
1 : $x \leftarrow\$ \mathcal{D}$
2 : $\tilde{\mathsf{pk}} \leftarrow (y, x)$
3 : $\mathcal{A}(\tilde{\mathsf{pk}}) \rightarrow (m^*, \tilde{\sigma}^*)$
4 : $(\sigma^*, \rho^*) \leftarrow \tilde{\sigma}^*$
5 : return $\rho^*$
```

Fig. 4: QOW game for adversary $\mathcal{B}$ (Theorem 2).

2.2 Bitcoin P2PKH

The HiddenPK transform differs from the approach taken in Bitcoin P2PKH of hiding the public key behind a hash. We give a generalized version of P2PKH as the HashedPK transform in Figure 5, applicable to any signature scheme Σ, and analyze its security.
Note that this transform requires a stronger assumption than our HiddenPK transform, namely collision resistance.

```
HashedPK-KeyGen($\lambda$)
1 : $(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{KeyGen}_\Sigma(\lambda)$
2 : $\mathsf{pk}_\mathcal{H} \leftarrow H(\mathsf{pk})$
3 : $\tilde{\mathsf{sk}} \leftarrow (\mathsf{sk}, \mathsf{pk})$
4 : return $(\tilde{\mathsf{sk}}, \mathsf{pk}_\mathcal{H})$
```

```
HashedPK-Sign($\tilde{\mathsf{sk}}$, msg)
1 : $(\mathsf{sk}, \mathsf{pk}) \leftarrow \tilde{\mathsf{sk}}$
2 : $\sigma \leftarrow \mathsf{Sign}_\Sigma(\mathsf{sk}, \mathsf{msg})$
3 : $\tilde{\sigma} \leftarrow (\sigma, \mathsf{pk})$
4 : return $\tilde{\sigma}$
```

```
HashedPK-Verify($\tilde{\sigma}$, $\mathsf{pk}_\mathcal{H}$, msg)
1 : $(\sigma, \mathsf{pk}) \leftarrow \tilde{\sigma}$
2 : if $\mathsf{pk}_\mathcal{H} = H(\mathsf{pk})$ :
        return $\mathsf{Verify}_\Sigma(\sigma, \mathsf{pk}, \mathsf{msg})$
3 : return 0
```

Fig. 5: HashedPK(Σ, λ, H) a generic transform from the Bitcoin P2PKH construction.

However, EUF-CMA security can be proven for the HashedPK construction with the additional assumption of collision resistance on H (hence a larger hash length and the need for a hash key). QEUF-KOA security can be proven as in Theorem 2.

Theorem 3 (Classical security of HashedPK). *If Σ is EUF-CMA and if H is collision resistant (with a common random* hk*) then* HashedPK(Σ, λ, H) *is EUF-CMA.*

Proof. The proof follows similarly to that of Theorem 1, by considering an adversary against collision resistance, instead of second pre-image resistance.

Theorem 4 (Quantum security of HashedPK). *Consider a digital signature scheme Σ with associated parameters λ, if the hash function H is quantum one-way over $\mathcal{D}$ and the statistical distance between the distribution of* pk *in the public key domain $(\mathcal{D}, +)$ and the uniform distribution in $\mathcal{D}$ is negligible,* HashedPK(Σ, pk, H) *is* QEUF-KOA.

Proof. The proof is similar to that of Theorem 2. □

3 Delayed Signatures

One issue that needs to be addressed is front-running. In our case, if at the end of the protocol a rushing quantum adversary has access to the signature before it is published, the adversary can recover the public key and derive the secret key to produce arbitrarily many new signatures before the original one is registered. We therefore need a mechanism to prevent this behaviour.

We assume that signatures are published in an append-only ledger where everyone can reliably check the contents of the ledger consistently and everyone can append to the ledger with some publication delay.

We propose the idea of a two-stage transaction in which the first stage would consist of announcing a commitment for the signature to be verified, and the second one would release the actual signature for verification. This two stage commit and reveal approach was introduced by Fawkescoin [4]. The high level idea is to commit to the signature $\tilde{\sigma}$ using a commitment scheme with opening value open that returns a commitment value com. Namely, we write $\mathsf{Commit}(\tilde{\sigma}, \mathsf{open}) \rightarrow \mathsf{com}$.

Formally, we consider a signature that is composed of two components, namely σ_1, σ_2. The first component (σ_1) is appended to the ledger as a separate transaction to commit to the signature. The submission of the second transaction follows after the first component being issued to the ledger. We now define the following delayed signature scheme and the corresponding security notion. Note that we switch the naming to *one-time* signatures because only the signatures that are committed are considered valid and hence, a signature released without a valid commitment does not represent a forgery.

Definition 2 (One-Time Delayed Signature). *A One-Time Delayed Signature consists of three algorithms*
$(\mathsf{Delay\text{-}KeyGen}, \mathsf{Delay\text{-}Sign}, \mathsf{Delay\text{-}Verify})$ *such that:*

– $(\mathsf{sk}', \mathsf{pk}') \leftarrow \mathsf{Delay\text{-}KeyGen}(\lambda)$*: A* PPT *algorithm that takes as input a security parameter λ and outputs a secret key* sk' *and the corresponding public key* pk'*.*

– $(\sigma_1, \sigma_2) \leftarrow \mathsf{Delay\text{-}Sign}(\mathsf{sk}', \mathsf{msg})$*: A* PPT *algorithm that takes as input a secret key* sk' *and the message to be signed* msg*. Outputs a two component signature (σ_1, σ_2) where σ_1 acts as a commitment for the σ_2 to be verified after the delay.*

– $0/1 \leftarrow \mathsf{Delay\text{-}Verify}(\mathsf{pk}', (\sigma_1, \sigma_2), \mathsf{msg})$*: A deterministic polynomial time algorithm that takes as input a public key* pk'*, a two component signature (σ_1, σ_2) and the message* msg *to be verified. Outputs* 1 *if (σ_1, σ_2) is a valid signature tuple for* msg*,* 0 *otherwise.*

Definition 3 (One-Time Delayed Signature Unforgeability). *A One-Time Delay signature scheme is* unforgeable *if for any quantum polynomial time adversary $\mathcal{A}$ the advantage of winning the* OTDU *game (Figure 6) is negligible in λ, that is:*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{OTDU}} = \Pr[\mathsf{OTDU}(\mathcal{A}) \rightarrow 1] \leq \mathsf{negl}(\lambda)$$

```
OTDU(𝒜)
1 : $(\mathsf{sk}', \mathsf{pk}') \leftarrow\$ \mathsf{Delay\text{-}KeyGen}(\lambda)$
2 : $\mathsf{msg} \leftarrow \perp$
3 : $\mathsf{queried} \leftarrow \mathsf{false}$
4 : $\mathcal{A}^{\mathsf{OSig}}(\mathsf{pk}') \rightarrow st, L$
5 : $\mathcal{A}(st, \sigma_2) \rightarrow s_1, s_2, m$
6 : if $\mathsf{Delay\text{-}Verify}(\mathsf{pk}', s_1, s_2, m) \neq 1$
7 :     return $0$
8 : return $\mathbb{1}_{s_1 \in L \wedge m \neq \mathsf{msg}}$
```

```
OSig(m)
1 : if queried
2 :     return $\perp$
3 : $\mathsf{queried} \leftarrow \mathsf{true}$
4 : $\mathsf{msg} \leftarrow m$
5 : $(\sigma_1, \sigma_2) \leftarrow \mathsf{Delay\text{-}Sign}(\mathsf{sk}', m)$
6 : return $\sigma_1$
```

Fig. 6: OTDU security game.

Intuitively, OTDU game consists of supplying the adversary with the public key pk', then the adversary outputs a list of σ_1 candidates in the list L. During this process, it is possible to query the OSig oracle once, to obtain a valid σ_1 for msg. This behaviour corresponds to the adversary observing a signer posting pk', σ_1 to the ledger. In the second execution of the adversary, it is given σ_2 and it outputs a two component signature s_1, s_2 and the forged message m. This corresponds to the adversary seeing the release of the actual signature within σ_2. The adversary wins if the forged signature verifies successfully, the first component s_1 has been generated in the first execution and the forged message m is not equal to msg.

3.1 Constructing a One-Time Delayed Signature from a HiddenPK Signature

We construct a *One-Time Delayed Signature* from a HiddenPK signature by committing to the signature (σ_1) and releasing the second part of the signature along with the opening (σ_2). The signature is only correctly verified after the signer releasing σ_2 and its opening along with the signature. More concretely, given a HiddenPK signature scheme Σ_{HPK}, a QOW hash function H and a commitment scheme Commit, we construct a delayed signature in Figure 7.

Theorem 5. (QEUF-KOA $\rightarrow$ OTDU) *For a given* QEUF-KOA *secure* HiddenPK *signature scheme Σ_{HPK}, and a random oracle* Commit *with output length λ, the resulting delayed signature scheme* DelayHiddenPK *(Figure 7) is unforgeable.*

The proof follows from the fact that the OSig oracle can be simulated by returning a random value as a random oracle would do. Moreover, it is possible to extract the forged signature ($\tilde{\sigma}$) value by intercepting the random oracle queries that $\mathcal{A}$ submits and obtain the output m from the output tuple (s_1, s_2, m). As a result, we have an adversary that wins QEUF-KOA game by returning $(\tilde{\sigma}, m)$ given $\tilde{\mathsf{pk}}$. We refer to Appendix B for the full proof.

However, *this model is not strong enough to be practically used.* When using the OTDU notion, we can still face two practical problems: a *Denial of Service*

```
Delay-KeyGen($\lambda$)
    $(\tilde{\mathsf{sk}}, \tilde{\mathsf{pk}}) \leftarrow\!\$\ \mathsf{KeyGen}_{\Sigma_{\mathsf{HPK}}}(\lambda)$
    return $(\tilde{\mathsf{sk}}, \tilde{\mathsf{pk}})$

Delay-Sign($\tilde{\mathsf{sk}}$, msg)
    $\tilde{\sigma} \leftarrow \mathsf{Sign}_{\Sigma_{\mathsf{HPK}}}(\tilde{\mathsf{sk}}, \mathsf{msg})$
    $\mathsf{open} \leftarrow\!\$\ \{0,1\}^{\lambda}$
    $\mathsf{com} \leftarrow \mathsf{Commit}(\tilde{\sigma}||\mathsf{msg}, \mathsf{open})$
    $\sigma_1 \leftarrow \mathsf{com}$
    $\sigma_2 \leftarrow (\tilde{\sigma}, \mathsf{open})$
    return $(\sigma_1, \sigma_2)$

Delay-Verify($\tilde{\mathsf{pk}}$, $(\sigma_1, \sigma_2)$, msg)
    $\mathsf{com} \leftarrow \sigma_1$
    $(\tilde{\sigma}, \mathsf{open}) \leftarrow \sigma_2$
    if $\mathsf{Commit}(\tilde{\sigma}||\mathsf{msg}, \mathsf{open}) \neq \mathsf{com}$
        return $0$
    return $\mathsf{HiddenPK\text{-}Verify}(\tilde{\sigma}, \tilde{\mathsf{pk}}, \mathsf{msg})$
```

Fig. 7: Delayed signature construction from a HiddenPK signature.

(DoS) attack consisting of flooding the ledger with rogue σ_1 commits, and a rushing adversary who would break the signature after seeing (σ_1, σ_2), and rushing to post a rogue (σ_1', σ_2') before σ_2 is published. We now introduce an enriched security model that captures both of these problems and introduce a three-stage signature process. The first problem is solved by defining an artificial relation between pk' and σ_1 so that pk' values cannot be paired up with arbitrary σ_1 values. The second problem is solved by adding extra mechanisms for explicitly controlling the opening and the closing of the list L on the ledger. We formalize the new approach as follows.

Definition 4 (One-Time Signatures with Delayed Ledger). *A One-Time Signature with Delayed Ledger consists of an append-only ledger L and the following algorithms:*

- $(\mathsf{sk}', \mathsf{pk}') \leftarrow \mathsf{DelayL\text{-}KeyGen}(\lambda)$*: A* PPT *algorithm that takes as input a security parameter and outputs a secret key* sk' *and the corresponding public key* pk'*.*
- $\mathsf{tok}_1 \leftarrow \mathsf{Open}(\mathsf{sk}')$*: Takes as input the secret key* sk'*, outputs the first token associated with the opening of the ledger L.*
- $\mathsf{tok}_2 \leftarrow \mathsf{Close}(\mathsf{sk}')$*: Takes as input the secret key* sk'*, outputs the second token associated with the closing of the ledger L.*
- $0/1 \leftarrow \mathsf{VerifyOpen}(\mathsf{pk}', \mathsf{tok}_1)$*: Takes as input the public key* pk' *and an opening token* tok_1*, outputs 1 if the opening value is consistent with the public key.*
- $0/1 \leftarrow \mathsf{VerifyClose}(\mathsf{pk}', \mathsf{tok}_2)$*: Takes as input the public key* pk' *and a closing token* tok_2*, outputs 1 if the opening value is consistent with the public key.*
- $(\sigma_1, \sigma_2) \leftarrow \mathsf{DelayL\text{-}Sign}(\mathsf{sk}', \mathsf{msg})$*: A* PPT *algorithm that takes as input a secret key* sk' *and the message to be signed* msg*. Outputs a two component signature (σ_1, σ_2) where σ_1 acts as a commitment for the σ_2 to be verified after a delay.*
- $0/1 \leftarrow \mathsf{DelayL\text{-}Verify}(\mathsf{pk}', (\sigma_1, \sigma_2), \mathsf{msg})$*: A deterministic polynomial time algorithm that takes as input a public key* pk'*, a two component signature*

(σ_1, σ_2) *and the message* msg *to be verified. Outputs* 1 *if* (σ_1, σ_2) *is a valid signature tuple for* msg*,* 0 *otherwise.*

Definition 5 (One-Time Signature Unforgeability with Delayed Ledger). *A One-Time Signature with Delayed Ledger scheme is unforgeable if for any quantum polynomial time adversary* $\mathcal{A}$ *the advantage of winning the* $\mathsf{OTDU\text{-}L}$ *game (Figure 8) is negligible in* λ*, that is:*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{OTDU\text{-}L}} = \Pr[\mathsf{OTDU\text{-}L}(\mathcal{A}) \to 1] \leq \mathsf{negl}(\lambda)$$

```
OTDU-L($\mathcal{A}$)
  $(\mathsf{sk}', \mathsf{pk}') \leftarrow\$ \mathsf{Delay\text{-}KeyGen}(\lambda)$
  $\mathsf{opened}, \mathsf{openwish} \leftarrow \mathsf{false}, \mathsf{false}$
  $\mathsf{closed}, \mathsf{closewish} \leftarrow \mathsf{false}, \mathsf{false}$
  $\mathsf{queried} \leftarrow \mathsf{false}$
  $L \leftarrow$ empty list
  $\mathsf{msg} \leftarrow \bot$
  $\mathcal{A}^{\mathsf{O}*}(\mathsf{pk}') \to s_1, s_2, m$
  reward if $(s_1 \in L) \wedge \mathsf{Delay\text{-}Verify}(\mathsf{pk}', s_1, s_2, m) \wedge m \neq \mathsf{msg}$

OSig($m$)
  if $\mathsf{queried} \vee \neg\mathsf{openwish} \vee \mathsf{closewish}$
    return $\bot$
  $\mathsf{queried} \leftarrow \mathsf{true}$
  $\mathsf{msg} \leftarrow m$
  $(\sigma_1, \sigma_2) \leftarrow \mathsf{Delay\text{-}Sign}(\mathsf{sk}', m)$
  return $\sigma_1$

Opost($s$)
  if $\mathsf{opened} \wedge \neg\mathsf{closed}$
    $L \leftarrow L \| s$
  return $\bot$

Oopen($t$)
  if $\mathsf{VerifyTokOpen}(\mathsf{pk}', t)$
    $\mathsf{opened} \leftarrow \mathsf{true}$
    reward if $\neg\mathsf{openwish}$
  return $\bot$

OwishOpen()
  $\mathsf{openwish} \leftarrow \mathsf{true}$
  return $\mathsf{Open}(\mathsf{sk}')$

Oclose($t$)
  if $\neg\mathsf{VerifyTokClose}(\mathsf{pk}', t)$
    return $\bot$
  $\mathsf{closed} \leftarrow \mathsf{true}$
  reward if $\neg\mathsf{closewish}$
  // returned after closure
  return $\sigma_2$

OwishClose()
  if $\neg\mathsf{queried} \vee \sigma_1 \notin L$
    return $\bot$
  $\mathsf{closewish} \leftarrow \mathsf{true}$
  return $\mathsf{Close}(\mathsf{sk}')$
```

Fig. 8: $\mathsf{OTDU\text{-}L}$ security game for the signature scheme with delayed ledger.

Intuitively, when a signer wants to sign a message, it opens the ledger L, it posts the signature commitment σ_1. It checks that σ_1 is well posted. It closes the ledger L and only releases σ_2 after checking that the ledger L is closed. Using the $\mathsf{OTDU\text{-}L}$ notion: The adversary can post arbitrary commitments to L as long as it is open. The adversary has to call $\mathsf{OwishOpen}$ in order to call Oopen. That is, it has to observe the correct opening token. Moreover, the adversary has to call $\mathsf{OwishClose}$ in order to call Oclose. That is, the commitment of the signer should appear in the list L and σ_2 can only be observed after the closure of L.

3.2 Constructing a One-Time Signature with Delayed Ledger

We can modify the delayed signature construction in Figure 7 by integrating two one-time tokens t_1 (for opening) and t_2 (for closing) to the secret key and their corresponding hashes to the public key. More concretely, we realize the construction as in Figure 9.

```
DelayL-KeyGen($\lambda$)
    $(\tilde{\mathsf{sk}}, \tilde{\mathsf{pk}}) \leftarrow\!\$\ \mathsf{Delay\text{-}KeyGen}(\lambda)$
    $t_1 \leftarrow\!\$\ \{0,1\}^\lambda, t_2 \leftarrow\!\$\ \{0,1\}^\lambda$
    $\mathsf{sk}' \leftarrow (\tilde{\mathsf{sk}}, t_1, t_2)$
    $\mathsf{pk}' \leftarrow (\tilde{\mathsf{pk}}, H(t_1), H(t_2))$
    return $(\mathsf{sk}', \mathsf{pk}')$

DelayL-Sign($\mathsf{sk}'$, msg)
    $(\tilde{\mathsf{sk}}, t_1, t_2) \leftarrow sk'$
    $(\sigma_1, \sigma_2) \leftarrow \mathsf{Delay\text{-}Sign}(\tilde{\mathsf{sk}}, \mathsf{msg})$
    return $(\sigma_1, \sigma_2)$

Open($\mathsf{sk}'$)
    $(\tilde{\mathsf{sk}}, t_1, t_2) \leftarrow sk'$
    return $t_1$

DelayL-Verify($\mathsf{pk}', (\sigma_1, \sigma_2), \mathsf{msg}$)
    $(\tilde{\mathsf{pk}}, H(t_1), H(t_2)) \leftarrow \mathsf{pk}'$
    return $\mathsf{Delay\text{-}Verify}(\mathsf{pk}', (\sigma_1, \sigma_2), \mathsf{msg})$

VerifyOpen($\mathsf{pk}', t$)
    $(\tilde{\mathsf{pk}}, H(t_1), H(t_2)) \leftarrow \mathsf{pk}'$
    return $\mathbb{1}_{H(t)=H(t_1)}$

VerifyClose($\mathsf{pk}', t$)
    $(\tilde{\mathsf{pk}}, H(t_1), H(t_2)) \leftarrow \mathsf{pk}'$
    return $\mathbb{1}_{H(t)=H(t_2)}$

Close($\mathsf{sk}'$)
    $(\tilde{\mathsf{sk}}, t_1, t_2) \leftarrow sk'$
    return $t_2$
```

Fig. 9: Delayed signature (with ledger) construction

Theorem 6. (OTDU $\rightarrow$ OTDU-L) *For a given* OTDU *signature scheme* (Delay-KeyGen, Delay-Sign, Delay-Verify)*, the resulting signature scheme with delayed ledger* OTDU-L *(Figure 9) is unforgeable.*

Proof. We need to show that

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{OTDU\text{-}L}} = \Pr[\mathsf{OTDU\text{-}L}(\mathcal{A}) \rightarrow 1] \leq \mathsf{negl}(\lambda)$$

We set G0 to be the OTDU-L game. Given an adversary $\mathcal{A}$ that wins G0, we reduce G0 to OTDU as follows:

– G1 : We replace Oopen by an oracle who does not reward if openwish is false. The difference between OTDU-L and G1 can express as an adversary who break QOW on $H(t_1)$, that is:

$$|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{G1}} - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{G0}}| \leq \mathsf{negl}(\lambda)$$

– G2 : We replace Oclose by an oracle which first checks if closewish is true and returns nothing if closewish is false. The difference between G1 and G2 can express as an adversary who breaks QOW on $H(t_2)$. Note that Oclose does not have a reward statement anymore.

$$|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{G2}} - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{G1}}| \leq \mathsf{negl}(\lambda)$$

The OTDU adversary $\mathcal{A}'$ simulates $\mathcal{A}$ playing $\mathsf{G2}$ and the oracles that are not present in the OTDU game. That is, it selects (t_1, t_2), simulates Oopen, $\mathsf{OwishOpen}, \mathsf{Oclose}, \mathsf{OwishClose}$ by using the sampled (t_1, t_2) values. It also simulates Opost by maintaining a list L. Furthermore, $\mathcal{A}'$ stops when it makes the Oclose query which closes L to return the list L and a state *st* to resume the simulation of $\mathcal{A}$ (Line 3 of the OTDU in Figure 6). When it resumes, σ_2 is obtained and provided to $\mathcal{A}$ for the rest of the simulation. Note that no calls to the oracle OSig has been made throughout the simulation. Hence, given an adversary $\mathcal{A}$ that wins the $\mathsf{OTDU\text{-}L}$ game, we can construct an adversary $\mathcal{A}'$ that wins OTDU. Since $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{ODTU}}$ is negligible in λ, we have:

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{OTDU\text{-}L}} \leq \mathsf{negl}(\lambda)$$

□

Note that it is possible to extend the *One-Time Signature with Delayed Ledger* construction and the corresponding $\mathsf{OTDU\text{-}L}$ notion to *n-time* signatures in a natural way. As long as the underlying signature scheme has $\mathsf{QEUF\text{-}KOA}$ security, it is possible to publish n signature commitments to the ledger under the same public key and wait for the ledger closure before releasing the signatures.

Lifting the Proofs to QROM: Since the challenger only makes a constant number of queries to the random oracle in OTDU and $\mathsf{OTDU\text{-}L}$ both Theroem 5 and Theorem 6 still holds in the QROM model. This is due to the *Lifting Theorem for Search-Type Games* (Theorem 4.2 from [17]). Note that the advantage of the adversary against ROM is divided by $(2q+1)^{2k}$ where q is the number of queries the adversary makes (polynomially bounded in our case) and k is the number of queries the challenger makes to the oracle ($k = 2$ in our case).

Discussion on Practical Instantiations: The construction based on the $\mathsf{OTDU\text{-}L}$ notion cannot be directly implemented in Bitcoin (we present the necessary Bitcoin Scripts in Appendix C). Mainly due to the fact that after a single *close* transaction (to close the original *open* transaction), no further *open* transactions can be made. However, this behaviour is not currently possible unless the miners manually check for this condition, which would result in a hard fork. Nevertheless, it is possible to practically instantiate this for any ledger that implements this closure mechanism.

Note that our approach of two-stage signatures targets the issue of avoiding publication of wrong stage-1 transactions for a given address. However this does not address the scenario in which a user floods the ledger with correct stage-1 commitments without ever paying the transaction during stage-2, but including stage-1 transactions does consume network resources. Moreover, there is no incentive for miners to even include the stage-1 transactions with commitments. This problem is well-known and has been studied since [4]. We further reflect on this problem by considering the following scenarios:

– **Honest stage-1 transactions with malicious opens**: In this scenario, the goal of the rushing adversary is to add many open transactions to increase the total transaction fee paid by the original signer of the transaction.
  In stage-1, the token t_1 is released and rushing adversaries can create rogue transactions with the same token t_1. This can only be done until the second token t_2 is released (i.e. stage-2). We stress that t_2 is released after t_1 is well posted (i.e. the block confirmation period has passed for the block containing t_1). For instance, Bitcoin requires 6 block confirmations which -assuming the block producing time is 10 minutes- would translate to an hour.
  If we assume that each miner includes only one transaction per block per t_1, this would multiply the transaction fee by the number of blocks in the block confirmation period.
– **Dishonest stage-1 transactions**: In this scenario, the goal of the adversary (not necessarily rushing) is to create as many stage-1 transactions as possible and never proceed with the stage-2 transaction to cause congestion over the network. In this setting, the miners would not get paid for the transactions that have been already included if stage-2 never occurs. The malicious signer would further lose assets.

Looking at these two scenarios, we see a relation between the denial of service surface of the system and the miner's motivation for including stage-1 transactions. If we motivate the miners to include stage-1 transactions, there is a better opportunity for the adversary to congest the network by dishonest stage-1 transactions, but also the opportunity for the miner to extract more fees from honest stage-1 transactions.

4 Extension to Threshold Signatures

In this section, we extend the construction of Section 2 to the case of multi-party signing, and more precisely to threshold signatures. In particular, this means that we present a threshold signature scheme with hidden public key.

Consider a threshold signature scheme Σ with associated parameters λ, we assume that $\mathsf{TSKeyGen}$ has a specific form i.e. it consists of an algorithm $((\mathsf{sk}_i, \mathsf{pk}_i)_{i\in[n]}) \leftarrow \mathsf{TSKeyGen}^1(\lambda)$, which is followed by a final computation $\mathsf{pk} \leftarrow \mathsf{Rec}((\mathsf{pk}_i)_{i\in S})$, where Rec is the linear reconstruction function from the associated linear secret sharing scheme in $\mathcal{D}$. We argue that this assumption is reasonable as it matches all the existing protocols. In fact, we give an example of a concrete application in Section E.1. Hence, we abuse it by calling the transform as a generic one.

Additionally, for simplicity we assume a *trusted setup* during $\mathsf{TSKeyGen}$ in the form of an honest dealer running $\mathsf{TSKeyGen}$. Appendix F discusses multiparty protocols for secure setup.
Given such a threshold signature scheme Σ with associated parameters λ, a (t, n) secret sharing scheme $(\mathsf{Shr}_t^n, \mathsf{Rec})$, and a hash function H, the generic threshold with hidden public key transform $\mathsf{ThreshHiddenPK}$ is describe in Figure 10.

Note that this transform makes the public key and the signature grow linear in size and makes the resulting signature transparent (i.e. it is possible to distinguish whether a signature is the result of a threshold signing process). However, the amount of computational overhead is minimal since the key generation is almost non-interactive. It is possible to construct a non-transparent transformation with constant key size by considering a more computationally demanding key generation process. This construction can be found in Appendix G.

```
ThreshHiddenPK-KeyGen($\lambda$)
  $(\mathsf{sk}_i, \mathsf{pk}_i)_{i\in[n]} \leftarrow\$ \mathsf{TSKeyGen}^1_{\Sigma}(\lambda)$
  for $i = 1, ..., n$ :
      $\rho_i \leftarrow\$ \mathcal{D}$
      $\tilde{\mathsf{sk}}_i \leftarrow (\mathsf{sk}_i, \rho_i)$
  $\tilde{\mathsf{pk}}_1 \leftarrow (H(\rho_i))_{i\in[n]}$
  $\tilde{\mathsf{pk}}_2 = (\mathsf{pk}_i + \rho_i)_{i\in[n]}$
  $\tilde{\mathsf{pk}} \leftarrow (\tilde{\mathsf{pk}}_1, \tilde{\mathsf{pk}}_2)$
  return $((\tilde{\mathsf{sk}}_i)_{i\in[n]}, \tilde{\mathsf{pk}})$

ThreshHiddenPK-Sign($(\tilde{\mathsf{sk}}_i)_{i\in S}, \mathsf{msg}$)
  Each $P_i$ does :
  $(\mathsf{sk}_i, \rho_i) \leftarrow (\tilde{\mathsf{sk}}_i)$
  $\sigma \leftarrow \mathsf{TSSign}_{\Sigma}((\mathsf{sk}_i)_{\in S}, \mathsf{msg})$
  $\tilde{\sigma} \leftarrow (\sigma, S, (\rho_i)_{i\in S})$
  return $\tilde{\sigma}$

ThreshHiddenPK-Verify($\tilde{\sigma}, \tilde{\mathsf{pk}}, \mathsf{msg}$)
  $(\sigma, S, (\rho_i)_{i\in S}) \leftarrow \tilde{\sigma}$
  $(\tilde{\mathsf{pk}}_1, \tilde{\mathsf{pk}}_2) \leftarrow \tilde{\mathsf{pk}}$
  for each $i \in S$ :
      if $H(\rho_i) \neq (\tilde{\mathsf{pk}}_1)_i$ :
          return 0
  for each $i \in S$ :
      $\mathsf{pk}_i \leftarrow (\tilde{\mathsf{pk}}_2)_i - \rho_i$
  $\mathsf{pk} \leftarrow \mathsf{Rec}((\mathsf{pk}_i)_{i\in S})$
  return $\mathsf{Verify}_{\Sigma}(\sigma, \mathsf{pk}, \mathsf{msg})$
```

Fig. 10: ThreshHiddenPK(Σ, λ, H) transform.

Theorem 7 (Classical security of ThreshHiddenPK). *The* ThreshHiddenPK(Σ, λ, H) *transform is existentially unforgeable under chosen-message attack (TEUF-CMA) if the underlying threshold signature scheme Σ is and if H is 2nd pre-image resistant.*

Proof. The proof technique can be extended from the single signer setting in a straightforward manner. We defer the full proof to Appendix D. □

We now define and analyze the quantum security of our protocol. We extend our definitions from the single signer setting to the threshold one, and define zero-time threshold signatures as threshold signatures that are quantum secure against existential forgeries under key-only attack.

Definition 6 (TQEUF-KOA security). *A (t, n)-threshold-signature scheme consisting of three algorithms* (TSKeyGen, TSSign, Verify) *is quantum existentially unforgeable under key-only attack (TQEUF-KOA) if for any quantum adversary $\mathcal{A}$, and for any set of corrupted players* Corr *chosen by $\mathcal{A}$ such that* $|\mathsf{Corr}| \leq t$, *the advantage of $\mathcal{A}$ in the* TQEUF-KOA *game is negligible i.e.*

$$\mathrm{Adv}^{\mathsf{TQEUF\text{-}KOA}}_{\mathcal{A}} = \Pr[\mathsf{TQEUF\text{-}KOA}(\mathcal{A}) \rightarrow 1] \leq \mathsf{negl}(\lambda)$$

where the TQEUF-KOA *game is defined in Figure 11.*

```
TQEUF-KOA($\mathcal{A}$, Corr)
1 : TSKeyGen($\lambda$) $\to$ ((sk$_i$)$_{i\in[n]}$, pk)
2 : $\mathcal{A}$(pk, (sk$_i$)$_{i\in\mathsf{Corr}}$) $\to$ ($\sigma^*, m^*$)
3 : return Verify($\sigma^*$, pk, $m^*$) = 1
```

Fig. 11: The TQEUF-KOA game for zero-time quantum security of threshold signatures.

Theorem 8 (Quantum security of ThreshHiddenPK). *The* ThreshHiddenPK(Σ, λ, H) *transform is quantum existentially unforgeable under key-only attack if H is quantum one-way and given ρ_i, r_i for $i \in [n]$ uniform and independent then $(H(\rho_i), \mathsf{pk}_i + \rho_i)_{i\in[n]}$ is indistinguishable from $(H(\rho_i), r_i)_{i\in[n]}$.*

Proof. Consider an adversary $\mathcal{A}$ for the TQEUF-KOA game applied to ThreshHiddenPK(Σ, λ, H). We construct an adversary $\mathcal{B}$ such that if $\mathcal{A}$ wins TQEUF-KOA then $\mathcal{B}$ breaks the one-wayness of $\mathcal{A}$. Given a quantum one-way challenge y, $\mathcal{B}$ generates key pairs for ThreshHiddenPK and samples a special index j out of the uncorrupted values, for which the hash is set to y. Then they call $\mathcal{A}$ on the public key and the corrupted secrets. Then $\mathcal{B}$ can extract the preimage ρ_j^* of y from the message-signature pair returned by $\mathcal{A}$. If $\mathcal{A}$ wins the TQEUF-KOA game then ρ_j^* will be a valid pre-image for the QOW game. Using the assumption on the distribution of pk_i and H, the view of $\mathcal{A}$ called as a subroutine of $\mathcal{B}$ is indistinguishable from the view of $\mathcal{A}$ in the TQEUF-KOA game. This implies that if $\mathcal{A}$ wins TQEUF-KOA then $\mathcal{B}$ wins QOW. Hence we have

$$\Pr[\mathsf{TQEUF\text{-}KOA}(\mathcal{A}) \to 1] \le \Pr[\mathsf{QOW} \to 1] = \mathsf{negl}(\lambda)$$

by assumption on the quantum one-wayness of H. □

Note that the assumption on the pk_i and H can be proven assuming that the vector of pk_is has high min-entropy (denoted H_∞) and that given $K = (k_1, ...k_n)$ and $X = (x_1, ..., x_n)$, the function $h_K(X) = (H(k_i - x_i))_{i\in[n]}$ is a universal hash function.
Let ℓ be the output length of H. We write $\mathsf{PK} = (\mathsf{pk}_i)_{i\in[n]}$, $\mathsf{PK}{+}\rho = (\mathsf{pk}_i{+}\rho_i)_{i\in[n]}$ as random vectors and U for a uniform random vector. We consider the function $h_K(\mathsf{PK}) = (H(k_i - \mathsf{pk}_i))_{i\in[n]}$ with $k_i = \rho_i + \mathsf{pk}_i$ of output length $n \cdot \ell$. We have that PK and $\mathsf{PK} + \rho$ are independent, and are independent from uniform U. Furthermore, $\mathsf{PK} + \rho$ is uniformly distributed since ρ_i is uniform for all $i \in [n]$. Since we assume that the min-entropy of PK is high, and h is a universal hash function, we can apply the Leftover Hash Lemma [12] and we get that the distribution of $(h_{\mathsf{PK}+\rho}(\mathsf{PK}), \mathsf{PK} + \rho)$ and that of $(U, \mathsf{PK} + \rho)$ are $\epsilon/2$-indistinguishable, with $\epsilon = \sqrt{2^{n\cdot\ell - \mathsf{H}_\infty(\mathsf{pk})}}$. Since $h_{\mathsf{PK}+\rho}(\mathsf{PK}) = (H(\rho_i))_{i\in[n]}$, we get the desired result.
Note that typically, $\mathsf{H}_\infty(\mathsf{pk}_i)$ is close to $t \cdot \log_2(|G|)$ for a group G as the pk_i are t-wise independent and close to uniform in G. This implies that $\ell \le \frac{t}{n} \cdot \log_2(|G|)$.

Acknowledgements

We thank Loïs Huguenin-Dumittan for discussions about lifting the proofs of Theorem 5 and Theorem 6 to the QROM setting.

References

1. Aumasson, J.P., Hamelink, A., Shlomovits, O.: A Survey of ECDSA Threshold Signing. Cryptology ePrint Archive, Report 2020/1390 (2020), https://ia.cr/2020/1390
2. Bernstein, D.J., Hülsing, A., Kölbl, S., Niederhagen, R., Rijneveld, J., Schwabe, P.: The SPHINCS+ signature framework. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. p. 2129–2146. CCS '19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3319535.3363229, https://doi.org/10.1145/3319535.3363229
3. Beullens, W.: Breaking Rainbow Takes a Weekend on a Laptop. In: Dodis, Y., Shrimpton, T. (eds.) Advances in Cryptology – CRYPTO 2022. pp. 464–479. Springer Nature Switzerland, Cham (2022)
4. Bonneau, J., Miller, A.: Fawkescoin a cryptocurrency without public-key cryptography, pp. 350–358. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Springer Verlag (2014). https://doi.org/10.1007/978-3-319-12400-1_35, publisher Copyright: © Springer International Publishing Switzerland 2014.
5. Chalkias, K., Brown, J., Hearn, M., Lillehagen, T., Nitto, I., Schroeter, T.: Blockchained Post-Quantum Signatures. In: 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData). pp. 1196–1203 (2018). https://doi.org/10.1109/Cybermatics_2018.2018.00213
6. Cozzo, D., Smart, N.P.: Sharing the LUOV: Threshold Post-quantum Signatures. In: Albrecht, M. (ed.) Cryptography and Coding. pp. 128–153. Springer International Publishing, Cham (2019)
7. Developper, B.: Developper guides - transactions. Bitcoin Developper, https://developer.bitcoin.org/devguide/transactions.html
8. Ding, J., Deaton, J., Vishakha, Yang, B.Y.: The Nested Subset Differential Attack. In: Canteaut, A., Standaert, F.X. (eds.) Advances in Cryptology – EUROCRYPT 2021. pp. 329–347. Springer International Publishing, Cham (2021)
9. Evans, D., Kolesnikov, V., Rosulek, M.: A Pragmatic Introduction to Secure Multi-Party Computation. Foundations and Trends in Privacy and Security **2**(2-3), 70–246 (2018). https://doi.org/10.1561/3300000019
10. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Robust Threshold DSS Signatures. In: Maurer, U. (ed.) Advances in Cryptology — EUROCRYPT '96. pp. 354–371. Springer Berlin Heidelberg, Berlin, Heidelberg (1996)
11. Goldreich, O., Micali, S., Wigderson, A.: A Completeness Theorem for Protocols with Honest Majority. Conference Proceedings of the Annual ACM Symposium on Theory of Computing (01 1987)
12. Impagliazzo, R., Levin, L.A., Luby, M.: Pseudo-Random Generation from One-Way Functions. In: Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing. p. 12–24. STOC '89, Association for Computing Machinery, New York, NY, USA (1989). https://doi.org/10.1145/73007.73009

13. van der Linde, W.: Post-quantum blockchain using one-time signature chains (2018)
14. Lindell, Y.: Secure Multiparty Computation. Commun. ACM **64**(1), 86–96 (dec 2020). `https://doi.org/10.1145/3387108`
15. NIST: Request for Additional Digital Signature Schemes for the Post-Quantum Cryptography Standardization Process. `https://www.nist.gov/news-events/news/2022/09/request-additional-digital-signature-schemes-post-quantum-cryptography`
16. Pettit, M.: Efficient Threshold-Optimal ECDSA. In: Conti, M., Stevens, M., Krenn, S. (eds.) Cryptology and Network Security. pp. 116–135. Springer International Publishing, Cham (2021)
17. Yamakawa, T., Zhandry, M.: Classical vs Quantum Random Oracles. In: Canteaut, A., Standaert, F.X. (eds.) Advances in Cryptology – EUROCRYPT 2021. pp. 568–597. Springer International Publishing, Cham (2021)

Supplementary Material

A Preliminaries

In this section, we provide the necessary definitions for the primitives that we use throughout this work.

Definition 7 (Synchronous Multiparty Protocol). *A synchronous multiparty protocol P is defined by algorithms P_i for $i = 1, ..., n$. We write $((\mathsf{out}_i)_{i\in I}, \mathsf{out}) \leftarrow P((\mathsf{in}_i)_{i\in I}, \mathsf{in})$ to mean that we run the following steps:*

1. *Set $m_{r,ext} = \mathsf{in}$, $st_{r,i} = \mathsf{in}_i$ for $r = 0$, all $i \in I$.*
   If in_i is not specified it is set to $\perp$ by default.
   Below, all undefined $m_{r,i}$ are set to $\perp$ by default.
2. *Repeat Step 3 until all P_i return a final state.*
3. *$(st_{r+1,i}, m_{r+1,i}) \leftarrow P_i(st_{r,i}, (m_{r,j})_{j=1...n,ext})$ for all $i \in I$ and $r \leftarrow r + 1$.*
4. *Set $\mathsf{out}_i = st_{r+1,i}$ and $\mathsf{out} = (m_{r+1,i})_{i\in I}$.*

The idea captured by Definition 7 is that P_i corresponds to one round of a protocol executed by one participant. Given some possible inputs by the other parties and themselves, they execute the round and return a broadcast message for all other participants. The input of the first round may be a common input in but can also include private inputs in_i for each participant. Similarly, the last round outputs a broadcast message out and private outputs out_i for each participant.

Definition 8 (Threshold Signature Scheme). *A (t, n)-threshold signature scheme with associated public parameters λ, message space $\mathcal{M}$, and public key domain $\mathcal{D}$ consists of two polynomial-time algorithms $\mathsf{TSKeyGen}$ and Verify and a synchronous multiparty protocol TSSign defined in the following way:*

- $((\mathsf{sk}_i)_{i\in[n]}, \mathsf{pk}) \leftarrow\$ \mathsf{TSKeyGen}(\lambda)$*: It takes as input the security parameter λ. It outputs a secret key share sk_i for each party P_i and a common public key pk.*
- $\sigma \leftarrow \mathsf{TSSign}((\mathsf{sk}_i)_{i\in S}, \mathsf{msg})$*: It takes as input the secret keys sk_i for $i \in S$ and the message to be signed msg. It outputs a signature σ.*
  Actually, we have
  $((\perp, ..., \perp), (\sigma, ..., \sigma)) \leftarrow \mathsf{TSSign}((\mathsf{sk}_i)_{i\in S}, \mathsf{msg})$, i.e. no private output and one common output σ, but we use the above notation for convenience and clarity.
- $0/1 \leftarrow \mathsf{Verify}(\sigma, \mathsf{pk}, \mathsf{msg})$*: A deterministic polynomial time algorithm which takes as input the public key, the message and the signature and outputs 1 if the signature is accepted and 0 otherwise.*

The scheme is correct if $\forall\ (\lambda, S, \mathsf{msg})$ such that $S \subset \{1, ..., n\}, |S| > t$, the $\mathsf{Correctness}$ *game returns 1 with probability 1.*

```
Correctness(λ, S, msg)
  1 :  $((\mathsf{sk}_i)_{i\in[n]}, \mathsf{pk}) \leftarrow\$ \mathsf{TSKeyGen}(\lambda)$
  2 :  $\sigma \leftarrow \mathsf{TSSign}((\mathsf{sk}_i)_{i\in S}, \mathsf{msg})$
  3 :  return $\mathsf{Verify}(\sigma, \mathsf{pk}, \mathsf{msg})$
```

For simplicity in this definition we assumed a *trusted setup* during TSKeyGen in the form of an honest dealer running TSKeyGen. **We will assume so in the following of this work, except in Section F where we consider multiparty protocols for secure setup.**

Before giving a formal security definition for threshold signatures, we give a reminder on the different adversarial models.([1])

1. Corruption type:
   - Static adversary: the corrupted participants are chosen before the beginning of the protocol.
   - Adaptive adversary: the adversary may corrupt participants after the protocol has started, and obtain their full view.
2. Obedience:
   - Honest-but-curious adversary: the adversary only learns information from the corrupted parties but cannot force them to deviate from the protocol.
   - Malicious: The corrupted parties may arbitrarily deviate from the protocol.

In the rest of this work, we consider *static malicious* adversaries.

Gennaro et al. define the security of threshold signatures in [10]. The following formalizes this definition, given our assumption about trusted set-up.

Definition 9 (TEUF-CMA Security). *A (t, n)-threshold signature scheme* (TSKeyGen, TSSign, Verify) *with security parameter λ is existentially unforgeable under chosen-message attack (TEUF-CMA) if for any probabilistic polynomial time* (PPT) *adversary $\mathcal{A}$, and for any set of corrupted players* Corr *chosen by $\mathcal{A}$ such that $|\mathsf{Corr}| \leq t$, the probability, taken over all the inputs and random tapes, of winning the* TEUF-CMA *game (Figure 12) is negligible in λ i.e*

$$\mathrm{Adv}^{\mathrm{TEUF\text{-}CMA}}_{\mathcal{A},\mathsf{Corr}} = \Pr[\mathrm{TEUF\text{-}CMA}(\mathcal{A}, \mathsf{Corr}) \rightarrow 1] \leq \mathsf{negl}(\lambda)$$

For non-threshold signatures, this boils down to EUF-CMA security.

Definition 10 (EUF-CMA security). *A digital signature scheme* (KeyGen, Sign, Verify) *is secure against existential forgery under chosen message attack if for all probabilistic polynomial time adversary $\mathcal{A}$ the advantage of $\mathcal{A}$ in the* EUF-CMA *game is negligible, i.e.*

$$\mathrm{Adv}^{\mathrm{EUF\text{-}CMA}}_{\mathcal{A}} = \Pr[\mathrm{EUF\text{-}CMA}(\mathcal{A}) \rightarrow 1] \leq \mathsf{negl}(\lambda)$$

where the EUF-CMA *game is defined in Figure 13.*

```
TEUF-CMA($\mathcal{A}$, Corr)
1 : $\mathsf{TSKeyGen}(\lambda) \to ((\mathsf{sk}_i)_{i\in[n]}, \mathsf{pk})$
2 : $L \leftarrow \emptyset$
3 : $st\{\} \leftarrow \emptyset$
4 : $\mathcal{A}^{\mathsf{RunSign}}(\mathsf{pk}, (\mathsf{sk}_i)_{i\in\mathsf{Corr}}) \to (\sigma^*, m^*)$
5 : return $\mathbb{1}_{\mathsf{Verify}(\sigma^*,\mathsf{pk},m^*)=1 \,\wedge\, m^*\notin L}$

RunSign($i, sid, (m_1, ...m_n, m_{ext)})$)
1 : if $st\{i, sid\}$ undefined
2 :     $st\{i, sid\} \leftarrow \mathsf{sk}_i$
3 :     $L \leftarrow L \cup \{m_{ext}\}$
4 : $M \leftarrow (m_1, ...m_n, m_{ext)})$
5 : $(st\{i, sid\}, res) \leftarrow \mathsf{TSSign}_i(st\{i, sid\}, M)$
6 : return $res$
```

Fig. 12: TEUF-CMA game for classical security of threshold signatures.

```
EUF-CMA($\mathcal{A}$)
1 : $\mathsf{KeyGen}(\lambda) \to (\mathsf{sk}, \mathsf{pk})$
2 : $\mathsf{L} \leftarrow \emptyset$
3 : $\mathcal{A}^{\mathsf{Osign}}(\mathsf{pk}) \to (m^*, \sigma^*)$
4 : if $m^* \in L$ :
        return 0
5 : return $\mathsf{Verify}(\sigma^*, \mathsf{pk}, m^*)$

Osign($m$)
1 : $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$
2 : $L \leftarrow L \cup \{m\}$
3 : return $\sigma$
```

Fig. 13: EUF-CMA game for classical signature security

We give further definitions for primitives used in this work. The first one is secret sharing as it is an essential tool for the design of threshold protocols.

Definition 11 (Secret Sharing [9]). *Let D be the domain of inputs and let S be the domain for secret shares. A (t, n)-secret sharing scheme consists of a* PPT *algorithm $\mathsf{Shr}_t^n : D \to S^n$ that generates n secret shares of a value from D and a deterministic polynomial time algorithm for k shares $\mathsf{Rec} : S^k \to D$ that reconstructs the original value from the shares such that the following properties hold:*

– ***Correctness**: For any $v \in D$, $(s_1, s_2, \ldots, s_n) \leftarrow \mathsf{Shr}(v)$ and any $I \subset \{s_1, .., s_n\}$ such that $\#I > t$:*

$$\Pr[\mathsf{Rec}((s_i)_{i\in I}) = v] = 1$$

– ***Security**: For any two values $v_1, v_2 \in D$ and any vector $\mathbf{s}$ of shares with less than $t + 1$ elements:*

$$\Pr[\mathbf{s} = \mathsf{Shr}(v_1)_{|\mathbf{s}|}] = \Pr[\mathbf{s} = \mathsf{Shr}(v_2)_{|\mathbf{s}|}]$$

where $|.|$ denotes the number of elements in a vector and $\mathsf{Shr}(v_1)_{|\mathbf{s}|}$ denotes a subspace of $\mathsf{Shr}(v_1)$ with $|\mathbf{s}|$ elements.

We proceed with some additional definitions regarding properties of hash functions that will be required in the rest of this work.

Definition 12 (2nd pre-image resistance, collision resistance, quantum one-wayness). *A hash function H with message domain $\mathcal{D}(\lambda)$, output domain $\{0,1\}^{\tau(\lambda)}$ (key space $\mathcal{K}(\lambda)$ for collision resistance) and security parameter λ is:*

– *2nd pre-image resistant if for any probabilistic polynomial time algorithm $\mathcal{A}$, we have*

$$\Pr[\mathsf{2PreIm}(\mathcal{A}) \to 1] \leq \mathsf{negl}(\lambda)$$

– *collision resistant if for any probabilistic quantum polynomial time algorithm $\mathcal{A}$, we have*

$$\Pr[\mathsf{CR}(\mathcal{A}) \to 1] \leq \mathsf{negl}(\lambda)$$

– *quantum one-way if for any probabilistic quantum polynomial time algorithm $\mathcal{A}$, we have*

$$\Pr[\mathsf{QOW}(\mathcal{A}) \to 1] \leq \mathsf{negl}(\lambda)$$

The 2PreIm*,* CR *and* QOW*games are described in Figure 14. For simplicity, the parameter λ will be omitted from the notation of $\mathcal{D}$, τ and $\mathcal{K}$.*

```
$\mathsf{2PreIm}(\mathcal{A})$
1 : $x \leftarrow\!\$\, \mathcal{D}$
2 : $\mathcal{A}(x) \to x'$
3 : return $\mathbb{1}_{H(x')=H(x) \land x \neq x'}$
```

```
$\mathsf{CR}(\mathcal{A})$
1 : $\mathsf{hk} \leftarrow\!\$\, \mathcal{K}$
2 : $\mathcal{A}(\mathsf{hk}) \to x, x'$
3 : return $\mathbb{1}_{H(\mathsf{hk},x)=H(\mathsf{hk},x') \land x \neq x'}$
```

```
$\mathsf{QOW}(\mathcal{A})$
1 : $x \leftarrow\!\$\, \mathcal{D}$
2 : $y \leftarrow H(x)$
3 : $\mathcal{A}(y) \to x'$
4 : return $\mathbb{1}_{H(x')=y}$
```

Fig. 14: 2PreIm, CR and QOW games for hash function security.

Finaly, we give some reminders on multi-party computations.

Definition 13 (Multi Party Computation (MPC)). *An n-party MPC protocol consists of n parties $(P_1, \ldots, P_n)$ with their respective private inputs $(\mathsf{in}_1, \ldots, \mathsf{in}_n)$ and an evaluation function $f(\mathsf{in}_1, \ldots, \mathsf{in}_n) = (\mathsf{out}_1, \ldots, \mathsf{out}_n)$ such that for any $i \in [n]$, P_i learns nothing more than in_i and out_i.*

The security of MPC protocols is given in the ideal vs. real paradigm. In the context of static malicious adversaries, we define the following distributions.

Definition 14 (Ideal vs. real distributions - malicious adversary ([9])). *Given an adversary $\mathcal{A}$, let $\mathsf{Corr}(\mathcal{A})$ be the set of parties corrupted by $\mathcal{A}$ and $\mathsf{Corr}(\mathsf{Sim})$ the set of parties corrupted by the ideal adversary* Sim*. These are fixed at the beginning of the protocol, and cannot be changed afterwards since we consider a static corruption model. We define two associated distributions.*

– $\mathsf{Real}_{\pi,\mathcal{A}}(\lambda; \{\mathsf{in}_i | i \notin \mathsf{Corr}(\mathcal{A})\})$*: Run the protocol with security parameter λ. Each honest party P_i runs the protocol honestly on given private input in_i. The messages of corrupted parties are chosen according to $\mathcal{A}$.*

We denote by $y_i = (\mathsf{out}_i, \mathsf{out})$ *the output of honest party* P_i *and* View_j *the final view of party* P_j.
The output is $(\{\mathsf{View}(\mathcal{A})\}, \{y_i | i \notin \mathsf{Corr}(\mathcal{A})\})$

– $\mathsf{Ideal}_{\mathcal{F},\mathsf{Sim}}(\lambda; \{\mathsf{in}_i | i \notin \mathsf{Corr}(\mathsf{Sim})\})$*: On input* $\{\mathsf{in}_i | i \notin \mathsf{Corr}(\mathsf{Sim})\}$*, run* Sim *until it outputs a set of inputs* $\{\mathsf{in}_j | j \in \mathsf{Corr}(\mathcal{A})\}$ *and keeps a state st.*
Compute $(y_1, .., y_n) \leftarrow \mathcal{F}(\mathsf{in}_1, ..., \mathsf{in}_n)$.
Return $\{y_j | j \in \mathsf{Corr}(\mathcal{A})\}$ *to* Sim.
Sim *returns as final output a set of simulated views* View^*.
The final output is $(\mathsf{View}^*, \{y_i | i \notin \mathsf{Corr}(\mathcal{A})\})$

Definition 15 (Malicious security for MPC ([9])). *A protocol* π *securely realizes an ideal functionality* $\mathcal{F}$ *in the presence of a malicious adversary if for every real-world adversary* $\mathcal{A}$*, there exist a simulator* Sim *with* $\mathsf{Corr}(\mathcal{A}) = \mathsf{Corr}(\mathsf{Sim})$ *such that, for all inputs of the honest parties* $\{x_i | i \notin \mathsf{Corr}(\mathcal{A})\}$ *the distributions*

$$\mathsf{Real}_{\pi,\mathcal{A}}(\lambda; \{\mathsf{in}_i | i \notin \mathsf{Corr}(\mathcal{A})\})$$
$$\mathsf{Ideal}_{\mathcal{F},\mathsf{Sim}}(\lambda; \{\mathsf{in}_i | i \notin \mathsf{Corr}(\mathsf{Sim})\})$$

are indistinguishable in λ.

If we assume honest majority, by Goldreich et al. [11], we have that for **any function** it is possible to achieve a secure multiparty protocol with fairness and guaranteed output delivery with information theoretic security. This assumes that the parties have access to private point-to-point channels as well as a broadcast channel ([14]). If we do not assume honest majority, we have to give up fairness or guaranteed output delivery.

B Proofs of Theorem 1 and 5

In this section, we give a detailed proof of Theorem 1 and Theorem 5. We recall them each time for completeness

Theorem 9 (Classical security of $\mathsf{HiddenPK}$**).** *The* $\mathsf{HiddenPK}(\varSigma, \lambda, H)$ *transform is secure against existential forgery under chosen-message attack if the underlying signature scheme* $\varSigma$ *is and if* H *is 2nd pre-image resistant.*

Proof. Consider an EUF-CMA adversary $\mathcal{A}'$ for $\mathsf{HiddenPK}(\varSigma, \lambda, H)$.

Let $\mathsf{2PreIm} = \{\text{Given } \rho \text{ from } \mathsf{sk}, \rho^* \text{ from } \sigma^* : \rho \neq \rho^*, H(\rho^*) = H(\rho)\}$. Since EUF-CMA$(\mathcal{A}') \to 1$ implies that $H(\rho) = H(\rho^*)$, we have:

$$\begin{aligned}\Pr[\text{EUF-CMA}(\mathcal{A}') \to 1] &= \Pr[\text{EUF-CMA}(\mathcal{A}') \to 1 \land \mathsf{2PreIm}] \\ &\quad + \Pr[\text{EUF-CMA}(\mathcal{A}') \to 1 \land \overline{\mathsf{2PreIm}}] \\ &\leq \Pr[\mathsf{2PreIm}] \\ &\quad + \Pr[\text{EUF-CMA}(\mathcal{A}') \to 1 \land \rho = \rho^*]\end{aligned}$$

Intuitively, the first term corresponds to the 2nd pre-image resistance of H and the second to the EUF-CMA security of Σ. We want to show that they are both negligible.

Let us consider an adversary $\mathcal{C}$ for 2nd pre-image resistance (Figure 15). We have $\mathsf{negl}(\lambda) = \Pr[\mathcal{C} \text{ wins}] \geq \Pr[\mathsf{2PreIm}]$ because of 2nd pre-image resistance.

Now let us consider an EUF-CMA adversary $\mathcal{A}$ for Σ (Figure 15). We have that the view of $\mathcal{A}'$ called as a subroutine of EUF-CMA($\mathcal{A}$) is the same as the view of $\mathcal{A}'$ in EUF-CMA($\mathcal{A}'$).
If we assume that $\mathcal{A}'$ succeeds and $\overline{\mathsf{2PreIm}}$ then $\mathcal{A}$ succeeds. Indeed, if we have $\overline{\mathsf{2PreIm}}$ and a success for $\mathcal{A}'$, it implies that $\rho = \rho^*$ and $\mathsf{HiddenPK\text{-}Verify}(m^*, \tilde{\sigma}^*, \tilde{\mathsf{pk}}) = 1$. Hence (m^*, σ^*) is a valid forgery for $\mathcal{A}$.
Therefore we have $\Pr[\text{EUF-CMA}(\mathcal{A}) \to 1] \geq \Pr[\text{EUF-CMA}(\mathcal{A}') \to 1 \wedge \overline{\mathsf{2PreIm}}]$ and the first term is negligible by assumption. □

```
$\mathcal{C}(\rho)$
1 : $(\mathsf{sk}, \mathsf{pk}) \leftarrow\$ \mathsf{KeyGen}_\Sigma(\lambda)$
2 : $\tilde{\mathsf{sk}} \leftarrow (\mathsf{sk}, \rho)$
3 : $\tilde{\mathsf{pk}} \leftarrow (H(\rho), \mathsf{pk} + \rho)$
4 : $\mathcal{A}'(\tilde{\mathsf{pk}})$
      1 : if query $m_i$ :
              $\mathcal{C}$ computes $\sigma_i = \mathsf{Sign}(\mathsf{sk}, m_i)$
              $\mathcal{C}$ returns $\sigma'_i = (\sigma_i, \rho)$ to $\mathcal{A}'$
      2 : return $(m, \tilde{\sigma})$
5 : $\tilde{\sigma} \to (\sigma^*, \rho^*)$
6 : return $\rho^*$
```

```
$\mathcal{A}^{\mathsf{OSign}}(\mathsf{pk})$
1 : $\rho \leftarrow\$ G$
2 : $\tilde{\mathsf{pk}} \leftarrow (H(\rho), \mathsf{pk} + \rho)$
3 : $\mathcal{A}'(\tilde{\mathsf{pk}})$
      1 : if query $m_i$ :
              $\sigma_i \leftarrow \mathsf{OSign}_\mathcal{A}(m_i)$
              $\mathcal{A}$ returns $(\sigma_i, \rho)$ to $\mathcal{A}'$
      2 : return $(m^*, \tilde{\sigma}^*)$
4 : $\tilde{\sigma}^* \to (\sigma^*, \rho^*)$
5 : return $(m^*, \sigma^*)$
```

Fig. 15: Adversary $\mathcal{C}$ calling $\mathcal{A}'$ for 2nd pre-image resistance (left) and adversary $\mathcal{A}$ calling $\mathcal{A}'$ for EUF-CMA($\mathcal{A}$) (right) (Theorem 1).

Theorem 10. ($\mathsf{QEUF\text{-}KOA} \to \mathsf{OTDU}$) *For a given* $\mathsf{QEUF\text{-}KOA}$ *secure* $\mathsf{HiddenPK}$ *signature scheme* Σ_{HPK}*, and a random oracle* Commit *with output length* λ*, the resulting delayed signature scheme* $\mathsf{DelayHiddenPK}$ *(Figure 7) is unforgeable.*

Proof. We set $\mathsf{G0}$ to be the OTDU game. Given an adversary $\mathcal{A}$ that wins $\mathsf{G0}$ we reduce $\mathsf{G0}$ to $\mathsf{QEUF\text{-}KOA}$ as follows:

- $\mathsf{G1}$: We put an additional constraint to $\mathsf{G0}$ that Commit does not have any collisions. Since Commit is a random oracle, the probability of having a collision is negligible. Hence, we have:

$$|\mathsf{Adv}^{\mathsf{G1}}_\mathcal{A} - \mathsf{Adv}^{\mathsf{G0}}_\mathcal{A}| \leq \mathsf{negl}(\lambda)$$

- $\mathsf{G2}$: We put another additional constraint that the s_1 that is output by $\mathcal{A}$ was returned from a Commit query that is queried before the final $\mathsf{Delay\text{-}Verify}$ (line 5 of OTDU). Note that this implies that the output s_2, m was queried to the random oracle in the first phase of $\mathcal{A}$. If the constraint is not satisfied, it implies that the adversary has successfully predicted a fresh random oracle query which is negligible in λ. That is:

$$|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{G2}} - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{G1}}| \leq \mathsf{negl}(\lambda)$$

- $\mathsf{G3}$: In order to simulate the second phase of $\mathcal{A}$ we intercept all random oracle queries s_2, parse the values $(\tilde{\sigma}, m, \mathsf{open})$ and their corresponding oracle replies s_1, check if
$\mathsf{HiddenPK\text{-}Verify}(\tilde{\sigma}, \tilde{\mathsf{pk}}, \mathsf{m}) = 1$ and we output the correct s_1, s_2, m triplet. Since $|L|$ is polynomial, we can exhaust L for all s_1 values and find the correct triplet with probability 1. Hence, this game is at least as good as $\mathsf{G2}$ in terms of advantage, that is:

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{G3}} \geq \mathsf{Adv}_{\mathcal{A}}^{\mathsf{G2}}$$

- $\mathsf{G4}$: We add the constraint that the query $(\tilde{\sigma}, m, \mathsf{open})$ from OSig was not queried by $\mathcal{A}$. Otherwise, it would imply that the adversary correctly guessed a correct opening open for a forged $\tilde{\sigma}$. Hence, the difference between $\mathsf{G4}$ and $\mathsf{G3}$ is negligible:

$$|\mathsf{Adv}_{\mathcal{A}}^{\mathsf{G4}} - \mathsf{Adv}_{\mathcal{A}}^{\mathsf{G3}}| \leq \mathsf{negl}(\lambda)$$

- $\mathsf{G5}$: Since Osig only returns the fresh commitment output σ_1 which is a random oracle output, we simulate Osig oracle by randomly sampling a σ_1 from the output domain of the random oracle. Hence, the advantage remains the same:

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{G5}} = \mathsf{Adv}_{\mathcal{A}}^{\mathsf{G4}}$$

The $\mathsf{QEUF\text{-}KOA}$ adversary $\mathcal{A}'$ runs the adversary $\mathcal{A}$ playing $\mathsf{G5}$. This produces a valid forgery $(\tilde{\sigma}, m)$ on the public key $\tilde{\mathsf{pk}}$. However, since $\mathsf{Adv}_{\mathcal{A},\Sigma_{\mathsf{HPK}}}^{\mathsf{QEUF\text{-}KOA}}$ is negligible, by combining the results for $\mathsf{G5}, \mathsf{G4}, \mathsf{G3}, \mathsf{G2}$ and $\mathsf{G1}$ we have:

$$\begin{aligned} \mathsf{Adv}_{\mathcal{A}}^{\mathsf{G0}} &\leq \mathsf{negl}(\lambda) \\ \mathsf{Adv}_{\mathcal{A}}^{\mathsf{OTDU}} &\leq \mathsf{negl}(\lambda) \end{aligned}$$

□

C Bitcoin Scripts

```
HashedPK(P2PKH): <sig> <pubKey> OP_DUP OP_HASH160
<pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG

HiddenPK-Verify: <sig> <rho> OP_DUP OP_SHA256
<pk1> OP_EQUALVERIFY <pk2> OP_SUB* OP_CHECKSIG

DelayL-Verify: <pk> <m|sig2>  OP_DUP OP_SHA256
<sig1> OP_EQUALVERIFY HiddenPK-Verify

VerifyOpen: <t1> OP_SHA256 <Ht1> OP_EQUAL

VerifyClose: <t2> OP_SHA256 <Ht2> OP_EQUAL
```

Fig. 16: Bitcoin Script implementations for the delayed signature construction in Figure 9. Note that the `HiddenPK-Verify` script would not work out of the box since `OP_SUB` has to be followed by an `OP_MOD` operation to realize the group addition but `OP_MOD` is currently disabled. Also note that although it is not possible to call subroutines, we call `HiddenPK-Verify` inside `DelayL-Verify` to present the logic in a more clear way.

D Proof of classical security of ThreshHiddenPK

We prove here in details Theorem 7. We re-state it below for completeness.

Theorem 11 (Classical security of ThreshHiddenPK). *The* ThreshHiddenPK(Σ, λ, H) *transform is existentially unforgeable under chosen-message attack (TEUF-CMA) if the underlying threshold signature scheme Σ is and if H is 2nd pre-image resistant.*

Proof. Consider an adversary $\mathcal{A}'$ for ThreshHiddenPK(Σ, λ, H) with set of corrupted players Corr.
Let $\mathsf{PreImRes} = \{\exists i$ such that given ρ_i, ρ_i^* from $\sigma^*; \rho_i \neq \rho_i^*$ and $H(\rho_i) = H(\rho_i^*)\}$. Observe that $\mathsf{TEUF\text{-}CMA}(\mathcal{A}') \to 1$ implies that for all i, $H(\rho_i) = H(\rho_i^*)$. Therefore, we have

$$\begin{aligned}\Pr[\mathsf{TEUF\text{-}CMA}(\mathcal{A}') \to 1] &= \Pr[\mathsf{TEUF\text{-}CMA}(\mathcal{A}') \to 1 \wedge \mathsf{PreImRes}] + \\ &\quad \Pr[\mathsf{TEUF\text{-}CMA}(\mathcal{A}') \to 1 \wedge \overline{\mathsf{PreImRes}}] \\ &\leq \Pr[\mathsf{PreImRes}] \\ &\quad + \Pr[\mathsf{TEUF\text{-}CMA}(\mathcal{A}') \to 1 \wedge (\forall i\ \rho_i = \rho_i^*)]\end{aligned}$$

Intuitively, the first term represents the 2nd pre-image resistance of H and the second the TEUF-CMA security of Σ.

Consider an adversary $\mathcal{C}$ for multi-target pre-image resistance (See Figure 17). It finds a pre-image for one out of n targets. We can reduce to a single target adversary by guessing which target will be solved. Since $n = poly(\lambda)$, we guess the correct target with probability $\frac{1}{n}$ which is non-negligible. We have $\Pr[\mathsf{PreImRes}] \leq \Pr[\mathsf{2PreIm}(\mathcal{C}) \to 1] = \mathsf{negl}(\lambda)$ because of 2nd pre-image resistance.

Now consider a TEUF-CMA adversary $\mathcal{A}$ for Σ in Figure 17. We have that the view of $\mathcal{A}'$ called as a subroutine of $\mathsf{TEUF\text{-}CMA}(\mathcal{A})$ is the same as the view of $\mathcal{A}'$ in $\mathsf{TEUF\text{-}CMA}(\mathcal{A}')$.
If we assume that $\mathcal{A}'$ succeeds and $\overline{\mathsf{PreImRes}}$ then $\mathcal{A}$ succeeds. Indeed, if we have $\overline{\mathsf{PreImRes}}$ and a success for $\mathcal{A}'$, it implies that for all i, $\rho_i = \rho_i^*$ and $\mathsf{ThreshHiddenPK\text{-}Verify}(m^*, \tilde{\sigma}^*, \tilde{\mathsf{pk}}) = 1$. Hence (m^*, σ^*) is a valid forgery for $\mathcal{A}$.
Therefore we have

$$\Pr[\mathsf{TEUF\text{-}CMA}(\mathcal{A}) \to 1] \geq \Pr[\mathsf{TEUF\text{-}CMA}(\mathcal{A}') \to 1 \wedge \overline{\mathsf{PreImRes}}]$$

and the first term is negligible by assumption.

```
C(ρ_1, ..., ρ_n)
1 :  $(\mathsf{sk}_i, \mathsf{pk}_i)_{i\in[n]} \leftarrow\$ \mathsf{TSKeyGen}^1_\Sigma(\lambda)$
2 :  $\tilde{\mathsf{sk}}_i \leftarrow (\mathsf{sk}_i, \rho_i)\ i = 1, ..., n$
3 :  $\tilde{\mathsf{pk}}_1 \leftarrow (H(\rho_i))_{i\in[n]}$
4 :  $\tilde{\mathsf{pk}}_2 \leftarrow ((\mathsf{pk}_i + \rho_i))_{i\in[n]}$
5 :  $\mathcal{A}'(\tilde{\mathsf{pk}}, (\tilde{\mathsf{sk}}_i)_{i\in\mathsf{Corr}})$
       1 :  if query $(i, sid, (m_1, ...m_n, m_{ext)}))$ :
       2 :    $\mathcal{C}$ does:
       3 :    if $st\{i, sid\}$ undefined
       4 :      $st\{i, sid\} \leftarrow \mathsf{sk}_i$
       5 :      $L \leftarrow L \cup \{m_{ext}\}$
       6 :      $M \leftarrow (m_1, ...m_n, m_{ext)})$
       7 :    $(st\{i, sid\}, res) \leftarrow \mathsf{TSSign}_i(st\{i, sid\}, M)$
       8 :    $\mathcal{C}$ returns $\mathsf{res}$ to $\mathcal{A}'$
       9 :  return $(\tilde{\sigma}^*, m^*)$
6 :  $\tilde{\sigma}^* \to (\sigma^*, S, (\rho_i^*)_{i\in S})$
7 :  if $\forall i\ \rho_i = \rho_i^*$ or $H(\rho_i^*) \neq H(\rho_i)$ :
8 :    return $\perp$
9 :  else return $(i, \rho_i^*)$
```

```
TEUF-CMA(A, Corr)
1 :  $\mathsf{TSKeyGen}(\lambda) \to ((\mathsf{sk}_i)_{i\in[n]}, \mathsf{pk})$
2 :  $L \leftarrow \emptyset$
3 :  $st\{\}$
4 :  $\mathcal{A}^{\mathsf{RunSign}}(\mathsf{pk}, (\mathsf{sk}_i)_{i\in\mathsf{Corr}})$
       1 :  $\rho_i \leftarrow\$ \mathcal{D}, i = 1, \ldots n$
       2 :  $(\tilde{\mathsf{sk}}_i) \leftarrow (\mathsf{sk}_i, \rho_i), i \in \mathsf{Corr}$
       3 :  $\tilde{\mathsf{pk}}_1 \leftarrow (H(\rho_i))_{i\in[n]}$
       4 :  $(\mathsf{pk}_i)_{i\in[n]} \leftarrow \mathsf{Shr}^n_t(\mathsf{pk})$
       5 :  $\tilde{\mathsf{pk}}_2 \leftarrow (\mathsf{pk}_i + \rho_i)_{i\in[n]}$
       6 :  Run $\mathcal{A}'(\tilde{\mathsf{pk}}, (\tilde{\mathsf{sk}}_i)_{i\in\mathsf{Corr}})$ :
       7 :    if query $(i, sid, (m_1, ...m_n, m_{ext)}))$ :
       8 :      $\mathsf{res} \leftarrow \mathsf{RunSign}^{\mathcal{A}}_i(i, sid, (m_1, ...m_n, m_{ext)}))$
       9 :      $\mathcal{A}$ returns $\mathsf{res}$ to $\mathcal{A}'$
       10 : $\mathcal{A}'$ returns $(m^*, \tilde{\sigma}^*)$ to $\mathcal{A}$
       11 : $\tilde{\sigma}^* \to (\sigma^*, S, (\rho_i^*)_{i\in S})$
       12 : return $(m^*, \sigma^*)$
5 :  return $\mathbb{1}_{\mathsf{Verify}(\sigma^*, \mathsf{pk})=1\ \wedge\ m^* \notin L}$
```

Fig. 17: Adversary $\mathcal{C}$ against 2nd pre-image resistance (left) and $\mathsf{TEUF\text{-}CMA}(\mathcal{A})$ calling $\mathcal{A}'$ (right) (Theorem 7).

E Tailoring the **HiddenPK** and **ThreshHiddenPK** transforms to ECDSA

In this section, we provide a tailored version of the HiddenPK to fit ECDSA more closely.

Observe that given an ECDSA signature $\sigma = (r, s)$ over an elliptic curve $E/\mathbb{F}_q$ with a point G of order n, we can actually recover two possible public key from the signature in the following way : Let $x(.)$ denote the x-coordinate of the associated point and set R^1, R^2 to be the two points with $x(R^i) = r$, then

$$\mathsf{pk}^i = \frac{s}{r} \cdot R^i - \left(\frac{H(m)}{r}\right) \cdot G$$

This gives rise to two possible public keys $\mathsf{pk}^1, \mathsf{pk}^2$. We denote this process $\mathsf{ComputePKs}(\sigma)$.
We can use that fact to remove the need to release ρ together with signature. Hence we modify the $\mathsf{HiddenPK}$ transform slightly. See Figure 18.
The advantage of this tailored transform is that the signing algorithm is just

```
HiddenPK-ECDSA-KeyGen($\lambda$)
1 : $(\tilde{\mathsf{sk}}, \tilde{\mathsf{pk}}) \leftarrow \mathsf{HiddenPK\text{-}KeyGen}(\lambda)$
2 : return $(\tilde{\mathsf{sk}}, \tilde{\mathsf{pk}})$

HiddenPK-ECDSA-Sign($\tilde{\mathsf{sk}}$, msg)
1 : $\sigma \leftarrow \mathsf{Sign}_{\mathrm{ECDSA}}(\mathsf{sk}, \mathsf{msg})$
2 : return $\sigma$

HiddenPK-ECDSA-Verify($\sigma, \tilde{\mathsf{pk}}, \mathsf{msg}$)
1 : $\mathsf{pk}^1, \mathsf{pk}^2 \leftarrow \mathsf{ComputePKs}(\sigma)$
2 : $(\tilde{\mathsf{pk}}_1, \tilde{\mathsf{pk}}_2) \leftarrow \tilde{\mathsf{pk}}$
3 : $\rho^i \leftarrow \tilde{\mathsf{pk}}_2 - \mathsf{pk}^i,\ i = 1, 2$
4 : if $H'(\rho^i) = \tilde{\mathsf{pk}}_1$ for some $i = 1, 2$ :
5 :     return 1
6 : return 0
```

Fig. 18: $\mathsf{HiddenPK\text{-}ECDSA}(E/\mathbb{F}_q, G, n, H, H')$ signature hiding the public key.

the regular ECDSA signing, the signature is not changed. Key generation comes from the $\mathsf{HiddenPK}$ transform and the verification algorithm uses our discussion above.

Theorem 12 (Security of $\mathsf{HiddenPK\text{-}ECDSA}$). *If H' is 2nd preimage resistant and ECDSA is EUF-CMA, then* $\mathsf{HiddenPK}$*-ECDSA is EUF-CMA. If H' is quantum one-way,* $\mathsf{HiddenPK\text{-}ECDSA}$ *is* $\mathsf{QEUF\text{-}KOA}$*.*

Proof. We show that this is equivalent to the generic transform of Figure 2 applied to ECDSA.

Assume that $\mathsf{HiddenPK}$ applied to ECDSA is EUF-CMA. Consider an EUF-CMA adversary $\mathcal{A}$ for $\mathsf{HiddenPK\text{-}ECDSA}$, we construct an adversary $\mathcal{B}$ for the EUF-CMA game of $\mathsf{HiddenPK}$ applied to ECDSA in Figure 19.

Clearly if $\mathcal{A}$ wins EUF-CMA($\mathcal{A}$) then $\mathcal{B}$ wins EUF-CMA($\mathcal{B}$). Hence, we have

$$\Pr[\text{EUF-CMA}(\mathcal{A}) \to 1] \leq \Pr[\text{EUF-CMA}(\mathcal{B}) \to 1] = \mathsf{negl}(\lambda)$$

Then, we apply Theorem 1 on EUF-CMA($\mathcal{B}$).

```
$\mathcal{B}^{\mathsf{Osign}}(\tilde{\mathsf{pk}})$
1 : $\mathcal{A}(\tilde{\mathsf{pk}})$
        1 : if query $m_i$
        2 :     $\mathcal{B}$ queries $\mathsf{Osign}_{\mathcal{B}}(m_i) \to \tilde{\sigma}_i$
                $(\sigma_i, \rho_i) \leftarrow \tilde{\sigma}_i$
                $\mathcal{B}$ returns $(\sigma_i)$ to $\mathcal{A}$
        3 : return $(m^*, \sigma^*)$
2 : $\mathsf{pk}^1, \mathsf{pk}^2 \leftarrow \mathsf{ComputePKs}(\sigma^*)$
3 : $\rho^i \leftarrow \tilde{\mathsf{pk}}_2 - \mathsf{pk}^i,\ i = 1, 2$
4 : if $H(\rho^i) = \tilde{\mathsf{pk}}_1$ :
5 :     return $(m^*, (\sigma^*, \rho^i))$
```

Fig. 19: Classical security equivalence of HiddenPK-ECDSA and HiddenPK applied to ECDSA (Theorem 12).

For quantum security, we similarly apply the above reduction, the only difference is that there is no Osign anymore, and we can apply Theorem 2. The hypothesis on the statistical distance is verified since the statistical distance between the distribution of pk and the uniform distribution in $\langle G \rangle$ is $\frac{1}{|\langle G \rangle|} = \frac{1}{n}$.

E.1 An application to Threshold ECDSA

We give an example of application to the Threshold ECDSA protocol described in [16].

We describe in Figure 20 the adapted Key Generation protocol in details. The signature protocol is basically the same as the one described in [16] with a broadcast round added at the end to return the value ρ_i for each participant. We additionally describe the verification protocol in Figure 21.

F Instantiating Multiparty Key Generation for ThreshHiddenPK

So far we have assumed a secure set-up for multiparty key generation. In this section, we discuss how to achieve it. A reminder on multi-party computations can be found at the end of section A.

Theorem 13 (Security of Key Generation). *If there exists a secure protocol* $\pi_{\mathsf{TSKeyGen}^1}$ *that securely realizes* $\mathsf{TSKeyGen}^1$*, then* Π *(Figure 22) securely realizes* ThreshHiddenPK-KeyGen.

```
ThreshHiddenPK-KeyGen($\lambda$)
      Each participant $P_i$ does :
 1 :  $a_{i\ell}, b_{i\ell} \leftarrow\!\!\$\, \mathbb{Z}_n$ for $j = 0, .., t$
 2 :  $f_i(x) \leftarrow \sum_{j=0}^{t} a_{ij}x^j$ ; $g_i(x) \leftarrow \sum_{j=0}^{t} b_{ij}x^j$
 3 :  $c_{i\ell} \leftarrow a_{i\ell} \cdot G + b_{i\ell}H$, $H$ generator
 4 :  Send$f_i(j), g_i(j)$ to each participant $P_j$
 5 :  Receive $f_j(i), g_j(i)$ for $j \neq i$
 6 :  Check $f_i(j) \cdot G + g_i(j) \cdot H = \sum_{\ell=0}^{t} j^t C_{i\ell}$ for $j \neq i$
 7 :  Participants that pass this check are added to the subset $\mathcal{Q}$
      If a participant fails the check, they must broadcast the values $f_i(j), g_i(j)$
 8 :  $\mathsf{sk}_i \leftarrow \sum_{j \in \mathcal{Q}} f_j(i)$ ; $\mathsf{pk}_i \leftarrow \mathsf{sk}_i \cdot G$
 9 :  $\rho_i \leftarrow\!\!\$\, \langle G \rangle$
10 :  $\tilde{\mathsf{sk}}_i \leftarrow (\mathsf{sk}_i, \rho_i)$
11 :  Compute $H(\rho_i)$ and broadcast it
12 :  Receive $H(\rho_j)$ for $j \neq i$
13 :  Broadcast $\mathsf{pk}_i + \rho_i$
14 :  Receive $\mathsf{pk}_j + \rho_j$ for $j \neq i$
15 :  $\tilde{\mathsf{pk}}_1 \leftarrow (H(\rho_i))_{i \in \mathcal{Q}} \tilde{\mathsf{pk}}_2 \leftarrow (\mathsf{pk}_i + \rho_i)_{i \in \mathcal{Q}}$
16 :  $\tilde{\mathsf{pk}} \leftarrow (\tilde{\mathsf{pk}}_1, \tilde{\mathsf{pk}}_2)$
17 :  return $((\tilde{\mathsf{sk}}_i)_{i \in \mathcal{Q}}, \tilde{\mathsf{pk}})$
```

Fig. 20: Key Generation for the ThreshHiddenPK transform applied to [16]

```
$\mathsf{ThreshHiddenPK\text{-}Verify}(\tilde{\sigma}, \tilde{\mathsf{pk}}, \mathsf{msg})$
1 : $(\sigma, S, (\rho_i)_{i \in S}) \leftarrow \tilde{\sigma}$
2 : $(\tilde{\mathsf{pk}}_1, \tilde{\mathsf{pk}}_2) \leftarrow \tilde{\mathsf{pk}}$
3 : for each $i \in S$ :
4 :     if $H(\rho_i) \neq (\tilde{\mathsf{pk}}_1)_i$ :
5 :         return 0
6 : for each $i \in S$ :
7 :     $\mathsf{pk}_i \leftarrow (\tilde{\mathsf{pk}}_2)_i - \rho_i$
8 : Use Lagrange interpolation with the $\mathsf{pk}_i = (\sum_{j \in Q} f_j(i)) \cdot G$
      to recover $\mathsf{pk} = (\sum_{j \in Q} f_j(0)) \cdot G$
9 : return $\mathsf{Verify}_{\Sigma}(\sigma, \mathsf{pk}, \mathsf{msg})$
```

Fig. 21: Verification for the ThreshHiddenPK transform applied to [16]

```
$\Pi$
1 : $(\mathsf{sk}_i, \mathsf{pk}_i)_{i \in [n]} \leftarrow \pi_{\mathsf{TSKeyGen}^1}$
2 :  Each participant $i$ does :
3 :     $\rho_i \leftarrow\$ \mathcal{D}$
4 :     $\tilde{\mathsf{sk}}_i \leftarrow (\mathsf{sk}_i, \mathsf{pk}_i)$
5 :     Compute and broadcast $H(\rho_i)$
6 :     Receive $H(\rho_j), j = 1, ..., n$
7 :     Compute and broadcast $\mathsf{pk}_i + \rho_i$
8 :     Receive $\mathsf{pk}_j + \rho_j, j = 1, ..., n$
9 :     $\tilde{\mathsf{pk}}_1 \leftarrow (H(\rho_i))_{i \in [n]}$
10 :    $\tilde{\mathsf{pk}}_2 \leftarrow (\mathsf{pk}_i + \rho_i)_{i \in [n]}$
11 : return $(\mathsf{sk}_i)_{i \in [n]}, \tilde{\mathsf{pk}}$
12 :
```

Fig. 22: Protocol Π realizing ThreshHiddenPK-KeyGen.

Proof. We want to show that given a real adversary $\mathcal{A}$ against Π, there exists an ideal adversary Sim such that

$$\mathsf{Real}(\mathcal{A}, \Pi) \sim \mathsf{Ideal}(\mathsf{Sim}, \mathsf{ThreshHiddenPK\text{-}KeyGen})$$

i.e. the two distributions are indistinguishable. (We abbreviate the notation from Def. 14 for clarity)
Given such an adversary $\mathcal{A}$, consider the adversary $\mathcal{A}'$ for $\pi_{\mathsf{TSKeyGen}^1}$ who runs $\mathcal{A}$ until the honest participants terminate in $\pi_{\mathsf{TSKeyGen}^1}$. $\mathcal{A}'$ outputs the state of $\mathcal{A}$ when $\pi_{\mathsf{TSKeyGen}^1}$ terminated so that the computation of $\mathcal{A}$ can be resumed.
Given $\mathsf{Real}(\mathcal{A}', \pi_{\mathsf{TSKeyGen}^1})$ we define an algorithm $\mathcal{B}$ that will play the role of the honest players in Π, i.e. $\mathcal{B}$ just follows the protocol Π from Line 2 onward on behalf of the honest players.

We have that the output of $\mathcal{B}$ is precisely $\mathsf{Real}(\mathcal{A}', \Pi)$.
Now given an adversary $\mathcal{A}'$ for $\pi_{\mathsf{TSKeyGen}^1}$, since $\pi_{\mathsf{TSKeyGen}^1}$ securely realizes $\mathsf{TSKeyGen}^1$ by assumption, there exists an ideal adversary Sim' such that

$$\mathsf{Ideal}(\mathsf{Sim}', \mathsf{TSKeyGen}^1) \sim \mathsf{Real}(\mathcal{A}', \pi_{\mathsf{TSKeyGen}^1})$$

We now construct an ideal adversary Sim for $\mathsf{ThreshHiddenPK\text{-}KeyGen}$ in the following way :

1. Sim runs Sim' and gets $\mathsf{Ideal}(\mathsf{Sim}', \mathsf{TSKeyGen}^1)$.
2. Sim simulates B on this output and gives the final output.

We have

$$\begin{aligned}
\mathsf{Ideal}(\mathsf{Sim}, \mathsf{ThreshHiddenPK\text{-}KeyGen}) &= B(\mathsf{Ideal}(\mathsf{Sim}', \mathsf{TSKeyGen}^1)) \\
&\sim B(\mathsf{Real}(\mathcal{A}', \pi_{\mathsf{TSKeyGen}^1})) \\
&= \mathsf{Real}(\mathcal{A}, \Pi)
\end{aligned}$$

G Alternative Extension to Threshold Signatures

In Section 4, we introduced an extension of our generic transformation to threshold signatures, but the construction has two downfalls. Namely the public key and signature sizes grow linearly with the number of participants and the output signature is transparent. In this section, we introduce an alternative extension to threshold signatures with constant public key and signature size and a non-transparent resulting signature. However, there is an additional computational overhead during key generation consisting of evaluating a hash function in a multi-party setting. Moreover, the security of the MPC implementation of KeyGen is not generic.

G.1 Description of the Transform

Given a threshold signature scheme Σ with associated parameters λ, we assume that $\mathsf{TSKeyGen}$ has a specific form i.e. it consists of an algorithm $((\mathsf{sk}_i, \mathsf{pk}_i)_{i\in[n]}) \leftarrow \mathsf{TSKeyGen}^1(\lambda)$, which is followed by a final computation $\mathsf{pk} \leftarrow \mathsf{Rec}((\mathsf{pk}_i)_{i\in S})$, where Rec is the linear reconstruction function from the associated linear secret sharing scheme in $\mathcal{D}$.

Given a threshold signature scheme Σ with associated parameters λ, a (t, n) secret sharing scheme $(\mathsf{Shr}_t^n, \mathsf{Rec})$, and a hash function H, an alternative (generic) threshold with hidden public key transform $\mathsf{CompactThreshHPK}$ is describe in Figure 23.

```
CompactThreshHPK-KeyGen($\lambda$)
1 : $(\mathsf{sk}_i, \mathsf{pk}_i)_{i\in[n]} \leftarrow\!\!\$\ \mathsf{TSKeyGen}^1_\Sigma(\lambda)$
2 : $\rho \leftarrow\!\!\$\ \mathcal{D}$
3 : $(\rho_i)_{i\in[n]} \leftarrow \mathsf{Shr}_t^n(\rho)$
4 : $\tilde{\mathsf{sk}}_i \leftarrow (\mathsf{sk}_i, \rho_i)$
5 : $\tilde{\mathsf{pk}}_1 \leftarrow H(\rho)$
6 : $\tilde{\mathsf{pk}}_2 = \mathsf{pk} + \rho$
7 : $\tilde{\mathsf{pk}} \leftarrow (\tilde{\mathsf{pk}}_1, \tilde{\mathsf{pk}}_2)$
8 : return $((\tilde{\mathsf{sk}}_i)_{i\in[n]}, \tilde{\mathsf{pk}})$

CompactThreshHPK-Sign($(\tilde{\mathsf{sk}}_i)_{i\in S}$, msg)
    Each $P_i$ does:
1 : $(\mathsf{sk}_i, \rho_i) \leftarrow (\tilde{\mathsf{sk}}_i)$
2 : $\sigma \leftarrow \mathsf{TSSign}_\Sigma((\mathsf{sk}_i)_{\in S}, \mathsf{msg})$
3 : Exchange $\rho_i$ with participants
4 : $\rho \leftarrow \mathsf{Rec}((\rho_i)_{i\in S})$
5 : $\tilde{\sigma} \leftarrow (\sigma, \rho)$
6 : return $\tilde{\sigma}$

CompactThreshHPK-Verify($\tilde{\sigma}, \tilde{\mathsf{pk}}$, msg)
1 : return $\mathsf{HiddenPK\text{-}Verify}_\Sigma(\tilde{\sigma}, \tilde{\mathsf{pk}}, \mathsf{msg})$
```

Fig. 23: Threshold signature hiding the public key: the $\mathsf{CompactThreshHPK}(\Sigma, \lambda, H)$ transform.

Note that in Figure 23, just as in the regular threshold setting, $\mathsf{HiddenPK\text{-}Verify}$ refers to the verification algorithm for the single signer version of our transform in Figure 2, namely $\mathsf{HiddenPK}$. Additionally, we remind that we consider a secure set-up, and therefore key generation is given as an ideal functionality, and we do not provide a specific protocol for realizing it.

G.2 Security of **CompactThreshHPK**

Here we prove the classical and quantum security of the $\mathsf{CompactThreshHPK}$ transform.

Theorem 14 (Classical security of $\mathsf{CompactThreshHPK}$**).**
A threshold signature $\mathsf{CompactThreshHPK}(\Sigma, \lambda, H)$ *is existentially unforgeable under chosen-message attack if the underlying threshold signature scheme* Σ *is and if* H *is 2nd pre-image resistant.*

Proof. The proof of the theorem follows exactly from 7 where the 2nd pre-image resistance only applies to a single target.

We now prove the quantum security of the non-generic transformation.

Theorem 15 (Quantum security of $\mathsf{CompactThreshHPK}$).
A threshold signature $\mathsf{CompactThreshHPK}(\Sigma, \lambda, H)$ *is zero-time quantum existentially unforgeable if* H *is quantum one-way, and if the statistical distance between the distribution of* pk *in the public key domain* $(\mathcal{D}, +)$ *and the uniform distribution in* $\mathcal{D}$ *is negligible.*

Proof. Consider an adversary $\mathcal{A}$ for the $\mathsf{TQEUF\text{-}KOA}$ game applied to $\mathsf{CompactThreshHPK}(\Sigma, \lambda, H)$.
We want to construct an adversary $\mathcal{B}$ such that if $\mathcal{A}$ wins $\mathsf{TQEUF\text{-}KOA}$ then $\mathcal{B}$ breaks the one-wayness of $\mathcal{A}$. See Figure 24.

```
QOW(B)
1 : $\rho \leftarrow\$ \mathcal{D}$
2 : $y = H(\rho)$
3 : $\mathcal{B}(y) \to \rho^*$
4 : return $\mathbb{1}_{H(\rho)=H(\rho^*)}$
```

```
B(y)
1 : $x \leftarrow\$ \mathcal{D}$
2 : $\tilde{\mathsf{pk}} \leftarrow (y, x)$
3 : $\mathsf{sk}_i \leftarrow\$ \mathcal{D}_{\mathsf{sk}}$ for $i = 1...n$
4 : $\rho_i \leftarrow\$ \mathcal{D}$ for $i = 1...n$
5 : $\tilde{\mathsf{sk}}_i \leftarrow (\mathsf{sk}_i, \rho_i)$ for $i = 1...n$
6 : $\mathcal{A}(\tilde{\mathsf{pk}}, (\tilde{\mathsf{sk}}_i)_{i \in \mathsf{Corr}}) \to (\tilde{\sigma}^*, m^*)$
7 : $\tilde{\sigma}^* \to (\sigma^*, \rho^*)$
8 : return $\rho^*$
```

Fig. 24: Adversary $\mathcal{B}$ against the QOW game for the Quantum Security of $\mathsf{CompactThreshHPK}$ (Theorem 15).

We argue that the view of $\mathcal{A}$ in this game is similar to the view of $\mathcal{A}$ playing $\mathsf{TQEUF\text{-}KOA}$.
The distribution of (y, x) and that of $\tilde{\mathsf{pk}}$ are similar by the same argument as in the proof of Theorem 2.
Now we argue that the distribution of sk_i and ρ_i given by $\mathcal{B}$ is indistinguishable from the real one by $\mathcal{A}$. Indeed, if Shr_t^n is secure then any set of up to t share is uniformly distributed (see Definition 11).
Therefore if $\mathcal{A}$ wins $\mathsf{TQEUF\text{-}KOA}$, $\mathcal{B}$ wins QOW. Hence,

$$\Pr[\mathsf{TQEUF\text{-}KOA}(\mathcal{A}) \to 1] \leq \Pr[\mathsf{QOW}(\mathcal{B}) \to 1] = \mathsf{negl}(\lambda)$$