Efficient Post-Quantum Secure Deterministic Threshold Wallets from Isogenies

Poulami Das*[1], Andreas Erwig†[2], Michael Meyer‡[3], and Patrick Struck§[4]

[1]CISPA Helmholtz Center for Information Security, Germany
[2]Technical University of Darmstadt, Germany
[3]University of Regensburg, Germany
[4]University of Konstanz, Germany

April 26, 2024

Abstract

Cryptocurrency networks crucially rely on digital signature schemes, which are used as an authentication mechanism for transactions. Unfortunately, most major cryptocurrencies today, including Bitcoin and Ethereum, employ signature schemes that are susceptible to quantum adversaries, i.e., an adversary with access to a quantum computer can forge signatures and thereby spend coins of honest users. In cryptocurrency networks, signature schemes are typically not executed in isolation, but within a so-called cryptographic wallet. In order to achieve security against quantum adversaries, the signature scheme and the cryptographic wallet must withstand quantum attacks.

In this work, we advance the study on post-quantum secure signature and wallet schemes. That is, we provide the first formal model for deterministic threshold wallets and we show a generic post-quantum secure construction from any post-quantum secure threshold signature scheme with rerandomizable keys. We then instantiate our construction from the isogeny-based signature scheme CSI-FiSh and we show that our instantiation significantly improves over prior work.

1 Introduction

Blockchain technology and cryptocurrencies have gained huge popularity over the past decade as they introduced a revolutionary digital payment paradigm that does not rely on a trusted authority such as banks or other financial institutions. The most fundamental building block of virtually all blockchain and cryptocurrency networks are digital signature schemes which are used as an authorization mechanism for transactions. More concretely, when a user Alice with public key pk_A wishes to send a payment of c coins to another user Bob identified by public key pk_B, Alice assembles a transaction of the form "Send c coins from pk_A to

*poulami.das@cispa.de
†andreas.erwig@tu-darmstadt.de
‡michael@random-oracles.org
§patrick.struck@uni-konstanz.de

pk_B" and attaches a valid signature under pk_A to the transaction. That is, the security of a cryptocurrency user's funds solely depends on the security of its signing secret key. The two most widely used signature schemes in the cryptocurrency space are the ECDSA and the Schnorr [Sch90] signature schemes, which are, unfortunately, both insecure in presence of a quantum adversary. More concretely, ECDSA and Schnorr rely on the hardness of computing the discrete logarithm of an element in a cyclic group of prime order, a mathematical problem that is assumed to be computationally hard for classical computers, but that can be solved efficiently by quantum computers using Shor's algorithm [Sho94].

Due to this inherent vulnerability against adversaries with access to quantum computers, several research works (e.g., [ESZ22, TMSM21, EEE20, EZS+19]) and industry projects (e.g., [QRL, Bit, Moc]) have started to investigate so-called post-quantum secure cryptocurrency networks, where all cryptographic building blocks are instantiated such that they can be executed on classical machines but remain secure even against quantum adversaries. Since the most essential building block of cryptocurrency networks is the signature scheme, a first step towards a post-quantum secure network is to replace its classically secure signature scheme, such as ECDSA or Schnorr, by a post-quantum secure variant. However, a simple exchange of the signature scheme is not sufficient to achieve full security against a quantum adversary: in the cryptocurrency space, signature schemes are typically executed within so-called *cryptographic wallets*, which are environments that securely store the signing keys of a user and execute the signing process. Naturally, the signature's secret key is only protected against quantum adversaries if both, the signature scheme itself and the wallet scheme, are post-quantum secure.

Cryptographic Wallets. Over the past decade, many works studied secure cryptographic wallets and proposed different constructions [GS15, LFA20, MPs19, AGKK19, KMOS21]. Eventually, the concept of deterministic wallets [Max11, Wui12] has been established as the de-facto standard, which was first formalized by Das et al. [DFL19]. At a high level, a deterministic wallet maintains a master signing secret/public key pair (msk, mpk) as well as a state St and uses the initial values to deterministically derive session signing keys. That is, a deterministic wallet defines two deterministic algorithms, a secret and a public key derivation algorithm. The secret key derivation algorithm takes as input msk, St and an identity ID, and outputs a session secret key sk_{ID}. The public key derivation algorithm is defined analogously. This deterministic key derivation has the important advantage that the wallet can derive many keys, while only having to store the master key pair and state.

Das et al. [DFL19] formalize deterministic wallets in the so-called hot/cold setting, where the wallet consists of two devices, a hot and a cold wallet. The hot wallet stores the master public key mpk and state St and is connected permanently to the Internet, whereas the cold wallet stores the master secret key msk and state St and only comes online when it has to generate a signature. The assumption is then that the cold wallet cannot get corrupted, since it is mostly offline.

Limitations of Previous Works. While deterministic wallets have been studied extensively in the classical setting [DFL19, DEF+21, FTS+18, YLY+22, ER22], to the best of our knowledge there exist only two prior works in the post-quantum setting [ADE+20, Hu23]. Alkeilani Alkadri et al. [ADE+20] initiated the study of post-quantum secure deterministic wallets in the hot/cold setting. To this end, the authors provided a generic construction of a

deterministic wallet and instantiated the construction with the lattice-based signature scheme qTesla [ABB+20]. Unfortunately, their work has two important drawbacks: (1) due to the relatively large public key and signature sizes of lattice-based signature schemes, the wallet construction of Alkeilani Alkadri et al. [ADE+20], is mainly a feasibility result and not suitable for deployment in practice; and (2) the assumption of incorruptible cold wallets is idealized and might not hold in practice, since a cold wallet can get compromised even if it is not connected to the internet, e.g., by an adversary with access to the cold wallet device. A follow-up work by Hu [Hu23] presents a deterministic wallet construction from the lattice-based signature scheme Falcon [FHK+18]. While the instantiation from Falcon allows for smaller public key and signature sizes, the work of Hu still relies on the assumption of incorruptible cold wallets. A very recent work by Das et al. [DEF+23] investigates deterministic threshold wallets in the classical setting, however without providing a formal model for such threshold wallets.

1.1 Our Contribution

In this work, we address the above shortcomings of previous works, namely we (1) provide an efficient construction of deterministic wallets in the post-quantum setting, (2) formally model deterministic threshold wallets, and (3) give an efficient instantiation of deterministic threshold wallets using the isogeny-based CSI-FiSh signature scheme [BKV19]. In detail, our contribution is as follows:

- In Section 3, we show that CSI-FiSh equipped with rerandomizable keys can instantiate the generic construction of deterministic wallets provided by Alkeilani Alkadri et al. [ADE+20]. Importantly, our instantiation from the isogeny-based signature scheme improves over both prior works [ADE+20, Hu23] w.r.t. public key and signature sizes, the two most important metrics for the practicality of post-quantum secure deterministic wallets.

- In Section 4, we present a model for deterministic wallets that does not rely on the hot/cold setting and therefore avoids the idealized assumption of incorruptible cold wallet devices. In our model, the wallet is thresholdized, i.e., it consists of several devices which are permanently online and which respectively store only a share of the wallet's master signing secret key. That is, in our model the master secret key remains secure as long as at most a certain number of wallet devices get compromised. While the idea for this model has first appeared in a work of Das et al. [DEF+23], we are the first to formalize it. We further show that a deterministic wallet in the threshold setting can be constructed generically from any threshold signature scheme with rerandomizable keys.

- Finally, in Section 6 we provide a concrete instantiation of our generic deterministic threshold wallet construction by translating our isogeny-based signature scheme with rerandomizable keys to a threshold variant.

We would like to hereby point out that the primitive of signature schemes with rerandomizable keys is not limited to wallets. Thus, our construction of a threshold signature scheme with rerandomizable keys might be independent interest, for instance to build sanitizable signature schemes [ACdMT05].

1.2 Related Work

1.2.1 Related Work on Deterministic Wallets.

In the past, many works have studied the concept of cryptographic wallets in general (e.g., [GS15, LFA20, MPs19, AGKK19, KMOS21]) and the concept of deterministic wallets in particular (e.g., [DFL19, DEF+21, FTS+18, YLY+22, ER22]). Most related to our work are the works of Alkeilani Alkadri et al. [ADE+20] and Das et al. [DEF+23]. The former studies the notion of deterministic wallets in the hot/cold setting which remain secure against a quantum adversary. That is, Alkeilani Alkadri et al. [ADE+20] present a formal model of deterministic wallets in the post-quantum setting, where the adversary has access to a quantum computer whereas honest parties have access to classical computers only. The authors present a generic construction of such a wallet scheme from any signature scheme with rerandomizable keys that satisfies certain properties and the authors show a concrete instantiation of their generic construction from the lattice-based signature scheme qTesla [ABB+20]. A follow-up work by Hu [Hu23] showed an instantiation from the lattice-based signature scheme Falcon [FHK+18], which significantly decreased the signature and public key sizes of the wallet scheme. Das et al. [DEF+23], on the other hand, translate the widely used BIP32 standard for hierarchical deterministic wallets [Wui12] to the threshold setting, essentially obtaining deterministic threshold wallets. However, the authors do not formally model threshold wallets and focus only on the classic setting, whereas our work is in the post-quantum setting.

1.2.2 Related Work on Rerandomizable and Threshold Signatures.

The notion of signature schemes with rerandomizable keys has first been introduced by Fleischhacker et al. [FKM+16] and has since been found to be a useful building block for the construction of deterministic wallets [DFL19, DEF+21, ER22]. Threshold signature schemes in the classical setting have been widely studied since several decades (e.g., [GJKR96, Bol03, CGJ+99, LJY14]) and more recently there have been works on threshold signatures in the post-quantum setting (e.g., [DM20, ASY22]). In a very recent work, Das et al. [DEF+23] combined the notions of signatures with rerandomizable keys and threshold signatures to the notion of threshold signatures with rerandomizable keys. In our work, we follow their notion and provide an instantiation in the post-quantum setting. Eaton et al. [ESS21] show that key-blinding—a very similar concept to rerandomizable signatures—can be applied to CSI-FiSh in the context of the Tor network.

Remark 1. *Very recently and independently of our work, Shaw and Dutta [SD23] developed similar results to our first contribution, i.e., instantiated a deterministic wallet using the CSI-FiSh signature scheme. However, our work primarily focuses on extending this to the threshold setting, which is not considered by Shaw and Dutta.*

Remark 2. *Other isogeny-based signature schemes like SQIsign [CSCD+23] might appear as suitable candidates for post-quantum wallets due to their small key and signature sizes. However, the threshold construction in our work relies on the underlying structure of CSI-FiSh based on group actions. in contrast, no threshold version of SQIsign or other isogeny-based signature schemes is known.*

2 Preliminaries

2.1 Notation

We denote the uniform random sampling of a value r from a set S by $r \leftarrow\!\!\$\ S$. For a deterministic algorithm A, we write $a \leftarrow A(x)$ to denote the execution of A on input x that outputs a. Likewise, for a probabilistic algorithm B, we write $b \leftarrow\!\!\$\ B(x)$ to denote the execution of B on input x that outputs b. For an interactive algorithm Π, we write $\langle \Pi(x_1), \cdots, \Pi(x_n) \rangle$ to denote the joint execution of Π by a set of n parties $\{P_1, \cdots, P_n\}$ where each party P_i for $i \in [n]$ uses x_i as input.

2.2 Adversary Model

In this work, we generally assume a semi-honest (or passive[1]) adversary. Similar to a fully malicious adversary, a semi-honest adversary learns all secret values and can observe all internal computation of a corrupted party, however, in contrast to a fully malicious adversary, the semi-honest adversary is restricted to follow the protocol instructions. We note that one can transform a scheme that is secure against a semi-honest adversary into a secure scheme against a fully malicious adversary by adding zero-knowledge proofs of correct behavior for each protocol instruction. In addition, we assume static corruptions throughout this paper, i.e., the adversary has to decide at the beginning of a security game which parties to corrupt. Finally, we consider a quantum adversary in this work, i.e., the adversary is assumed to have access to a quantum computer, through which it can access a random oracle in superposition [BDF+11].

2.3 Small-Range Distributions

In the proof of Theorem 2, we make use of Zhandry's small-range distributions, defined below. For such distributions, the number of potential outputs is small.

Definition 1 (Small-range distributions [Zha12])**.** *Let $\mathcal{X}$, $\mathcal{Y}$ be sets, r be an integer, D be a distribution on $\mathcal{Y}$, P be a random function from $\mathcal{X}$ to $[r]$, and $\vec{y} = (y_1, \ldots, y_r)$ be r samples of D. Define a function $\mathsf{H}\colon \mathcal{X} \to \mathcal{Y}$ by $\mathsf{H}(x) \mapsto y_{P(x)}$. The distribution of H, induced by P and $\vec{y}$, is called a* small-range distribution *with r samples of D.*

Zhandry developed a lemma providing an upper bound of an adversary in distinguishing a random oracle from one that is drawn from a small-range distribution. We recall this result below.

Lemma 1 ([Zha12])**.** *There is a universal constant C such that, for any set $\mathcal{X}$ and $\mathcal{Y}$, distribution D on $\mathcal{Y}$, integer l, and any quantum algorithm $\mathcal{A}$ making q queries to an oracle $\mathsf{H}\colon \mathcal{X} \to \mathcal{Y}$, the following two cases are indistinguishable, except with probability less than $\frac{Cq^3}{l}$:*

- *$\mathsf{H}(x) = y_x$ where $\vec{y}$ is a list of samples of D of size $|\mathcal{X}|$.*
- *H is drawn from the small-range distribution with l samples of D.*

[1] We use the terms "passive" and "semi-honest" interchangeably.

2.4 Signatures (with Rerandomizable Keys)

In the following, we recall the definitions of signature schemes and signature schemes with rerandomizable keys.

Definition 2 (Signature Scheme)**.** *A signature scheme* Sig *is a tuple of algorithms* $\mathsf{Sig} := (\mathtt{KGen}, \mathtt{Sign}, \mathtt{Ver})$ *which are defined as follows:*

$\mathtt{KGen}(1^\lambda)$ *is the key generation algorithm. Its input is a security parameter and its output is a key pair consisting of a secret key* sk *and a public key* pk*.*

$\mathtt{Sign}(sk, m)$ *is the signing algorithm. Its input is a secret key* sk *and a message* m *and its output is a signature* σ*.*

$\mathtt{Ver}(pk, m, \sigma)$ *is the verification algorithm. Its input is a public key* pk*, a message* m*, and a signature* σ*, and its output is a bit.*

Definition 3 (Signature Scheme with Rerandomizable Keys)**.** *A signature scheme with rerandomizable keys* $\mathtt{RSig}$ *consists of a tuple of algorithms* $\mathtt{RSig} := (\mathtt{KGen}, \mathtt{RandSK}, \mathtt{RandPK}, \mathtt{Sign}, \mathtt{Ver})$*, where* $\mathtt{KGen}$*,* $\mathtt{Sign}$*,* $\mathtt{Ver}$ *are defined as for a standard signature scheme (cf. Definition 2). For randomness space* $\mathcal{R}$*,* $(\mathtt{RandSK}, \mathtt{RandPK})$ *are two polynomial-time algorithms such that*

$\mathtt{RandSK}(sk, \rho)$ *is the secret key rerandomization algorithm that takes as input a secret key* sk *and a randomness* $\rho \in \mathcal{R}$ *and outputs a randomized secret key* sk'*.*

$\mathtt{RandPK}(pk, \rho)$ *is the public key rerandomization algorithm that takes as input a public key* pk *and a randomness* $\rho \in \mathcal{R}$ *and outputs a randomized public key* pk'*.*

A signature scheme with rerandomizable keys $\mathtt{RSig}$ *must satisfy the following properties:*

- *Rerandomizability of public keys: For all* $\lambda \in \mathbb{N}$*, all public keys* $(\cdot, pk) \leftarrow \mathtt{KGen}(1^\lambda)$ *and* $\rho \in \mathcal{R}$*, the distributions of* pk' *and* pk'' *are computationally indistinguishable, where* $pk' \leftarrow \mathtt{RandPK}(pk, \rho)$*, and* $pk'' \leftarrow \mathtt{KGen}(1^\lambda)$*.*
- *Correctness under rerandomizable keys: For all* $\lambda \in \mathbb{N}$*, all* $m \in \mathcal{M}$*, all* $(sk, pk) \leftarrow \mathtt{KGen}(1^\lambda)$*, all* $\rho \in \mathcal{R}$*, the rerandomized keys* $sk' \leftarrow \mathtt{RandSK}(sk, \rho)$ *and* $pk \leftarrow \mathtt{RandPK}(pk, \rho)$ *satisfy that* $\Pr[\mathtt{Verify}(pk', m, \mathtt{Sign}(sk', m)) = 1] \geq 1 - \mathrm{negl}(\lambda)$*.*
- *Simulatability: For all* $\lambda \in \mathbb{N}$*, all* $(sk, pk) \leftarrow \mathtt{KGen}(1^\lambda)$*, and all* $m \leftarrow \{0,1\}^*$*, there exists a polynomial-time algorithm* $\mathcal{T}$ *which on input* pk *and* m *outputs a signature* σ*. It must hold that for* $\kappa \in \mathrm{poly}(\lambda)$ *the distributions* $\{\sigma_1, \cdots, \sigma_\kappa\}$ *and* $\{\sigma'_1, \cdots, \sigma'_\kappa\}$ *are computationally indistinguishable where* $\sigma_i \leftarrow \mathcal{T}(pk, m)$ *and* $\sigma'_i \leftarrow \mathtt{RSig.Sign}(sk, m)$ *for* $i \in [\kappa]$*.*

Definition 4 (Unforgeability of signature schemes with honestly rerandomizable keys)**.** *A signature scheme with rerandomizable keys* $\mathtt{RSig}$ *is unforgeable under honestly rerandomizable keys if no efficient adversary* $\mathcal{A}$ *wins game* EUF-CMA-HRK *as described below with non-negligible probability in the security parameter* λ*.*

Game EUF-CMA-HRK*:*

- *The game initializes two lists* $\mathcal{S} \leftarrow \emptyset$ *and* $\mathcal{L} \leftarrow \emptyset$*, and executes* $(sk, pk) \leftarrow \mathtt{RSig.KGen}(1^\lambda)$*. Then* $\mathcal{A}$ *is run on input* pk*.*

- *The adversary obtains access to the following oracles:*
  - Rand*: Upon a query, this oracle samples a fresh random value from* $\mathcal{R}$ *as* $\rho \leftarrow\!\!\$\ \mathcal{R}$*, stores* ρ *in* $\mathcal{L}$*, and returns* ρ*.*
  - Sign*: On input* $(m\rho)$*, the oracle checks whether* $\rho \notin \mathcal{L}$ *and if so outputs* $\perp$*. Otherwise, it derives a public* $pk' \leftarrow \mathtt{RSig.RandPK}(pk, \rho)$ *key and a secret key* $sk' \leftarrow \mathtt{RSig.RandSK}(sk, \rho)$ *and executes* $\sigma \leftarrow \mathtt{RSig.Sign}(sk', m)$*. The oracle then stores the tuple* (pk', m) *in* $\mathcal{S}$*.*
- *The adversary outputs a forgery* (σ^*, m^*, ρ^*)*. The adversary wins the game, if, for* $pk^* \leftarrow \mathtt{RSig.RandPK}(pk, \rho^*)$*, it holds that: (1)* $\rho^* \in \mathcal{L}$*, (2)* $(pk^*, m^*) \notin \mathcal{S}$*, and (3)* $\mathtt{RSig.Ver}(pk^*, m^*, \sigma^*) = 1$*.*

The default security notion EUF-CMA for a signature scheme follows by removing the oracle Rand as well as all usage of the rerandomization algorithms.

2.5 Interactive Threshold Signatures (with Rerandomizable Keys)

In the following, we recall the notions of interactive threshold signatures (with rerandomizable keys) as described by Das et al. [DEF+23].

Definition 5 (Interactive Threshold Signature Scheme)**.** *An interactive* (t, n)*-threshold signature scheme* $\mathtt{TSig}$ *is executed among a set of* n *parties* $\{P_1, \cdots, P_n\}$ *and consists of a tuple of procedures* $\mathtt{TSig} := (\mathtt{KGen}, \mathtt{TSign}, \mathtt{Ver})$ *which are defined as follows:*

$\mathtt{KGen}(1^\lambda, t, n)$**:** *The probabilistic key generation algorithm takes as input a security parameter* λ *and two integers* $t, n \in \mathbb{N}$ *such that* $t < n$*. It outputs a public key* pk *and a set of secret key shares* $\{sk_1, \ldots, sk_n\}$ *such that each party* P_i *obtains* pk *and* sk_i*.*

$\mathtt{TSign}(sk_i, m)$**:** *The probabilistic interactive signing algorithm takes as input a secret key share* sk_i *for* $i \in [n]$ *and a message* m*. It outputs either a signature* σ *or* $\perp$*.*

$\mathtt{Ver}(pk, m, \sigma)$**:** *The deterministic verification algorithm takes as input a public key* pk*, a message* m *and a signature* σ *and outputs a bit* $0/1$*.*

Definition 6 (Interactive Threshold Signature Scheme with Rerandomizable Keys)**.** *An interactive* (t, n)*-threshold signature scheme with rerandomizable keys is a tuple of procedures* $\mathtt{TRSig} = (\mathtt{KGen}, \mathtt{RandSK}, \mathtt{RandPK}, \mathtt{TSign}, \mathtt{Ver})$ *where* $(\mathtt{KGen}, \mathtt{TSign}, \mathtt{Ver})$ *are defined as for interactive* (t, n)*-threshold signatures. We assume that the public parameters* pp *define a randomness space* $\mathcal{R} := \mathcal{R}(pp)$*. The algorithms* $\mathtt{RandSK}$ *and* $\mathtt{RandPK}$ *are defined as:*

$\mathtt{RandSK}(i, sk_i, \rho)$**:** *The deterministic secret key share rerandomization algorithm takes as input an index* $i \in [n]$*, a secret key share* sk_i *and a randomness* $\rho \in \mathcal{R}$ *and it outputs a rerandomized secret key share* sk_i'*.*

$\mathtt{RandPK}(pk, \rho)$**:** *The deterministic public key rerandomization algorithm takes as input a public key* pk *and a randomness* $\rho \in \mathcal{R}$ *and it outputs a rerandomized public key* pk'*.*

We require the following properties of a threshold signature scheme with rerandomizable keys:

- *Rerandomizability of public keys: For all $\lambda \in \mathbb{N}$, all $t, n \in \mathbb{N}$ with $t < n$, all $(\cdot, pk) \leftarrow \mathtt{KGen}(1^\lambda, t, n)$ and all $\rho \leftarrow\!\!\$\ \mathcal{R}$, the distributions of pk' and pk'' are computationally indistinguishable, where $pk' \leftarrow \mathtt{RandPK}(pk, \rho)$ and $(\cdot, pk'') \leftarrow \mathtt{KGen}(1^\lambda)$.*

- *Correctness under rerandomized keys: For all $\lambda, t, n \in \mathbb{N}$, with $t < n$, all $(\{sk_1, \ldots, sk_n\}, pk) \leftarrow \mathtt{KGen}(1^\lambda, t, n)$, all $\rho \leftarrow\!\!\$\ \mathcal{R}$, and all $m \in \mathcal{M}$, the rerandomized keys $\{sk'_i\}_{i\in[n]} \leftarrow \{\mathtt{RandSK}(i, sk_i, \rho)\}_{i\in[n]}$ and $pk' \leftarrow \mathtt{RandPK}(pk, \rho)$ satisfy:*

$$\Pr[\mathtt{Ver}(pk', m, \sigma) | \sigma \leftarrow \langle \mathtt{TSign}(sk'_1, m), \ldots, \mathtt{TSign}(sk'_n, m) \rangle] = 1.$$

- *Simulatability: For all $\lambda, t, n \in \mathbb{N}$, with $t < n$, all $(\{sk_1, \ldots, sk_n\}, pk) \leftarrow \mathtt{KGen}(1^\lambda, t, n)$, and all $m \in \{0,1\}^*$, there exists a polynomial-time algorithm $\mathcal{T}$ which on input pk and m outputs a signature σ. It must hold that for $\kappa \in \mathrm{poly}(\lambda)$ the distributions $\{\sigma_1, \ldots, \sigma_\kappa\}$ and $\{\sigma'_1, \ldots, \sigma'_\kappa\}$ are computationally indistinguishable where*

$$\sigma'_i \leftarrow \langle \mathtt{TSign}(sk_1, m), \ldots, \mathtt{TSign}(sk_n, m) \rangle \text{ and } \sigma_i \leftarrow \mathcal{T}(pk, m) \text{ for } i \in [\kappa].$$

Both of the above primitives must satisfy a security notion of unforgeability. For interactive threshold signature scheme, this notion is defined similarly to the unforgeability of a standard digital signature scheme with the differences that (1) the adversary is allowed to corrupt up to t parties in the beginning of the game and (2) the signing oracle is interactive and allows the adversary to learn the transcript of the signing protocol execution as well as all internally computed messages for corrupted parties.

For interactive threshold signature schemes with rerandomizable keys, Das et al. [DEF+23] define the notion of *unforgeability under honestly rerandomizable keys*, which we follow in this work (though Das et al. define the notion for a malicious adversary, whereas our definition is w.r.t. a semi-honest adversary). The main difference to the unforgeability notion of interactive threshold signature schemes is that the adversary can query the signing oracle on honestly rerandomized keys and win the game even with a valid forgery under an honestly rerandomized key. We denote the game by T-EUF-CMA-HRK and formally describe it below.

Definition 7 (Unforgeability of interactive threshold signature schemes with honestly rerandomizable keys)**.** *An interactive (t, n)-threshold signature scheme $\mathtt{TRSig}$ is unforgeable under honestly rerandomizable keys if no efficient adversary $\mathcal{A}$ wins game* T-EUF-CMA-HRK *as described below with more than negligible probability in the security parameter λ.*

Game T-EUF-CMA-HRK*:*

- *The adversary $\mathcal{A}$ outputs a list of corrupted parties $\mathcal{C}$, such that $|\mathcal{C}| \leq t$ and for all $i \in \mathcal{C}$ it holds that $i \in [n]$.*

- *The game computes $(\{sk_1, \ldots, sk_n\}, pk) \leftarrow \mathtt{TRSig.KGen}(1^\lambda, t, n)$ and initializes two lists $\mathcal{S} \leftarrow \{\epsilon\}$ and $\mathcal{L} \leftarrow \{\epsilon\}$. Then $\mathcal{A}$ is run on input pk and $\{sk_i\}_{i\in\mathcal{C}}$.*

- *The adversary obtains access to the following oracles:*

  - Rand*: Upon a query, this oracle samples a fresh random value from $\mathcal{R}$ as $\rho \leftarrow\!\!\$\ \mathcal{R}$, stores ρ in $\mathcal{L}$, and returns ρ.*

– Sign: *On input message m and a randomness ρ, the oracle checks whether $\rho \notin \mathcal{L}$ and if so outputs $\perp$. Otherwise, it derives a public key and secret key shares for honest parties with the randomness ρ, i.e., it computes $pk' \leftarrow \texttt{TRSig.RandPK}(pk, \rho)$ and $sk'_i \leftarrow \texttt{TRSig.RandSK}(i, sk_i, \rho)$ for all $i \in \{1, \cdots, n\} \setminus \mathcal{C}$. The oracle and the adversary then jointly execute the procedure* TRSig.TSign*, where the oracle runs all honest parties P_i on input (sk'_i, m). The oracle then stores the tuple (pk', m) in $\mathcal{S}$.*

- *The adversary outputs a forgery (σ^*, m^*, ρ^*). The adversary wins the game, if the following conditions hold:*
  1. $\rho^* \in \mathcal{L}$
  2. $(\rho^*, m^*) \notin \mathcal{S}$
  3. $\texttt{TRSig.Ver}(pk^*, m^*, \sigma^*) = 1$, *for* $pk^* \leftarrow \texttt{TRSig.RandPK}(pk, \rho^*)$.

Just as for EUF-CMA, the default security notion T-EUF-CMA for interactive threshold signatures can be restored by removing the oracle Rand and the usage of the rerandomization algorithms.

2.6 Isogeny Group Action

Let $p = 4 \cdot \prod_{i=1}^{n} \ell_i - 1$ be prime with small distinct odd primes ℓ_i. We write $\mathcal{E}$ for the set of supersingular elliptic curves E over $\mathbb{F}_p$, whose $\mathbb{F}_p$-rational endomorphism ring $\mathrm{End}_{\mathbb{F}_p}(E)$ is isomorphic to an order $\mathcal{O} \subset \mathbb{Q}(\sqrt{-p})$. The class group $\mathrm{cl}(\mathcal{O})$ of $\mathcal{O}$, given as the quotient of the fractional invertible ideals and the principal ideals of $\mathcal{O}$, induces a free and transitive group action $\mathrm{cl}(\mathcal{O}) \times \mathcal{E} \to \mathcal{E}$. For an ideal $\mathfrak{a} \in \mathrm{cl}(\mathcal{O})$ and a curve $E \in \mathcal{E}$, we write this group action as $\mathfrak{a} * E = E'$. This action can be expressed via isogenies: For any invertible ideal $\mathfrak{a}$ and any curve $E \in \mathcal{E}$, we can define the subgroup $E[\mathfrak{a}] = \bigcap_{\alpha \in \mathfrak{a}} \ker(\alpha)$, which uniquely (up to isomorphism) determines an isogeny $\varphi : E \to E'$ for some $E' \in \mathcal{E}$.

In order to efficiently evaluate this class group action, CSIDH usually fixes ideals $\mathfrak{l}_i = \langle \ell_i, \pi - 1 \rangle$ resp. $\mathfrak{l}_i^{-1} = \langle \ell_i, \pi + 1 \rangle$ with Frobenius endomorphism π, whose action is efficiently computable via an ℓ_i-isogeny. Hence, we can represent ideals as products $\mathfrak{a} = \prod_{i=1}^{n} \mathfrak{l}_i^{e_i}$, where the exponents e_i lie in a small range $[-B_i, B_i]$, and compute their action through a sequence of ℓ_i-isogenies. Assuming that different exponent vectors $(e_1, \ldots, e_n)$ for small bounds B_i only rarely represent the same ideal class, we can heuristically sample from the full class group by picking B_i such that the number of exponent vectors is $\prod_{i=1}^{n}(2B_i + 1) \approx \sqrt{p} \approx \#\mathrm{cl}(\mathcal{O})$. CSIDH uses this heuristic for sampling private keys as such exponent vectors, since uniform sampling would require knowledge of the class group structure and the efficient evaluation of the action of any ideal, both of which are computationally expensive problems in general.

Although this restriction translates to several complications for building CSIDH-based signature schemes, CSI-FiSh [BKV19] solves these problems for the CSIDH-512 parameter set using a 511-bit prime p. In particular, Beullens et al. computed $N = \#\mathrm{cl}(\mathcal{O})$ and the class group structure, and found that $\mathfrak{l}_1$ is a generator of $\mathrm{cl}(\mathcal{O})$ in this case. Hence, we can uniformly sample an integer $s \in \mathbb{Z}/N\mathbb{Z}$ as private key, corresponding to the ideal class of $\mathfrak{l}_1^s$. Following this, the corresponding exponent vector $(s, 0, \ldots, 0)$ can be transformed to an equivalent exponent vector of small norm, such that its action is efficiently computable.

Due to this embedding $\mathbb{Z}/N\mathbb{Z} \hookrightarrow \mathrm{cl}(\mathcal{O})$ given by $a \mapsto \mathfrak{g}^a$, we will write $[a]$ for $\mathfrak{g}^a$ with a fixed generator $\mathfrak{g}$ of $\mathrm{cl}(\mathcal{O})$ of order N, and denote the group action $\mathfrak{g}^a * E$ by $[a]E$. Note that in this notation we have $[a][b]E = [a + b]E$.

2.7 CSI-FiSh Signature Scheme

The CSI-FiSh signature scheme is based on a simple identification scheme. The prover possesses a secret $s \in \mathbb{Z}/N\mathbb{Z}$ and a corresponding public key $E_1 = [s]E_0$ for an initially chosen starting curve $E_0 \in \mathcal{E}$. The prover proves knowledge of s to a verifier as follows: (1) The prover samples $b \in \mathbb{Z}/N\mathbb{Z}$ and commits to $E_b = [b]E_0$; (2) The verifier sends a random challenge bit $c \in \{0, 1\}$; (3) The prover responds with $r = b - cs$; (4) The verifier accepts if $[r]E_c = E_b$.

The success probability for a prover to succeed without knowledge of s is $\frac{1}{2}$, and thus for security parameter λ, we have to repeat this scheme λ times. Using λ rounds and the Fiat-Shamir transform then leads to a signature scheme. To sign a message m, we sample $b_1, \ldots, b_\lambda \in \mathbb{Z}/N\mathbb{Z}$ and compute the commitment curves $E^{(i)} = [b_i]E_0$. We set $(c_1, \ldots, c_\lambda) = H(E^{(1)}, \ldots, E^{(\lambda)}, m)$ with a cryptographic hash function H and $c_i \in \{0, 1\}$, and compute the responses $r_i = b_i - c_i s$. This results in a signature $\sigma = (r_1, \ldots, r_\lambda, c_1, \ldots, c_\lambda)$. The verification proceeds similarly to the procedure above. On input $\sigma = (r_1, \ldots, r_\lambda, c_1, \ldots, c_\lambda)$, we recompute $E^{(1)}, \ldots, E^{(\lambda)}$ through $E^{(i)} = [r_i]E_{c_i}$. We then accept if $(c_1, \ldots, c_\lambda) = H(E^{(1)}, \ldots, E^{(\lambda)}, m)$, and reject otherwise.

Remark 3. *We can decrease the number of required rounds by picking k secrets $s^{(1)}, \ldots, s^{(k)}$ and corresponding public keys $E_i = [s^{(i)}]E_0$. The verifier can then pick challenges from $\{0, \ldots, k\}$, and the prover answers with $r = b - s^{(c)}$ if $c > 0$, or $r = b$ otherwise. This decreases the soundness error from $1/2$ to $1/(k+1)$, and leads to a tradeoff between increased public key size and smaller signature size due to the fewer rounds [DG19, BKV19].*

Remark 4. *In addition, when using the usual CSIDH starting curve E_0, we can double the challenge space by exploiting the fact that $[s]E$ is the quadratic twist of $[-s]E$. In particular, we can use the challenge space $c_i \in \{-k, \ldots, k\}$ and compute responses as $r_i = b_i - \mathtt{Sign}(c_i) \cdot s^{(|c_i|)}$. Using this trick, public key sizes reduce by factor 2, see [BKV19].*

3 Rerandomization of CSI-FiSh Keys

In this section, we present an instantiation of a signature scheme with rerandomizable keys (cf. Definition 3) from the CSI-FiSh signature scheme. Consider the basic CSI-FiSh scheme with a single secret $s \in \mathbb{Z}/N\mathbb{Z}$ and public key $E_1 = [s]E_0$. We can rerandomize this key pair by setting $s' = s + \rho$ and $E_1' = [\rho]E_1 = [\rho][s]E_0 = [s + \rho]E_0$ for a random $\rho \in \mathbb{Z}/N\mathbb{Z}$, as depicted in Fig. 1.

The theorem below shows that security of CSI-FiSh implies security of the corresponding signature scheme with rerandomizable keys. The proof follows from Theorem 4. which covers the more general setting of the threshold version of the scheme.

Theorem 1. *Let Σ be the CSI-FiSh signature scheme (cf. Fig. 1). Let further be Σ' be the rerandomizable signature scheme from Fig. 1. Assuming that Σ is unforgeable, Σ' is unforgeable against honestly rerandomized keys.*

It is easy to see that the construction achieves the property of *rerandomizability of public keys*. The *simulatability* property follows from the simulatability of CSI-FiSh signatures. Having established Theorem 1, we can use CSI-FiSh with reandomizable keys to instantiate the generic wallet construction given in [ADE+20]. This gives an alternative instantiation than the lattice-based presented in [ADE+20]. While isogeny-based signatures are generally slower

$\texttt{KGen}()$

$s \leftarrow_\$ \mathbb{Z}/N\mathbb{Z}$
$E_1 \leftarrow [s]E_0$
return $(pk, sk) \leftarrow (E_1, s)$

$\texttt{RandSK}(sk, \rho)$

parse sk **as** s
$s' \leftarrow s + \rho$
return s'

$\texttt{RandPK}(pk, \rho)$

parse pk **as** E_1
$E_{1,\rho} \leftarrow [\rho]E_1$
return $E_{1,\rho}$

$\texttt{Sign}(sk, m)$

parse sk **as** s
$b_1, \ldots, b_\lambda \leftarrow_\$ \mathbb{Z}/N\mathbb{Z}$
for $i = 1..\lambda$
  $E^{(i)} \leftarrow [b_i]E_0$
$(c_1, \ldots, c_\lambda) \leftarrow H(E^{(1)}, \ldots, E^{(\lambda)}, m)$
$r_i \leftarrow b_i - c_i s$
$\sigma \leftarrow (r_1, \ldots, r_\lambda, c_1, \ldots, c_\lambda)$
return σ

$\texttt{Ver}(pk, m, \sigma)$

parse pk **as** E_1
parse σ **as** $(r_1, \ldots, r_\lambda, c_1, \ldots, c_\lambda)$
for $i = 1..\lambda$
  $E^{(i)} \leftarrow [r_i]E_{c_i}$
if $(c_1, \ldots, c_\lambda) \neq H(E^{(1)}, \ldots, E^{(\lambda)}, m)$
  return 0
return 1

Figure 1: The signature scheme CSI-FiSh. Note that we give the scheme with the secret key containing a single secret s as opposed to multiple secrets described in the remark above. Adding the two algorithms `RandSK` and `RandPK` gives the signature scheme CSI-FiSh with rerandomizable keys.

in computation than the lattice-based signatures considered in [ADE+20], they come with the advantage of significantly smaller key and signature sizes. For blockchain applications such as deterministic wallets, the bottleneck is typically size rather than computation time and hence, isogeny-based signatures seem better suited for blockchains than lattice-based signatures. We provide a more detailed comparison in Section 6.

4 Deterministic Threshold Wallets

In this section, we first present our formal model of a deterministic threshold wallet scheme. We then show a generic construction from an interactive threshold signature scheme with rerandomizable keys and prove our construction secure.

4.1 Model

In the following, we provide a formal model of deterministic threshold wallets. The essential difference of a deterministic threshold wallet to an interactive threshold signature scheme with rerandomizable keys is that the initial key generation algorithm outputs a master key $(mpk, \{msk_1, \ldots, msk_n\})$ *as well as a state* St. This state is used during the rerandomization of the master public key and secret key shares to deterministically derive the respective randomness. This essentially allows the wallet to derive all randomness deterministically. We now provide the formal definition.

Definition 8 (Deterministic Threshold Wallet)**.** *A* (t, n)*-deterministic threshold wallet scheme* `DTW` *is executed among a set of* n *parties* $\{P_1, \ldots, P_n\}$ *and consists of a tuple of procedures*

$\mathtt{DTW} := (\mathtt{DTW.KGen}, \mathtt{DTW.RandSK}, \mathtt{DTW.RandPK}, \mathtt{DTW.TSign}, \mathtt{DTW.Verify})$, *which are defined as follows:*

$\mathtt{DTW.KGen}(1^\lambda, t, n)$**:** *The master key generation algorithm takes as input a security parameter λ, as well as two integers $t, n \in \mathbb{N}$ such that $t < n$. It outputs a master public key mpk, master secret key shares $\{msk_1, \ldots, msk_n\}$ as well as an initial state St such that each party P_i obtains mpk, msk_i, and St.*

$\mathtt{DTW.RandSK}(i, msk_i, St, ID)$**:** *The deterministic secret key derivation algorithm takes as input an index $i \in [n]$, a master secret key share msk_i, a state St, and an identity ID. It outputs a session secret key share sk_i^{ID}.*

$\mathtt{DTW.RandPK}(mpk, St, ID)$**:** *The deterministic public key derivation algorithm takes as input a master public key mpk, a state St, and an identity ID. It outputs a session public key pk^{ID}.*

$\mathtt{DTW.TSign}(sk_i^{ID}, m)$**:** *The probabilistic interactive signing algorithm takes as inputs a session secret key share sk_i^{ID} for $i \in [n]$, and a message m. It outputs a signature σ or $\perp$.*

$\mathtt{DTW.Verify}(pk^{ID}, m, \sigma)$**:** *The deterministic verification algorithm takes as input a session public key pk^{ID}, a message m, and a signature σ. It outputs 1 if σ is a valid signature for m under public key pk^{ID} and 0 otherwise.*

Definition 9 (Correctness of Deterministic Threshold Wallets)**.** *For all $t, n \in \mathbb{N}$ with $t < n$, all $(mpk, \{msk_1, \ldots, msk_n\}, St) \leftarrow\!\!\$\ \mathtt{DTW.KGen}(\lambda, t, n)$, and all $ID \in \{0,1\}^*$, we define pk^{ID} and sk_i^{ID} for $1 \leq i \leq n$ as*

$$sk_i^{ID} := \mathtt{DTW.RandSK}(i, msk_i, ID, St), \;\; pk^{ID} := \mathtt{DTW.RandPK}(mpk, ID, St).$$

$\mathtt{DTW}$ is correct if for all $\lambda, t, n \in \mathbb{N}$ with $t < n$, all $ID \in \{0,1\}^*$, all $m \in \mathcal{M}$, and all $(mpk, \{msk_1, \ldots, msk_n\}, St) \leftarrow\!\!\$\ \mathtt{DTW.KGen}(\lambda, t, n)$, it holds that

$$\Pr[\mathtt{DTW.Verify}(pk^{ID}, m, \sigma) = 1] = 1$$

where $\sigma \leftarrow\!\!\$\ \left\langle \mathtt{DTW.TSign}(sk_i^{ID}, m), \ldots, \mathtt{DTW.TSign}(sk_n^{ID}, m) \right\rangle$.

A deterministic threshold wallet must satisfy two security notions, namely unforgeability and unlinkability. The former is defined similarly to the corresponding notion of an interactive threshold signature scheme with rerandomizable keys and we formally define this property in Definition 11. The later is formally defined in Definition 10. Essentially, unlinkability guarantees that an adversary upon seeing a public key pk cannot distinguish whether pk has been derived from mpk or from an independently generated master public key. The idea behind this notion is that parties in a cryptocurrency network cannot link several payments to the same wallet. Importantly, in the threshold setting, unlinkability can only hold as long as no party is corrupted. As soon as a single party in the wallet scheme is corrupted, the adversary learns the state of the scheme and can derive session public keys itself.

Definition 10 (Unlinkability)**.** *A (t, n)-deterministic threshold wallet scheme $\mathtt{DTW}$ is unlikable if no efficient adversary $\mathcal{A}$ wins game WUNL as described below with more than negligible probability in the security parameter λ.*

Game WUNL*:*

- *The game computes* $(mpk, \{msk_1, \cdots, msk_n\}, St) \leftarrow\!\!\$\ \mathtt{DTW.KGen}(1^\lambda, t, n)$ *and initializes a list* $\mathcal{K} \leftarrow \{\epsilon\}$*. Then* $\mathcal{A}$ *is run on input* mpk*.*
- *The adversary obtains access to the following oracles:*
  - PK*: On input an identity* ID*, the oracle computes*

$$pk^{ID} \leftarrow \mathtt{DTW.RandPK}(mpk, ID, St)$$
$$sk_i^{ID} \leftarrow \mathtt{DTW.RandSK}(i, msk_i, ID, St)$$

    for all $i \in [n]$*. The oracle then sets* $\mathcal{K}[ID] \leftarrow (pk^{ID}, \{sk_i^{ID}\}_{i\in[n]})$ *and outputs* pk^{ID}*.*
  - Sign*: On input message* m *and an identity* ID*, the oracle aborts if* $\mathcal{K}[ID] = \bot$*. Otherwise, the oracle fetches* $(pk^{ID}, \{sk_i^{ID}\}_{i\in[n]}) \leftarrow \mathcal{K}[ID]$ *and executes*

$$\sigma \leftarrow\!\!\$\ \left\langle \mathtt{DTW.TSign}(sk_i^{ID}, m), \cdots, \mathtt{DTW.TSign}(sk_n^{ID}, m) \right\rangle.$$

    The oracle outputs σ*.*
- *Eventually, the adversary outputs an identity* ID^**. The game aborts if* $\mathcal{K}[ID^*] \neq \bot$*. Otherwise, the game samples a bit* $b \leftarrow\!\!\$\ \{0,1\}$ *and proceeds as follows:*
  - *If* $b = 0$*: Compute* $pk_0^{ID^*} \leftarrow \mathtt{DTW.RandPK}(mpk, ID^*, St)$ *and* $sk_{i,0}^{ID^*} \leftarrow \mathtt{DTW.RandSK}(i, msk_i, ID^*, St)$ *for* $i \in [n]$ *and set* $\mathcal{K}[ID^*] \leftarrow (pk_0^{ID^*}, \{sk_{i,0}^{ID^*}\}_{i\in[n]})$*.*
  - *If* $b = 1$*: Sample* $(\widetilde{mpk}, \{\widetilde{msk}_1, \cdots, \widetilde{msk}_n\}, \widetilde{St}) \leftarrow \mathtt{DTW.KGen}(1^\lambda, t, n)$ *and compute*

$$pk_1^{ID^*} \leftarrow \mathtt{DTW.RandPK}(\widetilde{mpk}, ID^*, \widetilde{St})$$
$$sk_{i,1}^{ID^*} \leftarrow \mathtt{DTW.RandSK}(i, \widetilde{msk}_i, ID^*, \widetilde{St})$$

    for $i \in [n]$*. Finally, set* $\mathcal{K}[ID^*] \leftarrow (pk_1^{ID^*}, \{sk_{i,1}^{ID^*}\}_{i\in[n]})$*.*
- *The game outputs public key* $pk_b^{ID^*}$ *to* $\mathcal{A}$*. The adversary then obtains access to oracles* PK *and* Sign *as described above and eventually outputs a bit* b'*.*
- *The adversary wins the game if* $b = b'$*.*

Definition 11 (Unforgeability)**.** *A* (t, n)*-deterministic threshold wallet scheme* DTW *is unforgeable if no efficient adversary* $\mathcal{A}$ *wins game* WUF *as described below with more than negligible probability in the security parameter* λ*.*

Game WUF*:*

- *The adversary* $\mathcal{A}$ *outputs a list of corrupted parties* $\mathcal{C}$*, such that* $|\mathcal{C}| \leq t$ *and for all* $i \in \mathcal{C}$ *it holds that* $i \in [n]$*.*
- *The game computes* $(mpk, \{msk_1, \cdots, msk_n\}, St) \leftarrow\!\!\$\ \mathtt{DTW.KGen}(1^\lambda, t, n)$ *and initializes a list* $\mathcal{S} \leftarrow \emptyset$*. Then* $\mathcal{A}$ *is run on input* mpk*,* $\{msk_i\}_{i\in\mathcal{C}}$*, and* St*.*

- *The adversary obtains access to the following signing oracle* Sign*: On input message m and an identity ID, the oracle derives the secret key shares $sk_i^{ID} \leftarrow \mathtt{DTW.RandSK}(i, msk_i, ID, St)$ for parties P_i with $i \in \{1, \ldots, n\} \setminus \mathcal{C}$ and the public key $pk^{ID} \leftarrow \mathtt{DTW.RandPK}(mpk, ID, St)$. The oracle and the adversary then jointly execute the procedure* DTW.TSign*, where the oracle runs all honest parties P_i on input (sk_i^{ID}, m). The oracle then stores the tuple (pk^{ID}, m) in $\mathcal{S}$.*

- *Eventually, the adversary outputs a forgery σ^*, a message m^* and an identity ID^*. The game computes*

$$pk^{ID^*} \leftarrow \mathtt{DTW.RandPK}(mpk, ID^*, St)$$

and the adversary wins the game, if the following conditions hold: (1) $(pk^{ID^*}, m^*) \notin \mathcal{S}$, and (2) $\mathtt{DTW.Verify}(pk^{ID^*}, m^*, \sigma^*) = 1$.

4.2 Construction

We now provide our generic construction of a deterministic threshold wallet scheme from any interactive threshold signature scheme with rerandomizable keys TRSig. The description of our construction appears in Figure 2.

$\mathtt{DTW.KGen}(\lambda, t, n)$

$(mpk, \{msk_1, \cdots, msk_n\}) \leftarrow\!\!\$\ \mathtt{TRSig.KGen}(1^\lambda, t, n)$
$St \leftarrow\!\!\$\ \{0,1\}^\lambda$
return $(mpk, \{msk_1, \cdots, msk_n\}, St)$

$\mathtt{DTW.RandSK}(i, msk_i, St, ID)$

$\rho \leftarrow \mathsf{H}(St, ID)$
return $\mathtt{TRSig.RandSK}(i, msk_i, \rho)$

$\mathtt{DTW.RandPK}(mpk, St, ID)$

$\rho \leftarrow \mathsf{H}(St, ID)$
return $\mathtt{TRSig.RandPK}(mpk, \rho)$

$\mathtt{DTW.TSign}(sk_i^{ID}, m)$

return $\mathtt{TRSig.TSign}(sk_i^{ID}, m)$

$\mathtt{DTW.Verify}(pk^{ID}, m, \sigma)$

return $\mathtt{TRSig.Ver}(pk^{ID}, m, \sigma)$

Figure 2: Generic construction of a threshold deterministic wallet from any threshold signature scheme with rerandomizable keys TRSig.

4.3 Security

In this section we state the security properties of threshold wallet unlinkability and unforgeability in the respective Theorems 3 and 2 for our deterministic threshold wallet construction from Fig. 2. The proof for the former follows essentially from [ADE+20] as the threshold property affects only the secret key and signing algorithm. While the proof idea of the later, although similar to [ADE+20], contains subtle changes dealing with the threshold setting and the changes in the construction, e.g., keeping a fixed, randomly chosen state rather than updating it. We present the full proof below.

Theorem 2. *Let* DTW *be the deterministic threshold wallet scheme according to Fig. 2. Assume* TRSig *is an interactive threshold signature scheme with rerandomizable public keys according*

to Definition 3 that is unforgeable according to Definition 7, and H *is a (quantum) random oracle, then* $\mathtt{DTW}$ *is unforgeable (cf. Definition 11).*

Proof. Let game G_0 be WUF instantiated with the generic construction from Fig. 2. Assume, for sake of contradiction, that there is an efficient quantum adversary $\mathcal{A}$ which has non-negligible advantage ϵ in game WUF, i.e., there exists a polynomial $p = p(\lambda)$ such that $p(\lambda) > \frac{1}{\epsilon(\lambda)}$ and $\Pr[\mathcal{A} \text{ wins } \mathsf{G}_0] = \epsilon$.

Now consider game G_1, where the random oracle H is sampled according to a small-range distribution. More precisely, let C be the constant from Lemma 1, q being the random oracle queries by $\mathcal{A}$, and p be the polynomial described above, then G_1 samples H from a small-range distribution using $l = 2Cpq^3$ samples. By Lemma 1, $\mathcal{A}$ can distinguish between the random oracles (and hence G_0 and G_1) with probability at most $\frac{Cq^3}{l} = \frac{Cq^3}{2Cq^3p} = \frac{1}{2p}$. Thus $\Pr[\mathcal{A} \text{ wins } \mathsf{G}_1] = \epsilon - \frac{1}{2p}$.

As the final step, we show—assuming the underlying interactive threshold signature scheme with rerandomizable keys $\mathtt{TRSig}$ to be unforgeable as per Definition 7—that the advantage of $\mathcal{A}$ in winning G_1 is negligible, thus yielding a contradiction. For this we construct a reduction $\mathcal{B}$ against $\mathtt{TRSig}$. First $\mathcal{B}$ samples a random state St. Then it runs $\mathcal{A}$ who will provide a set of corrupted parties $\mathcal{C}$ with $|\mathcal{C}| < t$. $\mathcal{B}$ will use the same set of corrupted parties in its game $\mathsf{T\text{-}EUF\text{-}CMA\text{-}HRK}$ and receives the public key pk along with the corrupted secret key shares $\{msk_i\}_{i\in\mathcal{C}}$ which, together with the randomly chosen state St, $\mathcal{B}$ gives to $\mathcal{A}$. Before the first query of $\mathcal{A}$, $\mathcal{B}$ makes $l = 2Cpq^3$ to its oracle Rand. From the l responses, $\mathcal{B}$ will sample a random oracle H drawn from a small-range distribution. For every query (m, ID) to Sign by $\mathcal{A}$, $\mathcal{B}$ proceeds as follows: It computes $\rho = \mathsf{H}(St, ID)$ and jointly executes the procedure $\mathtt{DTW.TSign}$ with its own signing oracle in game $\mathsf{T\text{-}EUF\text{-}CMA\text{-}HRK}$, where $\mathcal{B}$ lets $\mathcal{A}$ control the corrupted parties. Note that every ρ that $\mathcal{B}$ computes results in a response from Rand to the l queries that $\mathcal{B}$ made at the beginning of the game. Thus all of $\mathcal{B}$'s queries are permitted as ρ was added to list $\mathcal{L}$ in game $\mathsf{T\text{-}EUF\text{-}CMA\text{-}HRK}$.

Once $\mathcal{A}$ outputs its forgery (m^*, σ^*, ID^*), $\mathcal{B}$ computes $\rho^* \leftarrow \mathsf{H}(St, ID^*)$, $pk^*_{ID^*} \leftarrow \mathtt{TRSig.RandPK}(pk, \rho^*)$, and outputs (m^*, σ^*, ρ^*) as its forgery. For this to be a valid forgery according to $\mathsf{T\text{-}EUF\text{-}CMA\text{-}HRK}$, three properties have to be satisfied: (1) $\rho^* \in \mathcal{L}$, (2) $(pk^*_{ID^*}, m^*) \notin \mathcal{S}$, and (3) $\mathtt{TRSig.Ver}(pk^*_{ID^*}, m^*, \sigma^*) = 1$. The first condition follows simply from the fact that $\mathcal{B}$ sampled H from a small-range distribution using only samples that are in $\mathcal{L}$. If $\mathcal{A}$ is successful, it has never queried (m^*, ID^*) to its signing oracle, hence $\mathcal{B}$ has not queried $(m^*, \rho^*) = (m^*, \mathsf{H}(St, ID^*))$ to its signing oracle. Finally, validity of the signature output by $\mathcal{A}$ implies validity of the signature output by $\mathcal{B}$. Assuming $\mathtt{TRSig}$ to be T-EUF-CMA-HRK secure yields $\Pr[\mathcal{A} \text{ wins } \mathsf{G}_1] \leq \Pr[\mathcal{B} \text{ wins } \mathsf{T\text{-}EUF\text{-}CMA\text{-}HRK}] \leq \mathrm{negl}(\lambda)$. Thus we obtain $\frac{1}{2p} = \frac{1}{p} - \frac{1}{2p} \leq \epsilon - \frac{1}{2p} \leq \mathrm{negl}(\lambda)$, which yields the desired contradiction to the initial assumption that ϵ is non-negligible. □

Theorem 3. *Let* $\mathtt{DTW}$ *be the deterministic threshold wallet scheme according to Fig. 2. Assume* $\mathtt{TRSig}$ *is an interactive threshold signature scheme with rerandomizable public keys according to Definition 3) and* H *is a random oracle, then* $\mathtt{DTW}$ *is unlinkable (cf. Definition 10).*

As mentioned before, the proof of the above theorem follows essentially from [ADE+20] as the threshold property affects only the secret key and signing algorithm.

5 Rerandomization of Threshold CSI-FiSh Keys

In this section, we present an interactive threshold signature scheme with rerandomizable keys that is based on the threshold version of CSI-FiSh from [DM20].

5.1 Key Generation and Shamir Secret Sharing

As in [DM20] we assume for brevity that there is a trusted dealer, who performs the key generation and distributes key shares to the n involved participants $P_1, \ldots, P_n$ via a secure channel. However, we note that key generation can also be performed in a distributed and actively secure way, as described in [BDPV21].

Let $s \in \mathbb{Z}/N\mathbb{Z}$ be a CSI-FiSh secret key sampled as above. We use $(t+1)$-out-of-n Shamir secret sharing [Sha79] to generate key shares for $t < n$. The trusted dealer samples a polynomial $f(x)$ of degree t with coefficients in $\mathbb{Z}/N\mathbb{Z}$, such that $f(0) = s$. Participant P_i then receives the secret share $s_i = f(i)$ from the dealer. A set S of participants P_i of cardinality at least $t+1$ with $S \subset \{1, \ldots, n\}$ and $i \in S$ can then recover s through Lagrange polynomial interpolation:

$$s = f(0) = \sum_{i \in S} f(i) \cdot L_{0,i}^S, \quad \text{with} \quad L_{l,i}^S = \prod_{\substack{j \in S \\ j \neq i}} \frac{j-l}{j-i} \mod N.$$

Note that for composite N, the differences $j - i$ must be invertible in the ring $\mathbb{Z}/N\mathbb{Z}$ for all pairs (i, j), which means that the number of participants is restricted to $n < p_1$ for the smallest prime factor p_1 of N, see [DM20].

5.2 Threshold CSI-FiSh Signing

We briefly describe the threshold signing procedure from [DM20], which we will use for the rerandomized scheme. We assume that the private key s is shared as described above, such that participant P_i holds s_i. For simplicity, we assume that the set of participants taking part in signing is $S = \{1, \ldots, t+1\}$.

For generating the commitment curves, participant P_i samples λ integers $b_{i,j}$ for $1 \leq j \leq \lambda$, where λ is the security level resp. the number of rounds of the identification scheme we need to perform. The signing process uses $b_j = \sum_{i \in S} b_{i,j}$ for generating the commitment curves $E^{(j)} = [b_j]E_0$ in the following way. We set $E_0^{(j)} = E_0$ and compute $E^{(j)}$ in a round-robin fashion: P_i receives $E_{i-1}^{(j)}$, and outputs $E_i^{(j)} = [b_{i,j}]E_{i-1}^{(j)}$. Therefore, at the end of the process participant P_{t+1} outputs $E_{t+1}^{(j)} = \left[\sum_{i \in S} b_{i,j}\right] E_0 = [b_j]E_0 = E^{(j)}$. Each party can then obtain the challenges $(c_1, \ldots, c_\lambda) = \mathsf{H}(E^{(1)}, \ldots, E^{(\lambda)}, m)$ as above.

For generating the response r_j, each participant P_i outputs $r_{i,j} = b_{i,j} - c_j \cdot s_i \cdot L_{0,i}^S$. Thus, we can compute $r_j = \sum_{i \in S} r_{i,j} = \sum_{i \in S} b_{i,j} - c_j \cdot \sum_{i \in S} s_i \cdot L_{0,i}^S = b_j - c_j \cdot s$. The signature then is $(c_1, \ldots, c_\lambda, r_1, \ldots, r_\lambda)$. Note that running this with for arbitrary participant sets S of cardinality $k+1$ requires to fix S at the beginning of the procedure, such that each participant can compute $L_{0,i}^S$.

A transcript of the full signing process therefore contains the following communication between participants: the intermediate curves $E_i^{(j)}$ for each commitment curve $E^{(j)}$, and the partial responses $r_{i,j}$ for each r_j, where $i \in S$.

5.3 Rerandomization of Threshold CSI-FiSh Keys

We now show how to transform the interactive threshold signature scheme based on CSI-FiSh as described in the previous section into an interactive threshold signature scheme with rerandomizable keys. In order to do so, we must instantiate the key rerandomization algorithms `RandSK` and `RandPK`. Indeed, we use the same instantiation for these algorithms as Das et al. [DEF+23] for their interactive threshold ECDSA scheme with rerandomizable keys. Importantly, the two algorithms must be deterministic. Therefore, the `RandSK` algorithm computes the polynomial F, which shares randomness ρ, in a deterministic way using a cryptographic hash function G. `TSign` follows the same approach as described above, using the rerandomized key shares and public key. The following theorem shows the security of the scheme accompanied with a proof below. The properties of public key rerandomizability and simulatability follow similarly to the corresponding properties of our CSI-FiSh scheme with rerandomizable keys from Section 3.

$\texttt{RandSK}(i, sk_i, \rho)$	$\texttt{RandPK}(pk, \rho)$
parse sk_i **as** s_i	**parse** pk **as** E_1
for $j \in [t] : a_j \leftarrow \mathsf{G}(\rho, j)$	$E_{1,\rho} \leftarrow [\rho]E_1$
$F(x) \coloneqq a_t \cdot x^t + \cdots + a_1 \cdot x + \rho$	**return** $E_{1,\rho}$
$s'_i \leftarrow s_i + F(i)$	
return s'_i	

Figure 3: Key rerandomization algorithms for the threshold CSI-FiSh signature scheme. These algorithms are identical to the ones introduced by Das et al. [DEF+23] for the construction of an interactive threshold ECDSA scheme with rerandomizable keys. The `RandSK` algorithm uses a cryptographic hash function G, which allows to deterministically compute the polynomial F.

Theorem 4. *Let Σ be the threshold CSI-FiSh signature scheme (cf. Fig. 3). Assuming that Σ is unforgeable, the rerandomizable threshold CSI-FiSh signature scheme Σ' (cf. Fig. 3) is unforgeable.*

Proof. Let $\mathcal{A}$ be an efficient quantum adversary breaking unforgeability of Σ'. We construct the following PPT adversary $\mathcal{B}$ against the unforgeability of Σ. Adversary $\mathcal{B}$ receives the public key pk which it provides as input to $\mathcal{A}$. Prior to making any signature queries, $\mathcal{A}$ declares a set of indices $\mathcal{C}$, which defines the set of users that it wants to corrupt. $\mathcal{B}$ uses the same set $\mathcal{C}$ in its game $\mathsf{T\text{-}EUF\text{-}CMA}$ for which it obtains the corresponding secret key shares. These shares are then given to $\mathcal{A}$. Whenever $\mathcal{A}$ makes a signature query on a message m, a randomization value ρ, $\mathcal{B}$ makes a signature query on the same message m. Adversary $\mathcal{B}$ then obtains the signature σ and the transcript Tr. Wlog we assume that parties with indices $\{1, \ldots, t+1\}$ participate in the signature generation and we denote the set of these parties by $\mathcal{S}$ in the following. The transcript Tr that $\mathcal{B}$ obtains is $(E_i^{(j)}, r_{i,j})_{i,j \in [t+1] \times [\lambda]}$, where $E_i^{(j)} = [b_{i,j}]E_{i-1}^{(j)}$ and $r_{i,j} = b_{i,j} - c_j \cdot s_i \cdot L_{0,i}^S$. To provide a matching transcript for $\mathcal{A}$, $\mathcal{B}$ first computes $F(x) = a_t x^t + \cdots + a_1 x + \rho$, which allows to recover the shares $\rho_i = F(i)$ of the different participants. Subsequently, $\mathcal{B}$ computes $r'_{i,j} = r_{i,j} - c_j \cdot \rho_i \cdot L_{0,i}^S$. Finally, $\mathcal{B}$ sends $\mathcal{A}$ the signature σ along with the transcript $Tr' = (E_i^{(j)}, r'_{i,j})_{i,j \in [t+1] \times [\lambda]}$. Whenever $\mathcal{A}$ outputs

a forgery (m, σ', ρ), $\mathcal{B}$ proceeds as follows. It parses $\sigma' = (c_1, \ldots, c_\lambda, r'_1, \ldots, r'_\lambda)$, computes $r_i \leftarrow r'_i + c_i\rho$, sets $\sigma = (c_1, \ldots, c_\lambda, r_1, \ldots, r_\lambda)$, and outputs (m, σ).

It remains to argue the simulation of the signing oracle by $\mathcal{B}$ is correct and that that it outputs a valid forgery conditioning that $\mathcal{A}$ does. We start by arguing that every valid forgery by $\mathcal{A}$ results in a valid forgery by $\mathcal{B}$. The output by $\mathcal{A}$ being a valid forgery implies that $(c_1, \ldots, c_\lambda) = \mathsf{H}([r'_1]E'_{c_1}, \ldots, [r'_\lambda]E'_{c_\lambda}, m)$. By construction $\mathcal{B}$ outputs (m, σ), where $\sigma = (r_1, \ldots, r_\lambda, c_1, \ldots, c_\lambda)$. It holds that

$$\begin{aligned}
&\mathsf{H}([r_1]E_{c_1}, \ldots, [r_\lambda]E_{c_\lambda}, m) \\
&= \mathsf{H}([r'_1 + c_1\rho]E_{c_1}, \ldots, [r'_\lambda + c_\lambda\rho]E_{c_\lambda}, m) \\
&= \mathsf{H}([r'_1][c_1\rho]E_{c_1}, \ldots, [r'_\lambda][c_\lambda\rho]E_{c_\lambda}, m) \\
&= \mathsf{H}([r'_1]E'_{c_1}, \ldots, [r'_\lambda]E'_{c_\lambda}, m) = (c_1, \ldots, c_\lambda)\,,
\end{aligned}$$

where we distinguished between two cases: (1) if $c_i = 0$, then $[c_i\rho]E_{c_i} = [0]E_0 = E_0 = E'_0$ and (2) if $c_i = 1$, then $[c_i\rho]E_{c_i} = [\rho]E_1 = E'_1$. This yields that (m, σ) is a valid forgery against Σ.

Finally, we argue that $\mathcal{B}$ perfectly simulates oracle Sign for $\mathcal{A}$. When $\mathcal{A}$ makes a query for a message m together with some randomness ρ to its signing oracle, $\mathcal{B}$ obtains $\sigma = (r_1, \ldots, r_\lambda, c_1, \ldots, c_\lambda)$ from its own signing oracle Sign. By correctness of Σ, it holds that $(c_1, \ldots, c_\lambda) = \mathsf{H}([r_1]E_{c_1}, \ldots, [r_\lambda]E_{c_\lambda}, m)$. We have to show that the modified signature $\sigma' = (r'_1, \ldots, r'_\lambda, c_1, \ldots, c_\lambda)$ computed by $\mathcal{B}$ is a valid signature under the rerandomized public key. By construction, we have that $pk' = E'_1 = [\rho]E_1 = [\rho][s]E_0 = [\rho + s]E_0$, where ρ is the randomness that $\mathcal{A}$ obtain from querying its oracle Rand. For the sake of simplicity of notation, let $E'_0 = E_0$ denote the public starting curve for a rerandomized key. We then get

$$\begin{aligned}
&\mathsf{H}([r'_1]E'_{c_1}, \ldots, [r'_\lambda]E'_{c_\lambda}, m) \\
&= \mathsf{H}([r_1 - c_1\rho]E'_{c_1}, \ldots, [r_\lambda - c_\lambda\rho]E'_{c_\lambda}, m) \\
&= \mathsf{H}([r_1][-c_1\rho]E'_{c_1}, \ldots, [r_\lambda][-c_\lambda\rho]E'_{c_\lambda}, m) \\
&= \mathsf{H}([r_1]E_{c_1}, \ldots, [r_\lambda]E_{c_\lambda}, m) = (c_1, \ldots, c_\lambda)\,,
\end{aligned}$$

where we have again distinguished between two cases: (1) if $c_i = 0$, then $[-c_i\rho]E'_{c_i} = [0]E'_0 = E'_0 = E_0$ and (2) if $c_i = 1$, then $[-c_i\rho]E'_{c_i} = [-\rho]E'_1 = [-\rho][\rho]E_1 = [0]E_1 = E_1$. Thus, σ' is a valid signature under the rerandomized public key.

It is left to argue that $\mathcal{B}$ provides a valid transcript for $\mathcal{A}$. The mere difference in the transcript is the computation of the $r_{i,j}$. It is easy to see that $\mathcal{B}$ can generate the correct shares ρ_i for the participants which allows to provide a matching transcript by subtracting $c_j \cdot \rho_i \cdot L^S_{0,i}$ from the response $r_{i,j}$. □

By dropping the threshold part of the proof, we obtain a proof for Theorem 1.

6 Practical Instantiation

6.1 Blockchain Application

A transaction in a cryptocurrency system typically transfers values from one user to another. To verify that the sender is eligible to conduct such a transaction, the sender adds a signature and its public key to the transaction. As transactions are collected in blocks of limited

size, created at a roughly constant rate, the transaction throughput heavily depends on the signature and public key sizes.

Alkeilani Alkadri et al. [ADE+20] instantiate a post-quantum secure deterministic wallet with the lattice-based signature scheme qTESLA [ABB+20], leading to a size of roughly 17.5 KB per transaction for the combined size of the relatively large public keys (14,880B) and signatures (2,592B) at NIST security level I. Similarly, Hu [Hu23] recently presented a post-quantum secure deterministic wallet that is instantiated with the lattice-based signature scheme Falcon [FHK+18] which achieves a significantly better transaction size of around 1.66 KB. In contrast, our instantiations based on CSI-FiSh allow for even more compact transaction sizes. Following [BKV19, Table 3], the optimal parameter choice for CSI-FiSh in our case leads to public keys and signatures of 512B resp. 956B, and thus a combined size of 1,486B per transaction. That is, our instantiation with the CSI-FiSh scheme reduces the transaction size compared to the work of Alkeilani Alkadri et al. [ADE+20] by a factor of roughly 11 and compared to the work of Hu [Hu23] by roughly 174B. Importantly, the main focus of our work is to provide a post-quantum secure instantiation of a deterministic *threshold* wallet which does not rely on the assumption of incorruptible cold wallets. Neither of the two prior works considers threshold wallets and indeed, it seems hard to extend the previous schemes to the threshold setting, as thresholdizing lattice-based schemes is known to be inherently difficult [CS19].

In the context of blockchain applications, the running time of 1.48s for signing resp. verifying in the single-user setting is less relevant due to the longer block rate in the order of seconds or even minutes. We note that the runtime of CSI-FiSh can be significantly reduced using [MR18, MCR19, CCC+19].

6.2 Instantiating Threshold CSI-FiSh

For instantiating our passively secure threshold wallet scheme based on CSI-FiSh, the first step is the key generation that distributes key shares to participants. In [DM20], this is done by a trusted dealer. However, if necessary one can use the actively secure key generation approach CSI-RAShi [BDPV21]. Although this involves significantly more computational effort, it is a one-time computation, and may thus be interesting in practice. Using sharing-friendly keys from CSI-SharK [ABCP23], this effort can be reduced.

Since the class group in CSI-FiSh using CSIDH-512 has cardinality N with 3, 37, and 1407181 as smallest prime factors, we would be limited to two participants as described in Section 5.1. Following [DM20], we can instead use subgroups of of cardinality $N' = N/3$ resp. $N'' = N/(3 \cdot 37)$, allowing for up to 36 resp. more than a million participants at the cost of a slightly reduced bit security.

In terms of performance, the round-robin approach of threshold CSI-FiSh appears to have a negative impact on the runtime. However, we can layer the computations, such that participants can run computations in parallel. In particular, ignoring latency, the optimal parameter set from above allows for thresholds up to 28 without increasing the runtime of signing of 1.48s.

The exact quantum security of CSIDH-512 is debated [Pei20, BLMP19], and CSI-FiSh is limited to this parameter set. A potential alternative is SCALLOP [DFK+23], which allows for larger parameter sets equivalent to CSIDH-1024 with the same threshold construction. However, this comes at significant overhead in the runtime.

We note that our threshold scheme is only passively secure. There are actively secure

variants such as [CS20, MC22], which are expected to be rerandomizable as well. Thus, these alternatives could be used, again at the cost of significant overhead in signing time without impacting the signature sizes. If the additional overhead and communication between participants of the signing process is tolerable in specific blockchain settings, this might be a viable option to increase security even further.

Acknowledgments

This work was funded in part by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under SFB 1119 – 236615297, by the Hector Foundation II, by the German Federal Ministry of Education and Research (BMBF) project 6G-RIC (grant nr. 6KISK033), by the German Federal Ministry of Education and Research and the Hessen State Ministry for Higher Education, Research and the Arts within their joint support of the National Research Center for Applied Cybersecurity ATHENE, and by the European Research Council (ERC) under the European Union's Horizon 2020 and Horizon Europe research and innovation programs (grant CRYPTOLAYER-101044770).

References

[ABB+20] Erdem Alkim, Paulo S. L. M. Barreto, Nina Bindel, Juliane Krämer, Patrick Longa, and Jefferson E. Ricardini. The lattice-based digital signature scheme qTESLA. In Mauro Conti, Jianying Zhou, Emiliano Casalicchio, and Angelo Spognardi, editors, *ACNS 20, Part I*, volume 12146 of *LNCS*, pages 441–460. Springer, Heidelberg, October 2020.

[ABCP23] Shahla Atapoor, Karim Baghery, Daniele Cozzo, and Robi Pedersen. CSI-SharK: CSI-FiSh with Sharing-friendly Keys. In Leonie Simpson and Mir Ali Rezazadeh Baee, editors, *Information Security and Privacy - 28th Australasian Conference, ACISP 2023, Brisbane, QLD, Australia, July 5-7, 2023, Proceedings*, volume 13915 of *Lecture Notes in Computer Science*, pages 471–502. Springer, 2023.

[ACdMT05] Giuseppe Ateniese, Daniel H. Chou, Breno de Medeiros, and Gene Tsudik. Sanitizable signatures. In Sabrina De Capitani di Vimercati, Paul F. Syverson, and Dieter Gollmann, editors, *ESORICS 2005*, volume 3679 of *LNCS*, pages 159–177. Springer, Heidelberg, September 2005.

[ADE+20] Nabil Alkeilani Alkadri, Poulami Das, Andreas Erwig, Sebastian Faust, Juliane Krämer, Siavash Riahi, and Patrick Struck. Deterministic wallets in a quantum world. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 1017–1031. ACM Press, November 2020.

[AGKK19] Myrto Arapinis, Andriana Gkaniatsou, Dimitris Karakostas, and Aggelos Kiayias. A formal treatment of hardware wallets. In Ian Goldberg and Tyler Moore, editors, *FC 2019*, volume 11598 of *LNCS*, pages 426–445. Springer, Heidelberg, February 2019.

[ASY22] Shweta Agrawal, Damien Stehlé, and Anshu Yadav. Round-optimal lattice-based threshold signatures, revisited. In Mikolaj Bojanczyk, Emanuela Merelli, and

David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, July 4-8, 2022, Paris, France*, volume 229 of *LIPIcs*, pages 8:1–8:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[BDF+11] Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry. Random oracles in a quantum world. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT*, volume 7073 of *LNCS*, pages 41–69. Springer, Heidelberg, December 2011.

[BDPV21] Ward Beullens, Lucas Disson, Robi Pedersen, and Frederik Vercauteren. CSI-RAShi: Distributed key generation for CSIDH. In Jung Hee Cheon and Jean-Pierre Tillich, editors, *PQCrypto 2021*, pages 257–276. Springer, Heidelberg, 2021.

[Bit] Bitcoin post-quantum. https://bitcoinpq.org/.

[BKV19] Ward Beullens, Thorsten Kleinjung, and Frederik Vercauteren. CSI-FiSh: Efficient isogeny based signatures through class group computations. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part I*, volume 11921 of *LNCS*, pages 227–247. Springer, Heidelberg, December 2019.

[BLMP19] Daniel J. Bernstein, Tanja Lange, Chloe Martindale, and Lorenz Panny. Quantum circuits for the CSIDH: Optimizing quantum evaluation of isogenies. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 409–441. Springer, Heidelberg, May 2019.

[Bol03] Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In Yvo Desmedt, editor, *PKC 2003*, volume 2567 of *LNCS*, pages 31–46. Springer, Heidelberg, January 2003.

[CCC+19] Daniel Cervantes-Vázquez, Mathilde Chenu, Jesús-Javier Chi-Domínguez, Luca De Feo, Francisco Rodríguez-Henríquez, and Benjamin Smith. Stronger and faster side-channel protections for CSIDH. In Peter Schwabe and Nicolas Thériault, editors, *LATINCRYPT 2019*, volume 11774 of *LNCS*, pages 173–193. Springer, Heidelberg, October 2019.

[CGJ+99] Ran Canetti, Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Adaptive security for threshold cryptosystems. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 98–115. Springer, Heidelberg, August 1999.

[CS19] Daniele Cozzo and Nigel P. Smart. Sharing the LUOV: Threshold post-quantum signatures. In Martin Albrecht, editor, *17th IMA International Conference on Cryptography and Coding*, volume 11929 of *LNCS*, pages 128–153. Springer, Heidelberg, December 2019.

[CS20] Daniele Cozzo and Nigel P. Smart. Sashimi: Cutting up CSI-FiSh secret keys to produce an actively secure distributed signing protocol. In Jintai Ding and

Jean-Pierre Tillich, editors, *PQCrypto 2020*, pages 169–186. Springer, Heidelberg, 2020.

[CSCD+23] Jorge Chavez-Saab, Maria Corte-Real Santos, Luca De Feo, Jonathan Komada Eriksen, Basil Hess, David Kohel, Antonin Leroux, Patrick Longa, Michael Meyer, Lorenz Panny, Sikhar Patranabis, Christophe Petit, Francisco Rodríguez-Henríquez, Sina Schaeffler, and Benjamin Wesolowski. SQIsign: Algorithm specifications and supporting documentation, 2023. National Institute of Standards and Technology.

[DEF+21] Poulami Das, Andreas Erwig, Sebastian Faust, Julian Loss, and Siavash Riahi. The exact security of BIP32 wallets. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021*, pages 1020–1042. ACM Press, November 2021.

[DEF+23] Poulami Das, Andreas Erwig, Sebastian Faust, Julian Loss, and Siavash Riahi. Bip32-compatible threshold wallets. *Cryptology ePrint Archive*, 2023.

[DFK+23] Luca De Feo, Tako Boris Fouotsa, Péter Kutas, Antonin Leroux, Simon-Philipp Merz, Lorenz Panny, and Benjamin Wesolowski. SCALLOP: scaling the CSI-FiSh. In Alexandra Boldyreva and Vladimir Kolesnikov, editors, *Public-Key Cryptography - PKC 2023 - 26th IACR International Conference on Practice and Theory of Public-Key Cryptography, Atlanta, GA, USA, May 7-10, 2023, Proceedings, Part I*, volume 13940 of *Lecture Notes in Computer Science*, pages 345–375. Springer, 2023.

[DFL19] Poulami Das, Sebastian Faust, and Julian Loss. A formal treatment of deterministic wallets. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 651–668. ACM Press, November 2019.

[DG19] Luca De Feo and Steven D. Galbraith. SeaSign: Compact isogeny signatures from class group actions. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 759–789. Springer, Heidelberg, May 2019.

[DM20] Luca De Feo and Michael Meyer. Threshold schemes from isogeny assumptions. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part II*, volume 12111 of *LNCS*, pages 187–212. Springer, Heidelberg, May 2020.

[EEE20] Muhammed F. Esgin, Oguzhan Ersoy, and Zekeriya Erkin. Post-quantum adaptor signatures and payment channel networks. In Liqun Chen, Ninghui Li, Kaitai Liang, and Steve A. Schneider, editors, *ESORICS 2020, Part II*, volume 12309 of *LNCS*, pages 378–397. Springer, Heidelberg, September 2020.

[ER22] Andreas Erwig and Siavash Riahi. Deterministic wallets for adaptor signatures. In Vijayalakshmi Atluri, Roberto Di Pietro, Christian Damsgaard Jensen, and Weizhi Meng, editors, *ESORICS 2022 , Part II*, volume 13555 of *LNCS*, pages 487–506. Springer, Heidelberg, September 2022.

[ESS21] Edward Eaton, Douglas Stebila, and Roy Stracovsky. Post-quantum key-blinding for authentication in anonymity networks. In Patrick Longa and Carla Ràfols, editors, *LATINCRYPT 2021*, volume 12912 of *LNCS*, pages 67–87. Springer, Heidelberg, October 2021.

[ESZ22] Muhammed F. Esgin, Ron Steinfeld, and Raymond K. Zhao. Matrict+: More efficient post-quantum private blockchain payments. In *2022 IEEE SP*, 2022.

[EZS+19] Muhammed F. Esgin, Raymond K. Zhao, Ron Steinfeld, Joseph K. Liu, and Dongxi Liu. MatRiCT: Efficient, scalable and post-quantum blockchain confidential transactions protocol. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 567–584. ACM Press, November 2019.

[FHK+18] Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Prest, Thomas Ricosset, Gregor Seiler, William Whyte, Zhenfei Zhang, et al. Falcon: Fast-fourier lattice-based compact signatures over NTRU. *Submission to the NISTs PQC standardization process*, 36(5), 2018.

[FKM+16] Nils Fleischhacker, Johannes Krupp, Giulio Malavolta, Jonas Schneider, Dominique Schröder, and Mark Simkin. Efficient unlinkable sanitizable signatures from signatures with re-randomizable keys. In Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang, editors, *PKC 2016, Part I*, volume 9614 of *LNCS*, pages 301–330. Springer, Heidelberg, March 2016.

[FTS+18] Chun-I Fan, Yi-Fan Tseng, Hui-Po Su, Ruei-Hau Hsu, and Hiroaki Kikuchi. Secure hierarchical bitcoin wallet scheme against privilege escalation attacks. In *2018 IEEE Conference on Dependable and Secure Computing (DSC)*, pages 1–8, 2018.

[GJKR96] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Robust threshold DSS signatures. In Ueli M. Maurer, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 354–371. Springer, Heidelberg, May 1996.

[GS15] Gus Gutoski and Douglas Stebila. Hierarchical deterministic bitcoin wallets that tolerate key leakage. In Rainer Böhme and Tatsuaki Okamoto, editors, *FC 2015*, volume 8975 of *LNCS*, pages 497–504. Springer, Heidelberg, January 2015.

[Hu23] Mingxing Hu. Post-quantum secure deterministic wallet: Stateless, hot/cold setting, and more secure. Cryptology ePrint Archive, Paper 2023/062, 2023.

[KMOS21] Yashvanth Kondi, Bernardo Magri, Claudio Orlandi, and Omer Shlomovits. Refresh when you wake up: Proactive threshold wallets with offline devices. In *2021 IEEE Symposium on Security and Privacy*, pages 608–625. IEEE Computer Society Press, May 2021.

[LFA20] Adriano Di Luzio, Danilo Francati, and Giuseppe Ateniese. Arcula: A secure hierarchical deterministic wallet for multi-asset blockchains. In Stephan Krenn, Haya Shulman, and Serge Vaudenay, editors, *CANS 20*, volume 12579 of *LNCS*, pages 323–343. Springer, Heidelberg, December 2020.

[LJY14] Benoît Libert, Marc Joye, and Moti Yung. Born and raised distributively: fully distributed non-interactive adaptively-secure threshold signatures with short shares. In Magnús M. Halldórsson and Shlomi Dolev, editors, *33rd ACM PODC*, pages 303–312. ACM, July 2014.

[Max11] Gregory Maxwell. Deterministic wallets, 2011. https://bitcointalk.org/index.php?topic=19137.msg239768.

[MC22] Philipp Muth and Fabio Campos. On actively secure fine-grained access structures from isogeny assumptions. In Jung Hee Cheon and Thomas Johansson, editors, *Post-Quantum Cryptography - 13th International Workshop, PQCrypto 2022, Virtual Event, September 28-30, 2022, Proceedings*, volume 13512 of *Lecture Notes in Computer Science*, pages 375–398. Springer, 2022.

[MCR19] Michael Meyer, Fabio Campos, and Steffen Reith. On lions and elligators: An efficient constant-time implementation of CSIDH. In Jintai Ding and Rainer Steinwandt, editors, *PQCrypto 2019*, pages 307–325. Springer, Heidelberg, 2019.

[Moc] Mochimo. https://mochimo.org/.

[MPs19] Antonio Marcedone, Rafael Pass, and abhi shelat. Minimizing trust in hardware wallets with two factor signatures. In Ian Goldberg and Tyler Moore, editors, *FC 2019*, volume 11598 of *LNCS*, pages 407–425. Springer, Heidelberg, February 2019.

[MR18] Michael Meyer and Steffen Reith. A faster way to the CSIDH. In Debrup Chakraborty and Tetsu Iwata, editors, *INDOCRYPT 2018*, volume 11356 of *LNCS*, pages 137–152. Springer, Heidelberg, December 2018.

[Pei20] Chris Peikert. He gives C-sieves on the CSIDH. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 463–492. Springer, Heidelberg, May 2020.

[QRL] Quantum resistant ledger (qrl). https://github.com/theQRL/Whitepaper/blob/master/QRL_whitepaper.pdf.

[Sch90] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 239–252. Springer, Heidelberg, August 1990.

[SD23] Surbhi Shaw and Ratna Dutta. Compact stateful deterministic wallet from isogeny-based signature featuring uniquely rerandomizable public keys. In *CANS 2023*, Lecture Notes in Computer Science. Springer, 2023.

[Sha79] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11), 1979.

[Sho94] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th FOCS*, pages 124–134. IEEE Computer Society Press, November 1994.

[TMSM21] Erkan Tairi, Pedro Moreno-Sanchez, and Matteo Maffei. Post-quantum adaptor signature for privacy-preserving off-chain payments. In *FC 2021*, 2021.

[Wui12] Pieter Wuille. BIP32 Proposal. https://en.bitcoin.it/wiki/BIP_0032, 2012.

[YLY+22] Xin Yin, Zhen Liu, Guomin Yang, Guoxing Chen, and Haojin Zhu. Secure hierarchical deterministic wallet supporting stealth address. In Vijayalakshmi Atluri, Roberto Di Pietro, Christian Damsgaard Jensen, and Weizhi Meng, editors, *ESORICS 2022*, volume 13554 of *LNCS*, pages 89–109. Springer, Heidelberg, September 2022.

[Zha12] Mark Zhandry. How to construct quantum random functions. In *FOCS*, pages 679–687. IEEE Computer Society Press, October 2012.