Identity-Based Threshold Signatures from Isogenies

Shahla Atapoor

COSIC, KU Leuven, Leuven, Belgium.
firstname.lastname@kuleuven.be
September 23, 2023

Abstract. The identity-based signature, initially introduced by Shamir [Sha84], plays a fundamental role in the domain of identity-based cryptography. It offers the capability to generate a signature on a message, allowing any user to verify the authenticity of the signature using the signer's identifier information (e.g., an email address), instead of relying on a public key stored in a digital certificate. Another significant concept in practical applications is the threshold signature, which serves as a valuable tool for distributing the signing authority. The notion of an identity-based threshold signature scheme pertains to the distribution of a secret key associated with a specific identity among multiple entities, rather than depending on a master secret key generated by a public key generator. This approach enables a qualified group of participants to jointly engage in the signing process. In this paper, we present two identity-based threshold signature schemes based on isogenies, each of which addresses a different aspect of security. The first scheme prioritizes efficiency but offers security with abort, while the second scheme focuses on robustness. Both schemes ensure active security in the quantum random oracle model. To build these identity-based threshold signatures, we begin by modifying the identity-based signature scheme proposed by Shaw and Dutta [SD21], to accommodate the CSI-SharK signature scheme. Subsequently, we leverage the resulting identity-based signature and build two threshold schemes within the CSIDH (Commutative Supersingular Isogeny Diffie-Hellman) framework. Our proposed identity-based threshold signatures are designed based on CSI-SharK and can be easily adapted with minimal adjustments to function with CSI-FiSh.

Keywords: Identity-based signature · Identity-based threshold signature · Isogeny-based cryptography · CSI-SharK · CSI-FiSh · CSIDH

1 Introduction

In recent years, there has been a notable surge of interest in identity-based cryptography, initially introduced by Shamir [Sha84]. The rationale behind this heightened attention lies in its remarkable advantages over conventional certificate-based cryptography, primarily due to its ability to circumvent the arduous certificate management procedures inherent in the latter approach. Identity-based

identification serves as a fundamental element in identity-based cryptography, which was initially proposed in 2004 by Bellare et al. [BNN04] and Kurosawa et al. [KH04] as separate endeavors. In an identification scheme based on identity, each user designates their identity, such as an email address, as their public key. The *public key generator* (e.g., a dealer) generates the corresponding secret key for the user's identity by utilizing its master secret key. Subsequently, the user, assuming the role of a prover, can employ this secret key to establish its identity to a verifier who possesses the associated public key.

Shamir in the same work [Sha84] introduced the concept of an identity-based signature scheme. This innovation garnered significant attention and later gained further prominence through the Fiat-Shamir transformation [FS87]. The advent of identity-based signature schemes brought a fresh perspective to the field, as they enabled users to sign messages instead of merely authenticating their identities [KN09,Hes02]. Consequently, the verification process became the responsibility of the verifier, who checks the validity of the signature. On the other hand, due to a wide range of applications (e.g., in blockchains), threshold signature schemes have received more attention in recent years. Such schemes allow distributing the secret key into shares among multiple parties or devices, such that only a set of authorized parties can jointly sign a message to produce a single signature. Key recovery attacks on threshold signature schemes require more effort than on the non-threshold ones, as the adversary has to attack more than one device or party simultaneously. In the realm of identity-based signatures, Baek and Zheng [BZ04], for the first time, introduced the concept of secret sharing among multiple parties. They devised an identity-based threshold signature that utilizes bilinear pairings.

An illustrative application of the identity-based threshold signature scheme can be envisioned in the following scenario: Let us consider Alice, who serves as the president of a company. In this capacity, she has established an identity that represents the company and possesses a private key associated with this identity. Through the utilization of this private key, Alice can affix her signature to various documents. However, she harbors concerns regarding situations where she may be physically absent. Consequently, she desires to delegate this signing authority to a set of signature-generation servers. By employing this arrangement, signatures for a given message can be collectively generated by these servers. Importantly, any user can successfully verify the resulting signature using the company's publicly accessible identity, provided that the user obtains a specific number of partial signatures from the signature-generation servers.

In the wake of Shor's groundbreaking results [Sho94] on the vulnerabilities of Factoring and Discrete Logarithm (DL) problems to quantum attacks, a new wave of investigation emerged among researchers. Their focus shifted towards the exploration of post-quantum cryptographic techniques capable of constructing primitives and protocols resilient against the formidable threat posed by quantum adversaries. One prominent avenue of inquiry in this domain revolves around isogeny-based cryptography. The concept of employing isogenies as a cryptographic foundation was initially introduced by Couveignes [Cou06], followed by

the notable works of Rostovtsev and Stolbunov [RS06,Sto10]. These scholars embarked upon devising innovative methods to construct cryptographic protocols based on isogenies that could achieve post-quantum security. In [PCZ+20], Peng et al. introduced an identity-based signature scheme based on isogenies that accounts for post-quantum security concerns. Their construction is built based on CSI-FiSh signature scheme [BKV19]. In 2021, Shaw and Dutta [SD21] analysed Peng et al.'s scheme and uncovered vulnerabilities in both the identity-based signature scheme itself and its underlying trapdoor samplable relation. Then, Shaw and Dutta [SD21] presented a fixed version of their construction.

Our Contribution. We first modify the CSI-FiSh-based identity-based signature scheme of Shaw and Dutta [SD21] to work with the CSI-SharK scheme, which was recently proposed by Atapoor et al. [ABCP23a] and shown to outperform CSI-FiSh in the threshold setting. This translation allows us to leverage the properties of CSI-SharK, and obtain a new identity-based signature scheme from isogenies that has considerably shorter master secret key in comparison with the original scheme [SD21].

Next, we use the resulting identity-based signature scheme and propose two identity-based threshold signature schemes based on isogenies, each of which addresses a different aspect of security. Both protocols are designed to achieve active security, ensuring the security of the protocol even in the scenarios where parties are malicious and deviate from honest protocol execution. In our initial threshold signature scheme, honest parties have the ability to detect any misbehavior from malicious entities, resulting in the termination of the protocol execution. However, it is important to note that this protocol only achieves active security with abort and identification of the adversary remains unattainable.

To deal with the above concern, as the next contribution, we propose a robust scheme, which additionally guarantees the correctness of final output. The second threshold signature scheme ensures security against active adversaries and achieves robustness, thereby enabling the identification and expulsion of any malicious party responsible for protocol malfunctions or misconduct. The remaining parties then collaboratively reconstruct the compromised party's secret, facilitating the seamless continuation and completion of the protocol, ultimately yielding the final signature. Our technique to achieve robustness is inspired from the construction of ThreshER SharK signature scheme [ABCP23c]. ThreshER SharK is a Threshold, Efficient and Robust signature scheme that recently is proposed by Atapoor et al. [ABCP23c] and is built on top of CSI-SharK signature scheme [ABCP23a]. To achieve robustness, the distributed signing protocol needs to be run by all the parties, rather than a qualified set of them. Therefore, our second construction is less efficient in comparison with the first one, but it can achieve robustness.

Organization. Sec. 2 presents some preliminaries which will be used in the follow-up sections. In Sec. 3, we adapt the CSI-FiSh based identity-based signature scheme of Shaw and Dutta [SD21] to work with the CSI-SharK scheme [ABCP23a]. In Sec. 4, we present the first identity-based threshold signature based on isoge-

nies with abort. In Sec. 5, we propose the first robust identity-based threshold signature scheme based on isogenies. We conclude the paper in Sec. 6.

2 Preliminaries

Next, we provide an overview of several key concepts, which are used in the follow-up sections. Some of them are provided in App. 6.

Notation We use the notation $x \leftarrow X$ to represent the assignment of a uniformly random value to the variable x from the set X, assuming a uniform distribution over X. If $\mathcal{D}$ is a probability distribution over a set X, we indicate the assignment $x \leftarrow \mathcal{D}$ as the process of sampling from the set X according to the distribution $\mathcal{D}$. The concatenation of strings s_1 and s_2 is represented by $s_1 \| s_2$. When referring to a probabilistic polynomial-time (PPT) algorithm as A, the notation $a \leftarrow A$ represents the assignment of the output of A, where the probability distribution is over the random tape of A. Furthermore, we denote $\mathbb{Z}_N$ as the set of integers modulo N, expressed as $\mathbb{Z}/N\mathbb{Z}$. The function $\log(x)$ is defined as the logarithm of x with base 2.

2.1 Isogeny-based Cryptography

Isogenies are rational maps between elliptic curves that are also homomorphisms with respect to the natural group structure on these curves. Our investigation is limited to the set $\mathcal{E}$ of supersingular elliptic curves over prime fields $\mathbb{F}_p$ and separable $\mathbb{F}_p$-rational isogenies defined between them (the so-called CSIDH setting). Isogenies from an elliptic curve to itself are called endomorphisms. Under the addition and composition operations, the endomorphisms of elliptic curves form a ring. The subring of $\mathbb{F}_p$-rational endomorphism rings of curves in $\mathcal{E}$ is always isomorphic to an order $\mathcal{O}$ in the quadratic imaginary field $\mathbb{Q}(\sqrt{-p})$. Separable isogenies are uniquely defined by their kernel, which can be identified with the kernels of ideal classes in the ideal-class group $\mathsf{Cl}(\mathcal{O})$. As a result, we can see the class group as acting on the set $\mathcal{E}$ via a free and transitive group action.

To ensure efficient computation of isogenies, the prime p is usually chosen such that $p - 1 = 4 \prod_i \ell_i$, where the ℓ_i are small prime factors. The factor 4 ensures that $p \equiv 3 \bmod 4$ and that the special elliptic curve $E_0 : y^2 = x^3 + x$ is supersingular. Throughout this work, we assume that the class group $\mathsf{Cl}(\mathcal{O})$ is known, enabling the transformation of arbitrary ideals into efficiently computable isogeny chains of degrees l_i using the relation lattice. We note that this is not a trivial assumption as current class group computations in reach fall short of realistic security levels [BKV19, BS20, Pei20] or lead to very slow protocols [DF17]. We point out, however, that there are polynomial-time quantum algorithms to this end [Kit96]. We refer to [CLM+18, BKV19, Vél71, BDFLS20] for more details on the explicit computations of isogenies. For a more thorough introduction to isogenies and isogeny-based cryptography, we recommend [CLM+18, DF17, Sil09].

Finally, we note that class groups are generally of composite order. By working in cyclic subgroups of $\mathsf{Cl}(\mathcal{O})$ with generator $\mathfrak{g}$ and order $N \mid \#\mathsf{Cl}(\mathcal{O})$, we can redefine the group action as $[\,] : \mathbb{Z}_N \times \mathcal{E} \to \mathcal{E}$, where ideals of the form $\mathfrak{g}^a$ for $a \in \mathbb{Z}_N$ can be reduced modulo the relation lattice and efficiently computed. To work in a subgroup $\mathbb{Z}_{N'} \subset \mathbb{Z}_N$, we can simply use the generator $\mathfrak{g}^{N/N'}$. For the rest of this work, we always assume the choice of the subgroup $\mathbb{Z}_N$ to be such that $\{1, \ldots, n\}$ defines an exceptional set modulo N, i.e. that n is smaller than the smallest divisor of N. Next, we recall some security assumptions that are used in our studied and constructed protocols.

Definition 2.1 (Group Action Inverse Problem (GAIP) [CLM+18,DG19]). *Given two supersingular elliptic curves $E, E' \in \mathcal{E}$ over the same finite field $\mathbb{F}_p$ and with $\mathsf{End}_{\mathbb{F}_p}(E) \simeq \mathsf{End}_{\mathbb{F}_p}(E') \simeq \mathcal{O}$, find $a \in \mathbb{Z}_N$, such that $E' = [a]E$.*

Definition 2.2 (Multi-Target-GAIP [DG19,BKV19]). *Given $k+1$ supersingular elliptic curves $E_0, E_1, \ldots, E_k \in \mathcal{E}$ over $\mathbb{F}_p$ with the same $\mathbb{F}_p$-rational endomorphism ring, find $a \in \mathbb{Z}_N$, s. t. $E_i = [a]E_j$ for some $i, j \in \{0, \ldots, k\}$ with $i \neq j$.*

Definition 2.3 ($(c_0, c_1, \cdots, c_{k-1})$-Vectorization Problem with Auxiliary Inputs ($\mathbb{C}_{k-1}$-VPwAI) [BCP21]). *Given an element $E \in \mathcal{E}$ and the pairs $(c_i, [c_i x]E)_{i=1}^{k-1}$, where $\mathbb{C}_{k-1} = \{c_0 = 0, c_1 = 1, c_2, \ldots, c_{k-1}\}$ is an exceptional set, find $x \in \mathbb{Z}_N$.*

2.2 Identity-Based Signature Schemes

Next, we recall the definition of identity-based signatures from [SD21], which originally were proposed by Shamir in [Sha84].

Definition 2.4 (Identity-Based Signature Scheme). *An identity-based signature scheme consists of four PPT algorithms* $(\mathsf{Setup}, \mathsf{Extract}, \mathsf{Sign}, \mathsf{Verify})$.

- $(pp, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$*: Given the security parameter λ, it outputs public parameters pp and a master secret key* msk.
- $\mathsf{usk}_{\mathsf{id}} \leftarrow \mathsf{Extract}(pp, \mathsf{msk}, \mathsf{id})$*: Given pp,* msk*, and the user identity* id*, it outputs the user secret key* $\mathsf{usk}_{\mathsf{id}}$ *for the given* id.
- $\sigma \leftarrow \mathsf{Sign}(pp, m, \mathsf{usk}_{\mathsf{id}})$*: Given pp,* $\mathsf{usk}_{\mathsf{id}}$*, and a message m, it outputs σ.*
- $(1/0) \leftarrow \mathsf{Verify}(pp, \mathsf{id}, m, \sigma)$*: Given pp,* id*, m, and signature σ, it outputs 1 if σ is a valid signature on m, otherwise outputs 0.*

Definition 2.5 (Correctness). *For all* $(pp, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$*, all* $\mathsf{usk}_{\mathsf{id}} \leftarrow \mathsf{Extract}(pp, \mathsf{msk}, \mathsf{id})$*, all m and* id*, we have* $\mathsf{Verify}(pp, \mathsf{id}, m, \mathsf{Sign}(pp, m, \mathsf{usk}_{\mathsf{id}})) = 1$.

Definition 2.6 (Security). *An IDentity-based Signature (IDS) scheme is said to be secure against UnForgeability against chosen-identity and Chosen Message Attacks (UF-IDS-CMA) [SD21, PCZ+20] if for all PPT adversaries $\mathcal{A}$, there exists a negligible function ϵ such that*

$$Adv^{UF-IDS-CMA}_{IDS,\mathcal{A}}(\lambda) = \Pr[\mathcal{A} \text{ wins in } Exp^{UF-IDS-CMA}_{IDS,\mathcal{A}}(\lambda)] < \epsilon,$$

where the experiment $Exp^{UF-IDS-CMA}_{IDS,\mathcal{A}}(\lambda)$ is described in Fig. 1.

Input: The challenger $\mathcal{C}$ takes input the security parameter 1^λ, and generates $(pp, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$. It gives the public parameters pp to the adversary $\mathcal{A}$ while keeping the secret msk to itself.

Query Phase: $\mathcal{C}$ responds to polynomially many adaptive queries made by $\mathcal{A}$,

- Oracle $O_{\mathsf{Extract}(\mathsf{msk},\cdot)}$: On receiving queries on a user identity id from $\mathcal{A}$, the challenger $\mathcal{C}$ responds with her user secret key $\mathsf{usk}_{\mathsf{id}} \leftarrow \mathsf{Extract}(pp, \mathsf{msk}, \mathsf{id})$ for the given identity id.
- Oracle $O_{\mathsf{Sign}(\mathsf{usk}_{\mathsf{id}},\cdot)}$: On receiving queries on a message m, and a user identity id from the adversary $\mathcal{A}$, the challenger $\mathcal{C}$ responds with a signature $\sigma \leftarrow \mathsf{Sign}(pp, m, \mathsf{usk}_{\mathsf{id}})$ where $\mathsf{usk}_{\mathsf{id}} \leftarrow \mathsf{Extract}(pp, \mathsf{msk}, \mathsf{id})$ is the user secret key corresponding to the identity id.

Forgery: $\mathcal{A}$ eventually outputs a message $m^\star$, user identity $\mathsf{id}^\star$, and a forge signature $\sigma^\star$. $\mathcal{A}$ wins the game if $1 \leftarrow \mathsf{Verify}(pp, \mathsf{id}, m, \sigma)$, with the restriction that $\mathsf{id}^\star$ has not been queried to $O_{\mathsf{Extract}(\mathsf{msk},\cdot)}$ and $(m^\star, \mathsf{id}^\star)$ has not been queried to $O_{\mathsf{Sign}(\mathsf{usk}_{\mathsf{id}},\cdot)}$.

Fig. 1. $Exp^{UF-IDS-CMA}_{IDS,\mathcal{A}}(\lambda)$: UnForgability against Chosen Message Attacks.

2.3 Identity-Based Threshold Signature Scheme

We use the definition of an identity-based threshold signature scheme as proposed by Baek and Zheng [BZ04], which is outlined below:

Definition 2.7 (Identity-Based Threshold Signature Scheme). *A (t, n)-identity-based threshold signature consists* $(\mathsf{Setup}, \mathsf{Extract}, \mathsf{DKey}, \mathsf{DSign}, \mathsf{Verify})$*:*

- $(pp, \mathsf{msk}) \leftarrow \mathsf{Setup}(\lambda)$*: Given a security parameter λ, the algorithm generates the master secret key* msk *and the public parameters pp.*
- $\mathsf{usk}_{\mathsf{id}} \leftarrow \mathsf{Extract}(pp, \mathsf{msk}, \mathsf{id})$*: Given the public parameters pp, the master secret key* msk*, and a user identity* id*, the algorithm generates a private key* $\mathsf{usk}_{\mathsf{id}}$ *associated with* id*.*
- $\{\mathsf{sk}_l\}_{l=1}^n \leftarrow \mathsf{DKey}(pp, \mathsf{usk}_{\mathsf{id}}, n, t)$*: Given a private key* $\mathsf{usk}_{\mathsf{id}}$ *associated with an identity* id*, a number of signers n and a threshold parameter t, the algorithm generates n shares of* sk_l *and provides each one to the party P_l for $l = \{1, \cdots, n\}$.*
- $\sigma \leftarrow \mathsf{DSign}(pp, m, \{\mathsf{sk}_l\}_{l\in S}, \mathsf{id})$*: Given pp, a message m, shares* $\{\mathsf{sk}_l\}_{l\in S}$*, where $|S| = t + 1$, and* id*. Signers using a robust protocol jointly generate σ. Note that partial signatures of m computed by each party may be broadcast during the execution of the* DSign *protocol.*
- $1/0 \leftarrow \mathsf{Verify}(pp, \mathsf{id}, m, \sigma)$*: Given a signers' identity* id*, a message m and σ, the algorithm outputs 1 if the signature is valid, otherwise 0.*

Note that these definitions are specifically designed for (t, n) identity-based threshold signatures with abort security. It's important to distinguish between this level of security and a stronger concept known as "identifiable-abort." In the case of identifiable-abort, the system can reveal the identity of at least one malicious party in the event of an abort. Our first threshold signature, as proposed

in Sec. 4, provides security with abort but does not achieve the identifiable-abort property. This limitation arises from the fact that the DKey algorithm, executed by the dealer, does not output the verification keys (equivalent to partial public keys). Consequently, during the partial opening phase, honest parties are unable to identify the malicious party based on their partial opening. While it's possible to detect cheating and trigger an abort after summing up all partial signatures, but identifying the malicious party remains a challenge.

In the case of a robust identity-based threshold signature scheme, the algorithm DKey, which is executed by a trusted dealer, not only provides the secret keys $\{\mathsf{sk}_l\}_{l=1}^n$ but also outputs the verification keys $\{\mathsf{vk}_l\}_{l=1}^n$. Additionally, the signing protocol DSign, which involves all n parties and uses the secret keys $\{\mathsf{sk}_l\}_{l=1}^n$, also receives the verification keys $\{\mathsf{vk}_l\}_{l=1}^n$. It's worth noting that in a robust identity-based threshold signature system, when $t+1$ parties involve in the DSign protocol, it can achieve the identifiable-abort property. Lastly, it's important to highlight that the identity-based threshold signature with abort only requires a single honest signer to maintain security, whereas the identifiable-abort and robust versions of the signature require an honest majority of participants.

The following will consider the security definitions of an identity-based threshold signature, which are unforgeability and robustness [BZ04]. Note that the attacker is assumed to be static.

Definition 2.8 (Unforgeability Against Chosen Message and Identity Attack). *Let $\mathcal{A}^{IDTHS}$ (IDTHS: IDentity-based THreshold Signature), be an attacker assumed to be a probabilistic Turing machine. Consider the following game G^{IDTHS} in which $\mathcal{A}^{IDTHS}$ interacts with the challenger $\mathcal{C}^{IDTHS}$.*

Phase 1. The challenger runs the Setup *algorithm and gives $\mathcal{A}^{IDTHS}$ the resulting common parameters pp.*

Phase 2. $\mathcal{A}^{IDTHS}$ corrupts $t-1$ signature generation servers.

Phase 3. $\mathcal{A}^{IDTHS}$ issues a number of private key extraction queries, each of which consists of usk*. On receiving* usk*, the challenger runs the key extraction algorithm taking* usk *as input and obtains a corresponding private key x. The challenger gives x to $\mathcal{A}^{IDTHS}$.*

Phase 4. $\mathcal{A}^{IDTHS}$ submits a target identity $\mathsf{usk}^\star$*. On receiving* $\mathsf{usk}^\star$*, the challenger runs the key extraction algorithm taking* $\mathsf{usk}^\star$ *as input and obtains a corresponding private key $x^\star$. Subsequently, it runs the private key distribution algorithm taking $x^\star$ as input to share it among n signature generation servers. We denote the key shares by $x^{\star,i}$ for $i=1,\cdots,n$. The challenger gives $x^{\star,i}$ for $i=1,\cdots,t-1$, (private keys for the corrupted servers) to $\mathcal{A}^{IDTHS}$.*

Phase 5. $\mathcal{A}^{IDTHS}$ issues a number of signature generation queries, each of which consists of a message denoted by m. On receiving m, the challenger, on behalf of the uncorrupted servers, runs the signature generation algorithm taking $x^{\star,i}$ for $i=t,\cdots,n$ and m as input, and responds to $\mathcal{A}^{IDTHS}$ with σ output by the signature generation algorithm . Note that in this phase, $\mathcal{A}^{IDTHS}$ is allowed to issue private key extraction queries (identities) except for $\mathsf{usk}^\star$*. $\mathcal{A}^{IDTHS}$ is allowed to see partial signature broadcast during the execution.*

Phase 6. $\mathcal{A}^{IDTHS}$ *outputs* $(\mathsf{usk}^\star, \tilde{m}, \tilde{\sigma})$*, where* $\tilde{\sigma}$ *is a valid signature of the identity* $\mathsf{usk}^\star$ *on the message* $\tilde{m}$ *. A restriction here is that* $\mathcal{A}^{IDTHS}$ *must not make a private key extraction query for* $x^\star$ *and it must not make a signature generation query for* $\tilde{m}$*. We denote* $\mathcal{A}^{IDTHS}$*'s success by*

$$Succ^{UF-IDTHS-CMA}_{IDTHS,\mathcal{A}^{IDTHS}}(k) = \Pr[\mathsf{Verify}(pp, x^\star, \tilde{m}, \tilde{\sigma}) = 1].$$

We denote by

$$Succ^{UF-IDTHS-CMA}_{IDTHS,\mathcal{A}^{IDTHS}}(t, q_e, q_s)$$

the maximum of the attacker $\mathcal{A}^{IDTHS}$*'s success over all attackers* $\mathcal{A}^{IDTHS}$ *having running time* t_2 *and making at most* q_e *key extraction queries and* q_s *signature generation queries. The ID-based threshold signature scheme is said to be* (t, q_e, q_s, ϵ)*-UF-IDTHS-CMA secure if*

$$Succ^{UF-IDTHS-CMA}_{IDTHS,\mathcal{A}^{IDTHS}}(t, q_e, q_s) < \epsilon.$$

Definition 2.9 (Abort). *A* (t, n) *ID-based threshold signature scheme is said to be secure with abort if parties abort in the presence of an attacker that makes the corrupted signature generation servers deviate from the normal execution.*

Definition 2.10 (Robustness). *A* (t, n) *ID-based threshold signature scheme is said to be robust if it computes a correct output even in the presence of an attacker that makes the corrupted signature generation servers deviate from the normal execution.*

Note that in the case of the abort scenario, the verification process solely focuses on assessing the validity of the final signature. This final signature is acquired by combining the partial signatures generated by individual parties. It either accepts the signature as valid or aborts the process, all without singling out the malicious party. Conversely, in the robust case, the parties not only validate the final signature but also verify the individual signatures (referred to as "openings") to identify and disqualify any malicious participants. This verification process ensures the guaranteed delivery of the output.

In terms of achieving active security with either an *abort* or *robustness* approach, these definitions closely resemble the security criteria found in regular threshold signature schemes [GJKR96, CKM23]. However, when it comes to the algorithms involved, identity-based threshold signatures differ from regular threshold signatures. In identity-based schemes, a master secret key is employed to generate specific secret keys for each ID, followed by an algorithm that distributes these ID-specific secret keys to the corresponding ID holders. In contrast, regular threshold signatures utilize a single secret key that is shared among all n participating parties.

Commitment Schemes. In our protocols, we assume parties have access to a commitment functionality $\mathcal{F}_{\mathsf{Commit}}$, which allows one party to commit, and later open the value to a set of parties. We assume the opened value is only available to the targeted receivers and is sent over a secure communication channel. The description of $\mathcal{F}_{\mathsf{Commit}}$ is provided in Fig. 2, which can be easily implemented in the random oracle model.

Init: Given (Init, P_i, B) from all parties, this initializes a commitment functionality from party P_i to the parties in B. This is shown with $\mathcal{F}_{\mathsf{Commit}}^{i,B}$, if B is a singleton set $B = \{j\}$ then we write $\mathcal{F}_{\mathsf{Commit}}^{i,j}$, and if $B = \mathcal{P} \setminus \{i\}$ then we write $\mathcal{F}_{\mathsf{Commit}}^{P_i}$.
Commit: On input of $(\mathsf{Commit}, \mathsf{id}, \mathsf{data})$ from parties P_i and $(\mathsf{Commit}, \mathsf{id}, \perp)$ from all parties in B the functionality stores $(\mathsf{id}, \perp)$.
Open: On input of $(\mathsf{Commit}, \mathsf{id})$ from all players in $B \cup \{i\}$ the functionality retrieves the entry $(\mathsf{id}, \mathsf{data})$ and returns data to all parties in B.

Fig. 2. The Functionality $\mathcal{F}_{\mathsf{Commit}}$ [CS20].

2.4 k-MT-GAIP Distributed Key Generation

Now we recall the DKG protocol presented in the CSI-SharK framework by Atapoor et al. [ABCP23a](Fig. 3), which will be used in Sec. 4 to build our specific identity-based threshold signature scheme.

2.5 Shamir Secret Sharing.

A (t, n)-Shamir secret sharing scheme allows n parties to individually hold a share s_i of a common secret s, such that any subset of fewer than $t + 1$ parties are not able to learn any information about the secret s while any subset of at

Input: The fixed elliptic curve E_0 and a set Q of n parties.
Output: $([s_1]E_0, \ldots, [s_{k-1}]E_0)$

1. Parties individually sample $k-1$ secrets $s_i \in \mathbb{Z}_N$ shared between the parties, where $P_j \in Q$ holds $s_{1,j}, \ldots, s_{k-1,j}$ such that $s_i = \sum_{P_j \in Q} s_{i,j}$.
2. Define an ordering the players in $Q = \{P_1, \ldots, P_n\}$.
3. Each party P_j initialises an instance of $\mathcal{F}_{\mathsf{Commit}}$; call it $\mathcal{F}_{\mathsf{Commit}}^{P_j}$.
4. For $i = 1, \ldots, k-1$, each party P_j executes
   - $E_{i,P_j} \leftarrow [s_{i,j}]E_0$.
   - $\pi_{i,j}^1 \leftarrow \mathsf{NIZK}.P((E_0, E_{i,P_j}), s_{i,j})$. (Run the ID protocol for GAIP [BKV19])
   - Use $\mathcal{F}_{\mathsf{Commit}}^{P_j}$ where P_j submits input $(\mathsf{Commit}, \mathsf{id}_{P_j}, (E_{i,P_j}, \pi_{i,j}^1))$ and all other parties input $(\mathsf{Commit}, \mathsf{id}_{P_j}, \perp)$.
5. For $i = 1, \ldots, k-1$
   - Parties run $\mathcal{F}_{\mathsf{Commit}}^{P_j}$ with input $(\mathsf{Open}, \mathsf{id}_{P_j})$ & abort if $\mathcal{F}_{\mathsf{Commit}}^{P_j}$ returns abort.
   - All other players execute $\mathsf{NIZK}.V((E_0, E_{i,P_j}), \pi_{i,j}^1)$ and abort if the verification algorithm fails.
6. $E_1^0 \leftarrow E_0, E_2^0 \leftarrow E_0, \cdots, E_{k-1}^0 \leftarrow E_0$.
7. For $j = 1, \ldots, n$
   - Party P_j computes $E_1^j \leftarrow [s_{1,j}]E_1^{j-1}, \cdots, E_{k-1}^j \leftarrow [s_{k-1,j}]E_{k-1}^{j-1}$.
   - For $i = 1, \ldots, k-1$, compute $\pi_{i,j}^2 \leftarrow \mathsf{NIZK}.P((E_0, E_{i,P_j}, E_i^{j-1}, E_i^j), s_{i,j})$. (Run the argument in Fig. 10)
   - Broadcast $(E_1^j, E_2^j, \cdots, E_{k-1}^j, \pi_{1,j}^2, \ldots, \pi_{k-1,j}^2)$ to all players.
   - For $i = 1, \ldots, k-1$, all other players execute $\mathsf{NIZK}.V(E_0, E_{i,P_j}, E_i^{j-1}, E_i^j)$ and abort if the verification algorithm fails.
8. Return $(E_1^n, E_2^n, \ldots, E_{k-1}^n) = ([s_1]E_0, [s_2]E_0, \cdots, [s_{k-1}]E_0)$.

Fig. 3. Full-threshold k-MT-GAIP distributed key generation protocol [ABCP23a].

least $t+1$ parties are able to efficiently reconstruct the common secret s via Lagrange interpolation by computing $s = f(0) = \sum_{i \in S} s_i \cdot L_{0,i}^S$, where

$$L_{0,i}^S := \prod_{j \in S \setminus \{i\}} \frac{j}{j-i} \pmod N,$$

are Lagrange basis polynomials evaluated at 0. Any subset of fewer than $t+1$ parties are not able to find $s = f(0)$, as this is information-theoretically hidden, even given t shares. Since we will be working over the ring $\mathbb{Z}_N$ with N composite, the difference $j - i$ of any two elements in $i, j \in S$ must be invertible modulo N. If q' is the smallest prime factor of N, it is enough to require that $n < q'$. This is indeed the case of our applications.

3 Identity-Based Signatures From CSI-SharK

In this section, we modify the CSI-FiSh-based identity-based signature of Shaw and Dutta [SD21] to work with the CSI-SharK signature scheme [ABCP23a]. The primary benefit inherent in this adaptation arises from the singular nature of the secret key of CSI-SharK . The process of adapting Shaw and Dutta's signature [SD21] to the CSI-SharK signature is mostly alterations in the Setup and $\mathsf{Extract}$ algorithms. The Setup algorithm generates pp and a master secret key msk, where msk consists of S_0 different coefficients of a single secret value s. Since the soundness rate of the underlying ID protocol is $\frac{1}{1+S_0}$, one needs to amplify the soundness error rate by repeating the protocol T_0 times. The $\mathsf{Extract}$ algorithm gets $\mathsf{msk} := s$ and generates a new user secret key $\mathsf{usk}_{\mathsf{id}_{i,j}}$ for a specific id. The length of the user public key is S_1 and since the soundness rate of the protocol now is $\frac{1}{1+S_1}$, one needs to amplify the soundness error rate by repeating the protocol T_1 times. Note that both the Setup and $\mathsf{Extract}$ algorithms are executed by a trusted authority. Bellow, we describe the algorithms of the resulting identity-based signature scheme:

Setup ***Algorithm.*** Given the security parameter 1^λ, act as follows,

1. Select the integers $T_0, T_1, S_0 = 2^{\gamma_0} - 1$ and $S_1 = 2^{\gamma_1} - 1$ where γ_0, γ_1 are integers and $T_0 < S_0, T_1 < S_1$
2. Sample a cryptographic hash function $H : \{0,1\}^\star \to [0, S_0]^{T_0 S_1}$ and a public (super) exceptional set $\Xi_{S_0} := \{c_0 = 0, c_1 = 1, c_2, \cdots, c_{S_0}\}$
3. Sample $s \leftarrow\!\$\, \mathbb{Z}_N$,and for $i = 1$ to S_0 set: $E_i = [c_i s]E_0$
4. Return $pp = \{E_0, T_0, T_1, S_0, S_1, H,$
   $\Xi_{S_0} := \{c_0 = 0, c_1 = 1, c_2, \cdots, c_{S_0}\}, \{E_i\}_{i=1}^{S_0}\}$ and $\mathsf{msk} = s$.

$\mathsf{Extract}$ ***Algorithm.*** Given $(pp, \mathsf{msk}, \mathsf{id})$ act as follows,

1. Set $s_0 \leftarrow 0$ and for $i = 1$ to T_0, $j = 1$ to S_1: $r_{i,j} \leftarrow\!\$\, \mathbb{Z}_N$ and $R_{i,j} = [r_{i,j}]E_0$
2. Compute $u \leftarrow H(\mathsf{id} || \{R_{i,j}\}_{i=1,j=1}^{T_0,S_1})$
3. Parse u as $\{u_i \in [0, s_0]\}_{i=1}^{T_0 S_1}$ and Ξ_{S_0} as $\{c_0 = 0, c_1 = 1, c_2, \cdots, c_{S_0}\}$
4. For $i = 1$ to T_0, $j = 1$ to S_1 open: $x_{i,j} = r_{i,j} - c_{u_i} s \mod N$
5. Return $\mathsf{usk}_{\mathsf{id}_{i,j}} = (\{u_i\}_{i=1}^{T_0 S_1}, \{x_{i,j}\}_{i=1,j=1}^{T_0,S_1})$.

Sign ***Algorithm.*** Given $(pp, \mathsf{usk}_{\mathsf{id}_{i,j}}, m)$, act as follows,

1. Parse $\mathsf{usk}_{\mathsf{id}_{i,j}}$ to $(\{u_i\}_{i=1}^{T_0 S_1}, \{x_{i,j}\}_{i=1,j=1}^{T_0,S_1})$
2. For $i = 1$ to T_0 set: $x_{i,0} \leftarrow 0$
3. For $i = 1$ to T_0 do:
   (a) For $j = 1$ to S_1 compute: $X_{i,j} = [x_{i,j}]E_{u_i}$
4. For $i = 1$ to T_0 do:
   (a) For $j = 1$ to T_1 sample: $k_{i,j} \leftarrow\$ \mathbb{Z}_N$ and compute $K_{i,j} = [k_{i,j}]E_{u_i}$
5. Compute $v \leftarrow H'(m||\{K_{i,j}\}_{i=1,j=1}^{T_0,T_1})$
6. Parse v as $\{v_{i,j} \in [0, S_1]\}_{i=1,j=1}^{T_0,T_1}$
7. For $i = 1$ to T_0 do:
   (a) For $j = 1$ to T_1 open: $z_{i,j} = k_{i,j} - x_{i,v_{i,j}} \mod N$
8. Return $\sigma \leftarrow (\{z_{i,j}\}_{i=1,j=1}^{T_0,T_1}, \{X_{i,j}\}_{i=1,j=1}^{T_0,S_1}, v)$.

Verify ***Algorithm.*** Given $(pp, \mathsf{id}, m, \sigma)$, act as follow,

1. Retrieve $u = H(\mathsf{id}||\{X_{i,j}\}_{i=1,j=1}^{T_0,S_1})$ and parse v as $\{v_{i,j}\}_{i=1,j=1}^{T_0,T_1}$
2. For $i = 1$ to T_0 do:
   (a) For $j = 1$ to T_1 if: $v_{i,j} = 0$ then $K'_{i,j}[z_{i,j}]E_{u_i}$, else $K'_{i,j}[z_{i,j}]X_{i,v_{i,j}}$
3. Compute $v' \leftarrow H'(m||\{K'_{i,j}\}_{i=1,j=1}^{T_0,T_1})$
4. If $v' \neq v$ then return "Invalid" otherwise return "valid".

Correctness. The correctness of the resulting identity-based signature follows from the correctness of the CSI-SharK signature and the underlying identity-based ID protocol (reviewed in App. A.1, in Fig. 9).

Theorem 3.1. *Let IDS be the identity-based signature scheme outlined above. Let $\mathcal{A}$ be an adversary that breaks the UF-IDS-CMA security of IDS (defined in Def. 2.6). Then we can construct an impersonator I breaking the IMP-PA security (IMPersonation under Passive Attacks, defined in [SD21, Definition 6.12]) of the underlying ID protocol of the signature scheme (reviewed in Fig. 9).*

Efficiency. The efficiency of the resulting identity-based signature scheme is close to the original version, presented by Shaw and Dutta [SD21] except that, in contrast to the original scheme based on CSI-SharK, in our case the master secret key is a single element of $\mathbb{Z}_N$, rather than S_0 elements (of $\mathbb{Z}_N$).

4 Identity-Based Threshold Signature with Abort

This section presents an identity-based threshold signature scheme based on isogenies, using the CSI-SharK signature scheme [ABCP23a]. The identity-based threshold signature ensures security with abort, meaning that if any issues arise and the protocol cannot be followed, the involved parties will abort and cease the procedure. In Sec. 3, we described an identity-based signature using CSI-SharK which consists of PPT algorithms (Setup, Extract, Sign, Verify), where a

single signer is involved. Using the mentioned identity-based signature scheme, we build an identity-based threshold signature scheme consisting of five PPT algorithms ($\mathsf{Setup}, \mathsf{Extract}, \mathsf{DKey}, \mathsf{DSign}, \mathsf{Verify}$) (described in Def. 2.7). In the resulting threshold scheme, the algorithms ($\mathsf{Setup}, \mathsf{Extract}, \mathsf{Verify}$) are identical to the non-threshold case, where ($\mathsf{Setup}, \mathsf{Extract}$) are executed by a trusted authority. Due to this fact, to conserve space, we do not re-write the algorithms of ($\mathsf{Setup}, \mathsf{Extract}, \mathsf{Verify}$). The threshold variant includes an additional algorithm DKey, which is responsible for sharing the user secret key returned by the $\mathsf{Extract}$ algorithm. Similar to the non-threshold case, in the threshold version, we assume that a trusted authority runs the algorithms ($\mathsf{Setup}, \mathsf{Extract}, \mathsf{DKey}$). The trusted authority will employ the DKey algorithm to distribute the secret key $\mathsf{usk}_{\mathsf{id}_{i,j}}$ corresponding to id among all the parties sharing the same id. In this paper, we use the well-known Shamir's secret sharing scheme, although it's worth noting that alternative secret sharing schemes are also viable options. Next, we describe the procedures of the DKey and DSign algorithms:

DKey*:* The description of DKey is given in Fig. 4 which a dealer samples a random degree-t polynomial and uses Shamir's secret sharing to distribute the user's secret key $\mathsf{usk}_{\mathsf{id}_{i,j}}$ among multiple parties and sends the shares $x_{i,j,l}$ to each party via the secure channels.

Input: $(pp, \mathsf{usk}_{\mathsf{id}_{i,j}} = (\{u_i\}_{i=1}^{T_0 S_1}, \{x_{i,j}\}_{i=1,j=1}^{T_0,S_1}), n, t)$, where n is the number of parties and t is threshold parameter ($t+1$ parties can reconstruct the secret).
Output: Private $\{x_{i,j,l}\}_{l=1,i=1,j=1}^{l=n,i=T_0,j=S_1}$ and Public $\{F_{i,j,l}\}_{l=1,i=1,j=1}^{l=n,i=T_0,j=S_1}$.

An authority acts as follows:

1. Parse $\mathsf{usk}_{\mathsf{id}_{i,j}} = (\{u_i\}_{i=1}^{T_0 S_1}, \{x_{i,j}\}_{i=1,j=1}^{T_0,S_1})$
2. For $i = 1$ to T_0, $j = 1$ to S_1 do:
   (a) Sample a degree-t polynomial $f_{i,j}(X) = x_{i,j} + c_{i,j,1}X^1 + \cdots + c_{i,j,t}X^t$.
   (b) For $l = 1$ to n: set $x_{i,j,l} = f_{i,j}(l)$ as a secret for each party.
3. For $l = 1, \cdots, n$: send $x_{i,j,l}$ to party P_l securely.

Fig. 4. DKey algorithm for the proposed identity-based threshold signature.

DSign*:* Given the public parameters pp, the message m, the secret keys of each party $x_{i,j,l}$, the distributed signing algorithm DSign is employed to generate a signature. This algorithm is executed by a qualified set of parties $Q = \{1, \cdots, q\}$ and if any individual fails to adhere to the designated protocol, the honest parties will collectively abort the process and cease the ongoing computation. The description of DSign is given in Fig. 5. As it can be seen, during Step 3, the parties collaboratively compute the public key and determine $E_{i,j,L} = [x_{i,j,l}]E_{i,j,l-1}$, where $E_{i,j,l}$ is shared among the parties. This computation involves updating the curve $E_{i,j,l}$ with their respective shares $x_{i,j,l}$ in a round-robin way and providing a proof, using the Non-Interactive Zero-Knowledge (NIZK) argument (summarized in the App. A.2, in Fig. 10), to demonstrate that they correctly updated E_{u_i} with the secret $x_{i,j,l}$ received from the authority. Subsequently, the parties

must verify the proofs provided by all other parties using the verification process outlined in Fig. 10. Given that the process is executed in a round-robin manner, the final update is performed by the last participant in the qualified set Q. As outlined in Step 3b, the update contributed by party P_q yields the conclusive public key $E_{i,j} := E_{i,j,q}$. In Step 4, the involved parties execute the full-threshold DKG protocol (depicted in Fig. 3) for $k = 2$, and calculate the value of $K_{i,j}$. Note that when implementing this in practice, it is possible to parallelize these executions and loops. In Step 5, the challenge v is generated by hashing the concatenation of message m and $K_{i,j}$ computed in the previous step. Then, in Step 7, parties open their response $z_{i,j,l}$ and locally add them all together and achieve $z_{i,j}$. Finally, the algorithm returns the signature σ which consists of $(z_{i,j}, E_{i,j}, v)$.

Next, we argue the security of our new identity-based threshold signature scheme. To this end, in Fig. 6, we first describe the distributed signature functionality $\mathcal{F}_{\mathsf{DSign}}$ and then simulate our proposed threshold signing protocol.

Security of DKey***:*** We highlight that the DKey algorithm is executed by a trusted dealer and does not need to be simulate.

Input: $(pp, m, \{x_{i,j,l}\}_{l=1,i=1,j=1}^{l=q,i=T_0,j=S_1})$,
Output: $\sigma \leftarrow (\{z_{i,j}\}_{i=1,j=1}^{T_0,T_1}, \{E_{i,j}\}_{i=1,j=1}^{T_0,S_1}, v)$.

Signing Algorithm DSign**:** a qualified set of parties $\{P_1, \cdots, P_q\}$ act as follows,

1. For $l = 1$ to q each party for $i = 1$ to T_0: set $x_{i,0,l} \leftarrow 0$
2. For $i = 1$ to T_0 and $j = 1$ to S_1: set $E_{i,j,0} = E_0$
3. For $i = 1$ to T_0 and $j = 1$ to S_1 do:
   (a) For $l = 1$ to q do:
      i. Party P_l computes $E_{i,j,l} \leftarrow [x_{i,j,l}]E_{i,j,l-1}$
      // Parties use the NIZK argument (from App. A.2 Fig. 10), to prove they are updating the curve with the secret which they got from the dealer.
      ii. Compute $\pi_{i,j,l} \leftarrow \mathsf{NIZK}.P((E_0, F_{i,j,l}, E_{i,j,l-1}, E_{i,j,l}), x_{i,j,l})$
      iii. Broadcast $(E_{i,j,l}, \pi_{i,j,l})$
      // Parties use the verifier of NIZK argument (from App. A.2 Fig. 10), to verify the proof.
      iv. All players execute $\mathsf{NIZK}.V((E_0, F_{i,j,l}, E_{i,j,l-1}, E_{i,j,l}), \pi_{i,j,l})$ and abort if the verification algorithm fails.
   (b) Set $E_{i,j} = E_{i,j,q}$ and return $E_{i,j}$
4. For $i = 1$ to T_0 and $j = 1$ to T_1: given E_{u_i}, parties of the qualified set run Full-threshold 2-MT-GAIP given in Fig. 3, and generate $K_{i,j} = [k_{i,j}]E_{u_i}$
5. Compute $v \leftarrow H'(m||\{K_{i,j}\}_{i=1,j=1}^{T_0,T_1})$
6. Parse v as $\{v_{i,j} \in [0, S_1]\}_{i=1,j=1}^{T_0,T_1}$
7. For $i = 1$ to T_0, $j = 1$ to T_1, and $l = 1$ to q: party P_l opens $z_{i,j,l} = k_{i,j,l} - x_{i,v_{i,j},l} \mod N$
8. For $i = 1$ to T_0, $j = 1$ to T_1: parties compute $z_{i,j} = \sum_{l=1}^{q}(z_{i,j,l})$

Return $\sigma \leftarrow (\{z_{i,j}\}_{i=1,j=1}^{T_0,T_1}, \{E_{i,j}\}_{i=1,j=1}^{T_0,S_1}, v)$.

Fig. 5. DSign algorithm for the proposed identity-based threshold signature with abort.

We let $\mathcal{A}$ denote the set of parties controlled by the adversary.

Sign: On input of a message m the functionality proceeds as follows:

1. The functionality adversary waits for an input from the adversary.
2. If the input is not abort then the functionality generates a signature σ on the message m.
3. The signature is returned to the adversary, and the functionality again waits for input. If the input is again not abort then the functionality returns σ to the honest players.

Fig. 6. Distributed signature functionality $\mathcal{F}_{\mathsf{DSign}}$ [CS20].

Theorem 4.1. *The (t, n)-identity-based threshold signing protocol described in Fig. 5 is UF-IDTHS-CMA secure with abort in the quantum random oracle model (the hash functions are modelled as quantum random oracles), against a static adversary corrupting up to t parties, with $t < n/2$, if the identity-based signature scheme proposed in Sec. 3 is EUF-IDS-CMA secure.*

Proof. In theorem 3.1, we showed that the IDS scheme proposed in Sec. 3 is EUF-IDS-CMA secure. Next, we show that the DSign protocol presented in Fig. 5 securely implement the functionality $\mathcal{F}_{\mathsf{DSign}}$ (given in Fig. 6) in the $\mathcal{F}_{\mathsf{Commit}}$-hybrid model against an active adversary corrupting up to t parties.

DSign ***Simulation:*** The proof is analogous to that of Theorem 4.3 in Atapoor et al. [ABCP23a]. One key difference is in the case of Atapoor et al. [ABCP23a], parties get the commitment to $x_{i,j}$ from the distributed key generation phase, while in our case the trusted dealer publishes the commitments to $x_{i,j}$. Let P_l be the honest party. $\mathcal{A}$ and $\mathcal{S}$ engage in an execution of the DSign protocol in Fig. 5. The authority has committed to the secret shares of $x_{i,j}$. Now $\mathcal{A}$ and $\mathcal{S}$ proceed with the round-robin protocol for computing the public keys as in Step 3 of Fig. 5. All steps for honest players can be simulated exactly by following the real protocol, except for the party P_l which holds the unknown shares $x_{i,j,l}$ for $i = 1, \cdots, T_0$ and $j = 1, \cdots, S_1$. The input to this party in execution l will be $E_{i,j,l-1} = \left[\sum_{p=1}^{l-1} x_{i,j,p}\right] E_0$, while the output needs to be $E_{i,j,l} = \left[-\sum_{p=1}^{l-1} x_{i,j,p}\right] E_{i,j}$, so as to create the correct output curve $E_{i,j}$. The curve $E_{i,j,l}$ can thus be computed by $\mathcal{S}$ as $E_{i,j,l} := \left[-\sum_{p\neq l} x_{i,j,p}\right] E_{i,j}$. After computing $E_{i,j,l}$ the associated ZK proof can hence be simulated as well. If $\mathcal{A}$ deviates from the protocol in any way, this is caught by the ZK proofs and $\mathcal{S}$ will be able to abort. Thus if abort does not happen in the protocol, the simulator will output the same curve $E_{i,j}$.

Again in Step 4 of the Fig. 5, parties are running a full-threshold 2-MT-GAIP protocol from Fig. 3 to jointly compute $K_{i,j} = [k_{i,j}]E_{u_i}$. Next, w.l.o.g., we write the simulation for particular values of i, j, while it can also be extended for all $i = 1, \cdots, T_0$, and $j = 1, \cdots, T_1$. Let P_l be the honest party. $\mathcal{A}$ and $\mathcal{S}$ engage in an execution of the full-threshold 2-MT-GAIP protocol in Fig. 3. As each party needs to commit to its secret share of s, the simulator commits to a random share s_l^*, say $K_{P_l}^* = [s_l^*]E_{u_i}$, produces a simulated proof and then

commits to the curve and proof using the commitment scheme. If later the simulator is asked to open this, the simulator will equivocate the commitment so that it can be opened to the correct elliptic curve and proofs. Note that, the simulator is able to compute them after extracting the adversarial shares. From the $\pi^1_{P_p}$ (in Fig. 3), given in the commit phase, $\mathcal{S}$ is able to extract the values s_p entered by $\mathcal{A}$ in the first round of proofs. The extracted values s_p are now passed to the functionality, which completes them to a valid set of shares of the secret and returns the corresponding curves E_{u_i}, K. At this point, $\mathcal{S}$ has all the adversarial shares and the curves E_{u_i}, K. The honest share s_l is unknown to $\mathcal{S}$. Even though it does not have the honest share, it can fake the commitment by setting $K_{P_l} := \left[-\sum_{p\neq l} s_p\right] K$ which it can do by using the curves it got from the functionality and the adversarial shares that it got from the proofs-of-knowledge. Having E_{u_i} and K_{P_l}, $\mathcal{S}$ can simulate the corresponding proof. It then commits to this proof using $\mathcal{F}^{P_l}_{\mathsf{Commit}}$. The commitments can now be opened. Now $\mathcal{A}$ and $\mathcal{S}$ proceed with the round-robin protocol for computing the public keys as in Step 7 of the Fig. 3. All steps for honest players can be simulated exactly by following the real protocol, except for the party P_l which holds the unknown share s_l. The input to this party in execution l will be $K^{l-1} = \left[\sum_{p=1}^{l-1} s_p\right] E_{u_i}$ while the output needs to be $K^l = \left[-\sum_{p=1}^{l-1} s_p\right] K$, so as to create the correct output K. The curve K^l can thus be computed by $\mathcal{S}$ like it did for computing K_{P_l} and the associated ZK proof can hence be simulated as well. If $\mathcal{A}$ deviates from the protocol, this is caught by the ZK proofs and $\mathcal{S}$ will be able to abort.

In our simulation of full-threshold 2-MT-GAIP generation protocol the value $k_{i,j,l}$ is unknown and 'fixed' by the implicit equation given by the signature $(\{z_{i,j}\}_{i=1,j=1}^{T_0,T_1}, \{X_{i,j}\}_{i=1,j=1}^{T_0,S_1}, v)$ returned by the functionality which gives us $K_{i,j} = [k_{i,j}]E_{u_i} = [z_{i,j}]X_{i,v_{i,j}}$, where $v_{i,j}$ is the random challenge value obtained from the quantum random oracle, i.e., $v \leftarrow H'(m||\{K_{i,j}\}_{i=1,j=1}^{T_0,T_1})$. The final part of the signature which needs to be simulated is the output of $z_{i,j,l}$. We know what $\mathcal{A}$ *should* output and hence can define $z_{i,j,l} = z_{i,j} - \sum_{l\neq l'} z_{i,j,l'}$. If $\mathcal{A}$ deviates from the protocol in the final step and uses an invalid value of $z_{i,j,l'}$, then the adversary will learn the signature, but the honest players will abort; which realizes the ideal functionality described in Fig. 6. □

5 Robust Identity-Based Threshold Signature Scheme

Our threshold signature from the previous section does not provide a guarantee for output delivery. The protocol is susceptible to the Denial-of-Service (DoS) attack, which allows malicious parties to indefinitely deny the generation of the desired result. In this section, we extend our identity-based threshold signature from Sec. 4 to achieve robustness and assure output delivery. Our robust scheme is also build based on the CSI-SharK signature, however with some changes, at the cost of a longer master secret key, can be adapted to work with the CSI-FiSh signature as well. In the proposed robust identity-based threshold signature if

An authority runs DKey:

1. Parse $\mathsf{usk}_{\mathsf{id}_{i,j}} = (\{u_i\}_{i=1}^{T_0 S_1}, \{x_{i,j}\}_{i=1,j=1}^{T_0,S_1})$
2. For $i = 1$ to T_0, $j = 1$ to S_1 do:
   (a) Choose a random degree t poly $f_{i,j}(X) = x_{i,j} + c_{i,j,1}X^1 + \cdots + c_{i,j,t}X^t$
   (b) for $l = 1$ to n: set $x_{i,j,l} = f_{i,j}(l)$ as a secret for each party and create the corresponding commitment $F_{i,j,l} = [x_{i,j,l}]F_0$ for each secret $x_{i,j,l}$
      i. Choose a random degree t poly $g_{i,j,l}(Y) = x_{i,j,l} + d_{i,j,1}Y^1 + \cdots + d_{i,j,t}Y^t$ and reshare secrets $x_{i,j,l}$ obtained from the previous step
      ii. for $k = 1$ to n: set $w_{i,j,l,k} = g_{i,j,l}(k)$ as a new secret for each party's secret $x_{i,j,l}$.
3. For $p = 1, \cdots, n$: send all the related $(x_{i,j,l}, w_{i,j,l,k})_{i=1,j=1,l=1,k=1}^{i=T_0,j=S_1,l=n,k=n}$ to party P_p securely and publish $\{F_{i,j,l}\}_{i=1,j=1,l=1}^{i=T_0,j=S_1,l=n}$.

Fig. 7. DKey algorithm for the proposed robust identity-based threshold signature.

any dishonest behaviour by a participant occurs, the efforts invested thus far are not rendered futile. The involved parties are able to identify the malicious parties, exclude them from the protocol, reconstruct their shares, and seamlessly continue the protocol to achieve the correct output. In the rest of section, we are highlighting the distinctions between the signature discussed in Sec. 4 and the new one, without reiterating the similarities.

DKey*:* In Step 2 of the new DKey protocol (given in Fig. 7), in addition to generating the polynomial $f_{i,j}(X)$ for sharing the secret $x_{i,j}$, the authority generates another polynomial $g_{i,j(Y)}$ to re-share each share of $x_{i,j,l}$ among all the parties. At the end, the authority privately sends a share of this re-sharing, denoted as $w_{i,j,l,k}$, to each party, along with their respective share of $x_{i,j,l}$. Later, we show how the parties use these values to verify the opening responses in Step 7 of DSign algorithm (described in Fig. 8). This approach originally is used in the ThreshER SharK scheme [ABCP23c] to achieve robustness. However, it is not directly applicable within the context of our scheme. We will provide a detailed explanation as we proceed through the description of the DSign protocol.

DSign*:* Fig. 8 describes the procedure of our proposed robust DSign protocol. Compared to the DSign protocol from Fig. 5, in the robust DSign, the first key difference is that we need all the parties to be present in the signing procedure, as explained in Step 7 of Fig. 8. Steps 1-3b are the same as our previous DSign protocol. Then, in Step 4, parties run the robust Distributed Key Generation (DKG) protocol CSI-RAShi++ from [ABCP23c], to compute $K_{i,j}$. The robust DKG protocol CSI-RAShi++ works with Shamir secret sharing and is recently proposed as an improved version of the DKG protocol CSI-RAShi [BDPV21]. CSI-RAShi++ allows a set of parties to sample $[x]E$ in a fully distributed manner, such that at the end, each party gets a Shamir share of x. Using CSI-RAShi++ and our re-sharing from the DKey step allows us to achieve robustness. Creating the challenge in Step 6 remains the same as before. Step 7 is the subtle part of the protocol to achieve the robustness. In this step, we use the reshares of the shares of all parties (from the DKey step) along with the reshares

Signing Algorithm DSign**:** All Parties $\{P_1, \cdots, P_n\}$ act as follows,

1. For $l = 1$ to n party P_l for $i = 1$ to T_0: set $x_{i,0,l} \leftarrow 0$
2. Set $E_{i,j,0} = E_0$
3. For $i = 1$ to T_0 and $j = 1$ to S_1 do:
   (a) For $l = 1$ to n do:
      i. Party P_l computes $E_{i,j,l} \leftarrow [x_{i,j,l}]E_{i,j,l-1}$
      ii. Compute $\pi_{i,j,l} \leftarrow \mathsf{NIZK}.P((E_0, F_{i,j,l}, E_{i,j,l-1}, E_{i,j,l}), x_{i,j,l})$, using the NIZK argument (from App. A.2 Fig. 10),
      iii. Broadcast $(E_{i,j,l}, \pi_{i,j,l})$
      iv. All players execute $\mathsf{NIZK}.V((E_0, F_{i,j,l}, E_{i,j,l-1}, E_{i,j,l}), \pi_{i,j,l})$(from App. A.2 Fig. 10), and abort if the verification algorithm fails.
   (b) Set $E_{i,j} = E_{i,j,n}$ and return $E_{i,j}$
4. For $i = 1$ to T_0, $j = 1$ to T_1 , and For $l = 1$ to n, given E_{u_i}, parties run the DKG of CSI-RAShi++ Fig. 12 and generate $K_{i,j} = [k_{i,j}]E_{u_i}$.
   // Note that $b_{i,j,l}(X)$ which is a degree t polynomial sampled by parties during the DKG protocol of CSI-RAShi++ and the degree t polynomial of $g_{i,j}$ which was sampled by a trusted dealer in the DKey algorithm for re-sharing the share of the parties, both are using for checking and robustness.
5. Compute $v \leftarrow H'(m||\{K_{i,j}\}_{i=1,j=1}^{T_0,T_1})$
6. Parse v as $\{v_{i,j} \in [0, S_1]\}_{i=1,j=1}^{T_0,T_1}$
7. For $i = 1$ to T_0, $j = 1$ to T_1, and $l = 1$ to n do:
   (a) Each party P_l computes $z_{i,j,l}(Y) = b_{i,j,l}(Y) - g_{i,v_{i,j},l}(Y)$
   (b) Using their secret value shared during the NI-VSS protocol (from App. A.2 Fig. 10), namely $b_{i,j,l}(l)$ and $g_{i,j}^{l}(l)$ given by the authority, each party P_l' verifies

$$z_{i,j,l}(l') \stackrel{?}{=} b_{i,j,l}(l') - g_{i,v_{i,j},l}(l')$$

   (c) Whenever one of these checks fails, P_l' broadcasts a complaint against P_l When a player P_l has $t+1$ or more complaints against them, they are disqualified. The remaining players can then construct $z_{i,j,l}(0)$ by reconstructing both $b_{i,j,l}(0)$ and $g_{i,j,l}(0)$ using the information from the DKG and given by the authority. This is always possible when there are at least $t+1$ honest parties (honest majority).
   (d) Using $\{z_{i,j,l}(0)\}_{i=1}^{n}$, parties build the responses $z_{i,j}(0) = \sum_{i \in Q} z_{i,j,l}(0)$
8. Return $\sigma \leftarrow (\{z_{i,j}\}_{i=1,j=1}^{T_0,T_1}, \{E_{i,j}\}_{i=1,j=1}^{T_0,S_1}, v)$.

Fig. 8. DSign algorithm for the proposed robust identity-based threshold signature.

generated during the execution of CSI-RAShi++ DKG protocol, and verify the partial openings of individual parties. In this step, parties open a polynomial instead of a value. In the abort version, when the parties open a value it gives the possibility to find the misbehaviour and abort but they can not identify the malicious parties. But in this case, since we are in the honest majority setting, due to opening a polynomial, the parties can identify an adversary (using the reshares from the DKey and DKG steps) and disqualify him. Then, they can reconstruct his share and continue the computation until the end. Finally, par-

ties sum all the responses up and achieve the final $z_{i,j}$. The signature as before consists of $(\{z_{i,j}\}_{i=1,j=1}^{T_0,T_1}, \{E_{i,j}\}_{i=1,j=1}^{T_0,S_1}, v)$.

Theorem 5.1. *The (t, n)-identity-based threshold signing protocol described in Fig. 8, is UF-IDTHS-CMA secure and robust in the quantum random oracle model (the hash functions are modelled as quantum random oracles), against a static adversary corrupting up to t parties, with $t < n/2$, if the identity-based signature scheme proposed in Sec. 3 is EUF-IDS-CMA secure.*

Security Proofs. The security of DKey can be argued similar to the abort construction, given in Sec. 4, and the simulation of DSign is analogous to the proof of theorem 5.1, which is omitted. We highlight that, in this case, one key difference is that the security of the DSign protocol relies on the security of the robust CSI-RAShi++ DKG protocol from [ABCP23c].

6 Conclusion

We initiated our work by modifying the existing identity-based signature based on isogenies, as proposed by Shaw and Dutta [SD21], in order to align it with the CSI-SharK signature scheme. Subsequently, we proposed two identity-based threshold signature schemes in the CSIDH setting. Both of the proposed signatures possess active security within the quantum random oracle model, with the first one offering security with abort, while the second one is characterized by robustness. Although these novel constructions represent theoretical outcomes, they can be considered as the first step towards the development of identity-based threshold protocols that are based on isogenies. It is worth noting that any advancements made in the underlying protocols, e.g., CSI-SharK signature, CSI-RAShi++ DKG protocol, or even improvements in the computations of group actions, can be applied to our identity-based threshold signatures as well.

Acknowledgments

We thank the anonymous reviewers for their valuable comments and suggestions. We would also like to extend our special thanks to Elizabeth Crites for shepherding the final version of the paper and her insightful and valuable comments. Additionally, we would like to acknowledge Karim Baghery for his helpful discussions during the initial phases of the paper. This work has been supported in part by the FWO under an Odysseus project GOH9718N and by CyberSecurity Research Flanders with reference number VR20192203. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Cyber Security Research Flanders or the FWO.

References

ABCP23a. 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.

ABCP23b. Shahla Atapoor, Karim Baghery, Daniele Cozzo, and Robi Pedersen. Practical robust DKG protocols for CSIDH. In Mehdi Tibouchi and Xiaofeng Wang, editors, *Applied Cryptography and Network Security - 21st International Conference, ACNS 2023, Kyoto, Japan, June 19-22, 2023, Proceedings, Part II*, volume 13906 of *Lecture Notes in Computer Science*, pages 219–247. Springer, 2023.

ABCP23c. Shahla Atapoor, Karim Baghery, Daniele Cozzo, and Robi Pedersen. VSS from distributed ZK proofs and applications. In Jian Guo and Ron Steinfeld, editors, *Advances in Cryptology - ASIACRYPT 2023 - 29th International Conference on the Theory and Application of Cryptology and Information Security, Guangzhou, China, December 4-8, 2023, Proceedings*, Lecture Notes in Computer Science. Springer, 2023.

BCP21. Karim Baghery, Daniele Cozzo, and Robi Pedersen. An isogeny-based ID protocol using structured public keys. In M.B. Paterson, editor, *Cryptography and Coding - 18th IMA International Conference, IMACC 2021, Oxford, UK, December 14-15, 2021, Proceedings*, volume 13129 of *Lecture Notes in Computer Science*, pages 179–197. Springer, 2021.

BDFLS20. Daniel Bernstein, Luca De Feo, Antonin Leroux, and Benjamin Smith. Faster computation of isogenies of large prime degree. *arXiv preprint arXiv:2003.10118*, 2020.

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, *Post-Quantum Cryptography - 12th International Workshop, PQCrypto 2021, Daejeon, South Korea, July 20-22, 2021, Proceedings*, volume 12841 of *Lecture Notes in Computer Science*, pages 257–276. Springer, 2021.

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, *Advances in Cryptology – ASIACRYPT 2019, Part I*, volume 11921 of *Lecture Notes in Computer Science*, pages 227–247, Kobe, Japan, December 8–12, 2019. Springer, Heidelberg, Germany.

BNN04. Mihir Bellare, Chanathip Namprempre, and Gregory Neven. Security proofs for identity-based identification and signature schemes. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 268–286, Interlaken, Switzerland, May 2–6, 2004. Springer, Heidelberg, Germany.

BS20. Xavier Bonnetain and André Schrottenloher. Quantum security analysis of CSIDH. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020, Part II*, volume 12106 of *Lecture Notes in Computer Science*, pages 493–522, Zagreb, Croatia, May 10–14, 2020. Springer, Heidelberg, Germany.

BZ04. Joonsang Baek and Yuliang Zheng. Identity-based threshold signature scheme from the bilinear pairings. In *International Conference on Information Technology: Coding and Computing (ITCC'04), Volume 1, April 5-7, 2004, Las Vegas, Nevada, USA*, pages 124–128. IEEE Computer Society, 2004.

CKM23. Elizabeth C. Crites, Chelsea Komlo, and Mary Maller. Fully adaptive schnorr threshold signatures. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part I*, volume 14081 of *Lecture Notes in Computer Science*, pages 678–709. Springer, 2023.

CLM+18. Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. CSIDH: An efficient post-quantum commutative group action. In Thomas Peyrin and Steven Galbraith, editors, *Advances in Cryptology – ASIACRYPT 2018, Part III*, volume 11274 of *Lecture Notes in Computer Science*, pages 395–427, Brisbane, Queensland, Australia, December 2–6, 2018. Springer, Heidelberg, Germany.

Cou06. Jean Marc Couveignes. Hard homogeneous spaces. *IACR Cryptol. ePrint Arch.*, 2006:291, 2006.

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, *Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020, Paris, France, April 15-17, 2020, Proceedings*, volume 12100 of *Lecture Notes in Computer Science*, pages 169–186. Springer, 2020.

DF17. Luca De Feo. Mathematics of isogeny based cryptography. *arXiv preprint arXiv:1711.04062*, 2017.

DG19. Luca De Feo and Steven D. Galbraith. SeaSign: Compact isogeny signatures from class group actions. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019, Part III*, volume 11478 of *Lecture Notes in Computer Science*, pages 759–789, Darmstadt, Germany, May 19–23, 2019. Springer, Heidelberg, Germany.

FS87. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology – CRYPTO'86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194, Santa Barbara, CA, USA, August 1987. Springer, Heidelberg, Germany.

GJKR96. Rosario Gennaro, Stanisław Jarecki, Hugo Krawczyk, and Tal Rabin. Robust threshold DSS signatures. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 354–371. Springer, 1996.

Hes02. Florian Hess. Efficient identity based signature schemes based on pairings. In Kaisa Nyberg and Howard M. Heys, editors, *Selected Areas in Cryptography, 9th Annual International Workshop, SAC 2002, St. John's, Newfoundland, Canada, August 15-16, 2002. Revised Papers*, volume 2595 of *Lecture Notes in Computer Science*, pages 310–324. Springer, 2002.

KH04. Kaoru Kurosawa and Swee-Huay Heng. From digital signature to id-based identification/signature. In *Public Key Cryptography–PKC 2004: 7th International Workshop on Theory and Practice in Public Key Cryptography, Singapore, March 1-4, 2004. Proceedings 7*, pages 248–261. Springer, 2004.

Kit96. Alexei Y. Kitaev. Quantum measurements and the abelian stabilizer problem. *Electron. Colloquium Comput. Complex.*, TR96-003, 1996.

KN09. Eike Kiltz and Gregory Neven. Identity-based signatures. In Marc Joye and Gregory Neven, editors, *Identity-Based Cryptography*, volume 2 of *Cryptology and Information Security Series*, pages 31–44. IOS Press, 2009.

PCZ+20. Cong Peng, Jianhua Chen, Lu Zhou, Kim-Kwang Raymond Choo, and Debiao He. Csiibs: A post-quantum identity-based signature scheme based on isogenies. *J. Inf. Secur. Appl.*, 54:102504, 2020.

Pei20. Chris Peikert. He gives C-sieves on the CSIDH. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020, Part II*, volume 12106 of *Lecture Notes in Computer Science*, pages 463–492, Zagreb, Croatia, May 10–14, 2020. Springer, Heidelberg, Germany.

RS06. Alexander Rostovtsev and Anton Stolbunov. Public-key cryptosystem based on isogenies. *IACR Cryptol. ePrint Arch.*, 2006:145, 2006.

SD21. Surbhi Shaw and Ratna Dutta. Identification scheme and forward-secure signature in identity-based setting from isogenies. In Qiong Huang and Yu Yu, editors, *Provable and Practical Security - 15th International Conference, ProvSec 2021, Guangzhou, China, November 5-8, 2021, Proceedings*, volume 13059 of *Lecture Notes in Computer Science*, pages 309–326. Springer, 2021.

Sha84. Adi Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology – CRYPTO'84*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53, Santa Barbara, CA, USA, August 19–23, 1984. Springer, Heidelberg, Germany.

Sho94. Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, 1994.

Sil09. Joseph H Silverman. *The arithmetic of elliptic curves*, volume 106. Springer Science & Business Media, 2009.

Sto10. Anton Stolbunov. Constructing public-key cryptographic schemes based on class group action on a set of isogenous elliptic curves. *Advances in Mathematics of Communications*, 4(2):215, 2010.

Vél71. Jacques Vélu. Isogénies entre courbes elliptiques. *CR Acad. Sci. Paris, Séries A*, 273:305–347, 1971.

Appendix:

A Preliminaries

Here we recall all the needed protocols which due to page limitation we could not fit into the main body of the paper.

A.1 Identity-Based Identification Protocol from Isogenies

In Fig. 9, we review the interactive identity-based identification protocol [SD21] which is based on the CSI-FiSh Signature.

Prover $P(pp, usk_{id})$, **Verifier** $V(pp, \mathsf{id})$ act as follows:

1. P:
   (a) For $i = 1$ to T_0 calculates: $x_{i,0} \leftarrow 0$
   (b) for $i = 1$ to T_0, $j = 1$ to S_1 calculates: $X_{i,j} = [x_{i,j}]E_{u_i}$
   (c) for $i = 1$ to T_0, $j = 1$ to T_1 do: $k_{i,j} \leftarrow\$ \mathbb{Z}_N$ and $K_{i,j} = [k_{i,j}]E_{u_i}$
   (d) Send $Com = \{X_{i,j}\}_{i=1,j=1}^{T_0,S_1}, \{K_{i,j}\}_{i=1,j=1}^{T_0,T_1}$ to V.
2. V: calculates $V = \{v_{i,j}\}_{i=1,j=1}^{T_0,T_1} \leftarrow\$ [0, S_1]^{T_0,T_1}$ and sends $Ch = V$ to P.
3. P: for $i = 1$ to T_0, $j = 1$ to T_1, calculates: $z_{i,j} = k_{i,j} - x_{i,j}(\mod N)$ and send $Rsp = \{z_{i,j}\}_{i=1,j=1}^{T_0,T_1}$ to V.
4. V:
   (a) calculates $u \leftarrow H(id||\{X_{i,j}\}_{i=1,j=1}^{T_0,S_1})$.
   (b) for $i = 1$ to T_0, $J = 1$ to T_0 do:
      i. $K'_{i,j} = [z_{i,j}]E_{u_i}$ if $v_{i,j} = 0$
      ii. $K'_{i,j} = [z_{i,j}]X_{i,v_{i,j}}$ if $v_{i,j} \neq 0$
   (c) if $K_{i,j} = K'_{i,j}$ return 1 else 0.

Fig. 9. Interactive ID protocol between P and V.

Correctness. To prove the correctness of the identity-based identification protocol we show that $K_{i,j} = K'_{i,j}$ for all $i = \{1, \cdots, T_0\}$ and $j = \{1, \cdots, T_1\}$. For the case when $v_{i,j} \neq 0$, we have $K'_{i,j} = [z_{i,j}]X_{i,v_{i,j}} = [k_{i,j} - x_{i,v_{i,j}} + x_{i,v_{i,j}}]E_{u_i} = K_{i,j}$ On the other hand, when $v_{i,j} = 0$, then $z_{i,j} = k_{i,j}$ as $x_{i,0}$ is set to 0. Thus, $K'_{i,j} = [z_{i,j}]E_{u_i} = [k_{i,j}]E_{u_i} = K_{i,j}$.

Theorem A.1. *The above identity-based identification scheme IDID is (t, q_I, ϵ)-secure against impersonation under passive attack as per Definition* 6.12 *of [SD21], if H is a collision-resistant hash function and the signature scheme CSI-SharK is (t', q_S, ϵ')-secure against existential unforgeability under adaptive chosen message attack where*

$$t' \approxeq 2Dt, q_I = q_S, \epsilon \leq 1 - (1 - \sqrt[2]{\epsilon'})^{\frac{1}{D}} + \frac{1}{(S_1 + 1)^{T_0 T_1}}.$$

Here D is the number of parallel executions of reset instances.

Proof. The proof is analogous to the proof of Theorem 6.21 of the identity-based identification protocol presented in [SD21]. The only difference is in the key size which is effected in the algorithms of Setup and Extract. □

A.2 Proof of Knowledge of GAIP and Commitments

We modify the ZK proof originally introduced in [CS20], and extended to the structured public keys in [ABCP23a] to suit our specific requirements. Originally designed for public keys of length k, this proof must be tailored to suit our specific purpose. The presented ZK proof in Fig. 10 is an interactive protocol, which we will later show how to make non-interactive. In their actively secure distributed

$\mathsf{ZK}.P_1(F_0, F_0', E_1, E_1')$**:** The prover does:
1. $b \leftarrow \mathbb{Z}_N$; Set $\hat{F}_0 \leftarrow [b]F_0$.
2. compute $\hat{E}_1 \leftarrow [b]E_1$. Output $(\hat{F}_0, \hat{E}_1)$.

$\mathsf{ZK}.V_1(F_0, F_0', \hat{F}_0, E_1, E_1', \hat{E}_1)$**:** The verifier acts as below:
1. If $E_1 \neq E_0$ then sample $d \leftarrow \{0, 1\}$ and output it.
2. Else sample $d \leftarrow \{-1, 0, 1\}$ and output it.

$\mathsf{ZK}.P_2((F_0, F_0', \hat{F}_0, E_1, E_1', \hat{E}_1)$**:** Given d, the prover computes $r \leftarrow b - d \cdot s \bmod N$, and outputs r.

$\mathsf{ZK}.V_2((F_0, F_0', \hat{F}_0, E_1, E_1', \hat{E}_1), d, s)$**:** The verifier returns 1 if all the following checks pass, and otherwise returns 0. Note that ${E_1'}^t$ denotes the twist of a curve E_1'.
1. If $d = -1$ return $\left([r]{F_0'}^t = \hat{F}_0\right) \wedge \left([r]{E_1'}^t = \hat{E}_1\right)$.
2. If $d = 0$ return $\left([r]F_0 = \hat{F}_0\right) \wedge \left([r]E_1 = \hat{E}_1\right)$.
3. If $d = 1$ return $\left([r]F_0' = \hat{F}_0\right) \wedge \left([r]E_1' = \hat{E}_1\right)$.

Fig. 10. The HVZK argument for proving the commitment and the well-formedness of an updated public key [CS20].

protocols to guarantee that the parties follow the protocol, they had to commit to their secret shares and prove knowledge of the committed value [ABCP23a]. This proof can be done using the basic ID protocol used in the basic form of the CSI-FiSh [BKV19], which was initially proposed by Couveignes-Rostovtsev-Stolbunov [Cou06,RS06]. Furthermore, the parties will be required to prove that they indeed act with their committed secret value on some given elliptic curve to prove the correctness of generating/updating the public key. To this end, each party will need to prove knowledge of a witness s to the following language.

$$\mathbb{L} := \left\{ \left((F_0, F_0', E_1, E_1'),\ s \right) : (F_0' = [s]F_0) \bigwedge (E_1' = [s]E_1) \quad \right\}. \tag{1}$$

We summarize the underlying Σ-protocol in Fig. 10. Similar to [CS20], we consider two variants of the summarized Σ-protocol, one when $F_0 = E_1$ which we call the Special case, and the other when this condition does not hold, which is called the General case.

Making the Protocol Non-Interactive. As mentioned, the Σ-protocol in Fig. 10 is an HVZK public coin interactive argument and can be turned into a NIZK argument in the standard manner using a hash function $G : \{0,1\}^* \to \{0,1\}^{\mathsf{sec}}$ in the General case, or $G : \{0,1\}^* \to \{-1,0,1\}^{\lceil \mathsf{sec} \log_3 2 \rceil}$ in the Special case. Using a 'slow' hash function for G, as in the case of CSI-FiSh , which is 2^h times slower than a normal hash function, we can reduce the number of repetitions to $t_{\mathsf{ZK}}^{\mathsf{General}} = \mathsf{sec} - h$ or $t_{\mathsf{ZK}}^{\mathsf{Special}} = \lceil (\mathsf{sec} - h) \log_3 2 \rceil$, respectively.[1] In the resulting NIZK argument, we denote the prover and verifier by $\mathsf{NIZK}.P$ and $\mathsf{NIZK}.V$.

[1] As an example, we can choose $h = 16$ for $\mathsf{sec} = 128$ as is done in [BCP21] and [BKV19]. This gives $t_{\mathsf{ZK}}^{\mathsf{General}} = 112$ for the General case and $t_{\mathsf{ZK}}^{\mathsf{Special}} = 71$ for the Special case.

Both the prover and the verifier need to compute a total of $2t_{\mathsf{ZK}}$ group actions throughout this protocol, ignoring the cost of the other operations, as they are negligible in comparison to group action computations. The output size of the proof is composed of the hash output and the responses. The former has a size of approximately $t_{\mathsf{ZK}}^{\mathsf{General}}$ bits ($\log_2 3^{t_{\mathsf{ZK}}^{\mathsf{Special}}} \approx t_{\mathsf{ZK}}^{\mathsf{General}}$), while the latter consists of t_{ZK} elements from $\mathbb{Z}_N$, depending on either the Special or the General case.

Lemma A.1 ([CS20]). *The two algorithms* NIZK.P *and* NIZK.V *constitute a non-interactive zero-knowledge quantum proof of knowledge in the quantum random oracle model.*

A.3 CSI-RAShi++ DKG Protocol for CSIDH-based Primitives

We recall the most efficient actively secure DKG of CSI-RAShi++ , proposed in [ABCP23c], which in the identity-based threshold signature scheme we use to compute a key in a distributed manner. In the rest, we summarize the underlying protocols of the CSI-RAShi++ DKG.

Non-interactive Zero-knowledge Proofs. Fig. 11 describes the NIZK argument which is used in the DKG protocol of CSI-RAShi++ , summarized in Fig. 12.

Prover: Given a witness polynomial $f(X) \in \mathbb{Z}_N[X]_t$, an input $x = (x_1, \cdots, x_n)$, proceed as follows and output a proof π for Shamir relation (defined in [ABCP23c]).

1. Sample $b(X) \leftarrow \mathbb{Z}_N[X]_t$ uniformly at random;
2. For $i = 1, \ldots, n$: Sample $y_i, y_i' \leftarrow \{0,1\}^\lambda$ uniformly at random; Set $\mathsf{C}_i \leftarrow \mathcal{C}(b(i), y_i)$ and $\mathsf{C}_i' \leftarrow \mathcal{C}(x_i, y_i')$;
3. Set $d \leftarrow \mathcal{H}(\mathsf{C}, \mathsf{C}')$, where $\mathsf{C} = (\mathsf{C}_1, \ldots, \mathsf{C}_n), \mathsf{C}' = (\mathsf{C}_1', \ldots, \mathsf{C}_n')$;
4. Set $r(X) \leftarrow b(X) - d \cdot f(X) \mod N$;
5. Set $\pi := (\mathsf{C}, \mathsf{C}', r(X), \{\pi_i\}_{i=1}^n)$, where $\pi_i = (y_i, y_i')$;
6. Publish $(\mathsf{C}, \mathsf{C}', r(X))$; Send individual proof $\{\pi_i = (y_i, y_i')\}_{i=1}^n$ to verifier V_i.

Verification: For $i = 1, \cdots, n$, each verifier (shareholder) i has a statement $x_i \in \mathbb{Z}_N$, and a proof $((\mathsf{C}, \mathsf{C}', r(X)), (y_i, y_i'))$. Given the set of statements and proofs for $i \in 1, \ldots, n$ the verifiers (i.e., shareholders) proceed as follows:

1. Verifier i acts as below and outputs `true` or `false`.
   (a) If $\mathsf{C}_i' \neq \mathcal{C}(x_i, y_i')$ return `false`;
   (b) Set $d \leftarrow \mathcal{H}(\mathsf{C}, \mathsf{C}')$;
   (c) If $\mathsf{C}_i == \mathcal{C}(r(i) + d \cdot x_i, y_i)$ return `true`; otherwise `false`;
2. Return `true` if *all* the verifiers return `true`; otherwise returns `false`.

Fig. 11. A Non-Interactive Threshold ZK (NI-TZK) proof scheme for Shamir secret sharing [ABCP23c].

Verifiable Secret Sharing Step: This is done using the NI-VSS scheme, presented in Fig. 13, in a standard distributed manner. Namely, each party P_i one time plays the role of the dealer in Fig. 13, samples $f^{(i)}(X)$, and then in a verifiable manner shares $f^{(i)}(0)$ with other parties. In the end, all the shareholders get a share of the joint secret key x_0, where implicitly defined as $x_0 = \sum_{i \in Q} f^{(i)}(0)$ for a qualified set Q. Each party P_j obtains its share of x_0 as $x_j = \sum_{i \in Q} f^{(i)}(j)$.
PK Computation Step: This is done as in the (public key) computation step of the DKG protocol presented in [ABCP23b, Section 3]. Parties return the public key $[x_0]E_0$.

Fig. 12. CSI-RAShi++ : an efficient DKG protocol for a single PK $[x_0]E_0$.

Non-Interactive Verifiable Secret Sharing Used in the CSI-RAShi++ DKG. The first step of the DKG scheme of CSI-RAShi++ reviewed in Fig. 12, is a Non-Interactive Verifiable Secret Sharing step which is proposed in [ABCP23c], and recalled in Fig. 13.

Initialization: Parties $P_1, \cdots, P_n$ generate system parameters and each one registers a pk to facilitate secure communications.

Share: Given n and t, to share x_0, the dealer proceeds as follows:

1. Sample a uniformly random polynomial $f(X)$ of degree t with coefficients in a ring R, subject to $f(0) = x_0$.
2. For $i = 1, 2, \cdots, n$: set $x_i := f(i)$.
3. Given $f(X)$ and $x = (x_1, \cdots, x_n)$, run the prover of NI-TZK scheme in Fig. 11, and obtain the proof $\pi := (\mathsf{C}, \mathsf{C}', r(X), \{\pi_i\}_{i=1}^n)$.
4. Send the share and the individual proof (x_i, π_i) privately to party P_i and broadcast the elements $(\mathsf{C}, \mathsf{C}', r(X))$ as the proof.

Verification: To verify the received shares, $P_1, \cdots, P_n$ utilize their shares $\{x_i\}_{i=1}^n$ and run the verifier of the NI-TZK proof scheme given of Fig. 11. If the verification of P_i fails, then P_i broadcasts a complain against the dealer. If more than t shareholders complain against the dealer, then the *Verification* returns `false`. If P_i complains that his part of proof does not verify, the dealer broadcasts $(x_i, \pi_i := (y_i, y_i'))$ so that everyone can verify it using the verification algorithm of the NI-TZK scheme. If it passed the verification, the protocol continues as normal, otherwise the parties disqualify the dealer and *Verification* returns `false`. Since disqualifying the dealer or parties happens on the basis of only broadcasted information, at the end all the honest shareholders will agree on the same set of qualified parties $Q \subseteq \{1, 2, \cdots, n\}$ or will reject the final verification. At the end, if the verification returns `true`, all honest shareholders are sure that they have received a valid share of $x_0 = f(0)$, and any subset of size larger than t of them can retrieve the secret x_0.

Reconstruction: This can be done through two approaches: either by using Lagrange interpolation as in previous works or by employing a novel approach outlined below. In the new approach, the dealer reconstructs (i.e., reveals) the secret x_0 and also proves its validity, and for that the process proceeds as follows:

1. Given the witness $f(X)$, the dealer computes (reconstructs) $x_0 = f(0)$.
2. Using $f(X)$ and $x = (x_0, x_1, \cdots, x_n)$, run the prover of the NI-TZK scheme in Fig. 11 for $i = 0, 1, \ldots, n$, to prove that $f(0) = x_0 \wedge f(i) = x_i$ for $i = 1, \ldots, n$, and obtain $(x_0, y_0, y_0', \{\mathsf{C}_i, \mathsf{C}_i'\}_{i=0}^n, r(X), \{\pi_i\}_{i=1}^n)$. This allows the dealer to convince the shareholders that their shares come from a polynomial of degree t with free term $x_0 = f(0)$.
3. Send the individual proof $\pi_i := (y_i, y_i')$ privately to party P_i, and broadcast the elements $(x_0, y_0, y_0', \{\mathsf{C}_i, \mathsf{C}_i'\}_{i=0}^n, r(X))$.
4. Each shareholder P_i has secret values $(x_i, \pi_i := (y_i, y_i'))$, and a public proof $(x_0, y_0, y_0', \{\mathsf{C}_i, \mathsf{C}_i'\}_{i=0}^n, r(X))$. Given the set of statements and proofs, the shareholders run the verification of the NI-TZK scheme in Fig. 11 and return either `true` or `false`. Note that in this case, each shareholder P_i additionally checks if $\mathsf{C}_0' = \mathcal{C}(x_0, y_0') \wedge \mathsf{C}_0 == \mathcal{C}(r(0) + d \cdot x_0, y_0)$.
5. At the end, the algorithm return `true` if *all* the shareholders return `true`; otherwise return `false`. Returning `true`, confirms that the value x_0 is the reconstruction of the main secret value $f(0)$.

Fig. 13. The NI-VSS scheme of Atapoor, Baghery, Cozzo, and Pedersen [ABCP23c].