Compact Identity-based Signature and Puncturable Signature from SQISign

Surbhi Sahaw and Ratna Dutta[1]
[1] Indian Institute of Technology Kharagpur
surbhi_shaw@iitkgp.ac.in; ratna@maths.iitkgp.ac.in

Abstract. Puncturable signature (PS) offers a fine-grained revocation of signing ability by updating its signing key for a given message m such that the resulting punctured signing key can produce signatures for all messages except for m. In light of the applications of PS in proof-of-stake blockchain protocols, disappearing signatures and asynchronous transaction data signing services, this paper addresses the need for designing practical and efficient PS schemes. Existing proposals pertaining to PS suffer from various limitations, including computational inefficiency, false-positive errors, vulnerability to quantum attacks and large key and signature sizes. To overcome these challenges, we aim to design a PS from isogenies. We first propose an *Identity-Based Signature* (IBS) by employing the Short Quaternion and Isogeny Signature (SQISign). We provide a rigorous security analysis of our IBS and prove it is secure against *unforgeability under chosen identity and chosen message attacks*. More interestingly, our IBS achieves the most compact key and signature size compared to existing isogeny-based IBS schemes. Leveraging our proposed IBS, we introduce the *first* Short Quaternion and Isogeny Puncturable Signature (SQIPS) which allows for selective revocation of signatures and is supported by a comprehensive security analysis against *existential forgery under chosen message attacks with adaptive puncturing*. Our PS scheme SQIPS provides resistance from quantum attacks, enjoys small signature size and is free from false-positive errors.

Keywords: Puncturable signature, Isogenies, Identity-based signature, Post-quantum cryptography.

1 Introduction

With the proliferation of digital technology and the widespread adoption of online transactions, ensuring the privacy and security of sensitive data has emerged as a paramount concern. Cryptographic techniques lay the foundation for secure transactions by protecting the integrity and confidentiality of digital communications. Digital signatures are of particular importance among these cryptographic techniques as they enable parties to verify the authenticity and integrity of communications over the Internet. Puncturable signature (PS) is a variant of digital signature proposed by Bellare, Stepanovs and Waters [1] at EUROCRYPT 2016. It offers a fine-grained revocation of signing ability by updating the secret key

with selective messages. In contrast to a conventional digital signature, PS includes an additional algorithm known as Puncture which enables the signer to create punctured secret key with messages chosen by itself. Precisely, with the punctured secret key that has been punctured at a specific message m, the signer can sign on any message except for the punctured message m. The security definition of a PS requires that the adversary cannot forge signatures on punctured messages even though the punctured secret key is compromised.

Applications. Puncturable signatures have been identified as a versatile cryptographic primitive with numerous applications. These include improving the resilience of proof-of-stake blockchain protocols, designing disappearing signatures and securing asynchronous transaction data signing services. We delve deeper into these applications and their significance below:

- Proof of Stake (PoS) and Proof of Work (PoW) are two consensus mechanisms used in blockchain networks to validate transactions. While PoW requires substantial computational power, PoS relies on participants' cryptocurrency stake, resulting in a more energy-efficient approach. However, the majority of existing PoS protocols are prone to long-range attacks [9] [7]. In this attack, the attacker can tweak the historical records of the blockchain which could lead to double-spending of cryptocurrency or the deletion of prior transactions. PS provide a viable solution to construct practical PoS blockchain resilient to long-range attacks by enabling the selective revocation of past signatures. By puncturing prior used signatures associated with a specific stakeholder, the potential for an attacker to leverage accumulated stakes from the past and manipulate the blockchain's history is reduced. This prevents the forging of past signatures and deter long-range attacks.
- Puncturable signatures are essential building blocks for designing disappearing signature [10] in the bounded storage model. A disappearing signature refers to a signature scheme where the signature becomes inaccessible or "disappears" once the streaming of the signature stops. In the context of bounded storage model, a disappearing signature ensures that the signature can only be verified online and cannot be retained by any malicious party.
- Asynchronous transaction data signing services involve the signing and verification of transaction data in a non-interactive manner without necessitating all parties involved to be online simultaneously [15]. In this context, messages may be delayed and participants may not be available simultaneously due to factors like connectivity issues or delivery failures. PS have applications in ensuring the integrity and authenticity of transaction data in asynchronous signing services. By using PS, the transaction session identity can serve as a prefix that is subsequently punctured after the honest user signs the transaction data. This ensures that no other signature can exist for messages with the same prefix, thereby upholding the integrity of the transaction data.

Related Works. Several studies have been carried out pertaining to PS, exploring their potential applications and security properties. The notion of PS was first proposed by Bellare et al. [1] in 2016. However, their proposed scheme was based on indistinguishability obfuscation which resulted in excessive computational

overhead, rendering the scheme impractical. In a subsequent work, Halevi et al. [11] proposed a PS by combining a statistically binding commitment scheme with non-interactive zero-knowledge proofs. Their approach differed from the conventional PS schemes as it involved updating the public key instead of the secret key during each puncture operation which posed significant challenges in practical deployment. In 2020, Li et al. [13] presented a PS using a bloom filter that surpasses prior schemes in terms of signature size and algorithm efficiency. Additionally, the authors explored the application of PS in proof-of-stake blockchain protocols, specifically addressing the issue of long-range attacks caused by secret key leakage [9] [7]. However, their proposed scheme faced a notable challenge in the form of non-negligible false-positive errors, stemming from the probabilistic nature of the bloom filter data structure. Moreover, their proposed scheme was based on the Strong Diffie-Hellman (SDH) assumption in bilinear map setting and is thus susceptible to quantum attacks due to Shor's algorithm [18]. In light of the devastating consequences that quantum computers have had on the security of classical cryptosystems, Jiang et al. [12] proposed a generic construction of PS from identity-based signatures (IBS). Moreover, they presented different instantiations of their generic construction from lattice-based, pairing-based and multivariate-based assumptions. More precisely, their lattice-based instantiation leverages the efficient IBS proposed by Tian and Huang [19] and is based on the Short Integer Solution (SIS) assumption. Their pairing-based instantiation uses the identity-based version of Paterson's signature [20] which is based on the Computational Diffie-Hellman (CDH) assumption. The instantiation over multivariate assumption relies on ID-based Rainbow signature [4].

Contributions. The existing proposals for PS are undesirable for practical applications. Some PS schemes have large key and signature sizes as they rely on heavy cryptographic structures, making them computationally expensive and inefficient. The PS based on bloom filter suffers from non-negligible false-positive errors, providing economical benefits to the attackers in blockchain. Some PS schemes are prone to quantum attacks raising significant security concerns. To address these limitations, it is imperative to develop improved and more practical approaches to PS. In this work, we identify a gap in the existing literature, noting the absence of a construction for PS from isogenies. The emergence of isogeny-based cryptography as a promising candidate for post-quantum cryptosystems, characterized by its compact key sizes compared to other alternatives, has motivated us to focus on the design of an isogeny-based PS scheme. The compactness of isogeny-based cryptography makes it particularly appealing for practical applications, where efficiency and scalability are crucial factors. To show an instantiation of the generic construction of PS proposed by Jiang et al. [12], we seek an IBS scheme from isogenies. One of the main technical challenges encountered during our research is the absence of a suitable IBS based on isogenies to instantiate the generic construction. Though there exist two constructions of IBS from isogenies in the literature, none appears to be a suitable candidate to design PS. Peng et al. [16] proposed the first construction of IBS from isogenies. Unfortunately, their IBS scheme was proven to be flawed by Shaw

and Dutta [17] who provided a viable fix and designed an IBS scheme from ID-based identification scheme. However, we find that the IBS scheme of [17] has a large key and signature size, rendering it unsuitable for blockchain applications. Furthermore, both the prior IBS schemes are based on *Commutative Supersingular Isogeny Diffie-Hellman* (CSIDH) based group action [2] which suffers from a subexponential attack [5] leading to poor concrete efficiency. The somewhat unsatisfactory state-of-art motivates us to first design an IBS from isogenies with compact key and signature size.

The most recent and sophisticated *Short Quaternion and Isogeny Signature* (SQISign) by De Feo et al. [6] is the starting point in designing our IBS. The signature scheme SQISign is derived from a one-round, high soundness, interactive identification protocol. The combined size of the signature and public key of SQISign are an order of magnitude smaller than all other post-quantum signature schemes. We then employ our proposed IBS to design our PS from isogenies.
Thus, our main contributions in this paper are two-fold, as summarized below:

- *Firstly*, we design an IBS scheme from SQISign which we refer to as *Short Quaternion and Isogeny Identity-based Signature* (SQIIBS). We provide a rigorous security reduction showing it is secure against unforgeability under chosen identity and chosen message attacks (UF-IBS-CMA). We compare our scheme with the existing IBS schemes from isogenies and show that our scheme outperforms existing schemes in terms of key size and signature size which thereby reduces the storage and communication cost.
- Secondly, we employ our identity-based signature scheme SQIIBS to construct our PS from isogenies which we refer to it as Short Quaternion and Isogeny Puncturable Signature (SQIPS). We prove our scheme to be secure against *existential unforgeability under chosen message attacks with adaptive puncturing* (UF-CMA-AP). We also compare the features of our scheme with the existing PS schemes. Our scheme works for a pre-determined time of key punctures since the range of prefix space is fixed in advance. The size of the punctured secret key decreases linearly as the times of key puncture increase. Our scheme involves an efficient puncture operation that only contain a conversion from a bit string to a decimal integer and the deletion of a part in the current secret key. More positively, our scheme provides quantum security, enjoys small signature size and is free from false-positive errors.

2 Preliminaries

Let $\lambda \in \mathbb{N}$ denotes the security parameter. By $i \in [T]$, we mean i belongs to the set $\{1, 2, \ldots, T\}$. The symbol $\#S$ denotes the cardinality of S. By $\mathsf{bin}(x)$, we mean the binary representation of x. A function $\epsilon(\cdot)$ is negligible if for every positive integer c, there exists an integer k such that for all $\lambda > k$, $|\epsilon(\lambda)| < 1/\lambda^c$.

2.1 Quaternion Algebras, Orders and Ideals

Quaternion Algebras. For $a, b \in \mathbb{Q}^* = \mathbb{Q} \setminus \{0\}$, the quaternion algebra over $\mathbb{Q}$, denoted by $H(a,b) = \mathbb{Q} + i\mathbb{Q} + j\mathbb{Q} + k\mathbb{Q}$, is defined as a four-dimensional non-commutative vector space with basis $\{1, i, j, k\}$ such that $i^2 = a$, $j^2 = b$ and $k = ij = -ji$. Every quaternion algebra $H(a,b)$ is associated by a standard convolution $g : H(a,b) \to H(a,b)$ given by $g : \alpha = a_1 + a_2 i + a_3 j + a_4 k \to a_1 - a_2 i - a_3 j - a_4 k = \bar{\alpha}$. The *reduced norm* $\mathsf{nr} : H(a,b) \to \mathbb{Q}$ of a standard convolution g is the map $\mathsf{nr} : \alpha \to \alpha g(\alpha)$. In this work, we are interested in the quaternion algebra $\mathcal{B}_{p,\infty} = H(-1, -p)$ for some prime p.

Ideals and Orders. A *fractional ideal* $I = \alpha_1\mathbb{Z} + \alpha_2\mathbb{Z} + \alpha_3\mathbb{Z} + \alpha_4\mathbb{Z}$ is a $\mathbb{Z}$-lattice of rank four with $\{\alpha_1, \alpha_2, \alpha_3, \alpha_4\}$ a basis of $\mathcal{B}_{p,\infty}$. The *norm* of I, denoted by $\mathsf{nr}(I)$, is defined as the largest rational number such that $\mathsf{nr}(\alpha) \in \mathsf{nr}(I)\mathbb{Z}$ for any $\alpha \in I$. The *conjugate ideal* $\bar{I}$ of I is given by $\bar{I} = \{\bar{\alpha} \,|\, \alpha \in I\}$. An *order* is a subring of $\mathcal{B}_{p,\infty}$ that is also a fractional ideal. A *maximal order* $\mathcal{O}$ is an order that is not properly contained in any other order. The *left order* of a fractional ideal I, denoted by $\mathcal{O}_L(I)$, is defined as $\mathcal{O}_L(I) = \{\alpha \in \mathcal{B}_{p,\infty} \,|\, \alpha I \subseteq I\}$. Similarly, *right order* of a fractional ideal I, denoted by $\mathcal{O}_R(I)$, is defined as $\mathcal{O}_R(I) = \{\alpha \in \mathcal{B}_{p,\infty} \,|\, I\alpha \subseteq I\}$. Here I is said to be a left $\mathcal{O}_L(I)$-ideal or a right $\mathcal{O}_R(I)$-ideal or an $(\mathcal{O}_L(I), \mathcal{O}_R(I))$-ideal. An *Eichler order* is the intersection of two maximal orders inside $\mathcal{B}_{p,\infty}$. A fractional ideal I is called *integral* if $I \subseteq \mathcal{O}_L(I)$ or $I \subseteq \mathcal{O}_R(I)$. Two left $\mathcal{O}$-ideals I and J are *equivalent* if there exists $\beta \in \mathcal{B}_{p,\infty} \setminus \{0\}$ such that $I = J\beta$ and is denoted by $I \sim J$. A *special extremal order* is an order $\mathcal{O}$ in $\mathcal{B}_{p,\infty}$ which contains a suborder of the form $R + jR$ where $R = \mathbb{Z}[\omega] \subset \mathbb{Q}[i]$ is a quadratic order and ω has smallest norm in $\mathcal{O}$.

2.2 Elliptic curves, isogenies and Deuring's correspondence

Isogenies. Let E_1 and E_2 be two elliptic curves over a finite field F. An *isogeny* from E_1 to E_2 is a non-constant morphism $\varphi : E_1 \to E_2$ over F satisfying $\varphi(\Theta_{E_1}) = \Theta_{E_2}$ where Θ_{E_i} is the point at infinity of the curve E_i for $i = 1, 2$. The *degree* of the isogeny φ, denoted by $\mathsf{deg}(\varphi)$ is its degree as a rational map. A non-zero isogeny $\varphi : E_1 \to E_2$ is called *separable* if and only if $\mathsf{deg}(\varphi) = \#\mathsf{ker}(\varphi)$ where $\mathsf{ker}(\varphi) = \varphi^{-1}(\Theta_{E_2})$ is the kernel of φ. An isogeny φ is said to be *cyclic* (non-backtracking) if its kernel is a cyclic group. For any isogeny $\varphi : E_1 \to E_2$, there exists a unique *dual isogeny* $\hat{\varphi} : E_2 \to E_1$ satisfying $\varphi \circ \hat{\varphi} = [\mathsf{deg}(\varphi)]$, the multiplication-by- $\mathsf{deg}(\varphi)$ map on E_2. An isogeny from an elliptic curve E to itself is called an *endomorphism*. The set of all endomorphisms of E forms a ring under pointwise addition and composition, called the *endomorphism ring* of E and is denoted by $\mathsf{End}(E)$. For a supersingular elliptic curve E, the endomorphism ring $\mathsf{End}(E)$ is isomorphic to an order in a quaternion algebra. The *j-invariant* of an elliptic curve $E : y^2 = x^3 + Ax + B$ over F is given by $j(E) = 1728\frac{4A^3}{4A^3+27B^2}$.

Theorem 2.21. [2] *Given a finite subgroup G of an elliptic curve E_1, there exists a unique (up to F-isomorphism) elliptic curve E_2 and a separable isogeny $\varphi : E_1 \to E_2$ such that $\mathsf{ker}(\varphi) = G$ and $E_2 \coloneqq E_1/G$ with $\mathsf{deg}(\varphi) = \#\mathsf{ker}(\varphi)$.*

Throughout this work, we focus on supersingular curves over $F = \mathbb{F}_{p^2}$. We fix the curve $E_0 : y^2 = x^3 + x$ over $\mathbb{F}_{p^2}$ which has special extremal endomorphism ring $\mathsf{End}(E_0) = \mathcal{O}_0 = \langle 1, i, \frac{i+j}{2}, \frac{1+k}{2} \rangle$ where $i^2 = -1$, $j^2 = -p$ and $k = ij$.

Deuring's correspondence: *Deuring's correspondence* [8] establishes a one-to-one correspondence between the set of isomorphism classes of supersingular curves over $\mathbb{F}_{p^2}$ and the set of ideal classes of a given maximal order. Under this correspondence, we look into the connection between ideals in maximal orders of quaternions and separable isogenies between supersingular curves over $\mathbb{F}_{p^2}$.

Theorem 2.22. *Let $\varphi : E_0 \to E_1$ be a separable isogeny and $\mathcal{O}_0 = \mathsf{End}(E_0)$ and $\mathcal{O}_1 = \mathsf{End}(E_1)$ are the maximal orders corresponding to the endomorphism rings of E_0 and E_1. Then we define the corresponding left $\mathcal{O}_0$-ideal $I_\varphi = \{\alpha \in \mathcal{O}_0 \,|\, \alpha(P) = \Theta_{E_0}$ for all $P \in \mathsf{ker}(\varphi)\}$. Conversely, given a left $\mathcal{O}_0$-ideal I, we can define the kernel $E_0[I] = \cap_{\alpha \in I} E_0[\alpha] = \{P \in E_0 \,|\, \alpha(P) = \Theta_{E_0}$ for all $\alpha \in I\}$ and compute the separable isogeny $\varphi_I : E_0 \to E_0/E_0[I]$ that corresponds to I.*

Lemma 2.23. [6]. *Let $\mathcal{O}$ be a maximal order, I be a left $\mathcal{O}$-ideal and $\beta \in I \setminus \{0\}$. Then $\chi_I(\beta) = I\frac{\bar{\beta}}{\mathsf{nr}(I)}$ is a left $\mathcal{O}$-ideal equivalent to I and has norm $\frac{\mathsf{nr}(\beta)}{\mathsf{nr}(I)}$.*

Pushforward and pullback isogeny. Consider three elliptic curves E_0, E_1, E_2 over $\mathbb{F}_{p^2}$ and two separable isogenies $\varphi_1 : E_0 \to E_1$ and $\varphi_2 : E_0 \to E_2$ of coprime degrees N_1 and N_2 respectively. The *pushforward of φ_1 by φ_2* is denoted by $[\varphi_2]_*\varphi_1$ and is defined as the separable isogeny $[\varphi_2]_*\varphi_1$ from E_2 to some new curve E_3 such that $\mathsf{ker}([\varphi_2]_*\varphi_1) = \varphi_2(\mathsf{ker}(\varphi_1))$ and $\mathsf{deg}([\varphi_2]_*\varphi_1) = N_1$. Similarly, the *pushforward of φ_2 by φ_1* is denoted by $[\varphi_1]_*\varphi_2$ and is defined as the separable isogeny $[\varphi_1]_*\varphi_2 : E_1 \to E_3$ such that $\mathsf{ker}([\varphi_1]_*\varphi_2) = \varphi_1(\mathsf{ker}(\varphi_2))$ and $\mathsf{deg}([\varphi_1]_*\varphi_2) = N_2$. *Pullback isogeny* is the dual notion of pushforward isogeny. Consider two separable isogeniers $\varphi_1 : E_0 \to E_1$ and $\rho_2 : E_1 \to E_3$ of coprime degrees. The pullback of ρ_2 by φ_1 is denoted by $[\varphi_1]^*\rho_2$ and is defined as the separable isogeny $[\varphi_1]^*\rho_2$ from E_0 to a new curve E_4 satisfying $[\varphi_1]^*\rho_2 = [\hat{\varphi}_1]_*\rho_2$.

The pushforward and pullback terms can be extended to ideals as well. Consider a $(\mathcal{O}_0, \mathcal{O}_1)$-ideal J and a $(\mathcal{O}_0, \mathcal{O}_2)$-ideal K where $\mathcal{O}_0 = \mathsf{End}(E_0)$, $\mathcal{O}_1 = \mathsf{End}(E_1)$ and $\mathcal{O}_2 = \mathsf{End}(E_2)$. The *pushforward of J by K*, denoted by $[K]_*J$ is the ideal $I_{[\varphi_K]_*\varphi_J}$ corresponding to the pushforward isogeny $[\varphi_K]_*\varphi_J$. Consider a $(\mathcal{O}_1, \mathcal{O}_3)$-ideal L where $\mathcal{O}_1 = \mathsf{End}(E_1)$, $\mathcal{O}_3 = \mathsf{End}(E_3)$, then the *pullback of L by J*, denoted by $[J]^*L$ is defined as $[J]^*L = [\bar{J}]_*L$.

Lemma 2.24. [6] *Let I is an ideal with left order $\mathcal{O}_0$ and right order $\mathcal{O}$ and J_1, J_2 be $\mathcal{O}_0$-ideals with $J_1 \sim J_2$ and gcd($\mathsf{nr}(J_1)$, $\mathsf{nr}(J_2)$, $\mathsf{nr}(I)$) = 1. Suppose that $J_1 = \chi_{J_2}(\beta)$ and $\beta \in J_2 \cap \mathcal{O}_0 \cap \mathcal{O}$. Then $[I]_*J_1 \sim [I]_*J_2$ and $[I]_*J_1 = \chi_{[I]_*J_2}(\beta)$.*

2.3 SigningKLPT *algorithm*

We briefly review below the sub-algorithms invoked by the algorithm SigningKLPT. The details of which can be found in the work of De Feo et al. [6].

$\mathsf{Cornacchia}(M) \to (x, y)$: This algorithm on input $M \in \mathbb{Z}$ either outputs $\perp$ if M cannot be represented as $f(x, y)$ or returns a solution (x, y) to $f(x, y) = M$.

$\mathsf{EquivalentPrimeIdeal}(I) \to L \sim I$: This algorithm takes as input a left $\mathcal{O}_0$-ideal I represented by Minkowski reduced basis [14] $(\delta_1, \delta_2, \delta_3, \delta_4)$. It chooses an integer m, generates a random element $\delta = \Sigma_i x_i \delta_i$ with $x_i \in [-m, m]$ and checks if $\frac{\mathsf{nr}(\delta)}{\mathsf{nr}(I)}$ is a prime number. If not, it continues to generate random δ until it finds a $\delta \in I$ for which $\frac{\mathsf{nr}(\delta)}{\mathsf{nr}(I)}$ is a prime number. The algorithm outputs the ideal $L = \chi_I(\delta) = I\frac{\bar{\delta}}{\mathsf{nr}(I)}$ equivalent to I and of prime norm.

$\mathsf{EquivalentRandomEichlerIdeal}(I,\ N) \to L \sim I$: This algorithm takes as input a left $\mathcal{O}_0$-ideal I and an integer N and finds a random equivalent left $\mathcal{O}_0$-ideal L of norm coprime to N.

$\mathsf{FullRepresentInteger}_{\mathcal{O}_0}(M) \to \gamma$: This algorithm takes input an integer $M \in \mathbb{Z}$ with $M > p$ and outputs an element $\gamma \in \mathcal{O}_0$ with $\mathsf{nr}(\gamma) = M$ as follows.

i. Sets $m' = \lfloor\sqrt{\frac{4M}{p}}\rfloor$ and samples a random integer $z' \in [-m', m']$.
ii. Sets $m'' = \lfloor\sqrt{\frac{4M}{p} - (z')^2}\rfloor$ and samples a random integer $t' \in [-m'', m'']$.
iii. Sets $M' = 4M - p\big((z')^2 + (t')^2\big)$ and runs $\mathsf{Cornacchia}(M')$ until $\mathsf{Cornacchia}$ returns a solution (x', y') to $f(x', y') = M'$.
iv. If $x' \neq t' \pmod 2$ or $z' \neq y' \pmod 2$ then go back to Step (i).
v. The algorithm outputs $\gamma = x + yi + z\frac{i+j}{2} + t\frac{1+k}{2} \in \mathcal{O}_0$ of norm M where $x = \frac{x'-t}{2}, y = \frac{y'-z}{2}, z = z'$ and $t = t'$.

$\mathsf{IdealModConstraint}(I, \gamma) \to (C_0 : D_0)$: On input a left $\mathcal{O}_0$-ideal I of norm N and an element $\gamma \in \mathcal{O}_0$ of norm Nn, this algorithm outputs a projective point $(C_0 : D_0) \in \mathbb{P}^1(\mathbb{Z}/N\mathbb{Z})$ satisfying $\gamma\mu_0 \in I$ with $\mu_0 = (C_0 + \omega D_0)j \in Rj$.

$\mathsf{EichlerModConstraint}(I,\ \gamma,\ \delta) \to (C_0 : D_0)$: This algorithm takes input a left $\mathcal{O}_0$-ideal I of norm N, elements $\gamma,\ \delta \in \mathcal{O}_0$ of norms coprime to N and outputs a projective point $(C_0 : D_0) \in \mathbb{P}^1(\mathbb{Z}/N\mathbb{Z})$ satisfying $\gamma\mu_0\delta \in I$ where $\mu_0 = (C_0 + \omega D_0)j \in Rj$.

$\mathsf{FullStrongApproximation}_{\mathcal{S}}(N, C, D) \to \mu$: Taking as input a prime N, integers $C,\ D$ and a subset $\mathcal{S} \subset \mathbb{N}$, this algorithm outputs $\mu \in \mathcal{O}_0$ of norm in $\mathcal{S}$ satisfying $2\mu = \lambda\mu_0 + N\mu_1$ where $\mu_0 = (C_0 + \omega D_0)j \in Rj$, $\lambda \in \mathbb{Z}$ and $\mu_1 \in \mathcal{O}_0$. When $\mathcal{S} = \{d \in \mathbb{N} : d|D\}$ for some $D \in \mathbb{N}$, we simply write $\mathsf{FullStrongApproximation}_D$.

$\mathsf{CRT}_{M,N}(x, y) \to z$: This is the algorithm for Chinese Remainder Theorem which takes as input $x \in \mathbb{Z}_M$, $y \in \mathbb{Z}_N$ and returns $z \in \mathbb{Z}_{MN}$ satisfying $z \equiv x \pmod M$ and $z \equiv y \pmod N$ where M and N are coprime to each other.

We now describe the algorithm $\mathsf{SigningKLPT}_{\ell^e}(I_\tau, I)$ [6] which takes as input a prime l, a fixed $e \in \mathbb{N}$, a left $\mathcal{O}_0$ and a right $\mathcal{O}$-ideal I_τ of norm N_τ and a left $\mathcal{O}$-ideal I and outputs an ideal $J \sim I$ of norm ℓ^e. The steps involved in the algorithm $\mathsf{SigningKLPT}$ are illustrated in Fig. 1 and explicitly described below.

1. Runs the algorithm $\mathsf{EquivalentRandomEichlerIdeal}(I, N_\tau)$ to generate a random ideal $K \sim I$ with $gcd(\mathsf{nr}(K), N_\tau) = 1$. We denote the right order of the ideal K (or I) by $\mathcal{O}_2$.

2. Performs the pullback of the $(\mathcal{O}, \mathcal{O}_2)$- ideal K by the $(\mathcal{O}_0, \mathcal{O})$-ideal I_τ to obtain a $(\mathcal{O}_0, \mathcal{O}')$-ideal $K' = [I_\tau]^* K$ where $\mathcal{O}' = \mathsf{End}(E')$ for some curve E'.
3. Computes an ideal $L = K' \frac{\bar{\delta'}}{\mathsf{nr}(K')} = \chi_{K'}(\delta') \leftarrow \mathsf{EquivalentPrimeIdeal}(K')$ equivalent to K' but of prime norm N for some $\delta' \in K'$. (See Lemma 2.23)
4. Chooses $e_0 \in \mathbb{N}$ and runs the algorithm $\mathsf{FullRepresentInteger}_{\mathcal{O}_0}(N\ell^{e_0})$ to obtain an element $\gamma \in \mathcal{O}_0$ such that $\mathsf{nr}(\gamma) = N\ell^{e_0}$. Sets $e_1 = e - e_0 \in \mathbb{N}$.
5. Finds the projective point $(C_0 : D_0) \in \mathbb{P}^1(\mathbb{Z}/N\mathbb{Z}) \leftarrow \mathsf{IdealModConstraint}(L, \gamma)$ satisfying $\gamma\mu_0 \in L$ where $\mu_0 = (C_0 + \omega D_0)j \in Rj$.
6. Chooses $\delta \in \mathcal{O}_0$ with $gcd(\mathsf{nr}(\delta), N_\tau) = 1$ and runs the algorithm $\mathsf{EichlerModConstraint}(\mathbb{Z} + I_\tau, \gamma, \delta)$ on input the ideal $\mathbb{Z} + I_\tau$ of norm N_τ and elements $\gamma, \delta \in \mathcal{O}_0$ of norms coprime to N_τ to find the projective point $(C_1 : D_1) \in \mathbb{P}^1(\mathbb{Z}/N_\tau\mathbb{Z})$ satisfying $\gamma\mu_1\delta \in \mathbb{Z} + I_\tau$ where $\mu_1 = (C_1 + \omega D_1)j \in Rj$.
7. Computes $C \leftarrow \mathsf{CRT}_{N,N_\tau}(C_0, C_1)$ where C is the solution modulo NN_τ to the system of congruences $C \equiv C_0 \pmod N$ and $C \equiv C_1 \pmod{N_\tau}$ and $D \leftarrow \mathsf{CRT}_{N,N_\tau}(D_0, D_1)$ where D is the solution modulo NN_τ to the system of congruences $D \equiv D_0 \pmod N$ and $D \equiv D_1 \pmod{N_\tau}$. If $\ell^e p(C^2 + D^2)$ is not a quadratic residue, go back to Step 4 and repeat the process.
8. Executes the algorithm $\mathsf{FullStrongApproximation}_{\ell^\star}(NN_\tau, C, D)$ to generate $\mu \in \mathcal{O}_0$ of norm ℓ^{e_1} where $\ell^\star = \{\ell^\alpha : \alpha \in \mathbb{N}\}$.
9. Sets $\beta = \gamma\mu$, obtains the $(\mathcal{O}_0, \mathcal{O}')$-ideal $\chi_L(\beta) = L\frac{\bar{\beta}}{\mathsf{nr}(L)}$ (See Lemma 2.23) and computes the $(\mathcal{O}, \mathcal{O}_2)$- ideal $J = [I_\tau]_* \chi_L(\beta)$ by using pushforward of the ideal $\chi_L(\beta)$ by the $(\mathcal{O}_0, \mathcal{O})$-ideal I_τ. (See Lemma 2.24)
10. The algorithm then returns the ideal $J \sim I$.

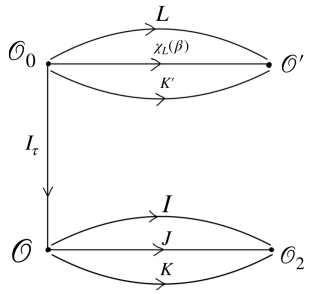

Fig. 1. Pictorial description of $\mathsf{SigningKLPT}$ algorithm

Correctness. Step 5 and Step 8 ensure $\beta \in L$ whereas Step 6 ensures $\beta \in \mathbb{Z} + I_\tau$. Also we have, $\mathsf{nr}(\beta) = \mathsf{nr}(\gamma)\mathsf{nr}(\mu) = N\ell^{e_0}\dot{\ell}^{e_1} = N\ell^e$ which implies $\mathsf{nr}(J) = \mathsf{nr}([I_\tau]_*\chi_L(\beta)) = \frac{\mathsf{nr}(\beta)}{\mathsf{nr}(L)} = \frac{N\ell^e}{N} = \ell^e$. Also, Lemma 2.24 applied to $\chi_L(\beta) = L\frac{\bar{\beta}}{\mathsf{nr}(L)} = \chi_{K'}(\delta')\frac{\bar{\beta}}{\mathsf{nr}(L)} = K'\frac{\bar{\delta'}}{\mathsf{nr}(K')}\frac{\bar{\beta}}{\mathsf{nr}(L)} = \chi_{K'}(\frac{\overline{\beta\delta'}}{\mathsf{nr}(L)})$ implies that $[I_\tau]_*\chi_L(\beta) \sim [I_\tau]_*K'$. This proves $J \sim K$ and we also have $K \sim I$, which implies $J \sim I$.

2.4 Signature Scheme

Definition 2.41. A *signature scheme* associated with a message space $\mathcal{M}$ is a tuple of probabilistic polynomial-time (PPT) algorithms $\mathsf{Sig} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ with the following syntax:

- $\mathsf{Sig.Setup}(1^\lambda) \to \mathsf{pp}$: A trusted party taking input 1^λ outputs the public parameter pp and makes it publicly available.
- $\mathsf{Sig.KeyGen}(\mathsf{pp}) \to (\mathsf{sk}, \mathsf{pk})$: On input pp, the user runs this algorithm to generate a signing and verification key pair $(\mathsf{sk}, \mathsf{pk})$.
- $\mathsf{Sig.Sign}(\mathsf{pp}, \mathsf{sk}, m) \to \sigma$: Taking input pp, sk and a message $m \in \mathcal{M}$, the signer executes this algorithm to generate a signature σ on the message m.
- $\mathsf{Sig.Verify}(\mathsf{pp}, \mathsf{pk}, m, \sigma) \to \mathsf{Valid/Invalid}$: On input pp, pk, $m \in \mathcal{M}$ and a signature σ, the verifier checks the validity of the signature σ on m.

Correctness. For all $\mathsf{pp} \leftarrow \mathsf{Sig.Setup}(1^\lambda)$, all $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Sig.KeyGen}(\mathsf{pp})$ and all signature $\sigma \leftarrow \mathsf{Sig.Sign}(\mathsf{pp}, \mathsf{sk}, m)$, it holds that

$$\mathsf{Sig.Verify}(\mathsf{pp}, \mathsf{pk}, m, \sigma) = \mathsf{Valid}$$

Definition 2.42. A signature scheme Sig is secure against *existential unforgeability under chosen-message attacks* ($\mathsf{UF\text{-}CMA}$) if for all PPT adversary $\mathcal{A}$, there exists a negligible function ϵ such that

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

where the experiment $\mathsf{Exp}^{\mathsf{UF\text{-}CMA}}_{\mathsf{Sig},\mathcal{A}}(\lambda)$ is depicted in Fig.2.

Setup: The challenger $\mathcal{C}$ generates the public parameter $\mathsf{pp} \leftarrow \mathsf{Sig.Setup}(1^\lambda)$ and secret-public key pair $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Sig.KeyGen}(\mathsf{pp})$. It forwards pp and pk to the adversary $\mathcal{A}$ while keeps sk secret to itself. It also maintains a list SList and initializes SList to $\emptyset$.

Query Phase: $\mathcal{A}$ issues polynomially many adaptive signature queries to the following oracle:

- $\mathcal{O}_{\mathsf{S}}(\mathsf{sk}, \cdot)$: On receiving a signature query on a message m, the challenger $\mathcal{C}$ checks if $m \notin \mathcal{M}$. If the check succeeds, it returns $\perp$. Otherwise, it computes a signature $\sigma \leftarrow \mathsf{Sig.Sign}(\mathsf{pp}, \mathsf{sk}, m)$ on the message m under the secret key sk and updates $\mathsf{SList} \leftarrow \mathsf{SList} \cup \{m\}$. It returns the computed signature σ to the adversary $\mathcal{A}$.

Forgery: The adversary $\mathcal{A}$ eventually submits a forgery (m^*, σ^*). The adversary $\mathcal{A}$ wins the game if $m^* \notin \mathsf{SList}$ and $\mathsf{Valid} \leftarrow \mathsf{Sig.Verify}(\mathsf{pp}, \mathsf{pk}, m^*, \sigma^*)$.

Fig. 2. $\mathsf{Exp}^{\mathsf{UF\text{-}CMA}}_{\mathsf{Sig},\mathcal{A}}(\lambda)$: Existential unforgeability under chosen-message attack

2.5 SQISign: an isogeny-based signature scheme

The signature scheme $\mathsf{SQISign}$ [6] comprises of four PPT algorithms (Setup, KeyGen, Sign, Verify) having the following interface:

$\mathsf{SQISign.Setup}(1^\lambda) \to \mathsf{pp}_{\mathrm{sgn}}$: A trusted authority runs this algorithm on input a security parameter 1^λ and performs the following steps:

i. Chooses a prime p and fixes the supersingular curve $E_0 : y^2 = x^3 + x$ over $\mathbb{F}_{p^2}$ with special extremal endomorphism ring $\mathsf{End}(E_0) = \mathcal{O}_0 = \langle 1, i, \frac{i+j}{2}, \frac{1+k}{2} \rangle$.
ii. Picks a smooth number $D = 2^e$ where $2^e > p^3$.
iii. Picks an odd smooth number $D_c = \ell^e$ where ℓ is a prime and $e \in \mathbb{N}$ and computes $\mu(D_c) = (\ell + 1) \cdot \ell^{e-1}$.

iv. Samples a cryptographic hash function $\mathcal{H}_1 : \mathbb{F}_{p^2} \times \{0,1\}^* \to [\mu(D_c)]$.
v. Samples an arbitrary function $\Phi_{D_c}(E, s)$ that maps a curve E and an integer $s \in [\mu(D_c)]$ to a non-backtracking isogeny of degree D_c from E [3].
vi. Sets the public parameter $\mathsf{pp}_{\text{sgn}} = (p, E_0, D_c, D, \mathcal{H}_1, \Phi_{D_c})$.

$\mathsf{SQISign.KeyGen}(\mathsf{pp}_{\text{sgn}}) \to (\mathsf{sk}, \mathsf{pk})$: On input pp_{sgn}, the key generation algorithm run by a user generates a signing-verification key pair $(\mathsf{sk}, \mathsf{pk})$ as follows:

i. Picks a random isogeny $\tau : E_0 \to E_A$ of degree N_τ.
ii. Sets the signing key $\mathsf{sk} = \tau$ and verification key $\mathsf{pk} = E_A$.

$\mathsf{SQISign.Sign}(\mathsf{pp}_{\text{sgn}}, \mathsf{sk}, m) \to \sigma$: Taking input pp_{sgn}, signing key $\mathsf{sk} = \tau$ and a message $m \in \{0,1\}^*$, the signer generates a signature σ on m as follows:

i. Picks a random commitment isogeny $\psi : E_0 \to E_1$.
ii. Computes $s = \mathcal{H}_1(j(E_1), m)$ and sets the challenge isogeny $\Phi_{D_c}(E_1, s) = \varphi$ where $\varphi : E_1 \to E_2$ is a non-backtracking isogeny of degree D_c.
iii. Computes $\bar{I}_\tau$, I_τ, I_ψ and I_φ corresponding to $\hat{\tau}$, τ, ψ and φ respectively.
iv. The signer having the knowledge of $\mathcal{O} = \mathsf{End}(E_A)$ through $\mathsf{sk} = \tau$ and $\mathcal{O}_2 = \mathsf{End}(E_2)$ through $\varphi \circ \psi : E_0 \to E_2$, executes the algorithm $\mathsf{SigningKLPT}_{2^e}(I_\tau, I)$ described in Section 2.3 on input the $(\mathcal{O}_0, \mathcal{O})$-ideal I_τ and the left $\mathcal{O}$-ideal $I = I_\varphi I_\psi \bar{I}_\tau$ to obtain a $(\mathcal{O}, \mathcal{O}_2)$-ideal $J \sim I$ of norm $D = 2^e$.
v. Constructs a cyclic isogeny $\eta : E_A \to E_2$ of degree D corresponding to the ideal J such that $\hat{\varphi} \circ \eta$ is cyclic. The signature is the pair $\sigma = (E_1, \eta)$.

$\mathsf{SQISign.Verify}(\mathsf{pp}_{\text{sgn}}, \mathsf{pk}, m, \sigma) \to \mathsf{Valid}/\mathsf{Invalid}$: The verifier verifies the validity of the signature $\sigma = (E_1, \eta)$ on the message m as follows:

i. Computes $s = \mathcal{H}_1(j(E_1), m)$ and then recovers the isogeny $\Phi_{D_c}(E_1, s) = \varphi$.
ii. Checks if η is an isogeny of degree D from E_A to E_2 and that $\hat{\varphi} \circ \eta : E_A \to E_1$ is cyclic.
iii. If all the checks succeed returns Valid, otherwise returns $\mathsf{Invalid}$.

Correctness. It follows from the correctness of $\mathsf{SigningKLPT}$ algorithm.

3 Security Aspect of SQISign

To prove the security of the signature scheme $\mathsf{SQISign}$, the authors resort to a computational assumption that formalises the idea that the isogeny η corresponding to the ideal J returned by the algorithm $\mathsf{SigningKLPT}$ is indistinguishable from a random isogeny of the same degree. Before defining the problem formally, we analyze the structure of η.

Lemma 3.01. [6] *Consider the ideal L and element $\beta \in L$ computed as in steps 3, 9 respectively of the algorithm* $\mathsf{SigningKLPT}$ *described in Section 2.3. The isogeny η corresponding to the output J of* $\mathsf{SigningKLPT}$ *algorithm is equal to $\eta = [\tau]_* \iota$ where ι is an isogeny of degree ℓ^e satisfying $\beta = \hat{\iota} \circ \varphi_L$.*

We recall the following notations before defining the (computationally) indistinguishable problem underlying the security of $\mathsf{SQISign}$.

$\mathcal{U}_{L,N_\tau}$: For a given ideal L of norm N, $\mathcal{U}_{L,N_\tau}$ denotes the set of all isogenies ι computed in Lemma 3.01 from elements $\beta = \gamma\mu \in L$ where γ is any

possible output of the algorithm $\mathsf{FullRepresentInteger}_{\mathcal{O}_0}$ and μ is computed by algorithm $\mathsf{FullStrongApproximation}$ in Step 8 of $\mathsf{SigningKLPT}$.

$\mathcal{P}_{N_\tau}$: We define $\mathcal{P}_{N_\tau} = \bigcup_{\mathcal{C}\in\mathsf{Cl}(\mathcal{O})} \mathcal{U}_{\mathcal{C},N_\tau}$ where we write $\mathcal{U}_{\mathcal{C},N_\tau}$ for $\mathcal{U}_{L,N_\tau}$ where $L \leftarrow \mathsf{EquivalentPrimeIdeal}(\mathcal{C})$ for an equivalence class $\mathcal{C}$ in the ideal class group $\mathsf{Cl}(\mathcal{O}_0)$ of $\mathcal{O}_0$.

$\mathsf{Iso}_{D,j(E)}$: Denotes the set of cyclic isogenies of degree D whose domain is a curve inside the isomorphism class of E.

$[\tau]_*\mathcal{P}$: Denotes the subset $\{[\tau]_*\varphi \mid \varphi \in \mathcal{P}\}$ of $\mathsf{Iso}_{D,j(E_0)}$ where $\mathcal{P}$ is a subset of $\mathsf{Iso}_{D,j(E)}$ and $\tau : E \to E_0$ is an isogeny with $\gcd(\mathsf{deg}(\tau), D) = 1$.

$\mathcal{K}$: a probability distribution on the set of cyclic isogenies whose domain is E_0, representing the distribution of $\mathsf{SQISign}$ private keys.

Definition 3.02. [6] Let p be a prime and D be a smooth integer. Let $\tau : E_0 \to E_A$ be a random isogeny drawn from $\mathcal{K}$ and let N_τ be its degree. Let Oracle_τ be an oracle sampling random elements in $[\tau]_*\mathcal{P}_{N_\tau}$. Let η be an isogeny of degree D whose domain curve is E. Given $p, D, \mathcal{K}, E_A, \eta$ and a polynomial number of queries to Oracle_τ, the *Real or Random Isogeny* problem is to determine where

1. whether η is uniformly random in $\mathsf{Iso}_{D,j(E_A)}$
2. or η is uniformly random in $[\tau]_*\mathcal{P}_{N_\tau}$.

Informally speaking, the problem states that the ideals output by the algorithm $\mathsf{SigningKLPT}$ are indistinguishable from uniformly random ideals of the same norm. The hardness assumption underlying the security of $\mathsf{SQISign}$ is the Real or Random Isogeny problem defined in Definition 3.02.

Theorem 3.03. [6] *The scheme* $\mathsf{SQISign}$ *is* $\mathsf{UF\text{-}CMA}$ *secure under the hardness of Real or Random Isogeny Problem defined in Definition 3.02.*

3.1 Identity-based signature

Definition 3.11. An *identity-based signature* is a tuple $\mathsf{IBS} = (\mathsf{Setup}, \mathsf{Extract}, \mathsf{Sign}, \mathsf{Verify})$ of four PPT algorithms with the following syntax:

$\mathsf{IBS.Setup}(1^\lambda) \to (\mathsf{pp}_{\mathrm{ibs}}, \mathsf{msk})$: The key generation centre (KGC) on input 1^λ generates a public parameter $\mathsf{pp}_{\mathrm{ibs}}$ and a master secret key msk.

$\mathsf{IBS.Extract}(\mathsf{pp}_{\mathrm{ibs}}, \mathsf{msk}, \mathsf{id}) \to \mathsf{usk}_{\mathsf{id}}$: The KGC runs this key extract algorithm on input the public parameter $\mathsf{pp}_{\mathrm{ibs}}$, the master secret key msk and user identity id. It generates the user secret key $\mathsf{usk}_{\mathsf{id}}$ for the given identity id.

$\mathsf{IBS.Sign}(\mathsf{pp}_{\mathrm{ibs}}, \mathsf{usk}_{\mathsf{id}}, m) \to \sigma$: Taking input the public parameter $\mathsf{pp}_{\mathrm{ibs}}$, user secret key $\mathsf{usk}_{\mathsf{id}}$ and a message m, the signer executes this randomized algorithm and outputs a signature σ on the message m.

$\mathsf{IBS.Verify}(\mathsf{pp}_{\mathrm{ibs}}, \mathsf{id}, m, \sigma) \to \mathsf{Valid}/\mathsf{Invalid}$: The verifier runs this deterministic algorithm on input the public parameter $\mathsf{pp}_{\mathrm{ibs}}$, an identity id, a message m and a signature σ to verify the validity of the signature σ.

Correctness. For all $(\mathsf{pp}_{\mathrm{ibs}}, \mathsf{msk}) \leftarrow \mathsf{IBS.Setup}(1^\lambda)$, all $\mathsf{usk}_{\mathsf{id}} \leftarrow \mathsf{IBS.Extract}(\mathsf{pp}_{\mathrm{ibs}}, \mathsf{msk}, \mathsf{id})$, all m and all id, it holds that

$$\mathsf{IBS.Verify}(\mathsf{pp}_{\mathrm{ibs}}, \mathsf{id}, m, \mathsf{IBS.Sign}(\mathsf{pp}_{\mathrm{ibs}}, \mathsf{usk}_{\mathsf{id}}, m)) \to \mathsf{Valid}.$$

Definition 3.12. An IBS scheme is said to be secure against *unforgeability under chosen identity and chosen message attacks* (UF-IBS-CMA) if for all PPT adversary $\mathcal{A}$, there exists a negligible function ϵ such that

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

where the experiment $\mathsf{Exp}^{\mathsf{UF\text{-}IBS\text{-}CMA}}_{\mathsf{IBS},\mathcal{A}}(\lambda)$ that formalizes the unforgeability game is described in Fig.3.

Setup: The challenger $\mathcal{C}$ takes input the security parameter 1^λ and generate $(\mathsf{pp}_{\mathrm{ibs}}, \mathsf{msk}) \leftarrow \mathsf{IBS.Setup}(1^\lambda)$. It gives the public parameter $\mathsf{pp}_{\mathrm{ibs}}$ to $\mathcal{A}$ while keeps the master secret key msk secret to itself. Also it maintains three lists Klist, Clist and Mlist and initializes each to $\emptyset$.
Query Phase: $\mathcal{C}$ responds to polynomially many adaptive queries made by $\mathcal{A}$ as follows:
– *Oracle* $\mathcal{O}_{\mathsf{Extract}}(\cdot)$: On receiving a query on a user identity id from $\mathcal{A}$, $\mathcal{C}$ checks whether $(\mathsf{id}, \mathsf{usk}_{\mathsf{id}}) \in \mathsf{Kist}$. If so, it returns $\mathsf{usk}_{\mathsf{id}}$ and appends id to CList. Otherwise, it generates $\mathsf{usk}_{\mathsf{id}} \leftarrow \mathsf{IBS.Extract}(\mathsf{pp}_{\mathrm{ibs}}, \mathsf{msk}, \mathsf{id})$, returns $\mathsf{usk}_{\mathsf{id}}$ and appends $(\mathsf{id}, \mathsf{usk}_{\mathsf{id}})$ to Klist and id to Clist.
– *Oracle* $\mathcal{O}_{\mathsf{Sign}}(\cdot)$: On receiving a query on a message m and a user identity id from $\mathcal{A}$, $\mathcal{C}$ computes $\mathsf{usk}_{\mathsf{id}}$ as in the extraction query, except for appending identity id to Clist. It then computes a signature $\sigma \leftarrow \mathsf{IBS.Sign}(\mathsf{pp}_{\mathrm{ibs}}, \mathsf{usk}_{\mathsf{id}}, m)$ and appends (m, id, σ) to Mlist.
Forgery: The adversary $\mathcal{A}$ eventually outputs a message m^*, user identity id^* and a forge signature σ^*. The adversary $\mathcal{A}$ wins the game if $\mathsf{IBS.Verify}(\mathsf{pp}_{\mathrm{ibs}}, \mathsf{id}^*, m^*, \sigma^*) \rightarrow \mathsf{Valid}$ with the restriction that $\mathsf{id}^* \notin \mathsf{Clist}$ and $(m^*, \mathsf{id}^*, \cdot) \notin \mathsf{Mlist}$.

Fig. 3. $\mathsf{Exp}^{\mathsf{UF\text{-}IBS\text{-}CMA}}_{\mathsf{IBS},\mathcal{A}}(\lambda)$: Unforgeability under chosen identity and chosen message attacks

3.2 Puncturable Signature Scheme

Definition 3.21. A *puncturable signature* is a tuple $\mathsf{PS} = (\mathsf{PS.Setup}, \mathsf{PS.Puncture}, \mathsf{PS.Sign}, \mathsf{PS.Verify})$ of PPT algorithms associated with a message space $\mathcal{M}$ and prefix space $\mathcal{P}$ that satisfy the following requirements. Note that, if $x \in \mathcal{P}$, then there exists some $m \in M$ with prefix x and every message m has a unique prefix.

- $\mathsf{PS.Setup}(1^\lambda) \rightarrow (\mathsf{pp}_{\mathrm{ps}}, \mathsf{sk}_0)$: On input 1^λ, the signer executes this algorithm to generate the public parameter $\mathsf{pk}_{\mathrm{ps}}$ and initial secret key sk_0.
- $\mathsf{PS.Puncture}(\mathsf{sk}, x') \rightarrow \mathsf{sk}'$: The signer takes as input its secret key sk and a prefix $x' \in \mathcal{P}$ and runs this randomized algorithm to output an updated secret key sk'. We say the prefix x' has been punctured and refer the updated secret key sk' as a punctured secret key.
- $\mathsf{PS.Sign}(\mathsf{pp}_{\mathrm{ps}}, \mathsf{sk}, m) \rightarrow \Sigma/\ \perp$: Taking input $\mathsf{pp}_{\mathrm{ps}}$, secret key sk and a message $m \in \mathcal{M}$, the signer runs this randomized algorithm to generate a signature Σ if the prefix $x' \in \mathcal{P}$ has not been punctured. Otherwise, it returns $\perp$.
- $\mathsf{PS.Verify}(\mathsf{pp}_{\mathrm{ps}}, m, \Sigma) \rightarrow \mathsf{Valid}/\mathsf{Invalid}$: This is a deterministic algorithm that takes as input the public parameter $\mathsf{pp}_{\mathrm{ps}}$, a message m and a signature Σ. It outputs Valid if Σ is a valid signature on m and $\mathsf{Invalid}$ otherwise.

Correctness. The scheme PS is correct if it satisfies the following conditions:

i. For any message $m \in \mathcal{M}$, any prefix $x' \in \mathcal{P}$ and any $(\mathsf{pp}_{\mathrm{ps}}, \mathsf{sk}_0) \leftarrow \mathsf{PS.Setup}(1^\lambda)$, it holds that $\mathsf{PS.Verify}(\mathsf{pp}_{\mathrm{ps}}, m, \mathsf{PS.Sign}(\mathsf{pp}_{\mathrm{ps}}, \mathsf{sk}_0, m)) \rightarrow \mathsf{Valid}$ where sk_0 is the initial non-punctured secret key.

ii. For any message $m \in \mathcal{M}$ with prefix $x' \in \mathcal{P}$ which has been punctured with secret key sk, it holds that $\mathsf{PS.Verify}(\mathsf{pp}_{\mathrm{ps}}, m, \mathsf{PS.Sign}(\mathsf{pp}_{\mathrm{ps}}, \mathsf{sk}', m)) \rightarrow \mathsf{Invalid}$ where $\mathsf{sk}' \leftarrow \mathsf{PS.Puncture}(\mathsf{sk}, x')$ is the punctured secret key corresponding to the prefix x'.

iii. For any message $m \in \mathcal{M}$ with prefix $x \in \mathcal{P}$ which has not been punctured, we have $\mathsf{PS.Verify}(\mathsf{pp}_{\mathrm{ps}}, m, \mathsf{PS.Sign}(\mathsf{pp}_{\mathrm{ps}}, \mathsf{sk}', m)) \rightarrow \mathsf{Valid}$ where $\mathsf{sk}' \leftarrow \mathsf{PS.Puncture}(\mathsf{sk}, x')$ is the punctured secret key corresponding to the prefix $x' \neq x$ of a message m' with $m' \neq m$.

Definition 3.22. A puncturable signature scheme PS is secure against *existential unforgeability under chosen-message attacks with adaptive puncturing* ($\mathsf{UF\text{-}CMA\text{-}AP}$) if for all PPT adversary $\mathcal{A}$, there exists a negligible function ϵ such that

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

where the experiment $\mathsf{Exp}^{\mathsf{UF\text{-}CMA\text{-}AP}}_{\mathsf{PS},\mathcal{A}}(\lambda)$ is described in Fig. 4.

Setup: The challenger $\mathcal{C}$ takes input the security parameter 1^λ and generates $(\mathsf{pp}_{\mathrm{ps}}, \mathsf{sk}_0) \leftarrow \mathsf{PS.Setup}(1^\lambda)$. It forwards $\mathsf{pp}_{\mathrm{ps}}$ to $\mathcal{A}$ while keeps sk secret to itself. It also maintains the set $\mathcal{Q}_{\mathrm{sig}}$ for signed messages and the set $\mathcal{Q}_{\mathrm{pun}}$ for punctured prefixes and initializes each to $\emptyset$.
Query Phase: The adversary $\mathcal{A}$ issues polynomially many adaptive queries to the oracles $\mathcal{O}_{\mathsf{Puncture}}(\mathsf{sk}, \cdot)$ and $\mathcal{O}_{\mathsf{Sign}}(\mathsf{pp}_{\mathrm{ps}}, \mathsf{sk}, \cdot)$ as follows:
- $\mathcal{O}_{\mathsf{Puncture}}(\mathsf{sk}, \cdot)$: Upon receiving a query on prefix x', the challenger $\mathcal{C}$ generates a punctured secret key $\mathsf{sk}' \leftarrow \mathsf{Puncture}(\mathsf{sk}, x')$ and updates $\mathcal{Q}_{\mathrm{pun}} \leftarrow \mathcal{Q}_{\mathrm{pun}} \cup \{x'\}$.
- $\mathcal{O}_{\mathsf{Sgn}}(\mathsf{sk}, \cdot)$: On receiving a signature query on a message m with prefix $x' \in \mathcal{P}$, the challenger $\mathcal{C}$ checks if $x' \in \mathcal{Q}_{\mathrm{pun}}$. If the check succeeds, it returns $\perp$. Otherwise, it computes the signature $\Sigma \leftarrow \mathsf{PS.Sign}(\mathsf{pp}_{\mathrm{ps}}, \mathsf{sk}, m)$ on the message m and updates $\mathcal{Q}_{\mathrm{sig}} \leftarrow \mathcal{Q}_{\mathrm{sig}} \cup \{m\}$. It returns the computed signature Σ to the adversary $\mathcal{A}$.

Challenge: The adversary $\mathcal{A}$ sends a target prefix x^* to the challenger $\mathcal{C}$ and issues additional puncture and signature queries as described in the Query phase.
Corruption Query: $\mathcal{C}$ returns the current secret key sk^* if $x^* \in \mathcal{Q}_{\mathrm{pun}}$ and $\perp$ otherwise.
Forgery: The adversary $\mathcal{A}$ eventually submits a forgery (m^*, Σ^*, x^*) where x^* is the prefix of m^*. $\mathcal{A}$ wins the game if $m^* \notin \mathcal{Q}_{\mathrm{sig}}$, $x^* \in \mathcal{Q}_{\mathrm{pun}}$ and $\mathsf{Valid} \leftarrow \mathsf{PS.Verify}(\mathsf{pp}_{\mathrm{ps}}, m^*, \Sigma^*)$.

Fig. 4. $\mathsf{Exp}^{\mathsf{UF\text{-}CMA\text{-}AP}}_{\mathsf{PS},\mathcal{A}}(\lambda)$: Existential unforgeability under chosen-message attacks with adaptive puncturing

4 Our Identity-based Signature from SQISign

In this section, we propose our identity-based signature from $\mathsf{SQISign}$. We refer to our scheme as *Short Quaternion and Isogeny Identity-based Signatures* (SQIIBS).
$\mathsf{SQIIBS.Setup}(1^\lambda) \rightarrow (\mathsf{pp}_{\mathrm{ibs}}, \mathsf{msk})$: A KGC on input the security parameter 1^λ generates the public parameter $\mathsf{pp}_{\mathrm{ibs}}$ and a master secret key msk as follows:

i. Same as the algorithm $\mathsf{SQISign.Setup}$ described in Section 2.5. Additionally, it picks a random isogeny $\tau_1 : E_0 \rightarrow E_A^{(1)}$.

ii. Publishes the public parameter $\mathsf{pp}_{\mathrm{ibs}} = (p, E_0, D_c, D, \mathcal{H}_1, \Phi_{D_c}, E_A^{(1)})$ and keeps the master secret key $\mathsf{msk} = \tau_1$ secret to itself.

$\mathsf{SQIIBS.Extract}(\mathsf{pp}_{\mathrm{ibs}}, \mathsf{msk}, \mathsf{id}) \rightarrow \mathsf{usk}_{\mathsf{id}}$: On input the public parameter $\mathsf{pp}_{\mathrm{ibs}} = (p, E_0, D_c, D, \mathcal{H}_1, \Phi_{D_c}, E_A^{(1)})$, master secret key $\mathsf{msk} = \tau_1$ and an identity id, the KGC executes this algorithm to generate the user secret key $\mathsf{usk}_{\mathsf{id}}$ as follows:

i. Picks a random isogeny $\tau_2 : E_0 \to E_A^{(2)}$.
ii. Selects a random commitment isogeny $\psi_1 : E_0 \to E_1^{(1)}$.
iii. Computes $s_1 = \mathcal{H}_1(j(E_1^{(1)}), \mathsf{bin}(j(E_A^{(2)}))||\mathsf{id})$ and sets $\Phi_{D_c}(E_1^{(1)}, s_1) = \varphi_1$ where $\varphi_1 : E_1^{(1)} \to E_2^{(1)}$ is a non-backtracking isogeny of degree D_c.
iv. Computes the ideals $\bar{I}_{\tau_1}$, I_{τ_1}, I_{ψ_1} and I_{φ_1} corresponding to the isogenies $\hat{\tau_1}$, τ_1, ψ_1 and φ_1 respectively.
v. The KGC having the knowledge of $\mathcal{O}^{(1)} = \mathsf{End}(E_A^{(1)})$ through τ_1 and $\mathcal{O}_2^{(1)} = \mathsf{End}(E_2^{(1)})$ through $\varphi_1 \circ \psi_1 : E_0 \to E_2^{(1)}$, executes the $\mathsf{SigningKLPT}_{2^e}(I_{\tau_1}, I_1)$ algorithm (Section 2.3) on input the $(\mathcal{O}_0, \mathcal{O}^{(1)})$-ideal I_{τ_1} and a left $\mathcal{O}^{(1)}$-ideal $I_1 = I_{\varphi_1} I_{\psi_1} \bar{I}_{\tau_1}$ to obtain a $(\mathcal{O}^{(1)}, \mathcal{O}_2^{(1)})$-ideal $J_1 \sim I_1$ of norm $D = 2^e$.
vi. Constructs the isogeny $\eta_1 : E_A^{(1)} \to E_2^{(1)}$ of degree D corresponding to the ideal J_1 such that $\hat{\varphi_1} \circ \eta_1 : E_A^{(1)} \to E_1^{(1)}$ is cyclic and sets $\mathsf{cert_{id}} = (E_1^{(1)}, \eta_1)$.
vii. Issues the user secret key $\mathsf{usk_{id}} = (\tau_2, \mathsf{cert_{id}} = (E_1^{(1)}, \eta_1))$.

$\mathsf{SQIIBS.Sign}(\mathsf{pp_{ibs}}, \mathsf{usk_{id}}, m) \to \sigma$: On input $\mathsf{pp_{ibs}} = (p, E_0, D_c, D, \mathcal{H}_1, \Phi_{D_c}, E_A^{(1)})$, user secret key $\mathsf{usk_{id}} = (\tau_2, \mathsf{cert_{id}})$ and a message $m \in \{0,1\}^*$, the signer generates a signature σ on m as follows:

i. Picks a random commitment isogeny $\psi_2 : E_0 \to E_1^{(2)}$.
ii. Computes $s_2 = \mathcal{H}_1(j(E_1^{(2)}), m)$ and sets the challenge isogeny $\Phi_{D_c}(E_1^{(2)}, s_2) = \varphi_2$ where $\varphi_2 : E_1^{(2)} \to E_2^{(2)}$ is a non-backtracking isogeny of degree D_c.
iii. Computes the ideal $\bar{I}_{\tau_2}$, I_{τ_2}, I_{ψ_2} and I_{φ_2} corresponding to the isogenies $\hat{\tau_2}$, τ_2, ψ_2 and φ_2 respectively.
iv. The signer having the knowledge of $\mathcal{O}^{(2)} = \mathsf{End}(E_A^{(2)})$ through τ_2 and $\mathcal{O}_2^{(2)} = \mathsf{End}(E_2^{(2)})$ through $\varphi_2 \circ \psi_2$, executes the algorithm $\mathsf{SigningKLPT}_{2^e}(I_{\tau_2}, I_2)$ described in Section 2.3 on input the $(\mathcal{O}_0, \mathcal{O}^{(2)})$-ideal I_{τ_2} and a left $\mathcal{O}^{(2)}$-ideal $I_2 = I_{\varphi_2} I_{\psi_2} \bar{I}_{\tau_2}$ to obtain a $(\mathcal{O}^{(2)}, \mathcal{O}_2^{(2)})$-ideal $J_2 \sim I_2$ of norm D.
v. Constructs the isogeny $\eta_2 : E_A^{(2)} \to E_2^{(2)}$ of degree D corresponding to the ideal J_2 such that $\hat{\varphi_2} \circ \eta_2 : E_A^{(2)} \to E_1^2$ is cyclic and sets $\bar{\sigma} = (E_1^{(2)}, \eta_2)$.
vi. Extracts $\mathsf{cert_{id}}$ from $\mathsf{usk_{id}}$ and sets the signature $\sigma = (\bar{\sigma}, E_A^{(2)}, \mathsf{cert_{id}})$.

$\mathsf{SQIIBS.Verify}(\mathsf{pp_{ibs}}, \mathsf{id}, m, \sigma) \to \mathsf{Valid/Invalid}$: The verifier employing $\mathsf{pp_{ibs}} = (p, E_0, D_c, D, \mathcal{H}_1, \Phi_{D_c}, E_A^{(1)})$ verifies the validity of signature $\sigma = (\bar{\sigma}, E_A^{(2)}, \mathsf{cert_{id}})$ on $m \in \{0,1\}^*$ as follows:

i. Parses $\sigma = (\bar{\sigma} = (E_1^{(2)}, \eta_2), E_A^{(2)}, \mathsf{cert_{id}} = (E_1^{(1)}, \eta_1))$.
ii. Computes $s_1 = \mathcal{H}_1(j(E_1^{(1)}), \mathsf{bin}(j(E_A^{(2)}))||\mathsf{id})$ and $s_2 = \mathcal{H}_1(j(E_1^{(2)}), m)$.
iii. Recovers the isogenies $\Phi_{D_c}(E_1^{(1)}, s_1) = \varphi_1$ and $\Phi_{D_c}(E_1^{(2)}, s_2) = \varphi_2$.
iv. Checks if η_1 is an isogeny of degree D from $E_A^{(1)}$ to $E_2^{(1)}$ and that $\hat{\varphi_1} \circ \eta_1 : E_A^{(1)} \to E_1^{(1)}$ is cyclic.
v. Checks if η_2 is an isogeny of degree D from $E_A^{(2)}$ to $E_2^{(2)}$ and that $\hat{\varphi_2} \circ \eta_2 : E_A^{(2)} \to E_1^{(2)}$ is cyclic.
vi. If all the checks succeed returns Valid, otherwise returns $\mathsf{Invalid}$.

Correctness. The correctness of our proposed scheme SQIIBS follows immediately from the correctness of $\mathsf{SQISign}$ signature described in Section 2.5.

4.1 Efficiency

A theoretical comparison of our scheme SQIIBS with the existing works on IBS from isogenies is provided in Table 1 and Table 2. We compare our scheme with the CSIDH-based IBS scheme by Peng et al. [16] as well as the recently proposed IBS scheme by Shaw et al. [17]. Table 2 depicts that the secret key size and signature size of the existing IBS scheme grows with the value of S_1. The exponential size of $S_1 = 2^{\eta_1 - 1}$ leads to large key and signatures, making them impractical for real-life applications. The user secret key in our scheme comprises of an elliptic curve over the field $\mathbb{F}_{p^2}$ and two isogenies of degree 2^e. The elliptic curve is represented by its j-invariant and thus it is of size $2\log p$. As discussed in [6], an isogeny of degree 2^e can be compressed to e bits where $e = \frac{15}{4}\log p$. Thus the user secret key is of size $2\log p + 2(\frac{15}{4})\log p = 2\log p + \frac{15}{2}\log p$. The signature in our scheme comprises of three elliptic curves over $\mathbb{F}_{p^2}$ and two isogenies of degree 2^e. Thus, the signature in our scheme is of size $3(2\log p) + 2(\frac{15}{4})\log p = 6\log p + \frac{15}{2}\log p$. Our scheme enjoys improved efficiency in terms of key and signature sizes which thereby reduces the storage and communication cost.

Table 1. Comparison of our SQIIBS with existing IBS schemes

Scheme	Security Analysis	Rejection Sampling	Security
Peng et al.'s IBS[16]	✗	✓	CSIDH
Shaw et al.'s IBS [17]	✓	✗	CSI-FiSh
Our Work	✓	✗	SQISign

CSIDH = Commutative Supersingular Isogeny Diffie-Hellman, CSI-FiSh = Commutative Supersingular Isogeny based Fiat-Shamir signature, SQISign = Short Quaternion and Isogeny Signature.

Table 2. Comparison of secret and signature size of our SQIIBS with existing IBS schemes from isogenies

Scheme	$\lvert\mathsf{usk}_{\mathsf{id}}\rvert$	$\lvert\sigma\rvert$
Peng et al.'s IBS[16]	$nT_1S_1\log(2I_1+1) + T_1S_1\log p$	$T_1T_2[n\log(2I_2+1) + \log S_1] + T_1S_1\log p$
Shaw et al.'s IBS [17]	$T_1S_1[\log S_0 + \log N]$	$T_1T_2[\log N + \log S_1] + T_1S_1\log p$
Our Work	$2\log p + \frac{15}{2}\log p$	$6\log p + \frac{15}{2}\log p$

Here $n \in \mathbb{N}$, p is a prime, $I_0, I_1 = \delta_0 I_0$, $I_2 = \delta_1 I_1$, T_1, T_2, $S_0 = 2^{\eta_0} - 1$ and $S_1 = 2^{\eta_1} - 1$ are integers with $T_1 < S_0$ and $T_2 < S_1$. N is the size of ideal class group for CSIDH-512 parameter set.

4.2 Security Analysis

Theorem 4.21. *Our proposed scheme* SQIIBS *is* UF-IBS-CMA *secure as the underlying signature scheme* SQISign *is* UF-CMA *secure.*

Proof. Let us assume that there exists an adversary $\mathcal{A}$ that wins the UF-IBS-CMA game with non-negligible probability. At the end of the game, $\mathcal{A}$ outputs a valid forgery $(m^*, \mathsf{id}^*, \sigma^*)$ where $\sigma^* = (\bar{\sigma}^*, (E_A^{(2)})^*, \mathsf{cert}_{\mathsf{id}^*})$. We employ the adversary $\mathcal{A}$ as a subroutine to design an adversary $\mathcal{B}$ that breaks the UF-CMA security of the signature scheme SQISign. To complete the security reduction, $\mathcal{B}$ simulating the IBS security game with $\mathcal{A}$ must embed the public key given to $\mathcal{B}$ by its UF-CMA challenger $\mathcal{C}$ into some part of the "target" which $\mathcal{A}$ takes as a target of forgery.

There are two attack points in our construction. The adversary $\mathcal{A}$ may either take the public parameter pp_{ibs} provided by $\mathcal{B}$ or it reuses the components $\mathsf{cert}_{\mathsf{id}^*}$ and $(E_A^{(2)})^*$ of the answer of the signing oracle on id^* and message $m \neq m^*$ for its forgery. We denote the later event as "REUSE". Then the advantage of $\mathcal{A}$ is given by $\Pr[\mathsf{Success}] = \Pr[\mathsf{Success}|\neg\mathsf{REUSE}] + \Pr[\mathsf{Success}|\mathsf{REUSE}]$ where Success is the event that $\mathcal{A}$ wins in $\mathsf{Exp}^{\mathsf{UF\text{-}IBS\text{-}CMA}}_{\mathsf{IBS},\mathcal{A}}(\lambda)$. For each of the two cases $\neg$REUSE and REUSE, we give reductions as follows:

Case 1 $\Pr[\mathsf{Success}|\neg\mathsf{REUSE}]$**:** We describe below how the UF-CMA adversary $\mathcal{B}$ plays the role of the challenger and simulates the experiment $\mathsf{Exp}^{\mathsf{UF\text{-}IBS\text{-}CMA}}_{\mathsf{IBS},\mathcal{A}}(\lambda)$.

Setup: The UF-CMA challenger $\mathcal{C}$ generates the public parameter $\mathsf{pp}_{\text{sgn}} = (p, E_0, D_c, D, \mathcal{H}_1, \Phi_{D_c})$ by executing the algorithm $\mathsf{SQISign.Setup}(1^\lambda)$ and computes a secret-public key pair $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{SQISign.KeyGen}(\mathsf{pp}_{\text{sgn}})$ where $\mathsf{sk} = \tau_1$ and $\mathsf{pk} = E_A^{(1)}$ and forwards pp_{sgn} and pk to the adversary $\mathcal{B}$. It keeps sk secret to itself. The challenger $\mathcal{C}$ also maintains a list SList and initializes SList to $\emptyset$. Upon receiving $\mathsf{pp}_{\text{sgn}} = (p, E_0, D_c, D, \mathcal{H}_1, \Phi_{D_c})$ and $\mathsf{pk} = E_A^{(1)}$ from $\mathcal{C}$, $\mathcal{B}$ sets $\mathsf{pp}_{\text{ibs}} = (p, E_0, D_c, D, \mathcal{H}_1, \Phi_{D_c}, E_A^{(1)})$ and sends it to $\mathcal{A}$. It also initializes the lists Klist, Clist, Mlist to $\emptyset$.

Query Phase: The adversary $\mathcal{B}$ responds to polynomially many adaptive queries made by $\mathcal{A}$ to the oracles $\mathcal{O}_{\mathsf{Extract}}$ and $\mathcal{O}_{\mathsf{Sign}}$ as follows:

– *Oracle* $\mathcal{O}_{\mathsf{Extract}}(\cdot)$: On receiving a query on a user identity id from $\mathcal{A}$, $\mathcal{B}$ checks whether $(\mathsf{id}, \mathsf{usk}_{\mathsf{id}}) \in \mathsf{Kist}$. If there exists such a pair in Klist, it returns $\mathsf{usk}_{\mathsf{id}}$ to $\mathcal{A}$ and appends id to CList. If $(\mathsf{id}, \mathsf{usk}_{\mathsf{id}}) \notin \mathsf{Kist}$, $\mathcal{B}$ picks a random isogeny $\tau_2 : E_0 \to E_A^{(2)}$ and queries its signing oracle $\mathcal{O}_{\mathsf{Sign}}(\mathsf{sk} = \tau_1, \cdot)$ simulated by $\mathcal{C}$ on the message $\mathsf{bin}(j(E_A^{(2)}))||\mathsf{id}$. Upon receiving the signature $\mathsf{cert}_{\mathsf{id}} = (E_1^{(1)}, \eta_1)$ from $\mathcal{C}$, the adversary $\mathcal{B}$ sets $\mathsf{usk}_{\mathsf{id}} = (\tau_2, \mathsf{cert}_{\mathsf{id}})$ and returns it to $\mathcal{A}$. The adversary $\mathcal{B}$ also appends $(\mathsf{id}, \mathsf{usk}_{\mathsf{id}})$ to Klist and id to Clist. The challenger $\mathcal{C}$ appends $\mathsf{bin}(j(E_A^{(2)}))||\mathsf{id}$ in Slist.

– *Oracle* $\mathcal{O}_{\mathsf{Sign}}(\cdot)$: On receiving a query on a message $m \in \{0,1\}^*$ and a user identity id from $\mathcal{A}$, $\mathcal{B}$ retrieves the pair $(\mathsf{id}, \mathsf{usk}_{\mathsf{id}})$ from Klist where $\mathsf{usk}_{\mathsf{id}} = (\tau_2, \mathsf{cert}_{\mathsf{id}})$ is the user secret key corresponding to id. If $(\mathsf{id}, \mathsf{usk}_{\mathsf{id}}) \notin \mathsf{Kist}$, $\mathcal{B}$ picks a random isogeny $\tau_2 : E_0 \to E_A^{(2)}$ and queries its signing oracle $\mathcal{O}_{\mathsf{S}}(\mathsf{sk} = \tau_1, \cdot)$ on the message $\mathsf{bin}(j(E_A^{(2)}))||\mathsf{id}$. Upon receiving the signature $\mathsf{cert}_{\mathsf{id}} = (E_1^{(1)}, \eta_1)$ under $\mathsf{sk} = \tau_1$ from $\mathcal{C}$, $\mathcal{B}$ sets $\mathsf{usk}_{\mathsf{id}} = (\tau_2, \mathsf{cert}_{\mathsf{id}})$. It then executes $\bar{\sigma} = (E_1^{(2)}, \eta_2) \leftarrow \mathsf{SQISign.Sign}(\mathsf{pp}_{\text{sgn}}, \tau_2, m)$, sets the signature $\sigma = (\bar{\sigma}, E_A^{(2)}, \mathsf{cert}_{\mathsf{id}})$ and sends it to $\mathcal{A}$. It also appends (m, id, σ) to Mlist.

Forgery: The adversary $\mathcal{A}$ eventually outputs a message m^*, user identity id^* and a forge signature σ^* where $\sigma^* = (\bar{\sigma}^*, (E_A^{(2)})^*, \mathsf{cert}_{\mathsf{id}^*})$. If $\mathcal{A}$ wins the UF-IBS-CMA game with non-negligible probability then $(m^*, \mathsf{id}^*, \sigma^*)$ must be a valid forgery. Thus, $\mathsf{IBS.Verify}(\mathsf{pp}_{\text{ibs}}, \mathsf{id}^*, m^*, \sigma^*) \to \mathsf{Valid}$ where $\mathsf{id}^* \notin \mathsf{Clist}$ and $(m^*, \mathsf{id}^*, \cdot) \notin \mathsf{Mlist}$. The adversary $\mathcal{B}$ submits $\mathsf{bin}(j((E_A^{(2)})^*))||\mathsf{id}^*, \mathsf{cert}_{\mathsf{id}^*}$ as a forgery to its own challenger $\mathcal{C}$.

The event $\neg$REUSE means $(\mathsf{id}^*, \mathsf{usk}_{\mathsf{id}^*}) \notin \mathsf{Klist}$ where $\mathsf{usk}_{\mathsf{id}^*} = (\tau_2^*, \mathsf{cert}_{\mathsf{id}^*})$. This implies that $(\mathsf{bin}(j((E_A^{(2)})^*))||\mathsf{id}^*) \notin \mathsf{Slist}$. Hence, the adversary $\mathcal{B}$ has output the valid forgery $(\mathsf{bin}(j((E_A^{(2)})^*))||\mathsf{id}^*, \mathsf{cert}_{\mathsf{id}}^*)$ such that $\mathsf{SQISign.Veriy}(\mathsf{pp}_{\mathrm{sgn}}, E_A^{(1)}, \mathsf{bin}(j((E_A^{(2)})^*))||\mathsf{id}^*, \mathsf{cert}_{\mathsf{id}}^*) \rightarrow \mathsf{Valid}$. From the security of $\mathsf{SQISign}$, it follows that $\Pr[\mathsf{Success}|\neg\mathsf{REUSE}]$ is negligible.

Case 2 $\Pr[\mathsf{Success}|\mathsf{REUSE}]$**:** In this case the adversary $\mathcal{A}$ reuses the components $\mathsf{cert}_{\mathsf{id}^*}$ and $(E_A^{(2)})^*$ of the answer of the signing oracle query on identity id^* and message $m \neq m^*$ for its forgery.

Setup: The UF-CMA challenger $\mathcal{C}$ generates the public parameter $\mathsf{pp}_{\mathrm{sgn}} = (p, E_0, D_c, D, \mathcal{H}_1, \Phi_{D_c})$ by executing the algorithm $\mathsf{SQISign.Setup}(1^\lambda)$ as in **Case 1** and computes a secret-public key pair $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{SQISign.KeyGen}(\mathsf{pp}_{\mathrm{sgn}})$ where $\mathsf{sk} = \tau_2$ and $\mathsf{pk} = E_A^{(2)}$ and forwards $\mathsf{pp}_{\mathrm{sgn}}$ and pk to the adversary $\mathcal{B}$. It keeps sk secret to itself. The challenger $\mathcal{C}$ maintains a list SList and initializes SList to $\emptyset$. Upon receiving $\mathsf{pp}_{\mathrm{sgn}} = (p, E_0, D_c, D, \mathcal{H}_1, \Phi_{D_c})$ and $\mathsf{pk} = E_A^{(2)}$ from the challenger $\mathcal{C}$, the adversary $\mathcal{B}$ picks a random isogeny $\tau_1 : E_0 \rightarrow E_A^{(1)}$, sets $\mathsf{pp}_{\mathrm{ibs}} = (p, E_0, D_c, D, \mathcal{H}_1, \Phi_{D_c}, E_A^{(1)})$, $\mathsf{msk} = \tau_1$ and sends $\mathsf{pp}_{\mathrm{ibs}}$ to $\mathcal{A}$. It initializes the lists Klist, Clist, Mlist to $\emptyset$ and chooses $r \leftarrow \{1, 2, \ldots, q(\lambda)\}$ where $q(\lambda)$ is the maximum number of queries by $\mathcal{A}$.

Query Phase: The adversary $\mathcal{B}$ responds to polynomially many adaptive queries to the oracles $\mathcal{O}_{\mathsf{Extract}}$ and $\mathcal{O}_{\mathsf{Sign}}$ made by $\mathcal{A}$. Let id' be the identity for which the r^{th} signing query of $\mathcal{A}$ was made.

– *Oracle* $\mathcal{O}_{\mathsf{Extract}}(\cdot)$: If $\mathcal{A}$ ever makes an extract query for the identity id', the experiment is aborted. On receiving a query on a user identity $\mathsf{id} \neq \mathsf{id}'$ from $\mathcal{A}$, $\mathcal{B}$ checks whether $(\mathsf{id}, \mathsf{usk}_{\mathsf{id}}) \in \mathsf{Kist}$. If there exists such a pair in Klist, it returns $\mathsf{usk}_{\mathsf{id}}$ and appends id to CList. If $(\mathsf{id}, \mathsf{usk}_{\mathsf{id}}) \notin \mathsf{Kist}$, it picks a random isogeny $\bar{\tau}_2 : E_0 \rightarrow \overline{E}_A^{(2)}$ and uses $\mathsf{msk} = \tau_1$ to compute $\overline{\mathsf{cert}}_{\mathsf{id}} = (\overline{E}_1^{(1)}, \bar{\eta}_1) \leftarrow \mathsf{SQISign.Sign}(\mathsf{pp}_{\mathrm{sgn}}, \tau_1, \mathsf{bin}(j(\overline{E}_A^{(2)}))||\mathsf{id})$. It then sets $\overline{\mathsf{usk}}_{\mathsf{id}} = (\bar{\tau}_2, \overline{\mathsf{cert}}_{\mathsf{id}})$ and returns it to $\mathcal{A}$. It appends $(\mathsf{id}, \overline{\mathsf{usk}}_{\mathsf{id}})$ to Klist and id to Clist.

– *Oracle* $\mathcal{O}_{\mathsf{Sign}}(\cdot)$: The adversary $\mathcal{B}$ receives signing queries on pairs (m, id) from the adversary $\mathcal{A}$. For the r^{th} signing query on (id', m) by $\mathcal{A}$, $\mathcal{B}$ first checks whether $(m, \mathsf{id}', \sigma) \in \mathsf{Mlist}$. If there exists such a tuple, the adversary $\mathcal{B}$ aborts the experiment. Otherwise, $\mathcal{B}$ computes $\mathsf{cert}_{\mathsf{id}'} = ((E_1^{(1)})', \eta_1') \leftarrow \mathsf{SQISign.Sign}(\mathsf{pp}_{\mathrm{sgn}}, \tau_1, \mathsf{bin}(j(E_A^{(2)}))||\mathsf{id}')$ using $\mathsf{msk} = \tau_1$ and queries its signing oracle $\mathcal{O}_{\mathsf{S}}(\mathsf{sk} = \tau_2, \cdot)$ on m. Upon receiving the signature $\bar{\sigma} = (E_1^{(2)}, \eta_2)$ on m from $\mathcal{C}$ under secret key $\mathsf{sk} = \tau_2$, $\mathcal{B}$ sets $\sigma' = (\bar{\sigma}, E_A^{(2)}, \mathsf{cert}_{\mathsf{id}'})$ and sends it to $\mathcal{A}$. The adversary $\mathcal{B}$ updates the Mlist with $(m, \mathsf{id}', \sigma')$ and the challenger $\mathcal{C}$ updates Slist with m. For the i^{th} query where $i \in \{r+1, \ldots, q(\lambda)\}$, on identity id' and a message m' by $\mathcal{A}$, the adversary $\mathcal{B}$ checks whether $(m', \mathsf{id}', \sigma') \in \mathsf{Mlist}$. If such a tuple exists, $\mathcal{B}$ answers the query from the Mlist, otherwise it proceeds as in the r^{th} signing query.

Upon receiving a query on a message m and identity $\mathsf{id} \neq \mathsf{id}'$, $\mathcal{B}$ retrieves the pair $(\mathsf{id}, \overline{\mathsf{usk}_{\mathsf{id}}})$ from Klist where $\overline{\mathsf{usk}_{\mathsf{id}}} = (\bar{\tau}_2, \overline{\mathsf{cert}_{\mathsf{id}}})$ is the user secret key corresponding to id. If $(\mathsf{id}, \mathsf{usk}_{\mathsf{id}}) \notin \mathsf{Kist}$, it picks a random isogeny $\bar{\tau}_2 : E_0 \rightarrow \overline{E}_A^{(2)}$ and uses its master secret key $\mathsf{msk} = \tau_1$ to compute $\overline{\mathsf{cert}_{\mathsf{id}}} = (\overline{E}_1^{(1)}, \bar{\eta}_1) \leftarrow \mathsf{SQISign.Sign}(\mathsf{pp}_{\mathrm{sgn}}, \tau_1, \mathsf{bin}(j(\overline{E}_A^{(2)}))||\mathsf{id})$ and sets $\overline{\mathsf{usk}_{\mathsf{id}}} = (\bar{\tau}_2, \overline{\mathsf{cert}_{\mathsf{id}}})$. It then computes the signature $\bar{\sigma} = (\overline{E}_1^{(2)}, \bar{\eta}_2) \leftarrow \mathsf{SQISign.Sign}(\mathsf{pp}_{\mathrm{ibs}}, \bar{\tau}_2, m)$ on m and sets $\sigma = (\bar{\sigma}, \overline{E}_A^{(2)}, \overline{\mathsf{cert}_{\mathsf{id}}})$ and sends it to $\mathcal{A}$. It appends (m, id, σ) to Mlist.

Forgery: If $\mathcal{A}$ eventually outputs a message m^*, user identity id^* and a forge signature σ^* where $\sigma^* = (\bar{\sigma}^*, (E_A^{(2)})^*, \mathsf{cert}_{\mathsf{id}^*})$ and the experiment was never aborted, $\mathcal{B}$ submits (m^*, σ^*) as a forgery to its own challenger $\mathcal{C}$. If $\mathcal{A}$ wins the $\mathsf{UF\text{-}IBS\text{-}CMA}$ game with non-negligible probability then $(m^*, \mathsf{id}^*, \sigma^*)$ must be a valid forgery. Thus, we have $\mathsf{IBS.Verify}(\mathsf{pp}_{\mathrm{ibs}}, \mathsf{id}^*, m^*, \sigma^*) = \mathsf{Valid}$, $\mathsf{id}^* \notin \mathsf{Clist}$ and $(m^*, \mathsf{id}^*, \cdot) \notin \mathsf{Mlist}$. Note that the condition $(m^*, \mathsf{id}^*, \cdot) \notin \mathsf{Mlist}$ means that the adversary $\mathcal{B}$ never queried its signing oracle $\mathcal{O}_{\mathsf{S}}(\tau_2, \cdot)$ on m^*.

With probability at least $1/q(\lambda)$, the experiment is not aborted and $\mathsf{id}' = \mathsf{id}^*$. The success probability of $\mathcal{B}$ in forging a signature for $\mathsf{SQISign}$ is thus at least $\Pr[\mathsf{Success}|\mathsf{REUSE}]/q(\lambda)$. From the security of $\mathsf{SQISign}$, it follows that this quantity must be negligible. Since q is polynomial in λ, we must have $\Pr[\mathsf{Success}|\mathsf{REUSE}]$ is negligible as well.

5 Puncturable Signature : Concrete Construction

We now describe our Short Quaternion and Isogeny Puncturable Signature (SQIPS) leveraging our scheme SQIIBS described in Section 4. Let $\mathcal{M} = \{0,1\}^*$ denotes the message space and $\mathcal{P} = \{0,1\}^l \subseteq \mathcal{M}$ be the prefix space of our PS.

$\mathsf{SQIPS.Setup}(1^\lambda) \rightarrow (\mathsf{pp}_{\mathrm{ps}}, \mathsf{sk}_0)$: On input 1^λ, the signer executes this algorithm to generate the public parameter $\mathsf{pk}_{\mathrm{ps}}$ and initial secret key sk as follows:

i. Invokes the algorithm $\mathsf{SQIIBS.Setup}(1^\lambda)$ to compute the key pair $(\mathsf{pp}_{\mathrm{ibs}}, \mathsf{msk})$ as follows:
   - Chooses a prime p and fixes the supersingular curve $E_0 : y^2 = x^3 + x$ over $\mathbb{F}_{p^2}$ with special extremal endomorphism ring $\mathcal{O}_0 = \langle 1, i, \frac{i+j}{2}, \frac{1+k}{2} \rangle$.
   - Picks a smooth number $D = 2^e$ where $2^e > p^3$.
   - Picks an odd smooth number $D_c = \ell^e$ where ℓ is a prime and computes $\mu(D_c) = (\ell + 1) \cdot \ell^{e-1}$.
   - Samples a cryptographic hash function $\mathcal{H}_1 : \mathbb{F}_{p^2} \times \{0,1\}^* \rightarrow [1, \mu(D_c)]$.
   - Samples an arbitrary function $\Phi_{D_c}(E, s)$ that maps a pair (E, s) of an elliptic curve E and an integer $s \in [1, \mu(D_c)]$ to a non-backtracking isogeny of degree D_c from E [3].
   - Picks a random isogeny $\tau_1 : E_0 \rightarrow E_A^{(1)}$.
   - Sets $\mathsf{pp}_{\mathrm{ibs}} = (p, E_0, D_c, D, \mathcal{H}_1, \Phi_{D_c}, E_A^{(1)})$ and $\mathsf{msk} = \tau_1$.

ii. For each prefix $x' \in \{0,1\}^l$, executes the algorithm $\mathsf{SQIIBS.Extract}(\mathsf{pp}_{\mathrm{ibs}}, \mathsf{msk} = \tau_1, x')$ to compute the key $\mathsf{usk}_{x'}$ and stores it in an array T of size 2^l.

- Picks a random isogeny $\tau_2 : E_0 \to E_A^{(2)}$.
- Selects a random commitment isogeny $\psi_1 : E_0 \to E_1^{(1)}$.
- Computes $s_1 = \mathcal{H}_1(j(E_1^{(1)}), \mathsf{bin}(j(E_A^{(2)}))||x')$ and sets the challenge isogeny $\Phi_{D_c}(E_1^{(1)}, s_1) = \varphi_1$ where $\varphi_1 : E_1^{(1)} \to E_2^{(1)}$ is a non-backtracking isogeny of degree D_c.
- Computes the ideals $\bar{I}_{\tau_1}$, I_{τ_1}, I_{ψ_1} and I_{φ_1} corresponding to the isogenies $\hat{\tau_1}$, τ_1, ψ_1 and φ_1 respectively.
- The signer having the knowledge of $\mathcal{O}^{(1)} = \mathsf{End}(E_A^{(1)})$ through τ_1 and $\mathcal{O}_2^{(1)} = \mathsf{End}(E_2^{(1)})$ through $\varphi_1 \circ \psi_1$, runs the $\mathsf{SigningKLPT}_{2^e}(I_{\tau_1}, I_1)$ algorithm (Section 2.3) on input the $(\mathcal{O}_0, \mathcal{O}^{(1)})$-ideal I_{τ_1} and a left $\mathcal{O}^{(1)}$-ideal $I_1 = I_{\varphi_1} I_{\psi_1} \bar{I}_{\tau_1}$ to obtain a $(\mathcal{O}^{(1)}, \mathcal{O}_2^{(1)})$-ideal $J_1 \sim I_1$ of norm $D = 2^e$.
- Constructs the isogeny $\eta_1 : E_A^{(1)} \to E_2^{(1)}$ of degree D corresponding to the ideal J_1 such that $\hat{\varphi_1} \circ \eta_1$ is cyclic and $\mathsf{cert}_{x'} = (E_1^{(1)}, \eta_1)$.
- Issues the user secret key $\mathsf{usk}_{x'} = (\tau_2, \mathsf{cert}_{x'} = (E_1^{(1)}, \eta_1))$.

iii. Sets $T[\mathsf{ind}_{x'}] = \mathsf{usk}_{x'}$ where $\mathsf{ind}_{x'} = (x')_{10} \in \{0, 1, \ldots, 2^l - 1\}$ is the decimal representation of the binary string x'.

iv. Sets the public parameter $\mathsf{pp}_{\mathrm{ps}} = \mathsf{pp}_{\mathrm{ibs}}$ and secret key $\mathsf{sk} = T$.

$\mathsf{SQIPS.Puncture}(\mathsf{sk}, x') \to \mathsf{sk}'$: The signer on input the secret key $\mathsf{sk} = T$ and a prefix $x' \in \{0,1\}^l$, computes $\mathsf{ind}_{x'} = (x')_{10}$ and sets $T[\mathsf{ind}] = 0$. It returns the updated punctured secret key $\mathsf{sk}' = T$ where the value corresponding to the index ind of the array T is made 0.

$\mathsf{SQIPS.Sign}(\mathsf{pp}_{\mathrm{ps}}, \mathsf{sk}, m) \to \Sigma / \perp$: Taking input $\mathsf{pp}_{\mathrm{ps}} = (p, E_0, D_c, D, \mathcal{H}_1, \Phi_{D_c}, E_A^{(1)})$, secret key $\mathsf{sk} = T$ and a message $m \in \{0,1\}^*$, the signer either generates a signature Σ if the prefix x' of m has not been punctured or it returns $\perp$.

i. Returns $\perp$ if $T[\mathsf{ind}_{x'}] = 0$.

ii. If $T[\mathsf{ind}_{x'}] \neq 0$, it retrieves the value $\mathsf{usk}_{x'} = (\tau_2, \mathsf{cert}_{x'} = (E_1^{(1)}, \eta_1)) = T[\mathsf{ind}_{x'}]$ from the array and executes the algorithm $\mathsf{SQIIBS.Sign}(\mathsf{pp}_{\mathrm{ibs}}, \mathsf{usk}_{x'}, m)$ as follows to generate a signature on m.

- Picks a random commitment isogeny $\psi_2 : E_0 \to E_1^{(2)}$.
- Computes $s_2 = \mathcal{H}_1(j(E_1^{(2)}), m)$ and $\Phi_{D_c}(E_1^{(2)}, s_2) = \varphi_2$ where $\varphi_2 : E_1^{(2)} \to E_2^{(2)}$ is a non-backtracking challenge isogeny of degree D_c.
- Computes the ideal $\bar{I}_{\tau_2}$, I_{τ_2}, I_{ψ_2} and I_{φ_2} corresponding to the isogenies $\hat{\tau_2}$, τ_2, ψ_2 and φ_2 respectively.
- The signer having the knowledge of $\mathcal{O}^{(2)} = \mathsf{End}(E_A^{(2)})$ through τ_2 and $\mathcal{O}_2^{(2)} = \mathsf{End}(E_2^{(2)})$ through $\varphi_2 \circ \psi_2$, runs the $\mathsf{SigningKLPT}_{2^e}(I_{\tau_2}, I_2)$ algorithm (Section 2.3) on input the $(\mathcal{O}_0, \mathcal{O}^{(2)})$-ideal I_{τ_2} and a left $\mathcal{O}^{(2)}$-ideal $I_2 = I_{\varphi_2} I_{\psi_2} \bar{I}_{\tau_2}$ to obtain a $(\mathcal{O}^{(2)}, \mathcal{O}_2^{(2)})$-ideal $J_2 \sim I_2$ of norm $D = 2^e$.
- Constructs the isogeny $\eta_2 : E_A^{(2)} \to E_2^{(2)}$ of degree D corresponding to the ideal J_2 such that $\hat{\varphi_2} \circ \eta_2 : E_A^{(2)} \to E_1^{(2)}$ is cyclic. It sets $\bar{\sigma} = (E_1^{(2)}, \eta_2)$.
- Extract $\mathsf{cert}_{x'}$ from $\mathsf{usk}_{x'}$ and sets the signature $\sigma = (\bar{\sigma}, E_A^{(2)}, \mathsf{cert}_{x'})$.

iii. Returns the puncturable signature $\Sigma = \sigma$.

$\mathsf{SQIPS.Verify}(\mathsf{pp}_{\mathrm{ps}}, m, \Sigma) \rightarrow \mathsf{Valid/Invalid}$: This algorithm takes as input $\mathsf{pp}_{\mathrm{ps}} = (p, E_0, D_c, D, \mathcal{H}_1, \Phi_{D_c}, E_A^{(1)})$, a message $m \in \{0,1\}^*$ and a signature $\Sigma = \sigma = (\bar{\sigma}, E_A^{(2)}, \mathsf{cert}_{x'})$ where $x' \in \{0,1\}^l$ is the prefix of the message $m \in \{0,1\}^*$. It outputs Valid if Σ is a valid signature on m and $\mathsf{Invalid}$ otherwise.

i. Executes the algorithm $\mathsf{SQIIBS.Verify}$ as follows to check the validity of the signature $\Sigma = \sigma = (\bar{\sigma}, E_A^{(2)}, \mathsf{cert}_{x'})$ on m.
   - Parses $\sigma = (\bar{\sigma} = (E_1^{(2)}, \eta_2), E_A^{(2)}, \mathsf{cert}_{x'} = (E_1^{(1)}, \eta_1))$.
   - Computes $s_1 = \mathcal{H}_1(j(E_1^{(1)}), \mathsf{bin}(j(E_A^{(2)}))||x')$ and $s_2 = \mathcal{H}_1(j(E_1^{(2)}), m)$.
   - Recovers the isogenies $\Phi_{D_c}(E_1^{(1)}, s_1) = \varphi_1$ and $\Phi_{D_c}(E_1^{(2)}, s_2) = \varphi_2$.
   - Checks if η_1 is an isogeny of degree D from $E_A^{(1)}$ to $E_2^{(1)}$ and that $\hat{\varphi_1} \circ \eta_1 : E_A^{(1)} \rightarrow E_1^{(1)}$ is cyclic.
   - Checks if η_2 is an isogeny of degree D from $E_A^{(2)}$ to $E_2^{(2)}$ and that $\hat{\varphi_2} \circ \eta_2 : E_A^{(2)} \rightarrow E_1^{(2)}$ is cyclic.
   - If all the checks succeed returns Valid, otherwise returns $\mathsf{Invalid}$.

Correctness. The correctness of our puncturable signature scheme SQIPS from isogenies follows from the correctness of our identity-based signature SQIIBS.

Theorem 5.01. *Our proposed puncturable signature* SQIPS *is* $\mathsf{UF\text{-}CMA\text{-}AP}$ *secure as the underlying identity-based signature* SQIIBS *is* $\mathsf{UF\text{-}IBS\text{-}CMA}$ *secure.*

Proof. Let us assume that there exists a PPT adversary $\mathcal{A}$ that wins the experiment $\mathsf{Exp}^{\mathsf{UF\text{-}CMA\text{-}AP}}_{\mathsf{SQIPS},\mathcal{A}}(\lambda)$ depicted in Fig 4 with a non-negligible advantage. We design an adversary $\mathcal{B}$ who simulates the PS security experiment $\mathsf{Exp}^{\mathsf{UF\text{-}CMA\text{-}AP}}_{\mathsf{SQIPS},\mathcal{A}}(\lambda)$, exploits $\mathcal{A}$ as a subroutine and wins the IBS security experiment $\mathsf{Exp}^{\mathsf{UF\text{-}IBS\text{-}CMA}}_{\mathsf{SQIIBS},\mathcal{B}}(\lambda)$ with the same advantage. Let $\mathcal{C}$ denotes the challenger for the security experiment $\mathsf{Exp}^{\mathsf{UF\text{-}IBS\text{-}CMA}}_{\mathsf{SQIIBS},\mathcal{B}}(\lambda)$.

Setup: The challenger $\mathcal{C}$ on input the security parameter 1^λ, computes $(\mathsf{pp}_{\mathrm{ibs}}, \mathsf{msk}) \leftarrow \mathsf{SQIIBS.Setup}(1^\lambda)$ and sends $\mathsf{pp}_{\mathrm{ibs}}$ to $\mathcal{B}$. Additionally, $\mathcal{C}$ executes the algorithm $\mathsf{SQIIBS.Extract}(\mathsf{pp}_{\mathrm{ibs}}, \mathsf{msk}, x')$ to compute the key $\mathsf{usk}_{x'}$ for each prefix $x' \in \{0,1\}^l$ and forms the array $T[\mathsf{ind}_{x'}] = \mathsf{usk}_{x'}$. Also it initiates three lists Klist, Clist and Mlist to $\emptyset$. Upon receiving the public parameter $\mathsf{pp}_{\mathrm{ibs}}$ from its own challenger $\mathcal{C}$, the adversary $\mathcal{B}$ sets $\mathsf{pp}_{\mathrm{ps}} = \mathsf{pp}_{\mathrm{ibs}}$ and forwards it to $\mathcal{A}$. It also initializes the sets $\mathcal{Q}_{\mathrm{sig}}$ for signed messages and $\mathcal{Q}_{\mathrm{pun}}$ for punctured prefixes to ϕ.

Query Phase: The adversary $\mathcal{A}$ issues polynomially many adaptive queries to the following oracles $\mathcal{O}_{\mathsf{Puncture}}(\mathsf{sk}, \cdot)$ and $\mathcal{O}_{\mathsf{Sgn}}(\mathsf{sk}, \cdot)$.
- $\mathcal{O}_{\mathsf{Puncture}}(\mathsf{sk} = T, \cdot)$: Upon receiving a query on prefix x', the challenger $\mathcal{C}$ updates $\mathcal{Q}_{\mathrm{pun}} \leftarrow \mathcal{Q}_{\mathrm{pun}} \cup \{x'\}$.
- $\mathcal{O}_{\mathsf{Sgn}}(\mathsf{sk} = T, \cdot)$: On receiving a signature query on a message $m \in \{0,1\}^*$, the adversary $\mathcal{B}$ checks if $x' \in \mathcal{Q}_{\mathrm{pun}}$ where x' is the prefix of m. If the check succeeds, it returns $\perp$. Otherwise, it issues a signature query on (m, x') for a with message m and identity x' to $\mathcal{C}$. The challenger $\mathcal{C}$ extracts $T[\mathsf{ind}_{x'}] = \mathsf{usk}_{x'}$ from $\mathsf{sk} = T$, computes the signature $\Sigma \leftarrow$

$\mathsf{SQIIBS.Sign}(\mathsf{pp}_{\mathrm{ibs}}, \mathsf{usk}_{x'}, m)$ and sends it to $\mathcal{B}$ who forwards it to $\mathcal{A}$. The adversary $\mathcal{B}$ updates $\mathcal{Q}_{\mathrm{sig}} \leftarrow \mathcal{Q}_{\mathrm{sig}} \cup \{m\}$.

Challenge: The adversary $\mathcal{A}$ sends a target prefix $x^* \in \{0,1\}^l$ to the adversary $\mathcal{B}$ which $\mathcal{B}$ forwards to $\mathcal{C}$ as the target identity. The adversary $\mathcal{A}$ can issue additional puncture and signature queries as described in the Query phase.

Corruption Query: Upon receiving a corruption query on $x^* \in \{0,1\}^l$, the adversary $\mathcal{B}$ returns $\perp$ if $x^* \notin \mathcal{Q}_{\mathrm{pun}}$. Otherwise, $\mathcal{B}$ queries its extract oracle $\mathcal{O}_{\mathsf{Extract}}(\cdot)$ for each prefix $x' \in \{0,1\}^l \setminus \{x^*\}$ and updates the array T with the response $\mathsf{usk}_{x'} \leftarrow \mathsf{SQIIBS.Extract}(\mathsf{pp}_{\mathrm{ibs}}, \mathsf{msk}, x')$ from $\mathcal{C}$ by setting $T[\mathsf{ind}_{x'}] = \mathsf{usk}_{x'}$. For each $x' \in \mathcal{Q}_{\mathrm{pun}}$, the adversary $\mathcal{B}$ deletes the related key by setting $T[\mathsf{ind}_{x'}] = 0$ and returns the current secret key $\mathsf{sk} = T$ to $\mathcal{A}$.

Forgery: $\mathcal{A}$ eventually submits a forgery (m^*, Σ^*, x^*) where x^* is the prefix of m^*. $\mathcal{B}$ uses the forgery of $\mathcal{A}$ to frame its own forgery (m^*, x^*, Σ^*).

If the adversary $\mathcal{A}$ wins the game then we have $m^* \notin \mathcal{Q}_{\mathrm{sig}}$, $x^* \in \mathcal{Q}_{\mathrm{pun}}$ and $\mathsf{Valid} \leftarrow \mathsf{SQIPS.Verify}(\mathsf{pp}_{\mathrm{ps}}, m^*, \Sigma^*)$. The condition $m^* \notin \mathcal{Q}_{\mathrm{sig}}$ means that $(m^*, x^*, \cdot) \notin \mathsf{Mlist}$. Also note that the adversary $\mathcal{B}$ has not made any extraction query on x^*, thus $x^* \notin \mathsf{Clist}$. Moreover, $\mathsf{Valid} \leftarrow \mathsf{SQIPS.Verify}(\mathsf{pp}_{\mathrm{ps}}, m^*, \Sigma^*)$ implies that $\mathsf{Valid} \leftarrow \mathsf{SQIIBS.Verify}(\mathsf{pp}_{\mathrm{ibs}}, m^*, \Sigma^*)$.

5.1 Comparison of our scheme SQIPS *with the existing puncturable signatures*

In Table 3, we compare our scheme with the existing schemes on PS. The PS scheme by Li et al. [13] is based on the τ-Strong Diffie-Hellman assumption (τ-SDH) in bilinear map setting and is proven secure in the random oracle model (ROM). Their scheme employs the probabilistic bloom filter data structure and suffers from non-negligible false-positive errors. Jiang et al. [12] designed a pairing-based PS which is free from false positive errors and is secure under the hardness of the Computational Diffie-Hellman (CDH) assumption in the standard model (SDM). However, none of these schemes are resistant to quantum attacks. The PS schemes from lattices and MPKC proposed by Jiang et al. [12] enjoy post-quantum security and are based on the hardness of Short Integer Solution (SIS) and Multivariate Quadratic polynomial (MQ) assumptions respectively. Our isogeny-based PS is post-quantum secure as it is based on SQISign cryptosystem and is also free from false-positive errors.

Table 3. Comparison of the existing puncturable signature schemes

Instantiation	Assumption	Security Model	Post-quantum	False-positive errors
Li et al. [13]	τ-SDH	ROM	✗	✓
Pairing Inst. [12]	CDH	SDM	✗	✗
Lattice Inst. [12]	SIS	ROM	✓	✗
Multivariate Inst. [12]	MQ		✓	✗
Our Isogeny Inst.	SQISign	ROM	✓	✗

References

1. Bellare, M., Stepanovs, I., Waters, B.: New negative results on differing-inputs obfuscation. In: Advances in Cryptology–EUROCRYPT 2016: 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II 35. pp. 792–821. Springer (2016)
2. Castryck, W., Lange, T., Martindale, C., Panny, L., Renes, J.: CSIDH: an efficient post-quantum commutative group action. In: Advances in Cryptology–ASIACRYPT 2018: 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2–6, 2018, Proceedings, Part III 24. pp. 395–427. Springer (2018)
3. Charles, D.X., Lauter, K.E., Goren, E.Z.: Cryptographic hash functions from expander graphs. Journal of CRYPTOLOGY 22(1), 93–113 (2009)
4. Chen, J., Ling, J., Ning, J., Ding, J.: Identity-based signature schemes for multivariate public key cryptosystems. The Computer Journal 62(8), 1132–1147 (2019)
5. Childs, A., Jao, D., Soukharev, V.: Constructing elliptic curve isogenies in quantum subexponential time. Journal of Mathematical Cryptology 8(1), 1–29 (2014)
6. De Feo, L., Kohel, D., Leroux, A., Petit, C., Wesolowski, B.: SQISign: compact post-quantum signatures from quaternions and isogenies. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 64–93. Springer (2020)
7. Deirmentzoglou, E., Papakyriakopoulos, G., Patsakis, C.: A survey on long-range attacks for proof of stake protocols. IEEE Access 7, 28712–28725 (2019)
8. Deuring, M.: Die typen der multiplikatorenringe elliptischer funktionenk ́orper. In: Abhandlungen aus dem mathematischen Seminar der Universit ́at Hamburg. vol. 14, pp. 197–272. Springer (1941)
9. Gaži, P., Kiayias, A., Russell, A.: Stake-bleeding attacks on proof-of-stake blockchains. In: 2018 Crypto Valley Conference on Blockchain Technology (CVCBT). pp. 85–92. IEEE (2018)
10. Guan, J., Zhandry, M.: Disappearing cryptography in the bounded storage model. In: Theory of Cryptography: 19th International Conference, TCC 2021, Raleigh, NC, USA, November 8–11, 2021, Proceedings, Part II 19. pp. 365–396. Springer (2021)
11. Halevi, S., Ishai, Y., Jain, A., Komargodski, I., Sahai, A., Yogev, E.: Non-interactive multiparty computation without correlated randomness. In: Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part III. pp. 181–211. Springer (2017)
12. Jiang, M., Duong, D.H., Susilo, W.: Puncturable Signature: A Generic Construction and Instantiations. In: Computer Security–ESORICS 2022: 27th European Symposium on Research in Computer Security, Copenhagen, Denmark, September 26–30, 2022, Proceedings, Part II. pp. 507–527. Springer (2022)
13. Li, X., Xu, J., Fan, X., Wang, Y., Zhang, Z.: Puncturable signatures and applications in proof-of-stake blockchain protocols. IEEE Transactions on Information Forensics and Security 15, 3872–3885 (2020)
14. Minkowski, H.: Uber die positiven quadratischen Formen und iiber Kettenbruchanliche Algorithmen. Ges. Abh. I pp. 243–260
15. Pass, R., Seeman, L., Shelat, A.: Analysis of the blockchain protocol in asynchronous networks. In: Advances in Cryptology–EUROCRYPT 2017: 36th Annual

International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30–May 4, 2017, Proceedings, Part II. pp. 643–673. Springer (2017)
16. Peng, C., Chen, J., Zhou, L., Choo, K.K.R., He, D.: CsiIBS: A post-quantum identity-based signature scheme based on isogenies. Journal of information Security and Applications 54, 102504 (2020)
17. Shaw, S., Dutta, R.: Identification scheme and forward-secure signature in identity-based setting from isogenies. In: Provable and Practical Security: 15th International Conference, ProvSec 2021, Guangzhou, China, November 5–8, 2021, Proceedings 15. pp. 309–326. Springer (2021)
18. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM review 41(2), 303–332 (1999)
19. Tian, M., Huang, L.: Identity-based signatures from lattices: simpler, faster, shorter. Fundamenta Informaticae 145(2), 171–187 (2016)
20. Yi, P., Li, J., Liu, C., Han, J., Wang, H., Zhang, Y., Chen, Y.: An efficient identity-based signature scheme with provable security. Information Sciences 576, 790–799 (2021)