Towards Post-Quantum Updatable Public-Key Encryption via Supersingular Isogenies

Edward Eaton, David Jao, and Chelsea Komlo

University of Waterloo

Abstract. In this work, we present the first post-quantum secure Updatable Public-Key Encryption (UPKE) construction. UPKE has been proposed in the literature as a mechanism to improve the forward-secrecy and post-compromise security of secure messaging protocols, but the hardness of all existing constructions to date rely on discrete logarithm assumptions. We focus our assessment on isogeny-based cryptosystems due to their suitability for performing a potentially unbounded number of update operations, a practical requirement for secure messaging where user conversations can occur over months, if not years.
We begin by formalizing two UPKE variants presented in the literature as Symmetric and Asymmetric UPKE. At a fundamental level, these variants differ in how encryption and decryption keys are updated, and consequently impact the design and security model for quantum-safe constructions.
We demonstrate that Asymmetric UPKE *cannot* be instantiated using existing isogeny-based constructions. We then describe a SIDH-based Symmetric UPKE construction that is possible in theory but requires improving existing mathematical limitations before a practical implementation is possible. Finally, we present a CSIDH-based Symmetric UPKE construction that can be instantiated using a parameter set in which the class group structure is fully known to ensure efficient uniform sampling and canonical representation to prevent leakage of secret keys. We discuss several open problems which are applicable to any cryptosystem with similar requirements for continuous operations over elements in the secret domain.

Keywords: isogenies, post-quantum cryptography, public-key encryption, secure messaging

1 Introduction

Secure communication protocols are quickly evolving [26, 25, 3], driven by the need to meet simultaneous usability and security requirements, such as asynchronous communication while ensuring forward secrecy and post-compromise security for conversations that can occur over months, if not years. *Updatable Public-Key Encryption* (UPKE) schemes have been proposed as a seemingly simple protocol solution to improve weak forward secrecy properties of existing secure messaging protocols such as the Signal and Message Layer Security (MLS) protocols [11, 23, 1, 19, 31, 2, 30]. In addition to standard public-key encryption functionality, UPKE schemes allow encryption and decryption keys to be asynchronously *updated* with fresh entropy, thereby *healing* the protocol by restoring security even after exposure against an adversary that can temporarily compromise a victim's local state. Unfortunately, the security of all UPKE schemes proposed to date relies on the hardness of the discrete logarithm problem.

In this work, we perform the first assessment of the viability of quantum-secure UPKE schemes, and present the first post-quantum secure UPKE construction. To guide our analysis, we formalize two UPKE variants presented in the literature which we call *Symmetric UPKE* and *Asymmetric UPKE*, and assess the extent to which existing isogeny-based cryptosystems can instantiate both variants. We model Asymmetric UPKE after a construction proposed by Jost, Maurer, and Mularczyk [23], in which encryption keys are updated using elements in the public domain, while decryption keys are updated using private values. We model Symmetric UPKE after a construction proposed by Alwen et al. [1] to improve the forward-secrecy and post-compromise security of TreeKEM [6], the group key-exchange primitive used by MLS, where both encryption and decryption keys are updated using the *same* private update value.

We demonstrate that Asymmetric UPKE *cannot* be instantiated by either Supersingular Isogeny Diffie Hellman (SIDH) nor Commutative Supersingular Isogeny Diffie-Hellman (CSIDH) due to limitations in their algebraic structure. We then present a series of steps demonstrating that SIDH can in theory be used for Symmetric UPKE constructions, but in practice is hindered due to existing mathematical limitations. We then present a CSIDH-based Symmetric UPKE construction which can be used today with the existing CSIDH-512 parameter set, or any CSIDH parameter set where the class group structure is fully known. The requirement of knowing the class group structure is needed to allow for unique group element representation and uniform sampling. Taken together, these properties mean that the value of a secret key after an update has occurred is independent of the previous secret key, providing strong forward secrecy and post-compromise guarantees.

We focus our analysis on isogeny-based cryptosystems as alternative quantum-secure cryptographic primitives have undesirable usability or efficiency trade-offs for secure messaging protocols, or simply cannot support the algebraic structure required for UPKE. In the setting of secure messaging, user conversations can potentially endure months, if not years, and so supporting ongoing protocol actions without bounds on the number of consecutive update operations is desirable. For example, lattice-based cryptosystems accumulate errors for each additional operation, and so require either bounding the number of operations or performing some expensive compression function to limit growth of errors [16, 17]. Code-based primitives similarly accumulate errors over repeated operations, and so have similar restrictions on the number of possible operations that can be performed [27]. Finally, multivariate and hash-based primitives are not a good fit for key-exchange protocols in general, much less protocols with more advanced requirements such as updatability[1].

Some Open Problems. As we demonstrate later in our work, while in theory SIDH can be used to instantiate a Symmetric UPKE construction, such a construction cannot be instantiated today due to existing mathematical limitations. Specifically, more efficient KLPT algorithms that can output isogenies of degree less than some prime p are required for our SIDH construction in order to guarantee forward-secrecy and post-compromise security for updated secret keys. However, the state of the art in KLPT algorithms today [14] can at best output an isogeny of degree $\frac{15}{4}\log_\ell(p)$ for $\lambda = 128$ bits in security, where the size of p is 2λ bits.

For CSIDH, the ability to sample group elements uniformly and a unique representation of those elements is needed to instantiate an efficient construction that is not limited in the number of update operations. However, such an improvement requires fully computing a class group $cl(\mathcal{O})$, where the only existing parameter sets [5] may provide less than 64 bits of post-quantum security [28, 7, 10]. Without such a structure, the distribution of the updated secret key is dependent on both the original secret as well as the update value, thereby exposing an information leak to an adversary that can temporarily compromise the victim's local state. Hence, similarly to constructions such as CSI-FiSh, our CSIDH-based UPKE construction would benefit from stronger CSIDH parameter sets whose class group can be fully computed. While today's classical computers can perform such operations in sub-exponential time (and so computing larger parameter sets today may be infeasible), at worst, quantum computers can certainly perform such operations more efficiently.

Finally, we wish to emphasize that any cryptosystem with similar requirements for continuous operations over elements in the secret domain will face similar limitations, so these improvements will have wider benefits beyond our constructions.

Contributions. In this work, we assess and give constructions for post-quantum secure updatable public-key encryption (UPKE) schemes using isogeny-based cryptosystems. Towards this end, we present the following contributions:

– We give formal definitions of Symmetric UPKE and Asymmetric UPKE as two UPKE variants previously presented in the literature. We also present IND-CPA-U, a formalization and generalized security model for proving IND-CPA security specifically for UPKE schemes, a setting in which the adversary is assumed to be able to adaptively choose update values as well as corrupt secret key material.

[1] Hash functions only rely on one-way functions and thus cannot provide the structure needed for key exchange. Multivariate schemes are generally built from a surjective trapdoor function that is difficult to build a key exchange protocol from, and with no obvious algebraic structure to allow for updating.

– We assess the capability of two isogeny-based public-key cryptosystems—SIDH and CSIDH—to instantiate Asymmetric and Symmetric UPKE constructions. Towards this end, we assess the algebraic structure defined by SIDH and CSIDH, and provide a comparison to two structures commonly used for classically secure Diffie-Hellman operations.

– We determine that Asymmetric UPKE *cannot* be instantiated by either SIDH or CSIDH due to limitations in their algebraic structure.

– We describe a series of steps leading to a SIDH-based Symmetric UPKE construction that is possible in theory, but requires further mathematical advancements to be instantiated in practice. We describe these advancements and how such improvements apply to related cryptographic protocols.

– We present a Symmetric UPKE construction that can be used today with CSIDH-512, or any CSIDH parameter set where the class group has been fully computed. We prove this construction to be IND-CPA-U secure, and also provide an implementation.

Related Work The most closely related work to our own is the already mentioned Alwen et al. work [1]; our Symmetric UPKE primitive is modeled after their construction. However, as mentioned, the work by Alwen et al. defines and applies an updatable public-key encryption scheme to TreeKEM as a mechanism to improve forward secrecy. This present paper is an effort to construct updatabale public-key encryption primitives using supersingular isogenies, to define a post-quantum variant suitable for similar use in asynchronous group-based key-exchange protocols akin to TreeKEM. Further, we prove security in a more robust model that models the adversary's capability to both adaptively choose update values for the victim as well as corrupt their local state. The work by Alwen et al. on updatable public-key encryption was in turn based upon work by Jost et al. [23], which is most akin to our Asymmetric UPKE notion.

Both secure messaging protocols and post-quantum protocols are still in active development. Efforts to combine the two (post-quantum secure messaging) are so far rare in the literature, although we refer to [8] as a recent example of exactly this. Their work constructs a version of Signal's X3DH protocol out of the (ring)-LWE problem.

Finally, we note that there exists a separate notion of "updatable encryption" in the literature [22]. In these schemes, a ciphertext is updated using an update token such that the encrypted message becomes an encryption under a new public key without the need to decrypt the message in the process. This notion should not be confused with the schemes we present here, which use update tokens to update the public key and rely on asymmetric encryption.

Organization. We give an overview of isogenies and isogeny-based cryptosystems in Section 2 and of updatable public-key encryption schemes in Section 3. In Section 4 we assess the extent to which SIDH and CSIDH can be used to instantiate Asymmetric and Symmetric UPKE constructions. We present a series of steps toward a SIDH Symmetric UPKE construction in Section 5 and a CSIDH Symmetric UPKE construction in Section 6. We conclude in Section 7.

2 Isogenies and Isogeny-Based Cryptography

Let p be a prime number and $\mathbb{F}_p$ as a finite field of characteristic p, and let E_0 and E_1 be elliptic curves defined over $\mathbb{F}_p$. An isogeny $\phi : E_0 \to E_1$ is a rational map from $E_0(\mathbb{F}_p)$ to $E_1(\mathbb{F}_p)$ which is also a group homomorphism [33]. Two curves over the same finite field are considered identical if they are the same up to isomorphism; such a correspondence can be determined by comparing the *j-invariant* of each curve, which remains a constant value for all isomorphic curves.

The endomorphism of an elliptic curve $\phi : E \to E$ is a rational map from E to itself, defined over an extension field $\mathbb{F}_{p^n}$. The set of all endomorphisms for an elliptic curve forms a ring under the operation of point-wise addition and composition; we denote this ring of endomorphisms as $End(E)$. When $End(E)$ is isomorphic to an order in a quaternion algebra, the curve is classified as *supersingular*, otherwise, $End(E)$ is isomorphic to to an imaginary quadratic field and the curve is classified as *ordinary*[33].

The *Deuring correspondence* proves that there is a one-to-one mapping between supersingular curves over the quadratic extension field $\mathbb{F}_{p^2}$ and maximal orders in a quaternion algebra, up to isomorphism. In short, $End(E)$ is isomorphic to a maximal order $\mathcal{O}$ in $\mathbf{Q}_{p,\infty}$ (the quaternion algebra over $\mathbf{Q}$ ramified at p and ∞), and for each $\mathcal{O} \in \mathbf{Q}_{p,\infty}$, there exists an isomorphism to some supersingular curve E defined over $\mathbb{F}_{p^2}$.

We next describe two existing cryptosystems—SIDH and CSIDH—whose security has been demonstrated to reduce to the hardness of the Supersingular Isogeny Problem, described in Definition 1.

Definition 1. ***Supersingular Isogeny Problem*** *[21] Given a finite field K and two supersingular elliptic curves E_1, E_2 defined over K such that $|E_1| = |E_2|$, compute an isogeny $\phi : E_1 \to E_2$*

2.1 Supersingular Isogeny Diffie-Hellman (SIDH)

Introduced by Jao and De Feo in 2011 [21], SIDH is a Diffie-Hellman like scheme defined using secret isogenies between supersingular elliptic curves to perform a key exchange protocol between two parties. SIDH can also be constructed as a PKE scheme [21], and has been adapted as a KEM with additional Fujisaki-Okamoto techniques [18] as an IND-CCA2 secure candidate for the ongoing NIST competition to standardize quantum-resistant key exchange protocols [20].

Performing key exchange via SIDH begins with each party agreeing to a starting public curve $E_0(\mathbb{F}_{p^2})$, where p is a prime of the form $2^{e_1}3^{e_2} + 1$, and a set of auxiliary points $((P_A, Q_A), (P_B, Q_B)) \in E_0$. For this work, we assume secret values—which define the kernel of an isogeny—are of the form $\langle [1]P + [n]Q \rangle$, such that participants only need to randomly generate the scalar n to define their secret key. Correspondingly, Alice selects $n_A \overset{\$}{\leftarrow} \mathbb{Z}_{2^{e_1}}$ which defines her secret isogeny $\phi_A : E_0 \to E_A$, such that $E_A = E_0/\langle [1]P_A + [n_A]Q_A \rangle$. Similarly, Bob selects $n_B \overset{\$}{\leftarrow} \mathbb{Z}_{3^{e_2}}$, which defines his secret isogeny $\phi_B : E_0 \to E_B$, such that $E_B = E_0/\langle [1]P_B + [n_B]Q_B \rangle$. Alice publishes her public key $(E_A, \phi_A(P_B), \phi_A(Q_B))$, while Bob publishes his public key $(E_B, \phi_B(P_A), \phi_B(Q_A))$.

After obtaining each other's public values, their shared secret is the j-invariant of the resulting secret curve E_{AB}, which we denote as $j(E_{AB})$. Alice arrives at E_{AB} by calculating $E_B/\langle \phi_B(P_A) + [n_A]\phi_B(Q_A) \rangle$ using her secret term n_A, whereas Bob calculates $E_A/\langle \phi_A(P_B) + [n_B]\phi_A(Q_B) \rangle$ using his secret term n_B. Alice and Bob obtain the same shared secret by finding the j-invariant of E_{AB}, as their resulting values

$$E_B/\langle \phi_B(P_A) + [n_A]\phi_B(Q_A) \rangle \cong E_A/\langle \phi_A(P_B) + [n_B]\phi_A(Q_B) \rangle$$

are equal up to isomorphism.

2.2 Commutative Supersingular Isogeny Diffie-Hellman (CSIDH)

As described in Section 2.1, SIDH performs "Diffie-Hellman-like" operations over a set of supersingular elliptic curves over the quadratic extension field $\mathbb{F}_{p^2}$ and isogenies between these curves. SIDH ensures commutativity of key-exchange operations between two participants by additional sending auxiliary points on each participant's public curves. Participants arrive at the same curve up to isomorphism by "dividing out" the starting elliptic curve by two non-intersecting subgroups.

CSIDH [9] builds upon the Conveignes-Rostovisev-Stolbunov [12, 32] scheme, but instead uses the graph of supersingular curves. The security of CSIDH is based on the Supersingular Isogeny problem defined in Definition 1, but CSIDH restricts the supersingular graph (where nodes are supersingular curves, and edges are isogenies) to curves defined over $\mathbb{F}_p$.

While the full ring of endomorphisms of supersingular curves over $\mathbb{F}_{p^2}$ (an order in a quaternion algebra) is non-commutative, restricting this graph to the subring of curves defined over $\mathbb{F}_p$ (also an order in a quaternion algebra) allows for commutativity of multiplication within the subring. To ensure that isogeny operations can be computed efficiently using Vélu's formula, p in CSIDH is defined to be of the form $p = 4 \cdot \ell_1 \cdot \ell_2, \ldots, \cdot \ell_d - 1$ for some set of d small primes. This set of ideals $[\ell]$ generates the class group $cl(\mathcal{O})$.

Similarly to SIDH, participants performing a key-exchange must agree to some starting curve curve $E_0(\mathbb{F}_p)$. A secret key in CSIDH is a vector $\boldsymbol{e} \in \mathbb{Z}^d$; each element in $\boldsymbol{e}$ is within some specified bound to ensure that these values are "small". This value $\boldsymbol{e}$ represents a secret ideal $\prod_{i=1}^{d} {\ell_i}^{e_i}$. By the Deuring correspondence, we know that this ideal corresponds to exactly one isogeny from the starting curve E_0 to some other curve in the graph.

Signature Schemes Building upon CSIDH: CSI-FiSh Recently, an efficient signature scheme based on the hardness of CSIDH-512 was presented [5]. While CSIDH has always been framed as a group action, there are some limitations to using it as such. The specific structure of the class group $cl(\mathcal{O})$ was previously not known, which made sampling from the group uniformly and having a unique representation of group elements impossible. The authors of [5] solved this problem by explicitly computing the structure of the $cl(\mathcal{O})$, which required a sub-exponential computation. They found that the group was cyclic, and the value N such that $cl(\mathcal{O}) \cong \mathbb{Z}_N$. As well, the authors described a methodology to translate the representation of group elements from $\mathbb{Z}_N$ to vectors $\boldsymbol{e} \in \mathbb{Z}^d$, so that the same method of applying the secret ideal $\prod_{i=1}^{d} \ell_i^{e_i}$ can be used.

This allows anyone to sample uniform group elements and have unique representations, so long as the CSIDH-512 parameter set is used. While the authors of CSI-FiSh used these properties to define a signature scheme, we will show how the same structure is useful in building updatable public-key encryption.

2.3 KLPT Algorithm

While SIDH and CSIDH rely on the hardness of finding a path in the ℓ-isogeny graph between two challenge curves (vertices in the supersingular graph), a "quaternion analog" of this ℓ-isogeny problem solves for a relation between the representation of the endomorphism rings of two curves as a maximal order of a quaternion algebra. In short, given a maximal order $\mathcal{O} \in \mathbf{Q}_{p,\infty}$, the problem is to find a left $\mathcal{O}_0$-ideal J with norm ℓ^k, and another left ideal $\mathcal{O}_0$ ideal I with the same norm. Doing so in turn corresponds to finding an isomorphism between the two challenge curves).

The KLPT algorithm—named for its authors Kohel, Lauter, Petit, and Tignol—presents one such algorithm for solving this ℓ-isogeny problem [24]. This algorithm accepts as input an integral I and finds an equivalent ideal $J \sim I$ for the specified norm. Such an algorithm is useful to produce an isogeny $\eta : E_0 \to E_2$ from two isogenies $\phi \circ \psi : E_0 \to E_2$, where $\phi : E_0 \to E_1$ and $\psi : E_1 \to E_2$, such that η is *independent* to ϕ and ψ.

The KLPT algorithm first described by Kohel, Lauter, Petit, and Tignol [24] and then improved by Petit and Smith [29] produces isogenies of degree $\frac{9}{2}\log_\ell(p)$ as output. However, recent work by De Feo, Leroux, Petit, and Wesolowski on an isogeny-based signature scheme SQI-Sign [14] improves the degree of this output to $\frac{15}{4}\log_\ell(p)$.

3 Updatable Public-Key Encryption (UPKE)

We first present definitions for updatable public-key encryption (UPKE) variants described in the literature. We then review the notions of security for any UPKE scheme.

Note that while Alwen et al. [1] use "updatable public-key encryption" to refer to the Symmetric UPKE scheme we define in Section 3.1[2], we use the UPKE terminology in a more generalized sense to refer to the broader class of any public-key encryption scheme that satisfy the notions we define below.

We do not include in our analysis optimally secure UPKE constructions that rely on identity-based encryption [19, 31], due to their lack of efficiency in comparison to real-world protocols such as MLS and Signal.

3.1 Definitions

We begin by formalizing the notion of a generalized UPKE scheme. We then present two variants of UPKE schemes presented previously in the literature [1, 23], which differ in how update values are generated and applied to an existing public-key encryption keypair.

Definition 2. Updatable Public-Key Encryption *An Updatable Public-Key Encryption scheme $\mathcal{U}$ is defined by six algorithms: a key generation algorithm KeyGen, an encryption algorithm Encrypt, a decryption algorithm Decrypt, an algorithm to generate update values GenerateUpdate, and protocols to update the private and public keys UpdatePrivate, UpdatePublic, respectively.*

The notions that any UPKE scheme should satisfy are as follows:

– *Correctness*: The scheme should correctly perform public-key encryption and decryption both before and after a series of update operations.
– *Forward secrecy*: If an attacker learns the secret key for epoch n, the updated secret keys in epochs $1 \ldots, n-1$ should not be recoverable by the attacker.
– *Post-compromise security*: If an attacker learns the secret key for epoch n, all updated secret keys in epochs $n, n+1, \ldots$ should not be recoverable by the attacker.
– *Asynchronicity*: Anyone with knowledge of a public key should be able to initiate an update, so that the update operation to the public key is immediately available, and only the update operation to the secret key should be performed *eventually.*
– *Key Indistinguishability*: An adversary that has access to both a freshly generated keypair and an updated keypair has a negligible advantage to distinguish between the two. Note that while such a property may be desirable in practice for privacy reasons, our reason for requiring this property is that we need it in our IND-CPA-U proof. However, it is possible that alternative proof strategies may not require key indistinguishability to prove IND-CPA-U security.

We provide a formalization for how UPKE schemes fulfill these notions in Section 3.2.

[2] Other terms in the literature also have been introduced such as "key-updatable public key encryption" for variants modeled by our Asymmetric UPKE variant [23, 19]; we simply use UPKE as the broader term to refer to any public-key encryption scheme that allows for key updates of any form.

Symmetric Updatable Public-Key Encryption We model the notion of Symmetric UPKE after a construction described by Alwen et al. [1] as a mechanism to improve the forward-secrecy and post-compromise security properties of TreeKEM, the group key-exchange primitive used to achieve key agreement for MLS. In the construction by Alwen et al., the sender of a message will also generate an update value, which is transmitted privately to the holder of the decryption key as part of the ciphertext along with the message. As such, the sender of a message can apply the update value to the other party's encryption key, and then the update value is applied by the receiver of the message to their decryption key. Notably, Symmetric UPKE uses the *same* update value μ when performing *UpdatePublic* and *UpdatePrivate*, meaning that the encryption and decryption keys are both updated using μ.

More formally, Symmetric UPKE instantiates $\mathcal{U}$ as follows:

KeyGen: Produces as output a public key pk and secret key sk.

Encrypt: Encrypts a message m using the public key pk, resulting in a ciphertext c.

Decrypt: Decrypts the ciphertext c using sk, producing as output the plaintext message m.

GenerateUpdate: Produces as output a randomly-generated update value μ.

UpdatePrivate: Takes as input the secret key sk and the update value μ and produces a deterministic output that is the updated secret key sk'.

UpdatePublic: Takes as input the public key pk and the update value μ, and produces a deterministic output that is the updated public key pk'. This operation is intended to be performed independently from *UpdatePrivate* by an entity that is *not* the same entity that performed *KeyGen*, as otherwise the keypair owner could simply perform *KeyGen* to generate completely fresh keys.

Correctness. Symmetric UPKE constructions are correct if they correctly perform *Encrypt* and *Decrypt* operations both before and after a series of *UpdatePublic* and *UpdatePrivate* operations have been performed.

Asymmetric Updatable Public-Key Encryption We model the notion of Asymmetric UPKE after a construction proposed by Jost, Maurer, and Mularczyk [23] that is based on the ElGamal cryptosystem to improve the security of secure messaging protocols such as Signal. In their construction, the owner of the decryption key first generates a private update value and applies this update to their decryption key, and then generates a public update value (corresponding to the private update value), and sends this public update value to the other party, who applies it to the encryption key. In short, in the Asymmetric UPKE setting, encryption and decryption keys are updated using *different* update values. Notably, the update value applied to the encryption key is in the public domain, while the update value applied to the decryption key is assumed to be secret.

More formally, Asymmetric UPKE schemes instantiate $\mathcal{U}$ as follows, where *KeyGen*, *Encrypt*, *Decrypt* remain identical to the Symmetric setting:

GenerateUpdate: Produces as output an update value μ that is a tuple comprised of public and private update values μ_{sk}, μ_{pk}.

UpdatePrivate: Accepts as input a secret key *sk* and a secret update value μ_{sk}, and produces as output an updated secret key sk'.

UpdatePublic: Accepts as input a public key *pk* and a update value μ_{pk}, and produces as output a updated public key pk'.

Correctness. As in the Symmetric setting, Asymmetric UPKE constructions are correct if both before and after performing a series of *UpdatePublic* and *UpdatePrivate* operations, *Encrypt* and *Decrypt* operations can be correctly performed.

3.2 Security

The security of UPKE schemes can be formalized using a modified notion of Indistinguishability under Chosen Plaintext Attacks (IND-CPA) security first introduced by Alwen et al. [1]. We now present a generalization of Alwen et al.'s definition of IND-CPA security which we define as "Indistinguishability under Chosen Plaintext Attacks with Updatability", or IND-CPA-U. Note that we present this notion of IND-CPA-U security assuming a Symmetric UPKE construction, but this security model extends easily to the Asymmetric UPKE setting by simply allowing the adversary to learn public update values when an honest update operation has been performed.

In Alwen et al.'s definition, the adversary is given the public key pk_0 and provides a sequence of updates $\mu_1, \ldots, \mu_\tau$. The public and private keys are updated accordingly and the adversary is issued an IND-CPA challenge under pk_τ. The public and secret key are updated again, this time with a secret update, and the adversary is given the resulting public *and* secret key. They then must respond to the IND-CPA challenge.

Their model illustrates the fundamental idea behind how security works for updatable encryption: the adversary may learn (or even control) either the update value or the secret key, but as long as they do not have both, the updated secret key remains secure. However they have the restriction that the adversary controls the updates prior to the IND-CPA challenge, and receives the secret key afterwards. We generalize this by allowing the adversary to adaptively choose whether they want to control the update or learn a secret key, with the restriction that the IND-CPA challenge can only be issued on a public key that has not been compromised in a straightforward way.

As with the Alwen et al. model, we begin by generating a keypair (pk_0, sk_0) and sending pk_0 to $\mathcal{A}$. We initialize $i \leftarrow 0$ (the most recent version of the keypair will be (pk_i, sk_i)). After this we let the adversary decide how the key will be updated. To this end, we provide our adversary with the following oracles they may call:

– The GiveUpdate(μ) oracle takes in an update value μ. It increments $i \leftarrow i+1$, and then generates

$$(pk_i, sk_i) \leftarrow UpdatePublic(pk_{i-1}, \mu), UpdatePrivate(sk_{i-1}, \mu)$$

before providing the new pk_i to the adversary.
– The FreshUpdate() oracle corresponds to updates happening that the adversary does not control or know the update value for. It generates a random $\mu \xleftarrow{\$} GenerateUpdate()$ and then calls GiveUpdate(μ), providing the new pk_i to the adversary.
– The Corrupt(j) oracle provides sk_j for a index $j \leq i$.

Eventually, the adversary requests a challenge on an index $j \leq i$ and provides messages (m_0, m_1). A bit $b \xleftarrow{\$} \{0,1\}$ is sampled and $c_b \leftarrow Encrypt(m_b, pk_j)$ is provided back to the adversary. After making further queries to the update and corruption oracles, the adversary must issue a bit b'. They are said to win if $b = b'$ and the index j is *fresh*. The freshness requirement ensures that the adversary cannot trivially win.

An index j is considered fresh if:

– The adversary has not called Corrupt(j), and
– There is not a sequence of updates (in either direction) all of which from GiveUpdate that connects the index j to an index k for which Corrupt(k) has been called.

In Figure 1 we visualize how the queries that the adversary has performed cause a given index to be considered fresh or not.

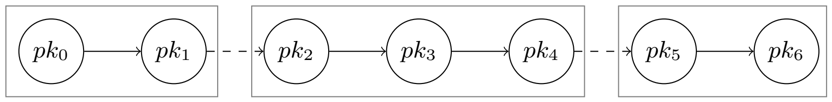

Fig. 1. A series of updates applied to public keys. In this diagram, updates that occur as a result of GiveUpdate queries are solid, while updates that are a result of FreshUpdate queries are dashed. Keys that are connected by updates that the adversary has provided or has knowledge of can be viewed as a block. If one index in a given block is compromised, leaking the secret key, then the secret key for any index in the block can be calculated. In our security model, such indices are not considered fresh, and if the adversary requests the challenge on such an index, they are penalized.

We denote the advantage of $\mathcal{A}$ as $CPAU_{adv} = 2 \cdot \Pr[\mathcal{A} \text{ wins}] - 1$.

Definition 3. *A UPKE scheme is IND-CPA-U secure if for any polynomial-time adversary $\mathcal{A}$, the value of $CPAU_{adv}$ is negligible.*

Unlike IND-CPA games for plain PKE schemes, the IND-CPA-U definition presented in Definition 3 captures the notion of forward secrecy and post-compromise security by allowing $\mathcal{A}$ to learn any secret key material and provide whatever update values that it wishes, and then determines the extent to which this knowledge grants the adversary additional advantage within the IND-CPA game, for which we enforce that the adversary does not know the encryption key. In short, updates that the adversary does not control "resets" their advantage to a negligible amount, even if they have compromised the secret key prior to the update (in other words, the scheme is post-compromise secure). Similarly, if the adversary learns the secret key after an honest update has occurred, their advantage in the IND-CPA game is negligible for that secret key prior to the update (i.e, the scheme achieves forward secrecy).

This model is tailored towards symmetric UPKE, but can be adapted to also model an adversary targeting an asymmetric UPKE. For an asymmetric scheme, we need to change how updates are considered, as there are now 'public' and 'private' parts to the update. This can be done by having the $\mathsf{FreshUpdate}$ oracle return the public part of the update.

4 Assessing Isogeny-Based UPKE Constructions

In order to construct a UPKE scheme, an *UpdatePrivate* operation should re-randomize the secret key. By applying updates from *GenerateUpdate*, the distribution of the new secret key should be independent from that of the previous one. This is what can ensure that the scheme satisfies forward-secrecy and post-compromise security. Achieving such a property is difficult with lattice systems. The obvious thing to do with a lattice-based system is to add together two (ring)-LWE samples to update a public key. However the corresponding secret key will then be the sum of two (ring)-LWE secrets. The distribution of the resulting secret will be dependent on the previous secret, making it difficult to argue for the security of such a system.

For this reason we assess the extent to which two isogeny-based public-key cryptosystems—SIDH and CSIDH—can support the construction of a updatable public-key encryption scheme. To begin, we review the required algebraic structure to perform both Asymmetric and Symmetric UPKE operations, and assess the capability to perform these operations using SIDH and CSIDH in comparison to two algebraic structures used to instantiate classically secure "Diffie-Hellman-like" schemes. We then discuss the barriers in constructing asymmetric UPKE from SIDH and CSIDH. In Section 5 we discuss the gaps in constructing a symmetric UPKE scheme from SIDH, and in Section 6 we present a Symmetric UPKE scheme from CSIDH.

We present a high-level overview of our analysis in Table 1.

4.1 Comparison of Algebraic Structure

We begin by assessing the extent to which SIDH and CSIDH can support performing Asymmetric or Symmetric UPKE operations. We compare these results to two classically secure algebraic structures over which Diffie-Hellman like operations are commonly performed; the multiplicative group of integers modulo some prime $\mathbb{Z}_p$ and elliptic curves over some finite field $E(\mathbb{F})$.

Required Structure for Asymmetric and Symmetric UPKE

Symmetric UPKE. In the setting for Symmetric UPKE, the same update value is applied to both the encryption and decryption keys, as further described in Section 3.1. Note that this update value is assumed to be efficiently and uniformly sampled to prevent information leakage. Symmetric UPKE relies upon similar group structure requirements as plain Diffie-Hellman key exchange, as while Symmetric UPKE requires some form of commutativity, a group action is not strictly necessary.

Asymmetric UPKE. In the setting for Asymmetric UPKE, performing updates requires a ring structure, as a group action must be defined. Let μ denote the secret value used to update the decryption key and g^μ be the public value used to update the encryption key. In a multiplicative group, the update operations to the encryption key can be thought of as performing $g^a \cdot g^\mu = g^{a+\mu}$. Updating the decryption key requires performing $a + \mu$. Similarly to Symmetric UPKE, Asymmetric UPKE assumes that the secret update value μ is efficiently and uniformly sampled.

Classical Algebraic Structures

Multiplicative Groups. Let $\mathbb{Z}_p$ be the group of integers modulo some prime $p \in \mathbb{Z}$, and generated by some generator $g \in \mathbb{Z}_p$. Here, $\mathbb{Z}_p$ supports the group operation of multiplying two group elements $A \cdot B$, as well as exponentiation of group elements by some scalar value. The group operation in $\mathbb{Z}_p$ is commutative.

Elliptic Curves over Finite Fields. Let $E(\mathbb{F})$ be an elliptic curve over some finite field, generated by a point $G \in E/\mathbb{F}$. Such finite fields can be of prime order given some prime modulus p or of a non-prime order but generate a subfield of prime order over which operations are performed, such as in the case of Curve25519 [4]. Here, $E(\mathbb{F})$ supports one group operation, that of point addition, where a group element is repeatedly added to itself, and point multiplication is supported by performing repeated addition operations.

Isogeny-Based Algebraic Structures

SIDH. Unlike groups such as $\mathbb{Z}_p$ or elliptic curves over some finite field, the endomorphism ring for supersingular curves over $\mathbb{F}_{p^2}$ is neither commutative nor defines a group action at all. To perform SIDH, participants require agreeing upon additional "auxiliary points" which ensure both parties arrive at the same shared secret after completing the protocol. This structure defined by SIDH can be thought of as a *semigroup*; although note that since any function can be considered a semigroup, this classification is not particularly informative.

		Asymmetric		Symmetric	
		Update Value	Update Operation	Update Value	Update Operation
	Secret	$b \xleftarrow{\$} \mathbb{Z}_p$	$a+b$	$b \xleftarrow{\$} \mathbb{Z}_p$	$a \cdot b$
$\mathbb{Z}_p$	Public	g^b	$(g^a \cdot g^b) = g^{a+b}$	b	g^{ab}
	Secret	$b \xleftarrow{\$} \mathbb{F}$	$a+b$	$b \xleftarrow{\$} \mathbb{F}$	$a \cdot b$
$E(\mathbb{F})$	Public	bG	$aG + bG = (a+b)G$	b	$a \cdot (b \cdot G)$
	Secret	X	X	$\phi_U : E_0 \to E_U$	$KLPT(\phi_A, \phi_u) \to \phi_{AU}$
SIDH	Public	X	X	$\phi_U : E_0 \to E_U$	$\phi_U(E_A)$
	Secret	X	X	$b \xleftarrow{\$} \mathbb{Z}_N$	$a + b \pmod{N}$
CSIDH	Public	X	X	b	$b \star E_A$

Table 1. Updatable Symmetric and Asymmetric UPKE Instantiated Across Isogeny-Based Cryptosystems and Two Classical Constructions for Comparison. "X" denotes operations which cannot be performed in that construction. "Secret" denotes the secret key operations; "Public" for public key operations. For CSIDH, the $\star$ operator denotes the group action defined in CSI-FiSh, which takes an element b of $\mathbb{Z}_N$ and interprets it as an ideal before applying it to the elliptic curve (further details in Section 2.2).

CSIDH. Couveignes [13] first defined the concept of a *hard homogeneous space* as a finite commutative group G acting over some set X, a generalization of the structure for a group suitable to performing Diffie-Hellman-like operations. Similarly to SIDH, CSIDH defines a finite commutative group consisting of the ideal-class group $cl(\mathcal{O})$ acting over the set of supersingular curves through the application of isogenies. However, importantly, the graph of supersingular curves in CSIDH is *restricted* to the field $\mathbb{F}_p$, thereby ensuring that $cl(\mathcal{O})$ can act freely and transitively over the group. As $cl(\mathcal{O})$ is commutative, CSIDH can support commutativity of operations over this set of supersingular curves via the group action.

As discussed in Section 2.2, group elements are represented by a vector $\boldsymbol{e} \in \mathbb{Z}^d$, representing the ideal $\prod_{i=1}^{d} \ell_i^{e_i}$. This representation is used because in general applying the group action is computationally difficult. To get around this, CSIDH uses a canonical set of generators $\mathfrak{g}_1, \ldots, \mathfrak{g}_d$ and works be only applying small powers of these generators to the starting curve E. Two major limitations of this representation are that it does not allow for uniform sampling over the class group, and the representation of group elements is not necessarily unique.

Beullens et al. [5] solved this problem, at least for the CSIDH-512 parameter set by explicitly computing the structure of the class group. For this parameter set, they found an N such that $cl(\mathcal{O}) \cong \mathbb{Z}_N$ and described a method to swap between the representation of group elements as a member of $\mathbb{Z}_N$ and of $\mathbb{Z}^d$. This allows for both uniform sampling and unique representation.

4.2 Assessing Isogeny-Based Asymmetric UPKE

We now determine the extent to which SIDH and CISDH can be used to instantiate Asymmetric UPKE schemes as defined in Section 3.1.

SIDH. An Asymmetric UPKE scheme must be able to "combine" public values (i.e, group elements) via some group operation, as described in Section 4.1. Further, the construction should support a second, distinct operation for "combining" private and public values.

As described in Section 4.1, SIDH supports only one operation—applying isogenies via Vélu's formula—over elements in the set of supersingular curves over $\mathbb{F}_{p^2}$. Hence, while SIDH can "combine" public and private values by some (non-group) operation, SIDH does not support the other requirement of Asymmetric UPKE of "combining" two group elements. More clearly, it is difficult to imagine how to apply one public value in SIDH—a supersingular curve—to another.

As such, SIDH *cannot* be used as an underlying public-key encryption primitive to construct an Asymmetric UPKE scheme.

CSIDH. Similarly to SIDH, CSIDH cannot be used for an Asymmetric UPKE construction. The private values in CSIDH form a commutative group, so one might hope for more algebraic structure to be useful. But for asymmetric UPKE we need some sort of structure in the public domain. Public values in CSIDH are still elliptic curves, and without an operation that can be applied between elliptic curves or a new way to separate private and public values, Asymmetric UPKE is also impossible to build with straightforward CSIDH.

5 Symmetric UPKE via SIDH

We now present a series of steps towards a SIDH-based Symmetric UPKE construction. While our end result does define a complete construction, future mathematical improvements must be made before it can be instantiated in a practical setting. We conclude by describing these improvements and their applications to related cryptographic schemes.

SIDH-based UPKE, first attempt. A first step towards a Symmetric UPKE scheme using SIDH has a similar form to plain public-key encryption via SIDH defined by Jao and De Feo [21]. However, here, the update step is performed using a SIDH key exchange operation, where the resulting curve from the exchange becomes the updated public key as opposed to the negotiated shared secret.

Alice performs key generation in exactly the same way as in plain SIDH, by first sampling a scalar $n_A \xleftarrow{\$} \mathbb{Z}_{2^{e_1}}$ which then defines her secret encryption key sk_A is the isogeny $\phi_A : E_0 \to E_A$, such that $E_A = E_0/\langle [1]P_A + [n_A]Q_A \rangle$. Alice then generates her public encryption key pk_A is the curve with auxiliary points $(E_A, \phi_A(P_B), \phi_A(Q_B))$. exactly the same way as in plain SIDH

Let's say now that Bob wishes to generate an update for Alice's keypair. To do so, he simply performs an SIDH KeyGen operation in exactly the same manner as Alice. He samples a secret update value $n_\mu \xleftarrow{\$} \mathbb{Z}_{3^{e_2}}$ which he uses to define a secret isogeny $\mu : E_0 \rightarrow E_\mu$. Bob then applies μ to derive Alice's new public key $E_{A\mu}$ by performing a SIDH key exchange operation, and then transmits μ to Alice (using an encrypted channel). Alice applies μ to her secret term similarly by performing a plain SIDH key exchange operation.

Differently to plain SIDH key exchange, Alice's updated public key now becomes the resulting curve $E_{A\mu}$, along with updated auxiliary points.

$$(E_{A\mu}, \phi_{A\mu}(P_B), \phi_{A\mu}(Q_B))$$

Alice's updated secret key sk_A correspondingly becomes the updated isogeny $\phi_{A\mu} : E_0 \rightarrow E_{A\mu}$, such that

$$\phi_{A\mu} = \langle \phi_\mu(P_A) + [n_A]\phi_\mu(Q_A) \rangle.$$

Problem 1: The naive scheme does not preserve forward secrecy or post-compromise security. While the naive scheme does enable performing updates to SIDH public and private keys via a secret update value, this approach does *not* fulfill the notions of forward secrecy or post-compromise security as required for UPKE schemes. We now describe how the scheme fails to be IND-CPA-U secure.

Recall that in the IND-CPA-U game described in Section 3.2, the adversary is allowed to perform the GiveUpdate query with update values of the adversary's own choosing, and can learn any keypair by performing Corrupt on some index j denoting the secret key of interest. The adversary wins the game if they have a non-negligible chance at guessing the contents of some ciphertext that has been encrypted after some series of τ update, for which the adversary cannot derive the corresponding sk_τ secret key.

In this naive construction, after applying the series of τ update values, the challenger's secret key sk_A will have the following form:

$$\phi_{(A1,\ldots,\tau)} = \langle \phi_1(\ldots(\phi_\tau(P_A))) + [n_A]\phi_1(\ldots(\phi_\tau(Q_A))) \rangle$$

It is easy to see that even after applying τ updates, the n_A term remains static. Consequently, the adversary can use their knowledge of sk_A and the update values after applying the τ updates to derive $\phi_{(A1,\ldots,\tau)}$, by solving a system of linear equations. Even if $\phi_{(A1,\ldots,\tau+1)}$ were instead represented as a group element $P_{\tau+1}$ that generates the kernel of $\phi_{(A1,\ldots,\tau+1)}$ (to avoid persisting the static n_A value), the adversary could still learn $\phi_{(A1,\ldots,\tau+1)}$ simply by solving the discrete logarithm problem for the generator point $Q_{\tau+1} = [n_A]\phi_1(\ldots(\phi_\tau + 1(Q_A)))$. While solving the discrete logarithm is hard in a classical setting with points defined over some ordinary curves, this problem is easy in a post-quantum setting and also in a classical setting over supersingular curves.

Proposed Fix 1: Use KLPT to output a new independent secret, ensuring forward secrecy and post-compromise security after performing an update. As described in Section 2, the KLPT algorithm finds equivalent ideals $J \sim I$ for a specified norm. Such an algorithm can be used to efficiently find an isogeny $\eta : E_0 \to E_2$, given the composition of two isogenies $\phi \circ \psi$, where $\phi : E_0 \to E_1$ and $\psi : E_1 \to E_2$, as enabled by the Deuring correspondence. Importantly, η is *independent* of ϕ and ψ.

Using KLPT to apply an update value to the keyholder's secret would ensure forward secrecy against an adversary that could learn the secret before the update value has been applied. In short, applying an update value would allow the protocol to be self-healing, as the keyholder could "ratchet forward" to a new independent secret that the adversary could not derive without also gaining knowledge of the corresponding update value.

Problem 2: The degree of the isogeny output by KLPT or SQI-Sign is larger than p. As described in Section 2.1, SIDH represents secret keys as an isogeny of degree either 2^e or 3^f, such that $p = 2^3 \cdot 3^f - 1$. To efficiently represent and apply these isogenies using Vélu's formula, SIDH represents each isogeny as a series of degree-two or degree-three isogenies whose composition represents the desired larger-degree isogeny. In other words, SIDH defines isogenies with a kernel that is the composition of small-degree generator points, as follows:

$$\phi : E_0 \to E_n = \langle P \rangle = \{\infty, 2P, 3P, \ldots, (2^3 - 1)P\}$$

Because elements in the endomorphism ring of curves over $\mathbb{F}_{p^2}$ do not commute, both parties publish auxiliary points (P_A, Q_A) and (P_B, Q_B) of order 2^3 and 3^f, respectively, which are then "pushed through" the other party's secret isogeny as part of the key exchange protocol. Doing so ensures that both parties derive the same curve up to isomorphism.

Let's examine how applying an update value μ to Alice's secret ϕ_A via KLPT behaves in the context of SIDH key exchange operations. As described in Section 2, the output of the KLPT algorithm defined by Kohel, Lauter, Petit, and Tignol [24] when provided as input the isogenies $\phi_A : E_0 \to E_A$ and $\mu : E_0 \to E_\mu$ is a new isogeny $\phi_{A\mu} : E_0 \to E_{A\mu}$ of approximately p^2.

Unfortunately, several issues immediately present themselves with this construction. Because the output isogeny $\phi_{A\mu}$ is of degree larger than p, Alice's original auxiliary points (P_A, Q_A) are no longer usable as the order of these points is $2^e \leq p$, and applying $\phi_{A\mu}$ sends these points to infinity.

Even the improvement on the original KLPT algorithm presented by De Feo, Kohel, Leroux, and Petit as part of the SQI-Sign signature scheme [14] resulting in a smaller output isogeny with a deterministic degree and uniformly-random path is insufficient; the output isogeny is still larger than p and thus the same problem remains.

Proposed Fix 2: Represent secret isogenies as a composition of several isogenies. One option to address the issue of isogenies of degree larger than

p is to represent each participant's secret as a ordered set of several isogenies whose composition equals the updated secret $\phi_{A\mu}$ output from KLPT. Here, each isogeny in this set can be of a fixed degree of either 2^e or 3^f, except for the variable-length remainder.

Correspondingly, each participant's public key must be represented as a set of public keys representing each "chunk" in the participant's secret, of the form $(E_0, \ldots, E_k, E_\ell, \ldots, E_{A\mu})$.

Problem 3: Only the owner of the secret key can update the public key. Unfortunately, the proposed approach to represent public keys as a series of curves does not fulfill the requirements for asynchronicity for UPKE constructions; namely that *anyone* other than the owner of the secret key should be able to perform non-interactive updates to the public key.

This is because KLPT outputs isogenies that are independent of the input terms. After Alice applies the update value μ, Alice's new public key will be a series of curves $(E_0, \ldots, E_k, E_\ell \ldots, E_{A\mu})$ corresponding to each "chunk" in Alice's updated secret. While external parties could non-interactively derive the starting and ending curves $E_0, E_{A\mu}$ in this series, they will not be able to derive intermediate curves E_k, E_ℓ, as these curves represent points in the updated walk from $E_0, E_{A\mu}$ that is determined by the output from KLPT. As such, this proposed fix does not satisfy the requirement for asynchronous updates, or that *anyone* can perform *UpdatePublic* with knowledge of only the public key.

Summary. In theory, a SIDH-based Symmetric UPKE construction is possible by using KLPT to perform updates to decryption keys, thereby ensuring forward-secrecy and post-compromise security as is required in practice. However, in practice, improved KLPT algorithms are required to output isogenies whose degree is smaller than the degree of each party's auxiliary points. Without such improvements, such schemes either lack forward secrecy or cannot fulfill certain properties of UPKE designs, namely that of asynchronicity.

We note that improvements in the efficiency of KLPT algorithms will benefit not only our construction, but any construction that similarly requires commutative actions over elements in the public domain.

6 Symmetric UPKE Construction via CSIDH

In the previous section, we saw that the main difficulty in constructing a UPKE scheme from SIDH was in updating the secret key. In order to update the secret key in a way that offers both forward secrecy and post-compromise security, we need a mechanism that completely incorporates the update value into the secret key. This mechanism should result in an entirely new secret key from which no information about the previous secret key can be gained. Furthermore, this new secret key should lie on the same domain as the previous one.

CSIDH UPKE, first attempt. CSIDH admits operations that are much closer to those used in the classical construction from Alwen et al. Recall that secret keys in CSIDH are represented by a vector of ℓ integers. For efficiency reasons, the integers are usually chosen to be within a bound B, for example, $B = 5$ so that all entries are between -5 and 5. Then the group element $[e_1, e_2, \ldots, e_\ell]$ represents the group element

$$\mathfrak{g}_1^{e_1} \mathfrak{g}_2^{e_2} \cdots \mathfrak{g}_\ell^{e_\ell},$$

for a set of canonical generators $\{\mathfrak{g}_i\}$. Since the group is commutative, we have that if g is represented by $[e_1, \ldots, e_\ell]$ and h is represented by $[f_1, \ldots, f_\ell]$ then $g \cdot h$ can be represented by $[e_1 + f_1, \ldots, e_\ell + f_\ell]$. A basic design for a symmetric UPKE scheme would then be for the update value to be a random group element, to update the public key by applying the group action, and to update the secret key by adding the group elements together.

Problem: Secret keys leak information about keys before an update. Unfortunately, this simple design is not secure. If each entry for the update value is drawn uniformly from $-B$ to B, then the distribution of each entry of the new public key is centered at the old public key. This leaks a certain amount of information about the old secret key. For example, if only one update has occurred, if an entry is $2B$ then the adversary immediately knows that the corresponding entry before the update must have been B.

One fix may be to increase the bound B in an attempt to show that leaking the secret key between certain updates still doesn't reveal enough of the secret key to allow a break. Such an analysis must be done carefully, but reveals another fundamental problem. As more updates occur, the size of each entry in the vector is likely to grow. The efficiency of CSIDH is directly dependent on the ℓ_1-norm of this vector, and so allowing it to grow with updates will result in a slower and slower decryption process, eventually becoming unacceptable.

Note that this is almost exactly the same problem that a first attempt at a lattice-based scheme would run into. If one were to define a scheme based on the LWE problem, then updates could be generated by sampling an LWE secret. The secret key would then be updated by adding the update value to the old secret. But as described for CSIDH, this will cause the error term to grow over time, eventually causing the system to fail. Furthermore, because errors are not chosen uniformly, the distribution of a secret will always be dependent on the previous secret, meaning some information about previous keys is leaked in the event of a compromise.

One technique to circumvent this problem that has been to employ rejection sampling, as in the signature scheme SeaSign [15]. However, rejection sampling only works when we can reject the group elements that would leak information on the secret key. Since the party selecting the update value is not the owner of the public key, rejection sampling is not an option in our scenario. Instead, we will need the group elements to be represented in a way that has better properties.

Proposed Fix: Use the class group structure. As mentioned in Section 2.2, the signature scheme CSI-FiSh has a different way to represent group elements. They computed the structure of the class group and found that the group (for the CSIDH-512 parameters) is cyclic and found its order, denoted N. To compute the group action (i.e., apply the isogeny to an elliptic curve) they convert the representation as an element of $\mathbb{Z}_N$ to one in $\mathbb{Z}^\ell$ and then apply the action as in CSIDH. Representing group elements as a member of $\mathbb{Z}_N$ gives a unique representation. It is also still very easy to apply the group operation in this representation — it is just addition modulo N.

We can now define our symmetric UPKE scheme based on the CSIDH group action. In a few sentences, the idea is simply to have the secret key and update value both be in $\mathbb{Z}_N$. To update a public key, we apply the group action, and to update the secret key we add modulo N. Because we can sample uniformly over $\mathbb{Z}_N$, we have that the updated secret key leaks no information about the previous secret key, as desired.

We now describe the scheme in full. For simplicity we will rely heavily on group action notation to describe the operations in our UPKE scheme. Let N be the order of the class group $cl(\mathcal{O}) \cong \mathbb{Z}_N$. To apply the group action onto a supersingular elliptic curve E (denoted $g \star E$), we first need to convert the element to a representation in $\mathbb{Z}^\ell$ with a low L_1 norm, and then apply the action as in the original CSIDH paper. The calculation of this value N as well as the methodology to convert the representation was a major contribution of the CSI-FiSh paper [5].

KeyGen(): Sample a uniform $g_{sk} \overset{\$}{\leftarrow} \mathbb{Z}_N$ and set $E_{pk} := g_{sk} \star E_0$. Return $(sk, pk) = (g_{sk}, E_{pk})$.

Encrypt(m, pk): Sample a uniform $g_{enc} \overset{\$}{\leftarrow} \mathbb{Z}_N$ and compute $K \leftarrow KDF(g_{enc} \star pk)$. Send $c = (g_{enc} \star E_0, DEM_K(m))$.

Decrypt(c, sk): Parse c as $(E_{enc}, ctxt)$. Compute $K \leftarrow KDF(sk \star E_{enc})$ and return $DEM_K^{-1}(ctxt)$.

GenerateUpdate(): Simply sample $\mu \overset{\$}{\leftarrow} \mathbb{Z}_N$.

UpdatePrivate(sk, μ): Return $sk' = g_{sk} + \mu \pmod N$.

UpdatePublic(pk, μ): Return $pk' = \mu \star E_{pk}$.

6.1 Proof

Here we show that our construction attains IND-CPA-U security, as described in Section 3.2. We will show a reduction from an adversary capable of winning the IND-CPA-U game to one that can win a plain IND-CPA game. By a plain IND-CPA game, we mean a game in which no calls to the *GenerateUpdate*, GiveUpdate, or Corrupt oracles are made.

Theorem 1. *Let $\mathcal{A}$ be an adversary capable of winning the IND-CPA-U game with advantage p that makes q_{gen} queries to the* FreshUpdate *oracle. We will construct an adversary capable of winning an IND-CPA game in time approximately equal to the running time of $\mathcal{A}$ with advantage $p/(q_{gen} + 1)$.*

Proof. As we are showing a reduction to a plain IND-CPA game, we will start by being given a public key pk^*. To begin, select a uniformly random index $i \xleftarrow{\$} \{0, \ldots, q_{gen}\}$. The idea of the proof is to set the public key after the ith $\mathsf{FreshUpdate}$ query to be pk^*, and hope that the adversary requests the IND-CPA-U challenge to be issued on a public key that occurs before the next $\mathsf{FreshUpdate}$. If we are correct, then the adversary's ability to distinguish which message was encrypted under pk^* (or a related key) will allow us to win the IND-CPA game.

At the start of the game, if $i = 0$ then we set $pk_0 \to pk^*$. Otherwise, we sample a new uniform pk_0 from $KeyGen$. From here we proceed as normal. If the adversary makes a corruption query, then we provide them with the corresponding private key. When a $\mathsf{GiveUpdate}(\mu)$ query is made, we update the secret and public key and make note of the μ value.

When the ith query to $\mathsf{FreshUpdate}$ is made, we set the resulting public key to pk^*. We carry on, and when the next $\mathsf{FreshUpdate}$ query is made we sample a fresh public key from $KeyGen$. If the adversary ever makes a $\mathsf{Corrupt}$ query on any of the keys between these $\mathsf{FreshUpdate}$ queries, then we abort. We will consider the probability of having to abort occurring momentarily.

Eventually, the adversary requests the IND-CPA-U challenge on a public key with index j. We hope that this index means a key that falls between the ith $\mathsf{FreshUpdate}$ and the $i+1$th call to $\mathsf{FreshUpdate}$. When this happens, the adversary submits m_0, m_1 as part of the challenge.

We then forward m_0, m_1 to receive back an encryption of m_b, consisting of $C = g \star E_0$ for a random g, as well as $DEM_K(m_b)$. Let $\mu_1, \mu_2, ..., \mu_k$ be k queries to $\mathsf{GiveUpdate}$ after the ith $\mathsf{FreshUpdate}$ query. We provide the adversary with $(-\mu_1 - \mu_2 - \cdots - \mu_k) \star C$ and $DEM_K(m_b)$.

Note that $K = KDF(g \star pk^*) = KDF((-\mu_1 - \cdots - \mu_k) \star g \star (\mu_1 + \cdots + \mu_k) \star pk^*)$, which means that the message is encrypted under the correct key. So, when the adversary submits a guess for b, we can guess the same value, and if the adversary is correct, so are we.

When we set the public key to pk^* after the ith call to $\mathsf{FreshUpdate}$, the adversary cannot notice that we have not genuinely updated the public key, unless they issue a $\mathsf{Corrupt}$ query. If such a $\mathsf{Corrupt}$ query is issued, we must abort. However, note that if our guess is correct, and a the IND-CPA query is requested in this segment of public keys, then no $\mathsf{Corrupt}$ query will be issued, or else the adversary's advantage is 0.

Because updates are sampled uniformly over $\mathbb{Z}_p$, the resulting public key is uniformly random over the public key space (this follows from the fact that the group action is regular). So after a $\mathsf{FreshUpdate}$ has occurred, the adversary has no information on the distribution of the secret key, and we can thus replace the public key with the challenge public key pk^*. The adversary has no advantage in distinguishing that we have done this. As a result, we have a $1/(1 + q_{gen})$ chance of correctly guessing where the challenge will be requested. If we are correct, the adversary does not change their behavior at all, as they have no advantage in distinguishing that we are not managing the game honestly. This means the chance that we abort is exactly $q_{gen}/(1+q_{gen})$.

Our advantage in winning the IND-CPA game is thus the adversary's advantage in winning the IND-CPA-U game times the probability we do not abort, which is $p/(1+q_{gen})$, as desired.

We note that the techniques in this proof can also be applied to the classical construction of Alwen et al. [1]. While they couple together the public key update and encryption functions, the same general strategy can be used to show that the stronger IND-CPA-U notion can be satisfied by their construction.

6.2 Implementation

The main challenge in an implementation of the scheme is selecting parameters. Because the scheme requires the structure of the class group to be known, our CSIDh-based scheme can only be instantiated if such a computation has been performed. This limits a current implementation to the CSIDH-512 parameter set, which aims for 64 bits of post-quantum security.[3] Computing the structure of the class group is a sub-exponential computation, and thus is unlikely to be performed for much larger parameter sets. However, with a quantum computer, the computation becomes entirely feasible. As such, the scheme may not be able to be instantiated until it is most needed (when a scalable quantum computer is finally a reality). This limitation presents a problem for transitional security, but remains an interesting possibility for the future.

Other than computing the class group, the main challenge in an implementation is in computing the group action. To compute the group action, the element of $\mathbb{Z}_N$ is converted to a vector in $\mathbb{Z}^\ell$, which represents the group element $\prod_{i=1}^{\ell} \mathfrak{g}_i^{e_i}$ for a vector $\boldsymbol{e}$ and set of generators $\{\mathfrak{g}_i\}_i$. This vector is then applied to the elliptic curve as is done in CSIDH.

Thus the additional complication over any other CSIDH implementation is in converting the element of $\mathbb{Z}_N$ to a vector of integers. This process is described in the CSI-FiSh paper, and the authors have provided code to do this (for the CSIDH-512 parameter set). The authors of CSI-FiSh found that the process of converting to a vector only makes a key negotiation 15% slower. Using their implementation of CSI-FiSh, we have a proof of concept script that illustrates the process of updating the secret and public keys. Our script is available at `https://github.com/tedeaton/CSIDH-UPKE`.

7 Conclusion

In this work, we perform the first assessment of the post-quantum readiness for *updatable* public-key encryption schemes by determining the extent to which two

[3] It has been contested in [28] that this parameter set may not achieve 64 bits of security. This has been disputed by others, but more recent analysis puts CSIDH-4096 at NIST level 1 security [10].

isogeny-based cryptosystems can be used to instantiate Symmetric and Asymmetric UPKE constructions. Because neither SIDH nor CSIDH define a group action among elements in the public domain, neither can support update operations among public update values and public encryption keys as is required by Asymmetric UPKE. However, both SIDH and CSIDH can be used to instantiate Symmetric UPKE, which only requires operations between elements in the secret domain, and applying elements in the secret domain to public values. The SIDH-based Symmetric UPKE construction, while possible in theory, requires mathematical improvements for a construction in practice. However, our CSIDH-based construction can be instantiated today using CSIDH-512 as the parameter set. We highlight several open problems that would improve our constructions, including the need for a more efficient KLPT algorithm to generate smaller-degree isogenies and stronger CSIDH parameter sets. Beyond just Symmetric UPKE, such improvements will benefit any protocol that requires ongoing and asynchronous randomization of secret terms.

8 Acknowledgments

We thank Martin Albrecht, Alex Davidson, and Fernando Virdia for discussion of lattice-based UPKE operations and forward secrecy in MLS in general. We thank Douglas Stebila for his review of our proof and suggestions on modeling an adaptive adversary that can both select update values and compromise a victim's local state. We thank Richard Barnes for his help in understanding the forward secrecy limitations of TreeKEM and details of the MLS protocol in general.

References

1. Joël Alwen, Sandro Coretti, Yevgeniy Dodis, and Yiannis Tselekounis. Security analysis and improvements for the IETF MLS standard for group messaging. *IACR Cryptology ePrint Archive*, 2019:1189, 2019.
2. Fatih Balli, Paul Rösler, and Serge Vaudenay. Determining the Core Primitive for Optimally Secure Ratcheting. *IACR Cryptol. ePrint Arch.*, 2020:148, 2020.
3. R. Barnes, B. Beurdouche, J. Millican, E. Omara, K. Cohn-Gordon, and R. Robert. The Message Layer Security (MLS) Protocol. `https://tools.ietf.org/pdf/draft-ietf-mls-protocol-09.pdf`, March 2020.
4. Daniel J. Bernstein. Curve25519: New Diffie-Hellman speed records. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *Public Key Cryptography — PKC 2006*, pages 207–228, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.
5. 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*, pages 227–247, Cham, 2019. Springer International Publishing.
6. Karthikeyan Bhargavan, Richard Barnes, and Eric Rescorla. TreeKEM: Asynchronous Decentralized Key Management for Large Dynamic Groups A protocol proposal for Messaging Layer Security (MLS). Research report, Inria Paris, May 2018.

7. Xavier Bonnetain and André Schrottenloher. Quantum security analysis of CSIDH. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020*, pages 493–522, Cham, 2020. Springer International Publishing.
8. Jacqueline Brendel, Marc Fischlin, Felix Günther, Christian Janson, and Douglas Stebila. Towards post-quantum security for Signal's X3DH handshake. In Michael J. Jacobson Jr., Orr Dunkelman, and Colin O'Flynn, editors, *Proc. 27th Conference on Selected Areas in Cryptography (SAC) 2020*, LNCS. Springer, October 2020. To appear. Cryptology ePrint Archive, Report 2019/1356. `http://eprint.iacr.org/2019/1356`.
9. Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. CSIDH: An efficient post-quantum commutative group action. In *Advances in Cryptology – ASIACRYPT 2018 – 24th International Conference, Proceedings, Part III*, volume 11274 of *Lecture Notes in Computer Science*, pages 395–427. Springer, 2018.
10. Jorge Chávez-Saab, Jesús-Javier Chi-Domínguez, Samuel Jaques, and Francisco Rodríguez-Henríquez. The SQALE of CSIDH: Square-root vélu Quantum-resistant isogeny Action with Low Exponents. Cryptology ePrint Archive, Report 2020/1520, 2020. `https://eprint.iacr.org/2020/1520`.
11. K. Cohn-Gordon, C. Cremers, B. Dowling, L. Garratt, and D. Stebila. A formal security analysis of the signal messaging protocol. In *2017 IEEE European Symposium on Security and Privacy (EuroS P)*, pages 451–466, 2017.
12. Jean-Marc Couveignes. Hard homogeneous spaces. Cryptology ePrint Archive, Report 2006/291, 2006. `https://eprint.iacr.org/2006/291`.
13. Jean-Marc Couveignes. Hard homogeneous spaces. Cryptology ePrint Archive, Report 2006/291, 2006. `https://eprint.iacr.org/2006/291`.
14. L. De Feo, D. Kohel, A. Leroux, C. Petit, and B. Wesolowski. Sqisign: Light post-quantum signatures from quaternions and isogenies. Fourteenth Algorithmic Number Theory Symposium, Rump Session, 2020. `https://math.mit.edu/~drew/ANTSXIV/RumpLeroux.pdf`.
15. Luca De Feo and Steven D. Galbraith. SeaSign: Compact isogeny signatures from class group actions. In *Advances in Cryptology – EUROCRYPT 2019 – 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Proceedings, Part III*, volume 11478 of *LNCS*, pages 759–789. Springer, 2019.
16. Craig Gentry. Fully Homomorphic Encryption Using Ideal Lattices. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, STOC '09, page 169–178, New York, NY, USA, 2009. Association for Computing Machinery.
17. Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013*. Springer Berlin Heidelberg, 2013.
18. Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. A modular analysis of the fujisaki-okamoto transformation. In Yael Kalai and Leonid Reyzin, editors, *Theory of Cryptography*, pages 341–371, Cham, 2017. Springer International Publishing.
19. Joseph Jaeger and Igors Stepanovs. Optimal channel security against fine-grained state compromise: The safety of messaging. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018*, pages 33–62, Cham, 2018. Springer International Publishing.
20. David Jao, Reza Azarderakhsh, Matthew Campagna, Craig Costello, Luca De Feo, Basil Hess, Aaron Hutchinson, Amir Jalali, Koray Karabina, Brian Koziel, Brian

LaMacchia, Patrick Longa, Michael Naehrig, Geovandro Pereira, Joost Renes, Vladimir Soukharev, and David Urbanik. Supersingular Isogeny Key Exchange. `https://sike.org/files/SIDH-spec.pdf`, 2019. last accessed 2020-04-20.

21. David Jao and Luca De Feo. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In Bo-Yin Yang, editor, *Post-Quantum Cryptography*, pages 19–34, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.
22. Yao Jiang. The Direction of Updatable Encryption does not Matter Much. *IACR Cryptol. ePrint Arch.*, 2020:622, 2020. To appear in Asiacrypt 2020.
23. Daniel Jost, Ueli Maurer, and Marta Mularczyk. Efficient ratcheting: Almost-optimal guarantees for secure messaging. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019*, pages 159–188, Cham, 2019. Springer International Publishing.
24. David Kohel, Kristin Lauter, Christophe Petit, and Jean-Pierre Tignol. On the quaternion ℓ-isogeny path problem. *LMS Journal of Computation and Mathematics*, 17(A):418–432, 2014.
25. Moxie Marlinspike and Trevor Perrin. The Double Ratchet Algorithm, 2016. `https://signal.org/docs/specifications/doubleratchet/`.
26. Moxie Marlinspike and Trevor Perrin. The X3DH Key Agreement Protocol, 2016. `https://signal.org/docs/specifications/x3dh/`.
27. Robert J McEliece. A public-key cryptosystem based on algebraic. *Coding Thv*, 4244:114–116, 1978.
28. Chris Peikert. He gives C-sieves on the CSIDH. Cryptology ePrint Archive, Report 2019/725, 2019. `https://eprint.iacr.org/2019/725`.
29. Christophe Petit and Spike Smith. An improvement to the quaternion analogue of the L-isogeny path problem. `https://crypto.iacr.org/2018/affevents/mathcrypt/medias/08-50_3.pdf`, 2018. last accessed 2020-10-29.
30. Bertram Poettering and Paul Rösler. Towards bidirectional ratcheted key exchange. In *Annual International Cryptology Conference*, pages 3–32. Springer, 2018.
31. Bertram Poettering and Paul Rösler. Asynchronous ratcheted key exchange. Cryptology ePrint Archive, Report 2018/296, 2018. `https://eprint.iacr.org/2018/296`.
32. Alexander Rostovtsev and Anton Stolbunov. Public-key cryptosystem based on isogenies. Cryptology ePrint Archive, Report 2006/145, 2006. `https://eprint.iacr.org/2006/145`.
33. Joseph Silverman. *The Arithmetic of Elliptic Curves*, volume 106. Springer New York, 01 2009.