CCA Secure Updatable Encryption from Non-Mappable Group Actions

Jonas Meers[1] and Doreen Riepel[2]

[1] Ruhr-University Bochum, Bochum, Germany
[2] University of California San Diego, La Jolla, USA
jonas.meers@rub.de, driepel@ucsd.edu

Abstract. Ciphertext-independent updatable encryption (UE) allows to rotate encryption keys and update ciphertexts via a token without the need to first download the ciphertexts. Although, syntactically, UE is a symmetric-key primitive, ciphertext-independent UE with forward secrecy and post-compromise security is known to imply public-key encryption (Alamati, Montgomery and Patranabis, CRYPTO 2019).

Constructing post-quantum secure UE turns out to be a difficult task. While lattices offer the necessary homomorphic properties, the introduced noise allows only a bounded number of updates. Group actions have become an important alternative, however, their structure is limited. The only known UE scheme by Leroux and Roméas (IACR ePrint 2022/739) uses effective triple orbital group actions which uses additional algebraic structure of CSIDH. Using an ideal cipher, similar to the group-based scheme SHINE (Boyd et al., CRYPTO 2020), requires the group action to be mappable, a property that natural isogeny-based group actions do not satisfy. At the same time, other candidates based on non-commutative group actions suffer from linearity attacks.

For these reasons, we explicitly ask how to construct UE from group actions that are not mappable. As a warm-up, we present BIN-UE which uses a bit-wise approach and is CPA secure based on the well-established assumption of weak pseudorandomness and in the standard model. We then construct the first actively secure UE scheme from post-quantum assumptions. Our scheme COM-UE extends BIN-UE via the Tag-then-Encrypt paradigm. We prove CCA security in the random oracle model based on a stronger computational assumption. We justify the hardness of our new assumption in the algebraic group action model.

Keywords: Updatable encryption, group actions, isogenies, algebraic group action model

1 Introduction

Updatable encryption (UE) allows to update a ciphertext to a new key without first decrypting the ciphertext and then re-encrypting it with the new key. Instead, the owner of the encryption key computes an update *token* which can then be used by a different party to update the ciphertext on behalf of the owner. This is especially useful in the context of cloud storage where a user might want to update the encryption key of some large encrypted file without the need to first download the file and re-encrypt it locally. This and other real-world applications have contributed to the fact that UE is a very active field of research [11, 29, 30, 32, 33, 36, 41]. All these works are concerned with the most general setting, where tokens can be created independently (i. e., they only depend on the old and new key) and allow to update all ciphertexts encrypted under the old key. This will also be our focus.

Security of UE. Since the introduction of UE by Boneh, Lewi, Montgomery and Raghunathan [9], security definitions have evolved. Definitions for standard symmetric encryption (SE) have been adapted to the setting of UE, capturing confidentiality and integrity of messages. In contrast to the standard SE setting, however, updating keys aims to provide stronger security, namely, forward secrecy and post-compromise security. This is captured in security definitions based on *epochs*, where some encryption keys and tokens may be revealed. Even in the presence of adaptive corruptions, we ask for indistinguishability under chosen message attacks and unlinkability of ciphertexts across updates, both of which are captured in the IND-UE-CPA definition of [11]. However, defining trivial attacks has turned out to be a complex and subtle task; various properties of UE schemes, which affect the "inferred" knowledge after a corruption, have been identified and result in slight adaptations [28, 41]. For sake of clarity, we will not elaborate further at this point and explain the necessary details in the main body of the paper. In this work, we also aim for stronger security for UE capturing chosen ciphertext attacks (IND-UE-CCA), where the adversary is given additional access to a decryption oracle.

Constructions of UE. In order for a scheme to allow for rotating keys and updates, proposed schemes make use of public-key primitives with homomorphic properties. Indeed, Alamati, Montgomery and Patranabis [3] show that any ciphertext-independent UE scheme that is forward and post-compromise secure implies public-key encryption. We provide an overview of existing constructions in Table 1. There exist various constructions that rely on (elliptic-curve) groups, e. g., RISE [32], SHINE [11] and the DDH-based instantiation of the Encrypt-and-MAC (E&M) construction in [30]. While one might hope that these schemes can be easily transferred to the group action setting, current candidates for group actions do not allow for this. We will elaborate more on these constructions in Section 1.1.

The emergence of quantum computers poses an undeniable threat to all UE constructions based on prime-order groups. Therefore, the interest in post-quantum secure constructions for UE has grown in recent years. One promising approach constructs UE based on lattices [28, 29, 41]. The security of these schemes relies on the Learning with Error (LWE) assumption and updates are computed using the homomorphic properties of the underlying lattice. More specifically, the UE constructions rely on a key homomorphic PKE. When using a key and message homomorphic PKE, an even stronger security notion that captures uni-directional updates can be achieved, albeit at the price of ciphertexts and keys growing with the number of epochs. For more details, we refer

	Scheme	**Security (IND)**	**Assumption**	**Model**
Groups	RISE [32]	(rand, UE, CPA)	DDH	Standard
	E&M [30]	(det, ENC/UPD, CCA)	DDH	ROM
	SHINE0 [11]	(det, UE, CCA)	DDH	Ideal Cipher
Pairings	NYUAE [30]	(rand, ENC/UPD, RCCA)	SXDH	Standard
	SS23 [48]	(rand, UE, CPA)[+]	SXDH	Standard
Lattices	Jia20 [29]	(rand, UE, CPA)	LWE	Standard
	Nis22 [41]	(rand, UE, CPA)	LWE	Standard
	GP23 [28]	(rand, UE, CPA)	LWE	Standard
Group Actions	GAINE0[*] [33]	(det, UE, CCA)	Wk-PR	Ideal Cipher
	TOGA-UE [33]	(det, UE, CPA)	P-CSSDDH	Standard
	BIN-UE (Sec. 3)	(det, UE, CPA)	Wk-PR	Standard
	COM-UE (Sec. 4)	(det, UE, CCA)	DLAI	ROM + AGAM

[*] no secure instantiation known

[+] satisfies a stronger definition with expiry epochs

Table 1. Overview of existing ciphertext-independent UE constructions and our new constructions BIN-UE and COM-UE. For each scheme, we note whether updates are performed using randomness (rand) or deterministically (det). Most schemes are analyzed using the IND-UE security definition from [11], while others use the weaker IND-ENC and IND-UPD definitions [32]. For a formal comparison of these definitions, we refer the reader to [11]. The definition of RCCA security [30] has been established for constructions with randomized updates.

to Section 1.3. The main drawback of these schemes, however, is the fact that encryption noise increases with each update, resulting in a finite number of updates that the schemes support. Furthermore, these schemes are currently not IND-UE-CCA secure.

1.1 The Difficulty of Constructing UE from Group Actions

Cryptographic group actions offer a post-quantum secure alternative to prime-order groups and are therefore another natural candidate for constructing post-quantum secure UE. Let $(\mathcal{G}, \cdot)$ be a group and $\mathcal{X}$ some set. A group action $\star : \mathcal{G} \times \mathcal{X} \to \mathcal{X}$ defines a map which is compatible with the group operation, i. e., $e \star x = x$ and $(g \cdot h) \star x = g \star (h \star x)$ for all $x \in \mathcal{X}$, $g, h \in \mathcal{G}$, where $e \in \mathcal{G}$ is the neutral element. Note in particular, that we do not assume any structure on $\mathcal{X}$ which will be one limiting factor in constructing efficient UE schemes. One popular instantiation of a cryptographic group action is CSIDH [15] which is based on isogenies between supersingular elliptic curves. Besides close variants like CSURF [12] and SCALLOP [22], CSIDH is the only known commutative group action that is believed to offer post-quantum security.

Using ElGamal Encryption. The group-based UE scheme RISE [32] uses the homomorphic properties of the ElGamal encryption scheme. Recall that the message in (standard) ElGamal encryption is represented by a group element which is then multiplied with the public key raised to an ephemeral secret. Due to the limited structure of group actions, however, we cannot adapt this approach.

For PKE, this issue can be resolved by applying a hash function first and then encrypt the message. For UE, this will destroy all the properties required for updatability. An alternative approach is that of SiGamal [37], which we will further discuss below.

Using Ideal Ciphers. We now turn to SHINE which is the group-based UE scheme from [11]. SHINE makes use of an ideal permutation to map the message (and a nonce) to a group element. Encryption is then simply exponentiation using the secret key. IND-UE-CPA security assumes DDH and the proof is carried out in the ideal cipher model which dates back to Shannon [47]. The advantage is that only exponentiation is used, thus, there is a straightforward generalization to the group action setting. The resulting scheme GAINE has been proposed by Leroux and Roméas [33] and they prove IND-UE-CPA security in the ideal cipher model, assuming weak pseudorandomness (Wk-PR) of the group action. The scheme can further be made IND-UE-CCA secure by adding a zero-padding to the message (and nonce), resulting in schemes SHINE0 and GAINE0. However, the way the ideal cipher is used adds an additional requirement, namely, the group action must be *mappable*. In short, a mappable group action comes equipped with an efficiently computable bijection $\pi : \{0,1\}^N \to \mathcal{X}$ which allows mapping a message to a set element and vice versa. It turns out that for many popular group actions like CSIDH, it is notoriously hard to define such a mapping [10, 38]. At the same time, there is currently no other (non-commutative) group action that is both mappable and satisfies Wk-PR. In fact, it was recently shown that many non-commutative group actions are susceptible to linearity attacks, making them unsuitable for UE [20]. This leaves us with no secure candidate instantiation for GAINE.

Triple Orbital Group Actions. For the reasons identified above, the authors in [33] define a new algebraic abstraction called *Triple Orbital Group Action* (TOGA). It combines a mappable group action and a weakly pseudorandom group action into a single structure. The resulting scheme, TOGA-UE, can then be instantiated with isogenies based on ideas developed for the SiGamal encryption scheme [37]. While TOGA-UE seems like a promising approach to avoid the ideal cipher model in the first place, its security relies on the weak-pseudorandomness of the *whole* TOGA. More precisely, for the instantiation given in [33] based on CSIDH this results in an assumption close to the P-CSSDDH assumption first defined in [37, Definition 10]. Unfortunately, P-CSSDDH does not reduce to the standard Wk-PR of CSIDH since in P-CSSDDH additional torsion point information is revealed. Although the revealed information is not sufficient to apply the SIDH attacks [13, 35, 46], it makes the hardness of P-CSSDDH less well understood. Further, apart from CSIDH and its variations there is currently no other (post-quantum secure) instantiation of TOGA as its definition is rather dedicated to SiGamal. Lastly, the authors point out that their construction is malleable; thus, IND-UE-CCA security of TOGA-UE or an adaptation of it seems currently out of reach.

CCA Security via Encrypt-and-MAC. As of now, we do not have any post-quantum secure UE scheme that achieves $\mathsf{IND\text{-}UE\text{-}CCA}$ security. Klooß, Lehmann and Rupp [30] give a DDH-based UE scheme based on the Encrypt-and-MAC ($\mathsf{E\&M}$) paradigm. Their scheme combines the above-mentioned ElGamal encryption scheme with an updatable PRF based on the construction by Naor, Pinkas and Reingold [40]. Ignoring for now that we cannot use the same ElGamal approach as elaborated above, we want to point out that the updatable PRF requires hashing into the group (i. e., for group actions this would be the set), which, although a weaker requirement than being mappable [33], is still an open question for isogeny-based group actions [10]. Given that known lattice-based construction also do not achieve $\mathsf{IND\text{-}UE\text{-}CCA}$ security, we do not have any actively secure UE from post-quantum assumptions.

1.2 Our Contributions

We aim to overcome these difficulties and our goal is to construct $\mathsf{detIND\text{-}UE\text{-}CCA}$ secure UE from any (non-commutative) group action without requiring the group action to be mappable.

Our first construction, which we call $\mathsf{BIN\text{-}UE}$, maps each bit of the message to a set element in a black-box way and then encrypts each bit under a different key. More specifically, for a key $k = (k_1, \cdots, k_n) \in \mathcal{G}^n$ and a message $M = (m_1, \ldots, m_n) \in \{0,1\}^n$, the encryption algorithm computes

$$\mathsf{BIN\text{-}UE.Enc}(k, M; (x_0, x_1)) \coloneqq (k_1 \star x_{m_1}, \ldots, k_n \star x_{m_n}) \ ,$$

where $x_0, x_1 \in \mathcal{X}$ are random set elements used as encryption randomness. Update tokens are of the form $\Delta_i = k_i^{\text{new}} \cdot (k_i^{\text{old}})^{-1}$ and applied to each element of the ciphertext individually. In order to allow for correct decryption, we will have to define an order on x_0 and x_1. We will explain the technicalities in Section 3, where we formally introduce the scheme. Apart from being applicable to non-mappable group actions like CSIDH [15], this approach comes with several additional advantages: First, we prove $\mathsf{IND\text{-}UE\text{-}CPA}$ security assuming that the group action is weakly pseudorandom, which is considered a standard assumption for CSIDH. Second, our proof does not require idealized models, that is, security holds in the *standard model.* This is a direct consequence of not needing a mappable group action. A natural disadvantage of the bit-wise approach is the large key size and overall efficiency of the scheme since both grow linearly in the bit-length of the messages. In group action based cryptography, however, this is a well understood and accepted compromise [1, 2, 6, 8, 23].

We then turn to constructing a scheme that satisfies $\mathsf{IND\text{-}UE\text{-}CCA}$ security, which is the main contribution of this paper. Our second scheme $\mathsf{COM\text{-}UE}$ extends $\mathsf{BIN\text{-}UE}$ via the Tag-then-Encrypt paradigm, where we encrypt messages of the form $(M, T) \in \{0,1\}^{n+t}$ with $\mathsf{BIN\text{-}UE}$. This means a key is now $k \in \mathcal{G}^{n+t}$. More specifically, to encrypt a message $M \in \{0,1\}^n$ with randomness r, we compute the following:

$$\mathsf{COM\text{-}UE.Enc}(k, M; r) \coloneqq \mathsf{BIN\text{-}UE.Enc}(k, M \| \mathsf{H}(M, r); r) \ ,$$

where hash function H is used to compute the tag. The advantage is that updatability works exactly as in $\mathsf{BIN\text{-}UE}$. In Section 4, we further elaborate on how this approach compares to the Encrypt-*and*-MAC construction of [30] and why Tag-*then*-Encrypt seems preferable here. While the composition is not secure in general [5], we prove that $\mathsf{COM\text{-}UE}$ is $\mathsf{IND\text{-}UE\text{-}CCA}$ secure under a new assumption and in the random oracle model. Our assumption $\mathsf{Multi\text{-}St\text{-}UP}$ is a non-standard and stronger variant of weak unpredictability for group actions [2]. In order to justify its hardness, we show that in the Algebraic Group Action Model (AGAM) [24], $\mathsf{Multi\text{-}St\text{-}UP}$ is implied by the Discrete Logarithm Problem with Auxilary Input [4]. Lastly, $\mathsf{COM\text{-}UE}$ has the same set of disadvantages as $\mathsf{BIN\text{-}UE}$.

To summarize, we give the first UE scheme from group actions that satisfies $\mathsf{IND\text{-}UE\text{-}CPA}$ security from standard assumptions. Further, we get the first UE scheme that satisfies $\mathsf{IND\text{-}UE\text{-}CCA}$ security from post-quantum, however non-standard, assumptions.

1.3 Further Related Work

Ciphertext-*dependent* UE, as considered in [7, 9, 17, 18, 26], generates tokens for individual ciphertexts. While, in general, this allows for more efficient constructions, it requires the owner of the data to download (a part of) the ciphertext in order to generate the update token.

The *direction of updates* plays an important role in determining the impact of key and token corruptions. While our schemes have bi-directional updates, where the old key can be derived from the new key (and token) and vice versa, recent works have studied uni- and no-directional updates [28, 29, 41] and their relations. While these properties are desirable in terms of security guarantees, schemes are also notoriously harder to construct. This was studied in further detail by [28] who aim at constructing UE from PKE generically. They show how to construct bi-directional UE scheme from a key homomorphic PKE scheme. Then, to construct UE with no-directional updates[1], they require PKE with key and message homomorphism. In this construction, ciphertext and key sizes grow linearly in the number of epochs. A similar construction was given in [36]. Unfortunately, it is not clear how to construct a key and message homomorphic PKE from group actions since it would require to combine set elements. Further note that the only known construction in this setting which does not have growing ciphertexts and keys relies on the strong assumption of indistinguishability obfuscation [41].

A *constructive* and *composable* view on UE was taken in [27] and [34], respectively. Further, Slamanig and Striecks [48] study a stronger security definition with more *fine-grained forward secrecy* via expiry epochs.

Updatable MACs [19] are a useful tool to ensure ciphertext integrity in UE. Unfortunately, known group-based constructions require hashing into the group

[1] No-directional and backward-leak uni-directional updates are shown to be equivalent [28]. They provide strictly stronger security than bi-directional updates which have been shown to be equivalent to forward-leak unidirectional updates [29].

and thus, do not directly transfer to the group action setting. Extending UE with signed ciphertext was recently considered in [45].

Proxy re-encryption is a public-key primitive which enables to re-encrypt a ciphertext, encrypted to one party's public key, such that it can be decrypted by another party's secret key. Constructions and security models have been considered in [16,21,31,36,43,44] and also compared to the UE setting [21,31].

Updatable *public-key* encryption has been constructed from isogenies [25], however, the goal here is to update public and secret keys asynchronously (and not to update ciphertexts) to achieve forward secrecy in messaging applications.

2 Preliminaries

Notation. We denote by $\prec_{\mathsf{lex}}$ the lexicographical order. For integers m, n where $m < n$, $[m, n]$ denotes the set $\{m, m + 1, ..., n\}$. For $m = 1$, we simply write $[n]$. For a set S, $s \xleftarrow{\$} S$ denotes that s is sampled uniformly and independently at random from S. $y \leftarrow \mathcal{A}(x_1, x_2, ...)$ denotes that on input $x_1, x_2, ...$ the probabilistic algorithm $\mathcal{A}$ returns y. $\mathcal{A}^{\mathrm{O}}$ denotes that algorithm $\mathcal{A}$ has access to oracle O. An adversary is a probabilistic algorithm. We will use game-based security notions, where $\Pr[\mathrm{G}(\mathcal{A}) \Rightarrow 1]$ denotes the probability that the final output of game G running adversary $\mathcal{A}$ is 1. The notation $[\![\mathsf{X}]\!]$ denotes a boolean test which returns 1 if the statement X is true and 0 otherwise.

2.1 Group Actions

We recall the definition of (restricted) effective group actions from [2], which provides an abstract framework to build cryptographic primitives relying on isogeny-based assumptions such as CSIDH.

Definition 1 (Group Action). *Let $(\mathcal{G}, \cdot)$ be a group with identity element $e \in \mathcal{G}$, and $\mathcal{X}$ a set. A map*

$$\star : \mathcal{G} \times \mathcal{X} \to \mathcal{X}$$

is a group action if it satisfies the following properties:

1. *Identity: $e \star x = x$ for all $x \in \mathcal{X}$.*
2. *Compatibility: $(g \cdot h) \star x = g \star (h \star x)$ for all $g, h \in \mathcal{G}$ and $x \in \mathcal{X}$.*

Remark 1. Throughout this paper, we assume the group action to be commutative and regular. The latter means that for any $x, y \in \mathcal{X}$ there exists precisely one $g \in \mathcal{G}$ satisfying $y = g \star x$.

Definition 2 (Effective Group Action). *Let $(\mathcal{G}, \mathcal{X}, \star)$ be a group action satisfying the following properties:*

1. *$\mathcal{G}$ is finite and there exist efficient (PPT) algorithms for membership testing, equality testing, (random) sampling, group operation and inversion.*

```
**Games** $\mathrm{G}_{\mathsf{EGA}}^{\mathsf{Wk\text{-}PR\text{-}}b}(\mathcal{A})$
$g \xleftarrow{\$} \mathcal{G}$
$b' \leftarrow \mathcal{A}^{\textsc{Sample}}$
**return** $b'$

**Oracle** $\textsc{Sample}$
$x, y \xleftarrow{\$} \mathcal{X}$
**if** $b = 0$: **return** $(x, y)$
**if** $b = 1$: **return** $(x, g \star x)$
```

Fig. 1. Games $\mathrm{G}_{\mathsf{EGA}}^{\mathsf{Wk\text{-}PR\text{-}}b}$, where $b \in \{0, 1\}$, capturing weak pseudorandomness of EGA.

2. *The set $\mathcal{X}$ is finite and there exist efficient algorithms for membership testing and to compute a unique representation.*
3. *There exists a distinguished element $\tilde{x} \in \mathcal{X}$ with known representation.*
4. *There exists an efficient algorithm to evaluate the group action, i. e. to compute $g \star x$ given g and x.*

Then we call $\tilde{x} \in \mathcal{X}$ the origin and $(\mathcal{G}, \mathcal{X}, \star, \tilde{x})$ an effective group action (EGA).

Remark 2 (Mappable EGA). Recalling the definition of [33], a mappable effective group action comes with a bijection $\pi : \{0, 1\}^N \rightarrow \mathcal{X}$ such that $N = \log|\mathcal{X}|$. Since it is unclear whether such a mapping exists for isogeny-based group actions [10, 38], we explicitly avoid making this assumption.

We will use the definition of weak pseudorandomness [2] which can be viewed as a multi-instance decisional Diffie-Hellman assumption for group actions.

Definition 3 (Weak Pseudorandomness). *Let $\mathsf{EGA} = (\mathcal{G}, \mathcal{X}, \star, \tilde{x})$ be an effective group action. Consider the games $\mathrm{G}_{\mathsf{EGA}}^{\mathsf{Wk\text{-}PR\text{-}0}}$ and $\mathrm{G}_{\mathsf{EGA}}^{\mathsf{Wk\text{-}PR\text{-}1}}$ for an adversary $\mathcal{A}$ as defined in Figure 1. We define the advantage of $\mathcal{A}$ in distinguishing the two games as*

$$\mathsf{Adv}_{\mathsf{EGA}}^{\mathsf{Wk\text{-}PR}}(\mathcal{A}) := |\Pr[\mathrm{G}_{\mathsf{EGA}}^{\mathsf{Wk\text{-}PR\text{-}1}}(\mathcal{A}) \Rightarrow 1] - \Pr[\mathrm{G}_{\mathsf{EGA}}^{\mathsf{Wk\text{-}PR\text{-}0}}(\mathcal{A}) \Rightarrow 1]| \ .$$

2.2 Updatable Encryption

We recall the definition of an updatable encryption scheme and security definitions from [11].

Syntax. An updatable encryption scheme UE for message space $\mathbf{M}$, key space $\mathbf{K}$ and ciphertext space $\mathbf{C}$ consists of the following algorithms:

- $\mathsf{KeyGen} \rightarrow k$: The key generation algorithm outputs a key $k \in \mathbf{K}$.
- $\mathsf{Enc}(k, M) \rightarrow C$: On input a key $k \in \mathbf{K}$ and a message $M \in \mathbf{M}$, the encryption algorithm computes a ciphertext C.
- $\mathsf{Dec}(k, C) \rightarrow \{M, \perp\}$: On input a key $k \in \mathbf{K}$ and a ciphertext $C \in \mathbf{C}$, the decryption algorithm outputs a message M or a special symbol $\perp$ indicating failure.
- $\mathsf{TokenGen}(k, k') \rightarrow \Delta$: On input two keys $k, k' \in \mathbf{K}$, the token generation algorithm outputs an update token Δ.
- $\mathsf{Upd}(\Delta, C) \rightarrow C'$: On input a token Δ and a ciphertext $C \in \mathbf{C}$, the update algorithm computes an updated ciphertext C'.

If a scheme is defined relative to some public parameters, we assume that they are implicit input to all algorithms. In this work, we only consider schemes where Dec, $\mathsf{TokenGen}$ and Upd are deterministic algorithms, which we will indicate accordingly when assigning outputs. Furthermore, we will sometimes make the encryption randomness and randomness space explicit and denote them with r and $\mathbf{R}$, respectively.

The execution of the scheme can be described by epochs. Each key and ciphertext belong to one epoch $\mathsf{e} \in \mathbb{N}$ which we write as $k^{(\mathsf{e})}$ and $C^{(\mathsf{e})}$, respectively. A token $\Delta^{(\mathsf{e}+1)}$ allows to update a ciphertext from epoch e to the next epoch $\mathsf{e}+1$. For correctness, we ask that an updated ciphertext still decrypts correctly under the respective key. This is captured in the following definition.

Definition 4 (Correctness of UE). *Let* UE *be an updatable encryption scheme and* $n_{\mathsf{e}} \in \mathbb{N}$*. We say that* UE *is* perfectly correct *if for any* $M \in \mathbf{M}$*, for* $\mathsf{e} \in [n_{\mathsf{e}} - 1]$*, it holds that*

$$\Pr[\mathsf{Dec}(k^{(n_{\mathsf{e}})}, C^{(n_{\mathsf{e}})}) = M] = 1 \ ,$$

where

- $k^{(\mathsf{e})}, \ldots, k^{(n_{\mathsf{e}})} \leftarrow \mathsf{KeyGen}$*,*
- $C^{(\mathsf{e})} \leftarrow \mathsf{Enc}(k^{(\mathsf{e})}, M)$*, and*
- $\Delta^{(i+1)} := \mathsf{TokenGen}(k^{(i)}, k^{(i+1)})$*,* $C^{(i+1)} := \mathsf{Upd}(\Delta^{(i+1)}, C^{(i)})$ *for* $i \in [\mathsf{e}, n_{\mathsf{e}}-1]$*.*

We will use the security definitions for UE schemes with deterministic and bi-directional updates from [11]. Below, we describe and define three different games, capturing $\mathsf{detIND\text{-}UE\text{-}CPA}$, $\mathsf{detIND\text{-}UE\text{-}CCA}$ and $\mathsf{INT\text{-}CTXT}$ security, respectively.

Confidentiality. In the game $\mathrm{G}_{\mathsf{UE}}^{\mathsf{detIND\text{-}UE\text{-}CPA}\text{-}b}$ (cf. Figure 2), which is parameterized by $b \in \{0, 1\}$, the adversary $\mathcal{A}$ has access to a (non-challenge) encryption oracle Enc, a challenge oracle Chall, an oracle Next to proceed to the next epoch, an update oracle Upd, an update oracle for challenges $\mathrm{Upd}\tilde{\mathrm{C}}$ and corruption oracles CorrKey and CorrToken to reveal keys and tokens, respectively.

At the beginning of the game, an initial key $k^{(0)}$ is drawn. The game then initializes variables: an epoch counter e, an encryption counter c, a challenge flag chall and empty sets $(\mathcal{L}, \tilde{\mathcal{L}}, \mathcal{C}, \mathcal{K}, \mathcal{T})$, where

- Set $\mathcal{L}$ stores non-challenge ciphertexts produced by Enc or Upd. It records tuples of the form $(\mathsf{c}, C, \mathsf{e})$. Oracle Upd only updates ciphertexts in $\mathcal{L}$.
- Set $\tilde{\mathcal{L}}$ stores the challenge ciphertext and updated versions of it. The first entry will always be the challenge ciphertext $\tilde{C}$ together with the epoch $\tilde{\mathsf{e}}$. Any call to Next automatically updates the challenge ciphertext into the new epoch, which $\mathcal{A}$ can fetch via $\mathrm{Upd}\tilde{\mathrm{C}}$.
- Set $\mathcal{C}$ stores all epochs in which $\mathcal{A}$ learned an updated version of the challenge ciphertext.
- Set $\mathcal{K}$ stores all epochs in which $\mathcal{A}$ corrupted the secret key.
- Set $\mathcal{T}$ stores epochs in which $\mathcal{A}$ corrupted the update token.

```
Games $\mathrm{G}_{\mathsf{UE}}^{\mathsf{detIND\text{-}UE\text{-}CPA\text{-}}b}(\mathcal{A})$ $\boxed{\mathrm{G}_{\mathsf{UE}}^{\mathsf{detIND\text{-}UE\text{-}CCA\text{-}}b}(\mathcal{A})}$
$k^{(0)} \leftarrow \mathsf{KeyGen}$
$\Delta^{(0)} := \perp$
$(\mathsf{e}, \mathsf{c}, \mathsf{chall}) := (0, 0, 0)$
$(\mathcal{L}, \tilde{\mathcal{L}}, \mathcal{C}, \mathcal{K}, \mathcal{T}) := (\emptyset, \emptyset, \emptyset, \emptyset, \emptyset)$
$b' \leftarrow \mathcal{A}^{\mathcal{O}}$
if $\mathcal{K}^* \cap \mathcal{C}^* \neq \emptyset$ or $\mathcal{I}^* \cap \mathcal{C}^* \neq \emptyset$ :
    return $b' \xleftarrow{\$} \{0,1\}$
return $b'$

Oracle Enc($M$)
$C \leftarrow \mathsf{Enc}(k^{(\mathsf{e})}, M)$
$\mathsf{c} := \mathsf{c} + 1$
$\mathcal{L} := \mathcal{L} \cup \{(\mathsf{c}, C, \mathsf{e})\}$
return $C$

Oracle Chall($\bar{M}, \bar{C}$)
if $\mathsf{chall} = 1$ return $\perp$                //only once
$\mathsf{chall} := 1$
$\tilde{\mathsf{e}} := \mathsf{e}$
if $(\cdot, \bar{C}, \mathsf{e} - 1) \notin \mathcal{L}$ :
    return $\perp$
if $b = 0$ :
    $\tilde{C}_{\mathsf{e}} \leftarrow \mathsf{Enc}(k^{(\mathsf{e})}, \bar{M})$
else if $b = 1$ :
    $\tilde{C}_{\mathsf{e}} := \mathsf{Upd}(\Delta^{(\mathsf{e})}, \bar{C})$
$\mathcal{C} := \mathcal{C} \cup \{\mathsf{e}\}$
$\tilde{\mathcal{L}} := \tilde{\mathcal{L}} \cup \{(\tilde{C}_{\mathsf{e}}, \mathsf{e})\}$
return $\tilde{C}_{\mathsf{e}}$

Oracle Next
$\mathsf{e} := \mathsf{e} + 1$
$k^{(\mathsf{e})} \leftarrow \mathsf{KeyGen}$
$\Delta^{(\mathsf{e})} := \mathsf{TokenGen}(k^{(\mathsf{e}-1)}, k^{(\mathsf{e})})$
if $\mathsf{chall} = 1$ :
    $\tilde{C}_{\mathsf{e}} := \mathsf{Upd}(\Delta^{(\mathsf{e})}, \tilde{C}_{\mathsf{e}-1})$

Oracle Upd($C_{\mathsf{e}-1}$)
if $\nexists j$ s.t. $(j, C_{\mathsf{e}-1}, \mathsf{e} - 1) \in \mathcal{L}$ :
    return $\perp$
$C_{\mathsf{e}} := \mathsf{Upd}(\Delta^{(\mathsf{e})}, C_{\mathsf{e}-1})$
$\mathcal{L} := \mathcal{L} \cup \{(j, C_{\mathsf{e}}, \mathsf{e})\}$
return $C_{\mathsf{e}}$

Oracle Upd$\tilde{\mathrm{C}}$        //return $\tilde{C}_{\mathsf{e}}$ from Next
if $\mathsf{chall} \neq 1$ return $\perp$
$\mathcal{C} := \mathcal{C} \cup \{\mathsf{e}\}$
$\tilde{\mathcal{L}} := \tilde{\mathcal{L}} \cup \{(\tilde{C}_{\mathsf{e}}, \mathsf{e})\}$
return $\tilde{C}_{\mathsf{e}}$

Oracle CorrKey($\hat{\mathsf{e}}$)
if $\hat{\mathsf{e}} > \mathsf{e}$ return $\perp$
$\mathcal{K} := \mathcal{K} \cup \{\hat{\mathsf{e}}\}$
return $k^{(\hat{\mathsf{e}})}$

Oracle CorrToken($\hat{\mathsf{e}}$)
if $\hat{\mathsf{e}} > \mathsf{e}$ return $\perp$
$\mathcal{T} := \mathcal{T} \cup \{\hat{\mathsf{e}}\}$
return $\Delta^{(\hat{\mathsf{e}})}$

┌──────────────────────────────────────────────┐
Oracle Dec($C$)
if $\mathsf{chall} = 1$ and $C \in \tilde{\mathcal{L}}^*$ :
    return $\perp$
return $\mathsf{UE.Dec}(k^{(\mathsf{e})}, C)$
└──────────────────────────────────────────────┘
```

Fig. 2. Games $\mathrm{G}_{\mathsf{UE}}^{\mathsf{detIND\text{-}UE\text{-}CPA\text{-}}b}$ (without code in boxes) and $\mathrm{G}_{\mathsf{UE}}^{\mathsf{detIND\text{-}UE\text{-}CCA\text{-}}b}$ (including code in boxes), where $b \in \{0,1\}$ and adversary $\mathcal{A}$ has access to oracles $\mathcal{O} := \{$Enc, Chall, Next, Upd, Upd$\tilde{\mathrm{C}}$, CorrKey, CorrToken$\}$. In $\mathrm{G}_{\mathsf{UE}}^{\mathsf{detIND\text{-}UE\text{-}CCA\text{-}}b}$, $\mathcal{A}$ additionally has access to oracle Dec. The definition of sets $\mathcal{K}^*, \mathcal{C}^*$, $\mathcal{I}^*$ and $\tilde{\mathcal{L}}^*$ to prevent trivial wins is given in the textual description.

The challenge oracle Chall takes as input a message $\bar{M}$ and a ciphertext $\bar{C}$ such that the ciphertext was the output of a query to Enc from the previous epoch. Then, depending on the bit b, the game either computes a fresh encryption of $\bar{M}$ or an update of ciphertext $\bar{C}$. As shown in [11], this captures indistinguishability of ciphertexts as well as unlinkability of updates at the same time. All oracles can be queried adaptively and multiple times, except the challenge oracle which may only be queried once.

At some point, $\mathcal{A}$ stops and outputs a bit b'. The game checks whether $\mathcal{A}$ could have trivially learned the bit based on its queries and the inferred knowledge. In order to perform the check, the following sets need to be computed:

- Set $\mathcal{K}^*$ contains all epochs for which the adversary corrupted the key or learned the key via a token. Here, it is important to note that we look at bi-directional updates, meaning the knowledge of a key $k^{(\mathsf{e})}$ and tokens $\Delta^{(\mathsf{e})}, \Delta^{(\mathsf{e}+1)}$ allows to compute both $k^{(\mathsf{e}-1)}, k^{(\mathsf{e}+1)}$. More formally, the set $\mathcal{K}^*$ is defined as $\mathcal{K}^* := \{\mathsf{e} \in [0, n] \mid \mathsf{CorrK}(\mathsf{e}) = \mathbf{true}\}$, where

$$\begin{aligned}\mathsf{CorrK}(\mathsf{e}) = \mathbf{true} \iff & (\mathsf{e} \in \mathcal{K}) \vee (\mathsf{CorrK}(\mathsf{e}-1) \wedge \mathsf{e} \in \mathcal{T}) \\ & \vee (\mathsf{CorrK}(\mathsf{e}+1) \wedge \mathsf{e}+1 \in \mathcal{T}) \ .\end{aligned}$$

- Set $\mathcal{T}^*$ contains all epochs for which the adversary corrupted the token or learned the token via a key. For this note that bi-directional updates allow to compute a token $\Delta^{(\mathsf{e})}$ from keys $k^{(\mathsf{e}-1)}$ and $k^{(\mathsf{e})}$, where we additionally consider keys contained in $\mathcal{K}^*$. More formally, we define the set of tokens as $\mathcal{T}^* := \{\mathsf{e} \in [0, n] \mid (\mathsf{e} \in \mathcal{T}) \vee (\mathsf{e} \in \mathcal{K}^* \wedge \mathsf{e}-1 \in \mathcal{K}^*)\}$.
- Set $\mathcal{C}^*$ contains the challenge epoch as well as all epochs for which the adversary knows updated versions of the challenge ciphertext. Since we consider deterministic updates, the knowledge of a ciphertext $\tilde{C}_{\mathsf{e}}$ along with tokens $\Delta^{(\mathsf{e})}, \Delta^{(\mathsf{e}+1)}$ allows to compute both corresponding ciphertexts $\tilde{C}_{\mathsf{e}-1}$ and $\tilde{C}_{\mathsf{e}+1}$. More formally, $\mathcal{C}^* := \{\mathsf{e} \in [0, n] \mid \mathsf{ChallEq}(\mathsf{e}) = \mathbf{true}\}$, where

$$\begin{aligned}\mathsf{ChallEq}(\mathsf{e}) = \mathbf{true} \iff & (\mathsf{e} \in \mathcal{C}) \vee (\mathsf{ChallEq}(\mathsf{e}-1) \wedge \mathsf{e} \in \mathcal{T}^*) \\ & \vee (\mathsf{ChallEq}(\mathsf{e}+1) \wedge \mathsf{e}+1 \in \mathcal{T}^*) \ .\end{aligned}$$

Let c be the counter value for the ciphertext used in the challenge and was previously output by Enc. In the following, we will denote this ciphertext as the challenge-input ciphertext. We define the two following sets:

- Set $\mathcal{I}$ contains the epoch in which the challenge-input ciphertext was first created and all epochs for which the adversary learned updates of the challenge-input ciphertext via Upd. More formally, $\mathcal{I} := \{\mathsf{e} \in [0, n] \mid (\mathsf{c}, \cdot, \mathsf{e}) \in \mathcal{L}\}$.
- Set $\mathcal{I}^*$ contains $\mathcal{I}$ and all epochs for which the adversary has inferred knowledge about the challenge-input ciphertext via corrupted tokens (similar to $\mathcal{C}^*$). More formally, $\mathcal{I}^* := \{\mathsf{e} \in [0, n] \mid \mathsf{ChallinputEq}(\mathsf{e}) = \mathbf{true}\}$, where

$$\begin{aligned}\mathsf{ChallinputEq}(\mathsf{e}) = \mathbf{true} \iff & (\mathsf{e} \in \mathcal{I}) \vee (\mathsf{ChallinputEq}(\mathsf{e}-1) \wedge \mathsf{e} \in \mathcal{T}^*) \\ & \vee (\mathsf{ChallinputEq}(\mathsf{e}+1) \wedge \mathsf{e}+1 \in \mathcal{T}^*) \ .\end{aligned}$$

Finally, in order to prevent trivial attacks, the game checks whether $\mathcal{K}^* \cap \mathcal{C}^* = \emptyset$ and $\mathcal{I}^* \cap \mathcal{C}^* = \emptyset$. If both checks succeed, the game returns $\mathcal{A}$'s output b' and otherwise it returns a random bit $b' \xleftarrow{\$} \{0, 1\}$.

When considering active security in the sense of $\mathsf{detIND\text{-}UE\text{-}CCA}$, the game additionally provides access to a decryption oracle Dec which allows to decrypt any ciphertext, except for the challenge ciphertext and updated versions of it. In order to detect trivial wins, we use set $\tilde{\mathcal{L}}^*$ which is updated when the challenge query is issued and when the challenge ciphertext is updated. In Supplementary Material B we recall the algorithm to update $\tilde{\mathcal{L}}^*$ as given in [11].

We now formally define the advantage of an adversary $\mathcal{A}$ in these games.

```
Game $\mathsf{G}_{\mathsf{UE}}^{\mathsf{INT\text{-}CTXT}}(\mathcal{A})$
$k^{(0)} \leftarrow \mathsf{UE.KeyGen}$
$\Delta^{(0)} := \bot$
$(\mathsf{e}, \mathsf{c}) := (0, 0)$
$(\mathsf{chall}, \mathsf{twf}, \mathsf{win}) := (0, 0, 0)$
$(\mathcal{L}, \mathcal{C}, \mathcal{K}, \mathcal{T}) = (\emptyset, \emptyset, \emptyset, \emptyset)$
$\mathcal{A}^{\mathcal{O}}$
if $\mathsf{twf} = 1$ :
    return $0$
return $\mathsf{win}$

Oracle Enc($M$)
$C \leftarrow \mathsf{UE.Enc}(k^{(\mathsf{e})}, M)$
$\mathsf{c} := \mathsf{c} + 1$
$\mathcal{L} := \mathcal{L} \cup \{(\mathsf{c}, C, \mathsf{e})\}$
return $C$

Oracle Try($\tilde{C}$)
if $\mathsf{chall} = 1$ return $\bot$
$\mathsf{chall} := 1$
if $\mathsf{e} \in \mathcal{K}^*$or $\tilde{C} \in \mathcal{L}^*$ :
    $\mathsf{twf} := 1$
if $\mathsf{UE.Dec}(k^{(\mathsf{e})}, \tilde{C}) \neq \bot$ :
    $\mathsf{win} := 1$

Oracle Next
$\mathsf{e} := \mathsf{e} + 1$
$k^{(\mathsf{e})} \leftarrow \mathsf{UE.KeyGen}$
$\Delta^{(\mathsf{e})} := \mathsf{UE.TokenGen}(k^{(\mathsf{e}-1)}, k^{(\mathsf{e})})$

Oracle Upd($C_{\mathsf{e}-1}$)
if $\nexists j$ s.t. $(j, C_{\mathsf{e}-1}, \mathsf{e}-1) \in \mathcal{L}$ :
    return $\bot$
$C_{\mathsf{e}} := \mathsf{UE.Update}(\Delta^{(\mathsf{e})}, C_{\mathsf{e}-1})$
$\mathcal{L} := \mathcal{L} \cup \{(j, C_{\mathsf{e}}, \mathsf{e})\}$
return $C_{\mathsf{e}}$

Oracle CorrKey($\hat{\mathsf{e}}$)
if $\hat{\mathsf{e}} > \mathsf{e}$ return $\bot$
$\mathcal{K} := \mathcal{K} \cup \{\hat{\mathsf{e}}\}$
return $k^{(\hat{\mathsf{e}})}$

Oracle CorrToken($\hat{\mathsf{e}}$)
if $\hat{\mathsf{e}} > \mathsf{e}$ return $\bot$
$\mathcal{T} := \mathcal{T} \cup \{\hat{\mathsf{e}}\}$
return $\Delta^{(\hat{\mathsf{e}})}$
```

Fig. 3. The INT-CTXT security game for an updatable encryption scheme UE. Adversary $\mathcal{A}$ has access to oracles $\mathcal{O} = \{\text{Enc}, \text{Next}, \text{Upd}, \text{CorrKey}, \text{CorrToken}, \text{Try}\}$.

Definition 5 (detIND-UE-CPA/CCA Security). *Let* XXX $\in$ {CPA, CCA} *and consider the games* $\mathsf{G}_{\mathsf{UE}}^{\mathsf{detIND\text{-}UE\text{-}XXX\text{-}0}}$ *and* $\mathsf{G}_{\mathsf{UE}}^{\mathsf{detIND\text{-}UE\text{-}XXX\text{-}1}}$ *for an updatable encryption scheme* UE *and an adversary* $\mathcal{A}$ *as defined in Figure 2. We define the advantage of* $\mathcal{A}$ *in distinguishing the two games as*

$$\begin{aligned}\mathsf{Adv}_{\mathsf{UE}}^{\mathsf{detIND\text{-}UE\text{-}XXX}}(\mathcal{A}) := |\Pr[&\mathsf{G}_{\mathsf{UE}}^{\mathsf{detIND\text{-}UE\text{-}XXX\text{-}1}}(\mathcal{A}) \Rightarrow 1] \\ &- \Pr[\mathsf{G}_{\mathsf{UE}}^{\mathsf{detIND\text{-}UE\text{-}XXX\text{-}0}}(\mathcal{A}) \Rightarrow 1]| \ .\end{aligned}$$

Ciphertext Integrity. In Figure 3, we define the ciphertext integrity game. The game is similar to the confidentiality games, providing the adversary access to an encryption oracle Enc, an oracle Next to proceed to the next epoch, an update oracle Upd and corruption oracles CorrKey and CorrToken. We do not need a challenge oracle or challenge-update oracle. Instead, the task in this game is to produce a non-trivial ciphertext $\tilde{C}$ such that $\tilde{C}$ decrypts successfully. Here, non-trivial refers to the fact that $\tilde{C}$ must not have been an output of Enc or Upd. As in the previous game, we also need to take into account the (inferred) knowledge through key and token corruptions. This is captured in oracle Try, where we use the sets $\mathcal{K}^*$ and $\mathcal{L}^*$. In Supplementary Material B, we recall the algorithm how to compute $\mathcal{L}^*$, as given in [11]. Note that we only allow one query to Try, as captured by the $\mathsf{INT\text{-}CTXT}^{\mathsf{S}}$ notion in [11], which is equivalent to a version with multiple Try queries. We now define INT-CTXT security as follows.

Definition 6 **($\mathsf{INT\text{-}CTXT}$ Security).** *Consider the game* $\mathrm{G}_{\mathsf{UE}}^{\mathsf{INT\text{-}CTXT}}$ *for an updatable encryption scheme* UE *and an adversary* $\mathcal{A}$ *as defined in Figure 3. We define the advantage of* $\mathcal{A}$ *in winning the game as*

$$\mathsf{Adv}_{\mathsf{UE}}^{\mathsf{INT\text{-}CTXT}}(\mathcal{A}) \coloneqq \Pr[\mathrm{G}_{\mathsf{UE}}^{\mathsf{INT\text{-}CTXT}}(\mathcal{A}) \Rightarrow 1] \ .$$

Finally, we recall the statement from [11] showing that $\mathsf{detIND\text{-}UE\text{-}CPA}$ and $\mathsf{INT\text{-}CTXT}$ together imply $\mathsf{detIND\text{-}UE\text{-}CCA}$.

Theorem 1 **(Theorem 3 in [11]).** *Let* UE *be an updatable encryption scheme. For any adversary* $\mathcal{A}$ *against the* $\mathsf{detIND\text{-}UE\text{-}CCA}$ *security of* UE *there exist adversaries* $\mathcal{B}, \mathcal{C}$ *against the* $\mathsf{detIND\text{-}UE\text{-}CPA}$ *and* $\mathsf{INT\text{-}CTXT}$ *security of* UE *such that*

$$\mathsf{Adv}_{\mathsf{UE}}^{\mathsf{detIND\text{-}UE\text{-}CCA}}(\mathcal{A}) \leq 2 \cdot \mathsf{Adv}_{\mathsf{UE}}^{\mathsf{INT\text{-}CTXT}}(\mathcal{B}) + \mathsf{Adv}_{\mathsf{UE}}^{\mathsf{detIND\text{-}UE\text{-}CPA}}(\mathcal{C}) \ ,$$

where the running times of $\mathcal{B}$ *and* $\mathcal{C}$ *are about that of* $\mathcal{A}$.

3 UE from Non-Mappable Group Actions

In this section, we construct a new updatable encryption scheme called $\mathsf{BIN\text{-}UE}$. Its main advantage is that $\mathsf{BIN\text{-}UE}$ does not require the group action to be mappable, making it possible to instantiate $\mathsf{BIN\text{-}UE}$ from plain CSIDH. Furthermore, its security relies on a standard assumption.

We define $\mathsf{BIN\text{-}UE}$ in Figure 4. On a high level, the bits of a message are encoded into the index of two basis elements $x_0, x_1 \in \mathcal{X}$. These basis elements are chosen randomly for each encryption, which eliminates the need for either a nonce or an ideal cipher to ensure $\mathsf{detIND\text{-}UE\text{-}CPA}$ security. For n-bit messages, the encryption key consists of n group elements. Each bit is then encrypted individually by applying a group element to one of the basis elements.

Decryption of a message first inverts the group action and subsequently detects whether the resulting element is x_0 or x_1. Of course, without further constraints it is a priori not clear which basis element was used for the 0-bit, which is why we sort the basis elements lexicographically and use the smaller one (according to $\prec_{\mathsf{lex}}$) for the 0-bit. To prevent decryption failures, we set the message space to be $\mathbf{M} \coloneqq \{0,1\}^n \setminus \{0^n, 1^n\}$ as otherwise the messages 0^n and 1^n would be indistinguishable.

Proposition 1. *The* $\mathsf{BIN\text{-}UE}$ *scheme is perfectly correct.*

Proof. Let $M = (m_1, \ldots, m_n) \in \mathbf{M}$ be an arbitrary message, $k^{(0)}, k^{(1)} \leftarrow \mathsf{UE.KeyGen}$ be two epoch keys for adjacent epochs. We show that

$$\Pr[\mathsf{UE.Dec}(k^{(1)}, C^{(1)}) = M] = 1$$

for $C^{(1)} \coloneqq \mathsf{UE.Update}(\Delta^{(1)}, C^{(0)})$, $\Delta^{(1)} \coloneqq \mathsf{UE.TokenGen}(k^{(0)}, k^{(1)})$ and $C^{(0)} \leftarrow \mathsf{UE.Enc}(k^{(0)}, M)$. The general case of non-adjacent epochs then follows via induction.

```
BIN-UE.KeyGen
$k := (k_1, \ldots, k_n) \xleftarrow{\$} \mathcal{G}^n$
return $k$

BIN-UE.Update$(C, \Delta)$
Let $\Delta = (\Delta_1, \ldots, \Delta_n)$
  and $C = (c_1, \ldots, c_n)$
return $(\Delta_1 \star c_1, \ldots, \Delta_n \star c_n)$

BIN-UE.Enc$(k, M)$
Let $M = (m_1, \ldots, m_n)$
$x_0, x_1 \xleftarrow{\$} \mathcal{X}$          // $x_0 \prec_{\mathsf{lex}} x_1$
for $i \in [n]$ :
   $c_i := k_i \star x_{m_i}$
return $c := (c_1, \ldots, c_n)$

BIN-UE.TokenGen$(k, k')$
Let $k = (k_1, \ldots, k_n)$ and $k' = (k'_1, \ldots, k'_n)$
return $\Delta := (k'_1 k_1^{-1}, \ldots, k'_n k_n^{-1})$

BIN-UE.Dec$(k, C)$
Let $C = (c_1, \ldots, c_n)$ and $k = (k_1, \ldots, k_n)$
for $i \in [n]$ :
   $x_i = k_i^{-1} \star c_i$
if $|\{x_1, \ldots, x_n\}| = 2$ :
   Let $\{\tilde{x}_0, \tilde{x}_1\} = \{x_1, \ldots, x_n\}$ and $\tilde{x}_0 \prec_{\mathsf{lex}} \tilde{x}_1$
   for $i \in [n]$ :
      $m_i := [\![x_i = \tilde{x}_1]\!]$
   return $M := (m_1, \ldots, m_n)$
return $\perp$
```

Fig. 4. The updatable encryption scheme scheme BIN-UE.

First, we have that

$$C^{(0)} = (C_1^{(0)}, \ldots, C_n^{(0)}) = (k_1^{(0)} \star x_{m_1}, \ldots k^{(0)} \star x_{m_n})$$

for two random set elements $x_0, x_1 \in \mathcal{X}$ with $x_0 \prec_{\mathsf{lex}} x_1$. Essentially, we encode a bit with either the (lexicographically) larger or smaller set element. Updating the ciphertext with the token $\Delta^{(1)}$ then becomes

$$C^{(1)} = (\Delta_1^{(1)} \star C_1^{(0)}, \ldots, \Delta_n^{(1)} \star C_n^{(0)}) = (k_1^{(1)} \star x_{m_1}, \ldots k_n^{(1)} \star x_{m_n})$$

because for each entry $\Delta_i^{(1)} \cdot k_i^{(0)} = k_i^{(1)} (k_i^{(0)})^{-1} \cdot k_i^{(0)} = k_i^{(1)}$. Lastly, decrypting $C^{(1)}$ with $k^{(1)}$ yields

$$((k_1^{(1)})^{-1} \star C_1^{(1)}, \ldots, (k_n^{(1)})^{-1} \star C_n^{(1)}) = (x_{m_1}, \ldots, x_{m_n})$$

Note that since $\mathbf{M}$ does not contain the bit strings 0^n and 1^n the above tuple contains exactly two distinct set elements. We are now left with detecting at each position whether the element x_{m_i} is the (lexicographically) larger or smaller element, yielding the desired bitstring $M = (m_1, \ldots, m_n)$. □

3.1 Security

In this section, we establish the security of BIN-UE. To this end, we introduce the following hardness assumption.

Definition 7 (Multi Strong Pseudorandomness (Multi-St-PR)). *Let* EGA $= (\mathcal{G}, \mathcal{X}, \star, \tilde{x})$ *and let* $n \in \mathbb{N}$. *Consider the game defined in Figure 5. We define the advantage of an adversary* $\mathcal{A}$ *winning the* Multi-St-PR *game as*

$$\mathsf{Adv}_{\mathsf{EGA}}^{\mathsf{Multi\text{-}St\text{-}PR}}(\mathcal{A}) := |\Pr[\mathsf{G}_{\mathsf{EGA}}^{\mathsf{Multi\text{-}St\text{-}PR\text{-}1}}(\mathcal{A}) \Rightarrow 1] - \Pr[\mathsf{G}_{\mathsf{EGA}}^{\mathsf{Multi\text{-}St\text{-}PR\text{-}0}}(\mathcal{A}) \Rightarrow 1]| \ .$$

```
Games $\mathrm{G}^{\mathsf{Multi\text{-}St\text{-}PR\text{-}}b}_{\mathsf{EGA}}(\mathcal{A})$
$g_1, \ldots, g_n \xleftarrow{\$} \mathcal{G}$
$b' \leftarrow \mathcal{A}^{\textsc{Eval}}$
return $b'$

Oracle $\textsc{Eval}(\beta \in \{0,1\}^n)$
$x_0, x_1, y_1, \ldots, y_n \xleftarrow{\$} \mathcal{X}$    // $x_0 \prec_{\mathsf{lex}} x_1$
if $b = 0$:
    return $(x_0, x_1, y_1, \ldots, y_n)$
if $b = 1$:
    return $(x_0, x_1, g_1 \star x_{\beta_1}, \ldots, g_n \star x_{\beta_n})$
```

Fig. 5. Games $\mathrm{G}^{\mathsf{Multi\text{-}St\text{-}PR\text{-}}b}_{\mathsf{EGA}}$ for EGA, where $b \in \{0, 1\}$.

```
Hybrid $\mathrm{H}_i(\mathcal{A})$
$g_1, \ldots, g_n \xleftarrow{\$} \mathcal{G}$
$b' \leftarrow \mathcal{A}^{\textsc{Eval}}$
return $b'$

Oracle $\textsc{Eval}(\beta)$
$x_0, x_1, y_1, \ldots, y_n \xleftarrow{\$} \mathcal{X}$    // $x_0 \prec_{\mathsf{lex}} x_1$
return $(x_0, x_1, g_1 \star x_{\beta_1}, \ldots, g_i \star x_{\beta_i}, y_{i+1}, \ldots, y_n)$
```

Fig. 6. The i-th hybrid.

Remark 3. The restriction that $x_0 \prec_{\mathsf{lex}} x_1$ is without loss of generality. More concretely, one could reduce the "sorted" variant of Multi-St-PR to an "unsorted" variant by just querying the unsorted Eval oracle until the output elements are sorted (which happens with probability 1/2). Note that an adversary can query Eval multiple times on the same input.

Remark 4. In contrast to the definition of $\mathbf{M}$, we do not require $\beta \neq 0^n, 1^n$ as it does not affect the reduction to the detIND-UE-CPA security of BIN-UE.

Although Multi-St-PR appears non-standard at first, we show that it reduces to the standard Wk-PR assumption.

Proposition 2 (**Wk-PR $\Rightarrow$ Multi-St-PR**). *Let* EGA $= (\mathcal{G}, \mathcal{X}, \star, \tilde{x})$. *For any adversary $\mathcal{A}$ against* Multi-St-PR *of* EGA*, there exists an adversary $\mathcal{B}$ against* Wk-PR *of* EGA *such that*

$$\mathsf{Adv}^{\mathsf{Multi\text{-}St\text{-}PR}}_{\mathsf{EGA}}(\mathcal{A}) \leq n \cdot \mathsf{Adv}^{\mathsf{Wk\text{-}PR}}_{\mathsf{EGA}}(\mathcal{B})$$

where n is defined as in Figure 5 and the running time of $\mathcal{B}$ is about that of $\mathcal{A}$.

Proof. We prove the statement via a hybrid argument where in each hybrid we embed the Wk-PR challenge at one position in the Multi-St-PR tuple.

Let $\mathrm{H}_0, \ldots, \mathrm{H}_n$ be hybrids where H_i is then defined as in Figure 6. It is clear that $\mathrm{H}_0 = \mathrm{G}^{\mathsf{Multi\text{-}St\text{-}PR\text{-}0}}_{\mathsf{EGA}}$ and $\mathrm{H}_n = \mathrm{G}^{\mathsf{Multi\text{-}St\text{-}PR\text{-}1}}_{\mathsf{EGA}}$. We now show that for any adversary $\mathcal{A}$ that distinguishes two adjacent hybrids there exists an adversary $\mathcal{B}$ against Wk-PR such that

$$|\Pr[\mathrm{H}_{i-1}(\mathcal{A}) \Rightarrow 1] - \Pr[\mathrm{H}_i(\mathcal{A}) \Rightarrow 1]| \leq \mathsf{Adv}^{\mathsf{Wk\text{-}PR}}_{\mathsf{EGA}}(\mathcal{B}) \ .$$

Consider the reduction in Figure 7. Evidently, if the oracle Sample returns tuples of the form $(x, g \star x)$ for uniformly random $x \in \mathcal{X}$ and some fixed $g \in \mathcal{G}$,

```
Adversary $\mathcal{B}^{\textsc{Sample}}$
$g_1, \ldots, g_{i-1} \xleftarrow{\$} \mathcal{G}$
$b' \leftarrow \mathcal{A}^{\textsc{Eval}}$
return $b'$

Oracle $\textsc{Eval}(\beta)$
$(x_0, y_0) \leftarrow \textsc{Sample}$          // $y_i = g \star x_i$ or $y_i \xleftarrow{\$} \mathcal{X}$
$(x_1, y_1) \leftarrow \textsc{Sample}$          // w.l.o.g. $x_0 \prec_{\mathsf{lex}} x_1$
$y_{i+1}, \ldots, y_n \xleftarrow{\$} \mathcal{X}$
if $\beta_i = 0$:
    $\tilde{y} \coloneqq y_0$
else :
    $\tilde{y} \coloneqq y_1$
return $(x_0, x_1, g_1 \star x_{\beta_1}, \ldots, g_{i-1} \star x_{\beta_{i-1}}, \tilde{y}, y_{i+1}, \ldots, y_n)$
```

Fig. 7. Adversary $\mathcal{B}$ for the proof of Proposition 2, simulating either the hybrid H_{i-1} or H_i to $\mathcal{A}$.

then $\mathcal{B}$ perfectly simulates H_i. Likewise, if $\textsc{Sample}$ returns tuples of the form (x, y) for uniformly random $x, y \in \mathcal{X}$, then $\mathcal{B}$ perfectly simulates H_{i-1}. Finally,

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{Multi\text{-}St\text{-}PR}}_{\mathsf{EGA}}(\mathcal{A}) &= |\Pr[\mathrm{G}^{\mathsf{Multi\text{-}St\text{-}PR\text{-}1}}(\mathcal{A}) \Rightarrow 1] - \Pr[\mathrm{G}^{\mathsf{Multi\text{-}St\text{-}PR\text{-}0}}(\mathcal{A}) \Rightarrow 1]| \\
&= |\Pr[\mathrm{H}_0(\mathcal{A}) \Rightarrow 1] - \Pr[\mathrm{H}_n(\mathcal{A}) \Rightarrow 1]| \\
&\leq \sum_{i=1}^{n} |\Pr[\mathrm{H}_{i-1}(\mathcal{A}) \Rightarrow 1] - \Pr[\mathrm{H}_i(\mathcal{A}) \Rightarrow 1]| \\
&= n \cdot \mathsf{Adv}^{\mathsf{Wk\text{-}PR}}_{\mathsf{EGA}}(\mathcal{B}) \ ,
\end{aligned}$$

which concludes the proof. □

We are now ready to establish the passive security of BIN-UE. We highlight that no ideal cipher is necessary to prove detIND-UE-CPA security of BIN-UE. Intuitively, the reason for this is that the definition of Multi-St-PR allows the adversary to supply the value β to the $\textsc{Eval}$ oracle. As we showed in the previous reduction this does not weaken the hardness of Multi-St-PR as it is essentially as hard as Wk-PR (up to a tightness loss). In the detIND-UE-CPA proof, however, this allows the reduction to directly forward messages to the $\textsc{Eval}$ oracle without needing to program an ideal cipher to ensure consistent randomness.

Theorem 2. *Let* BIN-UE *be the scheme described in Figure 4. For any adversary $\mathcal{A}$ against the* detIND-UE-CPA *security of* BIN-UE*, there exists an adversary $\mathcal{B}$ against* Wk-PR *such that*

$$\mathsf{Adv}^{\mathsf{detIND\text{-}UE\text{-}CPA}}_{\mathsf{BIN\text{-}UE}}(\mathcal{A}) \leq 2n(n_{\mathsf{e}} + 1)^3 \cdot \mathsf{Adv}^{\mathsf{Wk\text{-}PR}}_{\mathsf{EGA}}(\mathcal{B}) \ ,$$

where n_{e} is the number of epochs and n is the length of a message in bits. The running time of $\mathcal{B}$ is about that of $\mathcal{A}$.

We use the same proof technique as [11, 33], that is, a hybrid argument across insulated regions. Due to the strong similarities with the security proofs for GAINE and SHINE we defer the full proof to Supplementary Material C and provide a proof sketch below.

In each hybrid, we embed our challenge on the left boundary of the insulated region, also called the left *firewall.* Outside the insulated region, all keys and tokens are generated honestly, while inside the insulated region, tokens are simulated without knowing the corresponding keys. This allows to answer queries to both Upd and $\text{Upd}\tilde{\text{C}}$. Fresh ciphertexts are randomized using the "basis" elements x_0, x_1, eliminating the need for a nonce. Since in the Multi-St-PR game the adversary can control the bit string β, we do not need a programmable ideal cipher in order to embed messages into ciphertexts. Lastly, we note that updates are deterministic.

We give a reduction $\mathcal{B}$ which plays the Multi-St-PR game and perfectly simulates the i-th hybrid to $\mathcal{A}$ if the oracle Eval returns "real" tuples (i.e. $b = 1$) and a random game otherwise. More specifically, the values $g_1 \star x_{\beta_1}, \ldots, g_n \star x_{\beta_n}$ returned by the Eval oracle will be used as the ciphertext of a message $M = (\beta_1, \ldots, \beta_n)$. Therefore, we embed the elements $g_1, \ldots, g_n$ into the key of the chosen challenge epoch. Lastly, if the adversary $\mathcal{A}$ can correctly distinguish between an update and a fresh encryption, we assume that Eval returns "real" tuples. On the other hand if the Eval oracle returns random tuples, all encryptions and updates are truly random as well, making it impossible to distinguish an update from a fresh encryption.

3.2 Instantiation from CSIDH

CSIDH [15] is a popular cryptographic group action based on isogenies between supersingular elliptic curves over $\mathbb{F}_p$. Although CSIDH has many useful properties, it has the disadvantage of not being a *mappable* group action. Fortunately, BIN-UE does not require the group action to be mappable, which means it can be instantiated from plain CSIDH. Note that in contrast to many other (non-commutative) group actions [20], CSIDH is believed to be weakly pseudorandom, even in the post-quantum setting [14].

On the Performance of Updates. Because CSIDH is in general a *restricted* effective group action (see Supplementary Material A), computing an update token is very expensive. The reason is that any component $k_i^{(e)}$ of an epoch key is an element of the group $\mathcal{G}$, which is generated by the elements $(g_1, \ldots, g_\nu)$. Since the group action can only be efficiently evaluated for the elements $(g_1, \ldots, g_\nu)$ one therefore writes $k_i^{(e)}$ as a product

$$k_i^{(e)} = \prod_{j=1}^{\nu} g_j^{m_j}$$

for some exponents $m_j \in \mathbb{Z}$. For performance reasons, one further assumes that the m_j are coming from a small interval $[-\delta, \delta]$. This, however, means that a product $k_i^{(e+1)} \cdot (k_i^{(e)})^{-1}$ could result in exponents larger than $\pm\delta$. It is therefore necessary to reduce each element $k_i^{(e+1)} \cdot (k_i^{(e)})^{-1}$ modulo the so-called *lattice of* *relations* to get a short representation [6]. Unfortunately, a good basis for the lattice of relations is unknown in general (apart from the CSIDH-512 parameter set) and takes subexponential time to compute. Furthermore, even under the knowledge of a basis for the lattice of relation, the reduction essentially consists of a CVP instance which, depending on the quality of the lattice basis, takes subexponential time to solve asymptotically as well.

Recent developments like the SCALLOP group action [22] try to mitigate this issue, however, these alternatives are currently not practical as a single group action evaluation takes minutes to compute. Furthermore, the recently introduced group action CLAPOTIS [42] promises to be a proper EGA, however a complete description of the group action, a security analysis as well as performance results are currently not available. Lastly, we remark that the above performance issues and their possible mitigations also apply to GAINE and $\mathsf{TOGA\text{-}UE}$ [33].

4 An Actively Secure Variant

In this section, we introduce a generic transformation inspired by the *Tag-then-Encrypt* paradigm [39]. Although $\mathsf{detIND\text{-}UE\text{-}CCA}$ security of this transformation does not hold in general [5], we show that by applying the transformation to $\mathsf{BIN\text{-}UE}$ we get a new scheme called $\mathsf{COM\text{-}UE}$ that can indeed be proven $\mathsf{detIND\text{-}UE\text{-}CCA}$ secure in the random oracle model. The proof requires a non-standard assumption, however, we further show that in the AGAM [24] this assumption can be reduced to a variant of the group action discrete logarithm problem.

4.1 The Tag-then-Encrypt Transformation

We start by stating two important properties of an updatable encryption scheme that are needed for the transformation. To this end, let us recall the definitions of randomness-preserving and randomness-recoverable updatable encryption schemes [30].

Definition 8 (Randomness-Preserving UE). *Let* $\mathsf{UE} = (\mathsf{UE.KeyGen}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.TokenGen}, \mathsf{UE.Update})$ *be an updatable encryption scheme. Further, let* $r \in \mathbf{R}$ *be the explicit randomness of* $\mathsf{UE.Enc}$*, i. e.,*

$$\mathsf{UE.Enc}(k, M) = \mathsf{UE.Enc}(k, M; r) \ .$$

The scheme is called randomness-preserving (RP) *if for all keys* $k, k' \xleftarrow{\$} \mathsf{UE.KeyGen}$*, tokens* $\Delta \coloneqq \mathsf{UE.TokenGen}(k, k')$ *and messages* $M \in \mathbf{M}$ *we have*

$$\mathsf{UE.Update}(\Delta, \mathsf{UE.Enc}(k, M; r)) = \mathsf{UE.Enc}(k', M; r) \ .$$

Definition 9 (Randomness-Recoverable Updatable Encryption). *Let the notation be as in the previous definition. A scheme* UE *is called* randomness-recoverable (RR) *if for all keys* $k \xleftarrow{\$} \mathsf{UE.KeyGen}$*, messages* $M \in \mathbf{M}$ *and randomness* $r \in \mathbf{R}$*:*

$$\mathsf{UE.Dec}(k, \mathsf{UE.Enc}(k, M; r)) = (M, r) \ .$$

If UE fulfills both Definitions 8 and 9, then we say that UE is an RP/RR scheme or simply RP/RR for short. Evidently, $\mathsf{BIN\text{-}UE}$ is RP/RR with encryption randomness $r = (x_0, x_1)$.

The Transformation. Let $\mathsf{UE} = (\mathsf{UE.KeyGen}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.TokenGen}, \mathsf{UE.Update})$ be an RP/RR updatable encryption scheme with a message space that can be written as $\mathbf{M} = \mathbf{M}^+ \times \{0,1\}^t$ and randomness space $\mathbf{R}$. Further let $\mathsf{H} : \mathbf{M}^+ \times \mathbf{R} \to \{0,1\}^t$ be a cryptographic hash function. We define a new scheme $\mathsf{UE}^+ = (\mathsf{UE.KeyGen}, \mathsf{UE}^+\mathsf{.Enc}, \mathsf{UE}^+\mathsf{.Dec}, \mathsf{UE.TokenGen}, \mathsf{UE.Update})$ that is identical to UE except for:

- The message space is $\mathbf{M}^+$.
- $\mathsf{UE}^+\mathsf{.Enc}(k, M; r)$: Compute $M' = (M||\mathsf{H}(M, r))$ and subsequently return $C = \mathsf{UE.Enc}(k, M'; r)$.
- $\mathsf{UE}^+\mathsf{.Dec}(k, C)$: Compute $(M', r) = \mathsf{UE.Dec}(k, C)$ and parse M' as $(M||T)$. If $T = \mathsf{H}(M, r)$ then output M, else output $\perp$.

Evidently, the transformation only applies a preprocessing on the message M and therefore does not fundamentally change the way encryption and decryption work. Thus, UE^+ inherits correctness and $\mathsf{detIND\text{-}UE\text{-}CPA}$ security directly from the underlying scheme UE.

Lemma 1. *The following statements hold:*

- *If* UE *is (perfectly) correct, then* UE^+ *is also (perfectly) correct.*
- *If* UE *is* $\mathsf{detIND\text{-}UE\text{-}CPA}$ *secure, then* UE^+ *is also* $\mathsf{detIND\text{-}UE\text{-}CPA}$ *secure.*

Remark 5. Making the encryption randomness implicitly accessible to the adversary via the tag does not weaken security. In fact, for any RP/RR scheme an adversary can learn the randomness of a ciphertext by issuing a key corrupt followed by a regular decryption. When transitioning to the next epoch (which might be the challenge epoch) this randomness is preserved. This is, however, not problematic as the existing security proofs for schemes like GAINE, SHINE and $\mathsf{BIN\text{-}UE}$ show.

4.2 A Full Description of COM-UE

By applying the transformation from Section 4.1 to $\mathsf{BIN\text{-}UE}$ we get a new scheme called $\mathsf{COM\text{-}UE}$ which is formally defined in Figure 8. We let the message space be $\mathbf{M} = \{0,1\}^n \setminus \{0^n, 1^n\}$. Furthermore, we make use of a random oracle H such that $\mathsf{H} : \{0,1\}^n \times \mathcal{X}^2 \to \{0,1\}^t$, where $t \in \mathbb{N}$ determines the length of the tag. Lastly, we set $\ell = n + t$ to be the total number of set elements in a ciphertext.

On the surface, $\mathsf{COM\text{-}UE}$ has some similarities with existing constructions in the literature. In particular, one could view $\mathsf{COM\text{-}UE.Enc}(k, M; r)$ as

$$\mathsf{BIN\text{-}UE.Enc}(k, M; r)||\mathsf{BIN\text{-}UE.Enc}'(k', \mathsf{H}(M, r); r)$$

```
COM-UE.KeyGen
    $k := (k_1, \dots, k_\ell) \xleftarrow{\$} \mathcal{G}^\ell$
    return $k$

COM-UE.Update$(C, \Delta)$
    Let $\Delta = (\Delta_1, \dots, \Delta_\ell)$
        and $C = (c_1, \dots, c_\ell)$
    return $(\Delta_1 \star c_1, \dots, \Delta_n \star c_\ell)$

COM-UE.Enc$(k, M)$
    Let $M = (m_1, \dots, m_n)$
    $x_0, x_1 \xleftarrow{\$} \mathcal{X}$                    // $x_0 \prec_{\mathsf{lex}} x_1$
    $M' := (m'_1, \dots, m'_\ell) := (M \| \mathsf{H}(M, x_0, x_1))$
    for $i \in [\ell]$ :
        $c_i := k_i \star x_{m'_i}$
    return $c := (c_1, \dots, c_\ell)$
```

```
COM-UE.TokenGen$(k, k')$
    Let $k = (k_1, \dots, k_\ell)$ and $k' = (k'_1, \dots, k'_\ell)$
    return $\Delta := (k'_1 k_1^{-1}, \dots, k'_\ell k_\ell^{-1})$

COM-UE.Dec$(k, C)$
    Let $C = (c_1, \dots, c_\ell)$ and $k = (k_1, \dots, k_\ell)$
    for $i \in [\ell]$ :
        $y_i := k_i^{-1} \star c_i$
    if $|\{y_1, \dots, y_\ell\}| = 2$ :
        Let $\{\tilde{x}_0, \tilde{x}_1\} = \{y_1, \dots, y_\ell\}$  // $\tilde{x}_0 \prec_{\mathsf{lex}} \tilde{x}_1$
        for $i \in [\ell]$ :
            $m'_i := [\![ y_i = \tilde{x}_1 ]\!]$
        Parse $M' = (m'_1, \dots, m'_\ell) = (M \| T)$
        if $T = \mathsf{H}(M, \tilde{x}_0, \tilde{x}_1)$ :
            return $M$
    return $\perp$
```

Fig. 8. The scheme COM-UE. The random oracle H is defined in the main body.

for some modified BIN-UE.Enc$'$ that encrypts t bits instead of n. This resembles the Encrypt-*and*-MAC construction of [30] where a ciphertext has the form $C = (c, \tau)$ with $c = \mathsf{Enc}(k, M; r)$ and $\tau = \mathsf{PRF}(k_{\mathsf{PRF}}, (M, r))$ for some *updatable* pseudorandom function PRF.

However, on close inspection $\mathsf{BIN\text{-}UE.Enc}(k, \mathsf{H}(M, r); r)$ cannot be viewed as an updatable PRF. First, observe that we still need to randomize the encryption, which means that the output of the supposed pseudorandom function $\widehat{\mathsf{PRF}}(k_{\mathsf{PRF}}, \cdot) = \mathsf{BIN\text{-}UE.Enc}(k_{\mathsf{PRF}}, \cdot)$ does not only depend on the key k_{PRF}, but also on some additional randomness r, which is not intended for a PRF. Second, even if we assume that we could find a consistent definition of a "randomized PRF", it seems unavoidable that $\widehat{\mathsf{PRF}}$ commits to its own randomness r, which is additionally shared with $\mathsf{BIN\text{-}UE.Enc}(k, M; r)$. Although the latter issue can be circumvented, it still appears hard to reduce the INT-CTXT security of COM-UE exclusively to security guarantees provided by the function $\widehat{\mathsf{PRF}}$. We conclude that COM-UE cannot be viewed as a mere instantiation of the Encrypt-and-MAC construction of [30], making a dedicated security proof necessary.

4.3 Active Security of COM-UE

We now prove that in the random oracle model COM-UE is INT-CTXT secure under a new assumption that we call Multi Strong Unpredictability (Multi-St-UP). In a next step, we show that in the AGAM [24] Multi-St-UP reduces to the Discrete Logarithm Problem with Auxilary Input (DLAI). Due to Lemma 1 and Theorem 1 we get a post-quantum detIND-UE-CCA secure updatable encryption scheme from a reasonable assumption.

We begin by defining our new security assumption.

Definition 10 (Multi Strong Unpredictability (Multi-St-UP)). *Let* $\mathsf{EGA} = (\mathcal{G}, \mathcal{X}, \star, \tilde{x})$ *and* $t \in \mathbb{N}$. *Consider the game defined in Figure 9. We define the*

```
Game $\mathrm{G}^{\mathsf{Multi\text{-}St\text{-}UP}}_{\mathsf{EGA}}(\mathcal{A})$
$g_1, \dots, g_t \xleftarrow{\$} \mathcal{G}$
$\gamma \xleftarrow{\$} \{0,1\}^t$
$(y_1, \dots, y_t) \leftarrow \mathcal{A}^{\textsc{Eval},\textsc{Chall}}$
if $y_i = g_i \star \tilde{x}_{\gamma_i}$ for all $i \in [t]$ :
    return 1
return 0

Oracle $\textsc{Eval}$
$\beta \xleftarrow{\$} \{0,1\}^t$
$x_0, x_1 \xleftarrow{\$} \mathcal{X}$    // $x_0 \prec_{\mathsf{lex}} x_1$
return $(\beta, x_0, x_1, g_1 \star x_{\beta_1}, \dots, g_t \star x_{\beta_t})$

Oracle $\textsc{Chall}(\tilde{x}_0, \tilde{x}_1)$    //only one query
return $\gamma$
```

Fig. 9. The Multi-St-UP game for $t \in \mathbb{N}$, where we require $\tilde{x}_0 \prec_{\mathsf{lex}} \tilde{x}_1$.

advantage of an adversary $\mathcal{A}$ winning the Multi-St-UP *game as*

$$\mathsf{Adv}^{\mathsf{Multi\text{-}St\text{-}UP}}_{\mathsf{EGA}}(\mathcal{A}) := \Pr[\mathrm{G}^{\mathsf{Multi\text{-}St\text{-}UP}}_{\mathsf{EGA}}(\mathcal{A}) \Rightarrow 1] \ .$$

Intuitively, Multi-St-UP captures that for a fixed key $(g_1, \dots, g_t)$ it should be hard to come up with a correct ciphertext for a random message γ. Of course, this is not directly applicable to COM-UE as messages are, in general, chosen by the adversary and therefore not random. However, this intuition does apply to the second part of a plaintext, which in the case of COM-UE is a hash value $\mathsf{H}(M, r)$, assuming H is a random oracle. In the ROM we can then embed the responses from the Eval and Chall oracles into these hash values, making the second part of the plaintext uniformly random.

Theorem 3. *Let* COM-UE *be the scheme described in Section 4.2. For any adversary $\mathcal{A}$ against the* INT-CTXT *security of* COM-UE*, there exists an adversary $\mathcal{B}$ against* Multi-St-UP *of* EGA *such that*

$$\mathsf{Adv}^{\mathsf{INT\text{-}CTXT}}_{\mathsf{COM\text{-}UE}}(\mathcal{A}) \leq (q + n_E)(n_{\mathsf{e}} + 1) \cdot \mathsf{Adv}^{\mathsf{Multi\text{-}St\text{-}UP}}_{\mathsf{EGA}}(\mathcal{B}) + \frac{1}{2^t} + \frac{4n_E}{|\mathcal{X}|^2} \ ,$$

where n_{e} is the number of epochs, n_E is the number of encryption queries and q is the number of queries to the random oracle. The running time of $\mathcal{B}$ is about that of $\mathcal{A}$.

The following proof is based on the same idea as the INT-CTXT proof of SHINE0 [11, Theorem 5.1]. However, we improve the proof in two ways:

We first observe that the authors of [11] did not consider giving the adversary direct access to the ideal cipher[2]. Like in the random oracle model, each party should have access to the ideal cipher and therefore have the ability to query it on any input. In their proof, however, the authors implicitly assume that if the ideal cipher is defined on some input, then the output value must have been set during a call to the Enc oracle. This assumption, however, is insufficient and omits the case where an adversary queries the ideal cipher directly.

Secondly, our proof strategy is *tighter* than the one found in [11]. This is based on the observation that one can abort the adversary once it queried the Try oracle. Therefore, it is only necessary to guess the left border fwl of the insulated region instead of both the left *and* right border. If the guess for fwl

[2] Which plays a similar role as the random oracle in the case of COM-UE.

was correct, then the adversary calls Try before the end of the insulated region, essentially marking the end of the region itself.

Proof. Assume that at some point, the adversary $\mathcal{A}$ submits a ciphertext $\tilde{C}$ to the Try oracle, say in epoch e. Further, assume that no trivial win condition is triggered until that point and that $\tilde{C}$ is a well-formed $\mathsf{BIN\text{-}UE}$ ciphertext. Therefore, $\tilde{C}$ is a fresh ciphertext that was not produced by the game before and additionally there exists an insulated region around epoch e. Let $\mathsf{BIN\text{-}UE.Dec}(k^{(\mathsf{e})}, \tilde{C}) = ((M||T), x_0, x_1) \neq \perp$ where we make explicit that $\mathsf{BIN\text{-}UE.Dec}$ also outputs the randomness used for the ciphertext $\tilde{C}$. In the following, we denote the random oracle by H and store queries in a list $\mathsf{H}[\cdot]$, where each entry is initialized to be empty (i. e., $\perp$). We distinguish three cases:

- $\mathsf{H}[M, x_0, x_1]$ is undefined, i. e., (M, x_0, x_1) has not been queried to the random oracle. Denote the event that the adversary wins in this case with E_0.
- $\mathsf{H}[M, x_0, x_1]$ exists and its value was set during a direct query to H. Denote the event that the adversary wins in this case with E_1.
- $\mathsf{H}[M, x_0, x_1]$ exists and its value was set during an encryption query. Denote the event that the adversary wins in this case with E_2.

Note that these three events are mutually exclusive. In particular, if $\mathsf{H}[M, x_0, x_1]$ is defined then it must have been queried to oracle H before on that exact input. This can either happen via a direct query to H initiated by the adversary or via an encryption query, during which the $\mathsf{INT\text{-}CTXT}$ game needs to query H (cf. Figure 8 and recall that the $\mathsf{COM\text{-}UE.Dec}$ oracle is not present in the $\mathsf{INT\text{-}CTXT}$ game). We now have

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{INT\text{-}CTXT}}_{\mathsf{COM\text{-}UE}}(\mathcal{A}) &= \Pr[\mathrm{G}^{\mathsf{INT\text{-}CTXT}}_{\mathsf{COM\text{-}UE}}(\mathcal{A}) \Rightarrow 1] && (1)\\
&= \Pr[\mathsf{BIN\text{-}UE.Dec}(k^{(\mathsf{e})}, \tilde{C}) \neq \perp] && (2)\\
&\qquad \cdot \Pr[\mathrm{G}^{\mathsf{INT\text{-}CTXT}}_{\mathsf{COM\text{-}UE}}(\mathcal{A}) \Rightarrow 1 \mid \mathsf{BIN\text{-}UE.Dec}(k^{(\mathsf{e})}, \tilde{C}) \neq \perp] \\
&\quad + \Pr[\mathsf{BIN\text{-}UE.Dec}(k^{(\mathsf{e})}, \tilde{C}) = \perp] \\
&\qquad \cdot \Pr[\mathrm{G}^{\mathsf{INT\text{-}CTXT}}_{\mathsf{COM\text{-}UE}}(\mathcal{A}) \Rightarrow 1 \mid \mathsf{BIN\text{-}UE.Dec}(k^{(\mathsf{e})}, \tilde{C}) = \perp] \\
&\leq \Pr[\mathrm{G}^{\mathsf{INT\text{-}CTXT}}_{\mathsf{COM\text{-}UE}}(\mathcal{A}) \Rightarrow 1 \mid \mathsf{BIN\text{-}UE.Dec}(k^{(\mathsf{e})}, \tilde{C}) \neq \perp] && (3)\\
&\leq \Pr[E_0] + \Pr[E_1] + \Pr[E_2] \ , && (4)
\end{aligned}$$

where in Equation (3) we used the fact that

$$\Pr[\mathrm{G}^{\mathsf{INT\text{-}CTXT}}_{\mathsf{COM\text{-}UE}}(\mathcal{A}) \Rightarrow 1 \mid \mathsf{BIN\text{-}UE.Dec}(k^{(\mathsf{e})}, \tilde{C}) = \perp] = 0 \ .$$

First, observe that $\Pr[E_0] \leq \frac{1}{2^t}$ as the reduction can simply choose a random bit string in $\{0,1\}^t$ for the value $\mathsf{H}[M, x_0, x_1]$. The probability of this string being equal to T is exactly $\frac{1}{2^t}$, therefore $\tilde{C}$ is a well-formed $\mathsf{COM\text{-}UE}$ ciphertext with the same probability.

The other two probabilities follow from Lemmata 2 and 3 stated below. Collecting all the probabilities yields the statement in Theorem 3. □

Lemma 2. *Let the notation be as in Theorem 3. We then have*

$$\Pr[E_1] \leq q(n_\mathsf{e} + 1) \cdot \mathsf{Adv}^{\mathsf{Multi\text{-}St\text{-}UP}}_{\mathsf{EGA}}(\mathcal{B}) + \frac{2n_E}{|\mathcal{X}|^2} \ ,$$

where n_e is the number of epochs, n_E is the number of encryption queries and q is the number of queries to the random oracle.

Proof. Consider the reduction in Figure 10. The main idea is to embed the group elements $(g_1, \ldots, g_t)$ of the Multi-St-UP assumption into the second half of the secret key in the challenge epoch of the INT-CTXT game. To this end, the reduction first guesses the left border fwl of the firewall around the challenge epoch, resulting in an advantage loss of $(n_\mathsf{e} + 1)$. The reason why only the left border suffices is that we stop the adversary $\mathcal{A}$ once it queries the Try oracle since at that point the reduction has all the necessary information to extract the solution for the Multi-St-UP assumption. To the left of the challenge epoch, the reduction simulates keys and tokens by itself, whereas inside of the firewall, the reduction simulates *partial* update tokens. Note that we do not need to simulate anything to the right of the epoch where Try is queried. In fact, in the reduction in Figure 10 we simulate update tokens for all epochs $e \in [\mathsf{fwl}, n_\mathsf{e}]$ as we expect the adversary to query Try before the end of the insulated region. If instead the adversary queries for instance the CorrKey oracle for an epoch $\hat{\mathsf{e}} \geq \mathsf{fwl}$ then the guess for fwl was simply wrong and the reduction aborts. Furthermore, we do not need to keep track of trivial win conditions explicitly (in particular the twf and win flags).

Of course, the reduction needs to provide consistent encryption randomness (i. e., consistent basis elements x_0, x_1) which is especially important for ciphertexts that are updated into the firewall. We get around this issue in the same way as in the proof of Theorem 2: For each encryption query to the left of the firewall, reduction $\mathcal{B}$ queries its own Eval oracle to obtain a tuple $(\beta, x_0, x_1, y_1, \ldots, y_t)$ which it stores for later use. It then programs the random oracle such that the "tag" $T = \mathsf{H}[M, x_0, x_1]$ equals β and then encrypts $M' = (M||T)$ with the current secret key and randomness (x_0, x_1). If $\mathsf{H}[M, x_0, x_1]$ is already set, then this programming fails and the reduction needs to abort. Note that the probability of this event is bounded by $2n_E/|\mathcal{X}|^2$ (the additional factor of 2 comes from the fact that $x_0 \prec_{\mathsf{lex}} x_1$). In the description of $\mathcal{B}$ in Figure 10, we implicitly assume this event does not happen.

Once an update or a new encryption is requested for an epoch inside the firewall, reduction $\mathcal{B}$ uses the values $y_i = g_i \star x_{\beta_i}$ from memory to embed the elements g_i into the secret key elements $(k^{(\mathsf{fwl})}_{n+1}, \ldots, k^{(\mathsf{fwl})}_{n+t})$. Note that the first n elements of the secret key are always known to the reduction.

To embed the challenge of the Multi-St-UP game, reduction $\mathcal{B}$ guesses a query index q^* and embeds the challenge γ into the q^*-th query to the random oracle H. Recall that for event E_1 we assume that the solution $\tilde{C}$ that $\mathcal{A}$ presents to Try is essentially a fresh encryption of a message M since $\mathsf{H}[M, x_0, x_1]$ exists and $\mathcal{A}$ queried H on that exact input. In particular, the query $\mathsf{H}(M, x_0, x_1)$ did not occur during an Enc query. If the guess was correct and $\mathcal{A}$ successfully provided such a fresh ciphertext, then reduction $\mathcal{B}$ reverses potential (implicit) updates to the provided $\tilde{C}$, resulting in a valid ciphertext for epoch $\mathsf{e} = \mathsf{fwl}$. It is then easy to verify that the last t elements of that ciphertext are a valid solution for the Multi-St-UP assumption which completes the proof. □

Lemma 3. *Let the notation be as in Theorem 3. We then have*

$$\Pr[E_2] \leq n_E(n_{\mathsf{e}} + 1) \cdot \mathsf{Adv}_{\mathsf{EGA}}^{\mathsf{Multi\text{-}St\text{-}UP}}(\mathcal{B}) + \frac{2n_E}{|\mathcal{X}|^2} \ ,$$

where n_{e} is the number of epochs and n_E is the number of encryption queries.

The proof is almost identical to the proof of Lemma 2, except that now the reduction embeds the challenge during a call to Enc. We defer the full proof to Supplementary Material D.

4.4 Hardness of Multi-St-UP

Our new assumption Multi-St-UP is non-standard, but it abstracts away all the overhead of updatable encryption and allows us to focus on a single, succinct assumption. We now show that in the AGAM introduced by [24] Multi-St-UP is essentially as hard as the Discrete Logarithm Problem with Auxilary Input (DLAI). A more general version of DLAI was introduced in [4] and shown to be hard in the GGAM [24]. For a detailed description of the AGAM we refer the reader to Supplementary Material E.

Definition 11 (Discrete Logarithm Problem with Auxilary Input). *Let* $\mathsf{EGA} = (\mathcal{G}, \mathcal{X}, \star, \tilde{x})$*,* $m \in \mathbb{N}$ *be an integer and* $N = ord(\mathcal{G})$ *be prime. We say that an adversary* $\mathcal{A}$ *solves the Discrete Logarithm Problem with Auxilary Input* (DLAI) *if*

$$\mathcal{A}(z_0, \ldots, z_m) = h \ ,$$

where the z_i *are given by* $z_i = h^i \star \tilde{x}$ *for* $h \xleftarrow{\$} \mathcal{G}$*.*

Remark 6. The authors of [4] defined DLAI for arbitrary (composite) order groups which requires a more intricate choice of the exponents. In particular, the powers of h have to fulfill rather technical constraints related to invertibility modulo N. The results in this section can be adapted to the case of composite order groups at the expense of a much more convoluted notation. In particular, for cryptographic group actions like CSIDH one could constrain DLAI to a large prime-order subgroup of $\mathcal{G}$ as long as its group structure is known[3].

Proposition 3 (DLAI $\Rightarrow$ Multi-St-UP **in the AGAM).** *Let* $\mathcal{A}$ *be an algebraic adversary* $\mathcal{A}$ *against* Multi-St-UP *for* $t \in \mathbb{N}$ *such that* $\mathcal{A}$ *issues at most* q_{Eval}

[3] Knowing the group structure in CSIDH is necessary for updatability already, see Section 3.2.

```
Adversary $\mathcal{B}^{\textsc{Eval},\textsc{Chall}}$
$k^{(0)} \xleftarrow{\$} \mathsf{BIN\text{-}UE.KeyGen}$
$\Delta^{(0)} := \bot$
$\mathsf{e}, \mathsf{c}, \mathsf{q} := 0$
$\mathsf{H}[\cdot] := \bot$
$\mathcal{L} := \emptyset$
$\mathsf{fwl} \xleftarrow{\$} [0, n_\mathsf{e}]$
$q^* \xleftarrow{\$} [1, q]$
for $j \in [0, \mathsf{fwl} - 1]$ :
    $k^{(j)} \leftarrow \mathsf{BIN\text{-}UE.KeyGen}$
    $\Delta^{(j)} := \mathsf{BIN\text{-}UE.TokenGen}(k^{(j-1)}, k^{(j)})$ //†
for $j \in [\mathsf{fwl}, n_\mathsf{e}]$ :
    $k_1^{(j)}, \dots, k_n^{(j)} \xleftarrow{\$} \mathcal{G}^n$
for $j \in [\mathsf{fwl} + 1, n_\mathsf{e}]$ :
    for $\ell \in [1, n]$ :
        $\Delta_\ell^{(j)} := k_\ell^{(j)} / k_\ell^{(j-1)}$
    for $\ell \in [n + 1, t]$ :
        $\Delta_\ell^{(j)} \xleftarrow{\$} \mathcal{G}$
Run $\mathcal{A}^{\mathcal{O}}$
return $\bot$ to $\mathsf{G}^{\mathsf{Multi\text{-}St\text{-}UP}}_{\mathsf{EGA}}$    //Try never called

Oracle Enc($M$)
Let $M = (m_1, \dots, m_n)$
$\mathsf{c} := \mathsf{c} + 1$
$\tau := (\beta, x_0, x_1, y_1, \dots, y_t) \leftarrow \textsc{Eval}$
$\mathsf{H}[M, x_0, x_1] := \beta$
$M' := (M \| \mathsf{H}[M, x_0, x_1])$
if $\mathsf{e} < \mathsf{fwl}$ :
    $C_j^{(\mathsf{e})} := k_j^{(\mathsf{e})} \star x_{m'_j}$
else :
    for $j \in [1, n]$ :
        $C_j^{(e)} := k_j^{(e)} \star x_{m_j}$
    for $j \in [1, t]$ :
        $C_{n+j}^{(e)} := y_j$
$C^{(\mathsf{e})} := (C_1^{(\mathsf{e})}, \dots, C_{n+t}^{(\mathsf{e})})$
$\mathcal{L} := \mathcal{L} \cup \{(\mathsf{c}, M', C^{(\mathsf{e})}, \mathsf{e}, \tau)\}$
return $C^{(\mathsf{e})}$

Oracle Next
$\mathsf{e} := \mathsf{e} + 1$

Oracle CorrKey($\hat{\mathsf{e}}$)
if $\hat{\mathsf{e}} > \mathsf{e}$ : return $\bot$
if $\hat{\mathsf{e}} \geq \mathsf{fwl}$ : abort    //wrong guess for $\mathsf{fwl}$
return $k^{(\hat{\mathsf{e}})}$
```

```
Oracle Try($\tilde{C}$)
$(M', x_0, x_1) := \mathsf{BIN\text{-}UE.Dec}(k^{(\mathsf{e})}, \tilde{C})$
Let $M' = (M \| T)$
if $\mathsf{H}[M, x_0, x_1] \neq \gamma$ : abort    //wrong guess
Set $C^{(\mathsf{e})} = \tilde{C}$
for $\ell \in [\mathsf{e}, \mathsf{fwl}]$ :    //backwards iteration
    for $j \in [n + 1, n + t]$ :
        $C_j^{(\ell-1)} := (\Delta_j^{(\ell)})^{-1} \star C_j^{(\ell)}$
return $C_{n+1}^{(\mathsf{fwl})}, \dots, C_{n+t}^{(\mathsf{fwl})}$ to $\mathsf{G}^{\mathsf{Multi\text{-}St\text{-}UP}}_{\mathsf{EGA}}$

Oracle Upd($C^{(\mathsf{e}-1)}$)
if $\nexists \mathsf{c} : (\mathsf{c}, M', C^{(\mathsf{e}-1)}, \mathsf{e} - 1, \tau) \in \mathcal{L}$ :
    return $\bot$
Let $\tau = (\beta, x_0, x_1, y_1, \dots, y_t)$
if $\mathsf{e} < \mathsf{fwl}$ :
    Let $M' = (m'_1, \dots, m'_{n+t})$
    $C_j^{(\mathsf{e})} := k_j^{(\mathsf{e})} \star x_{m'_j}$
else :
    Let $M' = (M \| T)$
    for $j \in [1, n]$ :
        $C_j^{(\mathsf{fwl})} := k_j^{(\mathsf{fwl})} \star x_{m_j}$
    for $j \in [1, t]$ :
        $C_{n+j}^{(\mathsf{fwl})} := y_j$
    for $\ell \in [\mathsf{fwl} + 1, \mathsf{e}]$ :
        $C_j^{(\ell)} := \Delta_j^{(\ell)} \star C_j^{(\ell-1)}$
$\mathcal{L} := \mathcal{L} \cup \{(\mathsf{c}, M', C^{(\mathsf{e})}, \mathsf{e}, \tau)\}$
return $C^{(\mathsf{e})}$

Oracle H($M, x_0, x_1$)
if $\mathsf{H}[M, x_0, x_1] = \bot$ :
    $\mathsf{q} := \mathsf{q} + 1$
    if $\mathsf{q} = q^*$ :
        $\gamma \leftarrow \textsc{Chall}(x_0, x_1)$
        $\mathsf{H}[M, x_0, x_1] := \gamma$
    else :
        $\beta \xleftarrow{\$} \{0, 1\}^t$
        $\mathsf{H}[M, x_0, x_1] := \beta$
return $\mathsf{H}[M, x_0, x_1]$

Oracle CorrToken($\hat{\mathsf{e}}$)
if $\hat{\mathsf{e}} > \mathsf{e}$ : return $\bot$
if $\hat{\mathsf{e}} = \mathsf{fwl}$ : abort    //wrong guess for $\mathsf{fwl}$
return $\Delta^{(\hat{\mathsf{e}})}$
```

Fig. 10. The reduction $\mathcal{B}$ simulating game $\mathsf{G}^{\mathsf{INT\text{-}CTXT}}_{\mathsf{COM\text{-}UE}}$ and event E_1. The set of oracles that $\mathcal{A}$ has access to is defined as $\mathcal{O} = \{\textsc{Enc}, \textsc{Try}, \textsc{Next}, \textsc{Upd}, \textsc{CorrKey}, \textsc{CorrToken}, \mathrm{H}\}$. † indicates that the computation of $\Delta^{(0)}$ is skipped.

queries to oracle $\textsc{Eval}$*. Then, there exists an adversary* $\mathcal{B}$ *against* DLAI *for* $m = 2q_{\textsc{Eval}}(t+1)$ *such that*

$$\mathsf{Adv}^{\mathsf{Multi\text{-}St\text{-}UP}}_{\mathsf{EGA}}(\mathcal{A}) \leq \frac{1}{1-(3/4)^{t-1}} \cdot \mathsf{Adv}^{\mathsf{DLAI}}_{\mathsf{EGA}}(\mathcal{B})$$

and the running time of $\mathcal{B}$ *is about that of* $\mathcal{A}$*.*

Remark 7. Recall that in $\mathsf{COM\text{-}UE}$ the parameter t represents the output length of the random oracle H. Therefore, the tightness loss of our reduction is close to 1 for cryptographically sized parameters.

Proof. Let $q_{\textsc{Eval}}$ be the number of queries that $\mathcal{A}$ issues to the $\textsc{Eval}$ oracle. Recall that the DLAI challenge consists of $m+1$ many set elements $(z_i)_{i\in[0,m]}$ with $z_i = h^i \star \tilde{x}$, where $m = 2q_{\textsc{Eval}}(t+1)$ and t comes from the $\mathsf{Multi\text{-}St\text{-}UP}$ challenge. To simplify the proof and its notation, reduction $\mathcal{B}$ will not rerandomize set elements that it outputs to $\mathcal{A}$. Furthermore, we assume that adversary $\mathcal{A}$ also does not rerandomize elements it outputs. In the AGAM these simplifications are without loss of generality as we discuss in Supplementary Material E.1.

```
Adversary $\mathcal{B}(z_0,\dots,z_m)$
$c := 0$
$\gamma \xleftarrow{\$} \{0,1\}^t$
$y_1,\dots,y_t \leftarrow \mathcal{A}^{\textsc{Eval},\textsc{Chall}}$
$h := \mathsf{Extract}(y_1,\dots,y_t)$
return $h$

Oracle $\textsc{Chall}(\tilde{x}_0,\tilde{x}_1)$        // $\tilde{x}_0 \prec_{\mathsf{lex}} \tilde{x}_1$
return $\gamma$

Oracle $\textsc{Eval}$
$\beta \xleftarrow{\$} \{0,1\}^t$
$i_0 := 1 + 2c\cdot(t+1)$
$i_1 := 1 + (2c+1)\cdot(t+1)$
$x_0 := z_{i_0}$
$x_1 := z_{i_1}$        // w.l.o.g. $x_0 \prec_{\mathsf{lex}} x_1$
$c := c+1$
for $k \in [t]$
    $\nu := k + i_{\beta_k}$
    $y_k := z_\nu$
return $(\beta, x_0, x_1, y_1,\dots,y_t)$
```

Fig. 11. Adversary $\mathcal{B}$ for the proof of Proposition 3, where $m = 2q_{\textsc{Eval}}(t+1)$. The $\mathsf{Extract}$ algorithm is described in the main body of the proof. Recall that we do not explicitly rerandomize elements as discussed in Supplementary Material E.1.

Consider the reduction depicted in Figure 11. Here, reduction $\mathcal{B}$ implicitly embeds the group elements $(h, h^2, \dots, h^t)$ into the secret key $(g_1,\dots,g_t)$ as follows. Assume $c = 0$, i. e. the $\textsc{Eval}$ oracle is queried for the first time. $\mathcal{B}$ first samples a random $\beta \xleftarrow{\$} \{0,1\}^t$, computes the indices $i_0 = 1$ and $i_1 = t+2$ and subsequently defines

$$\begin{aligned} x_0 &= z_1 &&= h \star \tilde{x} \ , \\ x_1 &= z_{t+2} &&= h^{t+2} \star \tilde{x} \ . \end{aligned}$$

If the i-th bit $\beta_i = 0$, then $\mathcal{B}$ has to compute the element

$$h^i \star x_0 = h^i \star z_1 = h^i \cdot h \star \tilde{x} = z_{1+i} \ .$$

Query	Bitstring	Basis	h^1	h^2	h^3	h^4	h^5
EVAL	$\beta = 10010$	$x_0 = z_1$		$\enclose{circle}{z_3}$	z_4		z_6
		$x_1 = z_7$	z_8			$\enclose{circle}{z_{11}}$	
EVAL	$\beta = 01110$	$x_0 = z_{13}$	z_{14}				z_{18}
		$x_1 = z_{19}$		z_{21}	z_{22}	z_{23}	
CHALL	$\gamma = 10100$	$\tilde{x}_0 = z_3$		$\boxed{z_5}$		z_7	z_8
		$\tilde{x}_1 = z_{11}$	$\boxed{z_{12}}$		z_{14}		
EVAL	$\beta = 00101$	$x_0 = z_{25}$	z_{26}	z_{27}		z_{29}	
		$x_1 = z_{31}$			z_{34}		z_{36}

Table 2. Visualization of the proof. The individual rows of the table represent different queries to the oracles. The elements h^1 through h^5 represent the secret key elements. The set elements in the main body of the table represent the elements that EVAL outputs for each query. These elements together with the basis elements are all the elements $\mathcal{A}$ receives throughout its execution. The encircled elements are those elements that $\mathcal{A}$ chooses as its base elements $\tilde{x}_0, \tilde{x}_1$. The set elements in the challenge row represent the elements $y_1, \ldots, y_t$ that $\mathcal{A}$ has to produce. The boxed elements are set elements that $\mathcal{A}$ has not yet seen and therefore their representation must contain a power of h.

This amounts to a simple lookup of the element z_{i+1} that $\mathcal{B}$ was provided by the DLAI assumption. This method can be generalized to the case $\beta_i = 1$ and $c > 0$ (see lines 08-15 in Figure 11). Therefore $\mathcal{B}$ successfully embeds $(h, h^2, \ldots, h^t)$ into the secret key. See also Table 2 which further illustrates this idea for $t = 5$.

Eventually, say after the c-th query, $\mathcal{A}$ calls CHALL on two basis elements $(\tilde{x}_0, \tilde{x}_1)$. Since we assume $\mathcal{A}$ to be algebraic and to not rerandomize $(\tilde{x}_0, \tilde{x}_1)$ we can write

$$\tilde{x}_0 = z_{j_0}\ ,\ \tilde{x}_1 = z_{j_1}\ , \qquad j_0, j_1 \in [0, 2c \cdot (t+1)]\ .$$

$\mathcal{B}$ then returns a random $\gamma \xleftarrow{\$} \{0,1\}^t$ and continues simulating the EVAL oracle.

THE EXTRACT ALGORITHM. At some point $\mathcal{A}$ responds with the set elements $(y_1, \ldots, y_t)$. Since we assume $\mathcal{A}$ to be successful, we have that

$$y_i = h^i \star \tilde{x}_{\gamma_i} = \begin{cases} h^i \star z_{j_0}, & \text{if } \gamma_i = 0 \\ h^i \star z_{j_1}, & \text{if } \gamma_i = 1 \end{cases}$$

for all $i \in [1, t]$. However, since $\mathcal{A}$ is algebraic it must provide a representation (s, v) for each $y_i = h^i \star z_{j_*}$ relative to a previously received set element (which form a subset of $\{z_0, \ldots, z_m\}$). There are now two cases:

1. $\mathcal{A}$ has previously received $h^i \star z_{j_*} = z_{j_*+i}$ for each $i \in [1, t]$. In this case the representation could be trivial (up to rerandomization), i. e. $(s, v) = (1, z_{j_*+i})$.
2. $\mathcal{A}$ has not previously received $h^i \star z_{j_*}$ for at least one $i \in [1, t]$. In this case the corresponding representation (s, v) cannot be trivial. More concretely, if

$v = z_\ell$ then we must have $s = h^{j_* + i - \ell} \neq 1$ (again up to rerandomization). Call this event E.

Of course, $\mathcal{B}$ can only extract h in the latter case by taking an appropriate root of s. As we will argue next, however, this case is extremely likely.

Claim. $\Pr[E] \geq 1 - \left(\frac{3}{4}\right)^{t-1}$.

We lower bound the probability by splitting it up into two disjoint events which we analyze separately:

$$\begin{aligned}\Pr[E] = &\Pr[E \mid j_0, j_1 > m - t] \cdot \Pr[j_0, j_1 > m - t] + \qquad (5)\\ &\Pr[E \mid j_0 \leq m - t \vee j_1 \leq m - t] \cdot \Pr[j_0 \leq m - t \vee j_1 \leq m - t]\\ \geq &\min\left\{\Pr[E \mid j_0, j_1 > m - t], \Pr[E \mid j_0 \leq m - t \vee j_1 \leq m - t]\right\} \ . \qquad (6)\end{aligned}$$

Case 1. First assume that $j_0, j_1 > m - t$. Then

$$y_t = h^t \star z_{j_b} = h^t \cdot h^{j_b} \star \tilde{x} \notin \{z_0, \ldots, z_m\} \ , \quad b \in \{0, 1\} \ ,$$

because $t + j_b > m$. In that case, the representation (s, v) for y_t is never trivial, i. e. $y_t = s \star v$ with $s = h^\kappa$ for some $\kappa > 0$ and $v = z_\ell$ for some $\ell \in [0, m]$. Therefore we have

$$\Pr[E \mid j_0, j_1 > m - t] = 1$$

and in fact Equation (6) reduces to

$$\Pr[E] \geq Pr[E \mid j_0 \leq m - t \vee j_1 \leq m - t] \ .$$

Case 2. Now assume that (w.l.o.g.) $j_0 \leq m - t$. If $\gamma_i = 0$ for an index $i \in [1, t]$ then by definition the adversary $\mathcal{A}$ has to produce the element $z_{j_0 + i}$ and an accompanying representation. Clearly, this representation can only be trivial if $\mathcal{A}$ has received the element z_{j_0+i} from the Eval oracle before.

Concretely, if we let $\varDelta = (\delta_1, \ldots, \delta_t) \in \{0, 1\}^t$ be defined via

$$\delta_i = \begin{cases} 1 & \text{if } z_{j_0+i} \text{ was output by } \textsc{Eval} \\ 0 & \text{else} \end{cases}$$

then the previous paragraph can be rephrased as

$$\Pr[E] \geq \Pr[\exists i \in [1, t] : \delta_i = 0 \wedge \gamma_i = 0] \ .$$

For instance, in the running example in Table 2 we have $j_0 = 3$ and thus $\varDelta = (10111)$.

Claim. δ_i is a random bit if $j_0 + i \neq 1 + k \cdot (t + 1)$ for $k \in \mathbb{N}$. In particular, the β_i and γ_i are independent.

Essentially, the claim says that δ_i is a random bit except for those i where z_{j_0+i} appeared as one of the basis elements returned by the EVAL oracle (in which case we always have $\delta_i = 1$). Note that there can be at most one such index i in the interval $[j_0 + 1, \ldots, j_0 + t]$. For all other indices we have that δ_i essentially depends on the bit string β that was chosen for the corresponding EVAL query. Since each β is uniformly random, so are the δ_i.

Concretely, for each $i \in [1, t]$ the corresponding δ_i satisfies the following equality: Assume that z_{j_0} was returned in the c-th query to EVAL, which we write explicitly as $(\beta^{(c)}, x_0^{(c)}, x_1^{(c)}, y_1^{(c)}, \ldots, y_t^{(c)})$. We then have $x_0^{(c)} = z_\mu$, $x_1^{(c)} = z_{\mu+t+1}$ and $z_{j_0} = z_{\mu+\eta}$ for some appropriate indices $\mu \leq m - t$ and $0 \leq \eta \leq 2t + 1$. Furthermore, let $\beta^{(c+1)}$ be the bitstring returned in the subsequent EVAL query. We now have

$$\delta_i = \begin{cases} [\![\beta^{(c)}_{i+\eta} = 0]\!], & \text{if } i + \eta < t + 1 \\ 1, & \text{if } i + \eta = t + 1 \\ [\![\beta^{(c)}_{i+\eta-t-1} = 0]\!], & \text{if } t + 1 < i + \eta < 2t + 2 \\ 1 & \text{if } i + \eta = 2t + 2 \\ [\![\beta^{(c+1)}_{i+\eta-2t-3} = 0]\!], & \text{if } i + \eta > 2t + 2 \end{cases} .$$

Evidently, apart from two special cases in which $i+\eta = k \cdot (t+1)$ for some $k \in \mathbb{N}$, the δ_i only depend on the bitstrings $\beta^{(c)}, \beta^{(c+1)}$ and not on the challenge γ. In summary, we therefore have that

$$\Pr[E] \geq 1 - \Pr[\forall i \in [1, t] : \delta_i = 1 \vee \gamma_i = 1] \geq 1 - \left(\frac{3}{4}\right)^{t-1} .$$

Lastly, $\mathcal{B}$ can extract h from the non-trivial representation by first computing the indices j_b and ℓ. This can be done by merely comparing z_{j_b} and z_ℓ to each element z_i that $\mathcal{B}$ received from the DLAI assumption. $\mathcal{B}$ can then compute $\kappa = (j_b + t - \ell)$ and thus compute $h = s^{1/\kappa}$. □

Remark 8. The assumption that $x_0 \prec_{\mathsf{lex}} x_1$ in Figure 11, line 13 is without loss of generality as reduction $\mathcal{B}$ can simply swap the elements x_0, x_1 if necessary. It then has to keep track of this change throughout the reduction, but this does not fundamentally change the proof. Also keep in mind that the elements x_0, x_1 are implicitly rerandomized, which means that the distribution of these elements is still correct even after a swap.

Remark 9. The proof also applies to a group action that supports *twists*. That is, a group action that comes equipped with an efficient algorithm that on input $y = g \star \tilde{x}$ computes $y^{\mathsf{T}} = g^{-1} \star \tilde{x}$. However, in that case the probability of the event E changes to

$$\Pr[E] \geq 1 - \left(\frac{3}{4}\right)^{\lfloor t/2 \rfloor - 1} .$$

This is due to the fact that, given a set element y, the adversary knows the representation of *two* set elements instead of only one (namely y and y^{T}). This way we can only argue for $\lfloor \frac{t}{2} \rfloor - 1$ many independent random bits δ_i, decreasing the overall probability of the event E.

Corollary 1. *Let* COM-UE *be the scheme described in Section 4.2. For any algebraic adversary $\mathcal{A}$ against the* detIND-UE-CCA *security of* COM-UE*, there exist adversaries $\mathcal{B}, \mathcal{C}$ against* Wk-PR *and* DLAI *in the random oracle model such that*

$$\mathsf{Adv}^{\mathsf{detIND\text{-}UE\text{-}CCA}}_{\mathsf{COM\text{-}UE}}(\mathcal{A}) \leq 2\ell(n_{\mathsf{e}}+1)^3 \cdot \mathsf{Adv}^{\mathsf{Wk\text{-}PR}}_{\mathsf{EGA}}(\mathcal{B}) + \frac{(q+n_E)(n_{\mathsf{e}}+1)}{1-(3/4)^{t-1}} \cdot \mathsf{Adv}^{\mathsf{DLAI}}_{\mathsf{EGA}}(\mathcal{C}) + \frac{1}{2^t} ,$$

where $\ell = n + t$ is the length of a message and tag in bits, n_{e} is the number of epochs, n_E is the number of encryption queries and q is the number of queries to the random oracle.

Acknowledgments. We would like to thank Andre Esser, Eike Kiltz, Sabrina Kunzweiler and Edoardo Persichetti for interesting research discussions. The work of Jonas Meers and Doreen Riepel was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2092 CASA - 390781972. Doreen Riepel was further supported in part by Bellare's KACST grant.

References

1. Abdalla, M., Eisenhofer, T., Kiltz, E., Kunzweiler, S., Riepel, D.: Password-authenticated key exchange from group actions. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part II. LNCS, vol. 13508, pp. 699–728. Springer, Heidelberg (Aug 2022). https://doi.org/10.1007/978-3-031-15979-4_24
2. Alamati, N., De Feo, L., Montgomery, H., Patranabis, S.: Cryptographic group actions and applications. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part II. LNCS, vol. 12492, pp. 411–439. Springer, Heidelberg (Dec 2020). https://doi.org/10.1007/978-3-030-64834-3_14
3. Alamati, N., Montgomery, H., Patranabis, S.: Symmetric primitives with structured secrets. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part I. LNCS, vol. 11692, pp. 650–679. Springer, Heidelberg (Aug 2019). https://doi.org/10.1007/978-3-030-26948-7_23
4. Baghery, K., Cozzo, D., Pedersen, R.: An isogeny-based ID protocol using structured public keys. In: Paterson, M.B. (ed.) 18th IMA International Conference on Cryptography and Coding. LNCS, vol. 13129, pp. 179–197. Springer, Heidelberg (Dec 2021). https://doi.org/10.1007/978-3-030-92641-0_9
5. Berti, F., Pereira, O., Peters, T.: Reconsidering generic composition: The tag-then-encrypt case. In: Chakraborty, D., Iwata, T. (eds.) INDOCRYPT 2018. LNCS, vol. 11356, pp. 70–90. Springer, Heidelberg (Dec 2018). https://doi.org/10.1007/978-3-030-05378-9_4

6. Beullens, W., Kleinjung, T., Vercauteren, F.: CSI-FiSh: Efficient isogeny based signatures through class group computations. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part I. LNCS, vol. 11921, pp. 227–247. Springer, Heidelberg (Dec 2019). https://doi.org/10.1007/978-3-030-34578-5_9
7. Boneh, D., Eskandarian, S., Kim, S., Shih, M.: Improving speed and security in updatable encryption schemes. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part III. LNCS, vol. 12493, pp. 559–589. Springer, Heidelberg (Dec 2020). https://doi.org/10.1007/978-3-030-64840-4_19
8. Boneh, D., Guan, J., Zhandry, M.: A lower bound on the length of signatures based on group actions and generic isogenies. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part V. LNCS, vol. 14008, pp. 507–531. Springer, Heidelberg (Apr 2023). https://doi.org/10.1007/978-3-031-30589-4_18
9. Boneh, D., Lewi, K., Montgomery, H.W., Raghunathan, A.: Key homomorphic PRFs and their applications. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 410–428. Springer, Heidelberg (Aug 2013). https://doi.org/10.1007/978-3-642-40041-4_23
10. Booher, J., Bowden, R., Doliskani, J., Fouotsa, T.B., Galbraith, S.D., Kunzweiler, S., Merz, S.P., Petit, C., Smith, B., Stange, K.E., Ti, Y.B., Vincent, C., Voloch, J.F., Weitkämper, C., Zobernig, L.: Failing to hash into supersingular isogeny graphs. Cryptology ePrint Archive, Report 2022/518 (2022), https://eprint.iacr.org/2022/518
11. Boyd, C., Davies, G.T., Gjøsteen, K., Jiang, Y.: Fast and secure updatable encryption. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part I. LNCS, vol. 12170, pp. 464–493. Springer, Heidelberg (Aug 2020). https://doi.org/10.1007/978-3-030-56784-2_16
12. Castryck, W., Decru, T.: CSIDH on the surface. In: Ding, J., Tillich, J.P. (eds.) Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020. pp. 111–129. Springer, Heidelberg (2020). https://doi.org/10.1007/978-3-030-44223-1_7
13. Castryck, W., Decru, T.: An efficient key recovery attack on SIDH. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part V. LNCS, vol. 14008, pp. 423–447. Springer, Heidelberg (Apr 2023). https://doi.org/10.1007/978-3-031-30589-4_15
14. Castryck, W., Houben, M., Vercauteren, F., Wesolowski, B.: On the decisional diffie-hellman problem for class group actions on oriented elliptic curves. Cryptology ePrint Archive, Report 2022/345 (2022), https://eprint.iacr.org/2022/345
15. Castryck, W., Lange, T., Martindale, C., Panny, L., Renes, J.: CSIDH: An efficient post-quantum commutative group action. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part III. LNCS, vol. 11274, pp. 395–427. Springer, Heidelberg (Dec 2018). https://doi.org/10.1007/978-3-030-03332-3_15
16. Chandran, N., Chase, M., Liu, F.H., Nishimaki, R., Xagawa, K.: Re-encryption, functional re-encryption, and multi-hop re-encryption: A framework for achieving obfuscation-based security and instantiations from lattices. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 95–112. Springer, Heidelberg (Mar 2014). https://doi.org/10.1007/978-3-642-54631-0_6
17. Chen, H., Galteland, Y.J., Liang, K.: Cca-1 secure updatable encryption with adaptive security. Springer-Verlag (2023)
18. Chen, L., Li, Y., Tang, Q.: CCA updatable encryption against malicious re-encryption attacks. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part III. LNCS, vol. 12493, pp. 590–620. Springer, Heidelberg (Dec 2020). https://doi.org/10.1007/978-3-030-64840-4_20

19. Cini, V., Ramacher, S., Slamanig, D., Striecks, C., Tairi, E.: Updatable signatures and message authentication codes. In: Garay, J. (ed.) PKC 2021, Part I. LNCS, vol. 12710, pp. 691–723. Springer, Heidelberg (May 2021). https://doi.org/10.1007/978-3-030-75245-3_25
20. D'Alconzo, G., Scala, A.J.D.: Representations of group actions and their applications in cryptography. Cryptology ePrint Archive, Paper 2023/1247 (2023), https://eprint.iacr.org/2023/1247, https://eprint.iacr.org/2023/1247
21. Davidson, A., Deo, A., Lee, E., Martin, K.: Strong post-compromise secure proxy re-encryption. In: Jang-Jaccard, J., Guo, F. (eds.) ACISP 19. LNCS, vol. 11547, pp. 58–77. Springer, Heidelberg (Jul 2019). https://doi.org/10.1007/978-3-030-21548-4_4
22. De Feo, L., Fouotsa, T.B., Kutas, P., Leroux, A., Merz, S.P., Panny, L., Wesolowski, B.: SCALLOP: Scaling the CSI-FiSh. In: Boldyreva, A., Kolesnikov, V. (eds.) PKC 2023, Part I. LNCS, vol. 13940, pp. 345–375. Springer, Heidelberg (May 2023). https://doi.org/10.1007/978-3-031-31368-4_13
23. De Feo, L., Galbraith, S.D.: SeaSign: Compact isogeny signatures from class group actions. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part III. LNCS, vol. 11478, pp. 759–789. Springer, Heidelberg (May 2019). https://doi.org/10.1007/978-3-030-17659-4_26
24. Duman, J., Hartmann, D., Kiltz, E., Kunzweiler, S., Lehmann, J., Riepel, D.: Generic models for group actions. In: Boldyreva, A., Kolesnikov, V. (eds.) PKC 2023, Part I. LNCS, vol. 13940, pp. 406–435. Springer, Heidelberg (May 2023). https://doi.org/10.1007/978-3-031-31368-4_15
25. Eaton, E., Jao, D., Komlo, C., Mokrani, Y.: Towards post-quantum key-updatable public-key encryption via supersingular isogenies. In: AlTawy, R., Hülsing, A. (eds.) SAC 2021. LNCS, vol. 13203, pp. 461–482. Springer, Heidelberg (Sep / Oct 2022). https://doi.org/10.1007/978-3-030-99277-4_22
26. Everspaugh, A., Paterson, K.G., Ristenpart, T., Scott, S.: Key rotation for authenticated encryption. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part III. LNCS, vol. 10403, pp. 98–129. Springer, Heidelberg (Aug 2017). https://doi.org/10.1007/978-3-319-63697-9_4
27. Fabrega, A., Maurer, U., Mularczyk, M.: A fresh approach to updatable symmetric encryption. Cryptology ePrint Archive, Report 2021/559 (2021), https://eprint.iacr.org/2021/559
28. Galteland, Y.J., Pan, J.: Backward-leak uni-directional updatable encryption from (homomorphic) public key encryption. In: Boldyreva, A., Kolesnikov, V. (eds.) PKC 2023, Part II. LNCS, vol. 13941, pp. 399–428. Springer, Heidelberg (May 2023). https://doi.org/10.1007/978-3-031-31371-4_14
29. Jiang, Y.: The direction of updatable encryption does not matter much. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part III. LNCS, vol. 12493, pp. 529–558. Springer, Heidelberg (Dec 2020). https://doi.org/10.1007/978-3-030-64840-4_18
30. Klooß, M., Lehmann, A., Rupp, A.: (R)CCA secure updatable encryption with integrity protection. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part I. LNCS, vol. 11476, pp. 68–99. Springer, Heidelberg (May 2019). https://doi.org/10.1007/978-3-030-17653-2_3
31. Lee, E.: Improved security notions for proxy re-encryption to enforce access control. In: Lange, T., Dunkelman, O. (eds.) LATINCRYPT 2017. LNCS, vol. 11368, pp. 66–85. Springer, Heidelberg (Sep 2019). https://doi.org/10.1007/978-3-030-25283-0_4

32. Lehmann, A., Tackmann, B.: Updatable encryption with post-compromise security. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part III. LNCS, vol. 10822, pp. 685–716. Springer, Heidelberg (Apr / May 2018). https://doi.org/10.1007/978-3-319-78372-7_22
33. Leroux, A., Roméas, M.: Updatable encryption from group actions. Cryptology ePrint Archive, Report 2022/739 (2022), https://eprint.iacr.org/2022/739
34. Levy-dit-Vehel, F., Roméas, M.: A composable look at updatable encryption. Cryptology ePrint Archive, Report 2021/538 (2021), https://eprint.iacr.org/2021/538
35. Maino, L., Martindale, C., Panny, L., Pope, G., Wesolowski, B.: A direct key recovery attack on SIDH. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part V. LNCS, vol. 14008, pp. 448–471. Springer, Heidelberg (Apr 2023). https://doi.org/10.1007/978-3-031-30589-4_16
36. Miao, P., Patranabis, S., Watson, G.J.: Unidirectional updatable encryption and proxy re-encryption from DDH. In: Boldyreva, A., Kolesnikov, V. (eds.) PKC 2023, Part II. LNCS, vol. 13941, pp. 368–398. Springer, Heidelberg (May 2023). https://doi.org/10.1007/978-3-031-31371-4_13
37. Moriya, T., Onuki, H., Takagi, T.: SiGamal: A supersingular isogeny-based PKE and its application to a PRF. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part II. LNCS, vol. 12492, pp. 551–580. Springer, Heidelberg (Dec 2020). https://doi.org/10.1007/978-3-030-64834-3_19
38. Mula, M., Murru, N., Pintore, F.: Random sampling of supersingular elliptic curves. Cryptology ePrint Archive, Report 2022/528 (2022), https://eprint.iacr.org/2022/528
39. Namprempre, C., Rogaway, P., Shrimpton, T.: Reconsidering generic composition. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 257–274. Springer, Heidelberg (May 2014). https://doi.org/10.1007/978-3-642-55220-5_15
40. Naor, M., Pinkas, B., Reingold, O.: Distributed pseudo-random functions and KDCs. In: Stern, J. (ed.) EUROCRYPT'99. LNCS, vol. 1592, pp. 327–346. Springer, Heidelberg (May 1999). https://doi.org/10.1007/3-540-48910-X_23
41. Nishimaki, R.: The direction of updatable encryption does matter. In: Hanaoka, G., Shikata, J., Watanabe, Y. (eds.) PKC 2022, Part II. LNCS, vol. 13178, pp. 194–224. Springer, Heidelberg (Mar 2022). https://doi.org/10.1007/978-3-030-97131-1_7
42. Page, A., Robert, D.: Introducing clapoti(s): Evaluating the isogeny class group action in polynomial time. Cryptology ePrint Archive, Paper 2023/1766 (2023), https://eprint.iacr.org/2023/1766, https://eprint.iacr.org/2023/1766
43. Phong, L.T., Wang, L., Aono, Y., Nguyen, M.H., Boyen, X.: Proxy re-encryption schemes with key privacy from LWE. Cryptology ePrint Archive, Report 2016/327 (2016), https://eprint.iacr.org/2016/327
44. Polyakov, Y., Rohloff, K., Sahu, G., Vaikuntanathan, V.: Fast proxy re-encryption for publish/subscribe systems. ACM Trans. Priv. Secur. **20**(4) (sep 2017). https://doi.org/10.1145/3128607
45. Qian, C., Galteland, Y.J., Davies, G.T.: Extending updatable encryption: Public key, tighter security and signed ciphertexts. Cryptology ePrint Archive, Paper 2023/848 (2023), https://eprint.iacr.org/2023/848
46. Robert, D.: Breaking SIDH in polynomial time. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part V. LNCS, vol. 14008, pp. 472–503. Springer, Heidelberg (Apr 2023). https://doi.org/10.1007/978-3-031-30589-4_17
47. Shannon, C.E.: Communication theory of secrecy systems. Bell Systems Technical Journal **28**(4), 656–715 (1949)

48. Slamanig, D., Striecks, C.: Revisiting updatable encryption: Controlled forward security, constructions and a puncturable perspective. TCC 2023 (2023)

Supplementary Material

A Restricted Effective Group Actions and Twisting

In practice, the requirements from the definition of EGA are often too strong. Therefore we will consider the weaker notion of restricted effective group actions.

Definition 12 (Restricted Effective Group Action). *Let $(\mathcal{G}, \mathcal{X}, \star)$ be a group action and let $\boldsymbol{g} = (g_1, ..., g_\nu)$ be a generating set for $\mathcal{G}$. Assume that the following properties are satisfied:*

1. *The group $\mathcal{G}$ is finite and $\nu = \mathrm{poly}(\log(\#\mathcal{G}))$.*
2. *The set $\mathcal{X}$ is finite and there exist efficient algorithms for membership testing and to compute a unique representation.*
3. *There exists a distinguished element $\tilde{x} \in \mathcal{X}$ with known representation.*
4. *There exists an efficient algorithm that given $g_i \in \boldsymbol{g}$ and $x \in \mathcal{X}$, outputs $g_i \star x$ and $g_i^{-1} \star x$.*

Then we call $(\mathcal{G}, \mathcal{X}, \star, \tilde{x})$ a restricted effective group action (REGA).

Popular group actions like CSIDH [15] support so-called *twists*. We recall the definition of a twist in the group action setting.

Definition 13 (Effective Group Action with Twists). *Let $(\mathcal{G}, \mathcal{X}, \star, \tilde{x})$ be an* EGA. *We call it an Effective Group Action with Twists* (EGAT) *if there is an efficient algorithm that, given $y = g \star \tilde{x}$, computes $y^{\mathsf{T}} = g^{-1} \star \tilde{x}$.*

B Algorithms to Compute Sets $\mathcal{L}^*$ and $\tilde{\mathcal{L}}^*$

In Figure 12, we provide the description of algorithms to update sets $\tilde{\mathcal{L}}^*$ and $\mathcal{L}^*$ which are used to prevent trivial wins in the detIND-UE-CCA and INT-CTXT games.

```
Algorithm Update$\tilde{\mathcal{L}}^*$
if CHALL or UPD$\tilde{\mathrm{C}}$ happens:
  $\tilde{\mathcal{L}}^* := \tilde{\mathcal{L}}^* \cup \{(\tilde{C}, \cdot)\}$
if chall = 1 and CORRTOKEN happens:
  for $i \in \mathcal{T}^*$ and $(\tilde{C}_{i-1}, i-1) \in \tilde{\mathcal{L}}^*$
    $\tilde{\mathcal{L}}^* := \tilde{\mathcal{L}}^* \cup \{(\tilde{C}, i)\}$

Algorithm Update$\mathcal{L}^*$
if ENC or UPD happens:
  $\mathcal{L}^* := \mathcal{L}^* \cup \{(\cdot, C, \cdot)\}$
if CORRTOKEN happens:
  for $i \in \mathcal{T}^*$
    for $(j, C_{i-1}, i-1) \in \mathcal{L}^*$
      $C_i := \mathsf{Upd}(\Delta^{(i)}, C_{i-1})$
      $\mathcal{L}^* := \mathcal{L}^* \cup \{(j, C_i, i)\}$
```

Fig. 12. Algorithms to update sets $\tilde{\mathcal{L}}^*$ and $\tilde{\mathcal{L}}^*$ used in games in Figures 2 and 3.

C Full Proof of Theorem 2

Theorem 2. *Let* $\mathsf{BIN\text{-}UE}$ *be the scheme described in Figure 4. For any adversary* $\mathcal{A}$ *against the* $\mathsf{detIND\text{-}UE\text{-}CPA}$ *security of* $\mathsf{BIN\text{-}UE}$*, there exists an adversary* $\mathcal{B}$ *against* $\mathsf{Wk\text{-}PR}$ *such that*

$$\mathsf{Adv}^{\mathsf{detIND\text{-}UE\text{-}CPA}}_{\mathsf{BIN\text{-}UE}}(\mathcal{A}) \leq 2n(n_{\mathsf{e}}+1)^3 \cdot \mathsf{Adv}^{\mathsf{Wk\text{-}PR}}_{\mathsf{EGA}}(\mathcal{B}) \ ,$$

where n_{e} *is the number of epochs and* n *is the length of a message in bits. The running time of* $\mathcal{B}$ *is about that of* $\mathcal{A}$*.*

Proof. For simplicity, we reduce the $\mathsf{detIND\text{-}UE\text{-}CPA}$ security of $\mathsf{BIN\text{-}UE}$ to the $\mathsf{Multi\text{-}St\text{-}PR}$ assumption. The theorem then follows from Proposition 2.

THE HYBRIDS. We partition the non-corrupted key space as follows: $\{0,\ldots,n_{\mathsf{e}}\} \setminus \mathcal{K}^* = \bigcup_{(i,\mathsf{fwl}_i,\mathsf{fwr}_i)\in\mathcal{FW}}\{\mathsf{fwl}_i,\mathsf{fwr}_i\}$ where fwl_i and fwr_i are the firewalls of the i-th insulated region. For $b \in \{0,1\}$ we define the game G^b_i as $\mathrm{G}^{\mathsf{detIND\text{-}UE\text{-}CPA}\text{-}b}_{\mathsf{BIN\text{-}UE}}(\mathcal{A})$ except for:

1. The game picks $\mathsf{fwl}_i,\mathsf{fwr}_i \xleftarrow{\$} \{0,\ldots,n_{\mathsf{e}}\}$ randomly and if they are not the i-th firewall, the game aborts and returns a random bit b'. This loss is upper-bounded by $(n_{\mathsf{e}}+1)^2$.
2. For the challenge $(\bar{M},\bar{C})$ made in epoch $\tilde{\mathsf{e}}$ the game returns an update of $\bar{C}$ if $\tilde{\mathsf{e}} < \mathsf{fwl}_i$ and an encryption of $\bar{M}$ if $\tilde{\mathsf{e}} > \mathsf{fwr}_i$. Lastly, if $\mathsf{fwl}_i \leq \tilde{\mathsf{e}} \leq \mathsf{fwr}_i$ the game returns an update of $\tilde{C}$ if $b=1$ and an encryption of $\tilde{M}$ otherwise.
3. After $\mathcal{A}$ outputs b' the game will simply output b' as well if no trivial win condition was triggered and no `ABORT` occurred.

If $\mathsf{fwl}_i,\mathsf{fwr}_i$ are the correct values, then G^0_1 is $\mathrm{G}^{\mathsf{detIND\text{-}UE\text{-}CPA\text{-}0}}_{\mathsf{BIN\text{-}UE}}$, that is, inside each firewall the challenge is an encryption of $\bar{M}$. Let $n_r \leq n_{\mathsf{e}}+1$ be the total number of insulated regions. Then $\mathrm{G}^1_{n_r}$ is exactly $\mathrm{G}^{\mathsf{detIND\text{-}UE\text{-}CPA\text{-}1}}_{\mathsf{BIN\text{-}UE}}$, i. e., inside each firewall the challenge is an update of $\bar{C}$. Let E be the event that the guess for $\mathsf{fwl}_i,\mathsf{fwr}_i$ is correct. Subsequently we have for all $i \in [1,n_{\mathsf{e}}+1]$ and $b \in \{0,1\}$ that $\Pr[\mathrm{G}^b_i(\mathcal{A}) \Rightarrow 1 | \neg E] = 1/2$. Therefore

$$\begin{aligned}\Pr[\mathrm{G}^1_{n_r}(\mathcal{A}) \Rightarrow 1] &= \Pr[\mathrm{G}^1_{n_r}(\mathcal{A}) \Rightarrow 1|E]\cdot\Pr[E] + \Pr[\mathrm{G}^1_{n_r}(\mathcal{A}) \Rightarrow 1|\neg E]\cdot\Pr[\neg E]\\ &= \Pr[\mathrm{G}^{\mathsf{detIND\text{-}UE\text{-}CPA\text{-}1}}_{\mathsf{BIN\text{-}UE}}(\mathcal{A}) \Rightarrow 1]\cdot\frac{1}{(n_{\mathsf{e}}+1)^2} + \frac{1}{2}\left(1-\frac{1}{(n_{\mathsf{e}}+1)^2}\right)\\ \Pr[\mathrm{G}^0_1(\mathcal{A}) \Rightarrow 1] &= \Pr[\mathrm{G}^0_1(\mathcal{A}) \Rightarrow 1|E]\cdot\Pr[E] + \Pr[\mathrm{G}^0_1(\mathcal{A}) \Rightarrow 1|\neg E]\cdot\Pr[\neg E]\\ &= \Pr[\mathrm{G}^{\mathsf{detIND\text{-}UE\text{-}CPA\text{-}0}}_{\mathsf{BIN\text{-}UE}}(\mathcal{A}) \Rightarrow 1]\cdot\frac{1}{(n_{\mathsf{e}}+1)^2} + \frac{1}{2}\left(1-\frac{1}{(n_{\mathsf{e}}+1)^2}\right).\end{aligned}$$

Thus we get

$$|\Pr[\mathrm{G}^1_{n_r}(\mathcal{A}) \Rightarrow 1] - \Pr[\mathrm{G}^0_1(\mathcal{A}) \Rightarrow 1]| = \frac{1}{(n_{\mathsf{e}}+1)^2}\cdot\mathsf{Adv}^{\mathsf{detIND\text{-}UE\text{-}CPA}}_{\mathsf{BIN\text{-}UE}}(\mathcal{A}) \ .$$

Adversary $\mathcal{B}^{\textsc{Eval}}$

```
$b \xleftarrow{\$} \{0,1\}$
$k^{(0)} \leftarrow \mathsf{BIN\text{-}UE.KeyGen}$
$\Delta^{(0)} := \perp$
$\mathsf{e}, \mathsf{c}, \mathsf{twf}, \mathsf{chall} := 0$
$\mathcal{L}, \tilde{\mathcal{L}}, \mathcal{C}, \mathcal{K}, \mathcal{T} := \emptyset$
$\mathsf{fwl}_i, \mathsf{fwr}_i \xleftarrow{\$} [0, n_{\mathsf{e}}]$
for $j \in [\mathsf{fwl}_i - 1] \cup [\mathsf{fwr}_i + 1, n_{\mathsf{e}}]$ :
   $k^{(j)} \leftarrow \mathsf{BIN\text{-}UE.KeyGen}$
   $\Delta^{(j)} := \mathsf{BIN\text{-}UE.TokenGen}(k^{(j-1)}, k^{(j)})$  //†
for $j \in [\mathsf{fwl}_i + 1, \mathsf{fwr}_i]$ :
   $\Delta^{(j)} \xleftarrow{\$} \mathcal{G}^n$
$b' \leftarrow \mathcal{A}^{\mathcal{O}}$
if $\mathcal{K}^* \cap \mathcal{C}^* \neq \emptyset$ or $\mathcal{I}^* \cap \mathcal{C}^* \neq \emptyset$
or ABORT or $(i, \mathsf{fwl}_i, \mathsf{fwr}_i) \notin \mathcal{FW}$
   $b' \xleftarrow{\$} \{0,1\}$
return $[\![b = b']\!]$
```

Oracle $\textsc{Enc}(M)$

```
Let $M = (m_1, \ldots, m_n)$
$\mathsf{c} := \mathsf{c} + 1$
$\tau := (x_0, x_1, y_1, \ldots, y_n) \leftarrow \textsc{Eval}(M)$
if $\mathsf{e} \notin [\mathsf{fwl}_i, \mathsf{fwr}_i]$ :
   for $j \in [n]$: $C_j^{(\mathsf{e})} := k_j^{(\mathsf{e})} \star x_{m_j}$
else :
   for $j \in [n]$: $C_j^{(\mathsf{e})} := y_j$
   for $\ell \in [\mathsf{fwl}_i + 1, \mathsf{e}]$ :
      for $j \in [n]$: $C_j^{(\ell)} := \Delta_j^{(\ell)} \star C_j^{(\ell-1)}$
$C^{(\mathsf{e})} := (C_1^{(\mathsf{e})}, \ldots, C_n^{(\mathsf{e})})$
$\mathcal{L} := \mathcal{L} \cup \{(\mathsf{c}, M, C^{(\mathsf{e})}, \mathsf{e}, \tau)\}$
return $C^{(\mathsf{e})}$
```

Oracle $\textsc{Next}$

```
$\mathsf{e} := \mathsf{e} + 1$
```

Oracle $\textsc{CorrToken}(\hat{\mathsf{e}})$

```
if $\hat{\mathsf{e}} > \mathsf{e}$ return $\perp$
if $\hat{\mathsf{e}} \in \{\mathsf{fwl}, \mathsf{fwr} + 1\}$ : abort
$\mathcal{T} := \mathcal{T} \cup \{\hat{\mathsf{e}}\}$
return $\Delta^{(\hat{\mathsf{e}})}$
```

Oracle $\textsc{CorrKey}(\hat{\mathsf{e}})$

```
if $\hat{\mathsf{e}} > \mathsf{e}$ return $\perp$
if $\hat{\mathsf{e}} \in [\mathsf{fwl}, \mathsf{fwr}]$ : abort
$\mathcal{K} := \mathcal{K} \cup \{\hat{\mathsf{e}}\}$
return $k^{(\hat{\mathsf{e}})}$
```

Oracle $\textsc{Chall}(\bar{M}, \bar{C})$

```
if $\mathsf{chall} = 1$ return $\perp$                    //only once
$\mathsf{chall} := 1$
$\tilde{\mathsf{e}} := \mathsf{e}$
if $(\cdot, \cdot, \bar{C}, \mathsf{e} - 1, \tau') \notin \mathcal{L}$ :
   return $\perp$
if $b = 0$ :
   $\tau := (x_0, x_1, y_1, \ldots, y_n) \leftarrow \textsc{Eval}(\bar{M})$
   for $j \in [n]$ : $\tilde{C}_j^{(\mathsf{e})} := y_j$
else if $b = 1$ :
   Let $\tau' = (x'_0, x'_1, y'_1, \ldots, y'_n)$
   for $j \in [n]$ : $\tilde{C}_j^{(\mathsf{e})} := y'_j$
for $\ell \in [0, \mathsf{fwl}_i - 1]$ :
   for $j \in [n]$ :
      $\tilde{C}_j^{(\ell)} := \prod_{k=\ell+1}^{\mathsf{e}-2} (\Delta_j^{(k)})^{-1} \star \bar{C}_j$      //update
   $\tilde{C}^{(\ell)} := (\tilde{C}_1^{(\ell)}, \ldots, \tilde{C}_n^{(\ell)})$
for $\ell \in [\mathsf{fwl}_i + 1, \mathsf{fwr}_i]$ :
   for $j \in [n]$ :
      $\tilde{C}_j^{(\ell)} := \Delta_j^{(\ell)} \star \tilde{C}_j^{(\ell-1)}$                    //embed
   $\tilde{C}^{(\ell)} := (\tilde{C}_1^{(\ell)}, \ldots, \tilde{C}_n^{(\ell)})$
for $\ell \in [\mathsf{fwr}_i + 1, n_{\mathsf{e}}]$ :
   for $j \in [n]$ :
      $\tilde{C}_j^{(\ell)} := k_j^{(\ell)} \star x_{\beta_j}$          //fresh encryption
   $\tilde{C}^{(\ell)} := (\tilde{C}_1^{(\ell)}, \ldots, \tilde{C}_n^{(\ell)})$
$\tilde{\mathcal{L}} := \tilde{\mathcal{L}} \cup_{\mathsf{e}=0}^{n_{\mathsf{e}}} \{(\tilde{C}^{(\mathsf{e})}, \mathsf{e})\}$
return $\tilde{C}^{(\mathsf{e})}$
```

Oracle $\textsc{Upd}(C^{(\mathsf{e}-1)})$

```
if $\nexists\, \mathsf{c} : (\mathsf{c}, M, C^{(\mathsf{e}-1)}, \mathsf{e} - 1, \tau) \in \mathcal{L}$ :
   return $\perp$
Let $\tau = (x_0, x_1, y_1, \ldots, y_n)$
if $\mathsf{e} \notin [\mathsf{fwl}_i, \mathsf{fwr}_i]$ :
   Let $M = (m_1, \ldots, m_n)$
   $C_j^{(\mathsf{e})} := k_j^{(\mathsf{e})} \star x_{m_j}$
else :
   $C_j^{(\mathsf{fwl}_i)} := y_j$
   for $\ell \in [\mathsf{fwl}_i + 1, \mathsf{e}]$ :
      $C_j^{(\ell)} := \Delta_j^{(\ell)} \star C_j^{(\ell-1)}$
$C^{(\mathsf{e})} := (C_1^{(\mathsf{e})}, \ldots, C_n^{(\mathsf{e})})$
$\mathcal{L} := \mathcal{L} \cup \{(\mathsf{c}, M, C^{(\mathsf{e})}, \mathsf{e}, \tau)\}$
return $C_{\mathsf{e}}$
```

Oracle $\textsc{Upd}\tilde{\textsc{C}}$

```
if $\mathsf{chall} \neq 1$ return $\perp$
$\mathcal{C} := \mathcal{C} \cup \{\mathsf{e}\}$
find $(\tilde{C}^{(\mathsf{e})}, \mathsf{e}) \in \tilde{\mathcal{L}}$
return $\tilde{C}^{(\mathsf{e})}$
```

Fig. 13. The reduction $\mathcal{B}$ simulating game G_i^b. † indicates that the computation of $\Delta^{(\mathsf{fwr}_i+1)}$ and $\Delta^{(0)}$ is skipped.

Observe that the games G_{i-1}^1 and G_i^0 are identical. Therefore we have that

$$|\Pr[\mathrm{G}_{n_r}^1(\mathcal{A}) \Rightarrow 1] - \Pr[\mathrm{G}_1^0(\mathcal{A}) \Rightarrow 1]| \leq \sum_{i=1}^{n_r} |\Pr[\mathrm{G}_i^1(\mathcal{A}_i) \Rightarrow 1] - \Pr[\mathrm{G}_i^0(\mathcal{A}_i) \Rightarrow 1]|$$

and it is only left to show that

$$|\Pr[\mathrm{G}_i^1(\mathcal{A}_i) \Rightarrow 1] - \Pr[\mathrm{G}_i^0(\mathcal{A}_i) \Rightarrow 1]| \leq 2 \cdot \mathsf{Adv}_{\mathsf{EGA}}^{\mathsf{Multi\text{-}St\text{-}PR}}(\mathcal{B}) \ .$$

The i-th hybrid. Let $\mathcal{A}_i$ be an adversary that tries to distinguish G_i^0 from G_i^1. For all non-challenge queries, the responses of both games are identical. Therefore we assume that $\mathcal{A}_i$ asks for its challenge in the i-th insulated region. In that same region, the reduction will embed the samples from its Multi-St-PR oracle.

Consider the reduction $\mathcal{B}$ given in Figure 13. Here reduction $\mathcal{B}$ first samples a random bit b and then either simulates game G_i^b to $\mathcal{A}_i$ or a game where the challenge ciphertext is independent of $\bar{M}$ and $\bar{C}$. More concretely: In the case where the oracle Eval(β) returns "real" tuples of the form $(x_0, x_1, g_1 \star x_{\beta_1}, \dots, g_n \star x_{\beta_n})$ reduction $\mathcal{B}$ perfectly simulates the game G_i^b to adversary $\mathcal{A}$. In the case where the oracle Eval(β) returns "random" tuples of the form $(x_0, x_1, y_1, \dots, y_n)$ reduction $\mathcal{B}$ simulates a random game where in the i-th firewall encryptions and updates result in truly random (and therefore inconsistent) ciphertexts. Therefore both games G_i^0 and G_i^1 are completely indistinguishable in this case. Lastly, if $\mathcal{A}$ correctly guesses the bit b, then reduction $\mathcal{B}$ guesses that Eval returns "real" tuples. We now explain various aspects of reduction $\mathcal{B}$ in more detail.

Setup. In the setup phase $\mathcal{B}$ initially guesses the boundaries of the i-th firewall fwl_i and fwr_i. Subsequently, it generates all keys and update tokens in advance except for $k^{(\mathsf{fwl}_i)}, \dots, k^{(\mathsf{fwr}_i)}, \Delta^{(\mathsf{fwl}_i)}, \Delta^{(\mathsf{fwr}_i+1)}$. If $\mathcal{A}_i$ corrupts any of these keys or tokens then the guess for the i-th firewall was wrong and $\mathcal{B}$ aborts.

Non-Challenge Ciphertexts. In order to simulate non-challenge ciphertexts in epoch e, reduction $\mathcal{B}$ will proceed as follows:

- If $\mathcal{B}$ is asked for an encryption of a message $M = (m_1, \dots, m_n)$ it queries the Eval oracle on input M, receiving a tuple $\tau = (x_0, x_1, y_1, \dots, y_n)$. There are two cases:
  1. $\mathsf{e} \notin [\mathsf{fwl}_i, \mathsf{fwr}_i]$. In this case, the current epoch e is not inside the i-th firewall, so $\mathcal{B}$ can simply use the basis elements x_0, x_1 and its generated epoch key $k^{(\mathsf{e})}$ to compute an honest encryption $C^{(\mathsf{e})}$.
  2. $\mathsf{e} \in [\mathsf{fwl}_i, \mathsf{fwr}_i]$. In this case, the current epoch e belongs to the i-th firewall, therefore $\mathcal{B}$ does not know a corresponding epoch key. Nevertheless, $\mathcal{B}$ can use the elements $(y_1, \dots, y_n)$ as ciphertext $C^{(\mathsf{fwl}_i)}$ for epoch fwl_i and subsequently use its generated tokens $\Delta^{(\mathsf{fwl}_i)}, \dots, \Delta^{(\mathsf{e})}$ to update $C^{(\mathsf{fwl}_i)}$ to the desired epoch. This again yields the ciphertext $C^{(\mathsf{e})}$.

  Lastly, the tuple $(\mathsf{c}, M, C^{(\mathsf{e})}, \mathsf{e}, \tau)$ is stored in memory for later use (in particular, this allows $\mathcal{B}$ to provide consistent updates). Here c is simply a counter that counts the number of encryptions.

- If $\mathcal{B}$ is asked for an update of a ciphertext $C^{(\mathsf{e}-1)}$ from a previous epoch, it first recovers the corresponding tuple $(\mathsf{c}, M, C^{(\mathsf{e}-1)}, \mathsf{e}-1, \tau)$ from memory. Then $\mathcal{B}$ simply re-encrypts the message M as outlined above using the same value τ that was restored from memory.

Challenge Ciphertexts. Assume that the adversary makes the challenge query in epoch $\tilde{\mathsf{e}}$, providing a tuple $(\bar{M}, \bar{C})$ where $\bar{C}$ was created during the c-th encryption query. First, observe that the adversary cannot query the Upd oracle for an update of $\bar{C}$ for any epoch $\mathsf{e} \geq \mathsf{fwl}_i$ as this would trigger the trivial win condition $[\mathsf{fwl}_i, \mathsf{fwr}_i] \subseteq \mathcal{I}^* \cap \mathcal{C}^* \neq \emptyset$. Instead, the adversary can only obtain an update of the challenge ciphertext by querying Upd$\tilde{\text{C}}$.

Reduction $\mathcal{B}$ can simulate challenge-equal ciphertexts in an epoch e as follows:

- If $\tilde{\mathsf{e}} < \mathsf{fwl}_i$ then $\mathcal{B}$ can simply compute Upd$(\bar{C})$ with the token it created itself.
- If $\tilde{\mathsf{e}} > \mathsf{fwr}_i$ then $\mathcal{B}$ can simulate Enc$(\bar{M})$ using the key it created itself.
- If $\tilde{\mathsf{e}} \in [\mathsf{fwl}_i, \mathsf{fwr}_i]$ then $\mathcal{B}$ proceeds as follows. If $b = 0$ then $\mathcal{B}$ simulates Enc$(\bar{M})$ and if $b = 1$ then it simulates Upd$(\bar{C})$. Both oracles can be simulated the same way as in the case of non-challenge ciphertexts. In particular, Enc$(\bar{M})$ can be simulated by first querying Eval$(\bar{M})$, which results in a tuple $\tau = (x_0, x_1, y_1, \ldots, y_n)$. The elements $(y_1, \ldots, y_n)$ are now used as ciphertext $C^{(\mathsf{fwl}_i)}$ and updated to the required epoch $\tilde{\mathsf{e}}$ by using the update tokens $\Delta^{(\mathsf{fwl}_i)}, \ldots, \Delta^{(\tilde{\mathsf{e}})}$. Similarly, to simulate Upd$(\bar{C})$ the reduction first recovers the corresponding tuple $(\mathsf{c}, M, \cdot, \cdot, \tau)$ from memory and re-encrypts M using the value τ that was provided by Eval during the c-th encryption query. If necessary, $\mathcal{B}$ updates the ciphertext to the required epoch $\tilde{\mathsf{e}}$ by using the tokens $\Delta^{(\mathsf{fwl}_i)}, \ldots, \Delta^{(\tilde{\mathsf{e}})}$.

Eventually, $\mathcal{A}_i$ will output a bit b'. If $b = b'$ then $\mathcal{B}$ guesses that Eval returns "real" tuples of the form $(x_0, x_1, g_1 \star x_{\beta_1}, \ldots)$ and thus returns 1 to the Multi-St-PR game. Otherwise if $b \neq b'$ then $\mathcal{B}$ assumes that Eval returns "random" tuples $(x_0, x_1, y_1, \ldots)$ and therefore returns 0 to the Multi-St-PR game. Observe now that if Eval returns real tuples, then $\mathcal{B}$ perfectly simulates G_i^b. If instead Eval returns random tuples, then $\mathcal{B}$ simulates a completely random game to $\mathcal{A}_i$, which in turn means that $\mathcal{A}_i$ guesses the bit b with probability exactly $1/2$. In summary, we have

$$\begin{aligned}
\mathsf{Adv}_{\mathsf{EGA}}^{\mathsf{Multi\text{-}St\text{-}PR}}(\mathcal{B}) &= \left|\Pr[\mathrm{G}_{\mathsf{EGA}}^{\mathsf{Multi\text{-}St\text{-}PR\text{-}1}}(\mathcal{B}) \Rightarrow 1] - \Pr[\mathrm{G}_{\mathsf{EGA}}^{\mathsf{Multi\text{-}St\text{-}PR\text{-}0}}(\mathcal{B}) \Rightarrow 1]\right| \\
&= \left|\left(\frac{1}{2}\Pr[\mathrm{G}_i^0(\mathcal{A}_i) \Rightarrow 0] + \frac{1}{2}\Pr[\mathrm{G}_i^1(\mathcal{A}_i) \Rightarrow 1]\right) - \frac{1}{2}\right| \\
&= \left|\frac{1}{2}\left((1 - \Pr[\mathrm{G}_i^0(\mathcal{A}_i) \Rightarrow 1]) + \Pr[\mathrm{G}_i^1(\mathcal{A}_i) \Rightarrow 1]\right) - \frac{1}{2}\right| \\
&= \frac{1}{2}\left|\Pr[\mathrm{G}_i^1(\mathcal{A}_i) \Rightarrow 1] - \Pr[\mathrm{G}_i^0(\mathcal{A}_i) \Rightarrow 1]\right|.
\end{aligned}$$

Collecting all the probabilities yields the desired bound. □

D Full Proof of Lemma 3

Lemma 3. *Let the notation be as in Theorem 3. We then have*

$$\Pr[E_2] \leq n_E(n_{\mathsf{e}}+1) \cdot \mathsf{Adv}_{\mathsf{EGA}}^{\mathsf{Multi\text{-}St\text{-}UP}}(\mathcal{B}) + \frac{2n_E}{|\mathcal{X}|^2}\ ,$$

where n_{e} is the number of epochs and n_E is the number of encryption queries.

Proof. Consider the reduction in Figure 14. The main idea is again to embed the group elements $(g_1, \ldots, g_t)$ of the Multi-St-UP assumption into the second half of the secret key in the challenge epoch of the INT-CTXT game. To this end, the reduction first guesses the left border fwl of the firewall around the challenge epoch, resulting in an advantage loss of $(n_{\mathsf{e}}+1)$. To the left of the challenge epoch, the reduction simulates keys and tokens by itself, whereas inside of the firewall the reduction simulates *partial* update tokens. Note that we do not need to simulate anything to the right of the epoch where Try is queried. In fact, in the reduction in Figure 14 we simulate update tokens for all epochs $e \in [\mathsf{fwl}, n_{\mathsf{e}}]$ as we expect the adversary to query Try before the end of the insulated region. If instead the adversary queries for instance the CorrKey oracle for an epoch $\hat{\mathsf{e}} \geq \mathsf{fwl}$ then the guess for fwl was simply wrong and the reduction aborts. Furthermore, we do not need to keep track of trivial win conditions explicitly (in particular the twf and win flags).

The reduction provides consistent encryption randomness (i. e. consistent basis elements x_0, x_1) in the same way as in the proof of Theorem 2: For each encryption query to the left of the firewall, reduction $\mathcal{B}$ queries its own Eval oracle to obtain a tuple $(\beta, x_0, x_1, y_1, \ldots, y_t)$ which it stores for later use. It then programs the random oracle such that the "tag" $T = \mathsf{H}[M, x_0, x_1]$ equals β and then encrypts $M' = (M||T)$ with the current secret key and randomness (x_0, x_1). If $\mathsf{H}[M, x_0, x_1]$ is already set then this programming fails and the reduction needs to abort. Note that the probability of this event is bounded by $2n_E/|\mathcal{X}|^2$ (the additional factor of 2 comes from the fact that $x_0 \prec_{\mathsf{lex}} x_1$), which we will leave implicit in Figure 14.

Once an update or a new encryption is requested for an epoch inside the firewall, reduction $\mathcal{B}$ uses the values $y_i = g_i \star x_{\beta_i}$ from memory to embed the elements g_i into the secret key elements $(k_{n+1}^{(\mathsf{fwl})}, \ldots, k_{n+t}^{(\mathsf{fwl})})$. Note that the first n elements of the secret key are always known to the reduction.

To embed the challenge of the Multi-St-UP game, reduction $\mathcal{B}$ guesses a query index c^* and embeds the challenge γ into the c^*-th query to the Enc oracle. Recall that in the event E_2 the ciphertext $\tilde{C}$ provided by $\mathcal{A}$ is an updated version of a previous ciphertext $C^{(\mathsf{e})}$ since $\mathsf{H}[M, x_0, x_1]$ exists but was not queried directly by $\mathcal{A}$. Therefore, the only other possibility is that the value of $\mathsf{H}[M, x_0, x_1]$ was set during a call to Enc. If the guess was correct and $\mathcal{A}$ successfully provided such a ciphertext, then reduction $\mathcal{B}$ reverses potential (implicit) updates to the provided $\tilde{C}$, resulting in a valid ciphertext for epoch $\mathsf{e} = \mathsf{fwl}$. It is then easy to verify that the last t elements of that ciphertext are a valid solution for the Multi-St-UP assumption.

Adversary $\mathcal{B}^{\textsc{Eval},\textsc{Chall}}$

```
$k^{(0)} \leftarrow \mathsf{BIN\text{-}UE.KeyGen}$
$\Delta^{(0)} := \perp$
$\mathsf{e}, \mathsf{c} := 0$
$\mathsf{H}[\cdot] := \perp$
$\mathcal{L} := \emptyset$
$\mathsf{fwl} \xleftarrow{\$} [0, n_{\mathsf{e}}]$
$\mathsf{c}^* \xleftarrow{\$} [1, n_E]$
$\textbf{for } j \in [0, \mathsf{fwl} - 1]:$
    $k^{(j)} \leftarrow \mathsf{BIN\text{-}UE.KeyGen}$
    $\Delta^{(j)} := \mathsf{BIN\text{-}UE.TokenGen}(k^{(j-1)}, k^{(j)})$  // †
$\textbf{for } j \in [\mathsf{fwl}, n_{\mathsf{e}}]:$
    $k_1^{(j)}, \ldots, k_n^{(j)} \xleftarrow{\$} \mathcal{G}^n$
$\textbf{for } j \in [\mathsf{fwl} + 1, n_{\mathsf{e}}]:$
    $\textbf{for } \ell \in [1, n]:$
        $\Delta_\ell^{(j)} := k_\ell^{(j)} / k_\ell^{(j-1)}$
    $\textbf{for } \ell \in [n + 1, t]:$
        $\Delta_\ell^{(j)} \xleftarrow{\$} \mathcal{G}$
Run $\mathcal{A}^{\mathcal{O}}$
$\textbf{return } \perp$ to $\mathrm{G}_{\mathsf{EGA}}^{\mathsf{Multi\text{-}St\text{-}UP}}$   // $\textsc{Try}$ never called
```

Oracle $\textsc{Enc}(M)$

```
Let $M = (m_1, \ldots, m_n)$
$\mathsf{c} := \mathsf{c} + 1$
$\textbf{if } \mathsf{c} = \mathsf{c}^*:$
    $x_0, x_1 \xleftarrow{\$} \mathcal{X}$                    // $x_0 \prec_{\mathsf{lex}} x_1$
    $\gamma \leftarrow \textsc{Chall}(x_0, x_1)$
    $\mathsf{H}[M, x_0, x_1] := \gamma$
    $\tau = \perp$
$\textbf{else}:$
    $\tau := (\beta, x_0, x_1, y_1, \ldots, y_t) \leftarrow \textsc{Eval}$
    $\mathsf{H}[M, x_0, x_1] := \beta$
$M' := (M || \mathsf{H}[M, x_0, x_1])$
$\textbf{if } \mathsf{e} < \mathsf{fwl}:$
    $C_j^{(\mathsf{e})} := k_j^{(\mathsf{e})} \star x_{m'_j}$
$\textbf{else if } \mathsf{e} \geq \mathsf{fwl} \textbf{ and } \mathsf{c} \neq \mathsf{c}^*:$
    $\textbf{for } j \in [1, n]:$
        $C_j^{(e)} := k_j^{(e)} \star x_{m_j}$
    $\textbf{for } j \in [1, t]:$
        $C_{n+j}^{(e)} := y_j$
$\textbf{else}: \textbf{abort}$              // wrong guess for $\mathsf{fwl}$
$C^{(\mathsf{e})} := (C_1^{(\mathsf{e})}, \ldots, C_{n+t}^{(\mathsf{e})})$
$\mathcal{L} := \mathcal{L} \cup \{(\mathsf{c}, M', C^{(\mathsf{e})}, \mathsf{e}, \tau)\}$
$\textbf{return } C^{(\mathsf{e})}$
```

Oracle $\textsc{Next}$

```
$\mathsf{e} := \mathsf{e} + 1$
```

Oracle $\textsc{Try}(\tilde{C})$

```
$(M', x_0, x_1) := \mathsf{BIN\text{-}UE.Dec}(k^{(\mathsf{e})}, \tilde{C})$
Let $M' = (M || T)$
$\textbf{if } \mathsf{H}[M, x_0, x_1] \neq \gamma : \textbf{abort}$   // wrong guess $\mathsf{c}^*$
Set $C^{(\mathsf{e})} = \tilde{C}$
$\textbf{for } \ell \in [\mathsf{e}, \mathsf{fwl}]:$            // backwards iteration
    $\textbf{for } j \in [n + 1, n + t]:$
        $C_j^{(\ell-1)} := (\Delta_j^{(\ell)})^{-1} \star C_j^{(\ell)}$
$\textbf{return } C_{n+1}^{(\mathsf{fwl})}, \ldots, C_{n+t}^{(\mathsf{fwl})}$ to $\mathrm{G}_{\mathsf{EGA}}^{\mathsf{Multi\text{-}St\text{-}UP}}$
```

Oracle $\textsc{Upd}(C^{(\mathsf{e}-1)})$

```
$\textbf{if } \nexists \mathsf{c} : (\mathsf{c}, M', C^{(\mathsf{e}-1)}, \mathsf{e} - 1, \tau) \in \mathcal{L}:$
    $\textbf{return } \perp$
$\textbf{if } \tau = \perp : \textbf{abort}$   // wrong guess for $\mathsf{c}^*$ or $\mathsf{fwl}$
Let $\tau = (\beta, x_0, x_1, y_1, \ldots, y_t)$
$\textbf{if } \mathsf{e} < \mathsf{fwl}:$
    Let $M' = (m'_1, \ldots, m'_{n+t})$
    $C_j^{(\mathsf{e})} := k_j^{(\mathsf{e})} \star x_{m'_j}$
$\textbf{else}:$
    Let $M' = (M || T)$
    $\textbf{for } j \in [1, n]:$
        $C_j^{(\mathsf{fwl})} := k_j^{(\mathsf{fwl})} \star x_{m_j}$
    $\textbf{for } j \in [1, t]:$
        $C_{n+j}^{(\mathsf{fwl})} := y_j$
    $\textbf{for } \ell \in [\mathsf{fwl} + 1, \mathsf{e}]:$
        $C_j^{(\ell)} := \Delta_j^{(\ell)} \star C_j^{(\ell-1)}$
$\mathcal{L} := \mathcal{L} \cup \{(\mathsf{c}, M', C^{(\mathsf{e})}, \mathsf{e}, \tau)\}$
$\textbf{return } C^{(\mathsf{e})}$
```

Oracle $\mathrm{H}(M, x_0, x_1)$

```
$\textbf{if } \mathsf{H}[M, x_0, x_1] = \perp:$
    $\beta \xleftarrow{\$} \{0, 1\}^t$
    $\mathsf{H}[M, x_0, x_1] := \beta$
$\textbf{return } \mathsf{H}[M, x_0, x_1]$
```

Oracle $\textsc{CorrToken}(\hat{\mathsf{e}})$

```
$\textbf{if } \hat{\mathsf{e}} > \mathsf{e} : \textbf{return } \perp$
$\textbf{if } \hat{\mathsf{e}} = \mathsf{fwl} : \textbf{abort}$          // wrong guess for $\mathsf{fwl}$
$\textbf{return } \Delta^{(\hat{\mathsf{e}})}$
```

Oracle $\textsc{CorrKey}(\hat{\mathsf{e}})$

```
$\textbf{if } \hat{\mathsf{e}} > \mathsf{e} : \textbf{return } \perp$
$\textbf{if } \hat{\mathsf{e}} \geq \mathsf{fwl} : \textbf{abort}$          // wrong guess for $\mathsf{fwl}$
$\textbf{return } k^{(\hat{\mathsf{e}})}$
```

Fig. 14. The reduction $\mathcal{B}$ simulating game $\mathrm{G}_{\mathsf{COM\text{-}UE}}^{\mathsf{INT\text{-}CTXT}}$ and event E_2. The set of oracles that $\mathcal{A}$ has access to is defined as $\mathcal{O} = \{\textsc{Enc}, \textsc{Try}, \textsc{Next}, \textsc{Upd}, \textsc{CorrKey}, \textsc{CorrToken}, \mathrm{H}\}$. † indicates that the computation of $\Delta^{(0)}$ is skipped.

Compared to Lemma 2, there are a few more situations where the reduction needs to abort. For instance, $\mathcal{A}$ might ask for an update of $C^{(\mathsf{e})}$ in an epoch $\mathsf{e}' \geq \mathsf{fwl}$, which the reduction $\mathcal{B}$ cannot provide as it does not know the corresponding secret key $k^{(\mathsf{e}')}$. Furthermore, $\mathsf{H}[M, x_0, x_1]$ is programmed to be $\gamma = \textsc{Chall}(x_0, x_1)$, therefore the reduction is not provided with elements $(y_1, \ldots, y_t)$ that would normally be used to update a ciphertext into the firewall. However, if that case occurs then the guess for the firewall was simply wrong and the reduction aborts. A similar argument can be made for the other situations where the reduction needs to abort. This completes the proof. □

E The Algebraic Group Action Model

We recall a slightly adapted definition of the Algebraic Group Action Model introduced in [24].

Definition 14 (Algebraic Group Action Algorithms). *Let $(\mathcal{G}, \mathcal{X}, \star, \tilde{x})$ be a fixed cyclic effective group action. An algorithm $\mathcal{A}$ is called* algebraic *if for each output element $y \in \mathcal{X}$ it additionally provides a representation relative to a previously received set element. Concretely, if $(x_1, \ldots, x_\ell) \in \mathcal{X}^\ell$ is the list of received set elements so far, $\mathcal{A}$ additionally provides a group element $s \in \mathcal{G}$ and an element $v \in \{x_1, \ldots, x_\ell\}$ such that $y = s \star v$. If an oracle is queried on some set elements, then $\mathcal{A}$ also has to provide a representation for each set element contained in that query.*

Additionally, if the group action supports twists, we extend the representation by a bit b, indicating whether the base element was twisted before applying the group action. We require that all auxiliary input provided to the adversary which is not in $\mathcal{X}$ does not depend on elements from $\mathcal{X}$.

In the Algebraic Group Action Model (AGAM) we assume every adversary to be algebraic.

E.1 A Note on Rerandomization in the AGAM

In the AGAM we do not need to model rerandomization explicitly. Let $\mathcal{A}, \mathcal{B}, \mathcal{R}$ be three algorithms where $\mathcal{A}$ interacts with $\mathcal{R}$ and $\mathcal{R}$ interacts with $\mathcal{B}$. Assume $\mathcal{B}$ to be an algebraic algorithm.

First, we note that any rerandomization that $\mathcal{B}$ applies to an input element x is futile as $\mathcal{B}$ is an algebraic algorithm and therefore always has to output a corresponding representation (s, x) which gives away the rerandomization s. We can therefore assume that $\mathcal{B}$ does not rerandomize elements.

Next, we argue that we do not need to model $\mathcal{R}$'s rerandomization explicitly as well. For this, assume that $\mathcal{R}$ receives a set element $x \in \mathcal{X}$ from $\mathcal{A}$ that it wants to rerandomize and then forward to $\mathcal{B}$. For this, $\mathcal{R}$ samples a uniform $g \in \mathcal{G}$, stores g for later use and sends $y = g \star x$ to $\mathcal{B}$. If at any point $\mathcal{R}$ receives a set element z together with a representation (s, y) then certainly (sg, x) is an equivalent representation for z. Therefore, from $\mathcal{R}$'s perspective the set element and representation output by $\mathcal{B}$ can be turned into a representation that only involves the element output by $\mathcal{A}$. We can thus assume $\mathcal{R}$ to rerandomize elements implicitly without the need to make this rerandomization explicit in the proof.