Updatable Encryption from Group Actions

Antonin Leroux[1,2] and Maxime Roméas[1]

[1] LIX, CNRS, École polytechnique, INRIA, Institut Polytechnique de Paris, 91120 Palaiseau, France
[2] DGA
antonin.leroux@polytechnique.org
romeas@lix.polytechnique.fr

Abstract. Updatable Encryption (UE) allows to rotate the encryption key in the outsourced storage setting while minimizing the bandwith used. The server can update ciphertexts to the new key using a token provided by the client. UE schemes should provide strong confidentiality guarantees against an adversary that can corrupt keys and tokens.
This paper studies the problem of building UE in the group action framework. We introduce a new notion of Mappable Effective Group Action (MEGA) and show that we can build CCA secure UE from a MEGA by generalizing the SHINE construction of Boyd *et al.* at Crypto 2020. Unfortunately, we do not know how to instantiate this new construction in the post-quantum setting. Doing so would solve the open problem of building a CCA secure post-quantum UE scheme.
Isogeny-based group actions are the most studied post-quantum group actions. Unfortunately, the resulting group actions are not mappable. We show that we can still build UE from isogenies by introducing a new algebraic structure called Effective Triple Orbital Group Action (ETOGA). We prove that UE can be built from an ETOGA and show how to instantiate this abstract structure from isogeny-based group actions. This new construction solves two open problems in ciphertext-independent post-quantum UE. First, this is the first post-quantum UE scheme that supports an unbounded number of updates. Second, our isogeny-based UE scheme is the first post-quantum UE scheme not based on lattices.

Keywords: Updatable Encryption, Group Actions, Isogenies, Post-Quantum Cryptography

1 Introduction

Updatable Encryption (UE), introduced by Boneh *et al.* in 2013 [6], is a very useful primitive for storing encrypted data[3] on a cloud server. To fight against risks of seeing the secret key compromised, the client can always download its data, decrypt it, encrypt it under a new key and upload it back on the server. However, this solution uses too much bandwith to be considered practical. Dealing with key rotation while minimizing bandwith usage is the goal of UE. This work focuses on the *ciphertext-independent* variant of UE, where the client generates a single value, called a *token*, when rotating its key. This token can then be used by the server to update all of the client's ciphertexts under the latest key. Unlike symmetric encryption, UE schemes aim at preserving the confidentiality of the data in a setting where secret keys and update tokens can leak. The huge real-life applications of UE explains the recent renewed interest on the subject [25,22,9,21,29].

Related Work. Security notions for UE have evolved a lot since the original proposal of [6]. Lehmann and Tackmann [25] proposed two CPA security notions where the adversary can adaptively corrupt keys and tokens. Their IND-ENC notion requires fresh encryptions to be indistinguishable and their IND-UPD notion asks the same for updated ciphertexts. Klooß *et al.* [22] augmented the previous notions with CCA security and integrity protection. Boyd *et al.* [9] introduced the IND-UE notion which is stronger than previous ones and requires fresh encryptions to be indistinguishable from updated ciphertexts. They also show that a CPA UE scheme with ciphertext integrity (CTXT) is CCA.

As for UE constructions in the classical setting, RISE of [25], is an updatable variant of ElGamal where the public key is used in the token. [22] introduced two generic constructions based on encrypt-and-MAC

[3] UE is a variant of symmetric encryption

(secure under DDH) and on the Naor-Yung transform (secure under SXDH). Boyd *et al.* [9] proposed the permutation-based SHINE schemes, that achieve their stronger detIND-UE-CCA security notion in the ideal cipher model (under DDH).

In the post-quantum setting, Jiang [21] presented the first post-quantum UE scheme LWEUE (secure under LWE). In [29], Nishimaki introduced RtR, another LWE-based UE scheme, which is the first ciphertext-independent UE scheme that prevents the adversary from obtaining the new key from the knowledge of the update token and the old key. Nishimaki showed that UE schemes with this property have stronger security than those without. These two LWE-based schemes use homomorphic operations to re-randomize updated ciphertexts which has two main drawbacks. On one hand, ciphertext noise grows with each key update, which means that these schemes only support a bounded number of updates. On the other hand, using the homomorphic property and the knowledge of the update token, an adversary can craft ciphertexts of related messages which means that these schemes are not CCA secure (only randIND-UE-CPA).

We try to overcome these issues by using group actions to build UE. The first efficient post-quantum group action was introduced by Castryck *et al.* using isogenies [11]. Their construction is called CSIDH and it uses the group action of the class group of the quadratic order $\mathbb{Z}[\sqrt{-p}]$ on the supersingular curves defined over $\mathbb{F}_p$. The resulting group action is believed to be post-quantum one-way, i.e., hard to invert, and this has motivated to study the protocols that can be built generically upon a cryptographic group action. This is what is done for instance in the work of Alamati *et al.* [1]. Since, then, other proposals of post-quantum group action have been introduced such as the work of Tang *et al.* [37] or of Ji *et al.* [20], both based on multivariate problems. To our knowledge, there does not exists any UE scheme in the group action framework, even if we will show that the SHINE construction fits into that framework.

Eaton *et al.* [18] have introduced an isogeny-based "updatable encryption" scheme on the CSIDH group action. However, their construction does not achieve at all the same primitive. Their construction target public key encryption schemes and the updates are only applied to keys and not ciphertexts.Their construction shares some similarities with ours as they use elements of the class group as secret keys and the update mechanism is made of a group action computation.

Overview of the contributions. We present two new generic constructions of UE from abstract algebraic frameworks. The first construction is called GAINE for Group-Action Ideal-cipher Nonce-based Encryption. GAINE generalizes the SHINE [9] construction and builds UE from a weak pseudorandom group action as defined in [1]. In fact, SHINE is a concrete instantiation of GAINE for the group action of $\mathbb{Z}_q^*$ on any cyclic group H of order q by exponentiation. The hardness of the DDH problem over this group implies that the resulting group action is weak pseudorandom. For SHINE, and GAINE, to work, we need one other thing: that the set H is mappable, i.e., that there exists an invertible and efficient map π going from the space of messages to the set of the group action. The authors of SHINE showed how to build this map when H is the group of points of an elliptic curve. The idea is to use this map to translate messages as elements of the group H before encrypting them with exponentiation, using secret keys as exponents. In that setting, the token is simply an exponent and the update is just another exponentiation. Overall, SHINE is quite simple and we show that it adapts very naturally to the setting of cryptographic group actions. If the group action is abelian and transitive, a straightforward generalization of SHINE's security proof shows that we can instantiate GAINE with a group action that satisfies the analog of the classical DDH assumption for group actions. Moreover, we show that it is still possible to instantiate GAINE with a nonabelian group action. Only this time, we need the group action to be weak pseudorandom, i.e., an adversary cannot distinguish between many samples that or either of the form $(s_i, g \star s_i)$, where g is a random group element and the elements s_i are random set elements, or samples of the form (s_i, t_i) where s_i and t_i are random set elements. Moreover, we show that we can apply the transform used in SHINE to make our GAINE UE scheme CCA. Unfortunately, we do not know how to instantiate GAINE in the post-quantum setting. Indeed, there are currently two cryptographic group actions that are mappable and believed to be post-quantum. The first one was introduced by Tang *et al.* [37] and uses alternating trilinear forms, while the second one from Ji *et al.* [20] is based on tensors. These two group actions are not abelian and they are conjectured to be weak pseudorandom but only over a very small number of samples. However, when working with nonabelian group actions, our security proof for GAINE needs one sample per ciphertext. Thus, we are not currently able to

give a practical post-quantum instantiation for GAINE. Finding one would solve the problem left open by Jiang [21] at Asiacrypt2020 to build a CCA post-quantum UE scheme.

Isogeny-based cryptography provides another source of post-quantum weak pseudorandom group actions. However, in the case of the CSIDH cryptographic group action, it is notoriously hard to sample elements in the set [8]. Thus, we cannot really hope to instantiate GAINE with CSIDH or another similar group action from isogenies. Isogeny-based group actions have been more studied than their multivariate counterpart and while there are on-going discussions regarding the exact level of security reached by these group actions, the recent attacks against SIDH (see [12] for instance) do not apply to the setting of CSIDH, and so we can have some confidence in the fact that the underlying problems are hard. This is why we show how to build UE from the CSIDH group action. We circumvent the mappable requirement by using an idea of Moriya *et al.* for their SIGAMAL encryption scheme [27]. Intuitively, their idea is to see messages as scalars that are mapped to points of an elliptic curve using the scalar multiplication but this time in a group where discrete logarithm is easy so that we can decrypt efficiently. The points obtained in this manner are encrypted using isogenies. We obtain an analog of CSIDH by considering a set made of elements constituted by a curve and a point (and not just a curve). We refine the idea of Moriya *et al.* to get a scheme that is updatable. The way we circumvent the issue that CSIDH is not mappable could be of independent interest as there are numerous examples of protocols where this proves to be a big obstacle. We extracted an abstract framework of this idea to identify the algebraic structure required by our new UE scheme. This gave us what we call a TOGA for Triple Orbital Group Action. As the name suggests, there are three group actions involved in this scheme, each with a specific role, and we require the three different operations to interact in a very specific way that we summarized in Fig. 11. We introduce TOGA-UE, a generic UE protocol based on a TOGA family. Unfortunately, deriving a CCA encryption scheme from TOGA-UE seems hard and we leave that to future work. Finally, we show how to build a TOGA from the CSIDH group action. This gives an instantiation of post-quantum UE based on another family of assumptions. In particular, it is the first post-quantum UE scheme that does not suffer any limitations regarding the number of updates, contrary to the solutions based on lattices. This instantiation has the same security as CSIDH. Unfortunately, reaching higher security levels is currently out of reach because precomputations become impractical. We leave the problem of overcoming this obstacle to future work.

Outline of the paper. In Section 2 we introduce the necessary notions and backgrounds. Section 3 is dedicated to our first construction GAINE of UE from a MEGA. Our second UE scheme TOGA-UE is introduced in Section 4, where we present our new algebraic structure of TOGA before showing how to build UE from it. Finally, in Section 5, we show how to instantiate TOGA-UE.

2 Preliminaries

Notations. We use λ to denote the security parameter. For a finite set S, we use $s \xleftarrow{\$} S$ to sample uniformly from S. For a probability distribution $\mathcal{D}$ on a finite set S, we use $s \leftarrow \mathcal{D}$ to sample from $\mathcal{D}$. We use $\mathfrak{S}(S)$ to denote the set of permutations of a finite set S. For an algorithm $\mathcal{A}$ and an oracle $\mathcal{O}$, $\mathcal{A}$ having access to $\mathcal{O}$ is denoted by $\mathcal{A}^{\mathcal{O}}$.

2.1 Cryptographic group actions

In this section, we give a few reminders about group actions and how they can be endowed with hardness properties for cryptographic use. We use the framework of cryptographic group actions of Alamati *et al.* [1].

Definition 1 (Group Action). *Let G be a group for a law written multiplicatively and let S be a set. A group action of G on S is an operation $\star : G \times S \to S$ such that*

1. *If 1_G is the identity element of G, then for any $s \in S$, we have $1_G \star s = s$.*
2. *For any $g, h \in G$ and any $s \in S$, we have $(gh) \star s = g \star (h \star s)$.*

We may use the notation $(G, S, \star)$ to denote a group action. We stress that the group actions used in this work do not need to be abelian. A group action $(G, S, \star)$ partitions the set S into a disjoint union of *orbits* where the orbit of $s \in S$ is the set $\mathrm{Orb}(s) := \{g \star s \mid g \in G\} \subseteq S$.

Properties of group actions. Our group actions $(G, S, \star)$ can be:

1. Transitive: A group action is *transitive* if it has a single orbit, i.e., if for any $(s_1, s_2) \in S$, there exists $g \in G$ such that $g \star s_1 = s_2$. We can always obtain a transitive group action from any group action. Indeed, take $s \in S$, one can easily verify that $(G, \mathrm{Orb}(s), \star)$ is a transitive group action.
2. Free: A group action is *free* if for all $g \in G$, $g = 1_G$ if and only if there exists $s \in S$ such that $g \star s = s$.

Since we need to define computational assumptions related to group actions, we need a notion of efficiency.

Definition 2 (Effective Group Action [1]). *$(G, S, \star)$ is an effective group action (EGA), with respect to a parameter λ, if the following properties are satisfied:*

1. *The group G is finite and there exist PPT algorithms for:*
   (a) *Membership testing, i.e., to decide if a given bit string represents a valid element in G.*
   (b) *Equality testing, i.e., to decide if two bit strings represent the same group element in G.*
   (c) *Sampling, i.e., to sample an element g from a distribution $\mathcal{D}_G$ on G.*
   (d) *Operation, i.e., to compute gh for any $g, h \in G$.*
   (e) *Inversion, i.e., to compute g^{-1} for any $g \in G$.*
2. *The set S is finite and there exist efficient PPT algorithms for:*
   (a) *Membership testing.*
   (b) *Unique representation, i.e., given any arbitrary set element $s \in S$, compute a string $\hat{s}$ that canonically represents s.*
3. *There exists a distinguished element $s_0 \in S$, called the origin, such that its bit-string representation is known.*
4. *There exists an efficient algorithm that given (some bit-string representations of) any $g \in G$ and any $s \in S$, outputs $g \star s$.*

Definition 3 (Group Action Family). *We say that $\mathcal{GA}$ is a group action family if, for a security parameter λ, $\mathcal{GA}(\lambda)$ consists of a group action $(G, S, \star)$ where $|G|, |S| = \mathsf{poly}(\lambda)$.*

In the following, let $\mathcal{GA}$ be a group action family. We define weak pseudorandom group actions:

Definition 4 (Weak Pseudorandom Group Action [1]). *Let $(G, S, \star)$ be $\mathcal{GA}(\lambda)$ for some security parameter λ. Let $\mathcal{D}_G$ and $\mathcal{D}_S$ be distributions on G and S respectively. For $g \in G$, let $\pi_g : S \to S$ be the permutation defined by $\pi_g : s \mapsto g \star s$. For a permutation $f \in \mathfrak{S}(S)$, we use $f^{\$}$ to denote the randomized oracle that, when queried, samples $s \leftarrow \mathcal{D}_S$ and outputs $(s, f(s))$. We say that $(G, S, \star)$ is $(\mathcal{D}_G, \mathcal{D}_S)$-weakly pseudorandom if, for all PPT adversaries $\mathcal{A}$, we have:*

$$\mathbf{Adv}^{\mathsf{wk\text{-}PR}}_{\mathcal{GA},\mathcal{A}}(\lambda) := \left|\Pr[\mathbf{Exp}^{\mathsf{wk\text{-}PR\text{-}0}}_{\mathcal{GA},\mathcal{A}}(\lambda) = 1] - \Pr[\mathbf{Exp}^{\mathsf{wk\text{-}PR\text{-}1}}_{\mathcal{GA},\mathcal{A}}(\lambda) = 1]\right| \leq \mathsf{negl}(\lambda)$$

where $\mathbf{Exp}^{\mathsf{wk\text{-}PR\text{-}}b}_{\mathcal{GA},\mathcal{A}}(\lambda)$ is the experiment described in figure 1.

Informally, a group action $(G, S, \star)$ is $(\mathcal{D}_G, \mathcal{D}_S)$-weakly pseudorandom if there is no PPT adversary that can distinguish tuples of the form $(s_i, g \star s_i)$ from (s_i, u_i) where $g \leftarrow \mathcal{D}_G$ and each $s_i, u_i \leftarrow \mathcal{D}_S$. If both distributions are uniform, we omit them and we say that the group action is weakly pseudorandom.

Finally, we define weak unpredictable group actions:

$\mathbf{Exp}^{\mathsf{wk\text{-}PR}\text{-}b}_{\mathcal{GA},\mathcal{A}}(\lambda)$:

1. $(G, S, \star) \leftarrow \mathcal{GA}(\lambda)$
2. **if** $b = 0$
3. $\quad g \leftarrow \mathcal{D}_G$
4. $\quad \mathcal{O}.\mathsf{Sample} \leftarrow \pi_g^{\$}$
5. **else** $(b = 1)$
6. $\quad \pi \xleftarrow{\$} \mathfrak{S}(S)$
7. $\quad \mathcal{O}.\mathsf{Sample} \leftarrow \pi^{\$}$
8. $b' \leftarrow \mathcal{A}^{\mathcal{O}.\mathsf{Sample}}(1^\lambda, (G, S, \star))$
9. **if** $b' = b$
10. $\quad$ **return** 1
11. **else**
12. $\quad$ **return** 0

Fig. 1. Weak pseudorandom group action experiment. Recall that $\mathcal{D}_G$ and $\mathcal{D}_S$ are distributions on G and S respectively. For a permutation $f \in \mathfrak{S}(S)$, we use $f^{\$}$ to denote the randomized oracle that samples $s \leftarrow \mathcal{D}_S$ and outputs $(s, f(s))$.

Definition 5 (Weak Unpredictable Group Action [1]). *Let $(G, S, \star)$ be $\mathcal{GA}(\lambda)$ for some security parameter λ. Let $\mathcal{D}_G$ and $\mathcal{D}_S$ be distributions on G and S respectively. For $g \in G$, let $\pi_g : S \to S$ be the permutation defined by $\pi_g : s \mapsto g \star s$. For a permutation $f \in \mathfrak{S}(S)$, we use $f^{\$}$ to denote the randomized oracle that, when queried, samples $s \leftarrow \mathcal{D}_S$ and outputs $(s, f(s))$. We say that $(G, S, \star)$ is $(\mathcal{D}_G, \mathcal{D}_S)$-weakly unpredictable if, for all PPT adversaries $\mathcal{A}$, we have :*

$$\mathbf{Adv}^{\mathsf{wk\text{-}UP}}_{\mathcal{GA},\mathcal{A}}(\lambda) := \Pr[\mathcal{A}^{\pi_g^{\$}}(s^*) = \pi_g(s^*)] \leq \mathsf{negl}(\lambda)$$

where $g \leftarrow \mathcal{D}_G$ and $s^* \leftarrow \mathcal{D}_S$. We denote this experiment by $\mathbf{Exp}^{\mathsf{wk\text{-}UP}}_{\mathcal{GA},\mathcal{A}}$.

Informally, $(G, S, \star)$ is $(\mathcal{D}_G, \mathcal{D}_S)$-weakly unpredictable if, given polynomially many tuples of the form $(s_i, g \star s_i)$ where $g \leftarrow \mathcal{D}_G$ and each $s_i \xleftarrow{\$} \mathcal{D}_S$, there is no PPT adversary that can compute $g \star s^*$ for a given challenge $s^* \leftarrow \mathcal{D}_S$. If both distributions are uniform, we simply speak of a weakly unpredictable group action.

2.2 Updatable Encryption

In this section, we describe the syntax and security definitions of UE, we follow the presentations of [29,9,25,21]. An UE scheme operates in *epochs*, where an epoch is an index incremented with each key update. Let $n + 1$ be the maximum number of epochs (this is only for proof purposes).

Definition 6. *An updatable encryption scheme* UE *for message space $\mathcal{M}$ consists of a tuple of PPT algorithms* (UE.Setup, UE.KeyGen, UE.TokenGen, UE.Enc, UE.Dec, UE.Upd) *where:*

- $\mathsf{UE.Setup}(1^\lambda) \to \mathsf{pp}$*: The setup algorithm takes as input the security parameter and outputs a public parameter* pp*.*
- $\mathsf{UE.KeyGen}(\mathsf{pp}) \to k_\mathsf{e}$*: The key generation algorithm takes as input the public parameter* pp *and outputs an epoch key k_e.*
- $\mathsf{UE.Enc}(k, m) \to c$*: The encryption algorithm takes as input an epoch key k and a message m and outputs a ciphertext c.*
- $\mathsf{UE.Dec}(k, c) \to m$*: The encryption algorithm takes as input an epoch key k and a ciphertext c and outputs a message m or $\perp$.*
- $\mathsf{UE.TokenGen}(k_\mathsf{e}, k_{\mathsf{e}+1}) \to \Delta_{\mathsf{e}+1}$*: The token generation algorithm takes as input two keys of consecutive epochs* e *and* $\mathsf{e}+1$ *and outputs a token $\Delta_{\mathsf{e}+1}$.*

- $\mathsf{UE.Upd}(\Delta_{e+1}, c_e) \to c_{e+1}$: *The update algorithm takes as input a token* Δ_{e+1} *and a ciphertext* c_e *and outputs a ciphertext* c_{e+1}.

Definition 7 (Correctness). *For any* $m \in \mathcal{M}$, *for* $0 \le e_1 \le e_2 \le n+1$, *it holds that* $\Pr[\mathsf{UE.Dec}(k_{e_2}, c_{e_2}) \neq m] \le \mathsf{negl}(\lambda)$, *where* $\mathsf{pp} \leftarrow \mathsf{UE.Setup}(1^\lambda)$, $k_{e_1}, \ldots, k_{e_2} \leftarrow \mathsf{UE.KeyGen}(\mathsf{pp})$, $c_{e_1} \leftarrow \mathsf{UE.Enc}(k_{e_1}, m)$, *and* $\Delta_{i+1} \leftarrow \mathsf{UE.TokenGen}(k_i, k_{i+1})$, $c_{i+1} \leftarrow \mathsf{UE.Upd}(\Delta_{i+1}, c_i)$ *for* $i \in [e_1, e_2 - 1]$.

Security definitions. In all of our UE schemes, the Upd algorithm is *deterministic*. Thus, we only consider security definitions in the deterministic update setting.

Definition 8 (detIND-UE-atk [9]). *Let* $\mathsf{UE} = (\mathsf{UE.Setup}, \mathsf{UE.KeyGen}, \mathsf{UE.TokenGen}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd})$ *be an updatable encryption scheme. The* detIND-UE-atk *advantage, for* $\mathsf{atk} \in \{\mathsf{CPA}, \mathsf{CCA}\}$ *of an adversary* $\mathcal{A}$ *against* UE *is given by*

$$\mathbf{Adv}^{\mathsf{detIND\text{-}UE\text{-}atk}}_{\mathsf{UE},\mathcal{A}}(\lambda) := \left|\Pr[\mathbf{Exp}^{\mathsf{detIND\text{-}UE\text{-}atk\text{-}0}}_{\mathsf{UE},\mathcal{A}} = 1] - \Pr[\mathbf{Exp}^{\mathsf{detIND\text{-}UE\text{-}atk\text{-}1}}_{\mathsf{UE},\mathcal{A}} = 1]\right|$$

where the confidentiality experiment $\mathbf{Exp}^{\mathsf{detIND\text{-}UE\text{-}atk}\text{-}b}_{\mathsf{UE},\mathcal{A}}$ *is given in fig. 2.*

$\mathbf{Exp}^{\mathsf{detIND\text{-}UE\text{-}atk}\text{-}b}_{\mathsf{UE},\mathcal{A}}(\lambda)$

1. **do** $\mathsf{UE.Setup}(1^\lambda)$
2. $\mathtt{ors} \leftarrow \mathcal{O}.\{\mathsf{Enc}, \mathsf{Upd}, \mathsf{Next}, \mathsf{Corr}\}$
3. **if** $\mathsf{atk} = \mathsf{CCA}$
4. $\quad \mathtt{ors} \leftarrow \mathtt{ors} \cup \{\mathcal{O}.\mathsf{Dec}\}$
5. $(\bar{M}, \bar{C}) \leftarrow \mathcal{A}^{\mathtt{ors}}(1^\lambda)$
6. $\tilde{C}_e \leftarrow \mathcal{O}.\mathsf{Chall}(\bar{M}, \bar{C})$
7. $b' \leftarrow \mathcal{A}^{\mathtt{ors}, \mathcal{O}.\mathsf{Upd}\tilde{\mathsf{C}}}(\tilde{C}_e)$
8. **if** $\mathcal{K}^* \cap \mathcal{C}^* \neq \emptyset$ **or** $\mathcal{I}^* \cap \mathcal{C}^* \neq \emptyset$
9. $\quad \mathsf{twf} \leftarrow 1$
10. **if** $\mathsf{twf} = 1$
11. $\quad b' \xleftarrow{\$} \{0, 1\}$
12. **return** b'

Fig. 2. Description of the confidentiality experiment $\mathbf{Exp}^{\mathsf{detIND\text{-}UE\text{-}atk}\text{-}b}_{\mathsf{UE},\mathcal{A}}$ for scheme UE (with deterministic updates) and adversary $\mathcal{A}$, for $\mathsf{atk} \in \{\mathsf{CPA}, \mathsf{CCA}\}$. The oracles are given in fig. 3. Trivial win conditions, i.e., deciding the value of twf and computing $\mathcal{K}^*$, $\mathcal{C}^*$, $\mathcal{I}^*$ are discussed in sec. 2.2 and 2.2

Ciphertext integrity game: definitions and composition result. We follow the presentation of [9]. In the ciphertext integrity (CTXT) game, the adversary is given access to oracles $\mathcal{O}.\mathsf{Enc}$, $\mathcal{O}.\mathsf{Next}$, $\mathcal{O}.\mathsf{Upd}$ and $\mathcal{O}.\mathsf{Corr}$. At some point $\mathcal{A}$ attempts to provide a ciphertext forgery via the oracle $\mathcal{O}.\mathsf{Try}$ defined in Fig. 4. $\mathcal{A}$ wins the game if its forgery is valid, i.e.,, if it decrypts to a message and not $\perp$. If $\mathcal{A}$ is allowed to ask a single $\mathcal{O}.\mathsf{Try}$ query, we speak of the $\mathsf{INT\text{-}CTXT^s}$ notion. If $\mathcal{A}$ can send multiple $\mathcal{O}.\mathsf{Try}$ queries, we speak of the INT-CTXT notion instead. $\mathsf{INT\text{-}CTXT^s}$ and INT-CTXT are proved to be equivalent in [9, Lemma 1]. Thus, we only define the $\mathsf{INT\text{-}CTXT^s}$ advantage (in Definition 9).

Definition 9 ([9]). *Let* $\mathsf{UE} = \{\mathsf{UE.KeyGen}, \mathsf{UE.TokenGen}, \mathsf{UE.Enc}, \mathsf{UE.Dec}, \mathsf{UE.Upd}\}$ *be an UE scheme. The* $\mathsf{INT\text{-}CTXT^s}$ *advantage of an adversary* $\mathcal{A}$ *against* UE *is defined as*

$$\mathbf{Adv}^{\mathsf{INT\text{-}CTXT^s}}_{\mathsf{UE},\mathcal{A}}(\lambda) := \Pr[\mathbf{Exp}^{\mathsf{INT\text{-}CTXT^s}}_{\mathsf{UE},\mathcal{A}} = 1]$$

where the experiment $\mathbf{Exp}^{\mathsf{INT\text{-}CTXT^s}}_{\mathsf{UE},\mathcal{A}}$ *is given in Fig. 5.*

Setup(1^λ)

1. $\mathsf{pp} \leftarrow \mathsf{UE.Setup}(1^\lambda)$
2. $k_0 \leftarrow \mathsf{UE.KeyGen}(\mathsf{pp})$
3. $\Delta_0 \leftarrow \perp$
4. $\mathsf{e}, \mathsf{c} \leftarrow 0$
5. $\mathsf{phase}, \mathsf{twf} \leftarrow 0$
6. $\mathcal{L}, \tilde{\mathcal{L}}, \mathcal{C}, \mathcal{K}, \mathcal{T} \leftarrow \emptyset$

$\mathcal{O}.\mathsf{Enc}(M)$

1. $C \leftarrow \mathsf{UE.Enc}(k_\mathsf{e}, M)$
2. $\mathsf{c} \leftarrow \mathsf{c} + 1$
3. $\mathcal{L} \leftarrow \mathcal{L} \cup \{(\mathsf{c}, C, \mathsf{e})\}$
4. **return** C

$\mathcal{O}.\mathsf{Dec}(C)$

1. **if** $\mathsf{phase} = 1$ **and** $C \in \tilde{\mathcal{L}}$
2. $\quad \mathsf{twf} \leftarrow 1$
3. M **or** $\perp \leftarrow \mathsf{UE.Dec}(k_\mathsf{e}, C)$
4. **return** M **or** $\perp$

$\mathcal{O}.\mathsf{Next}$

1. $\mathsf{e} \leftarrow \mathsf{e} + 1$
2. $k_\mathsf{e} \leftarrow \mathsf{UE.KeyGen}(\mathsf{pp})$
3. $\Delta_\mathsf{e} \leftarrow \mathsf{UE.TokenGen}(k_{\mathsf{e}-1}, k_\mathsf{e})$
4. **if** $\mathsf{phase} = 1$
5. $\quad \tilde{C}_\mathsf{e} \leftarrow \mathsf{UE.Upd}(\Delta_\mathsf{e}, \tilde{C}_{\mathsf{e}-1})$

$\mathcal{O}.\mathsf{Upd}(C_{\mathsf{e}-1})$

1. **if** $(j, C_{\mathsf{e}-1}, \mathsf{e}-1) \notin \mathcal{L}$
2. $\quad$ **return** $\perp$
3. $C_\mathsf{e} \leftarrow \mathsf{UE.Upd}(\Delta_\mathsf{e}, C_{\mathsf{e}-1})$
4. $\mathcal{L} \leftarrow \mathcal{L} \cup \{(j, C_\mathsf{e}, \mathsf{e})\}$
5. **return** C_e

$\mathcal{O}.\mathsf{Corr}(\mathsf{inp}, \hat{\mathsf{e}})$

1. **if** $\hat{\mathsf{e}} > \mathsf{e}$
2. $\quad$ **return** $\perp$
3. **if** $\mathsf{inp} = \mathsf{key}$
4. $\quad \mathcal{K} \leftarrow \mathcal{K} \cup \{\hat{\mathsf{e}}\}$
5. $\quad$ **return** $k_{\hat{\mathsf{e}}}$
6. **if** $\mathsf{inp} = \mathsf{token}$
7. $\quad \mathcal{T} \leftarrow \mathcal{T} \cup \{\hat{\mathsf{e}}\}$
8. $\quad$ **return** $\Delta_{\hat{\mathsf{e}}}$

$\mathcal{O}.\mathsf{Chall}(\bar{M}, \bar{C})$

1. **if** $\mathsf{phase} \neq 1$
2. $\quad$ **return** $\perp$
3. $\mathsf{phase} \leftarrow 1$
4. $\bar{\mathsf{e}} \leftarrow \mathsf{e}$
5. **if** $(\cdot, \bar{C}, \mathsf{e}-1) \notin \mathcal{L}$
6. $\quad$ **return** $\perp$
7. **if** $b = 0$
8. $\quad \tilde{C}_\mathsf{e} \leftarrow \mathsf{UE.Enc}(k_\mathsf{e}, \bar{M})$
9. **else** $(b = 1)$
10. $\quad \tilde{C}_\mathsf{e} \leftarrow \mathsf{UE.Upd}(\Delta_\mathsf{e}, \bar{C})$
11. $\mathcal{C} \leftarrow \mathcal{C} \cup \{\mathsf{e}\}$
12. $\tilde{\mathcal{L}} \leftarrow \tilde{\mathcal{L}} \cup \{(\tilde{C}_\mathsf{e}, \mathsf{e})\}$
13. **return** $\tilde{C}_\mathsf{e}$

$\mathcal{O}.\mathsf{Upd}\tilde{\mathsf{C}}$

1. **if** $\mathsf{phase} \neq 1$
2. $\quad$ **return** $\perp$
3. $\mathcal{C} \leftarrow \mathcal{C} \cup \{\mathsf{e}\}$
4. $\tilde{\mathcal{L}} \leftarrow \tilde{\mathcal{L}} \cup \{(\tilde{C}_\mathsf{e}, \mathsf{e})\}$
5. **return** $\tilde{C}_\mathsf{e}$

Fig. 3. Oracles in security games for UE with deterministic updates. Computing the leakage sets is discussed in sec. 2.2

$\mathcal{O}.\mathsf{Try}(\tilde{C})$

1. **if** $\mathsf{phase} = 1$
2. $\quad$ **return** $\perp$
3. $\mathsf{phase} \leftarrow 1$
4. **if** $\mathsf{e} \in \mathcal{K}^*$ **or** $\tilde{C} \in \mathcal{L}^*$
5. $\quad \mathsf{twf} \leftarrow 1$
6. M **or** $\perp \leftarrow \mathsf{UE.Dec}(k_\mathsf{e}, \tilde{C})$
7. **if** $M \neq \perp$
8. $\quad \mathsf{win} \leftarrow 1$

Fig. 4. The oracle $\mathcal{O}.\mathsf{Try}$ for the $\mathsf{INT\text{-}CTXT^s}$ security notion.

$\mathbf{Exp}^{\mathsf{INT\text{-}CTXT^s}}_{\mathsf{UE},\mathcal{A}}(\lambda)$

1. **do** $\mathsf{UE.Setup}(1^\lambda)$
2. $\mathsf{win} \leftarrow 0$
3. $\mathcal{A}^{\mathcal{O}.\mathsf{Enc},\mathcal{O}.\mathsf{Next},\mathcal{O}.\mathsf{Upd},\mathcal{O}.\mathsf{Corr},\mathcal{O}.\mathsf{Try}}(\lambda)$
4. **if** $\mathsf{twf} = 1$
5. $\quad \mathsf{win} \leftarrow 0$
6. **return** win

Fig. 5. The $\mathsf{INT\text{-}CTXT^s}$ experiment for UE scheme UE and adversary $\mathcal{A}$. Trivial win conditions are discussed in Section 2.2.

Composition result for CPA, CTXT and CCA security. In [9, Theorem 3], Boyd *et al.* show the following generic composition result for UE: CPA + CTXT ⇒ CCA. We will use this result to prove that our GAINE scheme can be made detIND-UE-CCA secure. This is the first post-quantum UE scheme to attain this security notion.

Leakage sets. We follow the bookkeeping technique [25,9] to maintain the epoch leakage sets.

- $\mathcal{C}$: List of epochs in which the adversary learned an updated version of the challenge ciphertext (from $\mathcal{O}.\mathsf{Chall}$ or $\mathcal{O}.\mathsf{Upd}\tilde{\mathsf{C}}$).
- $\mathcal{K}$: List of epochs in which the adversary corrupted the encryption key.
- $\mathcal{T}$: List of epochs in which the adversary corrupted the update token.

The adversary can also learn the values of ciphertexts and their updates.

- $\mathcal{L}$: List of non-challenge ciphertexts (from $\mathcal{O}.\mathsf{Enc}$ or $\mathcal{O}.\mathsf{Upd}$) with entries of the form $(\mathsf{c}, C, \mathsf{e})$, where c is a counter incremented with each $\mathcal{O}.\mathsf{Enc}$ query.
- $\tilde{\mathcal{L}}$: List of updated versions of challenge ciphertext (created by $\mathcal{O}.\mathsf{Next}$ and returned by $\mathcal{O}.\mathsf{Upd}\tilde{\mathsf{C}}$), with entries of the form $(\tilde{C}, \mathsf{e})$.

Trivial wins via keys and ciphertexts. We consider the extended epoch leakage sets $\mathcal{C}^*$, $\mathcal{K}^*$ and $\mathcal{T}^*$ inferred from $\mathcal{C}$, $\mathcal{K}$ and $\mathcal{T}$. These extended sets are used to identify trivial wins, i.e., if $\mathcal{C}^* \cap \mathcal{K}^* \neq \emptyset$, then there exists an epoch in which the adversary knows the epoch key and a valid update of the challenge ciphertext. The challenger computes these sets once the adversary has finished running. Using [25], we show how to compute the extended epoch leakage sets $\mathcal{C}^*$, $\mathcal{K}^*$ and $\mathcal{T}^*$:

$$
\begin{aligned}
&\mathcal{K}^* \leftarrow \{\mathsf{e} \in \{0,\dots,n\} \mid \mathsf{CorrK}(\mathsf{e}) = \mathsf{true}\} \\
&\quad \mathsf{true} \leftarrow \mathsf{CorrK}(\mathsf{e}) \Leftrightarrow (\mathsf{e} \in \mathcal{K}) \vee (\mathsf{CorrK}(\mathsf{e}-1) \wedge \mathsf{e} \in \mathcal{T}) \vee (\mathsf{CorrK}(\mathsf{e}+1) \wedge \mathsf{e}+1 \in \mathcal{T}) \\
&\mathcal{T}^* \leftarrow \{\mathsf{e} \in \{0,\dots,n\} \mid (\mathsf{e} \in \mathcal{T}) \vee (\mathsf{e} \in \mathcal{K}^* \wedge \mathsf{e}-1 \in \mathcal{K}^*)\} \\
&\mathcal{C}^* \leftarrow \{\mathsf{e} \in \{0,\dots,n\} \mid \mathsf{ChallEq}(\mathsf{e}) = \mathsf{true}\} \\
&\quad \mathsf{true} \leftarrow \mathsf{ChallEq}(\mathsf{e}) \Leftrightarrow (\mathsf{e} = \tilde{\mathsf{e}}) \vee (\mathsf{e} \in \mathcal{C}) \vee (\mathsf{ChallEq}(\mathsf{e}-1) \wedge \mathsf{e} \in \mathcal{T}^*) \vee \\
&\qquad\qquad\qquad (\mathsf{ChallEq}(\mathsf{e}+1) \wedge \mathsf{e}+1 \in \mathcal{T}^*)
\end{aligned}
$$

Trivial wins via direct updates Define $\mathcal{I}$ as the set of epochs in which the adversary learned an updated version of the ciphertext given as challenge input ($\bar{C}$). Furthermore, define $\mathcal{I}^*$ to be the extended set in which the adversary has inferred information via token corruption. Since, in our case, the algorithm Upd is deterministic, an updated ciphertext is uniquely determined by a token and a ciphertext. Thus, the adversary trivially wins if $\mathcal{I}^* \cap \mathcal{C}^* \neq \emptyset$. Indeed, there exists an epoch in which the adversary knows the updated ciphertext of the challenge input $\bar{C}$ and a valid challenge-equal ciphertext. Comparing them allows the adversary to win the game.

In [9], $\mathcal{I}$ is computed by finding an entry in $\mathcal{L}$ that contains the challenge input $\bar{C}$. Then, note the query identifier c for that entry and scan $\mathcal{L}$ for other entries with this identifier $\mathcal{I} := \{\mathsf{e} \in \{0,\dots,n\} \mid (\mathsf{c}, \cdot, \mathsf{e}) \in \mathcal{L}\}$. We extend $\mathcal{I}$ into $\mathcal{I}^*$:

$$
\begin{aligned}
&\mathcal{I}^* \leftarrow \{\mathsf{e} \in \{0,\dots,n\} \mid \mathsf{ChallInpEq}(\mathsf{e}) = \mathsf{true}\} \\
&\quad \mathsf{true} \leftarrow \mathsf{ChallInpEq}(\mathsf{e}) \Leftrightarrow (\mathsf{e} \in \mathcal{I}) \vee (\mathsf{ChallInpEq}(\mathsf{e}-1) \wedge \mathsf{e} \in \mathcal{T}^*) \vee \\
&\qquad\qquad\qquad (\mathsf{ChallInpEq}(\mathsf{e}+1) \wedge \mathsf{e}+1 \in \mathcal{T}^*)
\end{aligned}
$$

Trivial wins in ciphertext integrity games. Recall that our UE schemes all have deterministic updates, we follow the analysis of [9,22]. The adversary can corrupt an epoch key and use it to forge ciphertexts in this epoch. Thus, we exclude this trivial win by setting twf to 1 when the adversary provides a forgery in an epoch in $\mathcal{K}^*$.

Next, suppose that the adversary knows a ciphertext $(C, \mathsf{e}_1) \in \mathcal{L}$ and tokens from epoch $\mathsf{e}_1 + 1$ to epoch e_2. Then, updating C to epoch e_2 provides a forgery in epoch e_2. We exclude this trivial wins by defining $\mathcal{L}^*$ to be the extended set of $\mathcal{L}$ in which the adversary has learned or inferred information via token corruption. If $\mathcal{O}.\mathsf{Try}$ receives a ciphertext of $\mathcal{L}^*$, it sets twf to 1. We give an algorithm of [9] to compute $\mathcal{L}^*$ during the game in Fig. 6.

Update $\mathcal{L}^*$

1. **if** $\mathcal{O}.\mathsf{Enc}$ **or** $\mathcal{O}.\mathsf{Upd}$ happens
2. $\quad \mathcal{L}^* \leftarrow \mathcal{L}^* \cup \{(\cdot, C, \cdot)\}$
3. **if** $\mathcal{O}.\mathsf{Corr}(\mathsf{token}, \cdot)$ happens
4. $\quad$ **for** $i \in \mathcal{T}^*$
5. $\quad\quad$ **for** $(j, C_{i-1}, i-1) \in \mathcal{L}^*$
6. $\quad\quad\quad C_i \leftarrow \mathsf{UE.Upd}(\Delta_i, C_{i-1})$
7. $\quad\quad\quad \mathcal{L}^* \leftarrow \mathcal{L}^* \cup \{(j, C_i, i)\}$

Fig. 6. Update procedure of [9] for list $\mathcal{L}^*$.

2.3 Elliptic curves and isogenies

Let K be a field. An elliptic curve over K is a smooth projective curve of genus 1 with a distinguished base point defined over K. For our purpose, it is enough to consider elliptic curves as curves defined by an equation of the form $E : y^2 = x^3 + ax + b$. The set of points $E(K)$ is made of the solutions (x, y) to the equation of the curve. A good generic reference on elliptic curves is the book from Silverman [36]. The set of points is a group for a law that we write $\oplus$. The scalar multiplication $[n] : E(K) \to E(K)$ is the operation $\oplus$ iterated n times. We write $E[n]$ for the kernel of the morphism $[n]$.

An isogeny $\varphi : E_1 \to E_2$ is a non-constant morphism sending the identity of E_1 to that of E_2. The degree of an isogeny is its degree as a rational map. When the degree $\deg(\varphi) = d$ is coprime to p, the isogeny is necessarily *separable*. An isogeny induces a homomorphism of groups $E_1(K) \to E_2(K)$ and, if separable, the kernel of φ is a group of order d. Such an isogeny is entirely described by its kernel, meaning that there is a one-to-one correspondence between separable isogenies (up to an isomorphism of the target curve) and finite subgroups of $E(\overline{K})$. An isogeny can be computed from its kernel G using Vélu's formula [38], in this case we write $\varphi : E \to E/G$. The degree of $\varphi \circ \psi$ is equal to $\deg(\varphi)\deg(\psi)$. For any isogeny $\varphi : E_1 \to E_2$, there exists a unique dual isogeny $\hat{\varphi} : E_2 \to E_1$, satisfying $\varphi \circ \hat{\varphi} = [\deg(\varphi)]$, the multiplication-by-$\deg(\varphi)$ map on E_2. Similarly $\hat{\varphi} \circ \varphi$ is the multiplication-by-$\deg(\varphi)$ map on E_1. An endomorphism of E is an isogeny $\theta : E \to E$. The set of endomorphisms is a ring with addition and composition that we write $\mathrm{End}(E)$.

Let us take K, a finite field of characteristic p. Over K, a curve is said to be *supersingular* when $\mathrm{End}(E)$ is a maximal order inside the quaternion algebra ramified at p and ∞. The Frobenius morphism is defined as $\pi : (x, y) \to (x^p, y^p)$, it sends any curve $E : y^2 = x^3 + ax + b$ to $E^{(p)} : y^2 = x^3 + a^p x + b^p$. π is the only isogeny of degree p between any two supersingular curves.

3 UE from group action

We generalize the SHINE scheme of Boyd *et al.* [9] using the framework of cryptographic group actions of Alamati *et al.* [1].

3.1 Generalizing SHINE to group actions

First, we introduce the novel *mappable* EGA (MEGA) definition. This new notion can be seen as a strengthened hashable group action (see [1]) as we will prove in Appendix B.

Definition 10 (Mappable EGA). *Let $(G, S, \star)$ be an EGA. We say that $(G, S, \star)$ is a mappable EGA if there exists an efficient bijection $\pi : \{0,1\}^N \to S$.*

Let $\mathcal{GA}$ be a MEGA family and let $(G, S, \star)$ be $\mathcal{GA}(\lambda)$: a MEGA with bijection $\pi : \{0,1\}^{m+v} \to S$, for integers m and v. Let $\mathcal{M} := \{0,1\}^m$ be the message space and $\mathcal{N} := \{0,1\}^v$ be the nonce space.

We present our generalization of the SHINE scheme [9] to group actions, which we call GAINE for Group Action Ideal-cipher Nonce-based Encryption, in figure 7. We use group elements as keys and set elements as messages. Encryption, decryption and updates boil down to a group action computation. Ciphertexts are randomized by adding a random nonce as input to π. The group action used in the original proposal of Boyd *et al.* [9] is the action of the group $\mathbb{Z}_q^*$ on S by exponentiation, where S is a cyclic group of prime order q in which the discrete logarithm problem is hard.

$\mathsf{Setup}(1^\lambda)$:

1. $(G, S, \star) \leftarrow \mathcal{GA}(\lambda)$
2. Choose π, m, v as above
3. $\mathsf{pp} \leftarrow (G, S, \star), \pi, m, v$
4. **return** pp

$\mathsf{KeyGen}(\mathsf{pp})$:

1. $k \xleftarrow{\$} G$.
2. **return** k.

$\mathsf{TokenGen}(k_\mathsf{e}, k_{\mathsf{e}+1})$:

1. $\Delta_{\mathsf{e}+1} \leftarrow k_{\mathsf{e}+1} k_\mathsf{e}^{-1}$
2. **return** $\Delta_{\mathsf{e}+1}$

$\mathsf{Upd}(\Delta_{\mathsf{e}+1}, C_\mathsf{e})$:

1. $C_{\mathsf{e}+1} \leftarrow \Delta_{\mathsf{e}+1} \star C_\mathsf{e}$
2. **return** $C_{\mathsf{e}+1}$

$\mathsf{Enc}(k_\mathsf{e}, M)$:

1. $N \xleftarrow{\$} \mathcal{N}$.
2. $C_\mathsf{e} \leftarrow k_\mathsf{e} \star \pi(N \| M)$
3. **return** C_e

$\mathsf{Dec}(k_\mathsf{e}, C_\mathsf{e})$:

1. $s \leftarrow \pi^{-1}(k_\mathsf{e}^{-1} \star C_\mathsf{e})$
2. Parse s as $N' \| M'$
3. **return** M'

Fig. 7. GAINE: our generalization of the SHINE scheme using group actions.

GAINE is correct, even for non-abelian group actions.

Proposition 1 (Correctness of updates). *Let $k_\mathsf{e}, k_{\mathsf{e}+1} \in G$ be two keys and $C_\mathsf{e} := k_\mathsf{e} \star s$ for some $s \in S$. If $\Delta_{\mathsf{e}+1} := \mathsf{TokenGen}(k_\mathsf{e}, k_{\mathsf{e}+1})$, then $\mathsf{Upd}(\Delta_{\mathsf{e}+1}, C_\mathsf{e}) = k_{\mathsf{e}+1} \star s$.*

Proof. By definition of TokenGen, we have $\Delta_{\mathsf{e}+1} := k_{\mathsf{e}+1} k_\mathsf{e}^{-1}$. By definition of Upd, we have

$$\mathsf{Upd}(\Delta_{\mathsf{e}+1}, C_\mathsf{e}) = \Delta_{\mathsf{e}+1} \star C_\mathsf{e} = (k_{\mathsf{e}+1} k_\mathsf{e}^{-1}) \star (k_\mathsf{e} \star s) = k_{\mathsf{e}+1} \star s$$

where the last equality holds because of the *compatibility* of the group action (see Item 2 of Definition 1) and because k_e^{-1} is the inverse of k_e in the group G.

Proposition 2 (Correctness). *The GAINE scheme is correct.*

Proof. Let $0 \leq \mathsf{e}_1 \leq \mathsf{e}_2 \leq n+1$ be two epochs and let us consider a ciphertext C_{e_2} updated through the successive tokens Δ_{i+1} for $i \in [\mathsf{e}_1, \mathsf{e}_2 - 1]$ from an initial ciphertext C_{e_1} that is the encryption of a message m under the key $k_{\mathsf{e}-1}$ as in Definition 7. By definition of $C_{\mathsf{e}_1} := \mathsf{Enc}(k_{\mathsf{e}_1}, M)$, we have $C_{\mathsf{e}_1} = k_{\mathsf{e}_1} \star \pi(N \| M)$ for

some random nonce N. By applying Proposition 1 on the updates of C_{e_1}, we have that $C_{\mathsf{e}_2} = k_{\mathsf{e}_2} \star \pi(N\|M)$. Then, we get

$$k_{\mathsf{e}_2}^{-1} \star C_{\mathsf{e}_2} = k_{\mathsf{e}_2}^{-1} \star (k_{\mathsf{e}_2} \star \pi(N\|M)) = 1_G \star \pi(N\|M) = \pi(N\|M)$$

where the penultimate equality holds because of the *compatibility* of the group action (see Item 2 of Definition 1) and the last equality holds because of the *identity* property of the group action (see Item 1 of Definition 1). Finally, we have $\pi^{-1}(k_{\mathsf{e}_2}^{-1} \star C_{\mathsf{e}_2}) = N\|M$ and $\mathsf{Dec}(k_{\mathsf{e}_2}, C_{\mathsf{e}_2})$ returns M.

3.2 Security - GAINE is detIND-UE-CPA secure

In theorem 1, we show that GAINE is detIND-UE-CPA in the ideal cipher model, if the group action is a weakly pseudorandom MEGA. The ideal cipher model, introduced by Shannon [35] and shown to be equivalent to the random oracle model by Coron *et al.* [14], gives all parties access to a permutation chosen randomly from all possible key permutations of appropriate length. The GAINE scheme acts on the outputs of the permutation with the epoch key to encrypt, so our reduction can "program" the transformation from permutation outputs to set elements.

Theorem 1 (GAINE is detIND-UE-CPA). *Let* GAINE *be the UE scheme described in figure 7 for a MEGA family $\mathcal{GA}$. For any ideal cipher model adversary $\mathcal{A}$, there exists a reduction $\mathcal{B}$ such that*

$$\mathbf{Adv}^{\mathsf{detIND\text{-}UE\text{-}CPA}}_{\mathsf{GAINE},\mathcal{A}}(\lambda) \leq \mathcal{O}(1)(n+1)^3 \cdot \mathbf{Adv}^{\mathsf{wk\text{-}PR}}_{\mathcal{GA},\mathcal{B}}(\lambda)$$

We follow the proof strategy of [9] and use their hybrid argument across insulated regions. In each hybrid, we can embed at one firewall of the insulated region and simulate all tokens within that insulated region to answer queries to both $\mathcal{O}.\mathsf{Upd}$ and $\mathcal{O}.\mathsf{Upd}\tilde{\mathsf{C}}$. In GAINE, we update a ciphertext from epoch e to epoch $\mathsf{e}+1$ by computing the action of the group element $k_{\mathsf{e}+1}k_{\mathsf{e}}^{-1}$. Fresh ciphertexts are randomized using a nonce N but updates are deterministic, thus our reduction will need to provide consistent ciphertexts to the adversary, i.e., the N value must be consistent.

We give a reduction $\mathcal{B}$ which receives a group action $(G, S, \star)$ and an oracle $\mathcal{O}.\mathsf{Sample}$ that returns either tuples of the form $(s_i, g \star s_i)$ or (s_i, u_i) where $g \xleftarrow{\$} G$ and $s_i, u_i \xleftarrow{\$} S$. $\mathcal{B}$ will use the tuples of $\mathcal{O}.\mathsf{Sample}$ to perfectly simulate the detIND-UE-CPA experiment for GAINE when those tuples are of the form $(s_i, g \star s_i)$ (and a random experiment otherwise). The idea is to embed g to a well chosen epoch key by using s_i as randomness and $g \star s_i$ as ciphertext value. Thus, if we know an efficient adversary $\mathcal{A}$ against the detIND-UE-CPA security of GAINE, using the hybrid argument of [9], $\mathcal{B}$ can use $\mathcal{A}$ to break the weak pseudorandomness of $(G, S, \star)$.

Proof. Play hybrid games. We partition the non corrupted key space as follows:
$\{0, \ldots, n\} \setminus \mathcal{K}^* = \cup_{(j,\mathsf{fwl}_j,\mathsf{fwr}_j)\in\mathcal{FW}}\{\mathsf{fwl}_j \ldots \mathsf{fwr}_j\}$, where fwl_i and fwr_i are firewalls of the i-th insulated region. For $b \in \{0,1\}$, define game $\mathcal{G}_i^b$ as $\mathbf{Exp}^{\mathsf{detIND\text{-}UE\text{-}CPA}\text{-}b}_{\mathsf{GAINE},\mathcal{A}}$ except for:

1. The game randomly picks $\mathsf{fwl}_i, \mathsf{fwr}_i \xleftarrow{\$} \{0, \ldots, n\}$ and if they are not the i-th firewalls, it aborts and returns a random bit b'. This loss is upper-bounded by $(n+1)^2$.
2. For the challenge (made in epoch $\tilde{\mathsf{e}}$ on input $(\bar{M}, \bar{C})$), the game returns an updated version of $\bar{C}$ if $\tilde{\mathsf{e}} < \mathsf{fwl}_i$ and it returns an encryption of $\bar{M}$ if $\tilde{\mathsf{e}} > \mathsf{fwr}_i$. Finally, if $\mathsf{fwl}_i \leq \tilde{\mathsf{e}} \leq \mathsf{fwr}_i$, the game returns an encryption of $\bar{M}$ if $b = 0$ and an updated version of $\bar{C}$ if $b = 1$.
3. After $\mathcal{A}$ outputs b', the game returns b' if $\mathsf{twf} \neq 1$ or some additional trivial win condition triggers.

If fwl_i, fwr_i are the desired values, then $\mathcal{G}_1^0$ is $\mathbf{Exp}^{\mathsf{detIND\text{-}UE\text{-}CPA}\text{-}0}_{\mathsf{GAINE},\mathcal{A}}$, i.e., all challenges are encryptions of $\bar{M}$. Let ℓ be the total number of insulated regions (bounded by $n+1$), such that $\mathcal{G}_\ell^1$ is $\mathbf{Exp}^{\mathsf{detIND\text{-}UE\text{-}CPA}\text{-}1}_{\mathsf{GAINE},\mathcal{A}}$, i.e., all challenges are updates of $\bar{C}$. Let E be the event that fwl_i and fwr_i are the desired values. By definition,

for any $1 \le i \le n+1$ and $b \in \{0,1\}$, we have $\Pr[\mathcal{G}_i^b = 1 \mid \neg E] = 1/2$. Then

$$\begin{aligned}\Pr[\mathcal{G}_\ell^1 = 1] &= \Pr[\mathcal{G}_\ell^1 = 1 \mid E] \cdot \Pr[E] + \Pr[\mathcal{G}_\ell^1 = 1 \mid \neg E] \cdot \Pr[\neg E] \\ &= \Pr[\mathbf{Exp}^{\mathsf{detIND\text{-}UE\text{-}CPA\text{-}1}}_{\mathsf{GAINE},\mathcal{A}} = 1] \cdot \frac{1}{(n+1)^2} + \frac{1}{2} \cdot (1 - \frac{1}{(n+1)^2}), \text{and} \\ \Pr[\mathcal{G}_1^0 = 1] &= \Pr[\mathbf{Exp}^{\mathsf{detIND\text{-}UE\text{-}CPA\text{-}0}}_{\mathsf{GAINE},\mathcal{A}} = 1] \cdot \frac{1}{(n+1)^2} + \frac{1}{2} \cdot (1 - \frac{1}{(n+1)^2})\end{aligned}$$

Thus, we have $|\Pr[\mathcal{G}_\ell^1 = 1] - \Pr[\mathcal{G}_1^0 = 1]| = \frac{1}{(n+1)^2} \cdot \mathbf{Adv}^{\mathsf{detIND\text{-}UE\text{-}CPA}}_{\mathsf{GAINE},\mathcal{A}}(\lambda)$.

Notice that the games $\mathcal{G}_{i-1}^1$ and $\mathcal{G}_i^0$ behave in the same way: for the challenge query and $\mathcal{O}.\mathsf{Upd}\tilde{\mathsf{C}}$, in an epoch in the first $i-1$ insulated regions, the reduction returns an update of $\bar{C}$, otherwise it returns an encryption of $\bar{M}$. Thus, for any $\ell \le n+1$, $|\Pr[\mathcal{G}_\ell^1 = 1] - \Pr[\mathcal{G}_1^0 = 1]| \le \sum_{i=1}^{\ell} |\Pr[\mathcal{G}_i^1 = 1] - \Pr[\mathcal{G}_i^0 = 1]|$. In the following, we prove that for any $1 \le i \le \ell$, $|\Pr[\mathcal{G}_i^1 = 1] - \Pr[\mathcal{G}_i^0 = 1]| \le 2\mathbf{Adv}^{\mathsf{wk\text{-}PR}}_{\mathcal{G},\mathcal{A},\mathcal{B}}(\lambda)$ for a reduction $\mathcal{B}$.

In hybrid i. Let $\mathcal{A}_i$ be an adversary trying to distinguish $\mathcal{G}_i^0$ from $\mathcal{G}_i^1$. For all queries concerning epochs outside of the i-th insulated region, the responses of both games are the same. Thus, we assume that $\mathcal{A}_i$ asks for at least one challenge ciphertext in an epoch within the i-th insulated region. This is where we will embed the weak pseudorandom group action samples in our reduction.

We construct a reduction $\mathcal{B}$, presented in Fig. 8, that is playing the weak pseudorandom group action game (Definition 4) and will simulate the responses of queries made by adversary $\mathcal{A}_i$. Since we do not assume the group action $(G, S, \star)$ to be abelian, we define $(\prod_{i=0}^{n} g_i) \star s := (g_0 g_1 \dots g_n) \star s$ for $s \in S$ and $g_0, \dots, g_n \in G$.

Recall that $\mathcal{A}_i$ is an adversary attempting to distinguish $\mathcal{G}_i^0$ from $\mathcal{G}_i^1$. $\mathcal{B}$ will try to use $\mathcal{A}$ to break the weak pseudorandomness of the group action $(G, S, \star)$. In $\mathbf{Exp}^{\mathsf{wk\text{-}PR}}_{\mathcal{G},\mathcal{A},\mathcal{B}}(\lambda)$, when $\mathcal{O}.\mathsf{Sample}$ returns pairs of the form $(s_j, g \star s_j)$ for $g \xleftarrow{\$} G$ and $s_j \xleftarrow{\$} S$, $\mathcal{B}$ will perfectly simulate the environment of $\mathcal{A}_i$ in $\mathcal{G}_i^b$. When $\mathcal{O}.\mathsf{Sample}$ returns pairs of the form (s_j, t_j) for $s_j, t_j \xleftarrow{\$} S$, $\mathcal{B}$ will give random inputs to $\mathcal{A}_i$ such that $\mathcal{A}_i$ distinguishes $\mathcal{G}_i^0$ from $\mathcal{G}_i^1$ with advantage 0. We explain how our reduction $\mathcal{B}$ does this without knowing which $\mathcal{O}.\mathsf{Sample}$ oracle was provided to it.

The reduction $\mathcal{B}$ receives the oracle $\mathcal{O}.\mathsf{Sample}$, takes $b \xleftarrow{\$} \{0,1\}$ and simulates $\mathcal{G}_i^b$. Whenever the reduction needs to provide an output of $\pi(\cdot)$ to $\mathcal{A}_i$, it chooses some set value $s \in S$ such that $\pi(\cdot) = s$. In this setting, computing π^{-1} is simply a lookup to this mapping of the ideal cipher π. We explain our simulation:

Initially,

1. $\mathcal{B}$ guesses the values of fwl_i and fwr_i.
2. $\mathcal{B}$ generates all keys and tokens 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 these keys and tokens, this means that the firewall guess is wrong and the reduction aborts the game using the Check algorithm of Appendix A.

$\mathcal{B}$ will operate so as to embed the value g used by $\mathcal{O}.\mathsf{Sample}$ to the key k_{fwl_i} and the value $g k_{\mathsf{fwl}_i - 1}^{-1}$ to the token Δ_{fwl_i}. If $\mathcal{O}.\mathsf{Sample}$ returns uniformly distributed pairs of set elements instead, all the ciphertexts inside insulated region i will be random set elements (no key or token could possibly explain these ciphertexts).

To simulate a non-challenge ciphertext that is:

- An $\mathcal{O}.\mathsf{Enc}$ query in epoch $\mathsf{e} \in \{0, \dots \mathsf{fwl}_i - 1\} \cup \{\mathsf{fwr}_i + 1, \dots, n\}$: $\mathcal{B}$ queries $\mathcal{O}.\mathsf{Sample}$ to get a pair $(s,t) \in S^2$. $\mathcal{B}$ uses s as a random value by programming $\pi(\cdot) \leftarrow s$ (so the randomness will be consistent with calls that $\mathcal{A}_i$ makes to $\mathcal{O}.\mathsf{Upd}$), computes the ciphertext $C_\mathsf{e} = k_\mathsf{e} \star s$ (the value of k_e is known to $\mathcal{B}$ in these epochs) and stores (s,t) in its memory for later use. To respond to $\mathcal{O}.\mathsf{Upd}$ queries in these epochs, $\mathcal{B}$ computes $C_\mathsf{e} = k_\mathsf{e} \star s$ using the randomness s generated during the first encryption of the input ciphertext.
- An $\mathcal{O}.\mathsf{Enc}$ query in epoch $\mathsf{e} \in \{\mathsf{fwl}_i, \dots, \mathsf{fwr}_i\}$: $\mathcal{B}$ queries $\mathcal{O}.\mathsf{Sample}$ to get a pair $(s,t) \in S^2$ and programs $\pi(\cdot) \leftarrow s$. It sets $C_{\mathsf{fwl}_i} = t$ (so that all ciphertexts will be encrypted under the key g in epoch fwl_i if $\mathcal{O}.\mathsf{Sample}$ returns pairs of the form $(s_j, g \star s_j)$) and updates C_{fwl_i} to the right epoch e using its simulated tokens (remember that $\mathcal{B}$ does not know the keys inside the i-th insulated region). To respond to $\mathcal{O}.\mathsf{Upd}$

Reduction $\mathcal{B}$ playing $\mathbf{Exp}^{\mathsf{wk\text{-}PR}\text{-}b^*}_{\mathcal{GA},\mathcal{B}}(\lambda)$

receive $(G, S, \star)$ and $\mathcal{O}.\mathsf{Sample}$
do $\mathbf{Setup}(1^\lambda)$
$\bar{M}, \bar{C} \leftarrow \mathcal{A}^{\mathtt{ors}}(\lambda)$
$\mathsf{phase} \leftarrow 1$
$\tilde{C}_{\tilde{\mathsf{e}}} \leftarrow \mathcal{O}.\mathsf{Chall}(\bar{M}, \bar{C})$
$b' \leftarrow \mathcal{A}^{\mathtt{ors}, \mathcal{O}.\mathsf{Upd}\tilde{\mathsf{C}}}(\tilde{C}_{\tilde{\mathsf{e}}})$
$\mathsf{twf} \leftarrow 1$ **if**
  $\mathcal{C}^* \cap \mathcal{K}^* \neq \emptyset$ **or**
  $\mathcal{I}^* \cap \mathcal{C}^* \neq \emptyset$
if `ABORT` occurred **or** $\mathsf{twf} = 1$
  $b' \xleftarrow{\$} \{0,1\}$
  return b'
if $(i, \mathsf{fwl}_i, \mathsf{fwr}_i) \notin \mathcal{FW}$
  $b' \xleftarrow{\$} \{0,1\}$
  return b'
if $b' = b$
  return 0
else
  return 1

$\mathbf{Setup}(1^\lambda)$

$b \xleftarrow{\$} \{0,1\}$
$\mathsf{pp} \leftarrow \mathsf{GAINE.Setup}(1^\lambda)$
$k_0 \leftarrow \mathsf{GAINE.KeyGen}(\mathsf{pp})$
$\Delta_0 \leftarrow \perp$
$\mathsf{e}, \mathsf{c}, \mathsf{phase}, \mathsf{twf} \leftarrow 0$
$\mathcal{L}, \tilde{\mathcal{L}}, \mathcal{C}, \mathcal{K}, \mathcal{T} \leftarrow \emptyset$
$\mathsf{fwl}_i, \mathsf{fwr}_i \xleftarrow{\$} \{0, \ldots, n\}$
for $j \in \{0, \ldots, \mathsf{fwl}_i - 1\}$ **do**
  $k_j \xleftarrow{\$} G; \Delta_j \leftarrow k_j k_{j-1}^{-1} \bowtie$
for $j \in \{\mathsf{fwr}_i + 1, \ldots, n\}$ **do**
  $k_j \xleftarrow{\$} G; \Delta_j \leftarrow k_j k_{j-1}^{-1} \bowtie$
for $j \in \{\mathsf{fwl}_i + 1, \ldots, \mathsf{fwr}_i\}$ **do**
  $\Delta_j \xleftarrow{\$} G$

$\mathcal{O}.\mathsf{Enc}(M)$

$\mathsf{c} \leftarrow \mathsf{c} + 1$
$(\mathsf{inf}_1, \mathsf{inf}_2) \leftarrow \mathcal{O}.\mathsf{Sample}()$
$\pi(N \| M) \leftarrow \mathsf{inf}_1$
if $\mathsf{e} \in \{0, \mathsf{fwl}_i - 1\} \cup \{\mathsf{fwr}_i + 1, \ldots, n\}$
  $C_{\mathsf{e}} \leftarrow k_{\mathsf{e}} \star \mathsf{inf}_1$
else
  $C_{\mathsf{fwl}_i} \leftarrow \mathsf{inf}_2$
  for $j \in \{\mathsf{fwl}_i + 1, \ldots, \mathsf{e}\}$ **do**
    $C_j \leftarrow \Delta_j \star C_{j-1}$
$\mathsf{inf} \leftarrow (\mathsf{inf}_1, \mathsf{inf}_2)$
$\mathcal{L} \leftarrow \mathcal{L} \cup \{(\mathsf{c}, C_{\mathsf{e}}, \mathsf{e}; \mathsf{inf})\}$
return C_{e}

$\mathcal{O}.\mathsf{Next}$

$\mathsf{e} \leftarrow \mathsf{e} + 1$

$\mathcal{O}.\mathsf{Upd}(C_{\mathsf{e}-1})$

if $(\mathsf{c}, C_{\mathsf{e}-1}, \mathsf{e} - 1; \mathsf{inf}) \notin \mathcal{L}$
  return $\perp$
if $\mathsf{e} \in \{1, \ldots, \mathsf{fwl}_i - 1\} \cup \{\mathsf{fwr}_i + 1, \ldots, n\}$
  $(\mathsf{inf}_1, \mathsf{inf}_2) \leftarrow \mathsf{inf}$
  $C_{\mathsf{e}} \leftarrow k_{\mathsf{e}} \star \mathsf{inf}_1$
else
  $(\mathsf{inf}_1, \mathsf{inf}_2) \leftarrow \mathsf{inf}$
  $C_{\mathsf{fwl}_i} \leftarrow \mathsf{inf}_2$
  for $j \in \{\mathsf{fwl}_i + 1, \ldots, \mathsf{e}\}$ **do**
    $C_j \leftarrow \Delta_j \star C_{j-1}$
$\mathcal{L} \leftarrow \mathcal{L} \cup \{(\mathsf{c}, C_{\mathsf{e}}, \mathsf{e}; \mathsf{inf})\}$
return C_{e}

$\mathcal{O}.\mathsf{Corr}(\mathsf{inp}, \hat{\mathsf{e}})$

do $\mathsf{Check}(\mathsf{inp}, \hat{\mathsf{e}}; \mathsf{e}; \mathsf{fwl}_i, \mathsf{fwr}_i)$
if $\mathsf{inp} = \mathsf{key}$
  $\mathcal{K} \leftarrow \mathcal{K} \cup \{\hat{\mathsf{e}}\}$
  return $k_{\hat{\mathsf{e}}}$
if $\mathsf{inp} = \mathsf{token}$
  $\mathcal{T} \leftarrow \mathcal{T} \cup \{\hat{\mathsf{e}}\}$
  return $\Delta_{\hat{\mathsf{e}}}$

$\mathcal{O}.\mathsf{Chall}(\bar{M}, \bar{C})$

if $(\mathsf{c}, \bar{C}, \tilde{\mathsf{e}} - 1; \mathsf{inf}) \notin \mathcal{L}$
  return `ABORT`
if $b = 0$
  $(s, t) \leftarrow \mathcal{O}.\mathsf{Sample}()$
  $\pi(N \| \bar{M}) \leftarrow s$
  $\tilde{C}_{\mathsf{fwl}_i} \leftarrow t$
else
  $(\mathsf{inf}_1, \mathsf{inf}_2) \leftarrow \mathsf{inf}$
  $\pi(N \| \bar{M}) \xleftarrow{\$} S$
  $\tilde{C}_{\mathsf{fwl}_i} \leftarrow \mathsf{inf}_2$
for $j \in \{0, \ldots, \mathsf{fwl}_i - 1\}$ **do**
  $\tilde{C}_j \leftarrow (\prod_{k=j}^{1} \Delta_k)(\prod_{k=1}^{\tilde{\mathsf{e}}-1} \Delta_k^{-1}) \star \bar{C}$ `//left`
for $j \in \{\mathsf{fwl}_i + 1, \ldots, \mathsf{fwr}_i\}$ **do**
  $\tilde{C}_j \leftarrow \Delta_j \star \tilde{C}_{j-1}$ `//embed`
for $j \in \{\mathsf{fwr}_i + 1, \ldots, n\}$ **do**
  $\tilde{C}_j \leftarrow k_j \star \pi(N \| \bar{M})$ `//right`
$\tilde{\mathcal{L}} \leftarrow \cup_{j=0}^{n} \{(\tilde{C}_j, j)$
return $\tilde{C}_{\mathsf{e}}$

$\mathcal{O}.\mathsf{Upd}\tilde{\mathsf{C}}$

$\mathcal{C} \leftarrow \mathcal{C} \cup \{\mathsf{e}\}$
find $(\tilde{C}_{\mathsf{e}}, \mathsf{e}) \in \tilde{\mathcal{L}}$
return $\tilde{C}_{\mathsf{e}}$

Fig. 8. Our reduction $\mathcal{B}$ for proof of th. 1 in hybrid i. inf encodes fixed programming information: it marks two set elements $(\mathsf{inf}_1, \mathsf{inf}_2)$ sampled with $\mathcal{O}.\mathsf{Sample}$. inf_1 is the randomness used during encryption and inf_2 is the ciphertext value in epoch fwl_i. `ors` refers to the set $\{\mathcal{O}.\mathsf{Enc}, \mathcal{O}.\mathsf{Next}, \mathcal{O}.\mathsf{Upd}, \mathcal{O}.\mathsf{Corr}\}$. $\bowtie$ indicates that Δ_0 and $\Delta_{\mathsf{fwr}_i+1}$ are skipped in the computation.

queries in these epochs, $\mathcal{B}$ uses the value t (if $t = g \star s$ then the randomness will still be consistent) generated during the first encryption of the input ciphertext as ciphertext value in epoch fwl_i and updates t to the right epoch e using its simulated tokens.

During the challenge call, the adversary will provide a ciphertext $\bar{C}$ which was created during the c-th call to $\mathcal{O}.\mathsf{Enc}$. The adversary cannot ask for an update of the c-th encryption in an epoch $\mathsf{e} \geq \mathsf{fwl}_i$, as this would trigger the trivial win condition $[\mathsf{fwl}_i, \mathsf{fwr}_i] \subseteq \mathcal{I}^* \cap \mathcal{C}^* \neq \emptyset$.

To simulate challenge-equal ciphertext in an epoch that is:

- To the left of the i-th insulated region: $\mathcal{B}$ simulates $\mathsf{GAINE.Upd}(\bar{C})$ using tokens that it created itself.
- Within the i-th insulated region: $\mathcal{B}$ simulates $\mathsf{GAINE.Upd}(\bar{C})$ if $b = 1$, and simulates $\mathsf{GAINE.Enc}(\bar{M})$ if $b = 0$. More precisely, if $\mathcal{O}.\mathsf{Sample}$ returns pairs of the form $(s_j, g \star s_j)$, $\mathcal{B}$ embeds g to k_{fwl_i} and $gk^{-1}_{\mathsf{fwl}_i - 1}$ to Δ_{fwl_i}. If $b = 0$, the reduction samples $(s, t) \leftarrow \mathcal{O}.\mathsf{Sample}()$, gives value s to $\pi(N \| \bar{M})$ and t to $\tilde{C}_{\mathsf{fwl}_i}$ (we want $k_{\mathsf{fwl}_i} = g$) since
$$\tilde{C}_{\mathsf{fwl}_i} = \mathsf{GAINE.Enc}(\bar{M}) = k_{\mathsf{fwl}_i} \star \pi(N \| \bar{M})$$
If $b = 1$, assume that $\bar{C}$ is an update of $\bar{C}_{\mathsf{e_c}}$, the output of the c-th $\mathcal{O}.\mathsf{Enc}$ query. $\mathcal{B}$ sampled $(s, t) \leftarrow \mathcal{O}.\mathsf{Sample}()$ and used s as randomness to create $\bar{C}_{\mathsf{e_c}}$ and to update it in epochs $\mathsf{e} < \mathsf{fwl}_i$. The reduction gives value t to $\tilde{C}_{\mathsf{fwl}_i}$ (we want $\Delta_{\mathsf{fwl}_i} = gk^{-1}_{\mathsf{fwl}_i - 1}$) since
$$\tilde{C}_{\mathsf{fwl}_i} = \mathsf{GAINE.Upd}(\bar{C}) = \Delta_{\mathsf{fwl}_i} \star (k_{\mathsf{fwl}_i - 1} \star s)$$
Furthermore, the reduction uses tokens $\Delta_{\mathsf{fwl}_i + 1}, \ldots, \Delta_{\mathsf{fwr}_i}$ to update $\tilde{C}_{\mathsf{fwl}_i}$ to simulate all challenge ciphertexts in epochs within the insulated region.
- To the right of the i-th insulated region: $\mathcal{B}$ simulates $\mathsf{GAINE.Enc}(\bar{M})$ using the keys that it created itself.

Eventually, $\mathcal{B}$ receives the output bit b' from $\mathcal{A}_i$. If $b' = b$, then $\mathcal{B}$ guesses that $\mathcal{O}.\mathsf{Sample}$ returned pairs of the form $(s_j, g \star s_j)$ (returns 0 to the $\mathsf{wk\text{-}PR}$ challenger), otherwise, $\mathcal{B}$ guesses that it has seen uniformly chosen pairs of set elements (returns 1). If $\mathcal{B}$ receives an oracle $\mathcal{O}.\mathsf{Sample}$ that samples pairs of the form $(s_j, g \star s_j)$, then $\mathcal{B}$ perfectly simulates the environment of $\mathcal{A}_i$ in $\mathcal{G}_i^b$. If $\mathcal{B}$ receives an oracle $\mathcal{O}.\mathsf{Sample}$ that samples pairs uniformly at random, then $\mathcal{B}$ wins with probability $1/2$. Thus,
$$\begin{aligned}
\mathbf{Adv}^{\mathsf{wk\text{-}PR}}_{\mathcal{G}\mathcal{A},\mathcal{B}}(\lambda) &= \left|1/2 - \Pr[\mathbf{Exp}^{\mathsf{wk\text{-}PR\text{-}0}}_{\mathcal{G}\mathcal{A},\mathcal{B}} = 1]\right| \\
&= \left|1/2 - (1/2 \Pr[\mathcal{G}_i^0 = 0] + 1/2 \Pr[\mathcal{G}_i^1 = 1])\right| \\
&= \left|1/2 - 1/2(1 - \Pr[\mathcal{G}_i^0 = 1]) - 1/2 \Pr[\mathcal{G}_i^1 = 1]\right| \\
&= 1/2 \left|\Pr[\mathcal{G}_i^0 = 1] - \Pr[\mathcal{G}_i^1 = 1]\right|
\end{aligned}$$

Finally, we get
$$\begin{aligned}
\frac{1}{(n+1)^2} \mathbf{Adv}^{\mathsf{detIND\text{-}UE\text{-}CPA}}_{\mathsf{GAINE},\mathcal{A}}(\lambda) &\leq \sum_{i=1}^{\ell} |\Pr[\mathcal{G}_i^1 = 1] - \Pr[\mathcal{G}_i^0 = 1]| \\
&= 2\ell \mathbf{Adv}^{\mathsf{wk\text{-}PR}}_{\mathcal{G}\mathcal{A},\mathcal{B}}(\lambda) \\
&\leq 2(n+1) \mathbf{Adv}^{\mathsf{wk\text{-}PR}}_{\mathcal{G}\mathcal{A},\mathcal{B}}(\lambda)
\end{aligned}$$
and thus $\mathbf{Adv}^{\mathsf{detIND\text{-}UE\text{-}CPA}}_{\mathsf{GAINE},\mathcal{A}}(\lambda) \leq 2(n+1)^3 \mathbf{Adv}^{\mathsf{wk\text{-}PR}}_{\mathcal{G}\mathcal{A},\mathcal{B}}(\lambda)$.

3.3 Post-quantum instantiations of GAINE

Good candidates for instantiating GAINE include the non-abelian group actions of the general linear group on the set of alternating trilinear forms introduced by Tang *et al.* [37] and on the set of k-tensors, for $k \geq 3$, by Ji *et al.* [20]. These two actions act on a finite vector space $V \simeq \mathbb{F}^n$ where $\mathbb{F}$ is a finite field and n an

integer. This implies that these two actions are mappable. Unfortunately, these two actions are also linear, meaning that the map $f_g : s \mapsto g \star s$ is a linear transformation of V. Given enough samples of the form $(s_i, g \star s_i)$ for a fixed g, one can compute $g \star s$ for any $s \in V$. This means that this action is not weak pseudorandom if the adversary gains access to too many samples of the weak pseudorandom experiment of Fig. 1. Since our security proof for GAINE uses one such sample per ciphertext, we can only use these two group actions to instantiate GAINE when there are few ciphertexts, which is unpractical. Moreover, Beullens [4] recently proposed new heuristic algorithms for solving the isomorphism problem of alternating trilinear forms more efficiently (this corresponds to the case when the adversary has access to only one sample in the weak pseudorandom experiment). Since the security of the group action of Tang *et al.* [37] is based on this problem, we cannot really hope to use this action to instantiate GAINE.

3.4 On the detIND-UE-CCA security of GAINE

In [9, sec. 5.1.1], a variant of SHINE with added ciphertext integrity, called SHINE0, is given by using $N\|M\|0^t$, for some t, as input of the permutation π during encryption and by checking that the 0 string is still present during decryption (if not $\perp$ is returned). This version of SHINE is shown to be detIND-UE-CCA secure under CDH [9, th. 5].

We define GAINE0 (see Fig. 9) similarly to SHINE0 and prove that it is detIND-UE-CCA secure if the group action is *weakly unpredictable*. Informally, recall that $(G, S, \star)$ is weakly unpredictable if, given polynomially many tuples of the form $(s_i, g \star s_i)$ where $g \xleftarrow{\$} G$ and each $s_i \xleftarrow{\$} S$, there is no PPT adversary that can compute $g \star s^*$ for a given challenge $s^* \xleftarrow{\$} S$. A full proof and precise definitions are given in the following Section 3.5 and Section 3.6.

The problematic ciphertext expansion of SHINE0 is reduced to almost nothing in the construction of OCBSHINE of [9, sec. 5.1.3] which is inspired by the authenticated encryption scheme OCB [33]. Once again, we can adapt this construction to GAINE.

That being said, finding a post-quantum instantiation for GAINE0 would make it the first post-quantum detIND-UE-CCA UE scheme. Indeed, the two LWE-based schemes of Jiang [21] and Nishimaki [29] are only randIND-UE-CPA secure. Finally, it is also worth noting that [9, Th 2.2, 2.4 & 2.6] showed that detIND-UE-CCA security implies the detIND-ENC-CCA and detIND-UPD-CCA security of [25], so GAINE0 is also CCA secure in the [25] sense.

3.5 GAINE with zeros: GAINE0

We add ciphertext integrity to GAINE using the technique of [9, sec. 5.1.1] for their SHINE0 scheme. Take message space $\mathcal{M} = \{0,1\}^m$ and nonce space $\mathcal{N} = \{0,1\}^v$. Let $(G, S, \star)$ be a MEGA with permutation $\pi : \{0,1\}^{m+v+t} \to S$. The encryption algorithm of GAINE0 feeds as input to π the concatenation of the message, the random nonce, and a zero string of length t. The decryption returns $\perp$ if the decrypted value does not end with 0^t. GAINE0 is defined in Fig. 9.

3.6 GAINE0 is INT-CTXT^s

We prove the following theorem.

Theorem 2 (GAINE0 is INT-CTXT^s). *Let* GAINE0 *be the UE scheme described in Fig. 9 for a MEGA family $\mathcal{GA}$. For any ideal cipher model adversary $\mathcal{A}$ that makes at most Q_E encryption queries before calling $\mathcal{O}$.*Try*, there exists a reduction $\mathcal{B}$ such that*

$$\mathbf{Adv}^{\mathsf{INT\text{-}CTXT^s}}_{\mathsf{GAINE0},\mathcal{A}}(\lambda) \leq \mathcal{O}(1)Q_E(n+1)^2\mathbf{Adv}^{\mathsf{wk\text{-}UP}}_{\mathcal{GA},\mathcal{B}}(\lambda) + \mathsf{negl}(\lambda)$$

where $\mathbf{Adv}^{\mathsf{wk\text{-}UP}}_{\mathcal{GA},\mathcal{B}}(\lambda)$ is defined in Definition 5

Remark 1. Combining the results of [9, Theorem 3], Theorem 1 and Theorem 2, we have that GAINE0 is detIND-UE-CCA.

We follow the proof technique of [9] and its presentation.

$\mathsf{Setup}(1^\lambda)$:

1. $(G, S, \star) \leftarrow \mathcal{GA}(\lambda)$
2. Choose π, m, v, t as above
3. $\mathsf{pp} \leftarrow (G, S, \star), \pi, m, v, t$
4. **return** pp

$\mathsf{KeyGen}(\mathsf{pp})$:

1. $k \xleftarrow{\$} G$.
2. **return** k.

$\mathsf{TokenGen}(k_\mathsf{e}, k_{\mathsf{e}+1})$:

1. $\Delta_{\mathsf{e}+1} \leftarrow k_{\mathsf{e}+1} k_\mathsf{e}^{-1}$
2. **return** $\Delta_{\mathsf{e}+1}$

$\mathsf{Upd}(\Delta_{\mathsf{e}+1}, C_\mathsf{e})$:

1. $C_{\mathsf{e}+1} \leftarrow \Delta_{\mathsf{e}+1} \star C_\mathsf{e}$
2. **return** $C_{\mathsf{e}+1}$

$\mathsf{Enc}(k_\mathsf{e}, M)$:

1. $N \xleftarrow{\$} \mathcal{N}$.
2. $C_\mathsf{e} \leftarrow k_\mathsf{e} \star \pi(N \| M \| 0^t)$
3. **return** C_e

$\mathsf{Dec}(k_\mathsf{e}, C_\mathsf{e})$:

1. $s \leftarrow \pi^{-1}(k_\mathsf{e}^{-1} \star C_\mathsf{e})$
2. Parse s as $N' \| M' \| Z$
3. **if** $Z = 0^t$
4. **return** M'
5. **else**
6. **return** $\perp$

Fig. 9. GAINE0: our generalization of the SHINE0 scheme using group actions.

Proof method. The challenger of the $\mathsf{INT\text{-}CTXT^s}$ game keeps a list of consistent values for ciphertexts, i.e.,, the underlying permutation output. Let $\tilde{C}$ be a forgery attempt sent to $\mathcal{O}.\mathsf{Try}$ in epoch $\tilde{\mathsf{e}}$ and let $\tilde{c} := k_{\tilde{\mathsf{e}}}^{-1} \star \tilde{C}$ be the underlying permutation output.

1. If $\tilde{c}$ is a new value, since π is a random permutation, then the $\mathsf{INT\text{-}CTXT^s}$ challenger simulates $\pi^{-1}(\tilde{c})$ to be a random string. The probability that this string ends by 0^t is negligible, and this carries over to the probability that the adversary wins the $\mathsf{INT\text{-}CTXT^s}$ game.
2. If $\tilde{c}$ already exists, suppose that this happens with probability p. We construct a reduction $\mathcal{B}$ playing the $\mathsf{wk\text{-}UP}$ game such that it wins with probability $p/(Q_E(n+1)^2)$. $\mathcal{B}$ guesses the location of the firewalls around the challenge epoch, embeds the $\mathsf{wk\text{-}UP}$ values and simulates the $\mathsf{INT\text{-}CTXT^s}$ game, using any successfully-forged ciphertext to compute the group action forgery for its $\mathsf{wk\text{-}UP}$ challenger.

Proof. The following proof is practically the same as in [9], we just replaced exponentiations by group actions and CDH by $\mathsf{wk\text{-}UP}$. We give the proof of [9] for completeness.

Note that the probability of a random string ends by 0^t is $1/2^t$. In the $\mathsf{INT\text{-}CTXT^s}$ game, the adversary ultimately sends a forgery C^* to the $\mathcal{O}.\mathsf{Try}$ oracle. If the trivial win condition does not trigger, then C^* is a new ciphertext to the challenger and there exists an insulated region around the challenge epoch. We split the proof into two parts on if $k_\mathsf{e}^{-1} \star C^*$ is a new value to the challenger:

1. If $k_\mathsf{e}^{-1} \star C^*$ is a new value, the random permutation π^{-1} will pick a random string a as the output of $\pi^{-1}(k_\mathsf{e}^{-1} \star C^*)$. The probability of a ending with 0^t is upper bounded by $1/2^t$.
2. If $k_\mathsf{e}^{-1} \star C^*$ is an existing value, denote this event as E_3, we claim that the probability of E_3 happens is very low. Which means it is hard to provide a valid forgery with a known permutation value. In other words, without the knowledge of the encryption key, it is difficult to provide a correct group action. Formally, we prove the following inequality in Lemma 1:

$$\Pr[E_3] := \Pr[k_\mathsf{e}^{-1} \star C^* \text{ exists}, C^* \text{ is new}] \leq Q_E(n+1)^2 \mathbf{Adv}^{\mathsf{wk\text{-}UP}}_{\mathcal{GA}}$$

In order to analyze the security, we define some events:

- $E_1 := \{C^* \text{ is new }\}$,

- $E_2 := \{k_{\mathsf{e}}^{-1} \star C^* \text{ is new}, C^* \text{ is new }\}$,
- Recall $E_3 := \{k_{\mathsf{e}}^{-1} \star C^* \text{ exists}, C^* \text{ is new }\}$.

Denote the experiment $\mathbf{Exp}^{\mathsf{INT\text{-}CTXT^s}}_{\mathsf{GAINE0},\mathcal{A}}$ to be $\mathbf{Exp}$. We have:

- $\Pr[\mathbf{Exp} = 1 \mid \neg E_1] = 0$.
- We proved $\Pr[\mathbf{Exp} = 1 \mid E_2] \leq 1/2^t$ in part 1.
- Events $\neg E_1, E_2, E_3$ are disjoint from each other, so $\Pr[\neg E_1] + \Pr[E_2] + \Pr[E_3] = 1$.
- We prove $\Pr[E_3] \leq Q_E(n+1)^2 \mathbf{Adv}^{\mathsf{wk\text{-}UP}}_{\mathcal{GA}}$ in Lemma 1.

Applying the above properties, we can compute the $\mathsf{INT\text{-}CTXT^s}$ advantage

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{INT\text{-}CTXT^s}}_{\mathsf{GAINE0},\mathcal{A}}(\lambda) &= \Pr[\mathbf{Exp} = 1] \\
&= \Pr[\mathbf{Exp} = 1 \mid \neg E_1] \cdot \Pr[\neg E_1] + \Pr[\mathbf{Exp} = 1 \mid E_2] \cdot \Pr[E_2] \\
&\quad + \Pr[\mathbf{Exp} = 1 \mid E_3] \cdot \Pr[E_3] \\
&= \Pr[\mathbf{Exp} = 1 \mid E_2] \cdot \Pr[E_2] + \Pr[\mathbf{Exp} = 1 \mid E_3] \cdot \Pr[E_3] \\
&\leq \Pr[\mathbf{Exp} = 1 \mid E_2] + \Pr[E_3] \\
&\leq 1/2^t + Q_E(n+1)^2 \mathbf{Adv}^{\mathsf{wk\text{-}UP}}_{\mathcal{GA}}
\end{aligned}$$

Lemma 1. *Let $\mathcal{GA}$ be the MEGA family used in* GAINE0. *Let $\mathcal{A}$ be an* $\mathsf{INT\text{-}CTXT^s}$ *adversary against* GAINE0 *that asks at most Q_E queries to $\mathcal{O}$.*Enc *before it sends its $\mathcal{O}$.*Try *query. Suppose that $\tilde{C}$ is a forgery attempt provided by $\mathcal{A}$ and that its corresponding permutation value is $\tilde{c}$. Define E to be the event that $\tilde{c}$ is an existed value but $\tilde{C}$ is a new one. Then, there exists a reduction $\mathcal{B}$ such that*

$$\Pr[E] \leq Q_E(n+1)^2 \mathbf{Adv}^{\mathsf{wk\text{-}UP}}_{\mathcal{GA},\mathcal{B}}$$

Proof. Suppose that $\mathcal{A}$ is an adversary against $\mathsf{INT\text{-}CTXT^s}$, and that it can provide a forgery such that $\tilde{C}$ is a new ciphertext but the underlying permutation value is an existing one with probability $\Pr[E]$. We give a reduction $\mathcal{B}$, in Fig. 10, that wins the wk-UP game with probability $\Pr[E]/(Q_E(n+1)^2)$.

$\mathcal{B}$ guesses the location of firewalls ($\hat{\mathsf{fwl}}$ and $\hat{\mathsf{fwr}}$) around the epoch when $\mathcal{O}$.Try is queried. Furthermore, it guesses which message (the h-th encryption) will be the underlying message of the forgery. Then, $\mathcal{B}$ receives the wk-UP values $(G, S, \star)$, $\mathcal{O}$.Sample and s^*, where $\mathcal{O}$.Sample returns tuples of the form $(s_i, g \star s_i)$ for a fixed $g \xleftarrow{\$} G$ and $s_i \xleftarrow{\$} S$. $\mathcal{B}$ embeds g (the group element used in $\mathcal{O}$.Sample) to $k_{\hat{\mathsf{fwl}}}$ by using the elements sampled by $\mathcal{O}$.Sample as ciphertexts in epoch $\hat{\mathsf{fwl}}$. On the h-th encryption, $\mathcal{B}$ embeds s^* to $\pi(N\|M\|0^t)$. When $\mathcal{B}$ receives a forgery $\tilde{C}$ for the h-th encryption in epoch $\tilde{\mathsf{e}} \in \{\hat{\mathsf{fwl}}, \ldots, \hat{\mathsf{fwr}}\}$, it can downgrade $\tilde{C}$ to epoch $\hat{\mathsf{fwl}}$ (where it embedded g to the epoch key). Then, $(\prod_{\mathsf{e}=\tilde{\mathsf{e}}}^{\hat{\mathsf{fwl}}+1} \Delta_{\mathsf{e}}^{-1}) \star \tilde{C} = g \star s^*$ with probability $\Pr[E]/(Q_E(n+1)^2)$, which is the advantage of winning the $\mathbf{Exp}^{\mathsf{wk\text{-}UP}}_{\mathcal{GA},\mathcal{B}}$ game.

3.7 Dealing with bad ciphertext expansion

The problematic ciphertext expansion of SHINE0 is reduced to almost nothing in the construction of OCBSHINE of [9, sec. 5.1.3] which is inspired by the authenticated encryption scheme OCB [33]. Once again, we can adapt this construction and proof to GAINE0 to get the OCBGAINE variant. See [9, Figure 27 & 28] for more details.

4 UE from Triple Orbital Group Action

Below, we present a new abstract algebraic structure that we call Triple Orbital Group Action (TOGA). The formulation of this framework is in fact motivated by the fact that we cannot instantiate GAINE with isogenies because it is notoriously hard [8] to hash into the set of supersingular elliptic curves. Indeed, GAINE

Reduction $\mathcal{B}$ playing $\mathbf{Exp}^{\mathsf{wk\text{-}UP}}_{\mathcal{GA},\mathcal{B}}(\lambda)$

$(G, S, \star)$, $\mathcal{O}.\mathsf{Sample}$ and s^*
do Setup(1^λ)
$\mathcal{A}^{\mathtt{ors}}(\lambda)$
if ABORT occurred **or** $\mathsf{twf} = 1$
  $\mathsf{win} \leftarrow 0$
else
  return win

Setup(1^λ)

$\mathsf{pp} \leftarrow \mathsf{GAINE0.Setup}(1^\lambda)$
$k_0 \leftarrow \mathsf{GAINE0.KeyGen}(\mathsf{pp})$
$\Delta_0 \leftarrow \perp$
$\mathsf{e}, \mathsf{c}, \mathsf{phase}, \mathsf{win}, \mathsf{twf} \leftarrow 0$
$\mathcal{L}^*, \mathcal{C}, \mathcal{K}, \mathcal{T} \leftarrow \emptyset$
$\hat{\mathsf{fwl}}, \hat{\mathsf{fwr}} \xleftarrow{\$} \{0, \ldots, n\}$
$\mathsf{h} \xleftarrow{\$} \{1, \ldots, Q_E\}$
for $j \in \{0, \ldots, \hat{\mathsf{fwl}} - 1\} \cup \{\hat{\mathsf{fwr}} + 1, \ldots, n\}$ **do**
  $k_j \xleftarrow{\$} G; \Delta_j \leftarrow k_j k_{j-1}^{-1}$ $\bowtie$
for $j \in \{\hat{\mathsf{fwl}} + 1, \ldots, \hat{\mathsf{fwr}}\}$ **do**
  $\Delta_j \xleftarrow{\$} G$

$\mathcal{O}.\mathsf{Enc}(M)$

$\mathsf{c} \leftarrow \mathsf{c} + 1$
if $\mathsf{c} = \mathsf{h}$
  if $\mathsf{e} < \hat{\mathsf{fwl}}$
    $\pi(N \| M \| 0^t) \leftarrow s^*$
    $C_{\mathsf{e}} \leftarrow k_{\mathsf{e}} \star s^*$
  else
    return ABORT
else
  $(\mathsf{inf}_1, \mathsf{inf}_2) \leftarrow \mathcal{O}.\mathsf{Sample}()$
  if $\mathsf{e} \in \{0, \hat{\mathsf{fwl}} - 1\} \cup \{\hat{\mathsf{fwr}} + 1, \ldots, n\}$
    $\pi(N \| M \| 0^t) \leftarrow \mathsf{inf}_1$
    $C_{\mathsf{e}} \leftarrow k_{\mathsf{e}} \star \mathsf{inf}_1$
  else
    $C_{\hat{\mathsf{fwl}}} \leftarrow \mathsf{inf}_2$
    for $j \in \{\hat{\mathsf{fwl}} + 1, \ldots, \mathsf{e}\}$ **do**
      $C_j \leftarrow \Delta_j \star C_{j-1}$
  $\mathsf{inf} \leftarrow (\mathsf{inf}_1, \mathsf{inf}_2)$
  $\mathcal{L}^* \leftarrow \mathcal{L}^* \cup \{(\mathsf{c}, C_{\mathsf{e}}, \mathsf{e}; \mathsf{inf})\}^{\succcurlyeq}$
return C_{e}

$\mathcal{O}.\mathsf{Next}$

$\mathsf{e} \leftarrow \mathsf{e} + 1$

$\mathcal{O}.\mathsf{Upd}(C_{\mathsf{e}-1})$

if $(\mathsf{c}, C_{\mathsf{e}-1}, \mathsf{e} - 1; \mathsf{inf}) \notin \mathcal{L}^*$
  return $\perp$
if $\mathsf{c} = \mathsf{h}$
  if $\mathsf{e} < \hat{\mathsf{fwl}}$
    $C_{\mathsf{e}} \leftarrow \Delta_{\mathsf{e}} \star C_{\mathsf{e}-1}$
  else
    return ABORT
else
  if $\mathsf{e} \in \{1, \ldots, \hat{\mathsf{fwl}} - 1\} \cup \{\hat{\mathsf{fwr}} + 1, \ldots, n\}$
    $(\mathsf{inf}_1, \mathsf{inf}_2) \leftarrow \mathsf{inf}$
    $C_{\mathsf{e}} \leftarrow k_{\mathsf{e}} \star \mathsf{inf}_1$
  else
    $(\mathsf{inf}_1, \mathsf{inf}_2) \leftarrow \mathsf{inf}$
    $C_{\hat{\mathsf{fwl}}} \leftarrow \mathsf{inf}_2$
    for $j \in \{\hat{\mathsf{fwl}} + 1, \ldots, \mathsf{e}\}$ **do**
      $C_j \leftarrow \Delta_j \star C_{j-1}$
$\mathcal{L}^* \leftarrow \mathcal{L}^* \cup \{(\mathsf{c}, C_{\mathsf{e}}, \mathsf{e}; \mathsf{inf})\}^{\succcurlyeq}$
return C_{e}

$\mathcal{O}.\mathsf{Corr}(\mathsf{inp}, \hat{\mathsf{e}})$

do $\mathsf{Check}(\mathsf{inp}, \hat{\mathsf{e}}; \mathsf{e}; \hat{\mathsf{fwl}}, \hat{\mathsf{fwr}})$
if $\mathsf{inp} = \mathsf{key}$
  $\mathcal{K} \leftarrow \mathcal{K} \cup \{\hat{\mathsf{e}}\}$
  return $k_{\hat{\mathsf{e}}}$
if $\mathsf{inp} = \mathsf{token}$
  $\mathcal{T} \leftarrow \mathcal{T} \cup \{\hat{\mathsf{e}}\}$
  for $i \in \mathcal{T}^*$ **do**
    for $(j, C_{i-1}, i - 1; \mathsf{inf}) \in \mathcal{L}^*$ **do**
      $C_i \leftarrow \mathcal{O}.\mathsf{Upd}(C_{i-1})$
      $\mathcal{L}^* \leftarrow \mathcal{L}^* \cup \{(j, C_i, i; \mathsf{inf})\}$
  return $\Delta_{\hat{\mathsf{e}}}$

$\mathcal{O}.\mathsf{Try}(\tilde{C})$

if $\mathsf{phase} = 1$
  return $\perp$
$\mathsf{phase} \leftarrow 1$
if $\tilde{\mathsf{e}} \in \mathcal{K}^*$ **or** $\tilde{C} \in \mathcal{L}^*$
  $\mathsf{twf} \leftarrow 1$
if $\tilde{\mathsf{e}} \notin \{\hat{\mathsf{fwl}}, \ldots, \hat{\mathsf{fwr}}\}$
  $\mathsf{twf} \leftarrow 1$
$y \leftarrow (\prod_{\mathsf{e}=\tilde{\mathsf{e}}}^{\hat{\mathsf{fwl}}+1} \Delta_{\mathsf{e}}^{-1}) \star \tilde{C}$ //$\tilde{\mathsf{e}} \geq \hat{\mathsf{fwl}}$
output y **to** $\mathbf{Exp}^{\mathsf{wk\text{-}UP}}_{\mathcal{GA},\mathcal{B}}$; **get** b
$\mathsf{win} \leftarrow b$

Fig. 10. Our reduction $\mathcal{B}$ for proof of Lemma 1. ors refers to the set $\{\mathcal{O}.\mathsf{Enc}, \mathcal{O}.\mathsf{Next}, \mathcal{O}.\mathsf{Upd}, \mathcal{O}.\mathsf{Corr}, \mathcal{O}.\mathsf{Try}\}$. $\bowtie$ indicates that Δ_0 and $\Delta_{\hat{\mathsf{fwr}}+1}$ are skipped in the computation. $\succcurlyeq$ indicates that inf is empty when $\mathsf{c} = \mathsf{h}$.

requires a MEGA and we prove, in Appendix B, that the existence of a one-way MEGA implies the existence of a hashable one-way EGA.

Let us start with a quick overview. A TOGA is made of three group actions, each with a distinct role. The main group action, that we write $(A, S, \star_A)$, is our starting point. The main ingredient to get a TOGA from the simple group action $\star_A$ is a congruence relation $\sim_A$. This relation allows us to derive a second group action $(A/\sim_A, S/\sim_S, \star_G)$, called the induced group action, of the quotient group on the quotient set (see Definition 11 for $\sim_S$). Of course, this induced group action is not mappable as we would not need a TOGA to build UE in this case. This time, we consider plaintexts as group elements of a third group action $(H, S, \star_H)$. For decryption to be possible, we assume that this action is efficiently invertible. We want $\star_H$ to commute with $\star_A$ but also that the orbits of $\star_H$ are exactly the classes of equivalences of $S/\sim_S$, which is what we call to be *orbital*. For a visualization of the interaction between the three group actions of a TOGA, see Fig. 11. The algebraic structure TOGA is explained in Section 4.1, while the computational model is given in Section 4.2. In Section 4.3, we show how to build UE from a TOGA.

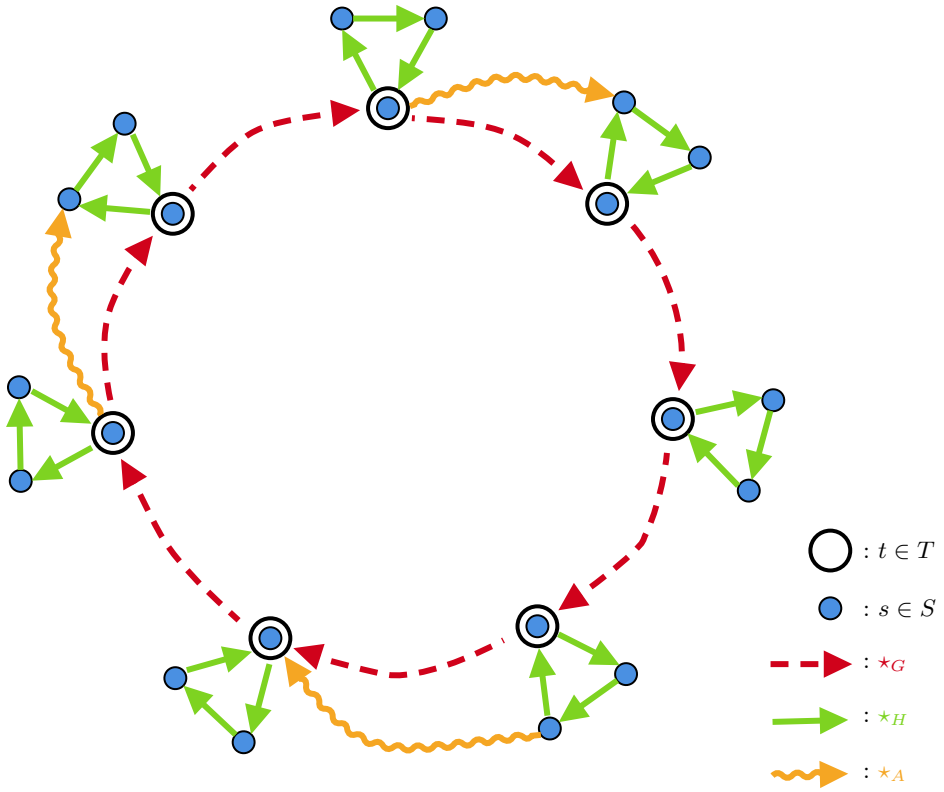

Fig. 11. Diagram for a TOGA $A, H, S, \star_A, \sim_A, \star_H$.

4.1 The algebraic structure

Let us assume that we have a group action $(A, S, \star_A)$ for an abelian multiplicative group A and a set S. We write 1_A for the neutral element of A. If there exists a congruence relation $\sim_A$ on A (we recall that a *congruence* on a set with an intern law is an equivalence relation compatible with the law, i.e., such that if $a_1 \sim_A a_2$ and $b_1 \sim_A b_2$ we have $a_1 b_1 \sim_A a_2 b_2$), then we get that $G = A/\sim_A$ is an abelian group for the law naturally derived from the multiplication in A.

Definition 11. *Let A be an abelian group and let $\sim_A$ be a congruence relation on A. Let S be a set and let $\star_A$ be a group action of A on S. The relation $\sim_S$ induced by $\sim_A$ and $\star_A$ is*

$$s_1 \sim_S s_2 \iff \exists a_1, a_2 \in A \text{ with } a_1 \sim_A a_2 \text{ such that } a_1 \star_A s_1 = a_2 \star_A s_2$$

Proposition 3. *Keeping the notations of Definition 11, we have that $\sim_S$ is an equivalence relation and $\star_A$ induces a group action $\star_G$ of $G := A/\sim_A$ on $T := S/\sim_S$.*

Proof. The relation $\sim_S$ is clearly reflexive and symmetric. For transitivity let us take $s_1, s_2, s_3 \in S$ with $s_1 \sim_S s_2$ and $s_2 \sim_S s_3$, we have $a_1 \star_A s_1 = a_2 \star_A s_2$ and $b_2 \star_A s_2 = b_3 \star_A s_3$, thus $a_1 b_2 \star_A s_1 = a_2 b_3 \star_A s_3$ and $a_1 b_2 \sim_A a_2 b_3$ since $\sim_A$ is a congruence. Let us write $G = A/\sim_A$ and $T = S/\sim_S$. First, we need to verify that the operation $\star_A$ is well-defined on the quotients. To see that, we need to verify that $a_1 \star_A s_1 \sim_S a_2 \star_A s_2$ when $a_1 \sim_A a_2$ and $s_1 \sim_S s_2$. This is true because we have $b_1 \sim_A b_2$ such that $b_1 \star_A s_1 = b_2 \star_A s_2$, and so $(a_2 b_1) \star_A (a_1 \star_A s_1) = (a_1 b_2) \star_A (a_2 \star_A s_2)$ with $a_2 b_1 \sim_A a_1 b_2$ because $\sim_A$ is a congruence. Then, we need to show that $\star_A : G \times T \to T$ verifies the usual group action properties from Definition 1. First, let us take $a \sim_A 1_A$. We must have $a \star_A s \sim_S s$ for any $s \in S$, which is clearly the case. Then, for any $a_1, a_2 \in A$, $s \in S$, we have the equality $(a_1 a_2) \star_A s = a_1 \star_A (a_2 \star_A s)$ and this equality remains true when considering the quotients G, T.

Definition 12. *Given $A, S, \star_A, \sim_A$ as in Proposition 3, the group action $\star_G$ of $A/\sim_A$ on $S/\sim_S$ is called the group action induced by $A, \star_A, \sim_A$ (or induced by A when it is clear from the context) and $(A, S, \star_A)$ is called the main group action.*

We obtain a third group action (hence the name of triple group action) by looking at the classes of equivalence of S. We want to consider these classes as the orbits of a third group action $\star_H : H \times S \to S$ for another abelian group H. By that we mean that, for any $s \in S$ and $h \in H$, we have $s \sim_S h \star_H s$ and that, for all $s' \sim_S s$, there exists $h \in H$ with $s' = h \star_H s$. Additionally, we need the group action $(H, S, \star_H)$ to be *free* because we will need to invert $\star_H$. When these constraints are respected we qualify the group action $(H, S, \star_H)$ to be *orbital*.

Finally, we want that $\star_A$ and $\star_H$ commute and that for any $a_1, a_2 \in A$ such that $a_1 a_2 \sim_A 1_A$, there exists a unique element $h(a_1, a_2) \in H$ such that $(a_1 a_2) \star_A s = h(a_1, a_2) \star_H s$ for any $s \in S$. With Proposition 4, we give a useful reformulation that will prove useful for the correctness of our scheme.

Proposition 4. *For any $a, b \in A$ with $a \sim_A b$, we have $a \star_A s = (h(a, c) h(b, c)^{-1}) \star_H (b \star_A s)$ for any $c \in A$ with $ac \sim_A 1_A$ and $s \in S$.*

Proof. We have $h(a, c) \star_H (b \star_A s) = (ac) \star_A (b \star_A s) = (abc) \star_A s = (bc) \star_A (a \star_A s) = h(b, c) \star_H (a \star_A s)$.

Definition 13 (TOGA). *When $A, H, S, \star_A, \sim_A, \star_H$ satisfy all the above properties we say that we have a Triple Orbital Group Action (TOGA).*

A visualization of a TOGA is given in Fig. 11. We give a simple example of a (pre-quantum) TOGA in Appendix C.

Remark 2. Note that A being a group is not really necessary for the UE scheme that we will introduce below. In fact, we only need that A is a monoid and that the quotient $A/\sim_A$ is a group. We only assumed that A is a group for simplicity.

4.2 Computational model

As for group actions, we define an ETOGA as an Effective TOGA:

Definition 14 (ETOGA). *A TOGA $A, H, S, \star_A, \sim_A, \star_H$ is effective if:*

1. *The group action $(H, S, \star_H)$ is an Effective and Easy Group Action (EEGA):*

(a) *The group action* $(H, S, \star_H)$ *is a free EGA.*
(b) *There is a PPT inversion algorithm* $\mathsf{Invert}_H : S^2 \to \{\bot\} \cup H$ *taking two elements* s_1, s_2 *and that outputs either* $\bot$ *when* $s_1 \not\sim_S s_2$ *or the element* $h \in H$ *such that* $s_1 = h \star_H s_2$.

2. *There exists a finite subset* $A' \subset A$ *such that:*
(a) *The class of equivalence of* A' *form a generating set of* G, *i.e.,* $G = A'/\sim_A$.
(b) *There is a PPT algorithm to compute* $a' \star_A s$ *for any* $s \in S$ *and* $a' \in A'$.
(c) *There exists a PPT algorithm* $\mathsf{Reduce}_A : A \to A'$ *that takes an element* $a \in A$ *and outputs* $a' \sim_A a$.
(d) *There exists a PPT algorithm to sample from* A' *in a distribution statistically close to the uniform distribution, we write* $a' \xleftarrow{\$} A'$ *for elements sampled in that manner.*
(e) *The distribution* $\mathcal{D}_G$ *that samples* $a' \xleftarrow{\$} A'$ *and returns the class of* $\mathsf{Reduce}_A(a')$ *in* G *is statistically close to the uniform distribution.*

3. *There exists a deterministic PPT algorithm* Reduce_S *to compute a canonical representative for equivalence classes in* $S/\sim_S$.

Remark 3. Note that the Reduce_A algorithm may or may not be deterministic. For efficiency, it is interesting to try to select the element a' in the class of a that minimizes the computation cost of $a' \star_A s$ for any $s \in S$.

Note that when $a_1 a_2 \sim 1_A$, we have $h(a_1, a_2) = \mathsf{Invert}_H((a_1 a_2) \star_A s, s)$ for any $s \in S$. Thus, we can define a PPT algorithm to compute $h(a_1, a_2)$ from Invert_H. We abuse notations and write h for this algorithm.

Since the function Reduce_S is deterministic, we can abuse notations and assimilate $T = S/\sim_S$ and $\mathsf{Reduce}_S(S)$ by identifying the elements of T to their canonical representative in S through Reduce_S. Using this, we sometimes apply the action $\star_A$ on the elements of T (it suffices to compose $\star_A$ with Reduce_S to obtain the canonical representative afterward).

4.3 The updatable encryption scheme

Let $\mathcal{TOGA}$ be an ETOGA family and let $(A, H, S, \star_A, \sim_A, \star_H)$ be $\mathcal{TOGA}(1^\lambda)$ for some λ. We fix a starting element $s_0 \in S$, and we also assume the existence of an invertible map $\psi : \mathcal{M} \to H$ where $\mathcal{M}$ is the space of the messages. We will use the function ψ to send the messages in the group H before encrypting them with $\star_H$. Then, decryption will rely on Invert_H. This operation is efficient by definition of an ETOGA. This principle basically solves the problem of needing our group action $\star_A$ to be mappable. The rest of our scheme follows the framework of GAINE with keys being elements of $A \times H$ and updates being obtained by applying $\star_A$ and $\star_H$. The security relies on the fact that the induced group action $(G, T, \star_G)$ is weakly pseudorandom. Our UE scheme TOGA-UE is given in Fig. 12.

Proposition 5 (Correctness of updates). *Let* $k_\mathsf{e}, k_{\mathsf{e}+1} = (a_\mathsf{e}, h_\mathsf{e}), (a_{\mathsf{e}+1}, h_{\mathsf{e}+1})$ *be two keys and* $C_\mathsf{e} = h_\mathsf{e} \star_H (a_\mathsf{e} \star_A s)$ *for some* $s \in S$. *If* $\Delta_{\mathsf{e}+1} = \mathsf{TokenGen}(k_\mathsf{e}, k_{\mathsf{e}+1})$, *then* $\mathsf{Upd}(\Delta_{\mathsf{e}+1}, C_\mathsf{e}) = h_{\mathsf{e}+1} \star_H (a_{\mathsf{e}+1} \star_A s)$.

Proof. We reuse the notation of TokenGen, we have for $c_\mathsf{e} = \mathsf{Reduce}_A(a_\mathsf{e}^{-1} a_{\mathsf{e}+1})$. Since $c_\mathsf{e} \sim_A a_{\mathsf{e}+1} a_\mathsf{e}^{-1}$, we have that $c_\mathsf{e} \star_A (a_\mathsf{e} \star_A s') = (h(c_\mathsf{e}, c_\mathsf{e}^{-1}) h(a_{\mathsf{e}+1} a_\mathsf{e}^{-1}, c_\mathsf{e}^{-1})^{-1}) \star_H ((a_{\mathsf{e}+1} a_\mathsf{e}^{-1} a_\mathsf{e}) \star_A s')$ by Proposition 4. The proof is completed by the fact that $(a_{\mathsf{e}+1} a_\mathsf{e}^{-1} a_\mathsf{e}) \star_A s' = a_{\mathsf{e}+1} \star_A s'$ and $h(c_\mathsf{e}, c_\mathsf{e}^{-1}) = 1_H$.

Proposition 6 (Correctness). *The* TOGA-UE *scheme is correct.*

Proof. Let $\mathsf{e}_1 \le \mathsf{e}_2 \le n+1$ be two epochs and let us consider a ciphertext c_{e_2} updated through the successive tokens Δ_{i+1} for $i \in [\mathsf{e}_1, \mathsf{e}_2 - 1]$ from an initial ciphertext c_{e_1} that is the encryption of a message m under the key $k_{\mathsf{e}-1}$ as in Definition 7. Each key k_i can be decomposed as $a_i, h_i \in A \times H$. By definition of $c_{\mathsf{e}_1} = \mathsf{Enc}(k_{\mathsf{e}_1}, m)$, we have $c_{\mathsf{e}_1} = (h_{\mathsf{e}_1} \psi(m)) \star_H a_{\mathsf{e}_1} \star_A \mathsf{Reduce}_S(s_1)$ for some $s_1 \in S$. By applying Proposition 5 on $s = \psi(m) \star_H \mathsf{Reduce}_S(s_1)$, we have that $c_{\mathsf{e}_2} = h_{\mathsf{e}_2} \psi(m) \star_A a_{\mathsf{e}_2} \star_A \mathsf{Reduce}_S(s_1)$ since $\star_A$ and $\star_H$ commute. Then, let us take any $b_{\mathsf{e}_2} \in A'$ such that $a_{\mathsf{e}_2} b_{\mathsf{e}_2} \sim_A 1_A$. By definition of h we know that $(b_{\mathsf{e}_2} a_{\mathsf{e}_2}) \star_A x = h(b_{\mathsf{e}_2}, a_{\mathsf{e}_2}) \star_H x$ for any $x \in S$. Thus, $s' = (h_{\mathsf{e}_2} h(b_{\mathsf{e}_2}, a_{\mathsf{e}_2}))^{-1} \star_H b_{\mathsf{e}_2} \star_A c_{\mathsf{e}_2} = \psi(M) \star_H \mathsf{Reduce}_S(s_1)$. Then, since the orbits of $\star_H$ are exactly the equivalence classes of S, we have $s' \sim_S s_1$ and so $\mathsf{Reduce}_S(s') = \mathsf{Reduce}_S(s_1)$. Thus, when we compute $\mathsf{Invert}_H(s', \mathsf{Reduce}_S(s'))$ we obtain $\psi(m)$ and the message is recovered by applying ψ^{-1}.

$\mathsf{Setup}(1^\lambda)$:

1. $(A, H, S, \star_A, \sim_A, \star_H) \leftarrow \mathcal{TOGA}(\lambda)$
2. Choose ψ, s_0 as above
3. $\mathsf{pp} \leftarrow (A, H, S, \star_A, \sim_A, \star_H, \psi, s_0)$
4. **return** pp

$\mathsf{KeyGen}(\mathsf{pp})$:

1. $a' \xleftarrow{\$} A'$
2. $h \xleftarrow{\$} H$
3. **return** $\mathsf{Reduce}_A(a'), h$

$\mathsf{TokenGen}(k_\mathsf{e}, k_{\mathsf{e}+1})$:

1. $(a_\mathsf{e}, h_\mathsf{e}) \leftarrow k_\mathsf{e}$
2. $(a_{\mathsf{e}+1}, h_{\mathsf{e}+1}) \leftarrow k_{\mathsf{e}+1}$
3. $c_\mathsf{e} \leftarrow \mathsf{Reduce}_A(a_\mathsf{e}^{-1} a_{\mathsf{e}+1})$
4. Compute $h = h(a_\mathsf{e}^{-1} a_{\mathsf{e}+1}, c_\mathsf{e}^{-1})$
5. **return** $c_\mathsf{e}, h h_{\mathsf{e}+1} h_\mathsf{e}^{-1}$

$\mathsf{Upd}(\Delta_{\mathsf{e}+1}, C_\mathsf{e})$:

1. $a, h \leftarrow \Delta_{\mathsf{e}+1}$
2. **return** $h \star_H (a \star_A C_\mathsf{e})$

$\mathsf{Enc}(k_\mathsf{e}, M)$:

1. $r' \xleftarrow{\$} A'$
2. $r \leftarrow \mathsf{Reduce}_A(r')$
3. $s = \mathsf{Reduce}_S(r \star_A s_0)$
4. $(a_\mathsf{e}, h_\mathsf{e}) \leftarrow k_\mathsf{e}$
5. **return** $(\psi(M) h_\mathsf{e}) \star_H (a_\mathsf{e} \star_A s)$

$\mathsf{Dec}(k_\mathsf{e}, C_\mathsf{e})$:

1. $(a_\mathsf{e}, h_\mathsf{e}) \leftarrow k_\mathsf{e}$
2. $b_\mathsf{e} \leftarrow \mathsf{Reduce}_A(a_\mathsf{e}^{-1})$
3. $h' \leftarrow h(a_\mathsf{e}, b_\mathsf{e})$
4. $s' \leftarrow (h_\mathsf{e} h')^{-1} \star_H (b_\mathsf{e} \star_A C_\mathsf{e})$
5. $s \leftarrow \mathsf{Reduce}_S(s')$
6. $M' \leftarrow \psi^{-1}(\mathsf{Invert}_H(s', s))$
7. **return** M'

Fig. 12. TOGA-UE: UE from ETOGA.

4.4 Security - TOGA-UE is detIND-UE-CPA secure

Let $A, H, S, \star_A, \sim_A, \star_H$ be an ETOGA, fix $s_0 \in S$ and let $(G, T, \star_G)$ be the group action induced by A (as in Definition 12) where $G := A/\sim_A$ and $T := S/\sim_S$. In Theorem 3, we show that our UE scheme TOGA-UE (described in Fig. 12) is detIND-UE-CPA secure if the group action $(G, T, \star_G)$ is weakly pseudorandom. We sample uniformly in T by sampling $g \xleftarrow{\$} G$ and returning the equivalence class of $g \star s_0$ in T.

Theorem 3 (TOGA-UE is detIND-UE-CPA). *Let* TOGA-UE *be the UE scheme described in figure 12 for an ETOGA family $\mathcal{TOGA}$. We define a group action family $\mathcal{GA}$, where $\mathcal{GA}(1^\lambda)$ is $(G, T, \star_G)$, the group action induced by $A \in \mathcal{TOGA}(1^\lambda)$ (as in Definition 12). For any adversary $\mathcal{A}$, there exists a reduction $\mathcal{B}$ such that*

$$\mathbf{Adv}^{\mathsf{detIND\text{-}UE\text{-}CPA}}_{\mathsf{TOGA\text{-}UE},\mathcal{A}}(\lambda) \leq \mathcal{O}(1)(n+1)^3 \cdot \mathbf{Adv}^{\mathsf{wk\text{-}PR}}_{\mathcal{GA},\mathcal{B}}(\lambda)$$

Proof. The proof uses the same hybrid argument as the one of Theorem 1, thus we only point out the differences between both proofs. Contrary to the proof of Theorem 1, we do not need to use the ideal cipher model. Indeed, in TOGA-UE, randomization of ciphertexts is not done through the permutation ψ. Thus, we do not need to "program" ψ to get consistent randomness throughout our reduction.

Our reduction $\mathcal{B}$, given in Fig. 13, starts by receiving a group action $(G, T, \star_G)$ and an oracle $\mathcal{O}.\mathsf{Sample}$ that returns either tuples of the form $(t_i, g \star_G t_i)$ or (t_i, u_i) where $g \xleftarrow{\$} G$ and $t_i, u_i \xleftarrow{\$} T$. We use the same hybrid argument over insulated regions as in Theorem 1. $\mathcal{B}$ will use the tuples of $\mathcal{O}.\mathsf{Sample}$ to perfectly simulate the detIND-UE-CPA experiment for TOGA-UE when those tuples are of the form $(t_i, g \star_G t_i)$. Thus, if we know an efficient adversary $\mathcal{A}$ against the detIND-UE-CPA security of TOGA-UE, using the hybrid argument of Theorem 1, $\mathcal{B}$ can use $\mathcal{A}$ to break the weak pseudorandomness of $(G, T, \star_G)$.

Our reduction $\mathcal{B}$ uses the following notations. Given a ciphertext C_e and a token Δ_e, we can downgrade C_e to epoch $\mathsf{e}-1$ like so:

1. $(c, h) \leftarrow \Delta_\mathsf{e}$
2. $b \leftarrow \mathsf{Reduce}_A(c^{-1})$

3. $h' \leftarrow h(b, c)$
4. $C_{\mathsf{e}-1} \leftarrow (hh')^{-1} \star_H (b \star_A C_{\mathsf{e}})$
5. **return** $C_{\mathsf{e}-1}$

For readability, we will use the (abuse of) notation $\Delta_{\mathsf{e}}^{-1} \star C_{\mathsf{e}}$ to denote this downgrade. Similarly, if $\Delta_{\mathsf{e}} = (c, h)$, we will use the notation $\Delta_{\mathsf{e}} \star C_{\mathsf{e}-1}$ to denote the update $h \star_H (c \star_A C_{\mathsf{e}-1})$.

In TOGA-UE, a ciphertext is of the form $C_{\mathsf{e}} := h_{\mathsf{e}} \star_H (a_{\mathsf{e}} \star_G r)$ with $k_{\mathsf{e}} := (a_{\mathsf{e}}, h_{\mathsf{e}})$, where $a_{\mathsf{e}} \xleftarrow{\$} G$, $h_{\mathsf{e}} \xleftarrow{\$} H$ and $r \xleftarrow{\$} T$ is the randomness used during the first encryption. Reduction $\mathcal{B}$ will try to embed the $\mathcal{O}.\mathsf{Sample}$ tuples in the i-th insulated region $[\mathsf{fwl}_i, \mathsf{fwr}_i]$. If $(r, s) \leftarrow \mathcal{O}.\mathsf{Sample}()$, $\mathcal{B}$ uses r as randomness for new ciphertexts. When updating ciphertext $C_{\mathsf{fwl}_i - 1} := h_{\mathsf{fwl}_i - 1} \star_H (a_{\mathsf{fwl}_i - 1} \star_G r)$ to epoch fwl_i, $\mathcal{B}$ sets $C_{\mathsf{fwl}_i} := h_{\mathsf{fwl}_i} \star_H s$ where h_{fwl_i} is simulated by $\mathcal{B}$. If (r, s) is of the form $(r, g \star_G r)$, $\mathcal{B}$ has embedded g into $k_{\mathsf{fwl}_i} := (g, h_{\mathsf{fwl}_i})$ and the randomness of the ciphertext stays consistent because of Proposition 5. Else, if (r, s) is a tuple of random elements of T, the ciphertexts inside the i-th insulated region are all random (there is no consistent key or randomness linking them).

Recall that a token $\Delta_{\mathsf{e}+1} := (c_{\mathsf{e}}, h h_{\mathsf{e}+1} h_{\mathsf{e}}^{-1})$ where $h_{\mathsf{e}}, h_{\mathsf{e}+1}$ are part of the epoch keys k_{e} and $k_{\mathsf{e}+1}$ and c_{e}, h are computed by Upd using those keys. When both keys are unknown (like in the i-th insulated region), c_{e} is uniformly distributed in G by Definition 14 item 2e. Recall that $h \in H$ is useful for the correction of updates (see Proposition 5) and that it is not independent of c_{e}. However, h_{e} and $h_{\mathsf{e}+1}$ are sampled uniformly in H and are not used in the computations of c_{e} and h. Since h_{e} and $h_{\mathsf{e}+1}$ are unknown to the adversary in the i-th insulated region, $h h_{\mathsf{e}+1} h_{\mathsf{e}}^{-1}$ is uniformly distributed in H and reduction $\mathcal{B}$ can perfectly simulate tokens inside the i-th insulated region.

Because of the correctness of updates in TOGA-UE (see Proposition 5) and of the observations above, when $\mathcal{O}.\mathsf{Sample}$ returns tuples of the form $(t_i, g \star_G t_i)$, the reduction $\mathcal{B}$ perfectly simulates the environment of the adversary $\mathcal{A}$ and we get a similar result as the one of Theorem 1.

On the CCA security of TOGA-UE. Unlike GAINE, making TOGA-UE CCA secure appears to be hard. Indeed, our construction has a pretty clear malleability property: let M, M' be two distinct messages, under the ψ map we get two elements $h := \psi(M), h' := \psi(M')$. Then, for any encryption c of the message M, we compute $h' h^{-1} \star_H c$ to obtain a valid encryption of M'. We leave the problem of making TOGA-UE CCA secure open for future work.

5 Instantiation from isogenies

The GAINE construction introduced in Section 3 requires a MEGA, i.e., that the underlying group action is mappable (which we showed was implying the group action to be hashable). Isogeny-based cryptography is one of the main provider of cryptographic group action so it is natural to ask if we can instantiate GAINE from them. Unfortunately, it is notoriously hard to hash into the set of supersingular curves [8] and this implies that building a MEGA from isogenies is probably very hard. This fact is what motivated the introduction of our new TOGA framework to build UE. In fact, our UE scheme is inspired by the SIGAMAL encryption scheme from Moriya, Onuki and Takagi [27]. Instead of encrypting messages as curves (wich would require to hash into the set of supersingular curves), we propose to encrypt messages as scalars similarly to what is done in SIGAMAL. In that sense, the instantiation of our UE scheme with isogenies can be considered as an updatable version of SIGAMAL. More precisely, our main group action will be the one of fractional ideals of quadratic orders on orientations of supersingular curves. Under the usual equivalence relation on ideals, the group action induced by this group action is the standard group action of the class group on oriented supersingular curves used in isogeny-based cryptography [11,13,17]. To obtain the richer structure of TOGA as in Definition 13, we consider orientations on curves with a level N-structure, i.e., curves that are enriched with a point of order N. In that setting, the group H is simply $\mathbb{Z}/N\mathbb{Z}^*$ and the action $\star_H$ is the scalar multiplication on the points of order N. For everything to be well-defined and behave as expected by our computational model, we must take N as a smooth number split in the quadratic order.

Reduction $\mathcal{B}$ playing $\mathbf{Exp}^{\mathsf{wk\text{-}PR}\text{-}b^*}_{\mathcal{GA},\mathcal{B}}(\lambda)$

receive $(G, T, \star_G)$ and $\mathcal{O}.\mathsf{Sample}$
do Setup(1^λ)
$\bar{M}, \bar{C} \leftarrow \mathcal{A}^{\mathtt{ors}}(\lambda)$
$\mathsf{phase} \leftarrow 1$
$\tilde{C}_{\tilde{\mathsf{e}}} \leftarrow \mathcal{O}.\mathsf{Chall}(\bar{M}, \bar{C})$
$b' \leftarrow \mathcal{A}^{\mathtt{ors}, \mathcal{O}.\mathsf{Upd}\tilde{\mathsf{C}}}(\tilde{C}_{\tilde{\mathsf{e}}})$
$\mathsf{twf} \leftarrow 1$ **if**
  $\mathcal{C}^* \cap \mathcal{K}^* \neq \emptyset$ **or**
  $\mathcal{I}^* \cap \mathcal{C}^* \neq \emptyset$
if ABORT occurred **or** $\mathsf{twf} = 1$
  $b' \xleftarrow{\$} \{0,1\}$
  return b'
if $(i, \mathsf{fwl}_i, \mathsf{fwr}_i) \notin \mathcal{FW}$
  $b' \xleftarrow{\$} \{0,1\}$
  return b'
if $b' = b$
  return 0
else
  return 1

Setup(1^λ)

$b \xleftarrow{\$} \{0,1\}$
$\mathsf{pp} \leftarrow \mathsf{TOGA\text{-}UE.Setup}(1^\lambda)$
$k_0 \leftarrow \mathsf{TOGA\text{-}UE.KeyGen}(\mathsf{pp})$
$\Delta_0 \leftarrow \perp$
$\mathsf{e}, \mathsf{c}, \mathsf{phase}, \mathsf{twf} \leftarrow 0$
$\mathcal{L}, \tilde{\mathcal{L}}, \mathcal{C}, \mathcal{K}, \mathcal{T} \leftarrow \emptyset$
$\mathsf{fwl}_i, \mathsf{fwr}_i \xleftarrow{\$} \{0, \ldots, n\}$
for $j \in \{0, \ldots, \mathsf{fwl}_i - 1\} \cup \{\mathsf{fwr}_i + 1, \ldots, n\}$ **do**
  $a_j \xleftarrow{\$} G, h_j \xleftarrow{\$} H; k_j \leftarrow (a_j, h_j)$
  $\Delta_j \leftarrow \mathsf{TOGA\text{-}UE.TokenGen}(k_j, k_{j+1})^{\bowtie}$
for $j \in \{\mathsf{fwl}_i + 1, \ldots, \mathsf{fwr}_i\}$ **do**
  $c_j \xleftarrow{\$} G, h_j \xleftarrow{\$} H; \Delta_j \leftarrow (c_j, h_j)$
$h_{\mathsf{fwl}_i} \xleftarrow{\$} H$

$\mathcal{O}.\mathsf{Enc}(M)$

$\mathsf{c} \leftarrow \mathsf{c} + 1$
$(\mathsf{inf}_1, \mathsf{inf}_2) \leftarrow \mathcal{O}.\mathsf{Sample}()$
if $\mathsf{e} \in \{0, \mathsf{fwl}_i - 1\} \cup \{\mathsf{fwr}_i + 1, \ldots, n\}$
  $(a_\mathsf{e}, h_\mathsf{e}) \leftarrow k_\mathsf{e}$
  $C_\mathsf{e} \leftarrow (\psi(M) h_\mathsf{e}) \star_H (a_\mathsf{e} \star_G \mathsf{inf}_1)$
else
  $C_{\mathsf{fwl}_i} \leftarrow (\psi(M) h_{\mathsf{fwl}_i}) \star_H \mathsf{inf}_2$
  for $j \in \{\mathsf{fwl}_i + 1, \ldots, \mathsf{e}\}$ **do**
    $C_j \leftarrow \Delta_j \star C_{j-1}$
$\mathsf{inf} \leftarrow (\mathsf{inf}_1, \mathsf{inf}_2, M)$
$\mathcal{L} \leftarrow \mathcal{L} \cup \{(\mathsf{c}, C_\mathsf{e}, \mathsf{e}; \mathsf{inf})\}$
return C_e

$\mathcal{O}.\mathsf{Next}$

$\mathsf{e} \leftarrow \mathsf{e} + 1$

$\mathcal{O}.\mathsf{Upd}(C_{\mathsf{e}-1})$

if $(\mathsf{c}, C_{\mathsf{e}-1}, \mathsf{e} - 1; \mathsf{inf}) \notin \mathcal{L}$
  return $\perp$
if $\mathsf{e} \in \{1, \ldots, \mathsf{fwl}_i - 1\} \cup \{\mathsf{fwr}_i + 1, \ldots, n\}$
  $(\mathsf{inf}_1, \mathsf{inf}_2, M) \leftarrow \mathsf{inf}$
  $(a_\mathsf{e}, h_\mathsf{e}) \leftarrow k_\mathsf{e}$
  $C_\mathsf{e} \leftarrow (\psi(M) h_\mathsf{e}) \star_H (a_\mathsf{e} \star_G \mathsf{inf}_1)$
else
  $(\mathsf{inf}_1, \mathsf{inf}_2, M) \leftarrow \mathsf{inf}$
  $C_{\mathsf{fwl}_i} \leftarrow (\psi(M) h_{\mathsf{fwl}_i}) \star_H \mathsf{inf}_2$
  for $j \in \{\mathsf{fwl}_i + 1, \ldots, \mathsf{e}\}$ **do**
    $C_j \leftarrow \Delta_j \star C_{j-1}$
$\mathcal{L} \leftarrow \mathcal{L} \cup \{(\mathsf{c}, C_\mathsf{e}, \mathsf{e}; \mathsf{inf})\}$
return C_e

$\mathcal{O}.\mathsf{Corr}(\mathsf{inp}, \hat{\mathsf{e}})$

do $\mathsf{Check}(\mathsf{inp}, \hat{\mathsf{e}}; \mathsf{e}; \mathsf{fwl}_i, \mathsf{fwr}_i)$
if $\mathsf{inp} = \mathsf{key}$
  $\mathcal{K} \leftarrow \mathcal{K} \cup \{\hat{\mathsf{e}}\}$
  return $k_{\hat{\mathsf{e}}}$
if $\mathsf{inp} = \mathsf{token}$
  $\mathcal{T} \leftarrow \mathcal{T} \cup \{\hat{\mathsf{e}}\}$
  return $\Delta_{\hat{\mathsf{e}}}$

$\mathcal{O}.\mathsf{Chall}(\bar{M}, \bar{C})$

if $(\mathsf{c}, \bar{C}, \tilde{\mathsf{e}} - 1; \mathsf{inf}) \notin \mathcal{L}$
  return ABORT
if $b = 0$
  $(s, t) \leftarrow \mathcal{O}.\mathsf{Sample}()$
  $r \leftarrow s$
  $\tilde{C}_{\mathsf{fwl}_i} \leftarrow (\psi(\bar{M}) h_{\mathsf{fwl}_i}) \star_H t$
else
  $(\mathsf{inf}_1, \mathsf{inf}_2, M) \leftarrow \mathsf{inf}$
  $r \xleftarrow{\$} T$
  $\tilde{C}_{\mathsf{fwl}_i} \leftarrow (\psi(M) h_{\mathsf{fwl}_i}) \star_H \mathsf{inf}_2$
for $j \in \{0, \ldots, \mathsf{fwl}_i - 1\}$ **do**
  $\tilde{C}_j \leftarrow (\prod_{k=j}^{1} \Delta_k)(\prod_{k=1}^{\tilde{\mathsf{e}}-1} \Delta_k^{-1}) \star \bar{C}$ `//left`
for $j \in \{\mathsf{fwl}_i + 1, \ldots, \mathsf{fwr}_i\}$ **do**
  $\tilde{C}_j \leftarrow \Delta_j \star \tilde{C}_{j-1}$ `//embed`
for $j \in \{\mathsf{fwr}_i + 1, \ldots, n\}$ **do**
  $(a_j, h_j) \leftarrow k_j$
  $\tilde{C}_j \leftarrow (\psi(\bar{M}) h_j) \star_H (a_j \star_G r)$ `//right`
$\tilde{\mathcal{L}} \leftarrow \cup_{j=0}^{n} \{(\tilde{C}_j, j)$
return $\tilde{C}_\mathsf{e}$

$\mathcal{O}.\mathsf{Upd}\tilde{\mathsf{C}}$

$\mathcal{C} \leftarrow \mathcal{C} \cup \{\mathsf{e}\}$
find $(\tilde{C}_\mathsf{e}, \mathsf{e}) \in \tilde{\mathcal{L}}$
return $\tilde{C}_\mathsf{e}$

Fig. 13. Our reduction $\mathcal{B}$ for proof of th. 3 in hybrid i. inf encodes fixed programming information: it marks two set elements $(\mathsf{inf}_1, \mathsf{inf}_2)$ sampled with $\mathcal{O}.\mathsf{Sample}$ and a plaintext M. inf_1 is the randomness used during encryption, inf_2 is used to compute the ciphertext value in epoch fwl_i and M is the plaintext. ors refers to the set $\{\mathcal{O}.\mathsf{Enc}, \mathcal{O}.\mathsf{Next}, \mathcal{O}.\mathsf{Upd}, \mathcal{O}.\mathsf{Corr}\}$. $\bowtie$ indicates that Δ_0 and $\Delta_{\mathsf{fwr}_i+1}$ are skipped in the computation.

5.1 A TOGA from isogenies.

We fix a prime p and consider the supersingular elliptic curves in characteristic p. In practice, they can always be defined over $\mathbb{F}_{p^2}$. We take a quadratic order $\mathfrak{O}$ of discriminant Δ and write $\mathfrak{K}$ for the quadratic imaginary field $\mathbb{Q} \otimes \mathfrak{O}$.

Definition 15. *For any elliptic curve E, a $\mathfrak{K}$-orientation is a ring homomorphism $\iota : \mathfrak{K} \hookrightarrow \mathrm{End}(E) \otimes \mathbb{Q}$. A $\mathfrak{K}$-orientation induces an $\mathfrak{O}$-orientation if $\iota(\mathfrak{O}) = \mathrm{End}(E) \cap \iota(\mathfrak{K})$. In that case, the couple (E, ι) is called an $\mathfrak{O}$-oriented curve and E is an $\mathfrak{O}$-orientable curve.*

In what follows, we consider the elements of $\mathcal{S}(p)/\pi$ rather than $\mathcal{S}(p)$ because the Frobenius π creates two orientations (one in E and one in $E^{(p)}$) from each optimal embedding of $\mathfrak{O}$ in a quaternion maximal order of $B_{p,\infty}$. Note that this is not the convention taken in [30,39] where orientations are not considered up to Galois conjugacy.

Definition 16. *$\mathcal{S}_{\mathfrak{O}}(p)$ is the set of $\mathfrak{O}$-oriented curves E, ι up to isomorphisms and Galois conjugacy.*

More concretely, we obtain an orientation E, ι by an endomorphism $\theta \in \mathrm{End}(E)$ such that $\iota(\mathfrak{O}) = \mathbb{Z}[\theta]$. Note that $\mathcal{S}_{\mathfrak{O}}(p)$ may be empty for some quadratic orders $\mathfrak{O}$ but it is also non-empty for an infinite number of order $\mathfrak{O}$ given any prime p.

The group action of fractional ideals. When we consider fractional $\mathfrak{O}$-ideal, we get an abelian group (for the multiplication operation). This group acts on the elements of $\mathcal{S}_{\mathfrak{O}}(p)$ by an operation that we write $\star_A$. This action is computed concretely using isogenies. Given an integral ideal $\mathfrak{a}$ and $E, \iota \in \mathcal{S}_{\mathfrak{O}}(p)$, we define the kernel of $\mathfrak{a}$ as $E[\mathfrak{a}] := \{P \in E[n(\mathfrak{a})] \mid \iota(\alpha)(P) = 0, \ \ \forall \alpha \in \mathfrak{a}\}$. The isogeny $\varphi^E_{\mathfrak{a}} : E \to E_{\mathfrak{a}}$ is simply the isogeny of kernel $E[\mathfrak{a}]$. Thus, we have $E_{\mathfrak{a}} = E/E[\mathfrak{a}]$ and $\iota_{\mathfrak{a}}(x) = \frac{1}{n(\mathfrak{a})} \varphi^E_{\mathfrak{a}} \circ \iota(x) \circ \hat{\varphi}^E_{\mathfrak{a}}$.

Fractional ideals can always be expressed as the multiplication of an integral ideal and a scalar in $\mathbb{Q}$. Thus, we will decompose the action of fractional ideals by applying first the action of the integral part as defined above, then by doing a scalar multiplication. We will explain how we propose to do this exactly a bit later when we introduce the set S.

Since our goal, in the end, is to get an ETOGA as defined in Section 4.2, we define the concrete group A that we will use as a subgroup of the group of $\mathfrak{O}$-ideals. This restriction is motivated by efficiency of the computation of $\star_A$. Indeed, the cost of computing an isogeny of degree D is in $O(\sqrt{D'})$ where D' is the biggest factor of D (see [3]). Thus, it is pointless to consider ideals that are multiples of prime ideals of big prime norm. This is why we fix an effective factor base of the $\mathfrak{O}$-ideals as a collection of ideals $\mathfrak{l}_1, \ldots, \mathfrak{l}_n$ where each $\mathfrak{l}_i$ is an ideal of norm ℓ_i for a small prime ℓ_i that is split in $\mathfrak{O}$ (here, small is to be considered with respect to some complexity parameter λ). Note that any such ideal $\mathfrak{l}_i$ has a *dual* ideal, usually denoted by $\overline{\mathfrak{l}_i}$, such that $\mathfrak{O}\ell_i = \mathfrak{l}_i\overline{\mathfrak{l}_i}$. Then, we can define the inverse of $\mathfrak{l}_i$ as $\mathfrak{l}_i^{-1} := \overline{\mathfrak{l}_i}/\ell_i$. Thus, $\mathfrak{l}_i^e$ is defined for any $e \in \mathbb{Z}$. With these definitions, we set our group A as $\langle \mathfrak{l}_1, \ldots, \mathfrak{l}_n \rangle_{\mathbb{Z}} := \{\prod_{i=1}^n \mathfrak{l}_i^{e_i} \ell_i^{f_i} \mid (e_1, \ldots, e_n) \in \mathbb{Z}^n, (f_1 \ldots, f_n) \in \mathbb{Z}^n\}$, the subgroup generated by our collection of prime ideals and their inverse. The implicit bijection between A and $\mathbb{Z}^{2n}$ given in our definition provides a way of representing the elements of A.

The standard cryptographic group action used in isogeny-based cryptography is obtained from our main group action of ideals by considering the equivalence relation $\sim_A$ defined as $\mathfrak{a} \sim_A \mathfrak{b}$ iff there exist non zero-elements $a, b \in \mathfrak{O}$ with $(a)\mathfrak{a} = (b)\mathfrak{b}$. It can be verified that $\sim_A$ is a congruence relation and so $A/\sim_A$ is an abelian group. If n is big enough A contains all equivalence classes of $\mathfrak{O}$-ideals. In that case, the group $A/\sim_A$ is called the *class group* of $\mathfrak{O}$ and written $\mathrm{Cl}(\mathfrak{O})$. By definition of $\mathcal{S}_{\mathfrak{O}}(p)$ (up to isomorphisms and Galois conjugacy), equivalent ideals act identically on the elements of $\mathcal{S}_{\mathfrak{O}}(p)$. Thus, the group action induced by $\star_A$ on $\mathcal{S}_{\mathfrak{O}}(p)$ as in Definition 12 is the usual isogeny-based group action. Since we clearly have $\ell\mathfrak{a} \sim_A \mathfrak{a}$ for any $\ell \in \mathbb{Q}^*$ and $\mathfrak{a} \in A$, we see that any ideal class of $A/\sim_A$ admits a representative of the form $\prod_{i=1}^n \mathfrak{l}_i^{e_i}$ where $(e_1, \ldots, e_n) \in \mathbb{Z}^n$. Thus, we can use elements of $\mathbb{Z}^n$ to represent the elements of G. More concretely, $\Phi : (e_1, \ldots, e_n) \mapsto \prod_{i=1}^n \mathfrak{l}_i^{e_i}$ yields an isomorphism between $A/\sim_A$ and $\mathbb{Z}^n/\mathcal{L}$ where $\mathcal{L}$ is the lattice generated by the $\mathbf{e} \in \mathbb{Z}^n$ such that $\Phi(\mathbf{e}) \sim_A 1_A$. The lattice $\mathcal{L}$ is usually called the *lattice of relations* of $\mathrm{Cl}(\mathfrak{O})$ for the factor basis $\mathfrak{l}_1, \ldots, \mathfrak{l}_n$.

The orbital group action. We now introduce our third group action (written $\star_H$ in Definition 13). To get this orbital group action, our idea is to use the group of points of the orientable elliptic curves. Let us take N a split integer in $\mathfrak{O}$ coprime with all the ℓ_i for $1 \leq i \leq n$. We can consider elements of the form E, ι, P where $P \in E[N]$ has order N. The group of fractional ideals of norm coprime with N acts on this set in the following manner: let us consider the fractional ideal $(a/b)\mathfrak{a}$ where $\mathfrak{a}$ is an integral ideal, we have the action $(a/b)\mathfrak{a} \star (E, \iota, P) = E_{\mathfrak{a}}, \iota_{\mathfrak{a}}, [ac]\varphi^E_{\mathfrak{a}}(P)$ where $c = b^{-1} \mod N$. However, in this setting, it seems hard to get a group action $\star_H$ as we desire. Indeed, equivalent ideals will send E, ι, P on E_1, ι_1, P_1 and E_2, ι_2, P_2 where we have $E_1, \iota_1 = E_2, \iota_2$ but we cannot tell anything about the points P_1, P_2 apart from the fact that they are two points of order N. Fortunately, if we restrict the set of points of order N we consider, the situation becomes a lot simpler. For that, if we have $\mathfrak{O} = \mathbb{Z}[\theta]$, it suffices to consider one eigenvalue of θ, i.e., one value ν such that $\ker(\iota(\theta) - \nu) \cap E[N]$ is a cyclic subgroup of order N. Since N is split, we know there exists one such eigenvalue ν and if $P \in \ker(\iota(\theta) - \nu) \cap E[N]$, we get that $\varphi^E\mathfrak{a}(P) \in \ker(\iota_{\mathfrak{a}}(\theta) - \nu) \cap E_{\mathfrak{a}}[N]$. Let us take

$$S = \{(E, \iota, P) \mid (E, \iota) \in \mathcal{S}_{\mathfrak{O}}(p) \text{ and } \langle P \rangle = E[N] \cap \ker(\iota(\theta) - \lambda\}. \tag{1}$$

With that choice of S, the image under the group action of two equivalent fractional ideals $\mathfrak{a}_1, \mathfrak{a}_2$ on $E, \iota, P \in S$ will give E_1, ι_1, P_1 and E_2, ι_2, P_2 with P_1 and P_2 in the same subgroup, i.e., there exists $\mu \in H = \mathbb{Z}/N\mathbb{Z}^*$ such that $P_1 = \mu P_2$. Moreover, it is easily verified that the scalar μ depends only on two ideals $\mathfrak{a}_1, \mathfrak{a}_2$ (and not on E, ι and P). This scalar μ is what we call $h(\mathfrak{a}_1, \mathfrak{a}_2)$ in our definition of a TOGA.

In that setting, for any scalar $h \in H$, we define $h \star_H (E, \iota, P)$ as $(E, \iota, [h]P)$ and it can be verified that $(A, H, S, \star_A, \sim_A, \star_H)$ is a TOGA.

5.2 Making the isogeny TOGA effective

To instantiate our protocol, we not only need a TOGA but an ETOGA as in Section 4.2. Below, we address the efficiency requirements. We try to stay generic in our approach and a more detailed example can be found in Section 5.3.

In fact, we are going to see that our proposed solution from isogeny is not exactly an ETOGA, but we can assume every operations to be practical assuming some (possibly-heavy) precomputation. In what follows, we discuss these limitations to understand what are the main obstacles.

The elements of S and the canonical representation of equivalence classes. We need to verify that the set S satisfies several properties. First, it needs to be finite which is our case (the cardinal is equal to $\varphi(N)h(\mathfrak{O})$ where φ is Euler's totient function). Regarding the existence of an origin s_0, it suffices to fix an element $s_0 \in$ S as the origin. There may be different ways of finding one, depending on the concrete $\mathfrak{O}$ and p, but a generic algorithm to compute one was described in [17] (in the context of generating backdoor curves for the Séta encryption scheme). Regarding the unique representation of the elements of S, it suffices to use a canonical representation of a class of isomorphic curves (using the j-invariant for instance). Once the curve E is fixed, we can deterministically derive a representation of the orientation ι and of any point P. The naive way of representing the point P would be by giving its coordinates x, y but we propose to use another way that is more compact in most cases. The idea is that given a basis P_1, P_2 of $E[N]$, any points $P \in E[N]$ is equal to $[a]P_1 + [b]P_2$ for some $a, b \in \mathbb{Z}/N\mathbb{Z}$. If the basis P_1, P_2 can be computed deterministically from E, the coefficients a, b are enough to recover the point P from the knowledge of E. This representation has size $2\log(N)$ which is usually a lot better than what we can expect with the naive method (that can be in $O(N \log(p))$ in the worst case). A deterministic algorithm to compute a basis of $E[N]$ from N can be easily derived from a deterministic algorithm $\mathsf{Point}(E, N, i)$ that takes an elliptic curve E and two integers i, N and outputs a point $P(i)$ of order N in E (the role of i is simply to index the points). There are numerous examples of such algorithms in the literature (see [40,28,2,15,32] for instance) so we do not describe one in detail. This algorithm will be useful for Reduce_S as well.

Membership testing for a given element E, ι, P in S consists in: veriyfing that E is supersingular, that ι is a correct orientation and that P is in $\ker(\iota(\theta) - \nu)$ and has order N. The first check can always be performed in polynomial time if p is in $\mathsf{poly}(\lambda)$ by counting the point of the curve E which is a well-sudied

task that can be solved in polynomial-time with the SEA algorithm. The second check will really depend on the choice of $\mathfrak{O}$ so it cannot be described generically but the idea is that we need to be able to verify the norm and trace of $\iota(\theta)$. For the third check, we simply need to be able to perform efficient operations on the N-torsion, so we need $E[N]$ to be defined over a field extension of degree polynomial in p. For the example of CSIDH given in Section 5.3 all these operations are efficient.

The last ingredient we need for S is the Reduce_S algorithm. Its goal is to determine a unique representative of the classes of S for the equivalence relation $\sim_S$. We recall that these classes correspond to the orbits of the group action of H. Thus, what we need concretely is an efficient way to compute a canonical point P_E of order N in $E[N] \cap \ker(\iota(\theta) - \nu)$ from E, ι. We will use the Point algorithm for that. Let us write ν', the second eigenvalues of $\iota(\theta)$.
$\mathsf{Reduce}_S(E, \iota, \cdot)$:

1. Compute $U = E[N] \cap \ker(\iota(\theta) - \nu)$ and $V = E[N] \cap \ker(\iota(\theta) - \nu')$.
2. Let $i = 0$, Repeat the following:
   (a) $P = \mathsf{Point}(E, N, i)$.
   (b) Compute the unique decomposition $P = P_U + P_V$ where $P_U \in U$ and $P_V \in V$.
   (c) $i = i + 1$.
3. Until P_V has order N.
4. Return E, ι, P_V.

The unique decomposition $P = P_U + P_V$ can be computed efficiently because the discrete logarithm problem is easy in $E[N]$.

Making $(H, S, \star_H)$ into an EEGA. First, note that it is easily verified that the group $H = \mathbb{Z}/N\mathbb{Z}^*$ is meeting all the requirements in terms of membership and equality testing, sampling and operations. Finally, the group action $\star_H$ being the scalar multiplication, it is clear that it can be performed in polynomial time when all the other operations can be done in PPT. For $(H, S, \star_H)$ to be an EEGA, we need to be able to invert efficiently $\star_H$. It is easily seen that Invert_H is simply a DLP in a cyclic subgroup of $E[N]$. For this operation to be efficient with the Pohlig-Hellman algorithm, we need N to be smooth (with smoothness bound polynomial in λ). Fortunately, this condition is rather agreeable with the other constraint regarding the field of definition of $E[N]$. Indeed, for any prime p, we know that $E[N]$ has a field of definition whose degree is polynomial (in the powersmoothness bound of N) for any p. In practice, we can even choose the prime p to ensure that $E[N]$ is defined over $\mathbb{F}_p$ or $\mathbb{F}_{p^2}$. This allows us to consider smooth values of N as well (for instance a power of 2).

The subset A'. The main requirement for A' is efficiency of $\star_A$. The complexity of this computation mainly depends on $n(\mathfrak{a})$ (because $\star_A$ consists in the computation of an isogeny of degree $n(\mathfrak{a})$). More concretely, the computation of an isogeny of degree $\prod_{i=1}^n \ell_i^{e_i}$ is in $O(n \max_{1\leq i\leq n} e_i \max_{1\leq i\leq n} \sqrt{\ell_i})$. Thus, we need that $n, \max_{1\leq i\leq n} \ell_i$ and $\max_{1\leq i\leq n} e_i$ are all polynomial in λ. We have already assumed that it was the case for n and the ℓ_i, so we only need to put a bound m on the exponent. This motivates to set A' as $\{\prod_{i=1}^n \mathfrak{l}_i^{e_i} \mid \mathbf{e} = (e_1, \ldots, e_n) \in \mathbb{Z}^n \text{ and } \|\mathbf{e}\|_\infty \leq m\}$, where $m = \mathsf{poly}(\lambda)$. Under the map Φ that we introduced in Section 5.1, we can sample elements of A' as image under Φ of random vectors in the ball $\mathcal{B}_\infty(m) = \{\mathbf{e} \in \mathbb{Z}^n \mid \|\mathbf{e}\|_\infty \leq m\}$ inside $\mathbb{Z}^n$ and it is clear that we can sample uniform vectors in that set. In the definition of an ETOGA, we also need that A' covers all equivalence classes of $A/\sim_A$ and that sampling in A' gives classes that are statistically distributed uniformly in G. We see that the cardinal of A' is $(2m+1)^n$. Thus, even when $\#A/\sim_A$ is exponential in λ, we can take values of $m, n = \mathsf{poly}(\lambda)$ such that A' is big enough to hope that all equivalence classes are covered and that sampling in A' provide a good distribution in $\mathrm{Cl}(\mathfrak{O})$. In practice, this is what we observe for the example given in Section 5.3.

Regarding the Reduce_A function, we use Φ again. Let us take an element $\mathfrak{a} \in A$. We know that this corresponds to an element of $\mathbb{Z}^{2n}$. Once again, since scalars are in the class of 1_A, we can simply ignore them and consider the equivalent ideal of the form $\prod_{i=1}^n \mathfrak{l}_i^{e_i}$ that corresponds to the vector $(e_1, \ldots, e_n) \in \mathbb{Z}^n$ under the map Φ. The ideal that we look for is $\Phi(\mathbf{e}')$ where $\mathbf{e}' \in \mathcal{B}_\infty(m)$ and $\mathbf{e}' \in \mathcal{L}$. If m is big enough, a correct solution e' is given by solving a Closest Vector Problem (CVP). Hence, we define a parameter

$\gamma \geq 1$ and look for a solution of the γ-CVP. Of course, the parameter γ and m need to be compatible in the sense that we want the solutions of our γ-CVP to be contained in $\mathcal{B}_\infty(m)$. Hence the Reduce_A algorithms consists in the following steps given an ideal $\mathfrak{a}$ as input: divide $\mathfrak{a}$ by its scalar factor to get another ideal $\mathfrak{a}'$, compute $\mathbf{e} = \Phi^{-1}(\mathfrak{a}')$, compute $\mathbf{e}'$ a solution for the γ-CVP for $\mathbf{e}$ and $\mathcal{L}$, output $\Phi(\mathbf{e}')$. The complexity of Reduce_A mainly depends on the complexity of solving the γ-CVP problem. The best known algorithm are sub-exponential in the dimension of the lattice $\mathcal{L}$ (here it is n) but the cost can be greatly reduced if the lattice $\mathcal{L}$ comes with a small basis of $\mathcal{L}$ (see [19]). This shifts the computational cost to the precomputation of a nice basis of $\mathcal{L}$. In any case, the computation of $\mathcal{L}$ is sub-exponential in the worst case so we might as well assume that we also compute a good basis. The choice of γ might also offer interesting tradeoffs between the cost of $\star_A$ and the cost of Reduce_A.

5.3 Concrete instantiation for isogeny TOGA and TOGA-UE

In this section, we describe concretely how to obtain our ETOGA from isogenies. For that, the most important choice is the one of the quadratic order $\mathfrak{O}$. While we have described the set $\mathcal{S}_\mathfrak{O}(p)$ and the action of $\mathfrak{O}$-ideals in full generality, in practice, there are only a few examples for which we know how to efficiently compute and represent the embedding ι and use it to compute the action of some $\mathfrak{O}$-ideals. We propose to use the CSIDH group action from Castryck *et al.* [11].

In CSIDH, we have $\mathfrak{O} = \mathbb{Z}[\sqrt{-p}] = \mathbb{Z}[\theta]$. In that case, we can show that $\mathcal{S}_\mathfrak{O}(p)$ is not empty and that the embedding ι is obtained with $\iota(\theta)$ as the Frobenius morphism $\pi : (x, y) \mapsto (x^p, y^p)$. This morphism is a well-defined endomorphism of the supersingular curve E if and only if $j(E) \in \mathbb{F}_p$ so membership testing can be done very easily. For a choice of p of the form $p = c\prod_{i=1}^{n} \ell_i \pm 1$, we end up with all the primes ℓ_i being split in $\mathfrak{O}$ and with efficient ℓ_i-isogeny computations which is why we consider ideals of norm divisible by the ℓ_i for the generators of A. The smooth integer N (which must be coprime with all the ℓ_i) can be chosen as a divisor of f if we want our points of N-torsion to be defined over $\mathbb{F}_p$. If we allow for extensions of bigger degree, we can take N as a smooth divisor of $\#E(\mathbb{F}_{p^k})$ for a small value of k. From there, the only remaining obstacle to get an ETOGA is the computation of the lattice of relation $\mathcal{L}$. Indeed, the computation of the class group's structure has sub-exponential classical complexity. For the prime of CSIDH-512, this structure and the corresponding lattice of relations was computed (breaking the record for the biggest class group computation) for the CSI-FiSh protocol [5]. However, given the sub-exponential complexity of this precomputation, it appears unrealistic to hope doing the same thing for bigger values of p. This is problematic because the level of security reached by CSIDH-512 is still unclear and could be quite far from the initially claimed NIST level-1 [31,7]. Moreover, our requirement of having the N-torsion defined over a small extension imposes even more constraints on the choice of p, which could imply to increase the size of p for the same level of security, thus making the precomputation even harder. Below, we give more details on the instantiation of our UE sheme based on CSIDH-512. We want to stress that establishing the exact level of security is out of the scope of this paper so we do not claim any particular security for this instantiation.

The number of small primes used in CSIDH is $n = 74$. It is constituted of the 73 smallest primes completed by $\ell_{74} = 587$. The cofactor c to construct the prime p is taken as $c = 4$. In that setting where the cofactor is very small, we cannot take N as a divisor of c, and so we need to look at the torsion defined over field extensions of $\mathbb{F}_p$. For instance, looking at supersingular curves over $\mathbb{F}_p$ with points over $\mathbb{F}_{p^6}$ and considering the quadratic twists as well, we can get a value $N \approx 2^{100}$ with a smoothness bound equal to 2^{26}. More precisely we get $N = 2 \cdot 3 \cdot 81331 \cdot 316423 \cdot 903311 \cdot 148811 \cdot 34785769$. In that setting, we can represent elements of S with roughly $712 = 2\log(N) + \log(p)$ bits.

As we explained, the lattice of relation $\mathcal{L}$ of dimension 74 and volume $\mathrm{Cl}(\mathfrak{O}) \approx \sqrt{p}$ was computed by the authors of the CSI-FiSh construction [5]. After obtaining a first basis for this lattice, they tried several solutions such as BKZ, and HKZ [23,34] to precompute a reduced basis in order to solve the CVP more efficiently (we remind the reader that solving a CVP is the main step in the Reduce_A algorithm). In CSI-FiSh, they propose to reduce further the norm of the solutions by applying the DLW algorithm [24]. This algorithm uses a list of short vectors of the lattice and searches for a solution by moving the initial vector with these short vectors. In their experiment, the authors of CSI-FiSh witnessed that the best performances,

considering the time of Reduce_A and the cost of the action computation as their metric, were obtained with solutions found by using a HKZ basis, applying Babai's nearest plane method and applying three times the DKW algorithm with a list of size 10000. In that case, the average ℓ_1 -norm is 213.97 ± 10.92 and they reported a computation cost of 135.41 ± 8.82 millions of cycles (for the execution of Reduce_A and the group action computation).

To estimate the cost of the group action computation in our case, we need to look at the additional cost of evaluating the isogenies on the points of order N. The other operations (such as scalar multiplication on the point of order N) are completely negligible in comparison. To estimate the cost of the evaluation, we can see that we need to perform the same operations for the usual group action computation but in $\mathbb{F}_{p^6}$. Thus, we can roughly estimate that the group action cost in our case will be the one of CSI-FiSh group action multiplied by the overhead caused by the $\mathbb{F}_{p^6}$-arithmetic in comparison to the $\mathbb{F}_p$ arithmetic. Depending on the way the arithmetic over $\mathbb{F}_{p^6}$ is implemented this should vary between 6 and 36 (mainly due to the multiplications over $\mathbb{F}_{p^6}$). We leave a more detailed study of performances for future work.

In terms of security, the CSIDH group action is believed to be weak pseudorandom. Generically, the weak pseudorandom problem for our generic isogeny-based TOGA is related to the $\mathfrak{O}$-DDH assumption studied in [12,10].

In conclusion, due to the hardness of computing the lattice of relations, CSIDH maybe not be the perfect candidate to instantiate our new construction, even though we do not have any better solutions for now. For another instantiation, we can mention the OSIDH key exchange [13] based on the same group action but with a different quadratic order. However, [16] exhibited an attack on OSIDH that explicitly uses the lattice of relations needed for our protocol so OSIDH is probably not a good match for us. The link between quadratic orders and supersingular curves is only beginning to reveal its full potential as most of the work we mentioned are less than five years old. In the future, we can hope that new constructions will be introduced and be more friendly to our scheme. We can also mention the group action introduced in [26]. This group action is in essence different from the one based on quadratic orders that we presented, so it is not clear if we could derive a TOGA from it, but it is a matter that could be worth investigating.

References

1. Navid Alamati, Luca De Feo, Hart Montgomery, and Sikhar Patranabis, *Cryptographic group actions and applications*, Advances in Cryptology – ASIACRYPT 2020: 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7–11, 2020, Proceedings, Part II (Berlin, Heidelberg), Springer-Verlag, 2020, p. 411–439.
2. Reza Azarderakhsh, David Jao, Kassem Kalach, Brian Koziel, and Christopher Leonardi, *Key compression for isogeny-based cryptosystems*, Proceedings of the 3rd ACM International Workshop on ASIA Public-Key Cryptography, ACM, 2016.
3. Daniel J. Bernstein, Luca De Feo, Antonin Leroux, and Benjamin Smith, *Faster computation of isogenies of large prime degree*, ANTS (2020).
4. Ward Beullens, *Graph-theorethic algorithms for the alternating trilinear form equivalence problem*, Cryptology ePrint Archive, Paper 2022/1528, 2022, https://eprint.iacr.org/2022/1528.
5. Ward Beullens, Thorsten Kleinjung, and Frederik Vercauteren, *Csi-fish: Efficient isogeny based signatures through class group computations*, International Conference on the Theory and Application of Cryptology and Information Security, Springer, 2019, pp. 227–247.
6. Dan Boneh, Kevin Lewi, Hart Montgomery, and Ananth Raghunathan, *Key homomorphic prfs and their applications*, Advances in Cryptology – CRYPTO 2013 (Berlin, Heidelberg) (Ran Canetti and Juan A. Garay, eds.), Springer Berlin Heidelberg, 2013, pp. 410–428.
7. Xavier Bonnetain and André Schrottenloher, *Quantum security analysis of CSIDH*, Advances in Cryptology - EUROCRYPT 2020, 2020, pp. 493–522.
8. Jeremy Booher, Ross Bowden, Javad Doliskani, Tako Boris Fouotsa, Steven D Galbraith, Sabrina Kunzweiler, Simon-Philipp Merz, Christophe Petit, Benjamin Smith, Katherine E Stange, et al., *Failing to hash into supersingular isogeny graphs*, arXiv preprint arXiv:2205.00135 (2022).
9. Colin Boyd, Gareth T. Davies, Kristian Gjøsteen, and Yao Jiang, *Fast and secure updatable encryption*, Advances in Cryptology – CRYPTO 2020: 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17–21, 2020, Proceedings, Part I, Springer-Verlag, 2020, p. 464–493.

10. Wouter Castryck, Marc Houben, Frederik Vercauteren, and Benjamin Wesolowski, *On the decisional diffie-hellman problem for class group actions on oriented elliptic curves*, Cryptology ePrint Archive (2022).
11. Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes, *Csidh: an efficient post-quantum commutative group action*, International Conference on the Theory and Application of Cryptology and Information Security, Springer, 2018, pp. 395–427.
12. Wouter Castryck, Jana Sotáková, and Frederik Vercauteren, *Breaking the decisional diffie-hellman problem for class group actions using genus theory*, Annual International Cryptology Conference, Springer, 2020, pp. 92–120.
13. Leonardo Colò and David Kohel, *Orienting supersingular isogeny graphs*, Number-Theoretic Methods in Cryptology 2019 (2019).
14. Jean-Sébastien Coron, Jacques Patarin, and Yannick Seurin, *The random oracle model and the ideal cipher model are equivalent*, Advances in Cryptology – CRYPTO 2008 (Berlin, Heidelberg) (David Wagner, ed.), Springer Berlin Heidelberg, 2008, pp. 1–20.
15. C. Costello, D. Jao, P. Longa, M. Naehrig, J. Renes, and D. Urbanik, *Efficient compression of SIDH public keys*, pp. 679–706, Springer International Publishing, 2017.
16. Pierrick Dartois and Luca De Feo, *On the security of osidh*, IACR International Conference on Public-Key Cryptography, Springer, 2022, pp. 52–81.
17. Luca De Feo, Cyprien Delpech de Saint Guilhem, Tako Boris Fouotsa, Péter Kutas, Antonin Leroux, Christophe Petit, Javier Silva, and Benjamin Wesolowski, *Séta: Supersingular encryption from torsion attacks*, International Conference on the Theory and Application of Cryptology and Information Security, Springer, 2021, pp. 249–278.
18. Edward Eaton, David Jao, Chelsea Komlo, and Youcef Mokrani, *Towards post-quantum updatable public-key encryption via supersingular isogenies*, Cryptology ePrint Archive (2020).
19. Thomas Espitau and Paul Kirchner, *The nearest-colattice algorithm: Time-approximation tradeoff for approx-cvp*, Open Book Series **4** (2020), no. 1, 251–266.
20. Zhengfeng Ji, Youming Qiao, Fang Song, and Aaram Yun, *General linear group action on tensors: A candidate for post-quantum cryptography*, Theory of Cryptography (Cham) (Dennis Hofheinz and Alon Rosen, eds.), Springer International Publishing, 2019, pp. 251–281.
21. Yao Jiang, *The direction of updatable encryption does not matter much*, Advances in Cryptology – ASIACRYPT 2020 (Cham) (Shiho Moriai and Huaxiong Wang, eds.), Springer International Publishing, 2020, pp. 529–558.
22. Michael Klooß, Anja Lehmann, and Andy Rupp, *(r)cca secure updatable encryption with integrity protection*, Advances in Cryptology – EUROCRYPT 2019 (Cham) (Y. Ishai and V. Rijmen, eds.), Springer International Publishing, 2019, pp. 68–99.
23. Aleksandr Korkine and G Zolotareff, *Sur les formes quadratiques*, Mathematische Annalen **6** (1873), no. 3, 366–389.
24. Thijs Laarhoven, *Sieving for closest lattice vectors (with preprocessing)*, International Conference on Selected Areas in Cryptography, Springer, 2016, pp. 523–542.
25. Anja Lehmann and Björn Tackmann, *Updatable encryption with post-compromise security*, Advances in Cryptology – EUROCRYPT 2018 (Cham) (J. B. Nielsen and V. Rijmen, eds.), Springer International Publishing, 2018, pp. 685–716.
26. Antonin Leroux, *A new isogeny representation and applications to cryptography*, Cryptology ePrint Archive (2021).
27. T. Moriya, H. Onuki, and T. Takagi, *Sigamal: a supersingular isogeny-based pke and its application to a prf*, International Conference on the Theory and Application of Cryptology and Information Security, Springer, 2020, pp. 551–580.
28. Michael Naehrig and Joost Renes, *Dual isogenies and their application to public-key compression for isogeny-based cryptography*, Advances in Cryptology – ASIACRYPT 2019 (Cham) (Steven D. Galbraith and Shiho Moriai, eds.), Springer International Publishing, 2019, pp. 243–272.
29. Ryo Nishimaki, *The direction of updatable encryption does matter*, Public-Key Cryptography – PKC 2022: 25th IACR International Conference on Practice and Theory of Public-Key Cryptography, Virtual Event, March 8–11, 2022, Proceedings, Part II (Berlin, Heidelberg), Springer-Verlag, 2022, p. 194–224.
30. Hiroshi Onuki, *On oriented supersingular elliptic curves*, Finite Fields and Their Applications **69** (2021), 101777.
31. Chris Peikert, *He gives C-sieves on the CSIDH*, Advances in Cryptology - EUROCRYPT 2020, 2020, pp. 463–492.
32. Geovandro C. C. F. Pereira, Javad Doliskani, and David Jao, *x-only point addition formula and faster torsion basis generation in compressed sike*, Cryptology ePrint Archive, Report 2020/431, 2020.
33. Phillip Rogaway, Mihir Bellare, John Black, and Ted Krovetz, *Ocb: A block-cipher mode of operation for efficient authenticated encryption*, Proceedings of the 8th ACM Conference on Computer and Communications Security (New York, NY, USA), CCS '01, Association for Computing Machinery, 2001, p. 196–205.

34. Claus-Peter Schnorr, *A hierarchy of polynomial time lattice basis reduction algorithms*, Theoretical computer science **53** (1987), no. 2-3, 201–224.
35. C. E. Shannon, *Communication theory of secrecy systems*, The Bell System Technical Journal **28** (1949), no. 4, 656–715.
36. P. W. Shor, *Algorithms for quantum computation: discrete logarithms and factoring*, Proceedings 35th Annual Symposium on Foundations of Computer Science, Nov 1994, pp. 124–134.
37. Gang Tang, Dung Hoang Duong, Antoine Joux, Thomas Plantard, Youming Qiao, and Willy Susilo, *Practical post-quantum signature schemes from isomorphism problems of trilinear forms*, Cryptology ePrint Archive, Report 2022/267, 2022.
38. J. Vélu, *Isogénies entre courbes elliptiques*, Comptes-Rendus de l'Académie des Sciences, Série I **273** (1971), 238–241.
39. B. Wesolowski, *Orientations and the supersingular endomorphism ring problem*, Cryptology ePrint Archive, Report 2021/1583, 2021.
40. Gustavo H. M. Zanon, Marcos A. Simplicio, Geovandro C. C. F. Pereira, Javad Doliskani, and Paulo S. L. M. Barreto, *Faster isogeny-based compressed key agreement*, Post-Quantum Cryptography (Tanja Lange and Rainer Steinwandt, eds.), Springer International Publishing, 2018, pp. 248–268.

A The Check algorithm

In our proofs, reductions play hybrid games and guess the location of the i-th insulated region. If the adversary sends a corrupt query inside this insulated region, the guess is wrong and reductions have to abort. We use the algorithm Check of [9], described in Fig. 14, to check if this event happens.

Check(inp, ê; e; fwl, fwr)

1. **if** ê > e
2. **return** ⊥
3. **if** inp = key **and** ê ∈ {fwl, . . . , fwr}
4. **return** ABORT
5. **if** inp = token **and** ê ∈ {fwl, . . . , fwr + 1}
6. **return** ABORT

Fig. 14. Algorithm Check of [9] used in our proofs. ê is the epoch in the adversary's request and e is the current epoch.

B The link between MEGA and hashable EGA

In the following, let $\mathcal{GA}$ be a group action family. We define one-way group actions.

Definition 17 (One-Way Group Action [1]). *Let $(G, S, \star)$ be $\mathcal{GA}(\lambda)$ for some security parameter λ. Let $\mathcal{D}_G$ and $\mathcal{D}_S$ be distributions on G and S respectively. For $s \in S$, let $f_s : G \to S$ be the function defined by $f_s : g \mapsto g \star s$. We say that $(G, S, \star)$ is $(\mathcal{D}_G, \mathcal{D}_S)$-one-way if, for all PPT adversaries $\mathcal{A}$, we have :*

$$\Pr[f_s(\mathcal{A}(s, f_s(g))) = f_s(g)] \leq \mathsf{negl}(\lambda)$$

where $s \leftarrow \mathcal{D}_S$ and $g \leftarrow \mathcal{D}_G$.

Informally, a group action $(G, S, \star)$ is $(\mathcal{D}_G, \mathcal{D}_S)$-one-way if, given a pair of set elements $(s, g \star s)$ where $s \leftarrow \mathcal{D}_S$ and $g \leftarrow \mathcal{D}_G$, there is no PPT adversary that can recover g. If $\mathcal{D}_S$ and $\mathcal{D}_G$ are uniform distributions, then we simply speak of an OW group action.

We can strengthen the Definition 2 of an EGA by replacing the existence of the origin s_0 by the following *Hashing to the set* axiom.

Definition 18 (Hashable EGA [1]). *Let $(G, S, \star)$ be an OW-EGA. We say that $(G, S, \star)$ is a hashable OW-EGA if there exists an efficient sampler $H : \{0,1\}^N \to S$ (where N depends on the security parameter), such that for all PPT adversaries $\mathcal{A}$, we have $\Pr[\mathcal{A}(i,j) \star H(i) = H(j)] \leq \mathsf{negl}(\lambda)$ for $i, j \xleftarrow{\$} \{0,1\}^N$.*

We show that our OW-MEGA is also a hashable OW-EGA.

Proposition 7. *Let $(G, S, \star)$ be an OW-MEGA with bijection π. Then, $(G, S, \star)$ is also a hashable OW-EGA with sampler $H := \pi$.*

Proof. Keeping the notations of the proposition, let $\mathcal{A}$ be an adversary that breaks the *Hashing to the set* axiom of the sampler π of $(G, S, \star)$ with probability ϵ. We build the following adversary $\mathcal{B}$ against the one-way property of $(G, S, \star)$.

1. $\mathcal{B}$ receives $(s, f_s(g)) = (s, g \star s)$ such that $s \xleftarrow{\$} S$ and $g \xleftarrow{\$} G$.
2. $\mathcal{B}$ calls $\mathcal{A}$ on input $(\pi^{-1}(s), \pi^{-1}(g \star s))$ and let $h \in G$ be the value returned by $\mathcal{A}$.
3. $\mathcal{B}$ outputs h.

By definition of $\mathcal{A}$, $\mathcal{A}$ returns h such that $h \star \pi(\pi^{-1}(s)) = \pi(\pi^{-1}(g \star s))$ with probability ϵ. This means that, with probability ϵ, $\mathcal{B}$ outputs h such that $h \star s = f_s(g)$ which is exactly breaking the one-way property of $(G, S, \star)$.

C A simple TOGA

In this section, we show how a simple TOGA (Definition 13) might look like. Let $S := U \times V$, where $U := \langle u \rangle$ is a cyclic (multiplicative) group of prime order q and $V := \langle v \rangle$ is a cyclic (multiplicative) group of order 2^n for some integer n.

Take $A := (\mathbb{Z}/q\mathbb{Z} \times \mathbb{Z}/2^n\mathbb{Z}, +)$. A acts on S through

$$\forall (a,b) \in A, \ \ \forall (x,y) \in S, \quad (a,b) \star_A (x,y) := (xu^a, yv^b)$$

one can easily verify that $(A, S, \star_A)$ is a group action. We define the following relation $\sim_A$ on A:

$$\forall (a_1,b_1), (a_2,b_2) \in A, \quad (a_1,b_1) \sim_A (a_2,b_2) \Leftrightarrow a_1 = a_2$$

one can easily verify that $\sim_A$ is a congruence relation on A. We have $G := A/\sim_A \simeq (\mathbb{Z}/q\mathbb{Z}, +)$. We recall the equivalence relation $\sim_S$ used in TOGA:

$$\forall s_1, s_2 \in S, \quad s_1 \sim_S s_2 \Leftrightarrow \exists c_1, c_2 \in A \text{ s.t. } c_1 \sim_A c_2 \text{ and } c_1 \star_A s_1 = c_2 \star_A s_2$$

Thus, for all $s_1, s_2 \in S$ such that $s_1 := (x_1, y_1)$ and $s_2 := (x_2, y_2)$, we have

$$\begin{aligned} s_1 \sim_S s_2 &\Leftrightarrow \exists a \in \mathbb{Z}/q\mathbb{Z}, b_1, b_2 \in \mathbb{Z}/2^n\mathbb{Z} \text{ s.t. } (a,b_1) \star_A s_1 = (a,b_2) \star_A s_2 \\ &\Rightarrow (x_1 u^a, y_1 v^{b_1}) = (x_2 u^a, y_2 v^{b_2}) \\ &\Rightarrow x_1 = x_2 \end{aligned}$$

Thus $T := S/\sim_S \simeq U$. Now take $H := (\mathbb{Z}/2^n\mathbb{Z}, +)$ and define $h \star_H (x,y) := (x, yv^h)$ for all $h \in H$ and $(x,y) \in S$. Clearly $\star_A$ and $\star_H$ commute, $\star_H$ is free and it is efficiently invertible using the Pohlig-Hellman algorithm for computing discrete logarithms. Moreover, it is also clear that each equivalence class of S is an orbit of $\star_H$. There remains one condition to check. Take $a_1, a_2 \in A$ such that $a_1 + a_2 \sim_A 1_A$, i.e., there exists $a \in \mathbb{Z}/q\mathbb{Z}$ and $b_1, b_2 \in \mathbb{Z}/2^n\mathbb{Z}$ such that $a_1 = (a, b_1)$, $a_2 = (-a, b_2)$ and $-a_2 = (a, -b_2)$. Then,

$$\forall (x,y) \in S, \quad (a_1 + a_2) \star_A (x,y) = (x, yv^{b_1+b_2}) = (b_1 + b_2) \star_H (x,y)$$

Moreover, $b_1 + b_2$ is the unique element of H satisfying the above equality. To conclude, we showed that $A, H, S, \star_A, \sim_A, \star_H$ satisfy Definition 13 and is thus a TOGA.