One-Round Authenticated Group Key Exchange from Isogenies

Atsushi FUJIOKA[1], Katsuyuki TAKASHIMA[2], and Kazuki YONEYAMA[3]

[1] Kanagawa University
[2] Mitsubishi Electric
[3] Ibaraki University

Abstract. This paper proposes two one-round authenticated group key exchange protocols from newly employed *cryptographic invariant maps* (CIMs): one is secure in the quantum random oracle model and the other resists against maximum exposure where a non-trivial combination of secret keys is revealed. The security of the former (resp. latter) is proved under the n-way decisional (resp. n-way gap) Diffie–Hellman assumption on the CIMs in the quantum random (resp. random) oracle model.
We instantiate the proposed protocols on the *hard homogeneous spaces* with limitation where the number of the user group is two. In particular, the protocols instantiated by using the *CSIDH, commutative supersingular isogeny Diffie–Hellman*, key exchange are currently more realistic than the general n-party CIM-based ones due to its realizability. Our two-party one-round protocols are secure against quantum adversaries.

Keywords: One-round authenticated group key exchange · Cryptographic invariant maps · Hard homogeneous spaces · Commutative supersingular isogeny Diffie–Hellman · G-CK model · G-CK$^+$ model · Quantum adversary.

1 Introduction

1.1 Background

Recently, National Institute of Standards and Technology (NIST) has initiated a process to standardize quantum-resistant public-key cryptographic algorithms [27], so, to study quantum-resistant cryptosystems is a hot research area. A wide range of quantum-resistant primitives (i.e., mathematical foundations) have been scrutinized by experts on cryptography and mathematics over the world. They include lattice-based, code-based, and multivariate cryptography. We treat with one (relatively) newly entered quantum-resistant primitive, which is called isogeny-based cryptography.

Key establishing over insecure channels is one of important cryptographic techniques. Recent researches on this have led to *authenticated key exchange* (AKE) and its multiparty extension, that is, *authenticated group key exchange* (AGKE). We then propose quantum-resistant AKE and AGKE schemes from isogenies on elliptic curves. In fact, we establish them on some abstract notions obtained from isogenies called *cryptographic invariant maps* (CIMs) and *hard homogeneous spaces* (HHSs).

HHS, CIM and CSIDH Key Exchange. In an unpublished but seminal paper [6], Couveignes initiated the research of isogeny-based cryptography where he formulated the basic notion of HHSs which is an abstract form of isogeny graphs and class groups of endomorphism rings of (ordinary) elliptic curves.

Independently, Rostovtsev and Stolbunov [29] proposed a Diffie–Hellman type key exchange from ordinary elliptic curve isogenies, which is now called RS key exchange and intensively studied very recently in [9]. While the RS key exchange uses ordinary curves, De Feo et al. employed supersingular isogenies for a practical key exchange protocol called supersingular isogeny Diffie–Hellman (SIDH) key exchange since ordinary isogeny problems suffer from subexponential quantum attacks. Jao et al. submitted an isogeny-based encryption scheme called SIKE (supersingular isogeny key encapsulation) to the NIST post-quantum cryptography competition, and the scheme is an enhanced form of the SIDH key exchange.

Castryck et al. [5] put forward a new HHS-based cryptographic construction called CSIDH (commutative SIDH) key exchange, which is constructed from a group action on the set of *supersingular elliptic curves defined over a prime field.* This ingenious key exchange opened a new research avenue in isogeny cryptography. As another new proposal, Boneh et al. [1] initiated to study a candidate multiparty non-interactive key exchange on CIMs, whose underlying structure is given by a HHS, (X, G), where X is a finite set and G is a finite abelian group, and the invariant map is defined on the n-th product X^n equipped with nice homomorphic (or equivariant) properties. As in the traditional Diffie–Hellman and pairing primitives, we can consider n-way computational, decisional, and gap Diffie–Hellman problems and assumptions on CIMs.

The notions of HHS and CIM give very concise conceptualizations of the above wonderful recent developments. We propose a generic conversion method from these key exchanges to authenticated ones. We next review the authenticated key exchanges and their importance.

Recently, Peikert quantified the attack complexity to CSIDH in a quantum setting, and concluded that CSIDH with a small parameter, e.g., CSIDH-512, might be breakable using the c-sieve [28]. Precisely, CSIDH-512 has the smaller security than the claimed 64-bit one. It means that we should use CSIDH with a bigger parameter, e.g., CSIDH-1024.

Authenticated Key Exchange (AKE). In an AKE protocol, two parties, called *initiator* and *responder*, have own static public keys, exchange ephemeral public keys, and compute a session key based on the public keys and the related secret keys. Roughly speaking, interactions between the initiator and the responder is called *round*, and the number of the interactions is called *round complexity.*

AKE protocols achieve that honest parties can establish a session key, and any malicious party cannot guess the session key. The latter condition is formulated in an indistinguishability game. Regarding to this security game, several models have been invented, and the Canetti–Krawczyk (CK) model was proposed to capture leakage of the session state [4]. After the proposal, several security requirements have been indicated such as *key compromise impersonation* (KCI), *weak perfect forward secrecy* (wPFS), and *maximal exposure attacks* (MEX) (refer to [19] for KCI, wPFS, and MEX). The CK model has been integrated with KCI, wPFS, and MEX to the CK$^+$ model [10].

Fujioka et al. [10] proposed a generic construction of CK$^+$-secure AKE from key encapsulation mechanism (KEM), and it means that we have a quantum-resistant AKE protocol when a quantum-resistant KEM exists. Longa gives an instantiation of their construction, and shows a two-round SIDH AKE protocol (AKE-SIDH-SIKE) which is CK$^+$-secure from a KEM scheme [24]. However, the round complexity of the (resultant) protocol is two as a responder has to compute a message depending on the incoming message from an initiator.

Galbraith proposed a one-round[4] SIDH-based AKE protocol (SIDH TS2) in [12] based on the Unified Model DH protocol by Jeong, Katz, and Lee [18]. The protocol is CK-secure under a decisional problem in classical random oracle model (ROM). Other one-round SIDH-based AKE protocols, SIDH UM and biclique SIDH, are proposed in [11]. The SIDH UM protocol is CK-secure in the quantum random oracle model (QROM) and the biclique SIDH protocol is CK$^+$-secure in the ROM.

Hövelmanns et al. [16] introduced a two-move (not one-round) generic AKE construction which is secure in (a variant of) CK model (called IND-StAA security) in the QROM. Since their generic construction is based on IND-CPA public key encryption (PKE), we can obtain isogeny-based AKE protocols from isogeny-based PKE such as SIKE [17].

To the best of our knowledge, we have neither CK-secure nor CK$^+$-secure one-round HHS-based AKE protocols.

Authenticated Group Key Exchange (AGKE). It is natural to extend two-party key exchange to n-party key exchange where $n > 2$, and actually, a group key exchange protocol can be constructed using a two-party key exchange protocol as a building block (e.g., [31]). However, this approach requires more round complexity than two, and this property holds for AGKE, also.

Several attempts have been done for one-round AGKE [2, 15, 26, 32, 23, 21]. Some do not satisfy important security properties, some have a limitation where the number of the group is three, and some are not quantum-resistant. Recently, it is shown that the CIM gives non-interactive key exchange for general n parties [1]. However, the protocol is not an AGKE one.

[4] Galbraith claims that the protocol is one-round however the description shows that it is two-round as the responder generates the response after receiving the first message [12].

Similar to those attempts, several security models for AGKE have been defined like ones for AKE (see a survey in [25]). Among them, we have the G-CK model (corresponding to the model with $(\mathsf{sfs}, \mathsf{scm})$-secrecy in [3]), the G-eCK model [26], and the G-CK$^+$ model [32]. The G-CK model is an AGKE variant of the CK model, and it captures leakage of the session state. The G-CK$^+$ model integrates the G-CK model with KCI, wPFS, and MEX. The G-eCK model is an AGKE variant of the eCK model [20], which also captures MEX. It is worth to note here that the G-CK$^+$ and G-eCK models are incomparable as the CK$^+$ and eCK models are so [8, 7].

One-round AGKE protocols secure in the G-CK or G-CK$^+$ model are given in [26, 32]. In those protocols, the number of the user group is limited to three, that is, they are tripartite key exchange.

On the other hand, Li and Yang [23] introduced one-round AGKE protocol from multilinear maps (MLMs), which is secure in the G-eCK model, and Lan et al. [21] introduced one-round AGKE protocol from indistinguishability obfuscation (iO), which is secure in a weak variant of the G-CK model. These protocols are not proved in the G-CK or G-CK$^+$ model, and quantum-resistance is not considered.

Thus, we do not have one-round AGKE protocols for general n-party ($n > 3$) secure in the G-CK or G-CK$^+$ model, additionally against quantum adversaries.

1.2 Our Contributions

One-Round AGKE from CIM. We propose two one-round AGKE protocols on the CIMs. One is called n-UM (n-Unified Model) which satisfies the G-CK security. The security of n-UM is proved under the n-way DDH assumption in the *quantum* random oracle model. The other is called BC n-DH (biclique n-Diffie–Hellman) which satisfies the G-CK$^+$ security. The security of BC n-DH is proved under the n-way GDH assumption in the random oracle model. The BC n-DH protocol requires that the number of the user group is bounded by logarithm of the security parameter. Comparison with existing one-round AGKE protocols is shown in Table 1.

Table 1. Comparison of one-round AGKE protocols.

	#parties	assumption	model	post-quantum?	proof
[15]	n	KEM, PRF	weak G-CK[5]	based on ingredients	StdM
[26]	3	gap-BDH	G-eCK	no	ROM
[32]	3	DBDH	G-CK$^+$	no	StdM
[23]	n	MLMs	G-eCK	no	StdM
[21]	n	iO	G-CK	no	StdM
n-UM	n	n-DDH	G-CK	yes	QROM
BC n-DH	n	n-GDH	G-CK$^+$	yes	ROM

[5] The model does not capture weak perfect forward secrecy (wPFS).

Note that it is not easy to prove the security of BC n-DH in the quantum random oracle model. In the reduction to the G-CK$^+$ security, the n-GDH solver needs to access the DDH oracle to maintain consistency among simulations of queries from the AGKE adversary. Hence, it is hard to prove the security of BC n-DH without the help of the DDH oracle. Also, the n-GDH solver wants to extract the answer of the n-GDH problem from a random oracle query by the AGKE adversary. However, the query may be a quantum state, and the solver cannot record a copy of the input due to the no-cloning theorem. Thus, such a proof strategy does not work. Recently, Zhandry [36] introduced a technique to record quantum queries. It is an open problem how to apply this technique to the proof.

On the other hand, in the reduction to the G-CK security in the security proof of n-UM, the adversary never expose ESKs. Thus, it is not necessary to access the DDH oracle to maintain consistency because the n-DDH solver knows all necessary secret keys to compute session keys except the test session. In the n-DDH assumption, the DH value (i.e., s_b in Definition 2.7) is given to the solver. It means that the n-DDH solver does not need to extract any information from random oracle queries by the AGKE adversary; and therefore, we can prove the G-CK security of n-UM in the quantum random oracle model.

Instantiating One-Round Two-Party AKE from HHS. We instantiate the proposed protocols on the HHS with limitation where the number of the user group is two. In particular, the CSIDH-based protocols are currently more realistic than the general n-party CIM-based ones due to its realizability. Our two-party one-round protocols are secure against quantum adversaries.

Compared to the previous SIDH-based one-round (two-party) AKE protocols [12, 11], the proposed protocols have several merits. While Galbraith et al. [13] proposed an active attack on the SIDH protocol by using the auxiliary points exchanged between users, the attack cannot be applied to our CSIDH-based ones since they include no auxiliary points. In [14], one attack scenario for the gap Diffie–Hellman (GDH) problem on the SIDH protocol is given since the degrees of isogenies used are fixed by public parameters as $\ell_i^{e_i}$ for small primes ℓ_i, e.g., $\ell_1 = 2, \ell_2 = 3$. As the CSIDH protocol uses random multiples consisting of several primes ℓ_i $(i = 1, \ldots, n)$ for the degrees and they are not fixed by public parameters, the attack cannot be applied to the CSIDH setting. Thus, the GDH assumption on CSIDH has no effective attacks at present, and we have a strong confidence on the security of our CSIDH-based BC protocol, which is reduced from the CSIDH GDH assumption. Comparison with existing isogeny-based AKE protocols is shown in Table 2.

2 Preliminaries

This work considers the PKI-based setting that each party locally keeps his own *static secret key* (SSK) and publishes a *static public key* (SPK) corresponding to the SSK. Validity of SPKs is guaranteed by a certificate authority. In a key

Table 2. Comparison of isogeny-based AKE protocols.

	assumption	model	#rounds	proof
SIDH TS2 [12]	SI-CDH	CK	1 [1]	ROM
AKE-SIDH-SIKE [24]	SI-DDH	CK$^+$	2	ROM
LJA [22]	SI-DDH	qCK	2	QROM
AKE$_{\text{SIDH-2}}$ [34]	SI-DDH	CK$^+$	2	ROM
SIDH UM [11]	SI-DDH	CK	1	QROM
biclique SIDH [11]	di-SI-GDH	CK$^+$	1	ROM
HKSU [16]	IND-CPA PKE	modified CK	2	QROM
HHS-UM	2-DDH	CK	1	QROM
HHS-BC	2-GDH	CK$^+$	1	ROM

exchange session, each party generates an *ephemeral secret key* (ESK) and sends an *ephemeral public key* (EPK) corresponding to the ESK. A session key is derived from these keys with a key derivation mechanism like a hash function modeled as the random oracle.

2.1 Post-Quantum G-CK and G-CK$^+$ Model

In this section, we revisit security models, the G-CK model [3][6] and the G-CK$^+$ model [32], for AGKE against quantum adversaries.

Note that we show a model specified to one-round protocols for simplicity. It can be trivially extended to any round protocol.

Protocol Participants and Initialization. Let $\mathcal{U} := (U_1, \ldots, U_{n_u})$ be a set of potential protocol participants, where the ID space is **IDS**. Each party U_i is modeled as a probabilistic polynomial-time (PPT) Turing machine w.r.t. security parameter κ while the adversary is modeled by a probabilistic polynomial time quantum Turing machine. For party U_i, we denote static secret (public) key by SSK_i (SPK_i) and ephemeral secret (public) key by ESK_i (EPK_i). Party U_i generates its own keys, SSK_i and SPK_i, and the static public key SPK_i is linked with U_i's identity in some systems like PKI.

Session. An invocation of a protocol is called a *session*. We suppose that a session contains n parties $(U_{j_1}, \ldots, U_{j_n})$, where $2 \leq n \leq n_u$. A session is managed by a tuple $(\Pi, \mathsf{role}_i, U_{j_\ell}, U_{j_1}, \ldots, U_{j_n})$, where $\Pi \in \mathbf{PRS}$ is a protocol identifier for the protocol ID space **PRS**, role_i is a role identifier, and U_{j_ℓ} is a party identifier. Role identifiers represents the order of party identities in protocols (i.e., in the two-party case, it corresponds to the initiator or the responder.). Hereafter, for simplicity, we can suppose that $U_{j_\ell} = U_\ell$ without loss of generality. If U_j is activated with $(\Pi, \mathsf{role}_i, U_j, U_1, \ldots, U_n, \mathit{Init})$, then U_j is called the *i-th player*. The role of a party in a session is decided by the lexicographic order of party

[6] In [3], several variants of security models are proposed. The G-CK model corresponds to the model with $(\mathsf{sfs}, \mathsf{scm})$-secrecy.

identities, and $\mathsf{role}_i \neq \mathsf{role}_{i'}$ for any i and i' $(i \neq i')$ in a session.[7] U_j outputs EPK_j, receives $EPK_{j'}$ from $U_{j'}$ for $j' = 1, \ldots, j-1, j+1, \ldots, n$, and computes the session key SK.

If U_j is the i-th player of a session, the session is identified by $\mathsf{sid} = (\Pi, \mathsf{role}_i, U_j, U_1, \ldots, U_n, EPK_j)$ or $\mathsf{sid} = (\Pi, \mathsf{role}_i, U_j, U_1, \ldots, U_n, EPK_1, \ldots, EPK_n)$. We say that U_j is the *owner* of session sid, if the third coordinate of sid is U_j. We say that U_j is a *peer* of session sid, if the third coordinate of sid is not U_j. We say that a session is *completed* if its owner computes the session key. We say $(\Pi, \mathsf{role}_{i'}, U_{j'}, U_1, \ldots, U_n, EPK_1, \ldots, EPK_n)$ is a *matching session* of $(\Pi, \mathsf{role}_i, U_j, U_1, \ldots, U_n, EPK_1, \ldots, EPK_n)$, where $i' \neq i$ and $j' \neq j$.

Adversary. The adversary $\mathcal{A}$, which is modeled as a PPT quantum Turing machine, controls all communications between parties including session activation and registrations of parties by performing the following adversary queries.

- Send(message): This query allows an adversary $\mathcal{A}$ to send the message to U_j instead of other parties. The message has the following form: $(\Pi, \mathsf{role}_i, U_j, U_1, \ldots, U_n, \mathit{Init})$ for session activation, or $(\Pi, \mathsf{role}_{i'}, U_{j'}, U_1, \ldots, U_n, EPK_1, \ldots, EPK_{j'}, \ldots, EPK_n)$. $U_{j'}$ runs on input the message according to the protocol, updates the internal state, and returns a response (if any). $\mathcal{A}$ learns the response from $U_{j'}$.
- $\mathsf{Establish}(U_j, SPK_j)$: This query allows $\mathcal{A}$ to introduce new parties. In response, if $U_j \notin \mathcal{U}$ (due to the uniqueness of identities) then U_j with the static public key SPK_j is added to $\mathcal{U}$. Note that $\mathcal{A}$ is not required to prove the possession of the corresponding secret key SSK_j. If a party is registered by a $\mathsf{Establish}$ query issued by $\mathcal{A}$, then we call the party *dishonest*. If not, we call the party *honest*.

To capture exposure of secret information, the adversary, $\mathcal{A}$, is allowed to issue the following queries.

- $\mathsf{SessionReveal}(\mathsf{sid})$: The adversary, $\mathcal{A}$, obtains the session key SK for the session sid if the session is completed.
- $\mathsf{StateReveal}(\mathsf{sid})$: The adversary, $\mathcal{A}$, obtains the session state of the owner of session sid if the session is not completed (the session key is not established yet). The session state includes all ephemeral secret keys and intermediate computation results except for immediately erased information but does not include the static secret key. Note that the protocol specifies what the session state contains.
- $\mathsf{StaticReveal}(U_j)$: This query allows $\mathcal{A}$ to obtain all static secret keys of the party U_j.

[7] This condition is necessary to keep correctness of the protocol. Some session key derivation process contain the evaluation of a function on inputting party identifiers according to roles of parties. If parties do not share their roles, the order of inputs cannot be consistent for parties.

- $\mathsf{EphemeralReveal}(\mathsf{sid})$: This query allows $\mathcal{A}$ to obtain all ephemeral secret keys of the owner of the session sid if the session is not completed (the session key is not established yet). It is necessary to represent a MEX situation that an adversary can reveal ESKs but is prevented to obtain other session state such that the adversary trivially wins.

Freshness. For the security definition, we need the notion of freshness. Freshness is the condition that an adversary cannot break the security in trivial ways, and necessary to make the experiment meaningful. For example, if the adversary can learn all secret information of a party in a session (i.e., the SSK and the ESK), it is not avoidable to reveal the session key of the session. To exclude such a trivial attack secret information learned by the adversary must be limited for the target session. Conversely, the definition of freshness must not limit any non-trivial attacks.

Let $\mathsf{sid}^* = (\Pi, \mathsf{role}_i, U_j, U_1, \ldots, U_n, EPK_1, \ldots, EPK_n)$ be a completed session between honest parties $(U_1, \ldots, U_n)$, which is owned by U_j. If a matching session exists, then let $\overline{\mathsf{sid}^*}_{j'}$ be a matching session of sid^* where the owner is $U_{j'}$.

Definition 2.1 (Freshness for G-CK Security). *We say session sid^* is* fresh *in the* G-CK *model if none of the following conditions hold:*

1. *The adversary, $\mathcal{A}$, poses* $\mathsf{SessionReveal}(\mathsf{sid}^*)$, *or* $\mathsf{SessionReveal}(\overline{\mathsf{sid}^*}_{j'})$ *if* $\overline{\mathsf{sid}^*}_{j'}$ *exists.*
2. *The adversary, $\mathcal{A}$, poses* $\mathsf{EphemeralReveal}()$ *for any session.*
3. *The adversary, $\mathcal{A}$, poses* $\mathsf{StateReveal}(\mathsf{sid}^*)$, *or* $\mathsf{StateReveal}(\overline{\mathsf{sid}^*}_{j'})$ *if* $\overline{\mathsf{sid}^*}_{j'}$ *exists.*
4. *The adversary, $\mathcal{A}$, poses* $\mathsf{StaticReveal}(U_{j'})$ *for any $U_{j'} \in (U_1, \ldots, U_n)$ if there exists a non-matching session of* sid^*.

Definition 2.2 (Freshness for G-CK$^+$ Security). *We say session sid^* is* fresh *in the* G-CK$^+$ *model if none of the following conditions hold:*

1. *The adversary, $\mathcal{A}$, poses* $\mathsf{SessionReveal}(\mathsf{sid}^*)$, *or* $\mathsf{SessionReveal}(\overline{\mathsf{sid}^*}_{j'})$ *if* $\overline{\mathsf{sid}^*}_{j'}$ *exists.*
2. *The adversary, $\mathcal{A}$, poses* $\mathsf{StateReveal}(\mathsf{sid}^*)$, *or* $\mathsf{StateReveal}(\overline{\mathsf{sid}^*}_{j'})$ *if* $\overline{\mathsf{sid}^*}_{j'}$ *exists.*
3. *The adversary, $\mathcal{A}$, poses both of* $\mathsf{StaticReveal}(U_j)$ *and* $\mathsf{EphemeralReveal}(\mathsf{sid}^*)$.
4. *The matching session,* $\overline{\mathsf{sid}^*}_{j'}$ *exists, and $\mathcal{A}$ poses both of* $\mathsf{StaticReveal}(U_{j'})$ *and* $\mathsf{EphemeralReveal}(\overline{\mathsf{sid}^*}_{j'})$.
5. *The matching session,* $\overline{\mathsf{sid}^*}_{j'}$ *does not exist, and $\mathcal{A}$ poses* $\mathsf{StaticReveal}(U_{j'})$.

Security Experiment. For the security definition, we consider the following security experiment. Initially, the adversary, $\mathcal{A}$, is given a set of honest users and makes any sequence of the queries described above. During the experiment, the adversary, $\mathcal{A}$, makes the following query.

- $\mathsf{Test}(\mathsf{sid}^*)$: Here, sid^* must be a fresh session. Select random bit $b \in_R \{0,1\}$, and return the session key held by sid^* if $b = 0$, and return a random key if $b = 1$.

The experiment continues until the adversary, $\mathcal{A}$, makes a guess b'. The adversary, $\mathcal{A}$, *wins* the game if the test session sid^* is still fresh and if the guess of the adversary, $\mathcal{A}$, is correct, i.e., $b' = b$. The advantage of the adversary, $\mathcal{A}$, is defined as $\mathbf{Adv}^{\text{agke}}_{\Pi,\mathcal{A}}(\lambda) = \Pr[\mathcal{A}\ wins] - \frac{1}{2}$ where agke is "g-ck" or "g-ck+" depending on the relying freshness condition. We define the security as follows.

Definition 2.3 (G-CK/G-CK$^+$ **Security).** *We say that a AGKE protocol* Π *is* post-quantum secure in the G-CK/G-CK$^+$ model *if the following conditions hold:*

1. *If all n honest parties complete matching sessions, then, except with negligible probability, they compute the same session key.*
2. *For any PPT quantum adversary $\mathcal{A}$, $\mathbf{Adv}^{\text{g-ck}}_{\Pi,\mathcal{A}}(\lambda)$ (resp. $\mathbf{Adv}^{\text{g-ck+}}_{\Pi,\mathcal{A}}(\lambda)$) is negligible in security parameter κ for the test session* sid^* *according to freshness for* G-CK *(resp.* G-CK$^+$*) security.*

2.2 Cryptographic Invariant Maps

Boneh et al. [1] recently introduced a new framework for constructing non-interactive group key exchange from isogenies on elliptic curves, which is called cryptographic invariant maps (CIM). The notion and assumptions on the CIM systems are introduced here. In Appendix A, we survey some candidates for CIM given in [1].

Definition 2.4 (Freeness and Transitivity [1]). *Let X be a finite set and let G be a finite abelian group. We say that G acts efficiently on X freely and transitively if there are an efficiently computable map $* : G \times X \to X$ and an efficiently computable group operation in G such that:*

- *the map is a group action: $g * (h * x) = (gh) * x$, and there is an identity element $id \in G$ such that $id * x = x$, for all $x \in X$ and all $g, h \in G$;*
- *the action is transitive: for every $(x, y) \in X \times X$ there is a $g \in G$ such that $g * x = y$; and*
- *the action is free: if $x \in X$ and $g, h \in G$ satisfy $g * x = h * x$, then $g = h$.*

The above pair (G, X) gives a convenient conceptual foundation called hard homogeneous spaces (HHSs), which is reviewed in Section 5.1. On the top of HHS given by (G, X), we build the notion of CIM as in the following definition (in a similar manner that bilinear maps are built on the top of cyclic groups in traditional cryptography literatures).

Definition 2.5 (Cryptographic Invariant Map (CIM) [1]). *By a cryptographic invariant map we mean a randomized algorithm* MapGen *that inputs a security parameter λ, outputs public parameters $pp = (X, S, G, e)$, and runs in time polynomial in λ, where:*

- *X and S are sets, and X is finite,*
- *G is a finite abelian group that acts efficiently on X freely and transitively,*
- *e is a deterministic algorithm that runs in time polynomial in λ and n, such that for each $n > 0$, algorithm e takes λ as input and computes a map $e_n : X^n \to S$ that satisfies:*
  - *Invariance property of e_n: for all $x \in X$ and $g_1, \ldots, g_n \in G$, $e_n(g_1 * x, \ldots, g_n * x) = e_n((g_1 \cdots g_n) * x, x, \ldots, x)$;*
  - *Non-degeneracy of e_n: for all i with $1 \le i \le n$ and $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n \in X$, the map $X \to S$ defined by $y \longmapsto e_n(x_1, \ldots, x_{i-1}, y, x_{i+1}, \ldots, x_n)$ is injective.*

The notation $x \leftarrow_R X$ will denote an independent uniform random variable x over the set X. Similarly, we use $x' \leftarrow_R Alg(y)$ to define a random variable x' that is the output of a randomized algorithm Alg on input y.

Definition 2.6 (n-way Computational Diffie–Hellman Assumption [1]). *We say that* MapGen *satisfies the n-way computational Diffie–Hellman assumption (n-CDH) if for every polynomial time quantum algorithm $\mathcal{S}$,*

$$\mathbf{Adv}_{\mathcal{S}}^{n\text{-CDH}}(\lambda) = \Pr[\mathcal{S}(pp, g_1 * x, \ldots, g_n * x) = e_{n-1}((g_1 \cdots g_n) * x, x, \ldots, x)]$$

is a negligible function of λ, when $pp \leftarrow_R$ MapGen(1^λ)*, $g_1, \ldots, g_n \leftarrow_R G$, and $x \leftarrow_R X$.*

Definition 2.7 (n-way Decisional Diffie–Hellman Assumption [1]). *Let consider the following two distributions, $\mathcal{D}_0$ and $\mathcal{D}_1$, where $pp \leftarrow_R$* MapGen(1^λ)*, $g_1, \ldots, g_n \leftarrow_R G$, and $x \leftarrow_R X$:*

- *$\mathcal{D}_0$ is $(pp, g_1 * x, \ldots, g_n * x, s_0)$ where $s_0 = e_{n-1}((g_1 \cdots g_n) * x, x, \ldots, x)$.*
- *$\mathcal{D}_1$ is $(pp, g_1 * x, \ldots, g_n * x, s_1)$ where s_1 is random in $Im(e_{n-1}) \subseteq S$.*

We say that MapGen *satisfies the n-way decisional Diffie–Hellman assumption (n-DDH) if for every polynomial time quantum algorithm $\mathcal{S}$,*

$$\mathbf{Adv}_{\mathcal{S}}^{n\text{-DDH}}(\lambda) = |\Pr[\mathcal{S}(z) = 1 | z \leftarrow \mathcal{D}_0] - \Pr[\mathcal{S}(z) = 1 | z \leftarrow \mathcal{D}_1]|$$

is a negligible function of λ.

Definition 2.8 (n-way Gap Diffie–Hellman Assumption). *We say that* MapGen *satisfies the n-way gap Diffie–Hellman assumption (n-GDH) if for every polynomial time quantum algorithm $\mathcal{S}$ which accesses the n-DDH oracle $\mathcal{O}(\cdot) =$ n-DDH$(\cdot)$,*

$$\mathbf{Adv}_{\mathcal{S}}^{n\text{-GDH}}(\lambda) = \Pr[\mathcal{S}^{\mathcal{O}}(pp, g_1 * x, \ldots, g_n * x) = e_{n-1}((g_1 \cdots g_n) * x, x, \ldots, x)]$$

is a negligible function of λ, when $pp \leftarrow_R$ MapGen(1^λ)*, $g_1, \ldots, g_n \leftarrow_R G$, and $x \leftarrow_R X$. For any input $(pp, x'_1, \ldots, x'_n, s')$ where $x'_i = g'_i * x$ $(i = 1, \ldots n)$, the n-DDH oracle $\mathcal{O}(\cdot) = n$-DDH$(\cdot)$ acts as follows:*

n-DDH$(pp, x'_1, \ldots, x'_n, s') = 0$ *if* $s' = e_{n-1}((g'_1 \cdots g'_n) * x, x, \ldots, x)$*, and*
n-DDH$(pp, x'_1, \ldots, x'_n, s') = 1$ *otherwise.*

In [1], several candidates for CIM are demonstrated including the theta null invariants, Igusa invariants, invariants for Kummer surfaces and Deligne invariants, but they do not give an appropriate one in Definition 2.5. Thus, to obtain a suitable invariant is a big open problem remained in [1].

3 n-UM : G-CK Secure n-Party Authenticated Group Key Exchange

In this section, we propose an one-round n-party AGKE scheme, n-Unified Model (n-UM), secure in the G-CK model. n-UM is based on CIM with MapGen. The security can be proved under the n-DDH assumption for MapGen in the quantum random oracle model.

3.1 Design Principle

To be secure in the G-CK model, the adversary must be prevented to impersonate any party in the test session. In other words, if the adversary does not know any SSK, the session key must be indistinguishable from a random key. Thus, it is necessary that all SSKs contribute to the session key derivation. The session key is an output of a hash function. In n-UM, each user has the static key pair $(t_i, T_i = t_i * x)$, and $Z_1 = e_{n-1}((t_1 \cdots t_n) * x, x, \ldots, x)$ is contained in the input of the hash function. Z_1 can be computed if one of $\{t_i\}$ is given.

On the other hand, in the G-CK model, the adversary can reveal SSKs if the test session has no non-matching session. It means that all EPKs are sent and received without interruption in the test session. In this case, the adversary can compute Z_1 with the revealed SSK. Hence, it is also necessary that all ESKs contribute to the session key derivation. In n-UM, each user generates the ephemeral key pair $(r_i, R_i = r_i * x)$, and $Z_2 = e_{n-1}((r_1 \cdots r_n) * x, x, \ldots, x)$ is contained in the input of the hash function. Z_2 can be computed if one of $\{r_i\}$ is given.

Therefore, all cases of the G-CK model can be covered.

3.2 Useful Techniques for Quantum Random Oracle Model

A problem on security proofs in the quantum random oracle model is how to generate random values for exponentially many positions in order to simulate outputs of the hash function. For a hash function $H : Dom \rightarrow Rng$, in the quantum random oracle model, the adversary poses a superposition $|\phi\rangle = \Sigma \alpha_x |x\rangle$ and the oracle returns $\Sigma \alpha_x |H(x)\rangle$. If Rng is large for a quantum polynomial-time simulator, it is difficult to generate all random output values of H to compute $\Sigma \alpha_x |H(x)\rangle$. Zhandry [35] showed a solution with the notion of k-wise independent function.

A weight assignment on a set $\mathcal{X}$ is a function $D : \mathcal{X} \to \mathbb{R}$ such that $\Sigma_{x \in \mathcal{X}} D(x) = 1$. A distribution on $\mathcal{X}$ is a weight-assignment D such that $D(x) \geq 0$ for all $x \in \mathcal{X}$. Consider the set of functions $H : \mathcal{X} \to \mathcal{Y}$ for sets $\mathcal{X}$ and $\mathcal{Y}$, denoted by $H_{\mathcal{X},\mathcal{Y}}$. We define the marginal weight assignment $D_{\mathcal{W}}$ of D on $H_{\mathcal{X},\mathcal{Y}}$ where the weight of a function $H_{\mathcal{W}} : \mathcal{W} \to \mathcal{Y}$ is equal to the sum of the weights of all $H \in H_{\mathcal{X},\mathcal{Y}}$ that agree with $H_{\mathcal{W}}$ on $\mathcal{W}$.

Definition 3.1 (k-wise equivalence). *We call two weight assignments D_1 and D_2 on $H_{\mathcal{X},\mathcal{Y}}$ k-wise equivalent if for all $\mathcal{W} \subseteq \mathcal{X}$ of size k, the marginal weight assignments $D_{1,\mathcal{W}}$ and $D_{2,\mathcal{W}}$ (of D_1 and D_2) over $H_{\mathcal{X},\mathcal{Y}}$ are identical.*

Definition 3.2 (k-wise independent function). *We call a function f k-wise independent function if f is k-wise equivalent to a random function.*

Lemma 3.1 (Theorem 3.1 in [35]). *Let A be a quantum algorithm making q quantum queries to an oracle $H : \mathcal{X} \to \mathcal{Y}$. If we draw H from some weight assignment D, then for every z, the quantity $\Pr_{H \leftarrow D}[A^H() = z]$ is a linear combination of the quantities $\Pr_{H \leftarrow D}[H(x_i) = r_i \ \forall i \in 1, \ldots, 2q]$ for all possible settings of the x_i and r_i.*

Lemma 3.2 (Theorem 6.1 in [35]). *If there exists $2q_i$-wise independent function, then any quantum algorithm A making q_i quantum queries to random oracles O_i can be efficiently simulated by a quantum algorithm B, which has the same output distribution, but makes no queries.*

Hence, a quantum algorithm B can simulate quantum random oracles in a polynomial-time. We use this simulation technique to simulate outputs of the hash function in the security proof of n-UM.

On the other hand, the other problem on security proofs in the quantum random oracle model is how to insert intended random values as the outputs of corresponding oracle inputs. Zhandry [35] showed a solution with the notion of semi-constant distributions $\mathbf{SC}_\omega$.

Definition 3.3 (Semi-constant distribution). *We define $\mathbf{SC}_\omega$, the semi-constant distribution, as the distribution over $H_{\mathcal{X},\mathcal{Y}}$ resulting from the following process:*

- *First, pick a random element y from $\mathcal{Y}$.*
- *For each $x \in \mathcal{X}$, do one of the following:*
  - *With probability ω, set $H(x) = y$. We call x a distinguished input to H.*
  - *Otherwise, set $H(x)$ to be a random element in $\mathcal{Y}$.*

Lemma 3.3 (Corollary 4.3 in [35]). *The distribution of outputs of a quantum algorithm making n_h queries to an oracle drawn from $\mathbf{SC}_\omega$ is at most a distance $\frac{3}{8} n_h^4 \omega^2$ away from the case when the oracle is drawn from the uniform distribution.*

$T_1 = t_1 * x$	$\cdots$	$T_i = t_i * x$	$\cdots$	$T_n = t_n * x$
$R_1 = r_1 * x$	$\cdots$	$R_i = r_i * x$	$\cdots$	$R_n = r_n * x$
	$\xrightarrow{R_1} \cdots \xleftarrow{R_i}$		$\xrightarrow{R_i} \cdots \xleftarrow{R_n}$	

$$Z_1 = e_{n-1}(T_1, \ldots, T_{i-1}, t_i * T_{i+1}, T_{i+2}, \ldots, T_n)$$
$$Z_2 = e_{n-1}(R_1, \ldots, R_{i-1}, r_i * R_{i+1}, R_{i+2}, \ldots, R_n)$$
$$SK = H(\Pi, U_1, \ldots, U_n, R_1, \ldots, R_n, Z_1, Z_2)$$

Fig. 1. Outline of n-UM Protocol.

We suppose that the simulation succeeds with probability ϵ if the adversary uses an inserted random value as the outputs of corresponding oracle inputs. If the probability that the adversary uses one of the points is ω, then the simulation succeeds with probability $\epsilon\omega - \frac{3}{8}n_h^4\omega^2$. By choosing ω to maximise the success probability, the simulation succeeds with probability $O(\epsilon^2/n_h^4)$. We use this simulation technique to insert a n-DDH instance into the hash function in the security proof of n-UM.

3.3 Protocol

Based on the above principle, we have the n-UM protocol (Fig. 1).

Public Parameters. We set Π = nUM. Let λ be a security parameter. Let MapGen be a generation algorithm of a cryptographic invariant map, and $(X, S, G, e) \leftarrow_R \mathsf{MapGen}(1^\lambda)$ and $x \leftarrow_R X$ are chosen. Let $H : \{0,1\}^* \to \{0,1\}^\lambda$ be a hash function modeled as a quantum random oracle. Public parameters are (Π, X, S, G, e, x, H).

Static Secret and Public Keys. Party U_i chooses $t_i \in G$ as the SSK. Then, U_i computes $T_i = t_i * x$ as the SPK.

Key Exchange. W.l.o.g, we suppose a session executed by $\mathbf{U} = (U_1, \ldots, U_n) \subseteq \mathcal{U}$.

1. U_i chooses $r_i \leftarrow_R G$ as the ESK, and computes $R_i = r_i * x$ as the EPK. Then, U_i broadcasts $(\Pi, \mathsf{role}_{i'}, U_i, R_i)$ to $\mathbf{U} \setminus U_i$.
2. On receiving $(\Pi, \mathsf{role}_{j'}, U_j, R_j)$ for all $j \neq i$, U_i computes $Z_1 = e_{n-1}(T_1, \ldots, T_{i-1}, t_i * T_{i+1}, \ldots, T_n)$ and $Z_2 = e_{n-1}(R_1, \ldots, R_{i-1}, r_i * R_{i+1}, \ldots, R_n)$.[8] Then, U_i generates the session key $SK = H(\Pi, U_1, \ldots, U_n, R_1, \ldots, R_n, Z_1, Z_2)$, and completes the session.

The session state of a session owned by U_i contains the ESK r_i, and intermediate computation R_i. Since other information that is computed after receiving the messages from other parties is immediately erased when the session key is established, such information is not contained in the session state.

[8] T_i and R_i are indexed in the cyclic manner in modulo n. For example, when $i = n$, then $Z_1 = e_{n-1}(t_n * T_1, \ldots, T_n)$ and $Z_2 = e_{n-1}(r_n * R_1, \ldots, R_n)$.

3.4 Security

Theorem 3.1. *Suppose that H is modeled as a quantum random oracle and that the n-DDH assumption holds. Then the n-UM protocol is a post-quantum* G-CK*-secure n-party authenticated key group exchange protocol in the quantum random oracle model.*

In particular, for any quantum adversary $\mathcal{A}$ against the n-UM protocol that runs in time at most t, involves at most n_u honest parties and activates at most n_s sessions, and makes at most n_h queries to the quantum random oracle and n_q SessionReveal *queries, there exists a n-DDH quantum solver $\mathcal{S}$ such that*

$$\mathbf{Adv}_{\mathcal{S}}^{n\text{-DDH}}(\lambda) \geq \frac{2\mathbf{Adv}_{\text{nUM},\mathcal{A}}^{\text{g-ck}}(\lambda)^2}{n_u^2 n_s^2 (8 n_h n_q + 3(n_h + n_q + 1)^4)},$$

where $\mathcal{S}$ runs in time t plus time to perform $\mathcal{O}\big((n_u + n_s)\lambda\big)$ group action operations.

Proof. Since H is modeled as a quantum random oracle, adversary $\mathcal{A}$ has only three ways to distinguish a session key of the test session from a random string.

- Guessing attack: $\mathcal{A}$ correctly guesses the session key.
- Key replication attack: $\mathcal{A}$ creates a session that is not matching to the test session, but has the same session key as the test session.
- Forging attack: $\mathcal{A}$ computes Z_1 and Z_2 used in the test session identified with (Π, role_i, U_j, $U_1, \ldots, U_n$, $R_1, \ldots, R_n$), and queries H with a superposition including ($\Pi, U_1, \ldots, U_n, R_1, \ldots, R_n$, Z_1, Z_2).

Since H is a quantum random oracle, the probability of guessing the output of H is $\mathcal{O}(1/2^\lambda)$. Since non-matching sessions have different communicating parties or ephemeral public keys, key replication is equivalent to finding H-collision; therefore the probability of succeeding key replication is $\mathcal{O}(n_s^2/2^\lambda)$.

Let M be the event that $\mathcal{A}$ wins the security experiment with n-UM, H be the event that $\mathcal{A}$ succeeds forging attack, and $\overline{\mathsf{H}}$ the complementary event of H. Thus we have $\Pr[\mathsf{M} \mid \overline{\mathsf{H}}] = \frac{1}{2}$, and therefore $\mathbf{Adv}_{\text{nUM},\mathcal{A}}^{\text{g-ck}}(\lambda) = \Pr[\mathsf{M}] - \frac{1}{2} \leq \Pr[\mathsf{M} \cap \mathsf{H}]$.

By the definition of freshness in the G-CK-model, there are two cases that $\mathcal{A}$ chooses a test session.

- E_1: $\mathcal{A}$ chooses a test session with a non-matching session.
- E_2: $\mathcal{A}$ chooses a test session without non-matching session, and reveals the static secret keys of the owner of the test session and the owners of its matching sessions.

In each case, we will show how to construct an n-DDH solver $\mathcal{S}$. Solver $\mathcal{S}$ is given a n-DDH instance $(X, S, G, e, x_1 = g_1 * x, \ldots, x_n = g_n * x, s)$.

E_1. $\mathcal{S}$ prepares n_u honest parties, selects n honest parties $\mathbf{U} = (U_1, \ldots, U_n)$ to whom $\mathcal{S}$ assigns the static public keys $\{T_i = x_i\}_{[1,n]}$. The remaining $n_u - n$ parties are assigned random static public and secret key pairs. $\mathcal{S}$ selects $i \leftarrow_R$

$\{1, \ldots, n_s\}$, and chooses i-th session sid^* among sessions, activated by $\mathcal{A}$, owned by U_j, and having intended peers $(U_1, \ldots, U_{j-1}, U_{j+1}, \ldots, U_n)$.

When $\mathcal{A}$ activates sessions containing an honest party except $\mathbf{U}$, $\mathcal{S}$ follows the protocol description. Since $\mathcal{S}$ knows static secret keys of at least one party, it can respond all queries faithfully. The only exception is the session owned by $\mathbf{U}$ because $\mathcal{S}$ does not know static secret keys of them. Then, $\mathcal{S}$ sets s as Z_1 in such sessions.

Also, $\mathcal{S}$ chooses $r_j \leftarrow_R G$ and $\zeta \leftarrow_R \{0,1\}^\lambda$ as the ephemeral secret key and the session key of sid^*, respectively. $R_j = r_j * x$ is the ephemeral public key corresponding to r_j. ζ is inserted as the output of H in the test session sid^* (i.e., the session key).

$\mathcal{S}$ has difficulty in responding hash queries because it needs to return superpositions corresponding to random values for exponentially many positions (The domain of H is $\mathbf{PRS} \times \mathbf{IDS}^n \times X^n \times S^2$). We solve this problem by using Lemma 3.2. Specifically, since the number of queries to H made by $\mathcal{A}$ is n_h for direct queries, n_q for $\mathsf{SessionReveal}$ queries, and one for the Test query, for the total of $n_h + n_q + 1$ queries, a $(n_h + n_q + 1)$-wise independent function is sufficient to simulate superposition of outputs. There is the other difficulty to correctly answer the n-DDH problem because $\mathcal{A}$ uses ζ with exponentially small probability if the position of ζ is only the corresponding input. We can also solve this problem by using Lemma 3.3. Specifically, the simulator inserts ζ in outputs for inputs $(\mathsf{\Pi}, \mathbf{U}', R'_{i_1}, \ldots, R'_{i_n}, Z'_1, Z'_2) \in \mathcal{X} \subset \mathbf{PRS} \times \mathbf{IDS}^n \times X^n \times S^2$. The probability that a randomly chosen input is contained in $\mathcal{X}$ is ω. If $\mathcal{A}$ chooses $(\mathsf{\Pi}, \mathbf{U}' = \mathbf{U}, R'_1, \ldots, R_j, \ldots, R'_n, Z'_1 = s, Z'_2 = e_{n-1}(R'_1, \ldots, R'_{j-1}, r_j * R'_{j+1}, \ldots, R'_n)) \in \mathcal{X}$ as the test session, then $\mathcal{S}$ can use the distinguishing capacity of $\mathcal{A}$ to distinguish the n-DDH challenge.

We use the game hopping technique in the security proof. Let $\mathbf{Adv}^{\text{g-ck},\mathbf{G}_i}_{\text{nUM},\mathcal{A}}(\lambda)$ be the advantage of $\mathcal{A}$ in $\mathbf{G}_i$.

- Let $\mathbf{G}_0$ be the standard attack game for the CK security. When $\mathcal{A}$ poses a superposition to quantum random oracle H, the superposition of output values corresponding to the input is returned to $\mathcal{A}$. Then, $\mathbf{Adv}^{\text{g-ck},\mathbf{G}_0}_{\text{nUM},\mathcal{A}}(\lambda) = \mathbf{Adv}^{\text{g-ck}}_{\text{nUM},\mathcal{A}}(\lambda)$.
- $\mathbf{G}_1$ is the same as $\mathbf{G}_0$ except that the game halts if $\mathcal{A}$ poses $\mathsf{Test}(\mathsf{sid})$ for $\mathsf{sid} \neq \mathsf{sid}^*$. Since sid^* is chosen from $n_u n_s$ sessions, it holds that $\mathbf{Adv}^{\text{g-ck},\mathbf{G}_1}_{\text{nUM},\mathcal{A}}(\lambda) \geq \frac{1}{n_u n_s} \mathbf{Adv}^{\text{g-ck},\mathbf{G}_0}_{\text{nUM},\mathcal{A}}(\lambda)$.
- Let $\omega \in (0,1)$ be chosen later, and $\mathcal{X}$ be a subset of $\mathbf{PRS} \times \mathbf{IDS}^n \times X^n \times S^2$ where $(\mathsf{\Pi}, \mathbf{U}', R'_{i_1}, \ldots, R'_{i_n}, Z'_1, Z'_2) \in \mathbf{PRS} \times \mathbf{IDS}^n \times X^n \times S^2$ is put in $\mathcal{X}$ with independent probability ω. $\mathbf{G}_2$ is the same as $\mathbf{G}_1$ except that the game halts if $(\mathsf{\Pi}, \mathbf{U}, R'_1, \ldots, R_j, \ldots, R'_n, s, e_{n-1}(R'_1, \ldots, R'_{j-1}, r_j * R'_{j+1}, \ldots, R'_n)) \notin \mathcal{X}$ for the test session $\mathsf{sid}^* = (\mathsf{\Pi}, \mathsf{role}_i, U_j, \mathbf{U}, R'_1, \ldots, R_j, \ldots, R'_n)$, $\mathcal{A}$ poses $\mathsf{SessionReveal}(\mathsf{\Pi}, \mathsf{role}_{i'}, U_{j'}, \mathbf{U}', R'_{i_1}, \ldots, R'_{i_n})$ such that $(\mathsf{\Pi}, \mathbf{U}', R'_{i_1}, \ldots, R'_{i_n}, Z'_1, Z'_2) \in \mathcal{X}$, or $\mathcal{A}$ poses $H(\mathsf{\Pi}, \mathbf{U}', R'_{i_1}, \ldots, R'_{i_n}, Z'_1, Z'_2)$ such that $(\mathsf{\Pi}, \mathbf{U}', R'_{i_1}, \ldots, R'_{i_n}, Z'_1, Z'_2) \in \mathcal{X}$. We note that $R'_1, \ldots, R'_{j-1}, R'_{j+1}, \ldots, R'_n$ can be decided by $\mathcal{A}$ because sid^* has no matching session, and $\mathcal{A}$ cannot poses

$\mathsf{SessionReveal}(\Pi, \mathsf{role}_i, U_j, \mathbf{U}, R'_1, \ldots, R_j, \ldots, R'_n)$ by the freshness condition.

$$\begin{aligned}\mathbf{Adv}^{\text{g-ck},\mathbf{G}_2}_{\text{nUM},\mathcal{A}}(\lambda) &\geq \omega(1 - \omega n_h n_q) \cdot \mathbf{Adv}^{\text{g-ck},\mathbf{G}_1}_{\text{nUM},\mathcal{A}}(\lambda) \\ &\geq \omega \mathbf{Adv}^{\text{g-ck},\mathbf{G}_1}_{\text{nUM},\mathcal{A}}(\lambda) - \omega^2 n_h n_q\end{aligned}$$

holds.

- $\mathbf{G}_3$ is the same as $\mathbf{G}_2$ except that ζ is set as $H(\Pi, \mathbf{U}', R'_{i_1}, \ldots, R'_{i_n}, Z'_1, Z'_2)$ for all $(\Pi, \mathbf{U}', R'_{i_1}, \ldots, R'_{i_n}, Z'_1, Z'_2) \in \mathcal{X}$, and hash values are randomly chosen for all other inputs. Now, H is distributed according to $\mathbf{SC}_\omega$. By Lemma 3.3, the output distribution of $\mathcal{A}$ in $\mathbf{G}_3$ is at most a distance $\frac{3}{8}(n_h + n_q + 1)^4\omega^2$ from that in $\mathbf{G}_2$. Hence, $\mathbf{Adv}^{\text{g-ck},\mathbf{G}_3}_{\text{nUM},\mathcal{A}}(\lambda) \geq \mathbf{Adv}^{\text{g-ck},\mathbf{G}_2}_{\text{nUM},\mathcal{A}}(\lambda) - \frac{3}{8}(n_h + n_q + 1)^4\omega^2$ holds.

Finally, we estimate $\mathbf{Adv}^{\text{g-ck},\mathbf{G}_3}_{\text{nUM},\mathcal{A}}(\lambda)$ by $\mathbf{Adv}^{n\text{-DDH}}_{\mathcal{S}}(\lambda)$ with the reduction to n-DDH problem. For simplicity, we assume that $\mathcal{S}$ has quantum access to two random oracles $H_1 : \mathbf{PRS} \times \mathbf{IDS}^n \times X^n \times S^2 \to \{0, 1\}^\lambda$ and $H_2 : \mathbf{PRS} \times \mathbf{IDS}^n \times X^n \times S^2 \to \{0, 1\}$ where H_2 outputs 1 with probability ω. Let $\mathcal{X}$ be the set of $(\Pi, \mathbf{U}', R'_{i_1}, \ldots, R'_{i_n}, Z'_1, Z'_2)$ such that $H_2(\Pi, \mathbf{U}', R'_{i_1}, \ldots, R'_{i_n}, Z'_1, Z'_2) = 1$. We can see that the above conditions are equivalent to $\mathbf{G}_3$. By Lemma 3.2, $\mathcal{S}$ can perfectly simulate H_1 and H_2 by using a $(n_h + n_q + 1)$-wise independent function without oracle accesses. $\mathcal{S}$ prepares R^{list} with entries of the form $(\Pi, \mathsf{role}_{i'}, U_{j'}, \mathbf{U}', R'_{i_1}, \ldots, R'_{i_n}, SK)$ and H^{list} with entries of the form $(\Pi, \mathbf{U}', R'_{i_1}, \ldots, R'_{i_n}, Z'_1, Z'_2, SK)$, and $\mathcal{S}$ maintains two lists for consistent responses to H and $\mathsf{SessionReveal}$ queries. On input $(X, S, G, e, x_1, \ldots, x_n, s)$, $\mathcal{S}$ works as follows:

- Choose $r_j \leftarrow_R G$ and $\zeta \leftarrow_R \{0, 1\}^\lambda$, and set $R_j = r_j * x$ and $\{T_i = x_i\}_{[1,n]}$ in sid^*. The remaining $n_u - n$ parties are assigned random static public and secret key pairs. Set $\mathsf{sid}^* = (\Pi, \mathsf{role}_i, U_j, \mathbf{U}, *, \ldots, R_j, \ldots, *)$.
- $\mathsf{Send}(\Pi, \mathsf{role}_{i'}, U_{j'}, \mathbf{U}', \mathit{Init})$: Solver $\mathcal{S}$ selects uniformly random ephemeral secret key $r'_{j'}$, computes ephemeral public key $R'_{j'} = r'_{j'} * x$ honestly, records $(\Pi, \mathsf{role}_{i'}, U_{j'}, \mathbf{U}', R'_{j'}, *)$ in List R^{list}, and returns it.
- $\mathsf{Send}(\Pi, \mathsf{role}_{i'}, U_{j'}, \mathbf{U}', R'_{i_1}, \ldots, R'_{j'-1}, R'_{j'+1}, \ldots, R'_{i_n})$: Solver $\mathcal{S}$ selects uniformly random ephemeral secret key $r'_{j'}$, and computes ephemeral public key $R'_{j'} = r'_{j'} * x$ honestly. If $\mathbf{U}' \neq \mathbf{U}$, then simulate $H(\Pi, \mathbf{U}', R'_{i_1}, \ldots, R'_{i_n}, e_{n-1}((t_{i_1} \cdots t_{i_n}) * x, x, \ldots, x), e_{n-1}(R'_{i_1}, \ldots, r'_{j'} * R'_{j'+1}, \ldots, R'_{i_n}))$. Otherwise, simulate $H(\Pi, \mathbf{U}, R'_{i_1}, \ldots, R'_{i_n}, s, e_{n-1}(R'_{i_1}, \ldots, r'_{j'} * R'_{j'+1}, \ldots, R'_{i_n}))$. Record $(\Pi, \mathsf{role}_{i'}, U_{j'}, \mathbf{U}', R'_{i_1}, \ldots, R'_{i_n}, SK)$ in List R^{list} as completed, and returns it, where SK is the output of H.
- $\mathsf{Establish}(U_{j'}, T_{j'})$: $\mathcal{S}$ responds to the query faithfully. Note that $\mathsf{Establish}$ for $U_{j'} \in \mathbf{U}$ is never posed by the freshness condition.
- $H(\cdot)$: $\mathcal{S}$ simulates a random oracle such that

$$\begin{aligned}&H(\Pi, \mathbf{U}', R'_{i_1}, \ldots, R'_{i_n}, Z'_1, Z'_2) \\ &\quad = \begin{cases} \zeta & \text{if } H_2(\Pi, \mathbf{U}', R'_{i_1}, \ldots, R'_{i_n}, Z'_1, Z'_2) = 1 \\ H_1(\Pi, \mathbf{U}', R'_{i_1}, \ldots, R'_{i_n}, Z'_1, Z'_2) & \text{otherwise} \end{cases}\end{aligned}$$

- $\mathsf{SessionReveal}(\cdot)$: When $\mathcal{A}$ poses $(\mathsf{\Pi}, \mathsf{role}_{i'}, U_{j'}, \mathbf{U}', R'_{i_1}, \ldots, R'_{i_n})$ such that $H_2(\mathsf{\Pi}, \mathbf{U}', R'_{i_1}, \ldots, R'_{i_n}, Z'_1, Z'_2) = 1$, then outputs a random bit and aborts. Otherwise, return $SK = H_1(\mathsf{\Pi}, \mathbf{U}', R'_{i_1}, \ldots, R'_{i_n}, Z'_1, Z'_2)$.
- $\mathsf{StateReveal}(\mathsf{sid})$: $\mathcal{S}$ responds to the query faithfully.
- $\mathsf{StaticReveal}(C)$: If C is queried before, $\mathcal{S}$ returns error. Otherwise, $\mathcal{S}$ responds to the query faithfully. Note that $\mathsf{StaticReveal}(U_j)$ for $U_j \in \mathbf{U}$ is never posed by the freshness condition.
- $\mathsf{Test}(\mathsf{sid})$: If $\mathsf{sid} \neq \mathsf{sid}^*$, then $\mathcal{S}$ aborts with failure. Otherwise, $\mathcal{S}$ responds ζ to the query.
- If adversary $\mathcal{A}$ outputs guess γ, $\mathcal{S}$ outputs γ.

$\mathcal{S}$ may abort in the simulation of $\mathsf{SessionReveal}$ and Test. Also, $\mathcal{S}$ may fail if $\mathcal{A}$ poses $H(\mathsf{\Pi}, \mathbf{U}', R'_{i_1}, \ldots, R'_{i_n}, Z'_1, Z'_2)$ such that $(\mathsf{\Pi}, \mathbf{U}', R'_{i_1}, \ldots, R'_{i_n}, Z'_1, Z'_2) \in \mathcal{X}$. However, in $\mathbf{G}_3$, these events do not occur because of the game hopping. If $(\mathsf{\Pi}, \mathbf{U}', R'_{i_1}, \ldots, R'_{i_n}, s, Z'_2) \in \mathcal{X}$, then $H(\mathsf{\Pi}, \mathbf{U}', R'_{i_1}, \ldots, R'_{i_n}, s, Z'_2) = \zeta$. In the case of $s = s_0 = e_{n-1}((g_1 \cdots g_n) * x, x, \ldots, x)$, the simulation of Test query is the same as the real session key. In the case of $s = s_1$, the simulation of Test query is the same as the random session key. Thus, $\mathbf{Adv}^{\text{g-ck},\mathbf{G}_3}_{\text{nUM},\mathcal{A}}(\lambda)$ is

$$\mathbf{Adv}^{\text{g-ck},\mathbf{G}_3}_{\text{nUM},\mathcal{A}}(\lambda) = \mathbf{Adv}^{n\text{-DDH}}_{\mathcal{S}}(\lambda).$$

Therefore, $\mathbf{Adv}^{\text{g-ck}}_{\text{nUM},\mathcal{A}}(\lambda)$ is

$$\mathbf{Adv}^{\text{g-ck}}_{\text{nUM},\mathcal{A}}(\lambda) \leq \frac{n_u n_s}{\omega} \mathbf{Adv}^{n\text{-DDH}}_{\mathcal{S}}(\lambda) + n_u n_s \omega \left(n_h n_q + \frac{3}{8}(n_h + n_q + 1)^4 \right).$$

The right side is minimized when $\omega = \frac{4\mathbf{Adv}^{\text{g-ck}}_{\text{nUM},\mathcal{A}}(\lambda)}{n_u n_s (8 n_h n_q + 3(n_h + n_q + 1)^4)}$.

E_2. $\mathcal{S}$ prepares n_u honest parties, selects n honest parties $\mathbf{U} = (U_1, \ldots, U_n)$, and assigns random static public and secret key pairs for all parties (i.e., $\mathcal{S}$ knows all $\{t_j\}_{[1,u]}$). $\mathcal{S}$ selects $i \leftarrow_R \{1, \ldots, n_s\}$, and chooses i-th session sid^* among sessions, activated by $\mathcal{A}$, owned by U_j and having intended peers $(U_1, \ldots, U_{j-1}, U_{j+1}, \ldots, U_n)$.

When $\mathcal{A}$ activates sessions containing honest parties, $\mathcal{S}$ follows the protocol description. Since $\mathcal{S}$ knows static secret keys of at least one peer, it can respond all queries faithfully. In sid^*, $\mathcal{S}$ assigns ephemeral public keys $\{R_j = x_j\}_{[1,n]}$, respectively. Then, $\mathcal{S}$ sets s as Z_2 in sid^*. Also, $\mathcal{S}$ chooses random $\zeta \in \{0,1\}^\lambda$ as the session key of sid^*. ζ is inserted in the output of H in the test session sid^* (i.e., the session key).

We use the game hopping technique in the security proof. Let $\mathbf{Adv}^{\text{g-ck},\mathbf{G}_i}_{\text{nUM},\mathcal{A}}(\lambda)$ be the advantage of $\mathcal{A}$ in $\mathbf{G}_i$.

- Let $\mathbf{G}_0$ be the standard attack game for the CK security. When $\mathcal{A}$ poses a superposition to quantum random oracle H, the superposition of output values corresponding to the input is returned to $\mathcal{A}$. Then, $\mathbf{Adv}^{\text{g-ck},\mathbf{G}_0}_{\text{nUM},\mathcal{A}}(\lambda) = \mathbf{Adv}^{\text{g-ck}}_{\text{nUM},\mathcal{A}}(\lambda)$.

- $\mathbf{G}_1$ is the same as $\mathbf{G}_0$ except that the game halts if $\mathcal{A}$ poses $\mathsf{Test}(\mathsf{sid})$ for $\mathsf{sid} \neq \mathsf{sid}^*$. Since sid^* is chosen from $n_u n_s$ sessions, $\mathbf{Adv}^{\text{g-ck},\mathbf{G}_1}_{\text{nUM},\mathcal{A}}(\lambda) \geq \frac{1}{n_u n_s}\mathbf{Adv}^{\text{g-ck},\mathbf{G}_0}_{\text{nUM},\mathcal{A}}(\lambda)$ holds.
- $\mathbf{G}_2$ is the same as the game, $\mathbf{G}_1$, except that ζ is set as $H(\Pi, \mathbf{U}, R_1, \ldots, R_n, e_{n-1}((t_1 \cdots t_n) * x, x, \ldots, x), s)$, and choose $H(\Pi, \mathbf{U}', R'_{i_1}, \ldots, R'_{i_n}, Z'_1, Z'_2)$ randomly for all other inputs. Now, H is distributed according to $\mathbf{SC}_\omega$ where ω is the probability of randomly selecting $(\Pi, \mathbf{U}, R_1, \ldots, R_n, e_{n-1}((t_1 \cdots t_n) * x, x, \ldots, x), s)$ from the domain, which is negligibly small. By Lemma 3.3, the output distribution of $\mathcal{A}$ in $\mathbf{G}_2$ is at most a distance $\frac{3}{8}(n_h + n_q + 1)^4\omega^2$ from that in $\mathbf{G}_1$. Hence, $\mathbf{Adv}^{\text{g-ck},\mathbf{G}_2}_{\text{nUM},\mathcal{A}}(\lambda) \geq \mathbf{Adv}^{\text{g-ck},\mathbf{G}_1}_{\text{nUM},\mathcal{A}}(\lambda) - \frac{3}{8}(n_h + n_q + 1)^4\omega^2 = \mathbf{Adv}^{\text{g-ck},\mathbf{G}_1}_{\text{nUM},\mathcal{A}}(\lambda) - \text{negl}$ holds.

Finally, we estimate $\mathbf{Adv}^{\text{g-ck},\mathbf{G}_2}_{\text{nUM},\mathcal{A}}(\lambda)$ by using $\mathbf{Adv}^{n\text{-DDH}}_{\mathcal{S}}(\lambda)$. For simplicity, we assume that $\mathcal{S}$ has quantum access to two random oracles $H_1 : \mathbf{PRS} \times \mathbf{IDS}^n \times X^n \times S^2 \rightarrow \{0,1\}^\lambda$ and $H_2 : \mathbf{PRS} \times \mathbf{IDS}^n \times X^n \times S^2 \rightarrow \{0,1\}$ where H_2 outputs 1 with probability ω. By Lemma 3.2, $\mathcal{S}$ can perfectly simulate H_1 and H_2 by using a $(n_h + n_q + 1)$-wise independent function without oracle accesses. $\mathcal{S}$ prepares R^{list} with entries of the form $(\Pi, \mathsf{role}_{i'}, U_{j'}, \mathbf{U}', R'_{i_1}, \ldots, R'_{i_n}, SK)$ and H^{list} with entries of the form $(\Pi, \mathbf{U}', R'_{i_1}, \ldots, R'_{i_n}, Z'_1, Z'_2, SK)$, and $\mathcal{S}$ maintains two lists for consistent responses to H and $\mathsf{SessionReveal}$ queries. On input $(X, S, G, e, x_1, \ldots, x_n, s)$, $\mathcal{S}$ works as follows:

- Choose $t_j \leftarrow_R G$ for all parties and $\zeta \leftarrow_R \{0,1\}^\lambda$, and set $\{T_j = t_j * x\}_{[1,n_u]}$ and $\{R_j = x_j\}_{[1,n]}$ in sid^*. Set $\mathsf{sid}^* = (\Pi, \mathsf{role}_i, U_j, \mathbf{U}, R_1, \ldots, R_n)$.
- $\mathsf{Send}(\Pi, \mathsf{role}_{i'}, U_{j'}, \mathbf{U}', \mathit{Init})$: Solver $\mathcal{S}$ selects uniformly random ephemeral secret key $r'_{j'}$, computes ephemeral public key $R'_{j'} = r'_{j'} * x$ honestly, records $(\Pi, \mathsf{role}_{i'}, U_{j'}, \mathbf{U}', R'_{j'}, *)$ in List R^{list}, and returns it.
- $\mathsf{Send}(\Pi, \mathsf{role}_{i'}, U_{j'}, \mathbf{U}', R'_{i_1}, \ldots, R'_{j'-1}, R'_{j'+1}, \ldots, R'_{i_n})$: Solver $\mathcal{S}$ selects uniformly random ephemeral secret key $r'_{j'}$, and computes ephemeral public key $R'_{j'} = r'_{j'} * x$ honestly. Simulate $H(\Pi, \mathbf{U}', R'_{i_1}, \ldots, R'_{i_n}, e_{n-1}((t_{i_1} \cdots t_{i_n}) * x, x, \ldots, x), e_{n-1}(R'_{i_1}, \ldots, r'_{j'} * R'_{j'+1}, \ldots, R'_{i_n}))$. Record $(\Pi, \mathsf{role}_{i'}, U_{j'}, \mathbf{U}', R'_{i_1}, \ldots, R'_{i_n}, SK)$ in List R^{list} as completed, and returns it, where SK is the output of H.
- $\mathsf{Establish}(U_{j'}, T_{j'})$: $\mathcal{S}$ responds to the query faithfully. Note that $\mathsf{Establish}$ for $U_{j'} \in \mathbf{U}$ is never posed by the freshness condition.
- $H(\cdot)$: $\mathcal{S}$ simulates a random oracle such that

$$\begin{aligned} &H(\Pi, \mathbf{U}', R'_{i_1}, \ldots, R'_{i_n}, Z'_1, Z'_2) \\ &\quad = \begin{cases} \zeta & \text{if } H_2(\Pi, \mathbf{U}', R'_{i_1}, \ldots, R'_{i_n}, Z'_1, Z'_2) = 1 \\ H_1(\Pi, \mathbf{U}', R'_{i_1}, \ldots, R'_{i_n}, Z'_1, Z'_2) & \text{otherwise} \end{cases} \end{aligned}$$

- $\mathsf{SessionReveal}(\cdot)$: When $\mathcal{A}$ poses $(\Pi, \mathsf{role}_{i'}, U_{j'}, \mathbf{U}', R'_{i_1}, \ldots, R'_{i_n})$ such that $H_2(\Pi, \mathbf{U}', R'_{i_1}, \ldots, R'_{i_n}, Z'_1, Z'_2) = 1$, then outputs a random bit and aborts. Otherwise, return $SK = H_1(\Pi, \mathbf{U}', R'_{i_1}, \ldots, R'_{i_n}, Z'_1, Z'_2)$.

- $\mathsf{StateReveal}(\mathsf{sid})$: $\mathcal{S}$ responds to the query faithfully.
- $\mathsf{StaticReveal}(C)$: If C is queried before, $\mathcal{S}$ returns error. Otherwise, $\mathcal{S}$ responds to the query faithfully.
- $\mathsf{Test}(\mathsf{sid})$: If $\mathsf{sid} \neq \mathsf{sid}^*$, then $\mathcal{S}$ aborts with failure. Otherwise, $\mathcal{S}$ responds ζ to the query.
- If adversary $\mathcal{A}$ outputs guess γ, $\mathcal{S}$ outputs γ.

$\mathcal{S}$ may abort in the simulation of $\mathsf{StaticReveal}$ and Test. However, in $\mathbf{G}_2$, these events do not occur because of the game hopping. If $(\Pi, \mathbf{U}', R'_{i_1}, \ldots, R'_{i_n}, Z'_1, s) \in \mathcal{X}$, then $H(\Pi, \mathbf{U}', R'_{i_1}, \ldots, R'_{i_n}, Z'_1, s) = \zeta$. In the case of $s = s_0 = e_{n-1}((g_1 \cdots g_n) * x, x, \ldots, x)$, the simulation of Test query is the same as the real session key. In the case of $s = s_1$, the simulation of Test query is the same as the random session key. Thus, $\mathbf{Adv}^{\text{g-ck},\mathbf{G}_2}_{\text{nUM},\mathcal{A}}(\lambda)$ is

$$\mathbf{Adv}^{\text{g-ck},\mathbf{G}_2}_{\text{nUM},\mathcal{A}}(\lambda) = \mathbf{Adv}^{n\text{-DDH}}_{\mathcal{S}}(\lambda).$$

Therefore, $\mathbf{Adv}^{\text{g-ck}}_{\text{nUM},\mathcal{A}}(\lambda)$ is

$$\mathbf{Adv}^{\text{g-ck}}_{\text{nUM},\mathcal{A}}(\lambda) \leq n_u n_s \cdot \mathbf{Adv}^{n\text{-DDH}}_{\mathcal{S}}(\lambda) + \text{negl}.$$

□

4 Biclique n-DH : G-CK$^+$ Secure n-Party Authenticated Group Key Exchange

In this section, we propose an one-round n-party AGKE scheme, biclique n-Diffie–Hellman (BC n-DH), secure in the G-CK$^+$ model. BC n-DH is based on CIM with MapGen. The security can be proved under the n-GDH assumption for MapGen in the random oracle model.

4.1 Design Principle

To be secure in the G-CK$^+$ model, the protocol resists against maximum exposure. In other words, the session key must be indistinguishable from a random key even if the adversary may obtain either static or ephemeral secret key of each party regarding to the session. Thus, it is necessary that the shared values must contain all combinations of static or ephemeral key of each party.

Hereafter, we use a notation, $I_n = \{1, \ldots, n\}$, and in this notation, n may be omitted as I when it is clear. In BC n-DH, party U_i computes all combinations of T_j and R_j with its static or ephemeral secret key, t_i or r_i. Then, the share values are $Z_\emptyset, \ldots, Z_I$ where their indexes are given as all elements of $\mathcal{P}(I)$, the power set of I, i,e., $Z_\emptyset = e_{n-1}((t_1 \cdots t_n) * x, x, \ldots, x), \ldots, Z_I = e_{n-1}((r_1 \cdots r_n) * x, x, \ldots, x)$. The session key is an output of a hash function whose inputs contains the above shared values.

It is worth to note here that we need to assume that the number of the user group is bounded by logarithm of the security parameter, λ. Otherwise, we need exponential computations in λ as the number of the shared values is 2^n.

4.2 Protocol

Based on the above principle, we have the BC n-DH protocol (Fig. 2).

Public Parameters. We set $\Pi = \text{BCnDH}$. Let λ be a security parameter. Let MapGen be a generation algorithm of a cryptographic invariant map, and $(X, S, G, e) \leftarrow_R \mathsf{MapGen}(1^\lambda)$ and $x \leftarrow_R X$ are chosen. Let $H : \{0,1\}^* \to \{0,1\}^\lambda$ be a hash function modeled as a random oracle. Public parameters are (Π, X, S, G, e, x, H).

Static Secret and Public Keys. Party U_i chooses $t_i \in G$ as the SSK. Then, U_i computes $T_i = t_i * x$ as the SPK.

Key Exchange. As in Section 3, we suppose a session executed by $\mathbf{U} = (U_1, \ldots, U_n) \subseteq \mathcal{U}$.

Note that the role identifier is decided by the lexicographic order of party identities, and thus i, the suffix of the the role identifier, role_i, can be computed as $i = f_{\mathbf{U}}(U_j)$ with a function, $f_{\mathbf{U}}$, when $\mathbf{U}$ is fixed. Hereafter, we omit the explanation regarding to the suffix of the the role identifier. In addition, we express that the role identifiers can be express with the elements in $\{0,1\}^{|n|}$ as their variety is n.

1. U_i chooses $r_i \leftarrow_R G$ as the ESK, and computes $R_i = r_i * x$ as the EPK. Then, U_i broadcasts $(\Pi, \mathsf{role}_{i'}, U_i, R_i)$ to $\mathbf{U} \setminus U_i$.
2. On receiving $(\Pi, \mathsf{role}_{j'}, U_j, R_1, \ldots, R_n)$, U_i computes $Z_\emptyset = e_{n-1}(T_1, \ldots, T_{i-1}, t_i{*}T_{i+1}, T_{i+2}, \ldots, T_n), \ldots, Z_I = e_{n-1}(R_1, \ldots, R_{i-1}, r_i{*}R_{i+1}, R_{i+2}, \ldots, R_n)$ as follows:[9] for all $P \in \mathcal{P}(I)$,
   - if $i \in P$, then $v_i = r_i$, and else if $i \notin P$, then $v_i = t_i$,
   - for all $k \in I$ $(k \neq i)$, if $k \in P$, then $V_k = R_k$, and else if $k \notin P$, then $V_k = T_k$, and
   - U_i computes Z_P as $Z_P = e_{n-1}(V_1, \ldots, V_{i-1}, v_i * V_{i+1}, V_{i+2}, \ldots, V_n)$.

   Then, U_i generates the session key $SK = H(\Pi, U_1, \ldots, U_n, R_1, \ldots, R_n, Z_\emptyset, \ldots, Z_I)$, and completes the session.

It is clear that $Z_P = e_{n-1}((v_1 \cdots v_n) * x, x, \ldots, x)$ for all P $(\in \mathcal{P}(I))$ where $v_k = r_k$ for $k \in P$ and $v_k = t_k$ for $k \notin P$ as $e_{n-1}((v_1 \cdots v_n) * x, x, \ldots, x) = e_{n-1}(V_1, \ldots, V_{i-1}, v_i * V_{i+1}, V_{i+2}, \ldots, V_n)$ and $V_k = v_k * x$ for all $k \in I$, and thus, every $U_i \in \mathbf{U}$ can share the same session key, SK.

The session state of a session owned by U_i contains the ESK r_i, and intermediate computation R_i. Since other information that is computed after receiving the messages from other parties is immediately erased when the session key is established, such information is not contained in the session state.

[9] T_i and R_i are indexed in the cyclic manner in modulo n.

$$
\begin{array}{ccccc}
T_1 = t_1 * x & \cdots & T_i = t_i * x & \cdots & T_n = t_n * x \\
\hline
R_1 = r_1 * x & \cdots & R_i = r_i * x & \cdots & R_n = r_n * x \\
 & \xrightarrow{R_1} \;\cdots\; \xleftarrow{R_i} & & \xrightarrow{R_i} \;\cdots\; \xleftarrow{R_n} & \\
\hline
\multicolumn{5}{c}{Z_\emptyset = e_{n-1}(T_1, \ldots, T_{i-1}, t_i * T_{i+1}, T_{i+2}, \ldots, T_n)} \\
\multicolumn{5}{c}{\vdots} \\
\multicolumn{5}{c}{Z_I = e_{n-1}(R_1, \ldots, R_{i-1}, r_i * R_{i+1}, R_{i+2}, \ldots, R_n)} \\
\multicolumn{5}{c}{SK = H(\Pi, U_1, \ldots, U_n, R_1, \ldots, R_n, Z_\emptyset, \ldots, Z_I)}
\end{array}
$$

Fig. 2. Outline of Biclique n-DH Protocol.

4.3 Security

Theorem 4.1. *Suppose that H is modeled as a random oracle and that the n-way GDH assumption holds for $\mathcal{S}$. Then the biclique n-DH protocol is a post-quantum* G-CK$^+$ *secure n-party authenticated group key exchange protocol in the random oracle model.*

In particular, for any AGKE quantum adversary $\mathcal{A}$ against the biclique n-DH protocol that runs in time at most t, involves at most n_u honest parties and activate at most n_s sessions, and makes at most n_h queries to the random oracle, there exists a n-way GDH quantum solver $\mathcal{S}$ such that

$$\mathbf{Adv}_{\mathcal{S}}^{n\text{-GDH}}(\lambda) \geq \min\left\{\frac{1}{n_u^n}, \frac{1}{n_u^{n-1} n_s}, \ldots, \frac{1}{n_u n_s^{n-1}}, \frac{1}{n_s^n}\right\} \cdot \mathbf{Adv}_{\text{BCnDH},\mathcal{A}}^{\text{g-ck+}}(\lambda),$$

where $\mathcal{S}$ runs in time t plus time to perform $\mathcal{O}\big((n_u + n_s)\lambda\big)$ group action operations and make $\mathcal{O}(n_h + n_s)$ queries to the n-DDH oracle.

Proof. Since H is modeled as a random oracle, adversary $\mathcal{A}$ has only three ways to distinguish a session key of the test session from a random string.

- Guessing attack: $\mathcal{A}$ correctly guesses the session key.
- Key replication attack: $\mathcal{A}$ creates a session that is not matching to the test session, but has the same session key as the test session.
- Forging attack: $\mathcal{A}$ computes $Z_\emptyset, \ldots, Z_I$ used in the test session identified with (Π, $U_1, \ldots,$ U_n, $R_1, \ldots,$ R_n), and queries H with (Π, $U_1, \ldots,$ U_n, $R_1, \ldots,$ R_n, $Z_\emptyset$, $\ldots$, Z_I).

Since H is a random oracle, the probability of guessing the output of H is $\mathcal{O}(1/2^\lambda)$. Since non-matching sessions have different communicating parties or ephemeral public keys, key replication is equivalent to finding H-collision; therefore the probability of succeeding key replication is $\mathcal{O}(n_s^2/2^\lambda)$. However to detect collision the adversary has to query with both inputs the random oracle, in particular query with $Z_\emptyset, \ldots,$ Z_I used in the test session as describe in Forging attack above.

Let M be the event that $\mathcal{A}$ wins the security experiment with BCnDH, H be the event that $\mathcal{A}$ succeeds forging attack, and $\overline{\mathsf{H}}$ the complementary event of H.

Thus we have $\Pr[\mathsf{M} \mid \overline{\mathsf{H}}] = \frac{1}{2}$, and therefore

$$\mathbf{Adv}^{\text{g-ck+}}_{\text{BCnDH},\mathcal{A}}(\lambda) = \Pr[\mathsf{M}] - \frac{1}{2} \leq \Pr[\mathsf{M} \cap \mathsf{H}]. \qquad (1)$$

By the definition of freshness in the G-CK$^+$ model, there are six cases that $\mathcal{A}$ chooses a test session.

- E_1: $\mathcal{A}$ chooses a test session without a matching session, and does not reveal the ephemeral secret key of the owner of the test session.
- E_2: $\mathcal{A}$ chooses a test session without a matching session, and does not reveal the static secret key of the owner of the test session.
- E_3: $\mathcal{A}$ chooses a test session with matching sessions, and does not reveal the ephemeral secret keys of the owner of the test session and of the owners of its matching sessions.
- E_4: $\mathcal{A}$ chooses a test session with matching sessions, and does not reveal the static secret keys of the owner of the test session and of the owners of its matching sessions.
- E_5: $\mathcal{A}$ chooses a test session with matching sessions, and does not reveal the ephemeral secret key of the owner of the test session and the static secret keys of the owners of its all matching sessions.
- E_6: $\mathcal{A}$ chooses a test session with matching sessions, and does not reveal the static secret key of the owner of the test session and the ephemeral secret keys of the owners of its all matching sessions.
- E_7: $\mathcal{A}$ chooses a test session with matching sessions, and does not reveal the ephemeral secret key of the owner of the test session, the ephemeral secret keys of the owners of some matching sessions, and the static secret keys of the owners of the other matching sessions.
- E_8: $\mathcal{A}$ chooses a test session with matching sessions, and does not reveal the static secret key of the owner of the test session, the static secret keys of the owners of some matching sessions, and the ephemeral secret keys of the owners of the other matching sessions.

In each case, we will show how to construct an n-GDH solver $\mathcal{S}$. Solver $\mathcal{S}$ is given an n-GDH instance $(\mathcal{S})$. Hereafter, MHE_i $(i = 1, \ldots, 6)$ denotes event $\mathsf{M} \cap \mathsf{H} \cap \mathsf{E}_i$.

At the end of the experiment, it is decided which event occurred. In other words for each event analysis below it is assumed that the event conditions are satisfied upon the adversary termination.

Before analyzing the events, we note that the session state of a session in the biclique n-DH protocol is equivalent to the ephemeral secret key in the session as no other information (except the static secret key) is necessary to compute the shared secrets and the session key.

E_1. $\mathcal{S}$ prepares n_u honest parties, selects $(n-1)$ party $\bar{U}_j$ $(j = 2, \ldots, n)$ to whom $\mathcal{S}$ assigns the static public key $T_j = x_j$. The remaining $(n_u - n + 1)$ parties are assigned random static public and secret key pairs. $\mathcal{S}$ selects $\bar{i} \leftarrow_R \{1, \ldots, n_s\}$ and chooses $\bar{i}$-th session sid^* among sessions, activated by $\mathcal{A}$ and owned by an honest party, $\bar{U}_1$, different from $\bar{U}_j$.

When $\mathcal{A}$ activates sessions between honest peers, $\mathcal{S}$ follows the protocol description. Since $\mathcal{S}$ knows static secret keys of at least one peer, it can respond all queries faithfully. The only exception is the session sid^*, for which $\mathcal{S}$ sets ephemeral public key of sid^* to x_1, and chooses a random $\zeta \in \{0,1\}^\lambda$ as the session key of sid^*.

The simulator has difficulty in responding queries related to $\bar{U}_j$ because $\mathcal{S}$ does not know the static secret key of $\bar{U}_j$. More precisely, for sessions owned by $\bar{U}_j$ with peers $\{U'_i\}$ controlled by $\mathcal{A}$, $\mathcal{S}$ cannot compute the shared secrets, $Z_\emptyset, \ldots,$ Z_I, but may have to answer $\mathsf{SessionReveal}$ queries. $\mathcal{A}$ could also derive session keys of these session by computing the shared secrets, $Z_\emptyset, \ldots, Z_I$, and query H. If these 2^n values, $Z_\emptyset, \ldots, Z_I$, do not coincide, then $\mathcal{S}$ fails its simulation. To handle this situations, $\mathcal{S}$ prepares $\mathrm{R}^{\mathrm{list}}$ with entries of the form (pid, rid, uid, $\mathrm{uid}_1, \ldots, \mathrm{uid}_n$, $W_1, \ldots, W_n$, $SK) \in \mathbf{PRS} \times \{0,1\}^{|n|} \times \mathbf{IDS} \times \mathbf{IDS}^n \times X^n \times \{0,1\}^\lambda$ and $\mathrm{H}^{\mathrm{list}}$ with entries of the form (pid, $\mathrm{uid}_1, \ldots, \mathrm{uid}_n$, $W_1, \ldots, W_n$, $Z_\emptyset, \ldots, Z_I$, $SK) \in \mathbf{PRS} \times \mathbf{IDS}^n \times X^n \times S^{2^n} \times \{0,1\}^\lambda$, where pid is a string which gives a protocol identifier, rid is a string which gives a role identifier, and uid_i is a string which gives a user identifier, and $\mathcal{S}$ maintains two lists for consistent responses to H and $\mathsf{SessionReveal}$ queries as follows. Below, Y is generated by $\mathcal{S}$ on behalf of U_j.

- $\mathsf{Send}(\Pi,\ \mathsf{role}_i,\ U_j,\ U_1, \ldots,\ U_n,\ \mathit{Init})$: Solver $\mathcal{S}$ selects uniformly random ephemeral secret key r_j, computes ephemeral public key $R_j = r_j * x$ honestly, records $(\Pi,\ \mathsf{role}_i,\ U_j,\ U_1, \ldots,\ U_n,\ R_j)$ in List $\mathrm{R}^{\mathrm{list}}$, and returns R_j.
- $\mathsf{Send}(\Pi,\ \mathsf{role}_i,\ U_j,\ U_1, \ldots,\ U_n,\ R_1, \ldots,\ R_n)$: If session $(\Pi,\ \mathsf{role}_i,\ U_j,\ U_1, \ldots,$ $U_n,\ R_j)$ is not recorded in List $\mathrm{R}^{\mathrm{list}}$, $\mathcal{S}$ records session $(\Pi,\ \mathsf{role}_i,\ U_j,\ U_1, \ldots,$ $U_n,\ R_1, \ldots,\ R_n)$ in List $\mathrm{R}^{\mathrm{list}}$ as not completed. Otherwise, $\mathcal{S}$ records the session in List $\mathrm{R}^{\mathrm{list}}$ as completed.
- $\mathsf{Establish}(U_{j'}, T_{j'})$: $\mathcal{S}$ responds to the query faithfully. Note that $\mathsf{Establish}$ for $U_{j'} \in \mathbf{U}$ is never posed by the freshness condition.
- $H(\cdot)$: $\mathcal{S}$ simulates a random oracle in the usual way except for queries of the form $(\Pi, U_1, \ldots, U_n, R_1, \ldots, R_n, Z_\emptyset, \ldots, Z_I)$. When $(\Pi, U_1, \ldots, U_n, R_1, \ldots,$ $R_n,\ Z_\emptyset,\ \ldots,\ Z_I)$ is queried, $\mathcal{S}$ responds to these queries in the following way:
    - if $(\Pi,\ U_1, \ldots,\ U_n,\ R_1, \ldots,\ R_n,\ Z_\emptyset, \ldots,\ Z_I,\ SK) \in \mathrm{H}^{\mathrm{list}}$ for some SK, $\mathcal{S}$ returns SK to $\mathcal{A}$.
    - else if the validity conditions, n-DDH$(pp,\ V_1, \ldots,\ V_n,\ Z_P) = 1$, hold for all $P \in \mathcal{P}(I)$ where $V_k = R_k$ for $k \in P$ and $V_k = T_k$ for $k \notin P$,
        - ∗ then if there exists $(\Pi,\ \mathsf{role}_i,\ U_j,\ U_1, \ldots,\ U_n,\ R_1, \ldots,\ R_n,\ SK) \in \mathrm{R}^{\mathrm{list}}$ for some i, $\mathcal{S}$ returns SK;
        - ∗ otherwise, $\mathcal{S}$ chooses $SK \leftarrow_R \{0,1\}^\lambda$, returns SK and stores $(\Pi,$ $\mathsf{role}_i,\ U_j,\ U_1, \ldots,\ U_n,\ R_1, \ldots,\ R_n,\ SK)$ in $\mathrm{R}^{\mathrm{list}}$ for all U_j $(j = 1, \ldots, n)$. $\mathcal{S}$ also stores the new tuple $(\Pi,\ U_1, \ldots,\ U_n,\ R_1, \ldots,\ R_n,$ $Z_\emptyset, \ldots,\ Z_I,\ SK)$ in $\mathrm{H}^{\mathrm{list}}$.
    - else $\mathcal{S}$ choose $SK \leftarrow_R \{0,1\}^\lambda$, returns it to $\mathcal{A}$ and stores the new tuples $(\Pi,\ \mathsf{role}_i,\ U_j,\ U_1, \ldots,\ U_n,\ R_1, \ldots,\ R_n,\ Z_\emptyset, \ldots,\ Z_I,\ SK)$ in $\mathrm{H}^{\mathrm{list}}$ for all U_j $(j = 1, \ldots, n)$.

- $\mathsf{SessionReveal}(\cdot)$: $\mathcal{S}$ simulates these queries in the usual way except for queries of the form (Π, role_i, $U_j, U_1, \ldots, U_n, R_1, \ldots, R_n$). When ($\Pi$, role_i, $U_j, U_1, \ldots,$ U_n, $R_1, \ldots R_n$) is queried, $\mathcal{S}$ does one of the following:
  - if there is no session with identifier (Π, role_i, U_j, $U_1, \ldots,$ U_n, $R_1, \ldots,$ R_n), the query is aborted.
  - else if (Π, role_i, U_j, $U_1, \ldots,$ U_n, $R_1, \ldots,$ R_n, SK) $\in \mathrm{R}^{\mathrm{list}}$ for some SK, $\mathcal{S}$ returns SK to $\mathcal{A}$.
  - else if (Π, $U_1, \ldots,$ U_n, $R_1, \ldots,$ R_n, $Z_\emptyset, \ldots,$ Z_I, SK) $\in \mathrm{H}^{\mathrm{list}}$ such that n-DDH(pp, $V_1, \ldots,$ V_n, Z_P) $= 1$, hold for all $P \in \mathcal{P}(I)$ where $V_k = R_k$ for $k \in P$ and $V_k = T_k$ for $k \notin P$, $\mathcal{S}$ returns SK and stores the new tuple (Π, role_i, U_j, $U_1, \ldots,$ U_n, $R_1, \ldots,$ R_n, SK) in $\mathrm{R}^{\mathrm{list}}$.
- $\mathsf{StateReveal}(\mathsf{sid})$: If the corresponding ephemeral public key is x_1, then solver $\mathcal{S}$ aborts with failure. Otherwise, solver $\mathcal{S}$ responds to the query faithfully.
- $\mathsf{StaticReveal}(U_{j'})$: If $U_{j'}$ is queried before, $\mathcal{S}$ returns error. Otherwise, $\mathcal{S}$ responds to the query faithfully.
- $\mathsf{Test}(\mathsf{sid})$: If sid is not the $\bar{\imath}$-th session of $\bar{U}_1$, then solver $\mathcal{S}$ aborts with failure. Otherwise, solver $\mathcal{S}$ responds to the query faithfully.
- If adversary $\mathcal{A}$ outputs guess γ, solver $\mathcal{S}$ aborts with failure.

Provided that E_1 occurs and $\mathcal{A}$ selects sid^* as the test session with peer $\bar{U}_j$ ($j = 2, \ldots, n$), the simulation does not fail. In this case, the session identifier of sid^* is (Π, role_i, $\bar{U}_1$, $\bar{U}_1, \ldots,$ $\bar{U}_n$, $R_1, \ldots,$ R_n), where $R_1 = x_1$ and other $\{R_i\}$ are the incoming ephemeral public keys of sid^*. If $\mathcal{A}$ wins the security game, it must have queried H with inputs $Z_P = n$-CDH(pp, $V_1, \ldots,$ V_n) for all $P \in \mathcal{P}(I)$ where $V_k = R_k$ for $k \in P$ and $V_k = T_k$ for $k \notin P$. To solve the n-CDH instance, $\mathcal{S}$ checks if there is an H query made by $\mathcal{A}$ of the form (Π, $\bar{U}_1, \ldots,$ $\bar{U}_n$, $R_1, \ldots,$ R_n, $Z_\emptyset, \ldots,$ Z_I), such that n-DDH(pp, $V_1, \ldots,$ V_n, Z_P) $= 1$, hold for all $P \in \mathcal{P}(I)$ where $V_k = R_k$ for $k \in P$ and $V_k = T_k$ for $k \notin P$. If such an H query exists, $\mathcal{S}$ outputs $Z_{\{2,\ldots,n\}}$ as the n-CDH answer where $Z_{\{2,\ldots,n\}} = n$-CDH(pp, R_1, $T_2, \ldots,$ T_n) $= n$-CDH(pp, $x_1, \ldots,$ x_n). With probability at least $\frac{1}{n_u^{n-1} n_s}$, the test session is sid^* with owner $\bar{U}_1$ and peers $\{\bar{U}_j\}$ ($j = 2, \ldots, n$). Thus the advantage of $\mathcal{S}$ is

$$\mathbf{Adv}_{\mathcal{S}}^{n\text{-GDH}}(\lambda) \geq \frac{1}{n_u^{n-1} n_s} \cdot \Pr[\mathsf{MHE}_1]. \tag{2}$$

Notice that in the above simulation $\mathcal{S}$ cannot respond to $\mathsf{StaticReveal}(\bar{U}_j)$ query. However, given that event E_1 occurs, $\mathcal{S}$ correctly guesses the test session and the test session is fresh at the end of the experiment, then $\mathcal{A}$ have not queried for the static secret keys of the test session whose peers are $\{\bar{U}_j\}$.

Such static key reveal queries would contradict the freshness of the test session and thus the simulation terminated without errors.

E_2. $\mathcal{S}$ prepares n_u honest parties, selects n distinct honest parties $\bar{U}_j$ ($j = 1, \ldots, n$), and assigns $\bar{U}_j$'s static public key as $T_j = x_j$, respectively. $\mathcal{S}$ assigns random static public and secret key pairs for the remaining ($n_u - n$) parties. $\mathcal{S}$ follows the protocol description when $\mathcal{A}$ activates session between honest peers, and simulate $\mathcal{A}$'s queries related to $\bar{U}_j$ as explained in E_1.

If $\mathcal{A}$ selected a session whose participants are $(\bar{U}_1, \ldots, \bar{U}_n)$ as the test session, and E_2 occurs, this simulation does not fail. Let $(\Pi, \mathsf{role}_i, \bar{U}_j, \bar{U}_1, \ldots, \bar{U}_n, R_1, \ldots, R_n)$ be the session identifier of the test session. Note that $\mathcal{S}$ generated R_i and so knows r_i. When $\mathcal{A}$ is successful, $\mathcal{S}$ checks if there is an H query made by $\mathcal{A}$ of the form $(\Pi, \bar{U}_1, \ldots, \bar{U}_n, R_1, \ldots, R_n, Z_\emptyset, \ldots, Z_I)$, such that n-DDH$(pp, V_1, \ldots, V_n, Z_P) = 1$, hold for all $P \in \mathcal{P}(I)$ where $V_k = R_k$ for $k \in P$ and $V_k = T_k$ for $k \notin P$. If such an H query exists, $\mathcal{S}$ outputs Z_I as the n-CDH answer where $Z_I = n$-CDH$(pp, T_1, \ldots, T_n) = n$-CDH$(pp, x_1, \ldots, x_n)$. With probability at least $\frac{1}{n_u^n}$, $\mathcal{A}$ will select a test session with owner $\bar{U}_j$ and peers $\{\bar{U}_{j'}\}$ $(j' = 1, \ldots, j-1, j+1, \ldots, n)$. Thus, the advantage of $\mathcal{S}$ is

$$\mathbf{Adv}_{\mathcal{S}}^{n\text{-GDH}}(\lambda) \geq \frac{1}{n_u^n} \cdot \Pr[\mathsf{MHE}_2]. \tag{3}$$

$\mathcal{S}$ cannot respond to any $\mathsf{StaticReveal}(\bar{U}_j)$ queries during the simulation. As before if event E_2 occurs, $\mathcal{S}$ correctly guesses the test session and the test session is fresh at the end of the experiment, then $\mathcal{A}$ have not queried for the static secret key of $\bar{U}_j$, and therefore the simulation does not terminate with error.

E_3. In the beginning, we explain the case where the test session has single matching session for simplicity. Then, we extend it to the case where the test session has several matching sessions later.

$\mathcal{S}$ prepares n_u honest parties, selects $(n-2)$ party $\bar{U}_j$ $(j = 3, \ldots, n)$ to whom $\mathcal{S}$ assigns the static public key $T_j = x_j$. The remaining $(n_u - n + 2)$ parties are assigned random static public and secret key pairs. $\mathcal{S}$ selects $\bar{i}, \bar{j} \leftarrow_R \{1, \ldots, n_s\}$, and chooses $\bar{i}$-th session sid^* and $\bar{j}$-th session $\overline{\mathsf{sid}^*}$ among sessions activated by $\mathcal{A}$ and owned by honest parties, $\bar{U}_1$ and $\bar{U}_2$, respectively. When activated, $\mathcal{S}$ sets the ephemeral public key of sid^* to be x_1 and of $\overline{\mathsf{sid}^*}$ to be x_2. Since $\mathcal{S}$ knows the static secret keys of all honest parties, it can respond all queries, faithfully, except those that related to sid^* and $\overline{\mathsf{sid}^*}$.

Provided that $\mathcal{A}$ selects sid^* as the test session, $\overline{\mathsf{sid}^*}$ as its matching session, and E_3 occurs, the simulation does not fail. Let $(\Pi, \mathsf{role}_i, \bar{U}_1, \bar{U}_1, \ldots, \bar{U}_n, R_1, \ldots, R_n)$ and $(\Pi, \mathsf{role}_{\hat{i}}, \bar{U}_2, \bar{U}_1, \ldots, \bar{U}_n, R_1, \ldots, R_n)$ be the session identifiers of sid^* and $\overline{\mathsf{sid}^*}$, respectively. When $\mathcal{A}$ wins the security game, $\mathcal{S}$ checks if there is an H query made by $\mathcal{A}$ of the form $(\Pi, \bar{U}_1, \ldots, \bar{U}_n, R_1, \ldots, R_n, Z_\emptyset, \ldots, Z_I)$, such that n-DDH$(pp, V_1, \ldots, V_n, Z_P) = 1$, hold for all $P \in \mathcal{P}(I)$ where $V_k = R_k$ for $k \in P$ and $V_k = T_k$ for $k \notin P$. If such an H query exists, $\mathcal{S}$ outputs $Z_{\{3,\ldots,n\}}$ as the n-CDH answer. With probability at least $\frac{1}{n_u^{n-2} n_s^2}$, $\mathcal{A}$ selects sid^* as the test session and $\overline{\mathsf{sid}^*}$ as its matching session.

$\mathcal{S}$ cannot respond to $\mathsf{StateReveal}$ queries against the test session and its matching during the simulation. However, under event E_3 adversary does not issue such queries, and hence the simulation does not fail.

In the case where the test session has two matching sessions. $\mathcal{S}$ prepares n_u honest parties, selects $(n-3)$ party $\bar{U}_j$ $(j = 4, \ldots, n)$ to whom $\mathcal{S}$ assigns the static public key $T_j = x_j$. $\mathcal{S}$ selects $\bar{i}, \bar{j}, \bar{k} \leftarrow_R \{1, \ldots, n_s\}$, and chooses $\bar{i}$-th session sid^*, and $\bar{j}$-th session $\overline{\mathsf{sid}^*}$, and $\bar{k}$-th session $\overline{\mathsf{sid}^*}'$ (the other matching session) among sessions activated by $\mathcal{A}$ and owned by honest parties $\bar{U}_1$, $\bar{U}_2$, and $\bar{U}_3$, respectively. When activated, $\mathcal{S}$ sets the ephemeral public key of sid^* to be x_1, of $\overline{\mathsf{sid}^*}$ to be x_2, and of $\overline{\mathsf{sid}^*}'$ to be x_3.

When $\mathcal{A}$ wins the security game, $\mathcal{S}$ checks if there is an H query made by $\mathcal{A}$ of the form (Π, $\bar{U}_1, \ldots,$ $\bar{U}_n$, $R_1, \ldots,$ R_n, $Z_\emptyset, \ldots,$ Z_I), such that n-DDH(pp, $V_1, \ldots, V_n$, Z_P) $= 1$, hold for all $P \in \mathcal{P}(I)$ where $V_k = R_k$ for $k \in P$ and $V_k = T_k$ for $k \notin P$. If such an H query exists, $\mathcal{S}$ outputs $Z_{\{4,\ldots,n\}}$ as the n-CDH answer. With probability at least $\frac{1}{n_u^{n-3} n_s^3}$, $\mathcal{A}$ selects sid^* as the test session, $\overline{\mathsf{sid}^*}$ as its matching session, and $\overline{\mathsf{sid}^*}'$ as its other matching session.

It is easy to extend the above consideration to the case of where the test session has several matching sessions. When the number of the matching sessions is k ($k \leq n-1$), with probability at least $\frac{1}{n_u^{n-(k+1)} n_s^{k+1}}$, $\mathcal{A}$ selects the test session and its matching sessions. Thus, the advantage of $\mathcal{S}$ is totally given as

$$\mathbf{Adv}_{\mathcal{S}}^{n\text{-GDH}}(\lambda) \geq \min\left\{\frac{1}{n_u^{n-2} n_s^2}, \frac{1}{n_u^{n-3} n_s^3}, \ldots, \frac{1}{n_s^n}\right\} \cdot \Pr[\mathsf{MHE}_3]. \tag{4}$$

E_4, E_5, E_6, E_7, and E_8. The analysis of E_4, E_5, E_6, E_7, and E_8 is similar to E_2, E_1, E_3, E_3, and E_1, respectively. We omit the details and provide only the conclusion. In each case, we can construct an n-GDH solver $\mathcal{S}$ as follows.

$$\mathbf{Adv}_{\mathcal{S}}^{n\text{-GDH}}(\lambda) \geq \frac{1}{n_u^n} \cdot \Pr[\mathsf{MHE}_4], \tag{5}$$

$$\mathbf{Adv}_{\mathcal{S}}^{n\text{-GDH}}(\lambda) \geq \frac{1}{n_u^{n-1} n_s} \cdot \Pr[\mathsf{MHE}_5], \tag{6}$$

$$\mathbf{Adv}_{\mathcal{S}}^{n\text{-GDH}}(\lambda) \geq \min\left\{\frac{1}{n_u^{n-1} n_s}, \frac{1}{n_u^{n-2} n_s^2}, \ldots, \frac{1}{n_u n_s^{n-1}}\right\} \cdot \Pr[\mathsf{MHE}_6], \tag{7}$$

$$\mathbf{Adv}_{\mathcal{S}}^{n\text{-GDH}}(\lambda) \geq \min\left\{\frac{1}{n_u^{n-2} n_s^2}, \frac{1}{n_u^{n-3} n_s^3}, \ldots, \frac{1}{n_u n_s^{n-1}}\right\} \cdot \Pr[\mathsf{MHE}_7], \tag{8}$$

$$\mathbf{Adv}_{\mathcal{S}}^{n\text{-GDH}}(\lambda) \geq \min\left\{\frac{1}{n_u^{n-1} n_s}, \frac{1}{n_u^{n-2} n_s^2}, \ldots, \frac{1}{n_u n_s^{n-1}}\right\} \cdot \Pr[\mathsf{MHE}_8]. \tag{9}$$

Analysis. Combining (1), ..., (9), we have

$$\mathbf{Adv}_{\mathcal{S}}^{n\text{-GDH}}(\lambda) \geq \min\left\{\frac{1}{n_u^n}, \frac{1}{n_u^{n-1} n_s}, \ldots, \frac{1}{n_u n_s^{n-1}}, \frac{1}{n_s^n}\right\} \cdot \mathbf{Adv}_{\text{BCnDH},\mathcal{A}}^{\text{g-ck+}}(\lambda).$$

During the simulation, the solvers $\mathcal{S}$ and $\mathcal{S}$ perform $\mathcal{O}\big((n_u + n_s)\lambda\big)$ group action operations for assigning static and ephemeral keys, and make $\mathcal{O}(n_h + n_s)$ times n-DDH oracle queries for simulating $\mathsf{SessionReveal}$ and the random oracle H queries. This completes the proof of Theorem 4.1. □

It is worth to note here that the above proof seems to work in the strong adversary model where a corrupted party can register any public key of its choice.

o

5 Two-Party Authenticated Key Exchanges from Hard Homogeneous Spaces

We give more realistic two-party AKE protocols than the general n-party CIM-based ones due to its realizability while the number of the user group is restricted to two.

5.1 Hard Homogeneous Spaces (HHS)

As a special case of cryptographic invariant map, we have the notion of hard homogeneous spaces (HHS), which was introduced by Couveignes in [6].

Definition 5.1 ([6, 5]). *A hard homogeneous space consists of a finite commutative group G acting freely and transitively on some set X. The following tasks are required to be easy (e.g., polynomial-time):*

- *Compute the group operations on G.*
- *Sample randomly from G with (close to) uniform distribution.*
- *Decide validity and equality of a representation of elements of X.*
- *Compute the action of a group element $g \in G$ on some $x \in X$.*

By attaching the identity map as a one-variable pairing, i.e., $e_1 = \mathrm{id} : X \to X$, we obtain the 1-variable cryptographic invariant map $pp = (X, X, G, e)$. On the system, we consider 2-way computational (resp., decisional, gap) Diffie–Hellman assumption.

Based on this group action $(g, x) \mapsto g * x$, we have the Diffie–Hellman type key exchange protocol (Fig. 3).

$$\begin{array}{lcl}
\textbf{Alice} & & \textbf{Bob} \\
a \leftarrow_R G : \text{ Alice}'\text{s secret key}, & \xrightarrow{a*x} & b \leftarrow_R G : \text{ Bob}'\text{s secret key}, \\
\text{compute } a * x, & \xleftarrow[b*x]{} & \text{compute } b * x, \\
SK_{\texttt{Alice}} = a * (b * x)\,. & & SK_{\texttt{Bob}} = b * (a * x)\,.
\end{array}$$

Fig. 3. Outline of HHS based DH Protocol.

The notion of HHS is realized by two instantiations: one is called Rostovtsev–Stolbunov system [29, 9] and the other is CSIDH system by Castryck et al. [5]. The former one is obtained from ordinary elliptic curves and their isogenies and the latter from $\mathbb{F}_p$-rational supersingular elliptic curves and their $\mathbb{F}_p$-rational isogenies, respectively. In particular, the CSIDH key exchange is practical and we focus on the CSIDH case in the following as a concrete instantiation.

Example of HHS: CSIDH [5] Castryck et al. [5] proposed a special form of cryptographic invariant map with $n = 1$. Namely, public parameters $pp = (X, S, G, e)$ includes $X = S = \mathcal{E}\ell\ell_p(\mathcal{O})$ which is the set of elliptic curves over $\mathbb{F}_p$ whose $\mathbb{F}_p$-rational endomorphism ring is some fixed quadratic order $\mathcal{O}$, $G = \mathrm{cl}(\mathcal{O})$ which is the ideal class group of $\mathcal{O}$ and e is the identity map $e_1 = \mathrm{id} : X \to X$. On the cryptographic invariant map, we can define 2-way computational (resp. decisional, gap) Diffie–Hellman problem and obtain associated assumptions. Based on the Diffie–Hellman assumption, Castryck et al. obtained a non-interactive key exchange called CSIDH.

Let K be a quadratic number field and $\mathcal{O} \subset K$ an order, that is, a subring which is a free $\mathbb{Z}$-module of rank 2. A fractional ideal of $\mathcal{O}$ is an $\mathcal{O}$-submodule of K of the form $\alpha\mathfrak{a}$, where $\alpha \in K^*$ and $\mathfrak{a}$ is an $\mathcal{O}$-ideal. A fractional $\mathcal{O}$-ideal $\mathfrak{a}$ is invertible if there exists a fractional $\mathcal{O}$-ideal $\mathfrak{b}$ such that $\mathfrak{ab} = \mathcal{O}$. If such a $\mathfrak{b}$ exists, we define $\mathfrak{a}^{-1} = \mathfrak{b}$. The set of invertible fractional ideals $I(\mathcal{O})$ forms an abelian group under ideal multiplication. This group contains the principal ideals $P(\mathcal{O})$ as a subgroup, hence we define the ideal class group of $\mathcal{O}$ as $\mathrm{cl}(\mathcal{O}) = I(\mathcal{O})/P(\mathcal{O})$. Every ideal class $[\mathfrak{a}] \in \mathrm{cl}(\mathcal{O})$ has an integral representative. Any integral ideal $\mathfrak{a}$ of $\mathcal{O}$ splits into a product of $\mathcal{O}$-ideals as $(\pi\mathcal{O})^r\mathfrak{a}_s$, where π is the p-power Frobenius endomorphism and $\mathfrak{a}_s \not\subseteq \pi\mathcal{O}$. This defines an elliptic curve $E/E[\mathfrak{a}]$ and an isogeny $\varphi_\mathfrak{a} : E \to E/E[\mathfrak{a}]$ of degree $N(\mathfrak{a})$: the separable part of $\varphi_\mathfrak{a}$ has kernel $E[\mathfrak{a}] = \cap_{\alpha\in\mathfrak{a}_s} \ker\alpha$, and the purely inseparable part consists of r iterations of Frobenius. The isogeny $\varphi_\mathfrak{a}$ and codomain $E/E[\mathfrak{a}]$ are both defined over $\mathbb{F}_p$ and are unique up to $\mathbb{F}_p$-isomorphism. Since principal ideals correspond to endomorphisms, two ideals lead to the same codomain if and only if they are equal up to multiplication by a principal fractional ideal. Moreover, every $\mathbb{F}_p$-isogeny ψ between curves in $\mathcal{E}\ell\ell_p(\mathcal{O})$ comes from an invertible $\mathcal{O}$-ideal in this way, and the ideal $\mathfrak{a}_s$ can be recovered from ψ as $\mathfrak{a}_s = \{\alpha \in \mathcal{O} \mid \ker\alpha \supseteq \ker\psi\}$. In other words,

Theorem 5.1 ([33, 30, 5]). *Let $\mathcal{O}$ be an order in an imaginary quadratic field. If $\mathcal{E}\ell\ell_p(\mathcal{O})$ is non-empty, then the ideal class group $\mathrm{cl}(\mathcal{O})$ acts on $\mathcal{E}\ell\ell_p(\mathcal{O})$ via*

$$\begin{aligned}\mathrm{cl}(\mathcal{O}) \times \mathcal{E}\ell\ell_p(\mathcal{O}) &\to \mathcal{E}\ell\ell_p(\mathcal{O})\\ ([\mathfrak{a}], E) &\mapsto E/E[\mathfrak{a}],\end{aligned}$$

where $\mathfrak{a}$ is chosen as an integral representative, and this action is free. Furthermore, if $\mathcal{E}\ell\ell_p(\mathcal{O})$ contains a supersingular curve, the action is transitive, else the action has exactly two orbits.

We denote $E/E[\mathfrak{a}]$ by $\mathfrak{a} * E$. Based on this group action, we have the CSIDH key exchange protocol (Fig. 4). For sampling from the class group $\mathrm{cl}(\mathcal{O})$, we follow the manner given in [5]. See Appendix B as well.

In the following, we have one-round two-party AKE protocols from HHS as special cases of multiparty key exchanges in the previous section. In the following, we describe the HHS-based G-CK (resp. G-CK$^+$) secure AKE protocols based on it.

$$\begin{array}{lcl}
\textbf{Alice} & & \textbf{Bob} \\
\mathfrak{a} \leftarrow_R \mathrm{cl}(\mathcal{O}) : \text{ Alice}'\text{s secret key}, & \xrightarrow{\mathfrak{a}*E} & \mathfrak{b} \leftarrow_R \mathrm{cl}(\mathcal{O}) : \text{ Bob}'\text{s secret key}, \\
\text{compute } \mathfrak{a}*E, & \xleftarrow[\mathfrak{b}*E]{} & \text{compute } \mathfrak{b}*E, \\
SK_{\texttt{Alice}} = \mathfrak{a}*(\mathfrak{b}*E)\,. & & SK_{\texttt{Bob}} = \mathfrak{b}*(\mathfrak{a}*E)\,.
\end{array}$$

Fig. 4. Outline of CSIDH Protocol.

5.2 G-CK Secure AKE Protocol (from HHS)

We give our HHS-based UM protocol. Public parameters are $pp = (X, G)$. We set $\Pi =$ HHS-UM, that is, the protocol ID is "HHS-UM." The secret-key space for initiators and responders is given by the group G.

User U_1 has static public key, $T_1 = t_1 * x$, where $t_1 \leftarrow_R G$, and t_1 is U_1's static secret key. User U_2 has static public key, $T_2 = t_2 * x$, where $t_2 \leftarrow_R G$, and t_2 is U_2's static secret key. Here, ephemeral secret keys for U_1 and U_2 are given as $r_1 \leftarrow_R G$, and $r_2 \leftarrow_R G$, respectively. U_1 sends a ephemeral public key R_1 as $R_1 = r_1 * x$ to U_2, U_2 sends back a ephemeral public key R_2 as $R_2 = r_2 * x$ to U_1.

U_1 computes $Z_1 = t_1 * T_2$, and $Z_2 = r_1 * R_2$, and then, obtains the session key SK as $SK = H(\Pi, U_1, U_2, R_1, R_2, Z_1, Z_2)$, where H is a hash function.

U_2 can computes the session key SK as $SK = H(\Pi, U_1, U_2, R_1, R_2, Z_1, Z_2)$ from $Z_1 = t_2 * T_1$, and $Z_2 = r_2 * R_1$ (Fig. 5).

It is clear that the session keys of both parties are equal.

The security of this scheme is given as a corollary of Theorem 3.1.

Corollary 5.1. *Suppose that H is modeled as a quantum random oracle and that the 2-DDH assumption holds on the HHS (X, G). Then the 2-UM protocol is a post-quantum* G-CK*-secure 2-party authenticated key exchange protocol in the quantum random oracle model.*

An instantiation of the HHS-UM protocol by using CSIDH protocol for the HHS is described in Appendix C.

5.3 G-CK$^+$ Secure AKE Protocol (from HHS)

We give our HHS-based biclique protocol. Public parameters are $pp = (X, G)$. We set $\Pi =$ HHS-BC, that is, the protocol ID is "HHS-BC." Static and ephemeral keys are the same as our HHS UM protocol. The secret-key space for initiators and responders is given by the group G.

User U_1 has static public key, $T_1 = t_1 * x$, where $t_1 \leftarrow_R G$, and t_1 is U_1's static key. User U_2, also, has static public key, $B = t_2 * x$, where $t_2 \leftarrow_R G$, and t_2 is U_2's static secret key. Here, ephemeral secret keys for U_1 and U_2 are given as $r_1 \leftarrow_R G$, and $r_2 \leftarrow_R G$, respectively. U_1 sends an ephemeral public key R_1 as $R_1 = r_1 * x$ to U_2, U_2 sends back an ephemeral public key R_2 as $R_2 = r_2 * x$ to U_1.

$$\begin{array}{ccc}
T_1 = t_1 * x & & T_2 = t_2 * x \\
\hline
R_1 = r_1 * x & \xrightarrow{R_1} & R_2 = r_2 * x \\
 & \xleftarrow{R_2} & \\
\hline
Z_1 = t_1 * T_2 & & Z_1 = t_2 * T_1 \\
Z_2 = r_1 * R_2 & & Z_2 = r_2 * R_1 \\
\multicolumn{3}{c}{SK = H(\Pi, U_1, U_2, R_1, R_2, Z_1, Z_2)}
\end{array}$$

Fig. 5. Outline of HHS UM Protocol.

$$\begin{array}{ccc}
T_1 = t_1 * x & & T_2 = t_2 * x \\
\hline
R_1 = r_1 * x & \xrightarrow{R_1} & R_2 = r_2 * x \\
 & \xleftarrow{R_2} & \\
\hline
Z_1 = t_1 * T_2 & & Z_1 = t_2 * T_1 \\
Z_2 = r_1 * T_2 & & Z_2 = t_2 * R_1 \\
Z_3 = t_1 * R_2 & & Z_3 = r_2 * T_1 \\
Z_4 = r_1 * R_2 & & Z_4 = r_2 * R_1 \\
\multicolumn{3}{c}{SK = H(\Pi, U_1, U_2, R_1, R_2, Z_1, Z_2, Z_3, Z_4)}
\end{array}$$

Fig. 6. Outline of HHS Biclique Protocol.

U_1 computes the non-trivial combinations of the ephemeral and static public keys as $Z_1 = t_1 * T_2$, $Z_2 = r_1 * T_2$, $Z_3 = t_1 * R_2$, and $Z_4 = r_1 * R_2$, and then, obtains the session key SK as $SK = H(\Pi, U_1, U_2, R_1, R_2, Z_1, Z_2, Z_3, Z_4)$, where H is a hash function.

U_2 can computes the session key SK as $SK = H(\Pi, U_1, U_2, R_1, R_2, Z_1, Z_2, Z_3, Z_4)$ from $Z_1 = t_2 * T_1$, $Z_2 = t_2 * R_1$, $Z_3 = r_2 * T_1$, and $Z_4 = r_2 * R_1$ (Fig. 6).

It is clear that the session keys of both parties are equal.

The security of this scheme is given as a corollary of Theorem 4.1.

Corollary 5.2. *Suppose that H is modeled as a random oracle and that the* 2-*way GDH assumption holds on the HHS (X, G). Then the biclique* 2-*DH protocol is a post-quantum* G-CK$^+$ *secure authenticated group key exchange protocol in the random oracle model.*

An instantiation of the HHS-BC protocol by using CSIDH protocol for the HHS is described in Appendix C.

References

1. Boneh, D., Glass, D., Krashen, D., Lauter, K., Sharif, S., Silverberg, A., Tibouchi, M., Zhandry, M.: Multiparty non-interactive key exchange and more from isogenies on elliptic curves. In: MATHCRYPT 2018 (2018), https://eprint.iacr.org/2018/665
2. Boyd, C., Nieto, J.M.G.: Round-optimal contributory conference key agreement. In: PKC 2003. pp. 161–174 (2003)
3. Bresson, E., Manulis, M., Schwenk, J.: On security models and compilers for group key exchange protocols. In: IWSEC 2007. pp. 292–307 (2007)
4. Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In: EUROCRYPT 2001. pp. 453–474 (2001)
5. Castryck, W., Lange, T., Martindale, C., Panny, L., Renes, J.: CSIDH: An efficient post-quantum commutative group action. In: ASIACRYPT 2018, Part III. pp. 395–427 (2018)
6. Couveignes, J.M.: Hard homogeneous spaces. IACR Cryptology ePrint Archive **2006**, 291 (2006), http://eprint.iacr.org/2006/291
7. Cremers, C.: Examining indistinguishability-based security models for key exchange protocols: the case of CK, CK-HMQV, and eCK. In: ASIACCS 2011. pp. 80–91 (2011)
8. Cremers, C.J.F.: Session-state Reveal is stronger than Ephemeral Key Reveal: Attacking the NAXOS authenticated key exchange protocol. In: ACNS 2009. pp. 20–33 (2009)

9. Feo, L.D., Kieffer, J., Smith, B.: Towards practical key exchange from ordinary isogeny graphs. In: ASIACRYPT 2018, Part III. pp. 365–394 (2018)
10. Fujioka, A., Suzuki, K., Xagawa, K., Yoneyama, K.: Strongly secure authenticated key exchange from factoring, codes, and lattices. Des. Codes Cryptography **76**(3), 469–504 (2015), a preliminary version appeared in PKC 2012 (2012)
11. Fujioka, A., Takashima, K., Terada, S., Yoneyama, K.: Supersingular isogeny Diffie–Hellman authenticated key exchange. In: ICISC 2018. pp. 1–19 (2019)
12. Galbraith, S.D.: Authenticated key exchange for SIDH. IACR Cryptology ePrint Archive **2018**, 266 (2018), http://eprint.iacr.org/2018/266
13. Galbraith, S.D., Petit, C., Shani, B., Ti, Y.B.: On the security of supersingular isogeny cryptosystems. In: ASIACRYPT 2016, Part I. pp. 63–91 (2016)
14. Galbraith, S.D., Vercauteren, F.: Computational problems in supersingular elliptic curve isogenies. IACR Cryptology ePrint Archive **2017**, 774 (2017), http://eprint.iacr.org/2017/774
15. Gorantla, M.C., Boyd, C., Nieto, J.M.G., Manulis, M.: Generic one round group key exchange in the standard model. In: ICISC 2009. pp. 1–15 (2009)
16. Hövelmanns, K., Kiltz, E., Schäge, S., Unruh, D.: Generic authenticated key exchange in the quantum random oracle model. IACR Cryptology ePrint Archive **2018**, 928 (2018), http://eprint.iacr.org/2018/276
17. Jao, D., et al.: Supersingular isogeny key encapsulation (SIKE). submission to NIST PQC Competition (2017)
18. Jeong, I., Katz, J., Lee, D.: One-round protocols for two-party authenticated key exchange. In: ANCS 2004. pp. 220–232 (2004)
19. Krawczyk, H.: HMQV: A high-performance secure Diffie–Hellman protocol. In: CRYPTO 2005. pp. 546–566 (2005)
20. LaMacchia, B.A., Lauter, K.E., Mityagin, A.: Stronger security of authenticated key exchange. In: ProvSec 2007. pp. 1–16 (2007)
21. Lan, X., Xu, J., Guo, H., Zhang, Z.: One-round cross-domain group key exchange protocol in the standard model. In: Inscrypt 2016. pp. 386–400 (2016)
22. LeGrow, J., Jao, D., Azarderakhsh, R.: Modeling quantum-safe authenticated key establishment, and an isogeny-based protocol. IACR Cryptology ePrint Archive **2018**, 282 (2018), http://eprint.iacr.org/2018/282
23. Li, Y., Yang, Z.: Strongly secure one-round group authenticated key exchange in the standard model. In: CANS 2013. pp. 122–138 (2013)
24. Longa, P.: A note on post-quantum authenticated key exchange from supersingular isogenies. IACR Cryptology ePrint Archive **2018**, 267 (2018), http://eprint.iacr.org/2018/267
25. Manulis, M.: Survey on security requirements and models for group key exchange. IACR Cryptology ePrint Archive **2006**, 388 (2006), http://eprint.iacr.org/2006/388
26. Manulis, M., Suzuki, K., Ustaoglu, B.: Modeling leakage of ephemeral secrets in tripartite/group key exchange. IEICE Transactions **96-A**(1), 101–110 (2013), a preliminary version appeared in ICISC 2009 (2010)
27. National Institute of Standards and Technology: Post-Quantum crypto standardization: Call for Proposals Announcement (December 2016), http://csrc.nist.gov/groups/ST/post-quantum-crypto/cfp-announce-dec2016.html
28. Peikert, C.: He gives c-sieves on the CSIDH. IACR Cryptology ePrint Archive **2019**, 725 (2019), http://eprint.iacr.org/2019/725
29. Rostovtsev, A., Stolbunov, A.: Public–key cryptosystem based on isogenies. IACR Cryptology ePrint Archive **2006**, 145 (2006), http://eprint.iacr.org/2006/145

30. Schoof, R.: Nonsingular plane cubic curves over finite fields. Journal of Combinatorial Theory, Series A **46**(2), 183–208 (1987)
31. Steiner, M., Tsudik, G., Waidner, M.: Key agreement in dynamic peer groups. IEEE Trans. Parallel Distrib. Syst. **11**(8), 769–780 (2000)
32. Suzuki, K., Yoneyama, K.: Exposure-resilient one-round tripartite key exchange without random oracles. In: ACNS 2013. pp. 458–474 (2013)
33. Waterhouse, W.C.: Abelian varieties over finite fields. Annales scientifiques de l'É.N.S., 4^e série **2**(4), 521–560 (1969)
34. Xu, X., Xue, H., Wang, K., Tian, S., Liang, B., Yu, W.: Strongly secure authenticated key exchange from supersingular isogeny. IACR Cryptology ePrint Archive **2018**, 760 (2018), http://eprint.iacr.org/2018/760
35. Zhandry, M.: Secure identity-based encryption in the quantum random oracle model. In: CRYPTO 2012. pp. 758–775 (2012)
36. Zhandry, M.: How to record quantum queries, and applications to quantum indifferentiability. IACR Cryptology ePrint Archive **2018**, 276 (2018), http://eprint.iacr.org/2018/276

Appendix

A Candidates for Cryptographic Invariant Maps [1]

Let E be an ordinary elliptic curve over a finite field $\mathbb{F}_q$ such that the ring $\mathbb{Z}[\pi]$ generated by its Frobenius endomorphism π is integrally closed. Then, $\mathbb{Z}[\pi]$ is the full endomorphism ring $\mathcal{O}$ of E. Denote by $\mathrm{Ab}(E)$ the set of abelian varieties over $\mathbb{F}_q$ that are a product of the form

$$(\mathfrak{a}_1 * E) \times \cdots \times (\mathfrak{a}_n * E) \cong (\mathfrak{a}_1 \cdots \mathfrak{a}_n) * E \times E^{n-1},$$

where $\mathfrak{a}_1, \ldots, \mathfrak{a}_n \in \mathrm{cl}(\mathcal{O})$, and assume that we can efficiently compute an isomorphism invariant for abelian varieties in $\mathrm{Ab}(E)$, i.e., isom: $\mathrm{Ab}(E) \to S$ to some set S that to any tuple $E_1, \ldots, E_n$ of elliptic curves isogenous to E associates an element $\mathrm{isom}(E_1 \times \cdots \times E_n)$ of S such that $\mathrm{isom}(E_1 \times \cdots \times E_n) = \mathrm{isom}(E'_1 \times \cdots \times E'_n)$ if and only if the products $E_1 \times \cdots \times E_n$ and $E'_1 \times \cdots \times E'_n$ are isomorphic as abelian varieties.

Based on such an isomorphism invariant isom, we construct a cryptographic invariant map. The algorithm $\mathsf{MapGen}(1^\lambda)$ computes a sufficiently large base field $\mathbb{F}_q$, and an elliptic curve E over $\mathbb{F}_q$ such that the ring $\mathbb{Z}[\pi]$ generated by its Frobenius endomorphism is integrally closed. The algorithm then outputs the public parameters $pp = (X, S, G, e)$ where $X = \mathcal{E}\ell\ell_q(\mathcal{O})$ is the isogeny class of E over $\mathbb{F}_q$, S is the codomain of the isomorphism invariant isom, $G = \mathrm{cl}(\mathcal{O})$ is the ideal class group of $\mathcal{O}$, and the map $e_n : X^n \to S$ is given by $e_n(E_1, \ldots, E_n) = \mathrm{isom}(E_1 \times \cdots \times E_n)$. It is shown that G acts on X freely and transitively as in Definition 2.4. This approach provides a cryptographic invariant map *assuming isom exists.* In [1], several candidates for isom are demonstrated including the theta null invariants, Igusa invariants, invariants for Kummer surfaces and Deligne invariants, but no invariant maps give an appropriate one in Definition 2.5. Thus, to obtain a suitable invariant is a big open problem remained in [1].

B Description of CSIDH [5]

Parameters, sampling from the class group, and evaluating the group action are given as follows.

Parameters. Fix a large prime p of the form $4 \cdot \ell_1 \cdots \ell_n - 1$, where the ℓ_i are small distinct odd primes. Fix the elliptic curve $E_0 : y^2 = x^3 + x$ over $\mathbb{F}_p$; it is supersingular since $p \equiv 3 \bmod 4$. The Frobenius endomorphism π satisfies $\pi^2 = -p$, so its $\mathbb{F}_p$-rational endomorphism ring is an order in the imaginary quadratic field $\mathbb{Q}(\sqrt{-p})$. More precisely, $\mathcal{O} = \mathrm{End}_p(E_0) = \mathbb{Z}[\pi]$, which has conductor 2.

Rational Elkies primes. The choice made above imply that the ℓ_i-isogeny graph is a disjoint union of cycles. Moreover, since $\pi^2 - 1 \equiv 0 \bmod \ell_i$, the ideals $\ell_i\mathcal{O}$ split as $\ell_i\mathcal{O} = \mathfrak{l}_i\overline{\mathfrak{l}_i}$, where $\mathfrak{l}_i = (\ell_i, \pi - 1)$ and $\overline{\mathfrak{l}_i} = (\ell_i, \pi + 1)$. In other words, all the ℓ_i are Elkies primes. Then, we can use the following algorithm to walk along the cycles: Find a basis of the ℓ_i-torsion and compute the eigenspaces of Frobenius; apply Vélu's formulas to a basis point of the correct eigenspace to compute the codomain.

Furthermore, the kernel of $\phi_{\mathfrak{l}_i}$ is the intersection of the kernels of the scalar multiplication $[\ell_i]$ and the endomorphism $\pi - 1$. That is, it is the subgroup generated by a point P of order ℓ_i which lies in the kernel of $\pi - 1$ or, in other words, is defined over $\mathbb{F}_p$. Similarly, the point generating the kernel of $\phi_{\overline{\mathfrak{l}_i}}$ is of order ℓ_i and defined over $\mathbb{F}_{p^2}$ but not $\mathbb{F}_p$.

Sampling from the class group. Ideally, we would like to know the exact structure of the ideal class group $\mathrm{cl}(\mathcal{O})$ to be able to sample elements uniformly at random. However, such a computation is currently not feasible for the size of discriminant we need, hence we resort to heuristic arguments. Assuming that the $\mathfrak{l}_i$ do not have very small order and are evenly distributed in the class group, we can expect ideals of the form $\mathfrak{l}_1^{e_1}\mathfrak{l}_2^{e_2}\cdots\mathfrak{l}_n^{e_n}$ for small e_i to lie in the same class only very occasionally. For efficiency reasons, it is desirable to sample the exponents e_i from a short range centered around zero, say $\{-m, \ldots, m\}$ for some integer m. Choosing m such that $2m+1 \geq \sqrt[n]{\#\mathrm{cl}(\mathcal{O})}$ is sufficient. Since the prime ideals $\mathfrak{l}_i$ are fixed global parameters, the ideal $\prod_i \mathfrak{l}_i^{e_i}$ may simply be represented as a vector $(e_1, \ldots, e_n)$.

Evaluating the class group action. *Since $\pi^2 = -p \equiv 1 \bmod \ell_i$, we are now in the favorable situation that the eigenvalues of Frobenius on all ℓ_i-torsion subgroups are +1 and -1.* Hence, we can efficiently compute the action of $\mathfrak{l}_i$ (resp. $\overline{\mathfrak{l}_i}$) by finding an $\mathbb{F}_p$-rational (resp. $\mathbb{F}_{p^2} \setminus \mathbb{F}_p$-rational) point of order ℓ_i and apply Vélu-type formulas. This step could simply be repeated for each ideal $\mathfrak{l}_i^{\pm 1}$ whose action is to be evaluated.

$$\begin{array}{ccc}
T_1 = \mathfrak{t}_1 * E & & T_2 = \mathfrak{t}_2 * E \\
\hline
R_1 = \mathfrak{r}_1 * E & \xrightarrow{R_1} & R_2 = \mathfrak{r}_2 * E \\
 & \xleftarrow{R_2} & \\
\hline
Z_1 = \mathfrak{t}_1 * T_2 & & Z_1 = \mathfrak{t}_2 * T_1 \\
Z_2 = \mathfrak{r}_1 * R_2 & & Z_2 = \mathfrak{r}_2 * R_1 \\
\multicolumn{3}{c}{SK = H(\Pi, U_1, U_2, R_1, R_2, Z_1, Z_2)}
\end{array}$$

Fig. 7. Outline of CSIDH UM Protocol.

$$\begin{array}{ccc}
T_1 = \mathfrak{t}_1 * E & & T_2 = \mathfrak{t}_2 * E \\
\hline
R_1 = \mathfrak{r}_1 * E & \xrightarrow{R_1} & R_2 = \mathfrak{r}_2 * E \\
 & \xleftarrow{R_2} & \\
\hline
Z_1 = \mathfrak{t}_1 * T_2 & & Z_1 = \mathfrak{t}_2 * T_1 \\
Z_2 = \mathfrak{r}_1 * T_2 & & Z_2 = \mathfrak{t}_2 * R_1 \\
Z_3 = \mathfrak{t}_1 * R_2 & & Z_3 = \mathfrak{r}_2 * T_1 \\
Z_4 = \mathfrak{r}_1 * R_2 & & Z_4 = \mathfrak{r}_2 * R_1 \\
\multicolumn{3}{c}{SK = H(\Pi, U_1, U_2, R_1, R_2, Z_1, Z_2, Z_3, Z_4)}
\end{array}$$

Fig. 8. Outline of CSIDH Biclique Protocol.

C CSIDH-based G-CK and G-CK$^+$ Secure AKE Protocols

We give our CSIDH-based UM protocol and biclique protocol. Public parameters are $pp = (p, \mathcal{O}, E, \{\ell_i\}, m)$. We set $\Pi =$ CSIDHUM (resp. $\Pi =$ CSIDHBC), that is, the protocol ID is "CSIDHUM" (resp "CSIDHBC"). The secret-key space for initiators and responders is given by the ideal class group $G = \mathrm{cl}(\mathcal{O})$. Static and ephemeral keys are the same for both CSIDH UM and biclique protocols.

User U_1 has static public key, $T_1 = \mathfrak{t}_1 * E$, where $\mathfrak{t}_1 \leftarrow_R G$, and $\mathfrak{t}_1$ is U_1's static secret key. User U_2 has static public key, $T_2 = \mathfrak{t}_2 * E$, where $\mathfrak{t}_2 \leftarrow_R G$, and $\mathfrak{t}_2$ is U_2's static secret key. Here, ephemeral secret keys for U_1 and U_2 are given as $\mathfrak{r}_1 \leftarrow_R G$, and $\mathfrak{r}_2 \leftarrow_R G$, respectively. U_1 sends a ephemeral public key R_1 as $R_1 = \mathfrak{r}_1 * x$ to U_2, U_2 sends back a ephemeral public key R_2 as $R_2 = \mathfrak{r}_2 * x$ to U_1.

Users U_1 and U_2 compute Z_1, Z_2 (resp. $Z_1, \ldots, Z_4$), and then, obtains the session key SK as $SK = H(\Pi, U_1, U_2, R_1, R_2, Z_1, Z_2)$ (resp. $SK = H(\Pi, U_1, U_2, R_1, R_2, Z_1, \ldots, Z_4)$), where H is a hash function as in Fig. 7 (resp. Fig. 8). It is clear that the session keys of both parties are equal. And, the security of the schemes is given in Corollary 5.1 (resp. Corollary 5.2).