Flexible Group Non-interactive Key Exchange in the Standard Model

Li Duan[1,2], Yong Li[2(✉)], and Lijun Liao[2]

[1] Paderborn University, Paderborn, Germany
liduan@mail.upb.de
[2] Huawei Technologies Düsseldorf, Düsseldorf, Germany
{li.duan,yong.li1,lijun.liao}@huawei.com

Abstract. In this paper, we constructed a non-interactive group key exchange protocol (GNIKE) with flexibility, i.e., the number of participants in the GNIKE is not predefined. Moreover, our GNIKE construction is only based on multilinear map and conventional cryptographic building blocks. The security proof of our GNIKE is in the standard model and relies on an n-exponent multilinear DDH assumption.

Keywords: Group non-interactive key exchange · Multilinear maps · Chameleon hash

1 Introduction

1.1 Scalable and Flexible Key Exchange

To communicate with Bob in an untrusted environment, Alice has to agree with Bob on a common secret first. In such a situation, key exchange (KE) protocols can be applied to ensure a shared secret (key) protected against adversaries. Upon the establishment of the shared key, symmetric cryptography can be used to protect the real data (payload) flowing between them.

If we go beyond the 2-party case, one of the major challenges faced by group KE protocol design is to achieve scalability and flexibility simultaneously. Scalability means the shared key should be established within constant rounds of communication [18]. This property is critical for real-world KE protocol, especially in situations where expensive interactions have to be avoided as much as possible. Taking the Internet of Things (IoT) as an example, the energy consumption of the IoT end-devices rockets when sending and receiving messages, but the power supply is usually a battery with limited capacity. Thus communication round reduction can drastically reduce the manufacture and maintenance cost of such devices, generating greater margin for the IoT service provider.

On the other hand, flexibility means the protocol initiator Alice can choose how many partners she wants to share the key. This property becomes essential if the communication network is constructed in an ad-hoc way without predefined

P. Y. A. Ryan and C. Toma (Eds.): SecITC 2021, LNCS 13195, pp. 210–227, 2022.
https://doi.org/10.1007/978-3-031-17510-7_15

sizes. IoT and other mobile devices tend to form temporary communication groups frequently, so a flexible KE can also help support such applications.

Viewing from the pure theoretical aspect, non-interactive authenticated key exchange (NIKE) protocols can be applied to archive good scalability, as no communication round is needed between different participants for key establishment [9]. More specifically, by non-interactive it means that the initiator can first compute the shared group session key offline without contacting others. Once the key is ready to use, the initiator sends out the first message to all her peers, which carries the encrypted application data with associated information for sender identification and group member recognition. After the message arrived, the receiver can then use the associated information to compute the shared group (session) key, decrypt the application data and continue with more communication. However, it is not trivial to design flexible NIKE protocol for groups larger than three and it has become an attractive research topic since 2010.

1.2 Group Non-interactive Key Exchange With and Without iO

Being proposed by Diffie and Hellman [8] in their celebrated work, the nature of NIKE was studied in depth by Freire *et al.* [9] in the 2 party case. Various security models were also formalized in [9]. Boneh *et al.* [5] constructed the first semi-static-secure GNIKE but with predefined number of users from indistinguishability obfuscator (iO) [2]. Hofheinz *et al.* [15] constructed the first adaptively secure GNIKE with and without trusted setup from universal sampler, which is instantiated with iO and other components. The security proof of their trust-setup-free scheme is in the random oracle (RO) model. Rao [29] constructed adaptively secure NIKE among bounded number of parties based on the existence of iO and multilinear map for the cases with and without trusted setup. Khurana *et al.* [19] constructed a NIKE for unbounded parties from iO with trusted setup and non-adaptive security. An overview of the comparable GNIKE works can be found in Table 1. Unlike the existing and provably secure GNIKE protocols mentioned above, our solution and its security directly depends on multilinear map.

Indistinguishability Obfuscation and Multilinear Map. In STOC 2021, it has been shown that iO can be constructed through a long line of bootstrapping from well-founded assumptions (LWE, structured PRG in $\mathbf{NC}^0$, LPN) [17] or from circular security assumptions [14], but most of the existing constructions of iO are in fact based on multilinear map [12,23–25,28].

Besides iO, multilinear map is frequently used to construct other interesting primitives, such as attribute based encryption [13] and revocable identity based encryption [27]. Up to now, there exist a limited number of multi-linear map proposals, such as GGH [11], CLT [7] and their variants [21]. Each proposal depends only on the hardness of one problem. GGH needs learning-with-error problem (LWE) and CLT needs Graded Decisional Diffie-Hellman [7]. Analysis of these multilinear map proposals has also been made, which often exposes weakness in the candidate but also inspires remedies [4,6,16].

As multilinear map usually depends only on one assumption and as fundamental as iO [26], building a GNIKE protocol directly on multilinear map results in lightweight design and simplified security arguments.

1.3 Our Contribution and the Outline of the Paper

Provable security has become a fundamental requirement for all newly proposed schemes or protocols. To formally address the security issues of GNIKE protocols, we propose the definition and the generic security models.

Most importantly, we construct a provably secure and scalable GNIKE protocol with limited trusted setup. This means that only the system wide public parameters (including one public key) is required to be trusted, while the participants can generate their own key pairs conforming with the public parameters.

Moreover, our GNIKE construction is only based on the existence of multilinear maps, besides more conventional cryptographic building blocks such as the chameleon hash. In addition, our main security theorem is proved in standard model relying on an n-exponent multi-linear DDH (nEMDDH) assumption without the use of random oracles.

Outline. The notations and definitions of cryptographic primitives are presented in Sect. 2. The model and definition of GNIKE can be found in Sect. 3. The main construction, the security theorem proof and its proof are provided in Sect. 4. The conclusion, as well as the future works, is presented in Sect. 6. The intractability of our complexity assumption is analyzed in Appendix A.

1.4 Other Related Works

Scalable Interactive Group Key Exchange. In 2003, Katz *et al.* [18] proposed the first scalable interactive group authenticated key exchange protocol, as well as a scalable compiler that can transform other passively secure group KE protocol to a group AKE, adding only one round of communication. The authors also enclosed a survey about the then existing group AKE protocols, focusing on the provable security and efficiency of these protocols. Abdalla *et al.* [1] presented a flexible group AKE protocol, with which the members of the main group can establish session keys for sub-groups interactively.

2 Preliminaries

Notations. We let $\kappa \in \mathbb{N}$ denote the security parameter, and 1^κ the unary string of length κ. Let $[n] = \{1, \ldots, n\} \subset \mathbb{N}$ be the set of integers between 1 and n. If S is a set, $a \xleftarrow{\$} S$ denotes the action of sampling a uniformly random element from S. The term $X||Y$ denotes the operation of concatenating two binary strings X and Y. Let $F(x) \xrightarrow{\$} y$ denote that a probabilistic algorithm $F()$ takes x as input and outputs y. Other notations will be introduced when they first appear.

Table 1. The comparison of GNIKE protocols involving n participants, w.r.t. computational complexity, where iO *prog. gen.* denotes to generate an obfuscated program with the iO and *prog. op.* denotes to call an obfuscated program.

Work	# iO prog. gen.	# prog. op.	# m-map
Here	0	0	n
Rao [29]	1 (n if without trusted setup)	n	n
Khurana [19]	1	n	n
Hofheinz [15]	1 (n if without trusted setup)	n	0

2.1 Chameleon Hash Functions

Chameleon hashes [20] are a class of trapdoor cryptographic hash functions that, without knowledge of the associated trapdoor, are resistant to the inversion and of collision attacks. On the other hand, the collisions can be efficiently found with the trapdoor information.

Definition 1 (Chameleon Hash). *A chameleon hash function* CH *is a tuple of three polynomial time algorithms* $\mathsf{CH} = (\mathsf{CH.GEN}, \mathsf{H}, \mathsf{CF})$.

- $\mathsf{CH.GEN}(1^\kappa) \xrightarrow{\$} (\mathsf{p}, \tau)$. *The non-deterministic key generation algorithm* $\mathsf{CH.GEN}(1^\kappa)$ *on input of a security parameter* 1^κ*, outputs a chameleon hash function key pair* (p, τ)*, where* p *is the public key of chameleon hash and* τ *the trapdoor key.*
- $\mathsf{H}_\mathsf{p}(m, r) \xrightarrow{\$} h$. *Let* $\mathbb{D}_{\mathcal{CH}}$ *be the space of messages,* $\mathbb{R}_{\mathcal{CH}}$ *the space of randomness and* $\mathbb{Z}_{\mathcal{CH}}$ *the space of hash values, all of which are parametrized by* 1^κ *and associated with* (p, τ)*. The polynomial algorithm* $\mathsf{H}_\mathsf{p}(m, r)$*, on input of the public key* p*, a message* $m \in \mathbb{D}_{\mathcal{CH}}$ *and a randomness* $r \in \mathbb{R}_{\mathcal{CH}}$*, computes a hash value* $h \in \mathbb{Z}_{\mathcal{CH}}$.
- $\mathsf{CF}_\tau(m, r) = (m^*, r^*)$. *The collision finding algorithm* $\mathsf{CF}_\tau(h, r)$ *takes as input the trapdoor key* τ*, a message* $m \in \mathbb{D}_{\mathcal{CH}}$ *and a randomness* $r \in \mathbb{R}_{\mathcal{CH}}$ *and it outputs a message* $m^* \in \mathbb{D}_{\mathcal{CH}}$ *and a randomness* $r^* \in \mathbb{R}_{\mathcal{CH}}$ *with* $\mathsf{H}_\mathsf{p}(m, r) = \mathsf{H}_\mathsf{p}(m^*, r^*)$*,* $m \neq m^*$ *and* $r \neq r^*$.

Definition 2 (Collision resistance). CH *is called* $(t_{\mathsf{CH}}, \epsilon_{\mathsf{CH}})$*-chameleon-hash if for all* t_{CH}*-time adversaries* $\mathcal{A}$ *it holds that*

$$\Pr\left[\begin{array}{c}(\mathsf{p}, \tau) \xleftarrow{\$} \mathsf{CH.GEN}(1^\kappa); (m, m^*, r, r^*) \xleftarrow{\$} \mathcal{A}(1^\kappa, \mathsf{H}, \mathsf{p}) : \\ m \neq m^* \bigwedge r \neq r^* \bigwedge \mathsf{H}_\mathsf{p}(m, r) = \mathsf{H}_\mathsf{p}(m^*, r^*)\end{array}\right] \leq \epsilon_{\mathsf{CH}}(\kappa),$$

where $\epsilon_{\mathsf{CH}}(\kappa)$ *is a negligible function in the security parameter* κ*, messages* $m, m^* \in \mathbb{D}_{\mathcal{CH}}$ *and randomness* $r, r^* \in \mathbb{R}_{\mathcal{CH}}$.

2.2 Multilinear Maps

In the following, we briefly recall some of the basic properties of multilinear maps as in [3].

Definition 3 (n-Multilinear Maps). *We state that a map* $\mathsf{nMAP} : \mathbb{G}_1 \times \ldots \times \mathbb{G}_n \rightarrow \mathbb{G}_T$ *is an* nMultilinear *map if it satisfies the following properties:*

1. $\mathbb{G}_1 \ldots \mathbb{G}_n$ *and* $\mathbb{G}_T$ *are groups of the same order.*
2. *if* $x_i \in \mathbb{Z}_p$, $X_i \in \mathbb{G}_i$ *and* $i = 1, \ldots n$, *then* $\mathsf{nMAP}(X_1^{x_1}, \ldots, X_n^{x_n}) = \mathsf{nMAP}(X_1, \ldots, X_n)^{\prod_{i=1}^{x} x_i}$,
3. *if* $g_i \in G_i$ *is a generator of* $\mathbb{G}_i$, *then* $g_T = \mathsf{nMAP}(g_1, \ldots, g_n)$ *is a generator of* $\mathbb{G}_T$, *where* $i = 1, \ldots n$.

We assume the existence of a group description generation algorithm nMGG.Gen, which takes as input a security parameter κ and a positive integer $n \in \mathbb{N}$. The output of $\mathsf{nMGG.Gen}(1^\kappa, n)$ is $\mathsf{MG} = (\mathbb{G}, \mathsf{nMAP})$, which contains a sequence of groups $\mathbb{G} = (\mathbb{G}_1, \ldots, \mathbb{G}_n, \mathbb{G}_T)$ each of a large prime order $p > 2^\kappa$ and a multilinear map nMAP over $\mathbb{G}$. Note that a multilinear map is symmetric when $\mathbb{G}_1 = \ldots = \mathbb{G}_n$ and $g_1 = \ldots = g_n$, otherwise the asymmetric case when different $\mathbb{G}_i$ was considered.

2.3 The n-Exponent Multilinear Decision Diffie-Hellman Assumption

First, let $\mathsf{GP} = (\mathbb{G}_1, g_1, \ldots, \mathbb{G}_n, g_n, \mathbb{G}_T, p, \mathsf{nMAP})$ denote the description of asymmetric multilinear group. For simplicity, we state the complexity assumption needed for our proof of security using symmetric multilinear maps, i.e. $\mathbb{G}_1 = \ldots = \mathbb{G}_n$, and $g_1 = \ldots = g_n$. The n-Exponent multilinear decisional Diffie-Hellman (nEMDDH) problem that is stated as follows: given a tuple $(g, g^a, g^b, R) \in \mathbb{G}^3 \times \mathbb{G}_T$ as input, where $a, b \in \mathbb{Z}_p$ and output yes if $\mathsf{nMAP}(g, \ldots, g)^{a^n b} = R$ and no otherwise.

Definition 4. *We say that the* nEMDDH *problem is* $(t, \epsilon_{\mathsf{nEMDDH}})$*-hard in* GP *if for all adversaries running in time* t, *it holds that*

$$\left| \Pr\left[\mathcal{A}(g, g^a, g^b, \mathsf{nMAP}(g, \ldots, g)^{a^n b}) = 1\right] - \Pr\left[\mathcal{A}(g, g^a, g^b, R) = 1\right] \right| \leq \epsilon_{\mathsf{nEMDDH}},$$

where $(g, g^a, g^b, R) \xleftarrow{\$} \mathbb{G}^3 \times \mathbb{G}_T$.

3 Group Non-interactive Key Exchange and Security Models

3.1 Group Non-interactive Key Exchange

Following Freire, et al. [9], we first present a generic definition of group non-interactive key exchange (GNIKE) in the public key setting. For a GNIKE protocol, each party of a group knows the others' public keys, and without requiring any interaction they can agree on a common shared key. The shared key is generated to be known only by the members of a group.

Let $\mathcal{K}_{\mathcal{GK}}$ be the space of shared keys, $\{\mathcal{PK}, \mathcal{SK}\}$ be key spaces of long-term public keys and private keys respectively, $\mathcal{IDS}$ the identity space of the parties, $\mathcal{K}_{\mathcal{GK}}$ the space of the shared group keys. Those spaces are associated with security parameter κ of the considered protocol. Let $\mathsf{GPK} = \{(ID_t, pk_{\mathsf{ID}_t})\}_n$ be the set of tuples to store the public information of all parties for GNIKE, where n is the size of the group, $t \in [n]$ and pk_{ID_t} the public key of the party with the identity $\mathsf{ID}_t \in \mathcal{IDS}$, and $\mathsf{GPK}_i = \mathsf{GPK} \setminus \{ID_i, pk_{\mathsf{ID}_i}\}$[1]. Each party with ID_t ($t \in [n]$) has a static key pair $(pk_{\mathsf{ID}_t}, sk_{\mathsf{ID}_t}) \in (\mathcal{PK}, \mathcal{SK})$. A general PKI-based GNIKE protocol consists of three polynomial time algorithms (GNIKE.Setup, GNIKE.KGen, GNIKE.SKG) with following semantics:

- $\mathsf{GNIKE.Setup}(1^\kappa) \rightarrow pms$: This algorithm takes as input a security parameter κ and outputs a set of system parameters stored in a variable pms.
- $\mathsf{GNIKE.KGen}(pms, \mathsf{ID}_i) \rightarrow (pk_{\mathsf{ID}_i}, sk_{\mathsf{ID}_i})$: This algorithm takes as input system parameters pms and a party's identity ID_i, and outputs a random key pair $(pk_{\mathsf{ID}_i}, sk_{\mathsf{ID}_i}) \in \{\mathcal{PK}, \mathcal{SK}\}$ for party ID_i.
- $\mathsf{GNIKE.SKG}(sk_{\mathsf{ID}_i}, \mathsf{ID}_i, \mathsf{GPK}_i) \rightarrow K_{\mathsf{GPK}}$: This algorithm take as input (sk_{ID_i}, ID_i) and the group public information $\mathsf{GPK}_i = \{(\mathsf{ID}_t, pk_{\mathsf{ID}_t})\}$, then outputs a shared key $K_{\mathsf{GPK}} \in \mathcal{K}_{\mathcal{GK}}$. Notice that this algorithm GNIKE.SKG is allowed to output a shared key K_{GPK}, even though some participants $\mathsf{ID}_i = \mathsf{ID}_j$, where $\mathsf{ID}_i, \mathsf{ID}_j \in \mathsf{GPK}$.
  For correctness, on input the same group description the algorithm GNIKE.SKG must satisfy the constraint:
  - $\mathsf{GNIKE.SKG}(sk_{\mathsf{ID}_i}, \mathsf{ID}_i, \mathsf{GPK}_i) = \mathsf{GNIKE.SKG}(sk_{\mathsf{ID}_j}, \mathsf{ID}_j, \mathsf{GPK}_j)$, where sk_{ID_i} and sk_{ID_j} are secret keys of parties $\mathsf{ID}_i, \mathsf{ID}_j$.

3.2 Security Models for GNIKE

Freire, et al. [9] formalized a list of security models for NIKE schemes in the 2-party setting, including the honest/dishonest key registration PKI system, denoted here as FHKP models. By generalizing these models into the n-party case ($n \geq 3$), various works have defined the static security [19,31] and the adaptive security [15,29]. It is the main difference between those two security definitions whether the adversary has to commit to a group $\mathcal{S}^*$ to be challenged on before issuing any other queries. Essentially, A protocol is said to be static secure if the adversary has to commit to $\mathcal{S}^*$ and adaptively secure otherwise. We follow the adaptive definition and in our models, the adversary can actively interact with RegisterHonestUID, RegisterCorruptUID, Corrupt, RevealHonestKey, RevealCorruptKey and Test oracles, which we will describe below.

Group Partner Identities. We say that a party P_{ID_i} is a *partner* of another party P_{ID_j}, if they share the same shared key. Notice that P_{ID_i} has multiple partners for GNIKE protocol. Each party can sequentially and concurrently execute the protocol multiple times with its (different) partners. This is modeled by allowing each

[1] If i = 1, then $\mathsf{GPK}_i = \mathsf{GPK}_1 = \{\mathsf{ID}_t, pk_{\mathsf{ID}_t}\}$, $t = \{2, \ldots, n\}$.

principal an unlimited number of instances with which to execute the protocol. We denote group partner identities of any instance s for GNIKE using a variable Gpid. The group partner identities of instance s, denoted here as Gpid^s, stores a set containing the identities of the players that intend to establish a shared key for instance s, where the identities are ordered lexicographically. $n = |\mathsf{Gpid}^s|$ is the number of the identities involved in this instance s. It means that n implies the group-size of key exchange. The Gpid of instance s is set to $\{\mathsf{ID}_1, \ldots, \mathsf{ID}_n\}$.

Adversarial Model. The adversary $\mathcal{A}$ is assumed to have complete control over all communication in the public network. $\mathcal{A}$ in our models is a PPT Turing Machine taking as input the security parameter κ and the public information (e.g. generic description of the environment mentioned above). $\mathcal{A}$'s interaction with the principals is modeled by the following oracles. In GNIKE there is no interaction among the parties, so it means that for modelling a active adversaries against GNIKE no Send query can be considered in the models.

We assume for simplicity a polynomial-size set $\mathsf{PP} = \{\mathsf{ID}_1, \ldots, \mathsf{ID}_i, \ldots, \mathsf{ID}_\ell\}$ of potential honest participants, where ID_i represents the identity of a party, and $i \in [\ell]$. In our models, $\mathcal{A}$ can control the number of the potential participants by the RegisterHonestUID and RemoveUID queries. Each identity ID_i is associated with a static key pair $(pk_{\mathsf{ID}_i}, sk_{\mathsf{ID}_i}) \in (\mathcal{PK}, \mathcal{SK})$. The number of the honest parties ($[\ell]$) is bounded by polynomials in the security parameter κ. Any subset of PP may decide at any point to establish a shared key. Notice that we do not assume that these subsets are always the same size for the flexibility of GNIKE. ID_i is chosen uniquely from the identity space $\mathcal{IDS}$ in the model.

- RegisterHonestUID(ID_i): This query allows $\mathcal{A}$ to register with an identity. $\mathcal{A}$ supplies an identity $\mathsf{ID}_i (i \in [\ell])$ and a static public key pk_{ID_i} is returned to $\mathcal{A}$.
- RegisterCorruptUID($\mathsf{ID}_i, pk_{\mathsf{ID}_i}$): This query allows $\mathcal{A}$ to register a new identity ID_i ($i \notin [\ell]$) and a static public key pk_{ID_i} on behalf of a party ID_i. In response, if the same identity ID_i (with a different public key) already exists for RegisterCorruptUID query, a failure symbol $\perp$ is returned to $\mathcal{A}$. Otherwise, ID_i with the static public key pk_{ID_i} is added, successfully. The parties registered by this query are called corrupted or adversary controlled.
- Corrupt(ID_i): This query allows $\mathcal{A}$ to obtain a secret key of a party with the identity ID_i that was registered as an honest user. The corresponding long-term secret key sk_{ID_i} is returned to $\mathcal{A}$.
- RevealHonestKey(Gpid, s): $\mathcal{A}$ supplies the group partner identities Gpid (in lexicographic order) for protocol instance s, and RevealHonestKey Oracle returns a shared key K^s_{Gpid} to $\mathcal{A}$. For the query the related identities in Gpid must be registered as honest.
- RevealCorruptKey(Gpid, s): This responds with the shared key K^s_{Gpid}. Notice that at least one of the involved identities in Gpid was registered as honest.
- Test(Gpid, s): $\mathcal{A}$ gives Gpid and instance s as inputs to Test Oracle. Test Oracle handles this query as follows: if one of the identities supplied by $\mathcal{A}$ was registered as corrupt or $K^s_{\mathsf{Gpid}} = \emptyset$, it then returns some failure symbol $\perp$. Otherwise it flips a fair coin b, samples a random element K_0 from key space $\mathcal{K}_{\mathcal{GK}}$, sets $K_1 = K^s_{\mathsf{Gpid}}$ to the real shared key, and returns K_b.

Secure GNIKE Protocols. Recall that actual security goals for GNIKE protocols are always specified by individual security games. Here we describe how $\mathcal{A}$ interacts with the security game. We define when an adversary is said to break GNIKE protocols. We firstly state the security game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

Security Game. The security game is played between $\mathcal{C}$ and $\mathcal{A}$, where the following steps are performed:

1. At the beginning of the game, the challenger $\mathcal{C}$ runs algorithm GNIKE.Setup with the security parameter κ as input. $\mathcal{C}$ gives the public parameters to $\mathcal{A}$.
2. Now the adversary $\mathcal{A}$ may start issuing RegisterHonestUID, RegisterCorruptUID, Corrupt, RevealHonestKey and RevealCorruptKey queries The numbers of queries made by $\mathcal{A}$ are bounded by polynomials in security parameter κ.
3. At some point, $\mathcal{A}$ may issue a Test-query during the experiment. Notice that $\mathcal{A}$ can issue an arbitrary number of Test-queries bounded by polynomials in κ for a strong model (GNIKE or mGNIKE.Heavy). In this case, to keep random keys consistent, the $\mathcal{C}$ responds the same random key for the same identities every time $\mathcal{A}$ queries for their shared key, in either order.
4. At the end of the game, $\mathcal{A}$ terminates with outputting a bit b' as its guess for b of Test query.

Definition 5 (Correctness). *We require that for all participants* $\mathsf{ID}_i, \mathsf{ID}_j \in \mathsf{Gpid}$ *for instance* s^* *involved in the same* GNIKE *and such that without a failure symbol* $\perp$*, the same valid shared key is established* $K^{s^*}_{\mathsf{Gpid}} = K^{s^*}_{\mathsf{ID}_i} = K^{s^*}_{\mathsf{ID}_j} \neq \mathsf{null}$.

Definition 6 (Freshness). *For the security definition, we need the notion about the freshness of* Test *oracle which formulates the restrictions on the adversary with respect to performing these above queries. Let* Gpid^{s^*} *denote the group partner identities for instance* s^* *(i.e. (*Gpid, s^**)) queried to* Test *oracle selected by* $\mathcal{A}$*. Then the* Test *oracle is said to be* fresh *if none of the following conditions holds:*

1. *There is a party with identity* $\mathsf{ID}_i \in \mathsf{Gpid}^{s^*}$ *which is established by adversary* $\mathcal{A}$ *via* RegisterCorruptUID *query, i.e.* ID_i *was registered as dishonest.*
2. $\mathcal{A}$ *makes a corrupt query* Corrupt *to any identity* ID_i*, where* $\mathsf{ID}_i \in \mathsf{Gpid}^{s^*}$.
3. $\mathcal{A}$ *either makes a query* $\mathsf{RevealHonestKey}(\mathsf{Gpid}, s^*)$ *for instance* s^**, or a query* $\mathsf{RevealHonestKey}(\mathsf{Gpid}, s)$ *to any* Gpid *of instance* s*, with* $\mathsf{Gpid}^s = \mathsf{Gpid}^{s^*}$.

Security of GNIKE protocols is now defined by requiring that the protocol is a shared key secure non-interactive key-exchange protocol, thus an adversary cannot distinguish the shared key from a random key.

Definition 7 (Non-interactive Shared Key Security). *A group non-interactive key exchange protocol* Σ *is called* (t, ϵ)*-shared-key-secure if for all*

adversaries $\mathcal{A}$ running within probabilistic polynomial time t in the security game as above and for some negligible probability $\epsilon = \epsilon(\kappa)$ in the security parameter κ, it holds that:

- *When $\mathcal{A}$ returns b' and it holds that*
  - *$\mathcal{A}$ has issued a query on* Test *oracle using (*Gpid$, s^*$*) without failure,*
  - *the* Test *oracle for (*Gpid$, s^*$*) is fresh throughout the security experiment*

  then the probability that b' equals the bit b sampled by the Test*-query is bounded by*

$$|\Pr[b = b'] - 1/2| \leq \epsilon$$

4 A Flexible GNIKE Protocol from Multilinear Maps

GNIKE protocols are often carried out in dynamic sets of the participants. One critical feature of GNIKE protocols is to ensure flexibility, i.e., one participant can choose the group members freely. In this section we present a flexible (and salable by definition) construction of group non-interactive key exchange GNIKE, which is provably secure in the standard model without assuming the existence of random oracles or iO. The chameleon hash function is used to bind the user identity and the corresponding initial randomness when generating long-term user key pairs.

4.1 Protocol Description

We describe the protocol in terms of the following three parts: system setup GNIKE.Setup, party-registration and long-term key generation GNIKE.KGen, and group shared key computation GNIKE.SKG.

1. GNIKE.Setup$(1^\kappa, n)$: The proposed protocol is composed of the following building blocks which are instantiated and initialized respectively in accordance of the security parameter 1^κ. Before the main protocol runs for the first time, an upper bound n on the size of the group is set in the initialization phase.
   - generate an nMultilinear map $\mathsf{MG} = (\mathbb{G}, g, \mathbb{G}_T, p, \mathsf{nMAP}) \xleftarrow{\$} \mathsf{GP.Gen}(1^\kappa, n)$, a random element $S \xleftarrow{\$} \mathbb{G}$, and a set of random values $\{u_l\}_{0 \leq l \leq n} \xleftarrow{\$} \mathbb{G}$, where n is the upper bound on the size of the group.
   - fix an identity space $\mathcal{ID} \subset \{0,1\}^*$.
   - parametrize a chameleon hash function $\mathsf{CH} = (\mathsf{CH.GEN}, \mathsf{H}, \mathsf{CF}) : (\mathbb{G} \times \mathcal{ID}) \times \mathbb{R}_{\mathsf{CH}} \to \mathbb{Z}_p^*$, i.e., $\mathbb{D}_{\mathcal{CH}} = (\mathbb{Z}_p^* \times \mathcal{ID})$ and $\mathbb{Z}_{\mathcal{CH}} = \mathbb{Z}_p^*$. $\mathbb{R}_{\mathcal{CH}} \subset \{0,1\}^*$ is a fixed the space of randomness. Let $\mathsf{CHAMKey} = (\mathsf{p}, \tau) \xleftarrow{\$} \mathsf{CH.GEN}(1^\kappa)$ be the pair of public key and trapdoor key.
   - select a random element $\Phi \xleftarrow{\$} \mathbb{G}$, denoted here as *padding* for achieving flexibility.

   The system parameter variable *pms* is $(\mathsf{MG}, \{u_l\}_{0 \leq l \leq n}, S, \Phi, \mathsf{CH}, \mathsf{p})$.

2. $\mathsf{GNIKE.KGen}(pms, \mathsf{ID}_{\hat{A}_i})$: On input the system parameter pms, the key generation algorithm $\mathsf{GNIKE.KGen}$ generates the long-term key pair for a party $\hat{A}_i$ as:
   - choose $a_{\hat{A}_i} \xleftarrow{\$} \mathbb{Z}_p^*, r_{\hat{A}_i} \xleftarrow{\$} \mathbb{R}_{\mathsf{CH}}$,
   - compute $Z_{\hat{A}_i} := g^{a_{\hat{A}_i}}$ and $t_{\hat{A}_i} = \mathsf{H_p}(Z_{\hat{A}_i} || \mathsf{ID}_{\hat{A}_i}, r_{\hat{A}_i})$, and
   - compute $Y_{\hat{A}_i} := (u_0 u_1^{(t_{\hat{A}_i})^1} u_2^{(t_{\hat{A}_i})^2} \dots u_n^{(t_{\hat{A}_i})^n})$ and $X_{\hat{A}_i} := Y_{\hat{A}_i}^{a_{\hat{A}_i}}$.

   The long-term key pair for $\hat{A}_i$: $PK_{\mathsf{ID}_{\hat{A}_i}} = (Z_{\hat{A}_i}, r_{\hat{A}_i}, X_{\hat{A}_i})$ and $SK_{\mathsf{ID}_{\hat{A}_i}} = a_{\hat{A}_i}$.
3. $\mathsf{GNIKE.SKG}(sk_{\mathsf{ID}_{\hat{A}_i}}, ID_{\hat{A}_i}, \mathsf{GPK}_{\hat{A}_i})$: On input a private key $sk_{\mathsf{ID}_{\hat{A}_i}}$ and an identity $ID_{\hat{A}_i}$ of a party $\hat{A}_i$ along with a set of the public parameters $\mathsf{GPK}_{\hat{A}_i}$ [2], algorithm $\mathsf{GNIKE.SKG}$ is executed by each of the parties $\hat{A}_1, \dots, \hat{A}_{n^*}$ as follows:
   - The party $\hat{A}_i$ first checks whether for all identities $\hat{A}_i, \hat{A}_j$ $(i, j \in [n^*], i \neq j)$ it holds that $ID_{\hat{A}_i} \neq ID_{\hat{A}_j}$. The user identity must be unique within each group domain
   - $\hat{A}_i$ computes $t_{\hat{A}_j} = \mathsf{H_p}(Z_{\hat{A}_j} || \mathsf{ID}_{\hat{A}_j}, r_{\hat{A}_j})$ and $Y_{\hat{A}_j} = u_0 u_1^{(t_{\hat{A}_j})^1} \dots u_n^{(t_{\hat{A}_j})^n}$ for all $j \in \{1, \dots, i-1, i+1, \dots, n^*\}$.
   - If $2 \leq n^* \leq n$ and $\forall j \in \{1, \dots, i-1, i+1, \dots, n^*\}$, it holds that

$$\mathsf{nMAP}(Y_{\hat{A}_j}, Z_{\hat{A}_j}, \underbrace{g, \dots, g}_{\text{n-2}}) = \mathsf{nMAP}(X_{\hat{A}_j}, \underbrace{g, \dots, g}_{\text{n-1}}) \tag{1}$$

     - If $n^* = n$, $\hat{A}_i$ computes the shared key as follows:

$$K_{ID_{\hat{A}_{1,\dots,n^*}}} = \mathsf{nMAP}(Z_{\hat{A}_1}, \dots, Z_{\hat{A_i-1}}, Z_{\hat{A}_{i+1}}, \dots, Z_{\hat{A}_n^*}, S)^{sk_{\mathsf{ID}_{\hat{A}_i}}},$$

     - Else, $\hat{A}_i$ adds $(n - n^*)$ Φ padding to the generation function of the shared key (i.e. nMAP)and computes the shared key as follows:

$$K_{ID_{\hat{A}_{1,\dots,n^*}}} = \mathsf{nMAP}(Z_{\hat{A}_1}, \dots, Z_{\hat{A}_{i-1}}, Z_{\hat{A}_{i+1}}, \dots, Z_{\hat{A}_n^*}, \underbrace{\Phi, \dots, \Phi}_{n-n^*}, S)^{sk_{\mathsf{ID}_{\hat{A}_i}}}$$

   - Else $\hat{A}_i$ terminates the protocol execution.

Correctness. In the case when $n^* = n$, for $\mathsf{ID}_{\hat{A}_i}$ we have

$$\begin{aligned} K_{ID_{\hat{A}_{1,\dots,n^*}}} &= \mathsf{nMAP}(Z_{\hat{A}_1}, \dots, Z_{\hat{A}_{i-1}}, Z_{\hat{A}_{i+1}}, \dots, Z_{\hat{A}_n^*}, S)^{sk_{\mathsf{ID}_{\hat{A}_i}}} \\ &= \mathsf{nMAP}(g, \dots, g, S)^{\prod_{i=1}^{n^*} a_{\hat{A}_i}} \end{aligned}$$

[2] $\mathsf{GPK}_{\hat{A}_i}$ is defined in 3.1, i.e. $\mathsf{GPK}_{\hat{A}_i} = (ID_{\hat{A}_1}, pk_{ID_{\hat{A}_1}} \dots, ID_{\hat{A}_{i-1}}, pk_{ID_{\hat{A}_{i-1}}}, ID_{\hat{A}_{i+1}}, pk_{ID_{\hat{A}_{i+1}}}, \dots, ID_{\hat{A}_{n^*}}, pk_{ID_{\hat{A}_{n^*}}})$.

For $\mathsf{ID}_{\hat{A}_j}$ we have

$$\begin{aligned} K'_{ID_{\hat{A}_{1,\ldots,n^*}}} &= \mathsf{nMAP}(Z_{\hat{A}_1}, \ldots, Z_{\hat{A}_{j-1}}, Z_{\hat{A}_{j+1}}, \ldots, Z_{\hat{A}^*_n}, S)^{sk_{\mathsf{ID}_{\hat{A}_j}}} \\ &= \mathsf{nMAP}(g, \ldots, g, S)^{\prod_{j=1}^{n^*} a_{\hat{A}_j}} \end{aligned}$$

By changing the name of variable it can be easily seen that $K_{ID_{\hat{A}_{1,\ldots,n^*}}} = K'_{ID_{\hat{A}_{1,\ldots,n^*}}}$. The correctness argument for the case when $2 \leq n^* < n$ is almost the same.

Rationale of the Construction. Given $(Z_{\hat{A}_i}, r_{\hat{A}_i}), \{u_l\}_{0 \leq l \leq n}$, it is straight forward to compute $Y_{\hat{A}_i}$ with the chameleon hash function CH for the party $\mathsf{ID}_{\hat{A}_i}$. If $X_{\hat{A}_i}$ is consistent with $\mathsf{ID}_{\hat{A}_i}$ and $(Z_{\hat{A}_i}, r_{\hat{A}_i})$, i.e., $X_{\hat{A}_i} = Y_{\hat{A}_i}^{a_{\hat{A}_i}}$, then it should satisfy the nMAP-equation (1) in $\mathsf{GNIKE.SKG}$. The nMAP-equation (1) not only checks the internal consistency of a public key, but also the consistency of the given public key with public parameters and the party identity. The random padding Φ and S provide extra flexibility and they also help eliminate the random oracle in the security analysis.

4.2 Security Analysis

For simplicity, we prove the security of the GNIKE scheme mentioned above in our security $\mathsf{GNIKE.Light}$ model. In the $\mathsf{GNIKE.Light}$ model, $\mathcal{A}$ is allowed to make the following queries: $\mathsf{RegisterHonestUID}$, $\mathsf{RegisterCorruptUID}$, and $\mathsf{RevealCorruptKey}$ to oracles, as well as a single Test query, while $\mathsf{Corrupt}$ and $\mathsf{RevealHonestKey}$ queries are forbidden. To prove our protocol's adaptive security as in $\mathsf{GNIKE.Heavy}$, we will lose a factor of $\binom{n}{n^*}$, where n is the group size bound and n^* the actual group size. Here the loss factor is exponential in the group size n^*. Hence, in order to make the adversarial advantage negligible, one may need to use a larger security parameter or to limit n^*.

Theorem 1. *Suppose that the* nEMDDH *problem is* $(t, \epsilon_{\mathsf{nEMDDH}})$*-hard in* GP *in the sense of Definition 4, and the* CH *is* $(t, \epsilon_{\mathsf{CH}})$*-secure chameleon hash function family. Then the proposed* GNIKE *protocol is* (t', ϵ)*-shared-key-secure in the sense of Definition 7 with* $t' \approx t$ *and* $\epsilon^{\mathsf{GNIKE.Light}}_{\mathcal{A},\mathsf{GNIKE}} \leq \epsilon_{\mathsf{CH}} + \epsilon_{\mathsf{nEMDDH}}$.

5 Proof of Theorem 1

Now, we prove Theorem 1 using the sequence-of-games approach, following [30]. The proof strategy to remove the random oracle is inspired by [10].

Let S_δ be the event that the adversary wins in game G_δ. Let $\mathsf{Adv}_\delta := \Pr[\mathsf{S}_\delta] - 1/2$ denote the advantage of $\mathcal{A}$ in game G_δ.

Game G_0. This is the original game with adversary $\mathcal{A}$ as described in the GNIKE.Light model. Thus we have that

$$\Pr[\mathsf{S}_0] = 1/2 + \epsilon^{\mathsf{GNIKE.Light}}_{\mathcal{A},\mathsf{GNIKE}} = 1/2 + \mathsf{Adv}_0.$$

Game G_1. In this game we want to make sure that there exist no chameleon hash collisions. Technically, we add an abort rule. More precisely, let $\mathsf{abort}_{\mathsf{CH}}$ be the event that the challenger aborts when there exist two distinct identifiers $\hat{\mathsf{H}}$ (e.g. $\mathsf{ID}_{\hat{\mathsf{H}}}$ is registered as Honest) and $\hat{\mathcal{A}}$ (e.g. $\hat{\mathcal{A}}$ is registered as Corrupt), with corresponding public keys $(Z_{\hat{\mathsf{H}}}, Y_{\hat{\mathsf{H}}}, r_{\hat{\mathsf{H}}})$ and $(Z_{\hat{\mathcal{A}}}, Y_{\hat{\mathcal{A}}}, r_{\hat{\mathcal{A}}})$ such that $\mathsf{H}_{\mathsf{p}}(Z_{\hat{\mathsf{H}}}||\mathsf{ID}_{\hat{\mathsf{H}}}, r_{\hat{\mathsf{H}}}) = \mathsf{H}_{\mathsf{p}}(Z_{\hat{\mathcal{A}}}||\mathsf{ID}_{\hat{\mathcal{A}}}, r_{\hat{\mathcal{A}}})$. If $\mathsf{abort}_{\mathsf{CH}}$ did not happen, the challenger continues as in G_0. Obviously the $\Pr[\mathsf{abort}_{\mathsf{CH}}] \leq \epsilon_{\mathsf{CH}}$, according to the security property of underlying chameleon hash function. Until the event $\mathsf{abort}_{\mathsf{CH}}$ happens, $\mathsf{G}_0 = \mathsf{G}_1$. Thus we have

$$|\mathsf{Adv}_0 - \mathsf{Adv}_1| \leq \epsilon_{\mathsf{CH}}.$$

Game G_2. This game proceeds as the previous one, but the challenger always replies to the Test query with a uniformly random element in $\mathbb{G}_T$. Thus the advantage of the adversary in this game is

$$\mathsf{Adv}_2 = 0.$$

Let $\epsilon = |\mathsf{Adv}_1 - \mathsf{Adv}_2|$ and $\epsilon_{\mathsf{nEMDDH}}$ be the advantage of any adversary against the nEMDDH problem (Definition 4). We claim that $\epsilon \leq \epsilon_{\mathsf{nEMDDH}}$. Due to page limit, we leave the complete simulation in Appendix B.

As described in Appendix B, $\mathcal{B}$ sets up all system parameters with the correct distributions and can simulate all queries of $\mathcal{A}$. So if $\mathcal{A}$ can distinguish the real case G_1 from the random case G_2, $\mathcal{B}$ can solve the nEMDDH problem. Therefore we have

$$\mathsf{Adv}_{\mathcal{B}} = \epsilon = |\mathsf{Adv}_1 - \mathsf{Adv}_2|$$

Since nEMDDH assumption holds, we also have $\epsilon = \mathsf{Adv}_{\mathcal{B}} \leq \epsilon_{\mathsf{nEMDDH}}$ and thus

$$|\mathsf{Adv}_1 - \mathsf{Adv}_2| \leq \epsilon_{\mathsf{nEMDDH}}.$$

Collecting the probabilities from Game G_0 to G_2, we have that

$$\epsilon^{\mathsf{GNIKE.Light}}_{\mathcal{A},\mathsf{GNIKE}} \leq \epsilon_{\mathsf{CH}} + \epsilon_{\mathsf{nEMDDH}}.$$

6 Conclusion and Future Works

We constructed a provably secure, flexible and scalable GNIKE protocol. The security is proved under the nEMDDH assumption in standard model. We leave it for future research to design a secure GNIKE protocol with tight security in the standard model.

A Intractability Analysis of n-Exponent Multilinear Diffie-Hellman Assumption

To analyze the intractability of the n-exponent multilinear Diffie-Hellman assumption, we relate it to another problem which is claimed to be hard [10,22]. This nMDDH problem is rephrased below in our notation.

Definition 8. *The n-multilinear decisional Diffie-Hellman (*nMDDH*) problem is* $(t, \epsilon_{\mathsf{nMDDH}})$*-hard in* $\mathsf{GP} = (\mathbb{G}, \mathbb{G}_T, p, \mathsf{nMAP}, g)$ *with multilinear map* nMAP*, if for all adversaries running in probabilistic polynomial time* t*, it holds that*

$$\left|\Pr\left[\mathcal{A}(g, g^a, \mathsf{nMAP}(g, \ldots, g)^{a^{n+1}}) = 1\right] - \Pr\left[\mathcal{A}(g, g^a, R) = 1\right]\right| \leq \epsilon_{\mathsf{nMDDH}},$$

where $(g, a, R) \xleftarrow{\$} \mathbb{G} \times \mathbb{Z}_p \times \mathbb{G}_T$.

With the following lemma, the complexity of nEMDDH can be demonstrated.

Lemma 1. *If the* nMDDH *problem is* $(t, \epsilon_{\mathsf{nMDDH}})$*-hard in* GP*, then the n-Exponent multilinear decisional Diffie-Hellman (*nEMDDH*) problem is* $(t_e, \epsilon_{\mathsf{nEMDDH}})$*-hard in* GP*, where* $t_e \approx t$*,* $\epsilon_{\mathsf{nMDDH}} = \epsilon_{\mathsf{nEMDDH}}$.

Proof. Let $\mathcal{A}_{\mathsf{nEMDDH}}$ be a nEMDDH adversary. We show how to construct another adversary $\mathcal{B}$ against the nMDDH problem instance $(\mathbb{G}, \mathbb{G}_T, p, \mathsf{nMAP}, g, g^a, R)$, with $R = \mathsf{nMAP}(g, g)^{a^{n+1}}$ or $R \xleftarrow{\$} \mathbb{G}_T$.
After receiving its own challenge $\mathcal{B}$ first chooses $c \xleftarrow{\$} \mathbb{Z}_p$ and sets implicitly $b = a \cdot c$ and then computes

$$R' = R^c,\ g^b = g^{ac} = (g^a)^c$$

$\mathcal{B}$ outputs whatever $\mathcal{A}_{\mathsf{nEMDDH}}$ outputs on $(\mathbb{G}, \mathbb{G}_T, p, \mathsf{nMAP}, g, g^a, g^b, R')$.
It is obvious that $\mathcal{B}$ runs in time $t_e \approx t$, if the exponentiation is efficient in GP. Note that since c is uniformly random in $\mathbb{Z}_p$, so is $g^b = (g^a)^c$ in $\mathbb{G}$. Moreover, if $R = \mathsf{nMAP}(g, g)^{a^{n+1}} = \mathsf{nMAP}(g, g)^{a^n a}$, then $R' = R^c = (\mathsf{nMAP}(g, g)^{a^n a})^c = \mathsf{nMAP}(g, g)^{a^n \cdot ac} = \mathsf{nMAP}(g, g)^{a^n \cdot b}$. Otherwise, R' remains uniformly random in $\mathbb{G}_T$. Therefore, $\mathcal{B}$ has generated perfectly an nEMDDH instance for $\mathcal{A}_{\mathsf{nEMDDH}}$ and $\epsilon_{\mathsf{nMDDH}} = \epsilon_{\mathsf{nEMDDH}}$.

B Cases in Game 2 in the Proof of Theorem 1

In game 2, we prove the claim by constructing a nEMDDH adversary $\mathcal{B}$ with advantage ϵ calling $\mathcal{A}$ as a sub-procedure. Let $(g, g^a, g^b, R) \in \mathbb{G}^3 \times \mathbb{G}_T$ be $\mathcal{B}$'s inputs, where $a, b \in \mathbb{Z}_p$. $\mathcal{B}$'s goal is to determine if $\mathsf{nMAP}(g, \ldots, g)^{a^n b} = R$. To set up the GNIKE instance, $\mathcal{B}$ as a challenger runs $\mathsf{GNIKE.Setup}(1^\kappa)$ to generate the system parameters, including a chameleon key pair $\mathsf{CHAMKey} = (\mathsf{p}, \tau)$ and the groups with nMultilinear map MG. It then randomly selects $s_{\hat{A}_1}, \ldots, s_{\hat{A}_n} \xleftarrow{\$} \{0,1\}^*$ and $r_{\hat{A}_i}, \ldots, r_{\hat{A}_i} \xleftarrow{\$} \mathbb{R}_{\mathsf{CH}}$.

Let $p(t) = \prod_{i=0}^{n}(t - t_{\hat{A}_i})^i = \Sigma_{i=0}^{n} p_i t^i = p_0 + p_1 t + \ldots + p_n t^n$ be a polynomial of degree n over $\mathbb{Z}_p$, where $t_{\hat{A}_i} := \mathsf{H_p}(s_{\hat{A}_i}, r_{\hat{A}_i})$. Next, a polynomial of degree n, $q(t) = \Sigma_{i=0}^{n} q_i t^i = q_0 + q_1 t + \ldots + q_n t^n$ is randomly selected over $\mathbb{Z}_p$. Consequently, $\mathcal{B}$ selects random values γ_1, γ_2, uniformly in $\mathbb{Z}_p^*$, sets then $\Phi = (g^a)^{\gamma_1} g^{\gamma_2}$, $u_i = (g^a)^{p_i} g^{q_i}$ for $0 \leq i \leq n$ and $S = g^b$. Since $q_i \xleftarrow{\$} \mathbb{Z}_p^*$, we have $u_i \xleftarrow{\$} \mathbb{G}$. Observably, $u_0 \ldots u_n^{t^n} = (g^a)^{p(t)} g^{q(t)}$. $\mathcal{B}$ then returns the public parameters $(\mathsf{MG}, \{u_i\}_{0 \leq i \leq n}, \Phi, S, \mathsf{p})$ to $\mathcal{A}$ to finish the set up phase. Thereafter, $\mathcal{B}$ answers the queries from $\mathcal{A}$ in the following ways.

- $\mathsf{RegisterHonestUID}(\mathsf{ID}_{\hat{A}_i})$: $\mathcal{A}$ supplies an identity $\mathsf{ID}_{\hat{A}_i}$ ($i \in [n]$) to be registered as honest. To answer this query, B selects $\alpha_i, \beta_i \xleftarrow{\$} \mathbb{Z}_p$, computes $Z_{\hat{A}_i} = (g^a)^{\alpha_i} g^{\beta_i}$. With the trapdoor key τ, $\mathcal{B}$ can extract $r_{\hat{A}_{i,ch}}$, such that

$$\mathsf{H_p}(Z_{\mathsf{ID}_{\hat{A}_i}} || \mathsf{ID}_{\hat{A}_i}, r_{\hat{A}_{i,ch}}) = \mathsf{H_p}(s_{\hat{A}_i}, r_{\hat{A}_i}) = t_{\hat{A}_i}.$$

  $\mathcal{B}$ then computes $X_{\mathsf{ID}_{\hat{A}_i}} = [(g^a)^{p(t_{\hat{A}_i})} g^{q(t_{\hat{A}_i})}]^{a\alpha_i + \beta_i} = [(g^a)^0 g^{q(t_{\hat{A}_i})}]^{a\alpha_i + \beta_i} = (g^{q(t_{\hat{A}_i})})^{a\alpha_i + \beta_i} = (g^a)^{\alpha_i q(t_{\hat{A}_i})} g^{\beta_i q(t_{\hat{A}_i})}$. Finally, $\mathcal{B}$ returns the public key $pk_{\mathsf{ID}_{\hat{A}_i}} = (Z_{\hat{A}_i}, X_{\hat{A}_i}, r_{\hat{A}_{i,ch}})$ to $\mathcal{A}$.
- $\mathsf{RegisterCorruptUID}(\mathsf{ID}_{\mathcal{A}_i}, pk_{\mathsf{ID}_{\mathcal{A}_i}})$: Upon receiving a public key $pk_{\mathsf{ID}_{\mathcal{A}_i}}$ and an identity $\mathsf{ID}_{\mathcal{A}_i}$ from $\mathcal{A}$. The public key $pk_{\mathsf{ID}_{\mathcal{A}_i}}$ is registered as corrupt if $\mathsf{ID}_{\mathcal{A}_i}$ has not been registered before.
- $\mathsf{RevealCorruptKey}$: $\mathcal{A}$ supplies two sets of identities $IDS_{\mathcal{A}}$ and IDS_H: $IDS_{\mathcal{A}} = \{\mathsf{ID}_{\mathcal{A}_1}, \ldots, \mathsf{ID}_{\mathcal{A}_{l_1}}\}$ denoted here as corrupt and $IDS_H = \{\mathsf{ID}_{H_1}, \ldots, \mathsf{ID}_{H_{l_2}}\}$ as honest and $l_1 + l_2 = n^*$.
  - *Case* 1 ($n^* = n$), where n is the upper bound on the size of the group: For this query at least one of the identities supplied by $\mathcal{A}$ was registered as honest ($1 \leq l_2 < n$), and all identities supplied by $\mathcal{A}$ must be unique. $\mathcal{B}$ then checks the public key of any corrupt identity in $IDS_{\mathcal{A}}$ using the multilinear equations to confirm that $pk_{\mathsf{ID}_{\mathcal{A}_i}}$ is of the form $(Z_{\mathsf{ID}_{\mathcal{A}_i}}, a_{\mathsf{ID}_{\mathcal{A}_i}}, r_{\mathsf{ID}_{\mathcal{A}_i}})$, where $Z_{\mathsf{ID}_{\mathcal{A}_i}} = g^{a_{\mathsf{ID}_{\mathcal{A}_i}}}$ and $a_{\mathsf{ID}_{\mathcal{A}_i}} = Y_{\mathsf{ID}_{\mathcal{A}_i}}^{a_{\mathsf{ID}_{\mathcal{A}_i}}} = [(g^a)^{p(t_{\mathsf{ID}_{\mathcal{A}_i}})} g^{q(t_{\mathsf{ID}_{\mathcal{A}_i}})}]^{a_{\mathsf{ID}_{\mathcal{A}_i}}}$. If the check fails, $\mathcal{B}$ rejects this query. $\mathcal{B}$ then computes the corresponding shared key $K_{IDS_{\mathcal{A}}, IDS_H}$ as follows:
    - $t_{\mathsf{ID}_{\mathcal{A}_i}} = \mathsf{H_p}(Z_{\mathsf{ID}_{\mathcal{A}_i}} || \mathsf{ID}_{\mathsf{ID}_{\mathcal{A}_i}}, r_{\mathsf{ID}_{\mathcal{A}_i}})$, where $i \in [l_1]$,
    - computes $p(t_{\mathsf{ID}_{\mathcal{A}_i}})$ and $q(t_{\mathsf{ID}_{\mathcal{A}_i}})$ using the polynomials $p(t)$ and $q(t)$[3],
    - $\{[a_{\mathsf{ID}_{\mathcal{A}_i}}]/[Z_{\mathsf{ID}_{\mathcal{A}_i}}^{q(t_{\mathsf{ID}_{\mathcal{A}_i}})}]\}^{p(t_{\mathsf{ID}_{\mathcal{A}_i}})^{-1}}$
      $= \{[(g^a)^{p(t_{\mathsf{ID}_{\mathcal{A}_i}})} g^{q(t_{\mathsf{ID}_{\mathcal{A}_i}})}]^{a_{\mathsf{ID}_{\mathcal{A}_i}}} / [(g^{a_{\mathsf{ID}_{\mathcal{A}_i}}})^{q(t_{\mathsf{ID}_{\mathcal{A}_i}})}]\}^{p(t_{\mathsf{ID}_{\mathcal{A}_i}})^{-1}} = (g^a)^{a_{\mathsf{ID}_{\mathcal{A}_i}}}$,
    - $Z^*_{\mathsf{ID}_{\mathcal{A}_i}} = [(g^a)^{a_{\mathsf{ID}_{\mathcal{A}_i}}}]^{\alpha_i} (Z_{\mathsf{ID}_{\mathcal{A}_i}})^{\beta_i} = (g^a)^{a_{\mathsf{ID}_{\mathcal{A}_i}} \alpha_i} (g^{a_{\mathsf{ID}_{\mathcal{A}_i}}})^{\beta_i} = (g^{a_{\mathsf{ID}_{\mathcal{A}_i}}})^{(a\alpha_i + \beta_i)} = Z_{\mathsf{ID}_{\mathcal{A}_i}}^{(a\alpha_i + \beta_i)}$

[3] Notice that $p(t_{\mathsf{ID}_{\mathcal{A}_i}}) \neq 0$.

∗ the shared key $K_{IDS_{\mathcal{A}},IDS_H}$[4]:

$$\mathsf{nMAP}(Z_{\mathsf{ID}_{\mathcal{A}_1}},\ldots,Z_{\mathsf{ID}_{\mathcal{A}_{i-1}}},Z^*_{\mathsf{ID}_{\mathcal{A}_i}},$$
$$Z_{\mathsf{ID}_{\mathcal{A}_{i+1}}}\ldots,Z_{\mathsf{ID}_{\mathcal{A}_{l_1}}},Z_{\mathsf{ID}_{H_1}},\ldots,Z_{\mathsf{ID}_{H_{i-1}}},Z_{\mathsf{ID}_{H_{i+1}}},\ldots,Z_{\mathsf{ID}_{H_{l_2}}},S)$$

- *Case* 2 ($2 \le n^* < n$): $\mathcal{B}$ proceeds as the same as *case* 1 mentioned above. For this case $\mathcal{B}$ firstly adds Φ as padding for $(n-n^*)$ times in the following nMAP equation, computes then the corresponding shared key $K_{IDS_{\mathcal{A}},IDS_H}$ as follows:
  ∗ the shared key $K_{IDS_{\mathcal{A}},IDS_H}$[5] is computed as

$$\mathsf{nMAP}(\ldots,Z_{\mathsf{ID}_{\mathcal{A}_{i-1}}},Z^*_{\mathsf{ID}_{\mathcal{A}_i}},Z_{\mathsf{ID}_{\mathcal{A}_{i+1}}},\ldots,Z_{\mathsf{ID}_{\mathcal{A}_{l_1}}},Z_{\mathsf{ID}_{H_1}},\ldots,Z_{\mathsf{ID}_{H_{i-1}}},$$
$$Z_{\mathsf{ID}_{H_{i+1}}},\ldots,Z_{\mathsf{ID}_{H_{l_2}}},\underbrace{\Phi,\ldots,\Phi}_{(n-n^*)},S)$$

– Test: Assume $R = \mathsf{nMAP}(g,\ldots,g)^{a^n b}$. $\mathcal{A}$ supplies n^* identities ($\mathsf{ID}_i, i \in [n^*]$) that were registered as honest.
- *Caes* 1 ($n^* = n$): $\mathcal{B}$ computes

$$\begin{aligned}K_b &= \mathsf{nMAP}(g,\ldots,g)^{(\Pi_{i=1}^{n^*}(a\alpha_i+\beta_i))b}\\ &= \mathsf{nMAP}(g,\ldots,g)^{(\sum_{k=0}^{n^*}\psi_k a^k)b}\\ &= \mathsf{nMAP}(g,\ldots,g)^{(a^n b\Pi\alpha_i)+(\sum_{k=0}^{(n^*-1)}\psi_k a^k)b}\\ &= R^{\Pi\alpha_i}\mathsf{nMAP}(g,\ldots,g)^{\sum_{k=0}^{(n^*-1)}(\psi_k a^k)b},\end{aligned}$$

  and returns K_b to $\mathcal{A}$.
- *Case* 2 ($2 \le n^* < n$): $\mathcal{B}$ computes

$$\begin{aligned}K_b &= \mathsf{nMAP}(g,\ldots,g)^{(\Pi_{i=1}^{n^*}(a\alpha_i+\beta_i))\overbrace{(\Pi_{i=(n^*+1)}^{n}(a\gamma_1+\gamma_2))}^{\Phi\ padding}b}\\ &= \mathsf{nMAP}(g,\ldots,g)^{(\sum_{k=0}^{n}\psi_k a^k)b}\\ &= \mathsf{nMAP}(g,\ldots,g)^{(a^n b\psi_n)+(\sum_{k=0}^{(n-1)}\psi_k a^k)b}\\ &= R^{\psi_n}\mathsf{nMAP}(g,\ldots,g)^{\sum_{k=0}^{(n-1)}(\psi_k a^k)b},\end{aligned}$$

  and returns K_b to $\mathcal{A}$.

[4] If $i = 1$, the shared key is computed as $\mathsf{nMAP}((g^a)^{x_{\mathsf{ID}_{\mathcal{A}_1}}},\ldots,Z_{\mathsf{ID}_{\mathcal{A}_{l_1}}},Z_{\mathsf{ID}_{H_1}},\ldots,Z_{\mathsf{ID}_{H_{i-1}}},Z_{\mathsf{ID}_{H_{i+1}}},\ldots,Z_{\mathsf{ID}_{H_{l_2}}},S)^{\alpha_i}$.

[5] If i = 1, the shared key is computed as $\mathsf{nMAP}((g^a)^{x_{\mathsf{ID}_{\mathcal{A}_1}}},Z_{\mathsf{ID}_{\mathcal{A}_{i+1}}},\ldots,Z_{\mathsf{ID}_{\mathcal{A}_{l_1}}},Z_{\mathsf{ID}_{H_1}},\ldots,Z_{\mathsf{ID}_{H_{i-1}}},Z_{\mathsf{ID}_{H_{i+1}}},\ldots,Z_{\mathsf{ID}_{H_{l_2}}},\underbrace{\Phi,\ldots,\Phi}_{(n-n^*)\Phi},S)^{\alpha_i}$.

- Finally, when $\mathcal{A}$ terminates by outputting a bit b, then $\mathcal{B}$ returns the same bit to nEMDDH challenger.

In each case of the Test query, the right side of the last equality is how $\mathcal{B}$ can compute K_b. According to other equalities, $\mathcal{B}$ can compute the real shared key if $R = \mathsf{nMAP}(g, \ldots, g)^{a^n b}$ holds, with known values of R, α_i, β_i, γ_1 and γ_2. This is exactly as in game G_1. On the other hand, if R is random, $\mathcal{B}$ will output an independent random value in $\mathbb{G}_T$. This is exactly as in game G_2.

References

1. Abdalla, M., Chevalier, C., Manulis, M., Pointcheval, D.: Flexible group key exchange with on-demand computation of subgroup keys. In: Bernstein, D.J., Lange, T. (eds.) AFRICACRYPT 2010. LNCS, vol. 6055, pp. 351–368. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-12678-9_21
2. Barak, B.: On the (Im)possibility of obfuscating programs. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 1–18. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44647-8_1
3. Boneh, D., Silverberg, A.: Applications of multilinear forms to cryptography (2002)
4. Boneh, D., Wu, D.J., Zimmerman, J.: Immunizing multilinear maps against zeroizing attacks. IACR Cryptology ePrint Archive 2014/930 (2014)
5. Dai, Y., Lee, J., Mennink, B., Steinberger, J.: The security of multiple encryption in the ideal cipher model. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8616, pp. 20–38. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44371-2_2
6. Cheon, J.H., Lee, C., Ryu, H.: Cryptanalysis of the new CLT multilinear maps. IACR Cryptology ePrint Archive 2015/934 (2015)
7. Coron, J.-S., Lepoint, T., Tibouchi, M.: Practical multilinear maps over the integers. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 476–493. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40041-4_26
8. Diffie, W., Hellman, M.: New directions in cryptography. IEEE Trans. Inf. Theory **22**(6), 644–654 (1976)
9. Freire, E.S.V., Hofheinz, D., Kiltz, E., Paterson, K.G.: Non-interactive key exchange. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 254–271. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36362-7_17
10. Freire, E.S.V., Hofheinz, D., Paterson, K.G., Striecks, C.: Programmable hash functions in the multilinear setting. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 513–530. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40041-4_28
11. Garg, S., Gentry, C., Halevi, S.: Candidate multilinear maps from ideal lattices. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 1–17. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38348-9_1
12. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. SIAM J. Comput. **45**(3), 882–929 (2016)

13. Garg, S., Gentry, C., Halevi, S., Sahai, A., Waters, B.: Attribute-based encryption for circuits from multilinear maps. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8043, pp. 479–499. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40084-1_27
14. Gay, R., Pass, R.: Indistinguishability obfuscation from circular security. In: Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing, pp. 736–749 (2021). https://doi.org/10.1145/3406325.3451070
15. Hofheinz, D., Jager, T., Khurana, D., Sahai, A., Waters, B., Zhandry, M.: How to generate and use universal samplers. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016. LNCS, vol. 10032, pp. 715–744. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53890-6_24
16. Hu, Y., Jia, H.: Cryptanalysis of GGH map. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9665, pp. 537–565. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49890-3_21
17. Jain, A., Lin, H., Sahai, A.: Indistinguishability obfuscation from well-founded assumptions. In Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing, pp. 60–73 (2021)
18. Katz, J., Yung, M.: Scalable protocols for authenticated group key exchange. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 110–125. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45146-4_7
19. Khurana, D., Rao, V., Sahai, A.: Multi-party key exchange for unbounded parties from indistinguishability obfuscation. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015. LNCS, vol. 9452, pp. 52–75. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48797-6_3
20. Krawczyk, H., Rabin, T.: Chameleon signatures. In: ISOC Network and Distributed System Security Symposium - NDSS 2000. The Internet Society, February (2000)
21. Langlois, A., Stehlé, D., Steinfeld, R.: GGHLite: more efficient multilinear maps from ideal lattices. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 239–256. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-55220-5_14
22. Li, Y., Yang, Z.: Strongly secure one-round group authenticated key exchange in the standard model. In: Abdalla, M., Nita-Rotaru, C., Dahab, R. (eds.) CANS 2013. LNCS, vol. 8257, pp. 122–138. Springer, Cham (2013). https://doi.org/10.1007/978-3-319-02937-5_7
23. Lin, H.: Indistinguishability obfuscation from SXDH on 5-linear maps and locality-5 PRGs. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10401, pp. 599–629. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_20
24. Lin, H., Tessaro, S.: Indistinguishability obfuscation from trilinear maps and block-wise local prgs. Cryptology ePrint Archive, Report 2017/250 (2017). https://doi.org/10.1007/978-3-319-63688-7_21, https://eprint.iacr.org/2017/250
25. Lin, H., Vaikuntanathan, V.: Indistinguishability obfuscation from ddh-like assumptions on constant-degree graded encodings. Cryptology ePrint Archive, Report 2016/795 (2016). https://eprint.iacr.org/2016/795
26. Paneth, O., Sahai, A.: On the equivalence of obfuscation and multilinear maps. IACR Cryptology ePrint Archive 2015/791 (2015)
27. Park, S., Lee, K., Lee, D.H.: New constructions of revocable identity-based encryption from multilinear maps. IEEE Trans. Inf. Forensics Secur. **10**(8), 1564–1577 (2015)

28. Pass, R., Seth, K., Telang, S.: Indistinguishability obfuscation from semantically-secure multilinear encodings. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8616, pp. 500–517. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44371-2_28
29. Rao, V.: Adaptive multiparty non-interactive key exchange without setup in the standard model. IACR Cryptology ePrint Archive 2014/910 (2014)
30. Shoup, V.: Sequences of games: a tool for taming complexity in security proofs (2004)
31. Yamakawa, T., Yamada, S., Hanaoka, G., Kunihiro, N.: Self-bilinear map on unknown order groups from indistinguishability obfuscation and its applications. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8617, pp. 90–107. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44381-1_6