Strongly Secure One-Round Group Authenticated Key Exchange in the Standard Model

Yong Li* and Zheng Yang**

Horst Görtz Institute for IT Security
Ruhr-University Bochum, Germany
{yong.li,zheng.yang}@rub.de

Abstract. One-round group authenticated key exchange (GAKE) protocols typically provide implicit authentication and appealing bandwidth efficiency. As a special case of GAKE – the pairing-based one-round tripartite authenticated key exchange (3AKE), recently gains much attention of research community due to its strong security. Several pairing-based one-round 3AKE protocols have recently been proposed to achieve provable security in the g-eCK model. In contrast to earlier GAKE models, the g-eCK model particularly formulates the security properties regarding resilience to the leakage of various combinations of long-term key and ephemeral session state, and provision of weak perfect forward secrecy in a single model. However, the g-eCK security proofs of previous protocols are only given under the random oracle model. In this work, we give a new construction for pairing-based one-round 3AKE protocol which is provably secure in the g-eCK model without random oracles. Security of proposed protocol is reduced to the hardness of Cube Bilinear Decisional Diffie-Hellman (CBDDH) problem for symmetric pairing. We also extend the proposed 3AKE scheme to a GAKE scheme with more than three group members, based on multilinear maps. We prove g-eCK security of our GAKE scheme in the standard model under the natural multilinear generalization of the CBDDH assumption.

Keywords: one-round, group key exchange, bilinear maps, multilinear maps.

1 Introduction

The situation where three or more parties share a secret key is often called group (conference) keying. A group authenticated key exchange protocol (GAKE) allows a set of parties communicating over public network to create a common shared key that is ensured to be known only to those entities. In a public key infrastructure (PKI) based GAKE protocol, each party typically possesses a pair

* Supported by Secure eMobility grant number 01ME12025.
** Corresponding Author Supported by CSC China. The author names are sorted lexicographically.

M. Abdalla, C. Nita-Rotaru, and R. Dahab (Eds.): CANS 2013, LNCS 8257, pp. 122–138, 2013.

of long-term public/private key. The public key is expected to be certified with a party's identity and corresponding private key is kept secretly for authentication. GAKE protocols are essentially generalized from two party authenticated key exchange (2AKE) protocols to the case of multiple parties. However, this brings new challenges not only in the design but also in the analysis of the GAKE protocols. The formal security model for GAKE was first studied by Bresson et al. [8], where the secrecy (indistinguishability) of the established group key and mutual authentication are modelled following the seminal work of the 2AKE model by Bellare and Rogaway [5]. Since then, figuring out new useful security properties for certain class of GAKE and modelling them become continuing trends.

One-Round GAKE. One import research direction in the research field of GAKE is to construct secure one-round protocol due to its appealing bandwidth-efficiency (in contrast to other multiple-round GAKE). A prominent example is the pairing-based tripartite protocol introduced by Joux [14] which extends the classical two-party Diffie-Hellman KE protocol to the three party case. However Joux's protocol is unauthenticated and subject to well known man-in-the-middle attacks. Hence how to transform Joux's protocol to a secure one-round protocol in presence of active adversaries turns out to be an interesting topic. Several attempts, e.g. [1,17,18,10], have been made to improve the original Joux's protocol. This has also pushed forward the development of security model for GAKE. Meanwhile, the most recently proposed one is the g-eCK model by Fujioka et al. [10]. The g-eCK model basically can be seen as a generalization from the two party eCK model [15]. In contrast to earlier GAKE models, e.g. [8,7,12], the peculiarity of g-eCK model is that it captures lots of desirable security properties regarding resilience to the leakage of various combinations of long-term key and ephemeral session state from target sessions (i.e. the test session and its partner session in the security game), and provision of weak perfect forward secrecy (wPFS) in a single model. So far the g-eCK model is known as one of the strongest security model for one-round GAKE[10]. Therefore proving security for one-round GAKE in the g-eCK model may provide more guarantees.

Motivations. In 2012, Fujioka et al. (FMSU) [10] generalized previous 3AKE protocols into one framework based on admissible polynomials which yields many further one-round 3AKE protocols. The generic FMSU protocol [10] was shown to satisfy g-eCK security. However its security proof is given in the random oracle model (ROM) [4] under a specific strong assumption, i.e. gap Bilinear Diffie-Hellman (GBDH) assumption [2]. It is well-known that the security proof in the random oracle model may not imply that corresponding protocol is secure in the real world. Several results, e.g., [9,3], have demonstrated that there exist schemes which are provably secure in the random oracle model, but are insecure as soon as one replaces the random oracle by any concrete hash functions. This also makes the schemes secure in the standard model to be more appealing than that in the random oracle model. So far we are not aware of previous GAKE protocols being able to achieve g-eCK security in the standard model. Hence, one of the open problems in research on GAKE is to construct a secure scheme in the

g-eCK model under standard assumptions without resorting to random oracles. Another important motivation of this paper is try to simplify the security proof for GAKE protocols under the g-eCK model from the perspective of reducing the freshness ceases that require to prove. Since under the g-eCK model, the freshness cases are related to the group size which are not a small amount. Taking the 3AKE as example, there might be fourteen freshness cases at all that may lead proof to be very tiresome. When the group size is very large, the situation might be worse because the possible freshness cases are exponential in the number of group members. Those facts make us necessary to somehow reduce the upper bound of the freshness cases that require to do proof simulation.

Contributions. We solve the above open problems by starting from 3AKE. We firstly give a concrete construction in Section 5 for one-round 3AKE protocol that is g-eCK secure in the standard model under standard assumptions. The proposed protocol is based on bilinear groups, target collision resistant hash function family, and pseudo-random function family. In order to withstand active attackers, each (either long-term or ephemeral) public key is required to be associated with some kind of 'tag' which is used to verify the consistency of corresponding public key. Those tags are particularly customized using specific weak Programmable Hash Functions (PHF) [13] for ephemeral key and long-term key respectively, whose output lies in a pairing group. Interestingly the proposed protocol is built to be able to run without knowing any priori information about its partners' long-term public key. Intuitively, these *tags* are what give us the necessary leverage to deal with the non-trivial g-eCK security. In order to facilitate the security analysis of 3AKE protocols in the g-eCK model, we introduce propositions to formally reduce fourteen freshness cases (which cover all freshness cases for 3AKE protocols) to four freshness cases. Then it is only necessary to prove the security of considered protocol under the reduced four freshness cases. It is not hard to check the validity of these reductions to all one-round 3AKE protocols in which the message sent by a party is independent of the messages sent by the other parties. Any g-eCK security analyzers for one-round 3AKE protocols might benefit from these results. We then provide a succinct and rigorous game-based security proof by reducing the g-eCK security of proposed 3AKE protocol in the standard model to breaking the cubic Bilinear Decisional Diffie-Hellman (CBDDH) assumption which is slightly modified from the Bilinear Decisional Diffie-Hellman (BDDH) assumption [14].

In the latter we present a GAKE scheme with constant maximum group size in Section 6 following the construction idea of 3AKE. Nevertheless the proposed GAKE scheme is based on the symmetric multilinear map which is first postulated by Boneh and Silverberg [6]. We prove g-eCK security of our scheme in the standard model under a natural multilinear generalization of the CBDDH assumption which is called n-Multiliear Decisional Diffie-Hellman Assumption (nMDDH). In particular we give a general game-based security proof for our proposed GAKE scheme which is given under any polynomial number of freshness cases.

2 Preliminaries

Notations. We let $\kappa \in \mathbb{N}$ denote the security parameter and 1^κ the string that consists of κ ones. Let a capital letter with a 'hat' denote an identity; without the hat the letter denotes the public key of that party. Let $[n] = \{1, \ldots, n\} \subset \mathbb{N}$ be the set of integers between 1 and n. If S is a set, then $a \xleftarrow{\$} S$ denotes the action of sampling a uniformly random element from S. Let '||' denote the operation concatenating two binary strings. In the sequel, we briefly describe the complexity assumptions which lay the foundation of our constructions. Besides we will also make use of target collision resistant hash function family and pseudo-random function family. The corresponding definitions can be found in [16].

Bilinear Groups. In the following, we briefly recall some of the basic properties of bilinear groups. Our AKE solution mainly consists of elements from a single group $\mathbb{G}$. We therefore concentrate on symmetric bilinear maps. Our pairing based scheme will be parameterized by a symmetric pairing parameter generator, denoted by $\mathsf{PG.Gen}$. This is a polynomial time algorithm that on input a security parameter 1^κ, returns the description of two multiplicative cyclic groups $\mathbb{G}$ and $\mathbb{G}_T$ of the same prime order p, generator g for $\mathbb{G}$, and a bilinear computable pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$.

Definition 1 (Symmetric Bilinear groups). *We call* $\mathcal{PG} = (\mathbb{G}, g, \mathbb{G}_T, p, e) \xleftarrow{\$} \mathsf{PG.Gen}(1^\kappa)$ *be a set of symmetric bilinear groups, if the function e is an (admissible) bilinear map and it holds that:*

1. ***Bilinear:*** $\forall (a, b) \in \mathbb{G}$ *and* $\forall (x, y) \in \mathbb{Z}_p$*, we have* $e(a^x, b^y) = e(a, b)^{xy}$.
2. ***Non-degenerate:*** $e(g, g) \neq 1_{\mathbb{G}_T}$*, is a generator of group* $\mathbb{G}_T$.
3. ***Efficiency:*** $\forall (a, b) \in \mathbb{G}$*, e is efficiently computable.*

Multilinear Groups. In the following, we recall the definition of symmetric multilinear groups introduced in [6]. We assume that a party can call a group generator $\mathsf{MLG.Gen}(1^\kappa, n)$ to obtain a set of multilinear groups. On input a security parameter κ and a positive integer $2 < n \in \mathbb{N}$, the polynomial time group generator $\mathsf{MLG.Gen}(1^\kappa, n)$ outputs two multiplicative cyclic groups $\mathbb{G}$ and $\mathbb{G}_T$ of the same prime order p, generator g for $\mathbb{G}$, and a n-multilinear map $me : \mathbb{G}^n \times \mathbb{G} \to \mathbb{G}_T$.

We summarize the properties of n-multilinear groups in the following definition.

Definition 2 (Symmetric Multilinear groups). *We call* $\mathcal{MLG} = (\mathbb{G}, \mathbb{G}_T, p, me) \xleftarrow{\$} \mathsf{MLG.Gen}(\kappa, n)$ *be a set of symmetric multilinear groups, if the n-multilinear map me holds that:*

1. ***n-multilinear:*** $\forall (c_1, \ldots, c_n) \in \mathbb{G}$ *and* $\forall (y_1, \ldots, y_n) \in \mathbb{Z}_p$*, we have* $me(c_1^{y_1}, \ldots, c_n^{y_n}) = me(c_1, \ldots, c_n)^{y_1 \cdots y_n}$.
2. ***Non-degenerate:*** $me(g, \ldots, g) \neq 1_{\mathbb{G}_T}$*, is a generator of group* $\mathbb{G}_T$.
3. ***Efficiency:*** $\forall (c_1, \ldots, c_n) \in \mathbb{G}$*, the operation* $me(c_1, \ldots, c_n)$ *is efficiently computable.*

Concrete multilinear maps can be found in [11] by Garg, Gentry, and Halvei. We here just focus on a general definition of symmetric n-multilinear groups without loss of generality.

Cube Bilinear Decisional Diffie-Hellman Assumption. With respect to our construction for one-round tripartite AKE, we need a new complexity assumption defined as follows.

Definition 3. *We say that the* CBDDH *problem relative to generator* PG.Gen *is* $(t, \epsilon_{\mathsf{CBDDH}})$*-hard, if the probability bound* $|\Pr[\mathsf{EXP}^{cbddh}_{\mathsf{PG.Gen},\mathcal{A}}(\kappa, n) = 1] - 1/2| \leq \epsilon_{\mathsf{CBDDH}}$ *holds for all adversaries* $\mathcal{A}$ *running in probabilistic polynomial time* t *in the following experiment:*

$\mathsf{EXP}^{cbddh}_{\mathsf{PG.Gen},\mathcal{A}}(\kappa, n)$
$\mathcal{PG} = (\mathbb{G}, g, \mathbb{G}_T, p, e) \xleftarrow{\$} \mathsf{PG.Gen}(1^\kappa)$;
$a, \gamma \xleftarrow{\$} \mathbb{Z}_p^*$;
$b \xleftarrow{\$} \{0,1\}$, *if* $b = 1$ $\Gamma \leftarrow e(g,g)^{a^3}$, *otherwise* $\Gamma \leftarrow e(g,g)^{\gamma}$;
$b' \leftarrow \mathcal{A}(1^\kappa, \mathcal{PG}, g^a, \Gamma)$;
if $b = b'$ *then return 1, otherwise return 0;*

where $\epsilon_{\mathsf{CBDDH}} = \epsilon_{\mathsf{CBDDH}}(\kappa)$ *is a negligible function in* κ.

The proof for the security of CBDDH assumption in the generic group model [19] is presented in the full version of this paper [16].

n-Multilinear Decisional Diffie-Hellman Assumption. We present a generalization of the CBDDH assumption in n-multilinear groups that we call the n-Multilinear Decisional Diffie-Hellman (nMDDH) assumption.

Definition 4. *We say that the* nMDDH *problem relative to generator* MLG.Gen *is* $(t, \epsilon_{\mathsf{nMDDH}})$*-hard, if the probability bound* $|\Pr[\mathsf{EXP}^{nmddh}_{\mathsf{PG.Gen},\mathcal{A}}(\kappa, n) = 1] - 1/2| \leq \epsilon_{\mathsf{nMDDH}}$ *holds for all adversaries* $\mathcal{A}$ *running in probabilistic polynomial time* t *in the following experiment:*

$\mathsf{EXP}^{nmddh}_{\mathsf{PG.Gen},\mathcal{A}}(\kappa)$
$\mathcal{MLG} = (\mathbb{G}, \mathbb{G}_T, g, p, me) \xleftarrow{\$} \mathsf{MLG.Gen}(\kappa, n)$;
$a, \gamma \xleftarrow{\$} \mathbb{Z}_p^*$, $b \xleftarrow{\$} \{0,1\}$;
$\Gamma \leftarrow me(g, \ldots, g)^{a^{n+1}}$ *if* $b = 1$, *otherwise* $\Gamma \leftarrow me(g, \ldots, g)^{\gamma}$;
$b' \leftarrow \mathcal{A}(1^\kappa, \mathcal{MLG}, g^a, \Gamma)$;
if $b = b'$ *then return 1, otherwise return 0;*

where $\epsilon_{\mathsf{nMDDH}} = \epsilon_{\mathsf{nMDDH}}(\kappa)$ *is a negligible function in* κ.

3 Security Model for Group Authenticated Key Exchange

In this section we present the formal security model for PKI-based group authenticated key-exchange (GAKE) protocols. In this model, while emulating the

real-world capabilities of an active adversary, we provide an 'execution environment' for adversaries following an important line of research [15,18,10] which is initiated by Bellare and Rogaway [5]. We formalize the capabilities of an adversary in a strong sense who is provided enormous power to take full control over the communication network (e.g., alter or inject messages as she wishes), in particular she may compromise long-term keys of parties or secret states of protocol instances at any time. Let $\mathcal{K}_{\mathsf{AKE}}$ be the key space of session key, and $\{\mathcal{PK}, \mathcal{SK}\}$ be key spaces for long-term public/private key respectively. Those spaces are associated with security parameter κ of considered protocol.

Execution Environment. In the execution environment, we fix a set of honest parties $\{\mathsf{ID}_1, \ldots, \mathsf{ID}_\ell\}$ for $\ell \in \mathbb{N}$, where ID is identity of a party which is chosen uniquely from space $\mathcal{IDS}$. Each identity is associated with a long-term key pair $(sk_{\mathsf{ID}_i}, pk_{\mathsf{ID}_i}) \in (\mathcal{SK}, \mathcal{PK})$ for entity authentication, and is indexed via integer $i \in [\ell]$ in the model. Note that those identities are also lexicographically indexed via variable $i \in [\ell]$. For public key registration, each party ID_i might be required to provide extra information (denoted by proof) to prove either the knowledge of the secret key or correctness of registered public key (via e.g. non-interactive proof of knowledge schemes). Each honest party ID_i can sequentially and concurrently execute the protocol multiple times with different indented partners, this is characterized by a collection of oracles $\{\pi_i^s : i \in [\ell], s \in [\rho]\}$ for $\rho \in \mathbb{N}$. Oracle π_i^s behaves as party ID_i carrying out a process to execute the s-th protocol instance, which has access to the long-term key pair $(sk_{\mathsf{ID}_i}, pk_{\mathsf{ID}_i})$ of ID_i and to all other public keys. Moreover, we assume each oracle π_i^s maintains a list of independent internal state variables with following semantics: (i) pid_i^s – storing a set of partner identities in the group with whom π_i^s intends to establish a session key (including ID_i itself), where the identities are ordered lexicographically; (ii) Φ_i^s – storing the oracle decision $\Phi_i^s \in \{\mathtt{accept}, \mathtt{reject}\}$; (iii) K_i^s – recording the session key $K_i^s \in \mathcal{K}_{\mathsf{KE}}$ for symmetric encryption; (iv) st_i^s – storing the maximum secret session states that are allowed to be leaked (e.g., the exponent of exchanged ephemeral public key); (v) T_i^s – storing the transcript of all messages sent and received by π_i^s during its execution, where the messages are ordered by round and within each round lexicographically by the identities of the purported senders.

All those variables of each oracle are initialized with empty string denoted by symbol $\emptyset$ in the following. At some point, each oracle π_i^s may complete the execution always with a decision state Φ_i^s. Furthermore, we assume that the session key is assigned to the variable K_i^s (such that $K_i^s \neq \emptyset$) iff oracle π_i^s has reached an internal state $\Phi_i^s = \mathtt{accept}$.

Adversarial Model. An adversary $\mathcal{A}$ in our model is a PPT Turing Machine taking as input the security parameter 1^κ and the public information (e.g. generic description of above environment), which may interact with these oracles by issuing the following queries.

- $\mathsf{Send}(\pi_i^s, m)$: The adversary can use this query to send any message m of his own choice to oracle π_i^s. The oracle will respond the next message m^* (if any) to be sent according to the protocol specification and its internal states. Oracle π_i^s would be initiated via sending the oracle the first message $m = (\top, \mathsf{pid}_i^s)$ consisting of a special initialization symbol $\top$ and a variable storing partner identities. After answering a Send query, the variables $(\mathsf{pid}_i^s, \Phi_i^s, K_i^s, st_i^s, T_i^s)$ might be updated depending on the specific protocol.
- $\mathsf{RevealKey}(\pi_i^s)$: Oracle π_i^s responds with the contents of variable K_i^s.
- $\mathsf{StateReveal}(\pi_i^s)$: Oracle π_i^s responds with the secret state stored in variable st_i^s, e.g. the random coins used to generate the session key.
- $\mathsf{Corrupt}(\mathsf{ID}_i)$: Oracle π_i^1 responds with the long-term secret key sk_{ID_i} of party ID_i if $i \in [\ell]$. After this query, oracles $\pi_i^s (s > 1)$ can still answer other queries.
- $\mathsf{RegisterCorrupt}(\mathsf{ID}_\tau, pk_{\mathsf{ID}_\tau}, \mathsf{proof}_{\mathsf{ID}_\tau})$: This query allows the adversary to register an identity ID_τ ($\ell < \tau$ and $\tau \in \mathbb{N}$) and a static public key pk_{ID_τ} on behalf of a party ID_τ, if ID_τ is unique and pk_{ID_τ} is ensured to be sound by evaluating the non-interactive proof $\mathsf{proof}_{\mathsf{ID}_\tau}$. We only require that the proof is non-interactive in order to keep the model simple. Parties established by this query are called dishonest.
- $\mathsf{Test}(\pi_i^s)$: This query may only be asked once throughout the experiment. Oracle π_i^s handles this query as follows: If the oracle has state $\Omega = \texttt{reject}$ or $K_i^s = \emptyset$, then it returns some failure symbol $\perp$. Otherwise it flips a fair coin b, samples a random element K_0 from key space $\mathcal{K}_{\mathsf{KE}}$, sets $K_1 = K_i^s$ to the real session key, and returns K_b.

We stress that the exact meaning of the $\mathsf{StateReveal}$ must be defined by each protocol separately, and each protocol should be proven secure to resist with such kind of state leakage as claimed. Namely a protocol should specify the content stored in the variable *st* during protocol execution. In order to protect those critical session states of AKE protocols, utilizing secure (e.g. tamper-proof) device might be a natural solution, namely at each party an untrusted host machine is used together with a secure hardware. In this way it is possible to adopt a 'All-and-Nothing' strategy to define the session states — namely we can assume that *all states* stored on untrusted host machine can be revealed via $\mathsf{StateReveal}$ query and *no state* would be exposed at secure device without loss of generality. The $\mathsf{RegisterCorrupt}$ query is used to model the chosen identity and public key attacks. In this query, the detail form of proof_τ (i.e. how to register an identity and corresponding public key) should be specified by each protocol. Please note that if the protocol allows for arbitrary key registration then one could set the parameter $\mathsf{proof} = \emptyset$. Basically, our execution environment is consistent to the g-eCK model [10] except for the $\mathsf{RegisterCorrupt}$ query. In the original g-eCK model, the adversary is allowed to register a public key (via AddUser query) by checking whether corresponding register key comes from the key space for public key. However in our model, we model the requirement of the key registration in a more general way via parameter proof.

Secure AKE Protocols. To formalize the notion that two oracles are engaged in an on-line communication, we define the partnership via *matching sessions*. We assume that messages in a transcript T_i^s are represented as binary strings.

Definition 5. *We say that an oracle π_i^s has a* matching session *to oracle π_j^t, if $\mathsf{pid}_i^s = \mathsf{pid}_j^t$ and π_i^s has sent all protocol messages and $T_i^s = T_j^t$.*

Definition 6 (Correctness). *Let π_i^s and π_j^t be two oracles. We say a GAKE protocol Σ is* correct*, if both oracles π_i^s and π_j^t accept such that π_i^s and π_j^t have matching sessions, then it holds that $K_i^s = K_j^t$.*

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

1. At the beginning of the game, the challenger $\mathcal{C}$ implements the collection of oracles $\{\pi_i^s : i \in [\ell], s \in [\rho]\}$, and generates ℓ long-term key pairs $(pk_{\mathsf{ID}_i}, sk_{\mathsf{ID}_i})$ and corresponding proof proof_i for all honest parties ID_i where the identity $\mathsf{ID}_i \in \mathcal{IDS}$ of each party is chosen uniquely. $\mathcal{C}$ gives adversary $\mathcal{A}$ $\{(\mathsf{ID}_1, pk_{\mathsf{ID}_1}, \mathsf{proof}_{\mathsf{ID}_1}), \ldots, (\mathsf{ID}_\ell, pk_{\mathsf{ID}_\ell}, \mathsf{proof}_{\mathsf{ID}_\ell})\}$ as input.
2. $\mathcal{A}$ may issue polynomial number of queries: Send, StateReveal, Corrupt, RegisterCorrupt and RevealKey.
3. At some point, $\mathcal{A}$ may issue a $\mathsf{Test}(\pi_i^s)$ query on an oracle π_i^s during the experiment but only once.
4. At the end of game, the $\mathcal{A}$ may terminate with outputting a bit b' as its guess for b of Test query.

For the security definition, we need the notion about the freshness of oracles which formulates the restrictions on the adversary with respect to performing these above queries.

Definition 7 (Freshness). *Let π_i^s be an accepted oracle.*
Let $\pi_S = \{\pi_j^t\}_{\mathsf{ID}_j \in \mathsf{pid}_i^s, j \neq i}$ be a set of oracles (if they exist), such that π_i^s has a matching session to π_j^t. Then the oracle π_i^s is said to be fresh if none of the following conditions holds:
(i) $\mathcal{A}$ queried $\mathsf{RegisterCorrupt}(\mathsf{ID}_j, pk_{\mathsf{ID}_j}, \mathsf{proof}_{\mathsf{ID}_j})$ with some $\mathsf{ID}_j \in \mathsf{pid}_i^s$; (ii) $\mathcal{A}$ queried either $\mathsf{RevealKey}(\pi_i^s)$ or $\mathsf{RevealKey}(\pi_j^t)$ for some oracle $\pi_j^t \in \pi_S$; (iii) $\mathcal{A}$ queried both $\mathsf{Corrupt}(\mathsf{ID}_i)$ and $\mathsf{StateReveal}(\pi_i^s)$; (iv) For some oracle $\pi_j^t \in \pi_S$, $\mathcal{A}$ queried both $\mathsf{Corrupt}(\mathsf{ID}_j)$ and $\mathsf{StateReveal}(\pi_j^t)$; (v) If $\mathsf{ID}_j \in \mathsf{pid}_i^s$ $(j \neq i)$ and there is no oracle π_j^t such that π_i^s has a matching session to π_j^t, $\mathcal{A}$ queried $\mathsf{Corrupt}(\mathsf{ID}_j)$.

Definition 8 (g-eCK Security). *We say that an adversary $\mathcal{A}$ (t, ϵ)-breaks the g-eCK security of a correct group AKE protocol Σ, if $\mathcal{A}$ runs the AKE security game within time t, and the following condition holds:*

- *If a* Test *query has been issued to a fresh oracle π_i^s, then the probability that the bit b' returned by $\mathcal{A}$ equals to the bit b chosen by the* Test *query is bounded by*

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

We say that a correct group AKE protocol Σ is (t, ϵ)-g-eCK-secure, if there exists no adversary that (t, ϵ)-breaks the g-eCK security of Σ.

4 Simplify the Security Proof for One-Round GAKE in the g-eCK Model

We first present a generic definition of one-round group authenticated key exchange (ORGAKE) to allow us to describe our generic result for this class of protocols. In a ORGAKE protocol, each party may send a single 'message' and this message is always assumed to be independent of the message sent by the other party without loss of generality. The independence property of sent messages is required since the session participants can't achieve mutual authentication in one-round and it enables parties to run protocol instances simultaneously (which is a key feature of one-round protocol). The key exchange procedure is done within two pass and a common shared session key is generated to be known only by session participants.

Let $\mathsf{GD} := ((\mathsf{ID}_1, pk_{\mathsf{ID}_1}), \ldots, (\mathsf{ID}_n, pk_{\mathsf{ID}_n}))$ be a list which is used to store the public information of a group of parties formed as tuple $(\mathsf{ID}_i, pk_{\mathsf{ID}_i})$, where n is the size of the group members which intend to share a key and pk_{ID_i} is the public key of party $\mathsf{ID}_i \in \mathcal{IDS}$ $(i \in [n])$. Let T denote the transcript storing the messages sent and received by a protocol instance at a party which are sorted orderly. A general PKI-based ORGAKE protocol may consist of four polynomial time algorithms $(\mathsf{ORGAKE.Setup}, \mathsf{ORGAKE.KGen}, \mathsf{ORGAKE.MF}, \mathsf{ORGAKE.SKG})$ with following semantics:

- $pms \leftarrow \mathsf{Setup}(1^\kappa)$: This algorithm takes as input a security parameter κ and outputs a set of system parameters storing in a variable pms.
- $(sk_{\mathsf{ID}}, pk_{\mathsf{ID}}, \mathsf{proof}_{\mathsf{ID}}) \xleftarrow{\$} \mathsf{ORGAKE.KGen}(pms, \mathsf{ID})$: This algorithm takes as input system parameters pms and a party's identity ID, and outputs a pair of long-term private/public key $(sk_{\mathsf{ID}}, pk_{\mathsf{ID}}) \in (\mathcal{PK}, \mathcal{SK})$ for party ID and a non-interactive proof for pk_{ID} (which is required during key registration.).
- $m_{\mathsf{ID}_1} \xleftarrow{\$} \mathsf{ORGAKE.MF}(pms, sk_{\mathsf{ID}_1}, r_{\mathsf{ID}_1}, \mathsf{GD})$: This algorithm takes as input system parameters pms and the sender ID_1's secret key sk_{ID_1}, a randomness $r_{\mathsf{ID}_1} \xleftarrow{\$} \mathcal{R}_{\mathsf{ORGAKE}}$ and the group information variable GD, and outputs a message to be sent in a protocol pass, where $\mathcal{R}_{\mathsf{ORGAKE}}$ is the randomness space.[1]
- $K \leftarrow \mathsf{ORGAKE.SKG}(pms, sk_{\mathsf{ID}_1}, r_{\mathsf{ID}_1}, \mathsf{GD}, \mathsf{T})$: This algorithm take as the input system parameters pms and ID_1's secret key sk_{ID_1}, a randomness $r_{\mathsf{ID}_1} \xleftarrow{\$} \mathcal{R}_{\mathsf{ORGAKE}}$ and the group information GD and a transcript T orderly recorded all protocol messages exchanged[2], and outputs session key $K \in \mathcal{K}_{\mathsf{ORGAKE}}$.

[1] We remark that the parameter GD of algorithm ORGAKE.MF is only optional, which can be any empty string if specific protocol compute the message without knowing any information about its indented partners.

[2] The detail order needs to be specified by each protocol.

For correctness, we require that, on input the same group description $\mathsf{GD} = ((\mathsf{ID}_1, pk_1), \ldots, (\mathsf{ID}_n, pk_n))$ and transcript T, algorithm $\mathsf{ORGAKE.SKG}$ satisfies the constraint: $\mathsf{ORGAKE.SKG}(pms, sk_{\mathsf{ID}_1}, r_{\mathsf{ID}_1}, \mathsf{GD}, \mathsf{T}) = \mathsf{ORGAKE.SKG}(pms, sk_{\mathsf{ID}_i}, r_{\mathsf{ID}_i}, \mathsf{GD}, \mathsf{T})$, where sk_{ID_i} is the secret key of a party $\mathsf{ID}_i \in \mathsf{GD}$ who generates randomness $r_{\mathsf{ID}_i} \in \mathcal{R}_{\mathsf{ORGAKE}}$ for $i \in [n]$.

Besides these algorithms, each protocol might consist of other steps such as long-term key registration and message exchange, which should be described by each protocol independently.

Simplify the Security Proof for One-round Tripartite AKE in the g-eCK model. We show how to reduce the complexity of the security proof of any one-round 3AKE protocol with the above form in the g-eCK model. To prove the security of a protocol in the g-eCK model, it is necessary to show the proof under all possible freshness cases formulated by Definition 7. Let oracle $\pi_{\hat{A}}^{s^*}$ be the test oracle with intended partner $\hat{B}$ and $\hat{C}$ for instance. If any adversary breaks the indistinguishability security property of am OR3AKE protocol, then at least one of the following fresh events must occur:

- **Event 0**: There are oracles $\pi_{\hat{B}}^{t^*}$ and $\pi_{\hat{C}}^{l^*}$, such that $\pi_{\hat{A}}^{s^*}$ has matching session to $\pi_{\hat{B}}^{t^*}$ and to $\pi_{\hat{C}}^{l^*}$ respectively.
- **Event 1**: There is an oracle $\pi_{\hat{F}}^{t^*}$ such that $\pi_{\hat{A}}^{s^*}$ and $\pi_{\hat{F}}^{t^*}$ have matching sessions but there is no oracle of $\hat{D}$ having matching session to $\pi_{\hat{A}}^{s^*}$, where $\hat{F}$ and $\hat{D}$ are parties such that $\hat{F}, \hat{D} \in \{\hat{B}, \hat{C}\}$ and $\hat{D} \neq \hat{F}$.
- **Event 2**: $\pi_{\hat{A}}^{s^*}$ has no matching session.

In the Table 1, we show the freshness cases regarding to $\mathsf{StateReveal}$ and $\mathsf{Corrupt}$ query which might be occurred in each event. Let 'nRS' denote the situation that the adversary did not issue $\mathsf{StateReveal}$ query to specific oracle, and 'nC' denote the situation adversary did not issue $\mathsf{Corrupt}$ query to corresponding party (e.g. the owner of certain oracle).

Table 1. Freshness Cases in Each Event

Event 0	$\pi_{\hat{A}}^{s^*}$	$\pi_{\hat{B}}^{t^*}$	$\pi_{\hat{C}}^{l^*}$
Case 1 (C1)	nRS	nRS	nRS
Case 2 (C2)	nC	nRS	nRS
Case 3 (C3)	nRS	nRS	nC
Case 4 (C4)	nC	nRS	nC
Case 5 (C5)	nRS	nC	nRS
Case 6 (C6)	nC	nC	nRS
Case 7 (C7)	nC	nC	nC
Case 8 (C8)	nRS	nC	nC

Event 1	$\pi_{\hat{A}}^{s^*}$	$\pi_{\hat{F}}^{t^*}$	$\hat{D}$
Case 9 (C9)	nRS	nRS	nC
Case 10 (C10)	nC	nRS	nC
Case 11 (C11)	nC	nC	nC
Case 12 (C12)	nRS	nC	nC

Event 2	$\pi_{\hat{A}}^{s^*}$	$\hat{B}$	$\hat{C}$
Case 13 (C13)	nC	nC	nC
Case 14 (C14)	nRS	nC	nC

In order to complete the proof, we must provide the security proofs under all fourteen cases that might be tiresome. However we introduce the following

general propositions to facilitate the proof of any OR3AKE protocols in the form of the above description. Our goal is to reduce the freshness cases which have the similar restrictions on adversary's queries.

Proposition 1. *If adversary* $\mathcal{A}_1$ $(t_1, \epsilon_{\mathcal{A}_1})$*-breaks the g-eCK security of a OR3AKE protocol* Σ *in case* $C2$*, then there exists an adversary* $\mathcal{A}_2$ *who can* $(t_2, \epsilon_{\mathcal{A}_2})$*-breaks the g-eCK security of* Σ *in case* $C5$*, such that* $t_1 \approx t_2$ *and* $\epsilon_{\mathcal{A}_1} = \epsilon_{\mathcal{A}_2}$*.*

Proposition 2. *If adversary* $\mathcal{A}_1$ $(t_1, \epsilon_{\mathcal{A}_1})$*-breaks the g-eCK security of a OR3AKE protocol* Σ *in case* $C3$ *(*$C5$*), then there exists an adversary* $\mathcal{A}_2$ *who can* $(t_2, \epsilon_{\mathcal{A}_2})$*-breaks the g-eCK security of* Σ *in case* $C9$*, such that* $t_1 \approx t_2$ *and* $\epsilon_{\mathcal{A}_1} = \epsilon_{\mathcal{A}_2}$*.*

Proposition 3. *If adversary* $\mathcal{A}_1$ $(t_1, \epsilon_{\mathcal{A}_1})$*-breaks the g-eCK security of a OR3AKE protocol* Σ *in case* $C7$*, then there exists an adversary* $\mathcal{A}_2$ *who can* $(t_2, \epsilon_{\mathcal{A}_2})$*-breaks the g-eCK security of* Σ *in case* $C11$*. If such adversary* $\mathcal{A}_2$ *exists, then there exists an adversary* $\mathcal{A}_3$ *who can* $(t_3, \epsilon_{\mathcal{A}_3})$*-breaks the g-eCK security of* Σ *in case* $C13$*. We have that* $t_1 \approx t_2 \approx t_3$ *and* $\epsilon_{\mathcal{A}_1} = \epsilon_{\mathcal{A}_2} = \epsilon_{\mathcal{A}_3}$*.*

Proposition 4. *If adversary* $\mathcal{A}_1$ $(t_1, \epsilon_{\mathcal{A}_1})$*-breaks the g-eCK security of a OR3AKE protocol* Σ *in case* $C4$*, then there exists an adversary* $\mathcal{A}_2$ *who can* $(t_2, \epsilon_{\mathcal{A}_2})$*-breaks the g-eCK security of* Σ *in case* $C10$*. If such adversary* $\mathcal{A}_2$ *exists, then there exists an adversary* $\mathcal{A}_3$ *who can* $(t_3, \epsilon_{\mathcal{A}_3})$*-breaks the g-eCK security of* Σ *in case* $C12$*. If such adversary* $\mathcal{A}_3$ *exists, then there exists adversary* $\mathcal{A}_4$ *who can* $(t_4, \epsilon_{\mathcal{A}_4})$*-breaks the g-eCK security of* Σ *in case* $C14$*. We have that* $t_1 \approx t_2 \approx t_3 \approx t_4$ *and* $\epsilon_{\mathcal{A}_1} = \epsilon_{\mathcal{A}_2} = \epsilon_{\mathcal{A}_3} = \epsilon_{\mathcal{A}_4}$*.*

Proposition 5. *If adversary* $\mathcal{A}_1$ $(t_1, \epsilon_{\mathcal{A}_1})$*-breaks the g-eCK security of a OR3AKE protocol* Σ *in case* $C6$*, then there exists an adversary* $\mathcal{A}_2$ *who can* $(t_2, \epsilon_{\mathcal{A}_2})$*-breaks the g-eCK security of* Σ *in case* $C8$*. If such adversary* $\mathcal{A}_2$ *exists, then there exists an adversary* $\mathcal{A}_3$ *who can* $(t_3, \epsilon_{\mathcal{A}_3})$*-breaks the g-eCK security of* Σ *in case* $C12$*. We have that* $t_1 \approx t_2 \approx t_3$ *and* $\epsilon_{\mathcal{A}_1} = \epsilon_{\mathcal{A}_2} = \epsilon_{\mathcal{A}_3}$*.*

The proofs of above propositions can be found in the full version of this paper [16]. Due to the above reductions, one could prove the security of any one-round 3AKE protocol in the g-eCK model only under freshness cases $C1$, $C9$, $C13$ and $C14$. This would be dramatically simplify the security proof. In the sequel, we call these freshness cases require to write proof as target freshness cease.

Towards Lower Bound of Target Freshness Cases for the Proof of One-round GAKE with Arbitrary Group Size in the g-eCK Model. In order to make the proof for one-round GAKE protocol in the g-eCK model to be more tight, we might also need to do the analogous reductions about the freshness cases as it is done for OR3AKE. So that we make a conjecture for the lower bound of target freshness cases for the proof of AKE protocol with arbitrary group size n in the g-eCK Model.

Conjecture 1. For any one-round group AKE protocol with members $n + 1$, we have $n + 2$ freshness cases that require proof simulations.

The proof idea of this conjecture is presented in [16].

5 A Tripartite AKE Protocol from Bilinear Maps

In this section we present a three party one-round AKE protocol based on symmetric bilinear groups, a target collision resistant hash function and a pseudo-random function family. The requirements for underlying building blocks are standard, the proposed protocol provides g-eCK security without random oracles.

5.1 Protocol Description

Setup: The proposed protocol takes as input the following building blocks which are initialized respectively in terms of the security parameter $\kappa \in \mathbb{N}$: (i) Symmetric bilinear groups $\mathcal{PG} = (\mathbb{G}, g, \mathbb{G}_T, p, e) \xleftarrow{\$} \mathsf{PG.Gen}(1^\kappa)$ and a set of random values $\{u_i\}_{0\leq i\leq 3} \xleftarrow{\$} \mathbb{G}$; (ii) a target collision resistant hash function $\mathsf{TCRHF}(hk_{\mathsf{TCRHF}}, \cdot) : \mathcal{K}_{\mathsf{TCRHF}} \times \mathbb{G} \to \mathbb{Z}_p$, where $\mathcal{K}_{\mathsf{TCRHF}}$ is the key space of TCRHF and $hk_{\mathsf{TCRHF}} \xleftarrow{\$} \mathsf{TCRHF.KG}(1^\kappa)$; and (iii) a pseudo-random function family $\mathsf{PRF}(\cdot,\cdot) : \mathbb{G}_T \times \{0,1\}^* \to \mathcal{K}_{\mathsf{AKE}}$. The system parameters encompass $pms := (\mathcal{PG}, \{u_i\}_{0\leq i\leq 3}, hk_{\mathsf{TCRHF}})$.

$\hat{A}$	$\hat{B}$	$\hat{C}$
$x \xleftarrow{\$} \mathbb{Z}_p^*, X := g^x$	$y \xleftarrow{\$} \mathbb{Z}_p^*, Y := g^y$	$z \xleftarrow{\$} \mathbb{Z}_p^*, Z := g^z$
$h_X := \mathsf{TCRHF}(X)$	$h_Y := \mathsf{TCRHF}(Y)$	$h_Z := \mathsf{TCRHF}(Z)$
$t_X := (u_0 u_1^{h_X} u_2^{h_X^2} u_3^{h_X^3})^x$	$t_Y := (u_0 u_1^{h_Y} u_2^{h_Y^2} u_3^{h_Y^3})^y$	$t_Z := (u_0 u_1^{h_Z} u_2^{h_Z^2} u_3^{h_Z^3})^z$
broadcast $(\hat{A}, A, t_A, X, t_X)$	*broadcast* $(\hat{B}, B, t_B, Y, t_Y)$	*broadcast* $(\hat{C}, C, t_C, Z, t_Z)$
$h_B := \mathsf{TCRHF}(B)$	$h_A := \mathsf{TCRHF}(A)$	$h_A := \mathsf{TCRHF}(A)$
$h_C := \mathsf{TCRHF}(C)$	$h_C := \mathsf{TCRHF}(C)$	$h_B := \mathsf{TCRHF}(B)$
$h_Y := \mathsf{TCRHF}(Y)$	$h_X := \mathsf{TCRHF}(X)$	$h_X := \mathsf{TCRHF}(X)$
$h_Z := \mathsf{TCRHF}(Z)$	$h_Z := \mathsf{TCRHF}(Z)$	$h_Y := \mathsf{TCRHF}(Y)$
$U_B := u_0 u_1^{h_B} u_2^{h_B^2} u_3^{h_B^3}$	$U_A := u_0 u_1^{h_A} u_2^{h_A^2} u_3^{h_A^3}$	$U_A := u_0 u_1^{h_A} u_2^{h_A^2} u_3^{h_A^3}$
$U_C := u_0 u_1^{h_C} u_2^{h_C^2} u_3^{h_C^3}$	$U_C := u_0 u_1^{h_C} u_2^{h_C^2} u_3^{h_C^3}$	$U_B := u_0 u_1^{h_B} u_2^{h_B^2} u_3^{h_B^3}$
$U_Y := u_0 u_1^{h_Y} u_2^{h_Y^2} u_3^{h_Y^3}$	$U_X := u_0 u_1^{h_X} u_2^{h_X^2} u_3^{h_X^3}$	$U_X := u_0 u_1^{h_X} u_2^{h_X^2} u_3^{h_X^3}$
$U_Z := u_0 u_1^{h_Z} u_2^{h_Z^2} u_3^{h_Z^3}$	$U_Z := u_0 u_1^{h_Z} u_2^{h_Z^2} u_3^{h_Z^3}$	$U_Y := u_0 u_1^{h_Y} u_2^{h_Y^2} u_3^{h_Y^3}$
reject if either	reject if either	reject if either
$e(t_B, g) \neq e(U_B, B)$ *or*	$e(t_A, g) \neq e(U_A, A)$ *or*	$e(t_A, g) \neq e(U_A, A)$ *or*
$e(t_C, g) \neq e(U_C, C)$ *or*	$e(t_C, g) \neq e(U_C, C)$ *or*	$e(t_B, g) \neq e(U_B, B)$ *or*
$e(t_Y, g) \neq e(U_Y, Y)$ *or*	$e(t_X, g) \neq e(U_X, X)$ *or*	$e(t_X, g) \neq e(U_X, X)$ *or*
$e(t_Z, g) \neq e(U_Z, Z)$	$e(t_Z, g) \neq e(U_Z, Z)$	$e(t_Y, g) \neq e(U_Y, Y)$
Each party has $\mathsf{sid} := \hat{A}\|\|A\|\|t_A\|\|X\|\|t_X\|\|\hat{B}\|\|B\|\|t_B\|\|Y\|\|t_Y\|\|\hat{C}\|\|C\|\|t_C\|\|Z\|\|t_Z$		
Each party rejects if some values recorded in sid are identical		
$k := e(BY, CZ)^{a+x}$	$k := e(AX, CZ)^{b+y}$	$k := e(AX, BY)^{c+z}$
$k_e := \mathsf{PRF}(k, sid)$	$k_e := \mathsf{PRF}(k, sid)$	$k_e := \mathsf{PRF}(k, sid)$

Fig. 1. One-round Tripartite AKE Protocol

Long-term Key Generation and Registration: On input pms, a party $\hat{A}$ may run an efficient algorithm $(sk_{\hat{A}}, pk_{\hat{A}}, \emptyset) \xleftarrow{\$} \mathsf{ORGAKE.KGen}(pms, \hat{A})$ to generate the long-term key pair as: $sk_{\hat{A}} = a \xleftarrow{\$} \mathbb{Z}_p^*, pk_{\hat{A}} = (A, t_A)$ where $A = g^a$, $t_A := (u_0 u_1^{h_A} u_2^{h_A^2} u_3^{h_A^3})^a$ and $h_A = \mathsf{TCRHF}(A)$. Please note that we allow arbitrary key registration, i.e. the adversary is able to query $\mathsf{RegisterCorrupt}(\hat{A}, pk_{\hat{A}}, \emptyset)$ with $\mathsf{proof}_{\hat{A}} = \emptyset$.

Protocol Execution: On input pms, the protocol among parties $\hat{A}$, $\hat{B}$ and $\hat{C}$ is depicted in the Figure 1.

Implementation and Session States: We assume that the maximum states of party $\hat{A}$ allowing for leakage consist of ephemeral private key x (resp. y and z for parties $\hat{B}$ and $\hat{C}$) – namely those values would be stored in the state variable st of each oracle at any time. For example this can be guaranteed by performing the computations for k and k_e on secure device. Note that the all pairing operations including $e(BY, CZ)$ can be done on host machine.

We notice that a party $\hat{A}$ has to do consistency check on long-term key in every sessions that might be wasteful. An alternative solution could make the Certificate Authority to check the consistency of long-term public key during key registration procedure. In this way, it might reduce two pairing operations for protocol execution and also the number of public key. To register a public key $pk_{\hat{A}} = A$, each party $\hat{A}$ should at least prove the consistency via tag t_A. Then the public key A is registered if $e(t_A, g) = e(A, u_0 u_1^{h_A} u_2^{h_A^2} u_3^{h_A^3})$. Thus this check would be done only once at CA. The downside of this approach is that it might increase the burden of CA. In particular, the tag t_A is required while querying the $\mathsf{RegisterCorrupt}(\hat{A}, pk_{\hat{A}}, \mathsf{proof}_{\hat{A}})$ in the security game, i.e. $\mathsf{proof}_{\hat{A}} = t_A$.

5.2 Security Analysis

We show the security of proposed protocol in the g-eCK model.

Theorem 1. *Assume each ephemeral key chosen during key exchange has bit-size $\lambda \in \mathbb{N}$. Suppose that the* CBDDH *problem is $(t, \epsilon_{\mathsf{CBDDH}})$-hard in the symmetric bilinear groups $\mathcal{PG}$, the* TCRHF *is $(t, \epsilon_{\mathsf{TCRHF}})$-secure target collision resistant hash function family, and the* PRF *is $(q, t, \epsilon_{\mathsf{PRF}})$-secure pseudo-random function family. Then the proposed protocol is (t', ϵ)-session-key-secure in the sense of Definition 8 with $t' \approx t$, $q \geq 3$ and $\epsilon \leq \frac{(\rho\ell)^2}{2^\lambda} + \epsilon_{\mathsf{TCRHF}} + 4(\rho\ell)^3 \cdot (\epsilon_{\mathsf{CBDDH}} + \epsilon_{\mathsf{PRF}})$.*

Proof sketch. We give a brief sketch of the proof of Theorem 1 (more details can be found in [16]). It is straightforward to see that two oracles accept with matching sessions would compute the same session key. Namely the proposed protocol is correct. In the sequel, we wish to show that the adversary is unable to distinguish random value from the session key of any fresh oracle.

To complete the proof of Theorem 1, we only need to prove the advantage of the adversary is negligible under target freshness cases $C1$, $C9$, $C13$ and $C14$, due to the reductions in Section 4. The proof proceeds in a sequence of

games, following [20]. Let S_δ be the event that the adversary wins the security experiment in Game G_δ under freshness cases in $\{C1, C9, C13, C14\}$. 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 $\mathcal{A}$. The system parameters are chosen honestly by challenger as protocol specification. Thus we have that $\Pr[\mathsf{S}_0] = 1/2 + \epsilon = 1/2 + \mathsf{Adv}_0$.

Game G_1. In this game, the challenger aborts, if during the simulation an ephemeral key replied by an oracle π_i^s but it has been sample by another oracle or sent by adversary before. We have that $\mathsf{Adv}_0 \leq \mathsf{Adv}_1 + \frac{(\rho\ell)^2}{2^\lambda}$.

Game G_2. In this game, the challenger aborts if two oracles output the same hash value of TCRHF. Thus we have $\mathsf{Adv}_1 \leq \mathsf{Adv}_2 + \epsilon_{\mathsf{TCRHF}}$.

Game G_3. This game proceeds as previous game, but $\mathcal{C}$ aborts if one of the following guesses fails: (i) the freshness case occurred to test oracle from the set $\{C1, C9, C13, C14\}$, (ii) the test oracle, (iii) its partner parties, and (iv) corresponding oracles (if any) each of which has a matching session to test oracle, in terms of specific guessed freshness case. Since there are four considered fresh cases, ℓ parties and at most ρ oracles for each party, then the probability that all above guesses of $\mathcal{C}$ are correct is at least $1/4(\rho\ell)^3$. Thus we have that $\mathsf{Adv}_2 \leq 4(\rho\ell)^3 \cdot \mathsf{Adv}_3$. Please note that there are at least three uncompromised (either long-term and ephemeral) Diffie-Hellman keys which are used by test oracle to generate its key material k^*, as otherwise the test oracle is not g-eCK-fresh any more. We call such guessed three uncompromised DH keys as *target DH keys*.

Game G_4. Technically, this game is proceeded as previous game, but the challenger $\mathcal{C}$ replaces the key material k_i^s with random value $\widetilde{k_i^s}$ for oracles $\{\pi_i^s : i \in [\ell], s \in [\rho]\}$ which satisfy the following conditions: (i) The k_i^s is computed involving the three *target DH keys*, and (ii) Those *target DH keys* used by π_i^s are from three distinct parties. If there exists an adversary $\mathcal{A}$ can distinguish the Game G_4 from Game G_3 then we can make use of it to solve the CBDDH problem. We therefore obtain that $\mathsf{Adv}_3 \leq \mathsf{Adv}_4 + \epsilon_{\mathsf{CBDDH}}$.

Game G_5. In this game, we change function $\mathsf{PRF}(\widetilde{k^*}, \cdot)$ to a truly random function for test oracle and its partner oracles (if they exist). Thus we have that $\mathsf{Adv}_4 \leq \mathsf{Adv}_5 + \epsilon_{\mathsf{PRF}}$ due to the security of PRF. Note that in this game the session key returned by Test-query is totally a truly random value which is independent to the bit b and any messages. Thus the advantage that the adversary wins this game is $\mathsf{Adv}_5 = 0$. Sum up the probabilities from Game G_0 to Game G_5, we proved this theorem.

6 A GAKE Construction from Multilinear Maps

An interesting work is to extend the proposed 3AKE scheme to GAKE scheme with more than three group members. Based on bilinear groups might be impossible to achieve so. Since we can not get an aggregate long-term shared key for a group of members from bilinear map. However, Boneh and Silverberg [6]

have given us inspiration on how to generalize the 3AKE to GAKE by exploiting multilinear maps.

6.1 Protocol Description

Setup: The proposed GAKE protocol takes as input the following building blocks which are initialized respectively in terms of the security parameter $\kappa \in \mathbb{N}$ and upper-bound of number of users $n+1$: (i) n-mulitilinear groups $\mathcal{MLG} = (\mathbb{G}, \mathbb{G}_T, g, p, me) \xleftarrow{\$} \mathsf{MLG.Gen}(\kappa, n)$ and a set of random values $\{u_j\}_{0\leq j\leq n+1} \xleftarrow{\$} \mathbb{G}$; (ii) a target collision resistant hash function $\mathsf{TCRHF}(hk_{\mathsf{TCRHF}}, \cdot) : \mathcal{K}_{\mathsf{TCRHF}} \times \mathbb{G} \to \mathbb{Z}_p$, where $hk_{\mathsf{TCRHF}} \xleftarrow{\$}$ $\mathsf{TCRHF.KG}(1^\kappa)$; and (iii) a pseudo-random function family $\mathsf{PRF}(\cdot,\cdot) : \mathbb{G}_T \times \{0,1\}^* \to \mathcal{K}_{\mathsf{AKE}}$. Let $pms := (\mathcal{MLG}, \{u_j\}_{0\leq j\leq n+1}, hk_{\mathsf{TCRHF}})$ be the variable used to store the public system parameters.

Long-term Key Generation and Registration: On input pms, a party $\hat{A}$ may run an efficient algorithm $(sk_{\hat{D}}, pk_{\hat{D}}, \emptyset) \xleftarrow{\$} \mathsf{ORGAKE.KGen}(pms, \hat{D})$ to generate the long-term key pair for a party $\hat{D}$ as: $sk_{\hat{D}} = d \xleftarrow{\$} \mathbb{Z}_p^*, pk_{\hat{D}} = (D, t_D)$, where $D = g^a$, $t_D := \prod_{j=0}^{n+1} u_j^{h_D^j}$ and $h_A = \mathsf{TCRHF}(A)$. Please note that we allow arbitrary key registration.

Let ω denote the size of group for a protocol instance such that $2 \leq \omega \leq n+1$. An important attribute for a GAKE protocol is the scalable group size. In the following we show our construction for protocol execution phase which is scalable with range between 2 and $n+1$.

Protocol Execution: We consider the protocol execution for a protocol instance with ω group members denoted by $(\hat{D}_1, \hat{D}_2, \ldots, \hat{D}_\omega)$, where each party $\hat{D}_i$ $(1 \leq i \leq \omega)$ has long-term key D_i. In the key exchange phase, each party $\hat{D}_i$ generates an ephemeral key $X_i = g^{x_i}$, computes tag $t_{X_i} := \prod_{j=0}^{n+1} u_j^{h_{X_i}^j}$ and broadcasts $(\hat{D}_i, D_i, t_{D_i}, X_i, t_{X_i})$ to its intended communication partners, where $x_i \xleftarrow{\$} \mathbb{Z}_p^*$ and $h_{X_i} := \mathsf{TCRHF}(X_i)$. Upon receiving all messages $\{\hat{D}_l, D_l, t_{D_1}, X_l, t_{X_l}\}_{1\leq l\leq\omega, l\neq i}$ from each session participant, the party $\hat{D}_i$ rejects the session if the consistency check on one of the received either long-term or ephemeral keys fails, i.e. $me(t_{W_l}, g, \ldots, g) \neq me(\prod_{j=0}^{n+1} u_j^{h_{W_l}^j}, W_l, g, \ldots, g)$ where $W_l \in \{D_l, X_l\}$ for $1 \leq l \leq \omega, l \neq i$ and $h_{W_l} = \mathsf{TCRHF}(W_l)$. The party $\hat{D}_i$ sets $\mathsf{sid} := \hat{D}_1||D_1||t_{D_1}||X_1||t_{X_1}||\ldots||\hat{D}_\omega||D_\omega||t_{D_\omega}||X_\omega||t_{X_\omega}$, and rejects the session if some values recorded in sid are identical. To this end, the party $\hat{D}_i$ generates the key material $k := me(D_1X_1, \ldots, D_{i-1}X_{i-1}, D_{i+1}X_{i+1}, \ldots, D_\omega X_\omega, \ldots, D_\omega X_\omega)^{d_i+x_i}$ and session key $k_e := \mathsf{PRF}(k, \mathsf{sid})$, where the values $D_0, X_0, D_{\omega+1}, X_{\omega+1}$ are 'empty' which should be omitted. Other parties in this group will do the similar procedures to generate the session key.

Please note that the scalability is achieved generally by setting all Diffie-Hellman keys after the position ω in n-multilinear map me to be $D_\omega X_\omega$. This is possible since at least one DH key in (D_ω, X_ω) is not compromised by adversary

in the security game. As otherwise such session is no longer fresh in terms of Definition 7.

Implementation and Session States: We assume that the maximum states of party $\hat{D}_i$ allowing for leakage from a session consist of ephemeral private key x_i – namely those values would be stored in the variable in the state variable *st* of each oracle at any time. The implementation scenario is similar to the three party case presented in Section 5.

6.2 Security Analysis

We show the security of above group AKE protocol in the g-eCK model.

Theorem 2. *Assume each ephemeral key chosen during key exchange has bit-size* $\lambda \in \mathbb{N}$. *Suppose that the* nMDDH *problem is* $(t, \epsilon_{\mathsf{nMDDH}})$*-hard in the symmetric multilinear groups* $\mathcal{MLG}$*, the* TCRHF *is* $(t, \epsilon_{\mathsf{TCRHF}})$*-secure target collision resistant hash function family, and the* PRF *is* $(q, t, \epsilon_{\mathsf{PRF}})$*-secure pseudo-random function family. Then the proposed protocol of size* $2 \leq \omega \leq n+1 \leq \ell$ *is* (t', ϵ)*-g-eCK-secure in the sense of Definition 8 with* $t' \approx t$, $q \geq n+1$ *and* $\epsilon \leq \frac{(\rho\ell)^2}{2^\lambda} + \epsilon_{\mathsf{TCRHF}} + (n+2)(\rho)^{n+1}\binom{\ell}{n+1} \cdot (\epsilon_{\mathsf{nMDDH}} + \epsilon_{\mathsf{PRF}})$.

The proof of theorem 2 is presented in the full version of this paper [16]. We lose a factor $(n+2)(\rho)^{n+1}\binom{\ell}{n+1}$ here which is exponential in group size n. Hence, in order to make the overall advantage of adversary to be negligible, one may need to use a larger security parameter or to limit the maximum group members.

Acknowledgments. We would like to thank the anonymous reviewers of CANS 2013 for their helpful comments.

References

1. Al-Riyami, S.S., Paterson, K.G.: Tripartite authenticated key agreement protocols from pairings. In: Paterson, K.G. (ed.) Cryptography and Coding 2003. LNCS, vol. 2898, pp. 332–359. Springer, Heidelberg (2003)
2. Baek, J., Safavi-Naini, R., Susilo, W.: Efficient multi-receiver identity-based encryption and its application to broadcast encryption. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 380–397. Springer, Heidelberg (2005)
3. Bellare, M., Boldyreva, A., Palacio, A.: An uninstantiable random-oracle-model scheme for a hybrid-encryption problem. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 171–188. Springer, Heidelberg (2004)
4. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Ashby, V. (ed.) ACM CCS 1993: 1st Conference on Computer and Communications Security, pp. 62–73. ACM Press (November 1993)
5. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 232–249. Springer, Heidelberg (1994)
6. Boneh, D., Silverberg, A.: Applications of multilinear forms to cryptography. IACR Cryptology ePrint Archive, Report 2002/80 (2002), `http://eprint.iacr.org/`

7. Bresson, E., Chevassut, O., Pointcheval, D.: Dynamic group Diffie-Hellman key exchange under standard assumptions. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 321–336. Springer, Heidelberg (2002)
8. Bresson, E., Chevassut, O., Pointcheval, D., Quisquater, J.-J.: Provably authenticated group Diffie-Hellman key exchange. In: ACM CCS 2001: Conference on Computer and Communications Security, pp. 255–264. ACM Press (November 2001)
9. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited (preliminary version). In: 30th Annual ACM Symposium on Theory of Computing, pp. 209–218. ACM Press (May 1998)
10. Fujioka, A., Manulis, M., Suzuki, K., Ustaoğlu, B.: Sufficient condition for ephemeral key-leakage resilient tripartite key exchange. In: Susilo, W., Mu, Y., Seberry, J. (eds.) ACISP 2012. LNCS, vol. 7372, pp. 15–28. Springer, Heidelberg (2012)
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)
12. Gorantla, M.C., Boyd, C., González Nieto, J.M.: Modeling key compromise impersonation attacks on group key exchange protocols. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 105–123. Springer, Heidelberg (2009)
13. Hofheinz, D., Jager, T., Kiltz, E.: Short signatures from weaker assumptions. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 647–666. Springer, Heidelberg (2011)
14. Joux, A.: A one round protocol for tripartite diffie-hellman. In: Bosma, W. (ed.) ANTS-IV 2000. LNCS, vol. 1838, pp. 385–394. Springer, Heidelberg (2000)
15. LaMacchia, B.A., Lauter, K., Mityagin, A.: Stronger security of authenticated key exchange. In: Susilo, W., Liu, J.K., Mu, Y. (eds.) ProvSec 2007. LNCS, vol. 4784, pp. 1–16. Springer, Heidelberg (2007)
16. 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. 123–142. Springer, Heidelberg (2013)
17. Lim, M.-H., Lee, S., Park, Y., Lee, H.: An enhanced one-round pairing-based tripartite authenticated key agreement protocol. In: Gervasi, O., Gavrilova, M.L. (eds.) ICCSA 2007, Part II. LNCS, vol. 4706, pp. 503–513. Springer, Heidelberg (2007)
18. Manulis, M., Suzuki, K., Ustaoglu, B.: Modeling leakage of ephemeral secrets in tripartite/group key exchange. In: Lee, D., Hong, S. (eds.) ICISC 2009. LNCS, vol. 5984, pp. 16–33. Springer, Heidelberg (2010)
19. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 256–266. Springer, Heidelberg (1997)
20. Shoup, V.: Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332 (2004), `http://eprint.iacr.org/`