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

Yong Li and Zheng Yang[1]

Horst Görtz Institute for IT Security
Chair for Network- and Data 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 bind-width 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, authenticated 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 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. [12], 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 Rogawary [7]. Since then, figuring out new useful security properties for certain class of GAKE and modelling them become continuing trends.

[1]Corresponding Author.

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 [24, 25] which extends the classical two-party Diffie-Hellman KE protocol [17] 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. [2, 31, 30, 33, 18], 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. [18]. The g-eCK model basically can be seen as a generalization from the two party eCK model [12, 28]. In contrast to earlier GAKE models [12, 11, 26, 13, 22], 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[18]. Therefore proving security for one-round GAKE in the g-eCK model may provide more guarantees.

Motivations. Manulis et al. [33] recently pointed out that the one-round 3AKE constructions [2, 31, 30] fail to achieve implicit key authentication, and introduced an improved protocol (which is called MSU protocol in the following). In 2012, Fujioka et al. (FMSU) [18] generalized previous 3AKE protocols into one framework based on admissible polynomials developed from [19] which yields many further one-round 3AKE protocols. Meanwhile the one-round version of MSU protocol [33] can be seen as a concrete instantiation of FMSU protocol.[1] The generic FMSU protocol [18] was shown to satisfy g-eCK security. However its security proof is given in the random oracle model (ROM) [6] under a specific strong assumption, i.e. gap Bilinear Diffie-Hellman (GBDH) assumption [3]. It is well-known that the security proof in the random oracle may not imply that corresponding protocol is secure in the real world. Several results, e.g., [14, 4], 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 need 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

[1]The MSU protocol consists of two communication rounds in which the first round is used to establish the session key and the key confirmation steps are done in the second round. But it is able to execute without the second round.

weak Programmable Hash Functions [23] 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. We make use of the pairing to provide a means of consistency checking that (both long-term and ephemeral) public keys coming from the adversary are in some sense of well-formed. Due to those tags, all public keys are mutually independent. Hence an active adversary is not able to lead non-partnered fresh sessions to generate co-related session keys. In particular, any active adversaries have to leave session key related secret information in those tags which can be extracted and exploited by the challenger (during the proof simulation) using corresponding trapdoor secret, e.g. the exponents of the group elements used to computing the tags. We make use of the fact those trapdoor information can't be trivially obtained by adversary. 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 [25].

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 [10]. Informally speaking, the symmetric multilinear groups are equipped with a n-multilinear maps $me : \mathbb{G}^n \to \mathbb{G}_T$ where $n \geq 2$ is an integer, $\mathbb{G}$ is a multiplicative cyclic group of large prime order p and $\mathbb{G}_T$ is the target group with the same order. Most recently, Garg, Gentry and Halvei [20] introduced a surprising candidate mechanism that would approximate multilinear maps in discrete-logarithm hard groups. Their result may open the opportunity to implement constructions using a multilinear map abstraction in practice. 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. This general proof is applicable when the group size of our GAKE protocol ranges from 2 to $n + 1$, that also implies the concrete security proof of our 3AKE protocol.

2 Preliminaries

In this section, we recall the required definitions for our result on proposed protocols.

Notations. We let $\kappa \in \mathbb{N}$ denote the security parameter and 1^κ the string that consists of κ ones. Let a 'hat' on top of a capital letter 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 $a \overset{\$}{\leftarrow} S$ denotes the action of sampling a uniformly random element from S. Let '$||$' denote the operation concatenating two binary strings.

2.1 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 $\mathcal{PG} = (\mathbb{G}, g, \mathbb{G}_T, p, e) \overset{\$}{\leftarrow} \mathsf{PG.Gen}(1^\kappa)$ 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.

2.2 Multilinear Groups

In the following, we recall the definition of symmetric multilinear groups introduced in [10]. 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 (Multilinear groups)**.** We call $\mathcal{MLG} = (\mathbb{G}, \mathbb{G}_T, p, me) \overset{\$}{\leftarrow} \mathsf{MLG.Gen}(\kappa, n)$ be symmetric multilinear groups, if the n-multilinear map me holds that:

1. **n-multilinear:** $\forall(c_1, \ldots, c_n) \in \mathbb{G}_1$ 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.

We here focus on symmetric n-multilinear groups, since our group AKE solution makes use of elements from a single group $\mathbb{G}$. Concrete multilinear maps can be found in [21, 20] by Garg, Gentry, and Halvei [21, 20]. We here just focus on a general definition of symmetric n-multilinear groups without loss of generality.

2.3 Cube Bilinear Decisional Diffie-Hellman Assumption

Let $\mathcal{PG} = (\mathbb{G}, g, \mathbb{G}_T, p, e)$ denote the description of symmetric bilinear group as Definition 1. The *Cube* Bilinear Decisional Diffie-Hellman (CBDDH) problem that is stated as follows: given $(g, g^a, e(g, g)^\gamma)$ for $(a, \gamma) \in (\mathbb{Z}_p^*)^2$ as input, output 1 if $e(g, g)^\gamma = e(g, g)^{a^3}$ and 0 otherwise.

Definition 3. We say that the CBDDH problem relative to generator $\mathsf{PG.Gen}$ 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:

```
$\mathsf{EXP}^{cbddh}_{\mathsf{PG.Gen},\mathcal{A}}(\kappa, n)$
    $\mathcal{PG} = (\mathbb{G}, g, \mathbb{G}_T, p, e) \overset{\$}{\leftarrow} \mathsf{PG.Gen}(1^\kappa)$;
    $(a, \gamma) \overset{\$}{\leftarrow} \mathbb{Z}_p^*$;
    $b \overset{\$}{\leftarrow} \{0, 1\}$, 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 the security parameter κ.

A relative hard problem is the Bilinear Decisional Diffie-Hellman (BDDH) problem [25], that given tuple $(g, g^a, g^b, g^c, e(g,g)^\gamma) \in \mathbb{G}^4 \times \mathbb{G}_T$ it is infeasible for any PPT adversary to distinguish whether or not $\gamma = abc$. It is not hard to see that if there exists an adversary $\mathcal{A}$ being able to solve the CBDDH problem with non-negligible advantage, then we can construct an efficient algorithm $\mathcal{B}$ using $\mathcal{A}$ to solve the BDDH problem. Since given a CBDDH challenge instance$(g, g^a, e(g,g)^\gamma)$, $\mathcal{B}$ can construct a BDDH challenge instance for $\mathcal{A}$ via choosing three random values $(b,c,d) \xleftarrow{\$} (\mathbb{Z}_p^*)^3$ and computing the instance as $(g, g^{ab}, g^{ac}, g^{ad}, e(g,g)^{\gamma bcd})$. Then if $\mathcal{A}$ is able to distinguish $e(g,g)^{\gamma bcd}$ whether or not $\gamma bcd = a^3bcd$, then $\mathcal{B}$ knows whetheror not $\gamma = a^3$. But we cannot do the reduction in a reverse direction. Nevertheless, the proof for the security of CBDDH assumption in the generic group model [34] is similar to the proof of the BDDH assumption [9, Appendix A], and is therefore omitted here.

2.4 n-Multiliear Decisional Diffie-Hellman Assumption

We present a generalization of the CBDDH assumption in n-multilinear groups $\mathcal{MLG} = (\mathbb{G}, \mathbb{G}_T, g, p, me)$ that we call the n-Multilinear Decisional Diffie-Hellman (nMDDH) assumption. Roughly speaking, the nMDDH problem is stated as follows: given $(g, g^a, me(g,\ldots,g)^\gamma)$ for $(a,\gamma) \in (\mathbb{Z}_p^*)^2$ as input, output 1 if $me(g,\ldots,g)^\gamma = me(g,\ldots,g)^{a^{n+1}}$ and 0 otherwise.

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

```
$\mathsf{EXP}^{nmddh}_{\mathsf{PG.Gen},\mathcal{A}}(\kappa)$
    $\mathcal{MLG} = (\mathbb{G}, \mathbb{G}_T, g, p, me) \xleftarrow{\$} \mathsf{MLG.Gen}(\kappa, n)$;
    $(a,\gamma) \xleftarrow{\$} \mathbb{Z}_p^*$;
    $b \xleftarrow{\$} \{0,1\}$, if $b = 1$ $\Gamma \leftarrow me(g_1,\ldots,g_1)^{a^{n+1}}$, 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 the security parameter κ.

2.5 Pseudo-Random Functions

Let $\mathsf{PRF} : \mathcal{K}_{\mathsf{PRF}} \times \mathcal{D}_{\mathsf{PRF}} \to \mathcal{R}_{\mathsf{PRF}}$ denote a family of deterministic functions, where $\mathcal{K}_{\mathsf{PRF}}$ is the key space, $\mathcal{D}_{\mathsf{PRF}}$ is the domain and $\mathcal{R}_{\mathsf{PRF}}$ is the range of PRF for security parameter κ. Let $\mathsf{RL} = \{(x_1,y_1),\ldots,(x_q,y_q)\}$ be a list which is used to record bit strings formed as tuple $(x_i,y_i) \in (\mathcal{D}_{\mathsf{PRF}}, \mathcal{R}_{\mathsf{PRF}})$ where $1 \le i \le q$ and $q \in \mathbb{N}$. So that in RL each x is associated with a y. Let $\mathsf{RF} : \mathcal{D}_{\mathsf{PRF}} \to \mathcal{R}_{\mathsf{PRF}}$ be a stateful uniform random function, which can be executed at most a polynomial number of q times and keeps a list RL for recording each invocation. On input a message $x \in \mathcal{D}_{\mathsf{PRF}}$, the function $\mathsf{RF}(x)$ is executed as follows:

- If $x \in \mathsf{RL}$, then return corresponding $y \in \mathsf{RL}$,
- Otherwise return $y \xleftarrow{\$} \mathcal{R}_{\mathsf{PRF}}$ and record (x,y) into RL.

Definition 5. We say that PRF is a $(q, t, \epsilon_{\mathsf{PRF}})$-secure pseudo-random function family, if it holds that $|\Pr[\mathsf{EXP}^{prf}_{\mathsf{PRF},\mathcal{A}}(\kappa) = 1] - 1/2| \le \epsilon_{\mathsf{PRF}}$ for all adversaries $\mathcal{A}$ running in probabilistic polynomial time t and making at most q oracle queries in the following experiment:

```
$\mathsf{EXP}^{prf}_{\mathsf{PRF},\mathcal{A}}(\kappa)$                          | $\mathcal{F}(b,x)$
  $b \xleftarrow{\$} \{0,1\}, k \xleftarrow{\$} \mathcal{K}_{\mathsf{PRF}}$         |   If $x \notin \mathcal{D}_{\mathsf{PRF}}$ then return $\perp$
  $b' \leftarrow \mathcal{A}^{\mathcal{F}(b,\cdot)}(\kappa)$                        |   If $b = 1$ then return $\mathsf{PRF}(k,x)$
  if $b = b'$ then return 1, otherwise return 0;     |   Otherwise return $\mathsf{RF}(x)$
```

where $\epsilon_{\mathsf{PRF}} = \epsilon_{\mathsf{PRF}}(\kappa)$ is a negligible function in the security parameter κ, and the number of allowed queries q is bound by t.

2.6 Target Collision-Resistant Hash Functions

Let $\mathsf{TCRHF} : \mathcal{K}_{\mathsf{TCRHF}} \times \mathcal{M}_{\mathsf{TCRHF}} \rightarrow \mathcal{Y}_{\mathsf{TCRHF}}$ be a family of keyed-hash functions associated with key space $\mathcal{K}_{\mathsf{TCRHF}}$, message space $\mathcal{M}_{\mathsf{TCRHF}}$ and hash value space $\mathcal{Y}_{\mathsf{TCRHF}}$. The public key $hk_{\mathsf{TCRHF}} \in \mathcal{K}_{\mathsf{CRHF}}$ of a hash function $\mathsf{TCRHF}(hk_{\mathsf{TCRHF}}, \cdot)$ is generated by a PPT algorithm $\mathsf{TCRHF.KG}(1^\kappa)$ on input security parameter κ.

Definition 6. TCRHF is called $(t, \epsilon_{\mathsf{TCRHF}})$-target-collision-resistant if for all t-time adversaries $\mathcal{A}$ it holds that

$$\Pr\begin{bmatrix} hk_{\mathsf{TCRHF}} \xleftarrow{\$} \mathsf{TCRHF.KG}(1^\kappa),\ m \xleftarrow{\$} \mathcal{M}_{\mathsf{TCRHF}},\ m' \leftarrow \mathcal{A}(1^\kappa, hk_{\mathsf{TCRHF}}, m), \\ m \neq m',\ m' \in \mathcal{M}_{\mathsf{TCRHF}},\ \mathsf{TCRHF}(hk_{\mathsf{TCRHF}}, m) = \mathsf{TCRHF}(hk_{\mathsf{TCRHF}}, m') \end{bmatrix} \leq \epsilon_{\mathsf{TCRHF}},$$

where the probability is over the random bits of $\mathcal{A}$.

Note that the notion of target collision resistance is both qualitatively and quantitatively weaker than the notion of (full) collision resistance. Commonly target collision resistant functions can be implemented with a dedicated cryptographic hash function like MD5 or SHA, as described in [16]. If the hash key hk_{TCRHF} is obvious from the context, we write $\mathsf{TCRHF}(m)$ for $\mathsf{TCRHF}(hk_{\mathsf{TCRHF}}, m)$.

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, 26, 28, 33, 18] which is initiated by Bellare and Rogaway [7]. 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). In practice, the concrete implementation of proof is up to the CA [1] and may be either interactive or non-interactive. Examples can be found in RFC 4210 [1] and PKCS#10. In this model we focus on non-interactive proof. 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:

- pid_i^s – A variable stores 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.

- Φ_i^s – A variable stores the oracle decision $\Phi_i^s \in \{\texttt{accept}, \texttt{reject}\}$.
- K_i^s – A variable records the session key $K_i^s \in \mathcal{K}_{\mathsf{KE}}$ for symmetric encryption.
- st_i^s – A variable stores the maximum secret session states that are allowed to be leaked (e.g., the exponent of exchanged ephemeral public key).
- T_i^s – A variable stores 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 $\Phi_i^s \in \{\texttt{accept}, \texttt{reject}\}$. 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 $\Phi_i^s = \texttt{accept}$.

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 ($\ell < \tau < \mathbb{N}$) 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.[2] 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, which corresponds to the *proof of knowledge* assumptions for public key registration as discussed in [32, 5]. 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 [18] 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. We first consider the correctness of a GAKE protocol to rule out those useless protocols.

Definition 7 (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 $K_i^s = K_j^t$.

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 8. 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 $T_i^s = T_j^t$.

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 for $i \in [\ell]$ where the identity $\mathsf{ID}_i \in \mathcal{IDS}$ of each party is chosen uniquely. $\mathcal{C}$ gives adversary $\mathcal{A}$ all identities, public keys and corresponding proofs $\{(\mathsf{ID}_1, pk_{\mathsf{ID}_1}, \mathsf{proof}_{\mathsf{ID}_1}), \ldots, (\mathsf{ID}_\ell, pk_{\mathsf{ID}_\ell}, \mathsf{proof}_{\mathsf{ID}_\ell})\}$ as input.

2. $\mathcal{A}$ may issue polynomial number of queries as aforementioned, namely $\mathcal{A}$ makes queries: Send, $\mathsf{StateReveal}$, $\mathsf{Corrupt}$, $\mathsf{RegisterCorrupt}$ and $\mathsf{RevealKey}$.

3. At some point, $\mathcal{A}$ may issue a $\mathsf{Test}(\pi_i^s)$ query on an oracle π_i^s during the experiment with only once.

4. At the end of the 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 9 (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:

1. $\mathcal{A}$ queried $\mathsf{RegisterCorrupt}(\mathsf{ID}_j, pk_{\mathsf{ID}_j}, \mathsf{proof}_{\mathsf{ID}_j})$ with some $\mathsf{ID}_j \in \mathsf{pid}_i^s$.

2. $\mathcal{A}$ queried either $\mathsf{RevealKey}(\pi_i^s)$ or $\mathsf{RevealKey}(\pi_j^t)$ for some oracle $\pi_j^t \in \pi_S$.

[2]Similar modelling technique involving secure hardware was previously used by Bresson et al. [11]. Basing the security models on specific implementation approach reduces the gap that often exists between formal models and practical security without loss of generality, and this also enables us to define the detailed content of $\mathsf{StateReveal}$ query.

3. $\mathcal{A}$ queried both $\mathsf{Corrupt}(\mathsf{ID}_i)$ and $\mathsf{StateReveal}(\pi_i^s)$.

4. For some oracle $\pi_j^t \in \pi_S$, $\mathcal{A}$ queried both $\mathsf{Corrupt}(\mathsf{ID}_j)$ and $\mathsf{StateReveal}(\pi_j^t)$.

5. 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 $\mathsf{Corrupt}(\mathsf{ID}_j)$.

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

Definition 10 (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 an oracle π_i^s without failure and π_i^s is fresh throughout the security game, 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.

- $(sk_{\mathsf{ID}}, pk_{\mathsf{ID}}, \mathsf{proof}_{\mathsf{ID}}) \overset{\$}{\leftarrow} \mathsf{ORGAKE.KGen}(pms, \mathsf{ID})$: 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.).

- $m_{\mathsf{ID}_1} \overset{\$}{\leftarrow} \mathsf{ORGAKE.MF}(pms, sk_{\mathsf{ID}_1}, r_{\mathsf{ID}_1}, \mathsf{GD})$: This algorithm takes as input system parameters pms and the sender ID_1's secret key sk_{ID_1}, a randomness $r_{\mathsf{ID}_1} \overset{\$}{\leftarrow} \mathcal{R}_{\mathsf{ORGAKE}}$ 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.[3]

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

- $K \leftarrow \mathsf{ORGAKE.SKG}(pms, sk_{\mathsf{ID}_1}, r_{\mathsf{ID}_1}, \mathsf{GD}, \mathsf{T})$: 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[4], and outputs session key $K \in \mathcal{K}_{\mathsf{ORGAKE}}$.

  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 9. Let oracle $\pi^{s^*}_{\hat{A}}$ 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 $\pi^{t^*}_{\hat{B}}$ and $\pi^{l^*}_{\hat{C}}$, such that $\pi^{s^*}_{\hat{A}}$ has matching session to $\pi^{t^*}_{\hat{B}}$ and to $\pi^{l^*}_{\hat{C}}$ respectively.
- **Event 1**: There is an oracle $\pi^{t^*}_{\hat{F}}$ such that $\pi^{s^*}_{\hat{A}}$ and $\pi^{t^*}_{\hat{F}}$ have matching sessions but there is no oracle at $\hat{D}$ having matching session to $\pi^{s^*}_{\hat{A}}$, 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**: $\pi^{s^*}_{\hat{A}}$ has no matching session.

Besides the restrictions regarding $\mathsf{RegisterCorrupt}$ and $\mathsf{RevealKey}$ queries which are 'deterministic', we particular may obtain different freshness events on whether the test oracle has matching sessions and corresponding freshness cases related to different combinations of $\mathsf{StateReveal}$ and $\mathsf{Corrupt}$ queries which are 'flexible' and determined by adversary's choice. However, among those different freshness cases, at least one would occur in the security game. 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).

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 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 $\epsilon_{\mathcal{A}_1} = \epsilon_{\mathcal{A}_2}$.*

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

Event 0	$\pi^{s^*}_{\hat{A}}$	$\pi^{t^*}_{\hat{B}}$	$\pi^{l^*}_{\hat{C}}$
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	$\pi^{s^*}_{\hat{A}}$	$\pi^{t^*}_{\hat{F}}$	$\hat{D}$
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	$\pi^{s^*}_{\hat{A}}$	$\hat{B}$	$\hat{C}$
Case 13 (C13)	nC	nC	nC
Case 14 (C14)	nRS	nC	nC

Table 1: Freshness Cases in Each Event

PROOF. Intuitively, in cases $C2$ and $C5$, the test oracle has matching sessions, then the adversary could selects either an oracle $\pi^{s^*}_{\hat{A}}$ or its partners $\pi^{t^*}_{\hat{B}}$ or $\pi^{l^*}_{\hat{C}}$ as test oracle since $\pi^{s^*}_{\hat{A}}$, $\pi^{t^*}_{\hat{B}}$ and $\pi^{l^*}_{\hat{C}}$ will compute the same session key. In both cases the adversary reveals the states of two oracles and corrupt a party. We show the security reduction as follows. $\mathcal{A}_2$ interacts with the AKE challenger $\mathcal{C}$ and tries to break the security of considered protocol under freshness case $C5$. It runs $\mathcal{A}_1$ as subroutine and responds all oracle queries except for the test oracle. When $\mathcal{A}_1$ issues the Test query to $\pi^{s^*}_{\hat{A}}$, $\mathcal{A}_2$ selects the matching partner $\pi^{t^*}_{\hat{B}}$ as the test oracle. When $\mathcal{A}_2$ receives the real session key or random key, $\mathcal{A}_2$ sends it to $\mathcal{A}_1$. If $\mathcal{A}_1$ outputs a bit, $\mathcal{A}_2$ outputs the same bit. Note that $\mathcal{A}_2$ can issue $\mathsf{StateReveal}(\pi^{s^*}_{\hat{A}})$ since it has matching session to the 'test oracle' (i.e. $\pi^{t^*}_{\hat{B}}$) from the view of $\mathcal{A}_2$. $\mathcal{A}_2$ can issue $\mathsf{Corrupt}(\hat{A})$ since from the view of $\mathcal{A}_2$ the party $\hat{A}$ is the intended partner of its 'test oracle'. So $\mathcal{A}_2$ can correctly respond to all queries issued by $\mathcal{A}_1$. Therefore, if $\mathcal{A}_1$ breaks the security of the considered protocol in case $C2$, $\mathcal{A}_2$ wins the game in case $C5$ with the same advantage as $\mathcal{A}_1$'s. $\square$

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 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 $\epsilon_{\mathcal{A}_1} = \epsilon_{\mathcal{A}_2}$.*

PROOF. This proof is similar to the proof of Proposition 1. $\mathcal{A}_2$ interacts with an AKE challenger $\mathcal{C}$ and tries to break the security of considered protocol under freshness case $C9$. It runs $\mathcal{A}_1$ as subroutine and responds all the oracle queries except for the test oracle. $\mathcal{A}_1$ issues the Test query to oracle $\pi^{s^*}_{\hat{A}}$ which has a matching session to oracle $\pi^{t^*}_{\hat{F}}$ and to $\pi^{l^*}_{\hat{D}}$ in the view of $\mathcal{A}_1$. Note that the oracle $\pi^{t^*}_{\hat{F}}$ is simulated by challenger $\mathcal{C}$, but the oracle $\pi^{l^*}_{\hat{D}}$ is simulated by $\mathcal{A}_2$ on behalf of $\hat{D}$ for $\mathcal{A}_1$ (since $\mathcal{A}_2$ is the challenger of $\mathcal{A}_1$). The message generated by oracle $\pi^{l^*}_{\hat{D}}$ is either generated by $\mathcal{A}_2$ or obtained from an oracle simulated by $\mathcal{C}$, which depends on $\mathcal{A}_2$'s choice. So that when $\mathcal{A}_2$ receives the real session key or random key, $\mathcal{A}_2$ sends it to $\mathcal{A}_1$. If $\mathcal{A}_1$ outputs a bit, $\mathcal{A}_2$ outputs the same bit. $\mathcal{A}_2$ can respond to any oracle queries issued by $\mathcal{A}_1$ since the restricted oracle queries are equivalent. Therefore, $\mathcal{A}_2$ breaks the security of the considered protocol in case $C9$ if $\mathcal{A}_1$ wins the game in case $C3$ ($C5$). $\square$

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 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 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 $\epsilon_{\mathcal{A}_1} = \epsilon_{\mathcal{A}_2} = \epsilon_{\mathcal{A}_3}$.*

PROOF. This proof is similar to the proofs of Proposition 1 and Proposition 2. $\mathcal{A}_2$ runs $\mathcal{A}_1$ as subroutine and responds all the oracle queries except for the test oracle. $\mathcal{A}_1$ issues the Test query to oracle $\pi^{s^*}_{\hat{A}}$ which has a matching session to oracle $\pi^{t^*}_{\hat{F}}$ and to $\pi^{l^*}_{\hat{D}}$ (in the view of $\mathcal{A}_1$). $\mathcal{A}_2$ could select $\pi^{t^*}_{\hat{F}}$ as test oracle which is possible since it has matching session to $\pi^{s^*}_{\hat{A}}$. So that when $\mathcal{A}_2$ receives the real session key or random key, $\mathcal{A}_2$ sends it to $\mathcal{A}_1$. If $\mathcal{A}_1$ outputs a bit, $\mathcal{A}_2$ outputs the same bit. $\mathcal{A}_2$ can respond to any oracle queries issued by $\mathcal{A}_1$ since the restricted oracle queries for those adversaries are equivalent. Therefore, $\mathcal{A}_2$ breaks the security of the considered protocol in case $C11$ if $\mathcal{A}_1$ wins the game in case $C7$. Analogously we have the reduction from $C11$ to $C13$, since the restricted oracle queries are the same to adversaries. □

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 $C4$ ($C6$), then there exists 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 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 $\epsilon_{\mathcal{A}_1} = \epsilon_{\mathcal{A}_2} = \epsilon_{\mathcal{A}_3} = \epsilon_{\mathcal{A}_4}$.*

Proof. This proof is similar to the proofs of Proposition 1, Proposition 2 and Proposition 3. Thus we omit the detail here for avoid repetition. □

The above reductions routes are informally depicted in the Figure 1.

$$\begin{array}{r} C2 \to C5(C3) \to C9 \\ C7 \to C11 \to C13 \\ C4(C6) \to C8 \to C12 \to C14 \end{array}$$

Figure 1: Reductions of g-eCK-Freshness for One-round Tripartite AKE

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 need 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. However, we might not be able to formally do so in a short page when the group size $n \in \mathbb{N}$ is a large integer, in which the total number of the freshness cases would be very large. So that we can only make certain conjecture for the lower bound of target freshness cases for the proof of AAKE protocol with arbitrary group size n in the 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.*

Proof. Please first note that each freshness case consists of two main aspects: (i) the status (e.g. the number) of matching sessions, (ii) the restrictions of adversary's queries. Since there are at most $n+1$ parties in a protocol instance for which can be queried either Corrupt or StateReveal. It is not hard to see that there are $n+1$ distinct events for the status of matching sessions, i.e. matching sessions of test oracle vary between n and 0. Let 'Event i' ($0 \leq i \leq n$) denote the situation that the test oracle has $n-i$ matching sessions (we use the similar representation approach as three party case). In Event i, there is 2^{n-i+1} distinct freshness cases, because either the test oracle or each oracle having matching session to test oracle has two distinct cases, i.e. it can be either corrupted or revealed states by adversary. Collect the number of freshness cases in each event, we have total number of freshness cases $2(2^{n+1}-1) = \sum_{i=0}^{n} 2^{n-i+1}$. From the reductions for three party case, we know that freshness cases have analogous or the same query restrictions could be somehow reduced. So that we only need to do proof simulations for these 'target' freshness cases having distinct query restrictions. We observe that there are $n+2$ such target freshness cases. Since each event would contribute one distinct freshness case, except for the Event n in which event there is two distinct freshness cases related to test oracle. Namely in Event i ($i \neq n$) that the test oracle has $n - i$ matching sessions, we have the distinct freshness case as: The adversary did not query StateReveal to these $n - i$ matching sessions (of test oracle) and did not query Corrupt to these i parties which have no matching session to test oracle. $\square$

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.

Design Principle. The challenge here is that we have to simultaneously cope with chosen identity and long-term public key (CIDPK) attacks (modeled by RegisterCorrupt query) and chosen ephemeral key (CEK) attacks (modeled by Send query) in presence of strong adversaries who can reveal non-trivial session states (via StateReveal query) and even compromise the long-term keys of participants (via Corrupt query). The CIDPK attack addresses the situation that the adversary registers dishonest identity and public key and tries to subvert the security, e.g. obtain information about honest user's secret key via small sub-group attacks [29]. The CEK attack addresses the situation that the adversary tries to manipulate the session key via exchanged ephemeral keys of her own choice. To deal with these complicated situations, we have to set up a proof simulation for our construction in the g-eCK model that is able to simulate all queries 'appropriately'.

Our main idea is to make use of the weak (3,poly)-PHF [23] to resist with not only CEK attacks but also CIDPK attacks under the g-eCK model. This is possible, since there are at most three (either long-term or ephemeral) public keys will not be compromised by adversary. However, we can't efficiently construct the protocol based on BDDH assumption. Because in a BDDH challenge instance, all DH keys are distinct to each other. Consider the most awkward case that there are at most three uncorrupted parties, each of which may possess a long-term key generated by a BDDH challenge value. Thus we might need at least three different weak (3,poly)-PHFs to plug in all BDDH challenge values in order to simulate the session keys correctly for all those uncorrupted oracles. To avoid this inefficient setting, the CBDDH assumption might be a perfect alternative choice. We could simultaneously embed CBDDH challenge value into these uncompromised DH keys and the parameters of weak (3,poly)-PHF in the security proof.

5.1 Protocol Description

We describe the protocol in terms of the following three parts: Setup, long-termkey generation and key registration, protocol execution, one could think of the general algorithms defined in Section 4 are implied in specific part.

Setup: The proposed protocol takes as input the following building blocks which are initialized respectively in terms of the security parameter $\kappa \in \mathbb{N}$:

- Symmetric bilinear groups $\mathcal{PG} = (\mathbb{G}, g, \mathbb{G}_T, p, e) \overset{\$}{\leftarrow} \mathsf{PG.Gen}(1^\kappa)$ and a set of random values $\{u_i\}_{0 \leq i \leq 3} \overset{\$}{\leftarrow} \mathbb{G}$,
- a target collision resistant hash function $\mathsf{TCRHF}(hk_{\mathsf{TCRHF}}, \cdot) : \mathcal{K}_{\mathsf{TCRHF}} \times \mathbb{G} \rightarrow \mathbb{Z}_p$, where $hk_{\mathsf{TCRHF}} \overset{\$}{\leftarrow} \mathsf{TCRHF.KG}(1^\kappa)$, and
- a pseudo-random function family $\mathsf{PRF}(\cdot, \cdot) : \mathbb{G}_T \times \{0,1\}^* \rightarrow \mathcal{K}_{\mathsf{AKE}}$.

The system parameter variable encompasses $pms := (\mathcal{PG}, \{u_i\}_{0 \leq i \leq 3}, hk_{\mathsf{TCRHF}})$.

$\hat{A}$	$\hat{B}$	$\hat{C}$
$(sk_{\hat{A}} = a \xleftarrow{\$} \mathbb{Z}_p^*,$	$(sk_{\hat{B}} = b \xleftarrow{\$} \mathbb{Z}_p^*,$	$(sk_{\hat{C}} = c \xleftarrow{\$} \mathbb{Z}_p^*,$
$pk_{\hat{A}} = (A, t_A) :=$	$pk_{\hat{B}} = (B, t_B) :=$	$pk_{\hat{C}} = (C, t_C) :=$
$(g^a, (u_0 u_1^{h_A} u_2^{h_A^2} u_3^{h_A^3})^a)$	$(g^b, (u_0 u_1^{h_B} u_2^{h_B^2} u_3^{h_B^3})^b)$	$(g^c, (u_0 u_1^{h_C} u_2^{h_C^2} u_3^{h_C^3})^c)$
$x \xleftarrow{\$} \mathbb{Z}_p^*, X := g^x$	$y \xleftarrow{\$} \mathbb{Z}_p^*, Y := g^y$	$z \xleftarrow{\$} \mathbb{Z}_p^*, Z := g^z$
$h_X := \mathsf{TCRHF}(X)$	$h_Y := \mathsf{TCRHF}(Y)$	$h_Z := \mathsf{TCRHF}(Z)$
$t_X := (u_0 u_1^{h_X} u_2^{h_X^2} u_3^{h_X^3})^x$	$t_Y := (u_0 u_1^{h_Y} u_2^{h_Y^2} u_3^{h_Y^3})^y$	$t_Z := (u_0 u_1^{h_Z} u_2^{h_Z^2} u_3^{h_Z^3})^z$
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

$k := e(BY, CZ)^{a+x}$	$k := e(AX, CZ)^{b+y}$	$k := e(AX, BY)^{c+z}$
$k_e := \mathsf{PRF}(k, sid)$	$k_e := \mathsf{PRF}(k, sid)$	$k_e := \mathsf{PRF}(k, sid)$

Figure 2: One-round Tripartite AKE Protocol

Long-term Key Generation and Registration: On input $pms := (\mathcal{PG}, \{u_i\}_{0 \le i \le 3}, hk_{\mathsf{TCRHF}})$, a party $\hat{A}$ may run an efficient algorithm $(sk_{\hat{A}}, pk_{\hat{A}}, \emptyset) \overset{\$}{\leftarrow} \mathsf{ORGAKE.KGen}(pms, \hat{A})$ to generate the long-term key pair as: $sk_{\hat{A}} = a \overset{\$}{\leftarrow} \mathbb{Z}_p^*, pk_{\hat{A}} = (A, t_A)$ 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 the system parameter pms, the protocol among parties $\hat{A}$, $\hat{B}$ and $\hat{C}$ is executed as following, which is also informally depicted in the Figure 2.

1. Upon activating a new session with participants $(\hat{A}, \hat{B}, \hat{C})$, the party $\hat{A}$ first chooses an ephemeral private key $x \overset{\$}{\leftarrow} \mathbb{Z}_p^*$ and compute ephemeral public key $X := g^x$. Next $\hat{A}$ computes $h_X := \mathsf{TCRHF}(X)$, and $t_X := (u_0 u_1^{h_X} u_2^{h_X^2} u_3^{h_X^3})^x$. To the end $\hat{A}$ broadcasts messages $(\hat{A}, A, t_A, X, t_X)$ to $\hat{B}$ and $\hat{C}$.

2. Upon activating a new session with participants $(\hat{A}, \hat{B}, \hat{C})$, the party $\hat{B}$ first chooses an ephemeral private key $y \overset{\$}{\leftarrow} \mathbb{Z}_p^*$ and compute ephemeral public key $Y := g^y$. Next $\hat{B}$ computes $h_Y := \mathsf{TCRHF}(Y)$, and $t_Y := (u_0 u_1^{h_Y} u_2^{h_Y^2} u_3^{h_Y^3})^y$. To the end $\hat{B}$ broadcasts messages $(\hat{B}, B, t_B, Y, t_Y)$ to $\hat{A}$ and $\hat{C}$.

3. Upon activating a new session with participants $(\hat{A}, \hat{B}, \hat{C})$, the party $\hat{C}$ first chooses an ephemeral private key $z \overset{\$}{\leftarrow} \mathbb{Z}_p^*$ and compute ephemeral public key $Z := g^z$. Next $\hat{C}$ computes $h_Z := \mathsf{TCRHF}(Z)$, and $t_Z := (u_0 u_1^{h_Z} u_2^{h_Z^2} u_3^{h_Z^3})^z$. To the end $\hat{A}$ broadcasts messages $(\hat{A}, \hat{B}, \hat{C}, \hat{A}, C, t_C, Z, t_Z)$ to $\hat{A}$ and $\hat{B}$.

4. Upon receiving $(\hat{B}, B, t_B, Y, t_Y)$ and $(\hat{C}, C, t_C, Z, t_Z)$, the party $\hat{A}$ sets identifier $\mathsf{sid} := \hat{A}||A||t_A||X||t_X||\hat{B}||B||t_B||Y||t_Y||\hat{C}||C||t_C||Z||t_Z$ and rejects the session if some values recorded in sid are identical. Next $\hat{A}$ computes $h_B = \mathsf{TCRHF}(B)$, $h_C = \mathsf{TCRHF}(C)$, $h_Y = \mathsf{TCRHF}(Y)$ and $h_Z = \mathsf{TCRHF}(Z)$ and rejects the session if either $e(t_B, g) \neq e(u_0 u_1^{h_B} u_2^{h_B^2} u_3^{h_B^3}, B)$ or $e(t_C, g) \neq e(u_0 u_1^{h_C} u_2^{h_C^2} u_3^{h_C^3}, C)$ or $e(t_Y, g) \neq e(u_0 u_1^{h_Y} u_2^{h_Y^2} u_3^{h_Y^3}, Y)$ or $e(t_Z, g) \neq e(u_0 u_1^{h_Z} u_2^{h_Z^2} u_3^{h_Z^3}, Z)$. Finally, $\hat{A}$ computes $k := e(BY, CZ)^{a+x}$ and session key $k_e := \mathsf{PRF}(k, sid)$.

5. Upon receiving $(\hat{A}, A, t_A, X, t_X)$ and $(\hat{C}, C, t_C, Z, t_Z)$, the party $\hat{B}$ sets identifier $\mathsf{sid} := \hat{A}||A||t_A||X||t_X||\hat{B}||B||t_B||Y||t_Y||\hat{C}||C||t_C||Z||t_Z$ and rejects the session if some values recorded in sid are identical. Next $\hat{B}$ computes $h_A = \mathsf{TCRHF}(A)$, $h_C = \mathsf{TCRHF}(C)$, $h_X = \mathsf{TCRHF}(X)$ and $h_Z = \mathsf{TCRHF}(Z)$ and rejects the session if either $e(t_A, g) \neq e(u_0 u_1^{h_A} u_2^{h_A^2} u_3^{h_A^3}, A)$ or $e(t_C, g) \neq e(u_0 u_1^{h_C} u_2^{h_C^2} u_3^{h_C^3}, C)$ or $e(t_X, g) \neq e(u_0 u_1^{h_X} u_2^{h_X^2} u_3^{h_X^3}, X)$ or $e(t_Z, g) \neq e(u_0 u_1^{h_Z} u_2^{h_Z^2} u_3^{h_Z^3}, Z)$. Finally, $\hat{B}$ computes $k := e(AX, CZ)^{b+z}$ and session key $k_e := \mathsf{PRF}(k, sid)$.

6. Upon receiving $(\hat{A}, A, t_A, X, t_X)$ and $(\hat{B}, B, t_B, Y, t_Y)$ the party $\hat{C}$ sets identifier $\mathsf{sid} := \hat{A}||A||t_A||X||t_X||\hat{B}||B||t_B||Y||t_Y||\hat{C}||C||t_C||Z||t_Z$ and rejects the session if some values recorded in sid are identical. Next $\hat{C}$ computes $h_B = \mathsf{TCRHF}(B)$, $h_C = \mathsf{TCRHF}(C)$, $h_X = \mathsf{TCRHF}(X)$ and $h_Y = \mathsf{TCRHF}(Y)$ and rejects the session if either $e(t_B, g) \neq e(u_0 u_1^{h_B} u_2^{h_B^2} u_3^{h_B^3}, B)$ or $e(t_C, g) \neq e(u_0 u_1^{h_C} u_2^{h_C^2} u_3^{h_C^3}, C)$ or $e(t_X, g) \neq e(u_0 u_1^{h_X} u_2^{h_X^2} u_3^{h_X^3}, X)$ or $e(t_Y, g) \neq e(u_0 u_1^{h_Y} u_2^{h_Y^2} u_3^{h_Y^3}, Y)$. Finally, $\hat{A}$ computes $k := e(AX, BY)^{a+x}$ and session key $k_e := \mathsf{PRF}(k, sid)$.

   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.

In a nutshell, other non-trivial states, e.g. the secret exponent $c + z$ and key material k, should be carefully protected. We stress that it is not allowed to simultaneously leak the ephemeral private key say z and secret key material say $k = e(AX, BY)^{c+z}$ to any attackers. Otherwise the protocol is insecure in the g-eCK model. Since such attacker can simply replay the ephemeral key say $X = g^x$ generated by test session owned by $\hat{A}$ to any session of $\hat{C}$ and extract non-trivial secret $e(AX, BY)^c$ from the knowledge of z and k, where Y could be chosen by the attacker on behalf of $\hat{B}$. Then it can break the security by sending any ephemeral keys $Z' = g^{z'}$ and Y of her own choice on behalf of $\hat{C}$ and $\hat{B}$ respectively to test session which generates the session key $\mathsf{PRF}(e(AX, BY)^{c+z'}, \mathsf{sid})$. Analogously the leakage of ephemeral private key z and corresponding secret exponent $c + z$ would lead to the expose of private key c. As well the leakage of only exponent $c+z$ would enable adversary to launch infinite replay attacks.

We remark that our scheme can satisfy perfect forward secrecy by increase key confirmation procedures in an extra round, but the protocol then would become less efficient. We leave this problem for future work, that is to construct secure one-round GAKE protocols in the g-eCK+ model [33].

5.2 Performance Improvement

In this section, we discuss the issue on how to improve the efficiency of proposed one-round tripartite protocol. Obviously, the consistency checks on both long-term and ephemeral keys are somewhat costly which requires four pairing operations in each session. Thus we mainly focus on the performance improvement concerning those consistency checks.

We first introduce an alternative consistency checking algorithm which is derived from the similar technique in [27] used to improve the efficiency of identity-based KEM scheme. The idea is to merge consistency checks on incoming Diffie-Hellman keys. In the new consistency check algorithm, upon receiving $(\hat{C}, \hat{A}, A, t_A, X, t_X)$ and $(\hat{C}, C, t_C, Z, t_Z)$ the party $\hat{A}$ may perform the following steps:

1. Compute $U_B := u_0 u_1^{h_B} u_2^{h_B^2} u_3^{h_B^3}$, $U_C := u_0 u_1^{h_C} u_2^{h_C^2} u_3^{h_C^3}$, $U_Y := u_0 u_1^{h_Y} u_2^{h_Y^2} u_3^{h_Y^3}$ and $U_Z := u_0 u_1^{h_Z} u_2^{h_Z^2} u_3^{h_Z^3}$.
2. Choose four random values $\theta_1, \theta_2, \theta_3, \theta_4 \xleftarrow{\$} \mathbb{Z}_p^*$.
3. Reject the session if $e(t_B^{\theta_1} t_C^{\theta_2} t_Y^{\theta_3} t_Z^{\theta_4}, g) \neq e(U_B^{\theta_1}, B) e(U_C^{\theta_2}, C) e(U_Y^{\theta_3}, Y) e(U_Z^{\theta_4}, Z)$.

We claim that the combined consistency check equation implies that all received tags are consistent. In order to prove our argument we define functions $\Delta_1(t_Y) := \frac{e(u_0 u_1^{h_Y} u_2^{h_Y^2} u_3^{h_Y^3}, Y)}{e(t_Y, g)}$, $\Delta_2(t_B) := \frac{e(u_0 u_1^{h_B} u_2^{h_B^2} u_3^{h_B^3}, B)}{e(t_B, g)}$, $\Delta_1(t_Z) := \frac{e(u_0 u_1^{h_Z} u_2^{h_Z^2} u_3^{h_Z^3}, Z)}{e(t_Z, g)}$ and $\Delta_1(t_C) := \frac{e(u_0 u_1^{h_C} u_2^{h_C^2} u_3^{h_C^3}, C)}{e(t_C, g)}$. Obviously, we have $\Delta_1(t_Y) = \Delta_2(t_B) = \Delta_1(t_Z) = \Delta_2(t_C) = 1$ if and only if t_Y, t_B, t_Z, t_C are consistent. Consequently, for random values $\theta_1, \theta_2, \theta_3, \theta_4 \xleftarrow{\$} \mathbb{Z}_p^*$, function $(\Delta_1(t_Y))^{\theta_1} (\Delta_2(t_B))^{\theta_2} (\Delta_3(t_Z))^{\theta_3} (\Delta_4(t_C))^{\theta_4}$ evaluates to 1 if t_Y, t_B, t_Z, t_C are consistent and to a random group value in $\mathbb{G}_T$ otherwise. This alternative consistency check algorithm substitutes one multiple-exponentiation for three pairing operations. Note that the above technique could be extended to merge more consistency check equations that would dramatically improve the efficiency of consistency check procedure, e.g. the consistency check operations in our upcoming one-round group AKE protocols in Section 6.

Furthermore, 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. $\mathsf{proof}_{\hat{A}} = t_A$.

5.3 Security Analysis

We show the security of proposed protocol in our strong security 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 10 with $t' \approx t$, $q \geq 3$ and $\epsilon \leq \frac{(\rho\ell)^2}{2^\lambda} + \epsilon_{\mathsf{TCRHF}} + 4(\rho\ell)^3 \cdot \epsilon_{\mathsf{CBDDH}} + \epsilon_{\mathsf{PRF}}$.*

The full proof of theorem 1 is presented in Appendix A.

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

6.1 Protocol Description

Setup: The proposed protocol takes as input the following building blocks which are initialized respectively in terms of the security parameter $\kappa \in \mathbb{N}$ and upper-bound of group size $n+1$:

- n-mulitilinear groups $\mathcal{MLG} = (\mathbb{G}, \mathbb{G}_T, g, p, me) \overset{\$}{\leftarrow} \mathsf{MLG.Gen}(\kappa, n)$ and a set of random values $\{u_j\}_{0 \leq j \leq n+1} \overset{\$}{\leftarrow} \mathbb{G}$.
- a target collision resistant hash function $\mathsf{TCRHF}(hk_{\mathsf{TCRHF}}, \cdot) : \mathcal{K}_{\mathsf{TCRHF}} \times \mathbb{G} \to \mathbb{Z}_p$, where $hk_{\mathsf{TCRHF}} \overset{\$}{\leftarrow} \mathsf{TCRHF.KG}(1^\kappa)$, and
- 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 $pms := (\mathcal{MLG}, \{u_j\}_{0 \leq j \leq n+1}, hk_{\mathsf{TCRHF}})$, a party $\hat{A}$ may run an efficient algorithm $(sk_{\hat{D}}, pk_{\hat{D}}, \emptyset) \overset{\$}{\leftarrow} \mathsf{ORGAKE.KGen}(pms, \hat{D})$ to generate the long-term key pair for a party $\hat{D}$ as: $sk_{\hat{D}} = d \overset{\$}{\leftarrow} \mathbb{Z}_p^*, pk_{\hat{D}} = (D, t_D)$, 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, i.e. the adversary is able to query $\mathsf{RegisterCorrupt}(\hat{D}, pk_{\hat{D}}, \emptyset)$ with $\mathsf{proof}_{\hat{D}} = \emptyset$.

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 $n+1$. Recall that the upper bound of group size is determined by the n-multilinear map.

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 $x_i \overset{\$}{\leftarrow} \mathbb{Z}_p^*$ 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. $me(t_{W_l}, g, \dots, g) \neq me(\prod_{j=0}^{n+1} u_j^{h_{W_l}^j}, W_l, g, \dots, g)$ where $W_l \in \{D_l, X_l\}$ for $1 \leq l \leq \omega, l \neq i$ and $h_{W_l} = \mathsf{TCRHF}(W_l)$.[5] The party $\hat{D}_i$ sets $\mathsf{sid} := \hat{D}_1||D_1||t_{D_1}||X_1||t_{X_1}||\dots||\hat{D}_\omega||D_\omega||t_{D_\omega}||X_\omega||t_{X_\omega}$, and rejects the session if some values recorded in sid are identical. To this end, the party $\hat{D}_i$ generates the key material $k := me(D_1X_1, \dots, D_{i-1}X_{i-1}, D_{i+1}X_{i+1}, \dots, D_\omega X_\omega, \dots, D_\omega X_\omega)^{d_i + x_i}$ 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 9.

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, namely generate the k and k_e on secure device.

Remark 1. The above construction implies the proposed tripartite AKE protocol in Section 5 if the parameter of n-multilinear map such that $n = 2$ which is equivalent to bilinear map. Then the scalable construction of GAKE could also yield a two party eCK secure AKE protocol that might be of independent interesting. It is not hard to see that the security of such two party AKE protocol can be proved without random oracles based on CBDDH assumption in the g-eCK model when group size equals two (i.e. then it implies the eCK model).

6.2 Security Analysis

We show the security of above group AKE protocol in our strong security 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 $2 \leq \omega \leq n+1$ is (t', ϵ)-g-eCK-secure in the sense of Definition 10 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 full proof of theorem 2 is presented in Appendix B.

References

[1] C. Adams, S. Farrell, T. Kause, and T. Mononen. Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP). RFC 4210 (Proposed Standard), September 2005.

[2] Sattam S. Al-Riyami and Kenneth G. Paterson. Tripartite authenticated key agreement protocols from pairings. In Kenneth G. Paterson, editor, *9th IMA International Conference on Cryptography and Coding*, volume 2898 of *Lecture Notes in Computer Science*, pages 332–359. Springer, December 2003.

[3] Joonsang Baek, Reihaneh Safavi-Naini, and Willy Susilo. Efficient multi-receiver identity-based encryption and its application to broadcast encryption. In Serge Vaudenay, editor, *PKC 2005: 8th*

[5]As for multilinear maps which are implemented with a series of bilinear maps, e.g.the framework by Garg et al. [20], one could use the bilinear map e instead of me in those consistency check operations for efficiency consideration. One could also use the similar technique in Section 5.2 to merge those consistency checks.

International Workshop on Theory and Practice in Public Key Cryptography, volume 3386 of *Lecture Notes in Computer Science*, pages 380–397. Springer, January 2005.

[4] Mihir Bellare, Alexandra Boldyreva, and Adriana Palacio. An uninstantiable random-oracle-model scheme for a hybrid-encryption problem. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 171–188. Springer, May 2004.

[5] Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 06: 13th Conference on Computer and Communications Security*, pages 390–399. ACM Press, October / November 2006.

[6] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93: 1st Conference on Computer and Communications Security*, pages 62–73. ACM Press, November 1993.

[7] Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In Douglas R. Stinson, editor, *Advances in Cryptology – CRYPTO'93*, volume 773 of *Lecture Notes in Computer Science*, pages 232–249. Springer, August 1994.

[8] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 409–426. Springer, May / June 2006.

[9] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 440–456. Springer, May 2005.

[10] Dan Boneh and Alice Silverberg. Applications of multilinear forms to cryptography. IACR Cryptology ePrint Archive, Report 2002/80, 2002. `http://eprint.iacr.org/`.

[11] Emmanuel Bresson, Olivier Chevassut, and David Pointcheval. Dynamic group Diffie-Hellman key exchange under standard assumptions. In Lars R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 321–336. Springer, April / May 2002.

[12] Emmanuel Bresson, Olivier Chevassut, David Pointcheval, and Jean-Jacques Quisquater. Provably authenticated group Diffie-Hellman key exchange. In *ACM CCS 01: 8th Conference on Computer and Communications Security*, pages 255–264. ACM Press, November 2001.

[13] Emmanuel Bresson, Mark Manulis, and Jörg Schwenk. On security models and compilers for group key exchange protocols. In Atsuko Miyaji, Hiroaki Kikuchi, and Kai Rannenberg, editors, *IWSEC 07: 2nd International Workshop on Security, Advances in Information and Computer Security*, volume 4752 of *Lecture Notes in Computer Science*, pages 292–307. Springer, October 2007.

[14] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In *30th Annual ACM Symposium on Theory of Computing*, pages 209–218. ACM Press, May 1998.

[15] Ran Canetti and Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In Birgit Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 453–474. Springer, May 2001.

[16] Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, 33(1):167–226, 2003.

[17] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[18] Atsushi Fujioka, Mark Manulis, Koutarou Suzuki, and Berkant Ustaoglu. Sufficient condition for ephemeral key-leakage resilient tripartite key exchange. In Willy Susilo, Yi Mu, and Jennifer Seberry, editors, *ACISP 12: 17th Australasian Conference on Information Security and Privacy*, volume 7372 of *Lecture Notes in Computer Science*, pages 15–28. Springer, July 2012.

[19] Atsushi Fujioka and Koutarou Suzuki. Designing efficient authenticated key exchange resilient to leakage of ephemeral secret keys. In Aggelos Kiayias, editor, *Topics in Cryptology – CT-RSA 2011*, volume 6558 of *Lecture Notes in Computer Science*, pages 121–141. Springer, February 2011.

[20] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices and applications. IACR Cryptology ePrint Archive,2002:610, 2012. `http://eprint.iacr.org/`.

[21] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT*, volume 7881 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2013.

[22] M. Choudary Gorantla, Colin Boyd, and Juan Manuel González Nieto. Modeling key compromise impersonation attacks on group key exchange protocols. In Stanislaw Jarecki and Gene Tsudik, editors, *PKC 2009: 12th International Conference on Theory and Practice of Public Key Cryptography*, volume 5443 of *Lecture Notes in Computer Science*, pages 105–123. Springer, March 2009.

[23] Dennis Hofheinz, Tibor Jager, and Eike Kiltz. Short signatures from weaker assumptions. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 647–666. Springer, December 2011.

[24] Antoine Joux. A one round protocol for tripartite diffie-hellman. In *ANTS*, pages 385–394, 2000.

[25] Antoine Joux. A one round protocol for tripartite Diffie-Hellman. *Journal of Cryptology*, 17(4):263–276, September 2004.

[26] Jonathan Katz and Moti Yung. Scalable protocols for authenticated group key exchange. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 110–125. Springer, August 2003.

[27] Eike Kiltz and David Galindo. Direct chosen-ciphertext secure identity-based key encapsulation without random oracles. In Lynn Margaret Batten and Reihaneh Safavi-Naini, editors, *ACISP 06: 11th Australasian Conference on Information Security and Privacy*, volume 4058 of *Lecture Notes in Computer Science*, pages 336–347. Springer, July 2006.

[28] Brian A. LaMacchia, Kristin Lauter, and Anton Mityagin. Stronger security of authenticated key exchange. In Willy Susilo, Joseph K. Liu, and Yi Mu, editors, *ProvSec 2007: 1st International Conference on Provable Security*, volume 4784 of *Lecture Notes in Computer Science*, pages 1–16. Springer, November 2007.

[29] Chae Hoon Lim and Pil Joong Lee. A key recovery attack on discrete log-based schemes using a prime order subgroup. In Burton S. Kaliski Jr., editor, *Advances in Cryptology – CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 249–263. Springer, August 1997.

[30] Meng-Hui Lim, Sanggon Lee, Youngho Park, and Hoonjae Lee. An enhanced one-round pairing-based tripartite authenticated key agreement protocol. In Osvaldo Gervasi and Marina L. Gavrilova, editors, *ICCSA (2)*, volume 4706 of *Lecture Notes in Computer Science*, pages 503–513. Springer, 2007.

[31] Chu-Hsing Lin and Hsiu-Hsia Lin. Secure one-round tripartite authenticated key agreement protocol from weil pairing. In *AINA*, pages 135–138. IEEE Computer Society, 2005.

[32] Steve Lu, Rafail Ostrovsky, Amit Sahai, Hovav Shacham, and Brent Waters. Sequential aggregate signatures and multisignatures without random oracles. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 465–485. Springer, May / June 2006.

[33] Mark Manulis, Koutarou Suzuki, and Berkant Ustaoglu. Modeling leakage of ephemeral secrets in tripartite/group key exchange. In Donghoon Lee and Seokhie Hong, editors, *ICISC 09: 12th International Conference on Information Security and Cryptology*, volume 5984 of *Lecture Notes in Computer Science*, pages 16–33. Springer, December 2009.

[34] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *Advances in Cryptology – EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266. Springer, May 1997.

[35] Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332, 2004. `http://eprint.iacr.org/`.

A Proof of Theorem 1

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. Without loss of generality, we consider that the adversary chooses the test oracle $\pi_{\hat{A}}^{s^*}$ executed with its indented partners $\hat{B}$ and $\hat{C}$.

Next we introduce the notations which might be used in the proof. Let the ephemeral keys generated by oracles $\pi_{\hat{A}}^{s}$, $\pi_{\hat{B}}^{t}$ and $\pi_{\hat{C}}^{t}$ are $X = g^x$, $Y = g^y$ and $Z = g^z$ respectively. we use the superscript '*' to highlight corresponding values processed by the test oracle and its partner oracles (if they exist), say the ephemeral key X^* generated by oracle $\pi_{\hat{A}}^{s^*}$. Let $D = g^d$, $F = g^f$, $W = g^w$ and $V = g^v$ denote the Diffie-Hellman (DH) keys received by an oracle π_i^s ($i \in [\ell]$) which are used to compute the session key, where these DH keys could be either ephemeral or long-term key.

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 (by Proposition 1, Proposition 2, Proposition 3 and Proposition 4.) in Section 4. The proof proceeds in a sequence of games, following [35, 8]. Let S_δ be the event that the adversary wins the security experiment in Game G_δ and freshness cases in the set $\{C1, C9, C13, C14\}$. 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. Meanwhile, the challenger chooses four uniform random values $r_i \xleftarrow{\$} \mathbb{Z}_p^*$ for $0 \le i \le 3$, and sets $u_i := g^{r_i}$ as public parameters. Thus we have that

$$\Pr[\mathsf{S}_0] = 1/2 + \epsilon = 1/2 + \mathsf{Adv}_0.$$

Game G_1. In this game, the challenger proceeds exactly like previous game, except that we add an abort rule. The challenger raises event $\mathsf{abort}_{\mathsf{eph}}$ and aborts, if during the simulation an ephemeral key (say X) replied by an oracle π_i^s but it has been sample by another oracle or sent by adversary before. Since there are $\rho\ell$ such ephemeral keys would be sampled uniform randomly from $\{0,1\}^\lambda$. Thus, the event $\mathsf{abort}_{\mathsf{eph}}$ occurs with probability $\Pr[\mathsf{abort}_{\mathsf{eph}}] \leq \frac{(\rho\ell)^2}{2^\lambda}$. We have that

$$\mathsf{Adv}_0 \leq \mathsf{Adv}_1 + \frac{(\rho\ell)^2}{2^\lambda}.$$

Note that the ephemeral key chosen by each oracle is unique in this game.

Game G_2. In this game we want to make sure that the received Diffie-Hellman keys are correctly formed. Technically, we add an abort condition, namely the challenger proceeds exactly as before, but raises event $\mathsf{abort}_{\mathsf{hash}}$ and aborts if there exist two distinct (either ephemeral or long-term) public keys M and N such that $\mathsf{TCRHF}(M) = \mathsf{TCRHF}(N)$. Obviously the $\Pr[\mathsf{abort}_{\mathsf{hash}}] \leq \epsilon_{\mathsf{TCRHF}}$, according to the security property of underlying hash function. 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 $\pi_{\hat{A}}^{s^*}$, (iii) its partner parties $\hat{B}$ and $\hat{C}$, and (iv) corresponding oracles (if any) $\pi_{\hat{D}}^{t^*}$ ($\hat{D} \in \{\hat{B}, \hat{C}\}$) such that $\pi_{\hat{A}}^{s^*}$ has a matching session to $\pi_{\hat{D}}^{t^*}$, 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

$$\mathsf{Adv}_2 \leq 4(\rho\ell)^3 \cdot \mathsf{Adv}_3.$$

Game G_4. Please first 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*.

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:

- The k_i^s is computed involving the three *target DH keys* which are guessed by $\mathcal{C}$ for test oracle, and
- Those *target DH keys* used by π_i^s are from three distinct parties.

The second condition is necessary, because the adversary can easily result in one oracle receiving DH keys from certain party which are all uncompromised via e.g. Send query $\mathsf{RegisterCorrupt}$ queries. On the other side, if those uncompromised DH keys are not from distinct parties, that might imply all DH keys from certain party are chosen (or revealed) by adversary. In this case, the adversary can compute the session key herself. The above two conditions are used to ensure that the changed key materials of oracles can not be trivially generated by adversary. This also enables us to embed CBDDH challenge instance into the simulation of those modified oracles.

If there exists an adversary $\mathcal{A}$ can distinguish the Game G_4 from Game G_3 then we can use it to construct a distinguisher $\mathcal{D}$ to solve the CBDDH problem as follows. Given a CBDDH challenge instance $(g, g^\mu, \Gamma) \in \mathbb{G}^2 \times \mathbb{G}_T$, the goal of $\mathcal{D}$ is to determine whether $\Gamma = e(g,g)^{\mu^3}$ or a random element from $\mathbb{G}_T$ where g is a generator of $\mathbb{G}$. Let $p(h) = p_0 + p_1 h + p_2 h^2 + p_3 h^3$ be a polynomial of degree 3 over

$\mathbb{Z}_p^*$. The detail form of this polynomial will be discussed in the simulation based on specific freshness case. Let $q(h) = q_0 + q_1h + q_2h^2 + q_3h^3$ be a random polynomial of degree 3 over $\mathbb{Z}_p^*$. In the following, $\mathcal{D}$ simulates the challenger for $\mathcal{A}$ as previous game but with the some modifications based on its correct guesses (otherwise it aborts).

1. **Case** $C1$. In this case $\mathcal{D}$ does the following modifications:
   (a) Set $X^* := g^{\mu r_x}, Y^* := g^{\mu r_y}$ and $Z^* := g^{\mu r_z}$ where $r_x, r_y, r_x \xleftarrow{\$} \mathbb{Z}_p^*$.
   (b) Compute the key material of test oracle and its partner oracles as:
      - $k^*_{\hat{A}} = k^*_{\hat{B}} = k^*_{\hat{C}} := \Gamma^{r_x r_y r_z} \cdot e(CZ^*, BY^*)^a \cdot e(Z^*, X^*)^b \cdot e(BY^*, X^*)^c$.
   (c) Compute those tags of test oracle and its partner oracles as:
      - $t^*_X := (X^*)^{r_3(h^*_X)^3 + r_2(h^*_X)^2 + r_1 h^*_X + r_0}$,
        $t^*_Y := (Y^*)^{r_3(h^*_Y)^3 + r_2(h^*_Y)^2 + r_1 h^*_Y + r_0}$ and
        $t^*_Z := (Z^*)^{r_3(h^*_Z)^3 + r_2(h^*_Z)^2 + r_1 h^*_Z + r_0}$.
2. **Case** $C9$. We assume there is an oracle $\pi^{t^*}_{\hat{B}}$ having matching session to test oracle without loss of generality. In this case $\mathcal{D}$ does the following modifications:
   (a) Set $C := g^{\mu r_c}, X^* := g^{\mu r_x}$ and $Y^* := g^{\mu r_y}$ where $r_c, r_x, r_y \xleftarrow{\$} \mathbb{Z}_p^*$.
   (b) Set polynomial $p(h)$ to satisfy that $p(h) = (h - h^*_X)(h - h^*_Y)(h - h_C)$, where $h^*_Y = \mathsf{TCRHF}(Y^*)$, $h^*_X = \mathsf{TCRHF}(X^*)$ and $h_C = \mathsf{TCRHF}(C)$.
   (c) Set $u_i = g^{\mu p_i} g^{q_i}$ for $0 \leq i \leq 3$.
   (d) Compute the tags $t_C = C^{q(h_C)}$, $t^*_Y = (Y^*)^{q(h^*_Y)}$ and $t^*_X = (Z^*)^{q(h^*_X)}$.
   (e) Compute the key material of test oracle and its partner oracle as:
      - $k^*_{\hat{A}} = k^*_{\hat{B}} := \Gamma^{r_x r_y r_c} \cdot e(CV^*, BY^*)^a \cdot e((\frac{t_V}{V^{q(h_V)}})^{r_x/p(h_V)}, BY^*) \cdot e(C, X^*)^b$.
   (f) Compute the key material of other oracles of $\hat{C}$ in terms of the following situations:
      - There are at most two DH keys in $\{F, V, D, W\}$ which are equivalent to keys in the set $\{X^*, Y^*\}$ and from different parties. Since the adversary can either register these keys as public key for dishonest users or replay them as ephemeral key. Recall that these DH keys in $\{F, V, D, W, C, Z\}$ should be distinct in corresponding sid. We assume that $E = X^*$ and $D = Y^*$ for example, then the $k^l_{\hat{C}}$ is computed as: $k^l_{\hat{C}} := \Gamma^{r_x r_y r_c} \cdot e((\frac{t_W}{W^{q(h_W)}})^{r_c/p(h_W)}, FV) \cdot e((\frac{t_V}{V^{q(h_V)}})^{r_c/p(h_V)}, D)$.
      - The DH keys (either long-term or ephemeral public key) from one party do not belong to the set $\{X^*, Y^*\}$. We assume that $\{F, V\} \notin \{X^*, Y^*\}$ for example, then the $k^l_{\hat{C}}$ is computed as:
      - $k^l_{\hat{C}} := e((\frac{t_E}{E^{q(h_E)}})^{r_c/p(h_E)}(\frac{t_V}{V^{q(h_V)}})^{r_c/p(h_V)}, DW) \cdot e(FV, DW)^z$ when $V \notin \{X^*, Y^*\}$.
3. **Case** $C13$. In this case $\mathcal{D}$ does the following modifications:
   (a) Set $A := g^{\mu r_a}$, $B := g^{\mu r_b}$, and $C := g^{\mu r_c}$, where $r_a, r_b, r_c \xleftarrow{\$} \mathbb{Z}_p^*$.
   (b) Set polynomial $p(h)$ to satisfy that $p(h) = (h - h_A)(h - h_B)(h - h_C)$, where $h_A = \mathsf{TCRHF}(A)$, $h_B = \mathsf{TCRHF}(B)$ and $h_C = \mathsf{TCRHF}(C)$.
   (c) Set $u_i = g^{\mu p_i} g^{q_i}$ for $0 \leq i \leq 3$. Please note that we have that $u_0 u_1^h u_2^{h^2} u_3^{h^3} = g^{\mu p(h)} g^{q(h)}$ and $u_0 u_1^h u_2^{h^2} u_3^{h^3} = g^{\mu p(h)} g^{q(h)}$.
   (d) Compute the tags $t_A = A^{q(h_A)}$, $t_B = B^{q(h_B)}$ and $t_C = C^{q(h_C)}$.
   (e) Replace the value $e(B, C)^a$ with $\Gamma^{r_a r_b r_c}$ when compute the key material k of oracles $\pi^s_{\hat{A}}$, $\pi^t_{\hat{B}}$ and $\pi^l_{\hat{C}}$ which involve all public keys (A, B, C) including the test oracle $\pi^{s^*}_{\hat{A}}$, more specifically:

- $k^s_{\hat{A}} := \Gamma^{r_a r_b r_c} \cdot e(CV, BW)^x \cdot e((\frac{t_W}{W^{q(h_W)}})^{r_a/p(h_W)}, C) \cdot e((\frac{t_V}{V^{q(h_V)}})^{r_a/p(h_V)}, BW)$.
- $k^t_{\hat{B}} := \Gamma^{r_a r_b r_c} \cdot e(CV, AW)^y \cdot e((\frac{t_W}{W^{q(h_W)}})^{r_b/p(h_W)}, C) \cdot e((\frac{t_V}{V^{q(h_V)}})^{r_b/p(h_V)}, AW)$.
- $k^l_{\hat{C}} := \Gamma^{r_a r_b r_c} \cdot e(BV, AW)^z \cdot e((\frac{t_W}{W^{q(h_W)}})^{r_c/p(h_W)}, B) \cdot e((\frac{t_V}{V^{q(h_V)}})^{r_c/p(h_V)}, AW)$.

(f) Compute the secret key material k for other oracles of parties $\hat{A}$, $\hat{B}$ and $\hat{C}$, following the similar approach as did in the proof of Case $C9$ when computing the key material for oracles of uncorrupted party $\hat{C}$ (i.e. the modification in the step 2f). The common point here is that we should simulate the key material for situations when there exist DH keys equal to challenged DH keys (i.e. A or B or C).

4. **Case** $C14$. In this case $\mathcal{D}$ does the following modifications:

(a) Set $X^* := g^{\mu r_x}$, $B := g^{\mu r_b}$ and $C := g^{\mu r_c}$ where $r_x, r_b, r_c \xleftarrow{\$} \mathbb{Z}^*_p$.

(b) Set polynomial $p(h)$ to satisfy that $p(h) = (h - h^*_X)(h - h_B)(h - h_C)$, where $h_B = \mathsf{TCRHF}(B)$, $h^*_X = \mathsf{TCRHF}(X^*)$ and $h_C = \mathsf{TCRHF}(C)$.

(c) Set $u_i = g^{\mu p_i} g^{q_i}$ for $0 \leq i \leq 3$.

(d) Compute the tag $t_B = B^{q(h_B)}$, $t_C = C^{q(h_C)}$ and the tag $t^*_X = (X^*)^{q(h^*_X)}$.

(e) Compute the key material $k^*_{\hat{A}}$ of test oracle $\pi^{s^*}_{\hat{A}}$, $k^t_{\hat{B}}$ of oracles $\pi^t_{\hat{B}}$ and $k^l_{\hat{C}}$ of oracles $\pi^l_{\hat{C}}$ which compute the session keys using public keys (X^*, B, C) as:

- $k^*_{\hat{A}} := \Gamma^{r_x r_b r_c} \cdot e(CV, BW)^a \cdot e((\frac{t_W}{W^{q(h_W)}})^{r_x/p(h_W)}, C) \cdot e((\frac{t_V}{V^{q(h_V)}})^{r_x/p(h_V)}, BW)$.
- $k^t_{\hat{B}} := \Gamma^{r_x r_b r_c} \cdot e(CV, X^*D)^y \cdot e((\frac{t_D}{D^{q(h_D)}})^{r_b/p(h_D)}, C) \cdot e((\frac{t_V}{V^{q(h_V)}})^{r_b/p(h_V)}, DX^*)$.
- $k^l_{\hat{C}} := \Gamma^{r_x r_b r_c} \cdot e(BW, X^*D)^z \cdot e((\frac{t_D}{D^{q(h_D)}})^{r_c/p(h_D)}, B) \cdot e((\frac{t_W}{W^{q(h_W)}})^{r_c/p(h_W)}, DX^*)$.

(f) Change the computation of secret key material k of other oracles of $\hat{B}$ and $\hat{C}$ following the similar approach as did in the proof of Case $C9$ when computing the key material for oracles of uncorrupted party $\hat{C}$. One could think of replacing the symbols (e.g. Y^*) in the step 2f with the symbols (e.g. B) in this case.

Those modified tags are consistent with the original form. we make use of the fact there is no collision on those hash values due to the result of previous game. To answer the $\mathsf{RevealKey}$ query for those modified oracles, the $\mathcal{D}$ will use the changed key material (e.g. $k_{\hat{B}}$) to compute the final session key as protocol specification. With respect to the other queries, the $\mathcal{D}$ simulates them honestly as the challenger using corresponding values chosen by herself. Without flipping the bit b, the Test-query is replied with the session key which is computed using modified key material. Based on the condition that all guesses of $\mathcal{D}$ are correct, if $\Gamma = e(g, g)^{\mu^3}$, then the simulation is equivalent to Game $\mathsf{G_3}$; otherwise the simulation is equivalent to Game $\mathsf{G_4}$. At the end of the game, $\mathcal{D}$ returns what $\mathcal{A}$ returns to the CBDDH challenger. If $\mathcal{A}$ can distinguish the real key from the random value, that implies $\mathcal{D}$ solves the CBDDH problem. We therefore obtain that

$$\mathsf{Adv}_3 \leq \mathsf{Adv}_4 + \epsilon_{\mathsf{CBDDH}}.$$

Game $\mathsf{G_5}$. In this game, we change function $\mathsf{PRF}(\widetilde{k^*_{\hat{A}}}, \cdot)$ to a truly random function for test oracle and its partner oracles (if they exist). We make use of the fact that the secret seed $\widetilde{k^*_{\hat{A}}}$ of test oracle is a truly random value. Any PPT algorithm distinguishing the Game $\mathsf{G_5}$ from Game $\mathsf{G_4}$ implies that it is able to break the security of the pseudo-random function PRF. Thus we have that

$$\mathsf{Adv}_4 \leq \mathsf{Adv}_5 + \epsilon_{\mathsf{PRF}}.$$

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 $\mathsf{G_0}$ to Game $\mathsf{G_5}$, we proved this theorem.

B Proof of Theorem 2

Basically, the proof can be generalized from the proof of Theorem 1 due to the intimate relationship between proposed 3AKE and GAKE schemes. We will focus on the largest group size $n+1$ without loss of generality, because we need to evaluate the maximum advantage of adversary to break the protocol. For the test query involving group of size smaller than $n+1$, the simulation is quite similar.

Let S_δ be the event that the adversary wins the security experiment 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 $\mathcal{A}$. The system parameters are chosen honestly by challenger as protocol specification. However, the challenger chooses $n+1$ uniform random values $\{r_j\} \xleftarrow{\$} \mathbb{Z}_p^*$ for $0 \le j \le n$, and sets $u_j := g^{r_j}$ as public parameters. Thus we have that

$$\Pr[\mathsf{S}_0] = 1/2 + \epsilon = 1/2 + \mathsf{Adv}_0.$$

Game G_1. This game proceeds as the same as the Game 2 in the proof of Theorem 1. With the similar argument from the proof of Game 2 of Theorem 1, we have that

$$|\mathsf{Adv}_0 - \mathsf{Adv}_1| \le \frac{(\rho\ell)^2}{2^\lambda} + \epsilon_{\mathsf{TCRHF}}.$$

Game G_2. 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 all $n+2$ possibilities, (ii) the test oracle, (iii) the n intended communication partners of test oracle, and (iv) every oracles (if they exist in terms of specific guessed freshness case) which have matching session to test oracle. Since there are $n+2$ fresh cases that need to do proof simulation, ℓ parties at all and at most ρ oracles for each party, then the probability that all above guesses of $\mathcal{C}$ are correct is at least $\frac{1}{(n+2)(\rho)^{n+1}\binom{\ell}{n+1}}$. Thus we have that

$$\mathsf{Adv}_1 \le (n+2)(\rho)^{n+1}\binom{\ell}{n+1} \cdot \mathsf{Adv}_2.$$

Game G_3. Please note that the g-eCK freshness definition guarantees that for our protocol there are at least $n+1$ Diffie-Hellman (DH) keys from all session participants of test fresh oracle are not compromised by adversary. We call such guessed $n+1$ uncompromised DH keys as *target DH keys*. 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:

- The k_i^s is computed involving the $n+1$ *target DH keys* which are guessed by $\mathcal{C}$ for test oracle, and
- Those *target DH keys* used by π_i^s are from $n+1$ distinct parties.

Of course if two oracles have matching sessions and satisfy both above conditions, then we could use the same modified random key material to generate corresponding session key. The above two conditions ensure that the changed key materials of oracles can not be trivially generated by adversary. This also enables us to embed nMDDH challenge instance into the simulation of all oracles satisfying above conditions. The second condition is used to exclude the situation that the DH keys from some party are all compromised in which case the adversary can simply compute the session key.

If there exists an adversary $\mathcal{A}$ can distinguish the Game 3 and 2 then we can use it to construct a distinguisher $\mathcal{D}$ to solve the nMDDH problem. Given a nMDDH challenge instance $(g, g^\mu, \Gamma) \in \mathbb{G}^2 \times \mathbb{G}_T$, the goal of $\mathcal{D}$ is to determine whether $\Gamma = me(g, \dots, g)^{\mu^{n+1}}$ or a random element from $\mathbb{G}_T$ where g is a generator of $\mathbb{G}$. Meanwhile, $\mathcal{D}$ simulates the challenger for $\mathcal{A}$ as previous game but with the following modifications based on its correct guesses (otherwise it aborts). We highlight that, after all those correct guesses, $\mathcal{D}$ knows the 'distribution' of all $n+1$ uncompromised target DH keys among honest parties and theirs oracles. Namely $\mathcal{D}$ knows the facts about which parties' long-term keys are not corrupted (if any) and which oracles' ephemeral keys are not revealed (if any), under specific guessed freshness cases. Let $p(h) = \sum_{j=0}^{n+1} p_j^{h^j} = (h - h_{W_1}) \dots (h - h_{W_{n+1}})$ be a polynomial of degree $n+1$ over $\mathbb{Z}_p^*$ such that $p(h_{W_1}) = p(h_{W_2}), \dots, = p(h_{W_{n+1}}) = 0$ where $h_{W_j} = \mathsf{TCRHF}(W_j)$ for $1 \leq j \leq n+1$ and each W_j is either uncorrupted long-term key D_j or uncompromised ephemeral key X_j in specific freshness case. Let $q(h) = \sum_{j=0}^{n+1} q_j^{h^j}$ be a random polynomial of degree $n+1$ over $\mathbb{Z}_p^*$. It will also set $u_j = g^{\mu p_j} g^{q_j}$ for $0 \leq j \leq n+1$. Meanwhile, we would plug the challenge value g^μ to all $n+1$ target uncompromised DH keys in specific (guessed) freshness case, i.e. $\mathcal{D}$ generates the DH key as $W_j = g^{\mu r_{w_j}}$ where $r_{w_j} \xleftarrow{\$} \mathbb{Z}_p^*$. Moreover, the tag t_{W_j} of W_j would be computed as $t_{W_j} = W_j^{q(h_{W_j})}$. The remaining problem is to simulate the $\mathsf{RevealKey}$ query and Test query correctly in terms of freshness case.

On the next we discuss how to simulate the key material for any oracle π_i^s ($i \in [\ell], s \in [\rho]$), including test oracle and its partner oracle (if they exists). In the sequel, we let $(D_1, t_{D_1}, X_1, t_{X_1})$ denote the values generated for oracle π_i^s, and let $\{D_j, t_{D_j}, X_j, t_{X_j}\}_{2 \leq j \leq n+1}$ denote a set of values received by oracle π_i^s.[6] We consider the following cases (which cover all) concerning the DH keys of π_i^s:

1. Case 1: the ephemeral key X_1 is generated from challenge value g^μ.

2. Case 2: the long-term key D_1 is generated from challenge value g^μ.

3. Case 3: neither long-term key D_1 nor ephemeral key X_1 is generated from challenge value g^μ.

It is not hard to see, in the Case 3 $\mathcal{D}$ can simulate the key honestly as protocol specification. Thus we only need to do modifications on oracles π_i^s under Case 1 and Case 2. With respect to the Case 1, the $d_1 \xleftarrow{\$} \mathbb{Z}_p^*$ is chosen by $\mathcal{D}$ as protocol specification and X_1 is generated using challenge value as $X_1 := g^{\mu r_{x_1}}$ where $r_{x_1} \xleftarrow{\$} \mathbb{Z}_p^*$. Then the tag t_{X_1} can be computed as $t_{X_1} := X_1^{q(h_{X_1})}$ and $h_{X_1} = \mathsf{TCRHF}(X_1)$. With respect to the Case 2, the $x_1 \xleftarrow{\$} \mathbb{Z}_p^*$ might be chosen by $\mathcal{D}$ and D_1 can be set as $D_1 := g^{\mu r_{d_1}}$ where $r_{d_1} \xleftarrow{\$} \mathbb{Z}_p^*$. The tag t_{D_1} can be computed as $t_{D_1} := D_1^{q(h_{D_1})}$ and $h_{D_1} = \mathsf{TCRHF}(D_1)$.

Let W_1 denote the DH key generated for oracle π_i^s such that $W_1 \in \{D_1, X_1\}$ and W_1 is generated using challenge value as $g^{\mu r_{w_1}}$ where $r_{w_1} \in \{r_{x_1}, r_{d_1}\}$ depending on the value of W_1. We further let $\overline{W}_1 = g^{\overline{w}_1}$ denote the DH key generated for oracle π_i^s such that $\overline{W}_1 \in \{D_1, X_1\}$ and $\overline{W}_1$ is not generated using challenge value. Then we could rewrite the key material k_i^s of oracle π_i^s as

$$\begin{aligned} k_i^s =& me(D_2X_2, \dots, D_{n+1}X_{n+1})^{d_1+x_1} = me(D_2X_2, \dots, D_{n+1}X_{n+1})^{\overline{w}_1+w_1} \\ =& me(D_2X_2, \dots, D_{n+1}X_{n+1})^{w_1} \cdot me(D_2X_2, \dots, D_{n+1}X_{n+1})^{\overline{w}_1}. \end{aligned}$$

Since the $\overline{w}_1$ is chosen by $\mathcal{D}$ then it is able to compute the value

$$\alpha = me(D_2X_2, \dots, D_{n+1}X_{n+1})^{\overline{w}_1}.$$

For both above cases, we further consider the following disjoint event that covers all possibilities.

- Event 1: Firstly, we consider the event that every DH key tuple (D_j, X_j) for $2 \leq j \leq n+1$ received by oracle π_i^s consists of one DH key that is computed using challenge value g^μ. As all values recorded

[6]Please forget the (subscripts) positions of DH keys recorded in sid_i^s of π_i^s for the time being. We here need to differentiate the DH keys generated by oracle π_i^s with other DH keys received by π_i^s in the following modification, even though those DH keys of π_i^s might be located in different position in sid_i^s rather than the first place.

in sid_i^s are distinct, so that in each received DH key tuple (D_j, X_j) there is at most one DH key that is generated using challenge value in this event. We further let $W_j = g^{\mu r_{w_j}}$ for $2 \leq j \leq n+1$ denote the DH key received by oracle π_i^s such that $W_j \in \{D_j, X_j\}$ and W_j is generated using challenge value. And we let $\overline{W}_j = g^{\overline{w}_j}$ for $2 \leq j \leq n+1$ denote the DH key received by oracle π_i^s such that $\overline{W}_j \in \{D_j, X_j\}$ and $\overline{W}_j$ is not generated using challenge value for $2 \leq j \leq n+1$. Then in this event, $\mathcal{D}$ could compute the key material k_i^s using the value Γ, randomness r_{w_1} and the value $g^{\mu \overline{w}_j}$ extracted from $t_{\overline{W}_j}$, and n-multilinear map operations. To elaborate the simulation of k_i^s, we rewrite the $\beta := me(D_2X_2, \ldots, D_{n+1}X_{n+1})^{w_1}$ as following:

$$\begin{aligned}
\beta := & me(g^{\mu \overline{w}_2 r_{w_1}}, D_3X_3, \ldots, D_{n+1}X_{n+1}) \cdot me(W_2, D_3X_3, \ldots, D_{n+1}X_{n+1})^{w_1} \\
= & me(g^{\mu \overline{w}_2 r_{w_1}}, D_3X_3, \ldots, D_{n+1}X_{n+1}) \cdot me(W_2, g^{\mu \overline{w}_3 r_{w_1}}, D_4X_4, \ldots, D_{n+1}X_{n+1}) \\
& \cdot me(W_2, W_3, D_4X_4, \ldots, D_{n+1}X_{n+1})^{w_1} \\
= & me(g^{\mu \overline{w}_2 r_{w_1}}, D_3X_3, \ldots, D_{n+1}X_{n+1}) \cdot me(W_2, g^{\mu \overline{w}_3 r_{w_1}}, D_4X_4, \ldots, D_{n+1}X_{n+1}) \\
& \cdot me(W_2, W_3, g^{\mu \overline{w}_4 r_{w_1}}, D_5X_5, \ldots, D_{n+1}X_{n+1}) \cdot \\
& \cdots \\
& \cdot me(W_2, W_3, W_4, \ldots, W_{n-1}, g^{\mu \overline{w}_n r_{w_1}}, D_{n+1}X_{n+1}) \\
& \cdot me(W_2, W_3, W_4, \ldots, W_{n-1}, W_n, g^{\mu \overline{w}_{n+1} r_{w_1}}) \cdot me(W_2, W_3, \ldots, W_{n+1})^{w_1}.
\end{aligned}$$

The above 'expansion' of the equation is only conceptual that is consistent to the original computation of β. However this enables us to embed the challenge value Γ into the key material k_i^s and compute k_i^s without knowing w_1. More specifically we change β to β' by replacing the value $me(W_2, W_3, \ldots, W_{n+1})^{w_1}$ in above computation of β with value $\Gamma^{r_{w_1} \cdots r_{w_{n+1}}}$ and computing values $g^{\mu \overline{w}_j}$ from tag $t_{\overline{W}_j}$ as $g^{\mu \overline{w}_j} = (\frac{t_{\overline{W}_j}}{\overline{W}_j^{q(h_{\overline{W}_j})}})^{\frac{1}{p(h_{\overline{W}_j})}}$ where $t_{\overline{W}_j} \in \{t_{D_j}, t_{X_j}\}$ and $2 \leq j \leq n+1$. Eventually we compute the key material $k_i^s = \alpha \cdot \beta'$ and use it to compute the final session key of oracle π_i^s.

- Event 2: On the second, we consider the event that there exists one DH key tuple (D_j, X_j) $(2 \leq j \leq n+1)$ received by oracle π_i^s which are all not generated using challenge value g^μ. Then, in order to simulate the key material k_i^s, the jobs of $\mathcal{D}$ are only to compute $g^{\mu d_j}$ from t_{D_j} (if D_j is chosen by adversary, as otherwise $\mathcal{D}$ knows corresponding exponent d_j) as $g^{\mu d_j} := (\frac{t_{D_j}}{D_j^{q(h_{D_j})}})^{\frac{1}{p(h_{D_j})}}$ and to compute $g^{\mu x_j}$ from t_{X_j} as $g^{\mu x_j} := (\frac{t_{X_j}}{X_j^{q(h_{X_j})}})^{\frac{1}{p(h_{X_j})}}$. Let $\{\eta_l\}$ for $1 \leq l \leq n-1$ be a set of variables each of which stores distinct integer number ranging from 2 to $n+1$ except for j. Thus the key material is generated as $k_i^s = \alpha \cdot me(g^{\mu d_j r_{w_1}} g^{\mu x_j r_{w_1}}, D_{\eta_1}X_{\eta_1}, \ldots, D_{\eta_{n-1}}X_{\eta_{n-1}})$, which is consistent to original form.

In a nutshell $\mathcal{D}$ is able to simulate all session keys appropriately in terms of the tags of both ephemeral key and long-term key. If $\Gamma = me(g, \ldots, g)^{\mu^{n+1}}$ then the simulation is exactly equivalent to previous game, otherwise it equals to this game. By applying the security of nMDDH assumption, we therefore obtain that

$$\mathsf{Adv}_2 \leq \mathsf{Adv}_3 + \epsilon_{\mathsf{nMDDH}}.$$

Game G_4. In this game, we change function $\mathsf{PRF}(\widetilde{k_i^*}, \cdot)$ to a truly random function for test oracle and its partner oracles (if they exist). We make use of the fact, that the secret seed $\widetilde{k_i^*}$ of test oracle is a truly random value. If there exists a polynomial time adversary $\mathcal{A}$ can distinguish the Game G_4 from Game

G_3. Then we can construct an algorithm $\mathcal{B}$ using $\mathcal{A}$ to break the security of PRF. Exploiting the security of PRF, we have that

$$\mathsf{Adv}_3 \leq \mathsf{Adv}_4 + \epsilon_{\mathsf{PRF}}.$$

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 $\mathsf{Adv}_4 = 0$.

Sum up the probabilities from Game G_0 to Game G_4, we proved this theorem.