PAPER *Special Section on Cryptography and Information Security*

Modeling Leakage of Ephemeral Secrets in Tripartite/Group Key Exchange*

Mark MANULIS[†a)], ***Nonmember***, **Koutarou SUZUKI**[††b)], ***Member***, ***and*** **Berkant USTAOGLU**[†††c)], ***Nonmember***

SUMMARY We propose a security model, referred as g-eCK model, for group key exchange that captures essentially all non-trivial leakage of static and ephemeral secret keys of participants, i.e., group key exchange version of extended Canetti-Krawczyk (eCK) model. Moreover, we propose the first one-round tripartite key exchange (3KE) protocol secure in the g-eCK model under the gap Bilinear Diffie-Hellman (gap BDH) assumption and in the random oracle model.

key words: *group key exchange, group-oriented extended Canetti-Krawczyk model, tripartite key exchange, gap Bilinear Diffie-Hellman assumption, random oracle model*

1. Introduction

Design and analysis of Key Exchange (KE) protocols is amongst the oldest research topics in cryptography that has found its direct way into practice. Although KE was introduced back in 1976 [23], it was not until 1993 when Bellare and Rogaway [9] made the first step towards capturing the security requirements for these protocols in a formal way, by designing a security model, which influenced later developments in the design and analysis of KE protocols. Research efforts on provable security in KE protocols, in the public key setting, can roughly be classified along two-party KE (2KE), e.g. [7], [11], [19], [20], [22], [34], [35], [41], [46], and group KE (GKE), e.g. [14], [17], [26], [31], [39], [40], [47], reaching out to other KE flavors such as password-based solutions [1], [4], [8], [10], [12], multi-factor KE protocols [44], and to flexible combinations of GKE and 2KE [2]. The most standard security notion, common to all KE flavors, takes its roots in [9] and is called authenticated key exchange security, or AKE-security for short. Although AKE-security has been modeled in different flavors, for different types of adversarial capabilities and corruptions, the common idea behind this notion is to protect the secrecy of exchanged keys by means of indistinguishability of some test session key from a randomly chosen one.

Manuscript received March 23, 2012.
Manuscript revised August 6, 2012.
[†]The author is with the University of Surrey, Guildford, Surrey, GU2 7XH, United Kingdom.
[††]The author is with the NTT Information Sharing Platform Laboratories, NTT Corporation, Musashino-shi, 180-8585 Japan.
[†††]The author is with the Izmir Institute of Technology, Urla, Izmir, 35430, Turkey.
[*]A part of this work is appeared in [40].
a) E-mail: mark@manulis.eu
b) E-mail: suzuki.koutarou@lab.ntt.co.jp
c) E-mail: bustaoglu@uwaterloo.ca
DOI: 10.1587/transfun.E96.A.101

In the field of two-party KE (2KE), the extended Canetti-Krawczyk (eCK) model, which captures essentially all non-trivial leakage of static and ephemeral secret keys of participants, has been proposed [35].

In the field of group KE (GKE), the models [17], [25], which capture (restricted) leakage of static and ephemeral secret keys of participants, incorporate key compromise impersonation (KCI) attacks and forward security (FS). However, no security definition, that captures essentially all non-trivial leakage of static and ephemeral secret keys of participants, is known yet.

From practical point of view, since there are many participants in the case of GKE, the possibility of leakage of static and ephemeral secret keys increases. So, such leakage is even more important to take care of in the case of group key exchange. From theoretical point of view, it is natural to consider all non-trivial leakage of static and ephemeral secret keys of participants by removing unnecessary restrictions from the models of [17], [25]. Thus, it is worthy goal to model extended Canetti-Krawczyk (eCK) security in the group KE (GKE).

In this paper, we propose a security model, referred as g-eCK model, for group key exchange that captures essentially all non-trivial leakage of static and ephemeral secret keys of participants, i.e., group key exchange version of extended Canetti-Krawczyk (eCK) model [35].

Moreover, we propose the first one-round authenticated tripartite key exchange (3KE) protocol that is secure in the g-eCK model and is based on the Joux's one-round (unauthenticated) 3KE [28], [29]. There have been several authenticated 3KE protocols [5], [36]–[38], [45], however, none of these protocols are secure in the g-eCK model. So, we provide the proposed protocol to demonstrate that the g-eCK model can be satisfied. The proposed protocol is secure in the g-eCK model under the gap Bilinear Diffie-Hellman (gap BDH) assumption and in the random oracle model. Although the gap BDH assumption is strong, we adopt the assumption to make the protocol efficient, i.e., we can use the BDH assumption by applying the twin Diffie-Hellman technique, but it requires twice number of keys and more pairing operations than the proposed protocol.

The proposed protocol is efficient, i.e., is one-round and requires 4 shared secrets, 4 pairing operations, and 4 exponential operations (including the exponentiation for the ephemeral public key).

1.1 Related Works

Ephemeral Key Leakage. The security model proposed in this paper (and [40]), which was stated in a more general GKE setting, considers a very strong attacker, who may adaptively compromise static and ephemeral secret keys used in the protocol session (with an obvious restriction that at least one of these keys per participant must remain secret). Leakage of ephemeral secrets, typically of exponents used in the computation of ephemeral public (Diffie-Hellman) keys, is especially damaging for implicitly authenticated protocols, where for better efficiency one may desire to pre-compute ephemeral public keys off-line, in which case used secrets must be stored temporarily, resulting in a higher risk for the security of the protocol. Even if ephemeral secret keys are chosen (and erased) within the protocol session, attacks exploiting side-channels of the implementation may threaten their secrecy. In general, motivation for considering leakage of ephemeral secrets in KE protocols stems from 2KE domain, e.g. as first mentioned in [19], [34] and explicitly modeled in AKE-security definitions from [35], [46]. Various efforts towards construction of 2KE leakage-resilient protocols have been taken, e.g. [24], [33], [35], [42], [43], [46]. In general, modeling and designing ephemeral key-leakage resilient KE protocols should not be taken for granted — Cremers [21], [22] demonstrated how various technical elements of 2KE models such as the notions of session ids and partnering as well as conditions for freshness of the test session may affect the strength of AKE-security definition with ephemeral key-leakage resilience, when it comes to comparability of models and 2KE protocols. The model proposed in this paper (and [40]) is so-far the only GKE security model that focuses on ephemeral key-leakage in test sessions and has recently been applied in [47], for the analysis of a two-round explicitly authenticated ephemeral key-leakage resilient GKE protocol.

GKE models. Group key exchange (GKE) protocols are essentially the generalization of 2KE protocols to the group case. However, this generalization brings additional problems both in the design and the analysis of the protocols. The first formal model for GKE protocol was described by Bresson et al. [14] inspired by the two-party approach in [9]. Many modifications and improvements appeared thereafter, see the survey in [39]. GKE models mainly focus on the *outsider security* which is modeled through the requirement of AKE-security, e.g. [13], [14], [18], [31], as this requirement deals explicitly with the secrecy of the established keys, which becomes meaningless if the adversary is an insider. Several models, e.g. [15], [17], [25], [26], [30], also consider the optional *insider security* aiming to prevent attacks by which insiders force parties to complete either with different keys (usually modeled as MA-security) or with keys that have some biased distribution (usually modeled as contributiveness). Protection against insider attacks, however, can often be achieved through generic compilers e.g. [15], [16], [30]. Another key difference amongst GKE models is in the treatment of corruptions: earlier models, e.g. [14], [31], considered *weak corruptions* allowing the adversary to obtain users' static keys, but not their ephemeral session secrets. Later models, e.g. [17], [18], [25] assumed *strong corruptions* allowing the adversary to learn both static private keys and session specific secrets through a single query. Manulis and Bresson [17], inspired by the two-party approach in [19] refined the notion of strong corruptions in GKE allowing the adversary to obtain static keys independently from ephemeral session secrets; yet, restricting the leakage of ephemeral secrets to sessions for which the adversary does not need to distinguish the key. The reason is that GKE protocols known today become insecure if ephemeral secrets used to compute a group key leak, in other words leaking ephemeral secrets of one session affects the security of other non-partnered sessions. As a result many GKE protocols are insecure if parties for better performance pre-compute their ephemeral secrets off-line. The model from [17] was subsequently strengthened by Gorantla et. al. [25] (also more recent journal version [27]) to address key compromise impersonation attacks.

Tripartite Key Exchange. In 2000, a powerful GKE subclass of tripartite KE (3KE) emerged with the introduction of pairings to KE by Joux [28], [29]. Thanks to the bilinear property only one communication round amongst three parties is necessary in [28], [29] to compute the session key, whereby each party has to communicate only a constant amount of bits (one group element) and perform a constant number of operations (one exponentiation and pairing evaluation). The original protocol in [28] was unauthenticated and so efforts were taken to achieve protection against active attacks, yet without sacrificing the unique efficiency properties of the protocol. This goal turned out to be non-trivial. In fact, adopting traditional authentication techniques such as digital signatures, as previously applied to unauthenticated 2KE Diffie-Hellman in [19], would inherently result in at least two rounds of communication amongst 3KE participants due to the necessary use of nonces for preventing replay attacks on the signed ephemeral public keys; as is also obvious from the possible application of signature-based GKE authentication compilers from [18], [30]–[32]. 3KE protocols with at least two communication rounds have also been known in other authentication settings, e.g. using passwords [3]. Intuitively, the only way to preserve one communication round with constant bit communication complexity from [28], [29] is to resort to an implicitly authenticated solution, in which session key is derived through a binding of static (long-term) and ephemeral (session-dependent) secrets. Multiple attempts to achieve this form of authentication for 3KE, e.g. [5], [36]–[38], [45] failed (as partly demonstrated in subsequent attempts against the previous ones and also summarized and extended in [40]) and the so-far only implicitly authenticated 3KE protocol that provably fulfills this goal is the protocol proposed by Manulis, Suzuki, and Ustaoglu in this paper (and [40]).

2. Proposed g-eCK Model

We propose a security model, refered as g-eCK model, for group key exchange that captures all non-trivial patterns of leakage of static and ephemeral secret keys of participants, i.e., group key exchange version of extended Canetti-Krawczyk (eCK) model [35]. All non-trivial patterns of leakage of static and ephemeral secret keys is described by Condition 3 and Condition 4 in the definition of freshness (Definition 1) using StaticKeyReveal query and StateReveal query. This model can be seen as an extension of the strong authenticated key exchange model for two-party protocols from [41] to the group setting and it is described using the classical notations and terminology of existing models for GKE protocols, e.g. [17], [25], [27], [31].

(1) Protocol Participants and Initialization.

Let $\mathcal{U} := \{U_1, \ldots, U_N\}$ be a set of potential protocol participants and each user $U_i \in \mathcal{U}$ is assumed to hold a static private/public key pair (s_i, S_i) generated by some algorithm $Gen(1^\kappa)$ on a security parameter 1^κ during the initialization phase.

(2) Protocol Sessions and Instances.

Any subset of $\mathcal{U}$ can decide at any time to execute a new protocol session and establish a common group key. Participation of some $U \in \mathcal{U}$ in multiple sessions is modeled through an number of *instances* $\{\Pi_U^s \mid s \in [1 \ldots n], U \in \mathcal{U}\}$, i.e., the Π_U^s is the s-th session of U. Each instance is invoked via a message to U with a *partner id*† $\mathtt{pid}_U^s \subseteq \mathcal{U}$, which encompasses the identities of all the intended session participants (note that $\mathtt{pid}_U^s$ also includes U). Then, we say that U owns the instance Π_U^s. In the invoked session, Π_U^s *accepts* if the protocol execution was successful, in particular Π_U^s holds then the computed *group key* K_U^s.

(3) Session State.

During the session execution, each participating Π_U^s creates and maintains a *session id* $\mathtt{sid}_U^s$ and an associated internal state $\mathtt{state}_U^s$ which in particular is used to maintain ephemeral secrets used by Π_U^s during the protocol execution. Concretely, $\mathtt{sid}_U^s = \{(m_{1,1}, \ldots, m_{u,1}), \ldots, (m_{1,r}, \ldots, m_{u,r})\}$, where $m_{i,j}$ is the j-th outgoing message from participant U_i in the session Π_U^s and $\mathtt{pid}_U^s = \{U_1, \ldots, U_u\}$.

We say that U *owns* session $\mathtt{sid}_U^s$ if the instance Π_U^s was invoked at U. Furthermore, we assume that instances that accepted or aborted delete all information in their respective states.

(4) Partnering.

Two instances Π_U^s and $\Pi_{U_*}^t$ are called *partnered* or *matching* if 1) Π_U^s and $\Pi_{U_*}^t$ have accepted, 2) $\mathtt{sid}_U^s = \mathtt{sid}_{U_*}^t$, and 3) $\mathtt{pid}_U^s = \mathtt{pid}_{U_*}^t$.

Note also that the notion of partnering is self-inclusive in the sense that any Π_U^s is partnered with itself. If the protocol allows a user U to initiate sessions with U, then the equality $\mathtt{pid}_U^s = \mathtt{pid}_{U_*}^t$ is a multi-set equality.

(5) Adversarial Model.

The adversary $\mathcal{A}$, modeled as a PPT machine, can schedule the protocol execution and mount own attacks via the following queries:

- AddUser(U, S_U): This query allows $\mathcal{A}$ to introduce new users. In response, if $U \notin \mathcal{U}$ (due to the uniqueness of identities) then U with the static public key S_U is added to $\mathcal{U}$; Note that $\mathcal{A}$ is not required to prove the possession of the corresponding secret key s_U††.
- Send(Π_U^s, m): With this query, $\mathcal{A}$ can deliver a message m to Π_U^s whereby U denotes the identity of its sender. $\mathcal{A}$ is then given the protocol message generated by Π_U^s in response to m (the output may also be empty if m is not required or if Π_U^s accepts). A special invocation query of the form Send$(U, (\text{'start'}, U_1, \ldots, U_n))$ with $U \in \{U_1, \ldots, U_n\}$ creates a new instance Π_U^s with $\mathtt{pid}_U^s = \{U_1, \ldots, U_n\}$ and provides $\mathcal{A}$ with the first protocol message.
- SessionKeyReveal(Π_U^s): This query models the leakage of session group keys and provides $\mathcal{A}$ with K_U^s. It is answered only if Π_U^s has accepted.
- StaticKeyReveal(U): This query provides $\mathcal{A}$ with the static private key s_U.
- StateReveal(Π_U^s): $\mathcal{A}$ is given the ephemeral secret information contained in $\mathtt{state}_U^s$ at the moment the query is asked. Note that the protocol specifies what the state contains.
- Test(Π_U^s): This query models the indistinguishability of the session group key according to the privately flipped bit τ. If $\tau = 0$ then $\mathcal{A}$ is given a random session group key, whereas if $\tau = 1$ the real K_U^s. The query can be queried only once and requires that Π_U^s has accepted.

Observe that via the Send(Π_U^s, m) query the adversary can control delivery of messages between instances, i.e., adversary can drop or modify messages. Therefore, the model captures active attacks such as replay and man in the middle attacks.

(6) Correctness.

A GKE protocol is said to be *correct* if in the presence of a benign††† adversary all instances invoked for the same protocol session accept with the same session group key.

(7) Freshness.

The classical notion of freshness of some instance Π_U^s is tra-

†Invocation should include the order of users and perhaps some additional information.

††In our security argument, we will only assume that S_U chosen by $\mathcal{A}$ must come from the ephemeral public key space, e.g., element of $\mathbb{G}$.

†††Benign adversary executes an instance of the protocol and faithfully delivers messages without any modification.

ditionally used to define the goal of AKE-security by specifying the conditions for the $\mathsf{Test}(\Pi_U^s)$ query. For example, the model in [31] defines an instance Π_U^s that has accepted as fresh if none of the following is true: (1) at some point, $\mathcal{A}$ asked $\mathsf{SessionKeyReveal}$ to Π_U^s or to any of its partnered instances; or (2) a query $\mathsf{StaticKeyReveal}(U_*)$ with $U_* \in \mathtt{pid}_U^s$ was asked before a Send query to Π_U^s or any of its partnered instances.

Unfortunately, these restrictions are not sufficient for our purpose since Π_U^s becomes immediately unfresh if the adversary gets involved into the protocol execution via a Send query after having learned the static key s_{U_*} of some user U_* those instance participates in the same session as Π_U^s.

The recent model in [17] defines freshness using the additional $\mathsf{AddUser}$ and $\mathsf{StateReveal}$ queries as follows. According to [17], an instance Π_U^s that has accepted is fresh if none of the following is true: (1) $\mathcal{A}$ queried $\mathsf{AddUser}(U_M, S_{U_M})$ with some $U_* \in \mathtt{pid}_U^s$; or (2) at some point, $\mathcal{A}$ asked $\mathsf{SessionKeyReveal}$ to Π_U^s or any of its partnered instances; or (3) a query $\mathsf{StaticKeyReveal}(U_*)$ with $U_* \in \mathtt{pid}_U^s$ was asked before a Send query to Π_U^s or any of its partnered instances; or (4) $\mathcal{A}$ queried $\mathsf{StateReveal}$ to Π_U^s or any of its partnered instances at some point after their invocation but before their acceptance.

Although this definition is already stronger than the one in [31] it is still insufficient for the main reason that it excludes the leakage of ephemeral secrets of instances in the period between the protocol invocation and acceptance. Also this definition of freshness does not model key compromise impersonation attacks.

The recent update of the freshness notion in [25], [27] addressed the lack of key compromise impersonation resilience. In particular, it modifies the above condition (3) by requiring that if there exists an instance $\Pi_{U_*}^t$ which is partnered with Π_U^s and $\mathcal{A}$ asked $\mathsf{StaticKeyReveal}(U_*)$ then all messages sent by $\mathcal{A}$ to Π_U^s on behalf of $\Pi_{U_*}^t$ must come from $\Pi_{U_*}^t$ intended for Π_U^s. This condition should allow the adversary to obtain static private keys of users prior to the execution of the attacked session while requiring its benign behavior with respect to the corrupted user during the attack.

Yet, this freshness requirement still prevents the adversary from obtaining ephemeral secrets of participants during the attacked session. What is needed is a freshness condition that would allow the adversary to corrupt users and reveal the ephemeral secrets used by their instances in the attacked session at will for the only exception that it does not obtain both the static key s_{U_*} *and* the ephemeral secrets used by the corresponding instance of U_*; otherwise security can no longer be guaranteed. In the following we define freshness taking into account all the previously mentioned problems.

Definition 1: An accepted instance Π_U^s is fresh if none of the following is true:

1. $\mathcal{A}$ queried $\mathsf{AddUser}(U_*, S_{U_*})$ with some $U_* \in \mathtt{pid}_U^s$; or
2. $\mathcal{A}$ queried $\mathsf{SessionKeyReveal}$ to Π_U^s or any of its accepted partnered instances; or
3. $\mathcal{A}$ queried both $\mathsf{StaticKeyReveal}(U_*)$ with $U_* \in \mathtt{pid}_U^s$ and $\mathsf{StateReveal}(\Pi_{U_*}^t)$ for some instance $\Pi_{U_*}^t$ partnered with Π_U^s; or
4. $\mathcal{A}$ queried $\mathsf{StaticKeyReveal}(U_*)$ with $U_* \in \mathtt{pid}_U^s$ and there exists no instance $\Pi_{U_*}^t$ partnered with Π_U^s.

Note that since $U \in \mathtt{pid}_U^s$ and since the notion of partnering is self-inclusive Condition 3 prevents the simultaneous corruption of static and ephemeral secrets for the corresponding instance Π_U^s as well. In case when users are allowed to own two partnering instances i.e., they can initiate protocols with themselves the last condition should be modified to say that the number of instances U equals the number of times U appears in $\mathtt{pid}_U^s$. Note also that the above definition captures key compromise impersonation resilience through Condition 4: $\mathcal{A}$ is allowed to corrupt participants of the test session in advance but then must ensure that instances of such participants have been honestly participating in the test session. In this way we exclude the trivial break of security where $\mathcal{A}$ reveals static keys of users prior to the test session and then actively impersonates those users during the session. On the other hand, as long as $\mathcal{A}$ remains benign with respect to such users their instances will still be considered as fresh.

(8) g-eCK Security.

We are ready to generalize the strong AKE-security definition from [35], [41] to a group setting.

Definition 2: Let P be a correct GKE protocol and τ be a uniformly chosen bit. We define the adversarial game $\mathsf{Game}_{\mathcal{A},\mathsf{P}}^{\text{ake-}\tau}(\kappa)$ as follows: after initialization, $\mathcal{A}$ interacts with instances via queries. At some point, $\mathcal{A}$ queries $\mathsf{Test}(\Pi_U^s)$, and continues own interaction with the instances until it outputs a bit τ'. If Π_U^s to which the Test query was asked is *fresh* at the end of the experiment then we set $\mathsf{Game}_{\mathcal{A},\mathsf{P}}^{\text{ake-}\tau}(\kappa) = \tau'$. We define

$$\mathsf{Adv}_{\mathcal{A},\mathsf{P}}^{\text{ake}}(\kappa) = \left|2\Pr[\tau = \tau'] - 1\right|$$

and denote with $\mathsf{Adv}_{\mathsf{P}}^{\text{ake}}(\kappa)$ the maximum of the advantage over all PPT adversaries $\mathcal{A}$. We say that a GKE protocol P provides g-eCK security if advantage $\mathsf{Adv}_{\mathsf{P}}^{\text{ake}}(\kappa)$ is negligible.

3. Proposed g-eCK Secure Tripartite Protocol

In this section, we propose a one-round tripartite key exchange (3KE) protocol secure in the g-eCK model under the gap Bilinear Diffie-Hellman (gap BDH) assumption and in the random oracle model.

3.1 Proposed 3KE Protocol

We propose a one-round 3KE protocol Π and prove in Theorem 4 that the proposed 3KE protocol Π is g-eCK secure under the gap BDH assumption in the random oracle model.

Let κ be the security parameter. Let $\mathbb{G}$ and $\mathbb{G}_T$ be cyclic groups of prime order q. Let $e : \mathbb{G} \times \mathbb{G} \mapsto \mathbb{G}_T$ be a non-degenerate bilinear map, called pairing, from group $\mathbb{G} \times \mathbb{G}$ to group $\mathbb{G}_T$. Let g and $g_T = e(g, g)$ be a generator of $\mathbb{G}$ and $\mathbb{G}_T$, respectively. Let $H : \{0,1\}^* \rightarrow \{0,1\}^\kappa$ be cryptographic hash function modeled as a random oracle. Let P be the protocol identifier of the 3KE protocol Π.

Let $D, E, F \in \mathbb{Z}_q$ be constants s.t. $D, E, F \neq 0, 1$ and published as the system parameter.

For a user U_A, we set U_A's static and ephemeral keys $A_0 = g^{a_0}$ and $A_1 = g^{a_1}$, respectively, and the lowercase letters are the private keys.

In the description, users U_A with static key A_0, U_B with static key B_0, and U_C with static key C_0 communicate with each other, and compute the session key by the following one-round 3KE protocol.

1. U_A selects a random ephemeral private key $a_1 \in_U \mathbb{Z}_q$, computes the ephemeral public key $A_1 = g^{a_1}$, stores ephemeral private key a_1 as state information, and broadcasts $(\mathsf{P}, (U_A, U_B, U_C), U_A, A_1)$ to U_B and U_C.
2. U_B selects a random ephemeral private key $b_1 \in_U \mathbb{Z}_q$, computes the ephemeral public key $B_1 = g^{b_1}$, stores ephemeral private key b_1 as state information, and broadcasts $(\mathsf{P}, (U_A, U_B, U_C), U_B, B_1)$ to U_C and U_A.
3. U_C selects a random ephemeral private key $c_1 \in_U \mathbb{Z}_q$, computes the ephemeral public key $C_1 = g^{c_1}$, stores ephemeral private key c_1 as state information, and broadcasts $(\mathsf{P}, (U_A, U_B, U_C), U_C, C_1)$ to U_A and U_B.
4. Upon receiving $(\mathsf{P}, (U_A, U_B, U_C), U_B, B_1)$ and $(\mathsf{P}, (U_A, U_B, U_C), U_C, C_1)$, U_A verifies $B_1, C_1 \in \mathbb{G}$, computes m shared secrets
$$\sigma_1 = e(B_0B_1, C_0C_1)^{a_0+Da_1}$$
$$\sigma_2 = e(B_0B_1^E, C_0C_1)^{a_0+a_1}$$
$$\sigma_3 = e(B_0B_1, C_0C_1^F)^{a_0+a_1}$$
$$\sigma_4 = e(B_0B_1^E, C_0C_1^F)^{a_0+Da_1}$$
obtains the session key $K = H(\sigma_1, \sigma_2, \sigma_3, \sigma_4, \mathsf{P}, U_A, A_0, A_1, U_B, B_0, B_1, U_C, C_0, C_1)$, and completes the session.
5. Upon receiving $(\mathsf{P}, (U_A, U_B, U_C), U_C, C_1)$ and $(\mathsf{P}, (U_A, U_B, U_C), U_A, A_1)$, U_B verifies $C_1, A_1 \in \mathbb{G}$, computes m shared secrets
$$\sigma_1 = e(C_0C_1, A_0A_1^D)^{b_0+b_1}$$
$$\sigma_2 = e(C_0C_1, A_0A_1)^{b_0+Eb_1}$$
$$\sigma_3 = e(C_0C_1^F, A_0A_1)^{b_0+b_1}$$
$$\sigma_4 = e(C_0C_1^F, A_0A_1^D)^{b_0+Eb_1}$$
obtains the session key $K = H(\sigma_1, \sigma_2, \sigma_3, \sigma_4, \mathsf{P}, U_A, A_0, A_1, U_B, B_0, B_1, U_C, C_0, C_1)$, and completes the session.
6. Upon receiving $(\mathsf{P}, (U_A, U_B, U_C), U_A, A_1)$ and $(\mathsf{P}, (U_A, U_B, U_C), U_B, B_1)$, U_C verifies $A_1, B_1 \in \mathbb{G}$, computes m shared secrets
$$\sigma_1 = e(A_0A_1^D, B_0B_1)^{c_0+c_1}$$
$$\sigma_2 = e(A_0A_1, B_0B_1^E)^{c_0+c_1}$$
$$\sigma_3 = e(A_0A_1, B_0B_1)^{c_0+Fc_1}$$
$$\sigma_4 = e(A_0A_1^D, B_0B_1^E)^{c_0+Fc_1}$$
obtains the session key $K = H(\sigma_1, \sigma_2, \sigma_3, \sigma_4, \mathsf{P}, U_A, A_0, A_1, U_B, B_0, B_1, U_C, C_0, C_1)$, and completes the session.

All users U_A, U_B, and U_C compute the same shared secrets
$$\sigma_1 = g_T^{(a_0+Da_1)(b_0+b_1)(c_0+c_1)}$$
$$\sigma_2 = g_T^{(a_0+a_1)(b_0+Eb_1)(c_0+c_1)}$$
$$\sigma_3 = g_T^{(a_0+a_1)(b_0+b_1)(c_0+Fc_1)}$$
$$\sigma_4 = g_T^{(a_0+Da_1)(b_0+Eb_1)(c_0+Fc_1)}$$
and so compute the same session key K.

The 3KE protocol corresponding to Example 3 requires 4 shared secrets, 4 pairing operations, and 4 exponential operations (including the exponentiation for the ephemeral public key).

Instead of that constants $D, E, F \in \mathbb{Z}_q$ is provided as a part of the system parameter, constants $D, E, F \in \mathbb{Z}_q$ can be generated on the fly by $D = H'(A_1), E = H'(B_1), F = H'(C_1)$, where H' is a hash function and A_1, B_1, C_1 are the ephemeral keys of parties U_A, U_B, U_C and $D, E, F \neq 0, 1$ holds with overwhelming probability.

3.2 Security

For the security of the proposed protocol, we need the gap Bilinear Diffie-Hellman (gap BDH) assumption[†] [6] described below. Let BCDH : $\mathbb{G}^3 \rightarrow \mathbb{G}_T$ be a BDH function $\mathrm{BCDH}(g^u, g^v, g^w) = e(g, g)^{uvw}$, and BDDH : $\mathbb{G}^3 \times \mathbb{G}_T \rightarrow \{0, 1\}$ be a predicate which takes an input $(g^u, g^v, g^w, e(g, g)^x)$ and returns bit 1 if $uvw = x \bmod q$ and bit 0 otherwise. An adversary $\mathcal{A}$ is given input $g^u, g^v, g^w \in_U \mathbb{G}$ selected uniformly random and oracle access to $\mathrm{BDDH}(\cdot,\cdot,\cdot,\cdot)$ oracle, and tries to compute $\mathrm{BCDH}(g^u, g^v, g^w)$. For adversary $\mathcal{A}$, we define advantage
$$Adv^{\mathrm{gapBDH}}(\mathcal{A}) = \Pr[g^u, g^v, g^w \in_U \mathbb{G},$$
$$\mathcal{A}^{\mathrm{BDDH}(\cdot,\cdot,\cdot,\cdot)}(g^u, g^v, g^w) = \mathrm{BCDH}(g^u, g^v, g^w)],$$
where the probability is taken over the choices of g^u, g^v, g^w and $\mathcal{A}$'s random tape.

Definition 3 (gap BDH assumption): We say that $\mathbb{G}$ and $\mathbb{G}_T$ satisfy the gap BDH assumption if, for all polynomial-time adversaries $\mathcal{A}$, advantage Adv^{gapBDH} $(\mathcal{A})$ is negligible in security parameter κ.

We now prove security of the proposed 3KE protocol

[†]In pairing group, we can use the pairing as DDH oracle, but we do not have BDDH oracle. So we need gap BDH assumption.

in the g-eCK model.

Theorem 4: If $\mathbb{G}$ and $\mathbb{G}_T$ are groups where the gap BDH assumption holds and H is a random oracle, the proposed 3KE protocol Π is secure in the g-eCK model.

The proof of Theorem 4 is provided in Appendix, we provide a intuitive discussion here.

Proof: (Sketch) All users U_A, U_B, and U_C can compute the same shared secrets as shown above and so can compute the same sassion key K.

The gap BDH solver $\mathcal{S}$ extracts the answer g_T^{uvw} of an instance ($U = g^u, V = g^v, W = g^w$) of the gap BDH problem using adversary $\mathcal{A}$. For instance, we assume the case that test session $\mathtt{sid}^*$, owner of which is user U_A, has no partnered sessions $\overline{\mathtt{sid}^*}$, owners of which are users U_B and U_C, adversary $\mathcal{A}$ is given a_0, and adversary $\mathcal{A}$ does not obtain a_1, b_0 and c_0 from the condition of the freshness. In this case, solver $\mathcal{S}$ can perfectly simulate StaticKeyReveal query by selecting random a_0 and setting $A_0 = g^{a_0}$, and solver $\mathcal{S}$ embeds the instance as $A_1 = U$ $(= g^u)$, $B_0 = V$ $(= g^v)$ and $C_0 = W$ $(= g^w)$ to extract g_T^{uvw} from the shared secrets

$$\sigma_1 = g_T^{(a_0+Da_1)(b_0+b_1)(c_0+c_1)},$$

$$\sigma_2 = g_T^{(a_0+a_1)(b_0+Eb_1)(c_0+c_1)},$$

$$\sigma_3 = g_T^{(a_0+a_1)(b_0+b_1)(c_0+Fc_1)},$$

$$\sigma_4 = g_T^{(a_0+Da_1)(b_0+Eb_1)(c_0+Fc_1)}.$$

The solver $\mathcal{S}$ can extract the answer of the gap BDH instance as follows. By eliminating the terms including a_0 using the knowledge of a_0, solver $\mathcal{S}$ can obtain

$$\sigma_1' = (\sigma_1 e(B_0B_1, C_0C_1)^{-a_0})^{1/D} = g_T^{a_1(b_0+b_1)(c_0+c_1)},$$

$$\sigma_2' = (\sigma_2 e(B_0B_1^E, C_0C_1)^{-a_0}) = g_T^{a_1(b_0+Eb_1)(c_0+c_1)},$$

$$\sigma_3' = (\sigma_3 e(B_0B_1, C_0C_1^F)^{-a_0}) = g_T^{a_1(b_0+b_1)(c_0+Fc_1)},$$

$$\sigma_4' = (\sigma_4 e(B_0B_1^E, C_0C_1^F)^{-a_0})^{1/D} = g_T^{a_1(b_0+Eb_1)(c_0+Fc_1)}.$$

By using these 4 linearly independent terms, solver $\mathcal{S}$ can extract the answer $g_T^{a_1b_0c_0} = g_T^{uvw}$ as

$$((\sigma_1'^E \sigma_2'^{-1})^F (\sigma_3'^E \sigma_4'^{-1})^{-1})^{1/(E-1)(F-1)} = g_T^{a_1b_0c_0}.$$

Notice that, in other cases, the solver $\mathcal{S}$ can eliminat the terms including the specific secret key using the knowledge of the secret key, can obtain 4 linearly independent terms, and can extract the answer g_T^{uvw} by solving the linear equation.

The solver $\mathcal{S}$ can check whether the shared secrets are correctly formed w.r.t. static and ephemeral public keys, and can simulate H and SessionKeyReveal queries consistently. More precisely, in the simulation of the $H(\sigma_1, \sigma_2, \sigma_3, \sigma_4, \mathsf{P}, U_A, A_0, A_1, U_B, B_0, B_1, U_C, C_0, C_1)$ query, solver $\mathcal{S}$ needs to check that the shared secrets $\sigma_1, \sigma_2, \sigma_3, \sigma_4$ are correctly formed, and if so return session key K being consistent with the previously answered SessionKeyReveal(P, U_X, U_A, A_0, A_1, U_B, B_0, B_1, U_C, C_0, C_1) ($X = A, B, C$) queries. The solver $\mathcal{S}$ can check if shared secrets $\sigma_1, \sigma_2, \sigma_3, \sigma_4$ are correctly formed w.r.t. the static and ephemeral public keys by asking BDDH oracle

$$\mathrm{BDDH}(A_0A_1^D, B_0B_1, C_0C_1, \sigma_1) = 1,$$

$$\mathrm{BDDH}(A_0A_1, B_0B_1^E, C_0C_1, \sigma_2) = 1,$$

$$\mathrm{BDDH}(A_0A_1, B_0B_1, C_0C_1^F, \sigma_3) = 1,$$

$$\mathrm{BDDH}(A_0A_1^D, B_0B_1^E, C_0C_1^F, \sigma_4) = 1,$$

and this implies $\sigma_1, \sigma_2, \sigma_3, \sigma_4$ are correctly formed.

Notice that, in other cases, the solver $\mathcal{S}$ can check if shared secrets $\sigma_1, \sigma_2, \sigma_3, \sigma_4$ are correctly formed or not by the same procedure.

□

4. Conclusion

We proposed a security model, referred as g-eCK model, for group key exchange that captures essentially all non-trivial leakage of static and ephemeral secret keys of participants, i.e., group key exchange version of extended Canetti-Krawczyk (eCK) model [35]. Moreover, we proposed the first one-round tripartite key exchange (3KE) protocol secure in the g-eCK model under the gap Bilinear Diffie-Hellman (gap BDH) assumption and in the random oracle model. The proposed protocol is efficient, i.e., is one-round and requires 4 shared secrets, 4 pairing operations, and 4 exponential operations (including the exponentiation for the ephemeral public key).

References

[1] M. Abdalla, J.-M. Bohli, M.I.G. Vasco, and R. Steinwandt, "(Password) authenticated key establishment: From 2-party to group," TCC 2007, LNCS, vol.4392, pp.499–514, Springer, 2007.

[2] M. Abdalla, C. Chevalier, M. Manulis, and D. Pointcheval, "Flexible group key exchange with on-demand computation of subgroup keys," AFRICACRYPT 2010, LNCS, vol.6055, pp.351–368, Springer, May 2010.

[3] M. Abdalla, P.-A. Fouque, and D. Pointcheval, "Password-based authenticated key exchange in the three-party setting," PKC 2005, LNCS, vol.3386, pp.65–84, Springer, 2005.

[4] M. Abdalla and D. Pointcheval, "A scalable password-based group key exchange protocol in the standard model," ASIACRYPT 2006, LNCS, vol.4284, pp.332–347, Springer, 2006.

[5] S.S. Al-Riyami and K.G. Paterson, "Tripartite authenticated key agreement protocols from pairings," 9th IMA International Conference, LNCS, vol.2898, pp.332–359, 2003.

[6] J. Baek, R. Safavi-Naini, and W. Susilo, "Efficient multi-receiver identity-based encryption and its application to broadcast encryption," PKC 2005, pp.380–397, 2005.

[7] M. Bellare, R. Canetti, and H. Krawczyk, "A modular approach to the design and analysis of authentication and key exchange protocols," STOC 1998, pp.419–428, ACM, 1998.

[8] M. Bellare, D. Pointcheval, and P. Rogaway, "Authenticated key exchange secure against dictionary attacks," EUROCRYPT 2000, LNCS, vol.1807, pp.139–155, Springer, 2000.

[9] M. Bellare and P. Rogaway, "Entity authentication and key distribution," Advances in Cryptology — CRYPTO'93, LNCS, vol.773, pp.232–249, 1993.

[10] S.M. Bellovin and M. Merritt, "Augmented encrypted key exchange: A password-based protocol secure against dictionary attacks and password file compromise," ACM CCS 1993, pp.244–250, ACM, 1993.
[11] S. Blake-Wilson, D. Johnson, and A. Menezes, "Key agreement protocols and their security analysis," 6th IMA International Conference, LNCS, vol.1355, pp.30–45, 1997.
[12] V. Boyko, P.D. MacKenzie, and S. Patel, "Provably secure password-authenticated key exchange using Diffie-Hellman," EUROCRYPT 2000, LNCS, vol.1807, pp.156–171, Springer, 2000.
[13] E. Bresson, O. Chevassut, and D. Pointcheval, "Dynamic group Diffie-Hellman key exchange under standard assumptions," Advances in Cryptology — EUROCRYPT'02, Lect. Notes Comput. Scie., vol.2332, pp.321–336, Springer, 2002.
[14] E. Bresson, O. Chevassut, D. Pointcheval, and J.-J. Quisquater, "Provably authenticated group Diffie-Hellman key exchange," Proc. 8th ACM Conference on Computer and Communications Security (CCS'01), pp.255–264, ACM Press, 2001.
[15] E. Bresson and M. Manulis, "Malicious participants in group key exchange: Key control and contributiveness in the shadow of trust," Proc. 4th Autonomic and Trusted Computing Conference (ATC 2007), Lect. Notes Comput. Sci., vol.4610, pp.395–409, Springer-Verlag, 2007.
[16] E. Bresson and M. Manulis, "Contributory group key exchange in the presence of malicious participants," IET Information Security, vol.2, no.3, pp.85–93, 2008.
[17] E. Bresson and M. Manulis, "Securing group key exchange against strong corruptions," Proc. ACM Symposium on Information, Computer and Communications Security (ASIACCS'08), pp.249–260, ACM Press, 2008. Full version in Intl. J. Applied Cryptography in 2008.
[18] E. Bresson, M. Manulis, and J. Schwenk, "On security models and compilers for group key exchange protocols," Proc. 2nd International Workshop on Security (IWSEC 2007), Lect. Notes Comput, Sci., vol.4752, pp.292–307, Springer-Verlag, Oct. 2007.
[19] R. Canetti and H. Krawczyk, "Analysis of key-exchange protocols and their use for building secure channels," Advances in Cryptology — EUROCRYPT'01, LNCS, vol.2045, pp.453–474, 2001.
[20] K.-K.R. Choo, "Secure key establishment," Advances in Information Security, vol.41, Springer, 2009.
[21] C. Cremers, "Session-state reveal is stronger than ephemeral key reveal: Attacking the NAXOS key exchange protocol," M. Abdalla, D. Pointcheval, P.-A. Fouque, and D. Vergnaud, eds., Applied Cryptography and Network Security, 7th International Conference, ACNS 2009, LNCS, vol.5536, pp.20–33, Springer Verlag, June 2009.
[22] C.J. Cremers, "Examining indistinguishability-based security models for key exchange protocols: The case of CK, CK-HMQV, and eCK," ASIACCS 2011, pp.80–91, ACM, New York, NY, USA, 2011.
[23] W. Diffie and M.E. Hellman, "New directions in cryptography," IEEE Trans. Inf. Theory, vol.IT-22, no.6, pp.644–654, Nov. 1976.
[24] A. Fujioka and K. Suzuki, "Designing efficient authenticated key exchange resilient to leakage of ephemeral secret keys," CT-RSA, LNCS, vol.6558, pp.121–141, 2011.
[25] M.C. Gorantla, C. Boyd, and J.M. González-Nieto, "Modeling key compromise impersonation attacks on group key exchange protocols," Public Key Cryptography — PKC 2009, LNCS, vol.5443, pp.105–123, 2009.
[26] M.C. Gorantla, C. Boyd, and J.M. González-Nieto, "Universally composable contributory group key exchange," Proc. 4th International Symposium on Information, Computer, and Communications Security (ASIACCS'09), pp.146–156, ACM, 2009.
[27] M.C. Gorantla, C. Boyd, J.M. González-Nieto, and M. Manulis, "Modeling key compromise impersonation attacks on group key exchange protocols," ACM Trans. Inf. Syst. Secur., vol.14, no.4, p.28, 2011.
[28] A. Joux, "A one round protocol for tripartite Diffie-Hellman," in W. Bosma, ed., Algorithmic Number Theory 4th International Symposium, ANTS-IV, Proceedings, LNCS, vol.1838, pp.385–393, Springer, 2000.
[29] A. Joux, "A one round protocol for tripartite Diffie-Hellman," J. Cryptology, vol.17, no.4, pp.263–276, 2004.
[30] J. Katz and J.S. Shin, "Modeling insider attacks on group key-exchange protocols," Proc. 12th ACM Conference on Computer and Communications Security (CCS'05), pp.180–189, ACM Press, 2005.
[31] J. Katz and M. Yung, "Scalable protocols for authenticated group key exchange," D. Boneh, ed., Advances in Cryptology — CRYPTO 2003, LNCS, vol.2729, pp.100–125, Springer, 2003.
[32] J. Katz and M. Yung, "Scalable protocols for authenticated group key exchange," J. Cryptology, vol.20, no.1, pp.85–113, 2007.
[33] M. Kim, A. Fujioka, and B. Ustaoglu, "Strongly secure authenticated key exchange without NAXOS' approach," IWSEC 2009, LNCS, vol.5824, pp.174–191, Springer, 2009.
[34] H. Krawczyk, "HMQV: A high-performance secure Diffie-Hellman protocol," R. Cramer, ed., Advances in Cryptology — CRYPTO 2005, LNCS, vol.3621, pp.546–566, Springer Verlag, 2005.
[35] B. LaMacchia, K. Lauter, and A. Mityagin, "Stronger security of authenticated key exchange," Provable Security: First International Conference, ProvSec 2007, LNCS, vol.4784, pp.1–16, 2007.
[36] M.-H. Lim, S. Lee, and H. Lee, "Cryptanalysis on improved one-round Lin-Li's tripartite key agreement protocol," Cryptology ePrint Archive, Report 2007/411.
[37] M.-H. Lim, S. Lee, Y. Park, and H. Lee, "An enhanced one-round pairing-based tripartite authenticated key agreement protocol," O. Gervasi and M.L. Gavrilova, eds., Computational Science and Its Applications — ICCSA 2007, LNCS, vol.4406, pp.503–513, Springer, 2007.
[38] C.-H. Lin and H.-H. Lin, "Secure one-round tripartite authenticated key agreement protocol from Weil pairing," Y. Shibata and T.K. Shih, eds., 19th International Conference on Advanced Information Networking and Applications — AINA05, vol.2, pp.135–138, IEEE, 2005.
[39] M. Manulis, "Survey on security requirements and models for group key exchange," Technical Report 2006/02, Horst-Görtz Institute, Network and Data Security Group, Jan. 2008.
[40] M. Manulis, K. Suzuki, and B. Ustaoglu, "Modeling leakage of ephemeral secrets in tripartite/group key exchange," ICISC 2009, LNCS, vol.5984, pp.16–33, 2010.
[41] A. Menezes and B. Ustaoglu, "Comparing the pre- and post-specified peer models for key agreement," Information Security and Privacy — ACISP 2008, LNCS, vol.5107, pp.53–68, Springer, 2008.
[42] D. Moriyama and T. Okamoto, "An eCK-secure authenticated key exchange protocol without random oracles," ProvSec, LNCS, vol.5848, pp.154–167, Springer, 2009.
[43] T. Okamoto, "Authenticated key exchange and key encapsulation in the standard model," ASIACRYPT 2007, LNCS, vol.4833, pp.474–484, Springer, 2007.
[44] D. Pointcheval and S. Zimmer, "Multi-factor authenticated key exchange," ACNS 2008, LNCS, vol.5037, pp.277–295, 2008.
[45] K. Shim, "Efficient one round tripartite authenticated key agreement protocol from Weil pairing," IET Electronics Letters, vol.39, no.2, pp.208–209, 2003.
[46] B. Ustaoglu, "Comparing *SessionStateReveal* and *EphemeralKeyReveal* for Diffie-Hellman protocols," ProvSec09, 2009.
[47] J. Zhao, D. Gu, and M.C. Gorantla, "Stronger security model of group key agreement," ASIACCS 2011, pp.435–440, ACM, 2011.

Appendix: Proof of Theorem 4

In this section, we provide the proof of Theorem 4. We need the gap BDH(Bilinear Diffie-Hellman) assumption, where one tries to compute BCDH(U, V, W) accessing the BDDH oracle. Here, we denote BCDH(g^u, g^v, g^w) $= e(P, P)^{uvw}$, and the BDDH oracle on input $(g^u, g^v, g^w, e(g, g)^x)$ returns the bit 1 if $uvw = x$ and the bit 0 otherwise. We also need two variants of gap BDH assumption where one tries to compute BCDH(U, U, U) or BCDH(U, U, W) instead of BCDH(U, V, W). We call the first/second variant as the cubic/square gap BDH assumption, respectively. These two variants are equivalent to the gap BCDH assumption as follows. Given a challenge U of the cubic gap BDH assumption, one sets $V = U^s, W = U^t$ for random integers $s, t \in_R [1, p-1]$, and then can compute BCDH$(U, V, W)^{1/st}$ = BCDH(U, U, U). Given a challenge U, V, W of the gap BDH assumption, one sets $U_1 = UVW = g^{u+v+w}$, $U_2 = UVW^{-1} = g^{u+v-w}$, $U_3 = UV^{-1}W = g^{u-v+w}$, $U_4 = UV^{-1}W^{-1} = g^{u-v-w}$, and then can compute BCDH(U, V, W) from BCDH(U_i, U_i, U_i) $i = 1, \ldots, 4$. One can show the equivalence of the square gap BDH assumption similarly.

Let κ denote the security parameter, and let $\mathcal{A}$ be a polynomially (in κ) bounded adversary. We assume that $\mathcal{A}$ succeeds in an environment with n users $\{U_i\}$, activates at most s instances $\{\Pi^j_{U_i}\}$ within a user U_i. We use $\mathcal{A}$ to construct a gap BDH solver $\mathcal{S}$ that succeeds with non-negligible probability. The adversary $\mathcal{A}$ is said to be successful with non-negligible probability if $\mathcal{A}$ wins the distinguishing game with probability $1/2 + p(\kappa)$, where $p(\kappa)$ is non-negligible, and the event M denotes a successful $\mathcal{A}$.

Let $\Pi^t_{U_A}$ be the test instance owned by user U_A with session id $\mathtt{sid}^t = (\mathsf{P}, U_A, U_A, A_0, A_1, U_B, B_0, B_1, U_C, C_0, C_1)$. Let Π be any completed instance owned by an honest user with session id $\mathtt{sid}$ such that $\mathtt{sid} \neq \mathtt{sid}^t$. Let H^* be the event that $\mathcal{A}$ queries $(Z_1, \ldots, Z_m, \mathtt{sid}^t)$ to H, where $Z_1, \ldots, Z_4$ are correctly formed. Let $\overline{H^*}$ be the complement of event H^*. Since $\mathtt{sid}$ and $\mathtt{sid}^t$ are distinct, the inputs to the key derivation function H are different for $\mathtt{sid}$ and $\mathtt{sid}^t$. Since H is a random oracle, $\mathcal{A}$ cannot obtain any information about the session key of test instance $\Pi^t_{U_A}$ from the session key of instance Π. Hence $\Pr(M \wedge \overline{H^*}) \leq 1/2$ and $\Pr(M) = \Pr(M \wedge H^*) + \Pr(M \wedge \overline{H^*}) \leq \Pr(M \wedge H^*) + 1/2$, and we have $\Pr(M \wedge H^*) \geq p(\kappa)$. Henceforth the event $M \wedge H^*$ is denoted by M^*.

We will consider the not exclusive classification of all possible events in the following tables. In the tables and hereafter, we denote by $(A_0 = A, A_1 = X), (B_0 = B, B_1 = Y), (C_0 = C, C_1 = Z)$ the static and ephemeral public keys of users U_A, U_B, U_C in the test session $\mathtt{sid}^t$. Events can be classified not exclusively as in Table A·1 when A, B, C are distinct, as in Table A·2 when $A = B \neq C$, as in Table A·3 when $A = C \neq B$, as in Table A·4 when $A \neq B = C$, and as in Table A·5 when $A = B = C$. Since the classification covers all possible events, at least one event $E_{xy} \wedge M^*$ in the tables occurs with non-negligible probability if event M^* occurs with non-negligible probability. We will investigate each of these events in the following subsections, we provide detailed description for event $E_{1a} \wedge M^*$, which is the most difficult case, and outline for other events.

A.1 Event $E_{1a} \wedge M^*$

A.1.1 Setup

The algorithm $\mathcal{S}$ begins by establishing n honest users that are assigned random static key pairs. $\mathcal{S}$ embed instance (U, V, W) of gap BDH problem as follows. $\mathcal{S}$ randomly selects three users U_A, U_B, U_C and integer $j \in_R [1, s]$. $\mathcal{S}$ selects static and ephemeral key pairs on behalf of honest users with the following exceptions. The j-th ephemeral public key X selected on behalf of U_A is chosen to be U, the static public key B selected on behalf of U_B is chosen to be V, and the static public key C selected on behalf of U_C is chosen to be W, $\mathcal{S}$ does not possess the corresponding static and ephemeral private keys. We denote static and ephemeral public keys of user U_i by S_i and X_i.

A.1.2 Simulation

$\mathcal{S}$ activates $\mathcal{A}$ on this set of users and simulates oracle queries as follows.

1. $\mathsf{Send}(U_i, ('start', \mathsf{P}, U_i, U_j, U_k))$: $\mathcal{S}$ selects ephemeral private key x_i randomly, computes ephemeral public key $X_i = g^{x_i}$, returns $(\mathsf{P}, U_i, U_j, U_k, X_i)$, and records it.
2. $\mathsf{Send}(\Pi^l_{U_i}, (\mathsf{P}, U_i, U_j, U_k, X_j, X_k))$: If $(\mathsf{P}, U_i, U_j, U_k, X_i)$ is recorded, $\mathcal{S}$ records instance $\Pi^l_{U_i}$ is completed. Otherwise, $\mathcal{S}$ records instance $\Pi^l_{U_i}$ is not completed.
3. $\mathsf{SessionKeyReveal}(\Pi^l_{U_i} = (\mathsf{P}, U_i, U_i, S_i, X_i, U_j, S_j, X_j, U_k, S_k, X_k))$: $\mathcal{S}$ maintains list L_S of query $\Pi^l_{U_i}$ and answered session key K.
   a. If instance $\Pi^l_{U_i}$ is not completed, $\mathcal{S}$ returns error.
   b. Else if instance $\Pi^l_{U_i}$ is recorded in L_S, $\mathcal{S}$ returns recorded session key K.
   c. Else if $(Z_1, \ldots, Z_m, \mathsf{P}, U_i, S_i, X_i, U_j, S_j, X_j, U_k, S_k, X_k)$ is recorded in L_H, $\mathcal{S}$ verifies whether $Z_1, \ldots, Z_m$ are correctly formed w.r.t. $S_i, X_i, S_j, X_j, S_k, X_k$ or not by the procedure *Check* described below. If $Z_1, \ldots, Z_m$ are correctly formed, $\mathcal{S}$ returns recorded session key K and records it in L_S.
   d. Otherwise, $\mathcal{S}$ returns random session key K, and records it in L_S.
4. $H(Z_1, \ldots, Z_m, \mathsf{P}, U_i, S_i, X_i, U_j, S_j, X_j, U_k, S_k, X_k)$: $\mathcal{S}$ maintains list L_H of H query and answered hash value K.
   a. If $(Z_1, \ldots, Z_m, \mathsf{P}, U_i, S_i, X_i, U_j, S_j, X_j, U_k, S_k, X_k)$ is recorded in L_H, $\mathcal{S}$ returns recorded hash value K.
   b. Else if instance $\Pi^l_{U_i} = (\mathsf{P}, U_i, U_i, S_i, X_i, U_j, S_j, X_j, U_k, S_k, X_k)$ is recorded in L_S, $\mathcal{S}$ verifies whether $Z_1, \ldots, Z_m$ are correctly formed w.r.t. $S_i, X_i, S_j, X_j, S_k, X_k$ or not by the procedure *Check*

described below. If $Z_1, \ldots, Z_m$ are correctly formed, $\mathcal{S}$ returns recorded session key K and records it in L_H.

c. Else if $(\mathsf{P}, U_i, U_i, S_i, X_i, U_j, S_j, X_j, U_k, S_k, X_k)$ is corresponding to the test instance $(\mathsf{P}, U_A, U_A, A, X = U, U_B, B = V, Y, U_C, C = W, Z)$, $\mathcal{S}$ verifies whether $Z_1, \ldots, Z_m$ are correctly formed w.r.t. $S_i, X_i, S_j, X_j, S_k, X_k$ or not by the procedure *Check* described below using the knowledge of a_0. If $Z_1, \ldots, Z_m$ are correctly formed, $\mathcal{S}$ computes the answer of the gap BDH problem by the procedure *Extract* described below using the knowledge of a_0. Then $\mathcal{S}$ stops and is successful by outputting answer of gap BDH problem.

d. Otherwise, $\mathcal{S}$ returns random hash value K, and records it in L_H.

5. $H_e(X_i)$: $\mathcal{S}$ simulates random oracle in the usual way.
6. $\mathsf{StateReveal}(\Pi^l_{U_i})$: If ephemeral public key of instance $\Pi^l_{U_i}$ is U, then $\mathcal{S}$ aborts with failure, otherwise responds to the query faithfully.
7. $\mathsf{StaticKeyReveal}(U_i)$: If static public key of user U_i is V or W, then $\mathcal{S}$ aborts with failure, otherwise responds to the query faithfully.
8. $\mathsf{AddUser}(U_i, S)$: $\mathcal{S}$ responds to the query faithfully.
9. $\mathsf{Test}(\Pi^l_{U_i})$: If ephemeral public key of the owner is U and static public keys of the other users are V, W in instance $\Pi^l_{U_i}$, then $\mathcal{S}$ responds to the query faithfully, otherwise $\mathcal{S}$ aborts with failure.
10. If $\mathcal{A}$ outputs a guess γ, $\mathcal{S}$ aborts with failure.

(1) *Extract.*

The solver $\mathcal{S}$ can extract the answer of the gap BDH instance as follows. By eliminating the terms including a_0 using the knowledge of a_0, solver $\mathcal{S}$ can obtain

$$\sigma'_1 = (\sigma_1/e(B_0B_1, C_0C_1)^{-a_0})^{1/D} = g_T^{a_1(b_0+b_1)(c_0+c_1)},$$

$$\sigma'_2 = (\sigma_2/e(B_0B_1^E, C_0C_1)^{-a_0}) = g_T^{a_1(b_0+Eb_1)(c_0+c_1)},$$

$$\sigma'_3 = (\sigma_3/e(B_0B_1, C_0C_1^F)^{-a_0}) = g_T^{a_1(b_0+b_1)(c_0+Fc_1)},$$

$$\sigma'_4 = (\sigma_4/e(B_0B_1^E, C_0C_1^F)^{-a_0})^{1/D} = g_T^{a_1(b_0+Eb_1)(c_0+Fc_1)}.$$

By using these 4 linearly independent terms, solver $\mathcal{S}$ can extract the answer $g_T^{a_1b_0c_0} = g_T^{uvw}$ as

$$((\sigma'^E_1\sigma'^{-1}_2)^F(\sigma'^E_3\sigma'^{-1}_4)^{-1})^{1/(E-1)(F-1)} = g_T^{a_1b_0c_0}.$$

(2) *Check.*

The solver $\mathcal{S}$ can check if shared secrets $\sigma_1, \sigma_2, \sigma_3, \sigma_4$ are correctly formed w.r.t. the static and ephemeral public keys by asking BDDH oracle

$$\mathrm{BDDH}(A_0A_1^D, B_0B_1, C_0C_1, \sigma_1) = 1,$$

$$\mathrm{BDDH}(A_0A_1, B_0B_1^E, C_0C_1, \sigma_2) = 1,$$

$$\mathrm{BDDH}(A_0A_1, B_0B_1, C_0C_1^F, \sigma_3) = 1,$$

$$\mathrm{BDDH}(A_0A_1^D, B_0B_1^E, C_0C_1^F, \sigma_4) = 1,$$

and this implies $\sigma_1, \sigma_2, \sigma_3, \sigma_4$ are correctly formed.

A.1.3 Analysis

The simulation of $\mathcal{A}$ environment is perfect except with negligible probability. The probability that $\mathcal{A}$ selects the instance, where ephemeral public key of the owner is U and static public keys of the other users are V, W, as the test instance Π^t is at least $1/(n^3s)$. Suppose this is indeed the case, $\mathcal{S}$ does not abort as in Step 9, and suppose event $E_{1a} \wedge M^*$ occurs, $\mathcal{S}$ does not abort in Step 7 and Step 6.

Under event M^* except with negligible probability, $\mathcal{A}$ queries H with $(\mathsf{P}, U_A, U_A, A, X = U, U_B, B = V, Y, U_C, C = W, Z)$ Therefore $\mathcal{S}$ is successful as described in Step 4c and does not abort as in Step 10.

Hence, $\mathcal{S}$ is successful with probability $Pr(S) \geq p_{1a}/(n^3s)$, where p_{1a} is probability that $E_{1a} \wedge M^*$ occurs.

A.2 Other Events

Event $E_{1b} \wedge M^*$ can be handled same as the event $E_{1a} \wedge M^*$ in Sect. A.1, except that $\mathcal{S}$ embeds gap BDH instance (U, V, W) as $A = U, B = V, C = W$.

Event $E_{2a} \wedge M^*$ can be handled same as the event $E_{1a} \wedge M^*$ in Sect. A.1, except that $\mathcal{S}$ embeds gap BDH instance (U, V, W) as $X = U, Y = V, Z = W$.

Event $E_{2b} \wedge M^*$ can be handled same as the event $E_{1a} \wedge M^*$ in Sect. A.1, except that $\mathcal{S}$ embeds gap BDH instance (U, V, W) as $A = U, Y = V, Z = W$.

Event $E_{3a} \wedge M^*$ can be handled same as the event $E_{1a} \wedge M^*$ in Sect. A.1, except that $\mathcal{S}$ embeds gap BDH instance (U, V, W) as $X = U, B = V, Z = W$.

Event $E_{3b} \wedge M^*$ can be handled same as the event $E_{1a} \wedge M^*$ in Sect. A.1, except that $\mathcal{S}$ embeds gap BDH instance (U, V, W) as $A = U, B = V, Z = W$.

Event $E_{3'a} \wedge M^*/E_{3'b} \wedge M^*$ can be handled same as the event $E_{3a} \wedge M^*/E_{3b} \wedge M^*$, because of symmetry of B and C.

A.3 Cases of Reflection Attack

In the case of $A = B \neq C$, events $E^1_{1b}, E^1_{2a}, E^1_{3b}, E^1_{3'a}$ in Table A· 2 can be handled same as events $E_{1b}, E_{2a}, E_{3b}, E_{3'a}$ in Table A· 1, with condition $A = B \neq C$ and square gap BDH assumption.

In the case of $A = C \neq B$, events $E^{1'}_{1b}, E^{1'}_{2a}, E^{1'}_{3a}, E^{1'}_{3'b}$ in Table A· 3 can be handled same as events $E_{1b}, E_{2a}, E_{3a}, E_{3'b}$ in Table A· 1, with condition $A = C \neq B$ and square gap BDH assumption.

In the case of $A \neq B = C$, events $E^2_{1a}, E^2_{1b}, E^2_{2a}, E^2_{2b}$ in Table A· 4 can be handled same as events $E_{1a}, E_{1b}, E_{2a}, E_{2b}$ in Table A· 1, with condition $A \neq B = C$ and square gap BDH assumption.

In the case of $A = B = C$, events E^3_{1b}, E^3_{2a} in Table A· 5 can be handled same as events E_{1b}, E_{2a} in Table A· 1, with condition $A = B = C$ and cubic gap BDH assumption.

Table A·1 Classification of events, when A, B, C are distinct. "ok" means the static key is not revealed, or a partnered instance exists and its ephemeral key is not revealed. "r" means the static or ephemeral key may be revealed. "r/n" means the ephemeral key may be revealed if the corresponding partnered instance exists, or no corresponding partnered instance exists.

	A	X	B	Y	C	Z
E_{1a}	r	ok	ok	r/n	ok	r/n
E_{1b}	ok	r	ok	r/n	ok	r/n
E_{2a}	r	ok	r	ok	r	ok
E_{2b}	ok	r	r	ok	r	ok
E_{3a}	r	ok	ok	r/n	r	ok
E_{3b}	ok	r	ok	r/n	r	ok
$E_{3'a}$	r	ok	r	ok	ok	r/n
$E_{3'b}$	ok	r	r	ok	ok	r/n

Table A·2 Classification of events, when $A = B \neq C$.

	A	X	$B = A$	Y	C	Z
E^1_{1b}	ok	r	ok	r/n	ok	r/n
E^1_{2a}	r	ok	r	ok	r	ok
E^1_{3b}	ok	r	ok	r/n	r	ok
$E^1_{3'a}$	r	ok	r	ok	ok	r/n

Table A·3 Classification of events, when $A = C \neq B$.

	A	X	B	Y	$C = A$	Z
$E^{1'}_{1b}$	ok	r	ok	r/n	ok	r/n
$E^{1'}_{2a}$	r	ok	r	ok	r	ok
$E^{1'}_{3a}$	r	ok	ok	r/n	r	ok
$E^{1'}_{3'b}$	ok	r	r	ok	ok	r/n

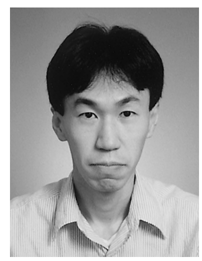
Table A·4 Classification of events, when $A \neq B = C$.

	A	X	B	Y	$C = B$	Z
E^2_{1a}	r	ok	ok	r/n	ok	r/n
E^2_{1b}	ok	r	ok	r/n	ok	r/n
E^2_{2a}	r	ok	r	ok	r	ok
E^2_{2b}	ok	r	r	ok	r	ok

Table A·5 Classification of events, when $A = B = C$.

	A	X	$B = A$	Y	$C = A$	Z
E^3_{1b}	ok	r	ok	r/n	ok	r/n
E^3_{2a}	r	ok	r	ok	r	ok

Mark Manulis received his M.Sc. degree in Computer Science in 2003 from Technische Universität Braunschweig, Germany and his Dr.-Ing. degree in 2007 from Ruhr-Universität Bochum, Germany. Having worked as postdoctoral researcher from 2007 to 2009 at Université catholique de Louvain, Belgium and as assistant professor from 2009 to 2012 at Technische Universität Darmstadt, Germany, he is currently associate professor at the University of Surrey, UK. His research interests are applied cryptography, network security, and privacy.



Koutarou Suzuki received the B.Sc., M.Sc., and Ph.D. degrees from the University of Tokyo in 1994, 1996, and 1999, respectively. He joined NTT, Nippon Telegraph and Telephone Corporation, in 1999. He is engaged in research on public key cryptography at NTT Information Sharing Platform Laboratories. He received the SCIS Paper Award in 2002. He is a member of the Information Processing Society of Japan (IPSJ).

Berkant Ustaoglu received his B.S. degree in Mathematics in 2002 from Boğaziçi University, Turkey. In 2003 and 2008 he received his BMath and Ph.D. degrees from University of Waterloo, Waterloo, Canada. Dr. Ustaoğlu was a postdoctoral fellow in NTT Information Sharing Platform Laboratories, and was a visiting faculty member of Sabancı University, Orhanli - Tuzla, Istanbul 34956, Turkey. He is presently with Izmir Institute of Technology, Urla, Izmir, 35430, Turkey.