Modeling Leakage of Ephemeral Secrets in Tripartite/Group Key Exchange

Mark Manulis[1], Koutarou Suzuki[2], and Berkant Ustaoglu[2]

[1] Cryptographic Protocols Group, TU Darmstadt & CASED, Germany
mark@manulis.eu
[2] NTT Information Sharing Platform Laboratories
3-9-11 Midori-cho Musashino-shi Tokyo 180-8585, Japan
{suzuki.koutarou, ustaoglu.berkant}@lab.ntt.co.jp

Abstract. Recent advances in the design and analysis of secure two-party key exchange (2KE) such as the leakage of ephemeral secrets used *during* the attacked sessions remained unnoticed by the current models for group key exchange (GKE). Focusing on a special case of GKE — the tripartite key exchange (3KE) — that allows for efficient one-round protocols, we demonstrate how to incorporate these advances to the multi-party setting. From this perspective our work closes the most pronounced gap between provably secure 2KE and GKE protocols.
The proposed 3KE protocol is an implicitly authenticated protocol with one communication round which remains secure even in the event of ephemeral secret leakage. It also significantly improves upon currently known 3KE protocols, many of which are insecure. An optional key confirmation round can be added to our proposal to achieve the explicitly authenticated protocol variant.

1 Introduction

Bellare and Rogaway [2] and Blake-Wilson, Johnson and Menezes [3] independently proposed models for analyzing security of two-party key exchange (2KE) protocols in the shared and public key settings, respectively. In their approach an adversary is given the ability to interact with parties and controls the communication with the simple goal of distinguishing a test session key from a random key. Motivating with the signed variant[1] of the classical unauthenticated Diffie-Hellman [13] protocol, Canetti and Krawzcyk [10] argued that it is desirable to augment the 2KE adversary with the ability to learn session-specific and protocol-defined ephemeral information that is not related to the test session. LaMacchia, Lauter and Mityagin [25] allowed leakage of some test session specific ephemeral information under certain conditions. Menezes and Ustaoglu [31] extended the timing of the information leakage. All these developments were within the framework of two-party key exchange.

[1] In the signed Diffie-Hellman protocol users sign outgoing ephemeral public keys with their static keys.

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. [5] inspired by the two-party approach in [2]. Many modifications and improvements appeared thereafter, see the survey in [30]. GKE models mainly focus on the *outsider security* which is modeled through the requirement of AKE-security, e.g. [5,4,21,9], as this requirement deals explicitly with the secrecy of the established keys, which becomes meaningless if the adversary is an insider. Yet, several models, e.g. [20,6,8,15,14], 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). Several compilers have been proposed to augment AKE-secure protocols with security against insider attacks, e.g. [20,6,7]. Beside consideration of outsider and insider security GKE models differ in the treatment of corruptions. Earlier GKE models, e.g. [5,21], considered *weak corruptions* allowing the adversary to obtain users' static keys, but not their ephemeral session secrets. Later models, e.g. [9,8,14] assumed *strong corruptions* allowing the adversary to learn both static private keys and session specific secrets through a single query. Manulis and Bresson [8], inspired by the two-party approach in [10] 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. Gorantla et. al. [14] subsequently strengthened [8] by considering key compromise impersonation attacks.

Despite of their significant improvement over the years GKE models remain incomparable to the 2KE models in terms of security guarantees they provide. In contrast to the 2KE models such as [23,31], GKE models do not consider leakage of ephemeral secrets for the session which is to be proven AKE-secure. In this paper we aim to fix the gap between 2KE and GKE models. Focusing on AKE-security we first revise the latest GKE models to accommodate leakage of ephemeral secrets against the attacked session. In order to illustrate that our model is reasonable and practical for our analysis we focus on three-party key exchange (3KE), which is a special class of GKE protocols and come up with a provably secure solution that resists these stronger leakage attacks.

Notation. Let $\hat{e} : \mathbb{G} \times \mathbb{G} \mapsto \mathbb{G}_T$ be a non-degenerate bilinear map from a group $\mathbb{G}$ to a group $\mathbb{G}_T$ both of prime order q. Let P be a generator of $\mathbb{G}$; for a user U_A we set U_A's static and ephemeral keys $S_A = s_A P$ and $X_A = x_A P$, respectively. The lowercase letters are the private keys.

2 Three vs two party key establishment

Antoine Joux [17,18] used properties of pairings to extends the classical (unauthenticated) two-party Diffie-Hellman protocol [13] to the case of three parties, preserving the optimal one-round communication complexity. Since then tripartite key exchange as a special form of group key exchange has gained attention of the research community and several attempts have been made to improve the original protocol in order to enlarge the class of attacks it can resist.

2.1 Authenticating outgoing messages

Shim [32] argued that Joux's protocol fails to a variant of the well know person-in-the-middle attack against the (unauthenticated) Diffie-Hellman protocol. To address that shortcoming Shim proposed a protocol where Alice broadcasts

$$T_A = x_A s_A P. \tag{1}$$

Upon exchanging these ephemeral public keys the parties compute $t = \hat{e}(P,P)^{s_A s_B s_C}$ and the session key

$$k = \mathtt{H}(\hat{e}(P,P)^{x_A x_B x_C s_A s_B s_C t}, U_A, U_B, U_C). \tag{2}$$

Shim's protocol fails to key compromise impersonation attack [33,29]. Suppose Malice sends two ephemeral public keys uP and vP to Alice on behalf of Bob and Charlie respectively. With the knowledge of Alice's static private key Malice can compute t; with the knowledge of u Malice can also compute $k = \hat{e}(vP, T_A)^{ut}$, which is the key Alice computes. Lin and Lin [29] observe that the attack is possible since Shim's protocol does not authenticate the T_A's origin. To resolve such issues one venue is introducing new elements into the communicated messages: along the ephemeral public key a user could append extra information that identifies messages' origin or provides evidence for following protocol specifications. For example [11] requires that along the message in Shim's protocol Alice also computes and broadcasts $X_A = x_A P$; the suggested session key is

$$k = \hat{e}(P,P)^{x_A x_B x_C s_A s_B s_C}.$$

Suppose, however, that an adversary Malice can obtain a certificate for an ephemeral public key X_A used by Alice. Malice can then send $X_M = S_A$, $T_M = T_A$, and the certificate to Bob and Charlie. As a result Bob and Charlie will believe the session key is shared with Malice, whereas the key is shared with Alice, who correctly identifies all session peers. This example resembles Kaliski's on-line unknown key share (UKS) attack [19] on the MQV protocol [26]. It is plausible [22, §7.3], that the ephemeral public keys pre-computed for efficiency reasons are not as securely stored as the ephemeral private keys. In that case the UKS attack can be made off-line implying that timing information leakage has important security consequences.

The modification to Shim's protocol in [29] requires that in addition to T_A, Alice also computes and broadcasts

$$m_A = \mathtt{H}(s_A, x_A) \tag{3}$$
$$u_A = (s_A x_A)^{-1}(m_A + s_A) \bmod q. \tag{4}$$

Bob and Charlie verify Alice's message by computing $t_A = {u_A}^{-1} \bmod q$, $z_A = t_A m_A \bmod q$, and checking that

$$T_A \stackrel{?}{=} z_A P + t_A S_A. \tag{5}$$

The session key (as computed by Alice) is

$$\begin{aligned} k &= \mathtt{H}(\hat{e}(S_B + T_B, S_C + T_C)^{s_A + s_A x_A}, U_A, U_B, U_C) \\ &= \mathtt{H}(\hat{e}(P, P)^{(s_A + s_A x_A)(s_B + s_B x_B)(s_C + s_C x_C)}, U_A, U_B, U_C). \end{aligned} \tag{6}$$

Malice can easily circumvent the verification by selecting a random integer m_B, setting $T_B = -m_B P - S_B$, $u_B = -1 \bmod q$ and sending these values on Bob's behalf to Alice and Charlie; see also [28, §4.1]. Alice (as well as Charlie) compute $t_B = -1^{-1} = -1 \bmod q$, $z_B = -m_B \bmod q$ and verify Equation 5 namely,

$$T_B = z_B P + t_B S_B = -m_B P - S_B.$$

Subsequently, Alice computes the key

$$\begin{aligned} k &= \mathtt{H}(\hat{e}(S_B + T_B, S_C + T_C)^{s_A + s_A x_A}, U_A, U_B, U_C) \\ &= \mathtt{H}(\hat{e}(S_B - m_B P - S_B, S_C + T_C)^{s_A + s_A x_A}, U_A, U_B, U_C) \\ &= \mathtt{H}(\hat{e}(P, P)^{(s_A + s_A x_A)(-m_B)(s_C + s_C x_C)}, U_A, U_B, U_C). \end{aligned} \tag{7}$$

With the knowledge of m_B Malice can compute the same session key.

Lim et al. [28] further propose a "fix" to the above problem that requires additional information in the messages and further verification procedures. However, as observed in [27, §4.2], u_A relates the static and ephemeral key such that given the static private key s_A an adversary can derive the ephemeral private key x_A and thereafter recover the session key, so protocols with u_A as in Equation 4 do not provide forward secrecy. As an alternative [27] suggests $W_A = x_A \mathtt{H}(x_A)(S_A)$, $n_A = \mathtt{H}(T_A, W_A, p_A)$ for a time stamp p_A, and

$$s_A = (s_A x_A \mathtt{H}(x_A))^{-1}(m_A + s_A n_A) \bmod q. \tag{8}$$

The above examples aimed to provide certain assurances about incoming messages without allegedly sacrificing security. Compilers can be viewed as an abstraction to such approach, at the expense of overhead like complicated messages or more communication rounds. A more rigorous analysis of that approach can be found in [21,16].

Ephemeral key leakage has been motivated for two party key agreement protocols [10,23,34], but so far we did not include it in our analysis. In Equation 8

if s_A is leaked the adversary cannot obtain x_A, but if x_A is leaked, then the adversary can easily obtain the static secret s_A. Furthermore, in [24] authors observed that within a party cryptographic primitives can share the source of randomness; if the source is weak then signature schemes such as DSA can leak static private keys. Therefore, in the presence of leakage of ephemeral private information compilers' based solutions are non-trivial to adopt.

2.2 Al-Riyami and Patterson protocols

Al-Riyami and Patterson [1] proposed four one round three party key agreement protocols. The design aims to "avoid the use of expensive signature computations". The protocols broadcast a message consisting of a single ephemeral public key along with necessary certificates, but differ in the key derivation procedures which are inspired by two-party protocols. These protocols inherit vulnerabilities from the underlying two-pass protocols, but suggest that lessons from two-party protocols should be applied to three-party protocols. In the TAK-4 protocol, akin to MQV [26] and HQMV [23], Alice, Bob and Charlie after exchanging static-ephemeral key pairs (S_A, X_A), (S_B, X_B) and (S_C, X_C), respectively, compute the session key

$$k = \hat{e}(P,P)^{(x_A+\mathtt{H_e}(X_A,S_A)s_A)(x_B+\mathtt{H_e}(X_B,S_B)s_B)(x_C+\mathtt{H_e}(X_C,S_C)s_C)}.$$

Given, the complicated HMQV security argument it is not surprising that no security argument for TAK-4 is provided. In fact as described in [1] TAK-4 fails to the following UKS attack in which Alice and Bob will falsely think that they share a key with Malice, whereas Charlie correctly identifies his peers as Alice and Bob. In the attack Malice, who owns a certificate for the public key 1_G[2], intercepts all public keys and computes $X_M = X_C + \mathtt{H_e}(X_C, S_C)S_C$, implicitly defining $x_M = x_C + \mathtt{H_e}(X_C, S_C)s_C$. Note that

$$\begin{aligned} k &= \hat{e}(P,P)^{(x_A+\mathtt{H_e}(X_A,S_A)s_A)(x_B+\mathtt{H_e}(X_B,S_B)s_B)(x_C+\mathtt{H_e}(X_C,S_C)s_C)} \\ &= \hat{e}(P,P)^{(s_A+\mathtt{H_e}(S_A,S_A)s_A)(x_B+\mathtt{H_e}(X_B,S_B)s_B)(x_C+\mathtt{H_e}(X_C,S_C)s_C+\mathtt{H}(X_M,1_G)0)} \\ &= \hat{e}(P,P)^{(s_A+\mathtt{H_e}(S_A,S_A)s_A)(x_B+\mathtt{H_e}(X_B,S_B)s_B)(x_M+\mathtt{H}(X_M,1_G)0)}. \end{aligned}$$

Therefore, by sending (S_M, X_M) instead of (S_C, X_C) to Alice or Bob, Malice successfully mounts a UKS attack on TAK-4. The possibility of such attacks is acknowledged in [1], which also offers two alternatives to prevent them. The requirements in [1] do not prevent the adversary from mounting the above attack thus the more sound approach is to include identities in the key derivation as typically done in two party key agreement. In general fewer assumptions and primitives are better as they leave less room for security vulnerabilities.

[2] The element 1_G is the identity element in $\mathbb{G}$.

2.3 Ephemeral information leakage

In general, the security considerations important for two-party protocols are also relevant for multi-party protocols. Motivation for ephemeral information leakage is independent from number of users involved in a key agreement protocol. Primitives used in compilers often assume no ephemeral key leakage. Thus, it is worth considering implicitly authenticated key exchange protocols.

Ephemeral keys introduce further security aspects. For example, in Shim's protocol leaking static keys does not reveal the past session keys, but an adversary that can access one ephemeral and one static private key from different users can compute the session key. So, for a party concerned with forward secrecy with respect to its own static key, there is a difference if its peer static or ephemeral private key is leaked: the session key is still secure in the former case but no longer in the latter.

3 Implicitly authenticated tripartite protocol

Informally, in our proposed protocol $\mathtt{P}$ parties exchange ephemeral and static keys and derive the keying material as described bellow. Optionally, there can be a key confirmation round.

Initialization. User U_i performs:

1. Select an ephemeral private key $x_i \in_R [1, q]$ and compute $X_i = g^{x_i}$.
2. Create a session state, identified by $(\mathtt{P}, U_i, X_i)$ that contains only (x_i, X_i).

Communication. Upon receiving request: $(\mathtt{P}, U_i, U_{i+1}, U_{i+2}, \mathtt{rl})$, user U_i broadcasts $(1|\mathtt{P}, U_0, U_1, U_2, \mathtt{rl}, X_i)$.

Derivations. Upon receiving the first round of messages U_i does the following:

1. Verify that $X_{i+1}, X_{i+2} \in \mathbb{G}^*$.
2. Compute $\mathtt{sid}^i = \mathtt{P}|U_0|X_0|U_1|X_1|U_2|X_2$.
3. Compute $\mathrm{KeyDer}(U_i, \mathtt{rl}, \mathtt{sid}^i, x_i, s_i)$.

Completion. To complete the session U_i does:

1. Destroy the session state.
2. Accept the session key k.

Key material. On input $(U_i, \mathtt{rl}, \mathtt{sid}^i, x_i, s_i)$ the auxiliary key derivation KeyDer computes:

1. Compute $\mathrm{F}_0 = \mathtt{H_e}(X_0)$, $\mathrm{F}_1 = \mathtt{H_e}(X_1)$ and $\mathrm{F}_2 = \mathtt{H_e}(X_2)$.
2. Compute
$$\sigma_0 = \begin{cases} (\hat{e}(X_1 + S_1, X_2 + S_2))^{x_0 + \mathrm{F}_0 s_0} \text{ if } \mathtt{rl} = 0 \\ (\hat{e}(X_0 + \mathrm{F}_0 S_0, X_2 + S_2))^{x_1 + s_1} \text{ if } \mathtt{rl} = 1 \\ (\hat{e}(X_0 + \mathrm{F}_0 S_0, X_1 + S_1))^{x_2 + s_2} \text{ if } \mathtt{rl} = 2 \end{cases} \tag{9}$$
3. Compute
$$\sigma_1 = \begin{cases} (\hat{e}(X_1 + \mathrm{F}_1 S_1, X_2 + S_2))^{x_0 + s_0} \text{ if } \mathtt{rl} = 0 \\ (\hat{e}(X_0 + S_0, X_2 + S_2))^{x_1 + \mathrm{F}_1 s_1} \text{ if } \mathtt{rl} = 1 \\ (\hat{e}(X_0 + S_0, X_1 + \mathrm{F}_1 S_1))^{x_2 + s_2} \text{ if } \mathtt{rl} = 2 \end{cases} \tag{10}$$

4. Compute

$$\sigma_2 = \begin{cases} (\hat{e}(X_1 + S_1, X_2 + \mathrm{F}_2 S_2))^{x_0+s_0} \text{ if } \mathtt{rl} = 0 \\ (\hat{e}(X_0 + S_0, X_2 + \mathrm{F}_2 S_2))^{x_1+s_1} \text{ if } \mathtt{rl} = 1 \\ (\hat{e}(X_0 + S_0, X_1 + S_1))^{x_2+\mathrm{F}_2 s_2} \text{ if } \mathtt{rl} = 2 \end{cases} \quad (11)$$

5. Compute

$$\sigma_3 = \begin{cases} (\hat{e}(X_1 + \mathrm{F}_1 S_1, X_2 + \mathrm{F}_2 S_2))^{x_0+\mathrm{F}_0 s_0} \text{ if } \mathtt{rl} = 0 \\ (\hat{e}(X_0 + \mathrm{F}_0 S_0, X_2 + \mathrm{F}_2 S_2))^{x_1+\mathrm{F}_1 s_1} \text{ if } \mathtt{rl} = 1 \\ (\hat{e}(X_0 + \mathrm{F}_0 S_0, X_1 + \mathrm{F}_1 S_1))^{x_2+\mathrm{F}_2 s_2} \text{ if } \mathtt{rl} = 2 \end{cases} \quad (12)$$

6. Return $k = \mathtt{H}(\sigma_0, \sigma_1, \sigma_2, \sigma_3, \mathtt{sid}^i)$.

Instances with the same session id $\mathtt{sid}$, and hence with the same ephemeral public keys and partners, compute the same output since

$$\begin{aligned} \mathtt{H}(\sigma_0, \sigma_1, \sigma_2, \sigma_3, \mathtt{sid}^i) = \mathtt{H}\big(& (\hat{e}(P,P))^{(x_0+\mathrm{F}_0 s_0)(x_1+s_1)(x_2+s_2)}, \\ & (\hat{e}(P,P))^{(x_0+s_0)(x_1+\mathrm{F}_1 s_1)(x_2+s_2)}, \\ & (\hat{e}(P,P))^{(x_0+s_0)(x_1+s_1)(x_2+\mathrm{F}_2 s_2)}, \\ & (\hat{e}(P,P))^{(x_0+\mathrm{F}_0 s_0)(x_1+\mathrm{F}_1 s_1)(x_2+\mathrm{F}_2 s_2)}, \mathtt{sid}^i \big). \end{aligned} \quad (13)$$

A special attention should be paid to the content of the internal state which by definition contains only the ephemeral private keys used by session throughout the protocol execution. Neither the static private key s_i, nor the values $\sigma_0 \ldots \sigma_3$, nor the derived key material become part of the session state. This is different from the definition used in [12], where the model allows the adversary to learn the complete state of the Turing machine. Our formulation is similar to the more common approach for two party Diffie-Hellman protocols, see for example [10,23,34], where the session state consists only of the ephemeral private key x_i used by U_i.

To include key confirmation, the output of $\mathtt{H}$ is modified to (k_m, k). Furthermore, after Derivation and before Completion users perform the following:

Confirmation. To execute key confirmation U_i does:

1. Compute tags $T_0 = \mathtt{H}_3(k_m, U_0, X_0, \mathtt{sid}^i)$, $T_1 = \mathtt{H}_3(k_m, U_1, X_1, \mathtt{sid}^i)$, and $T_2 = \mathtt{H}_3(k_m, U_2, X_2, \mathtt{sid}^i)$.
2. Record[3] T_{i+1} and T_{i+2}, and delete k_m.
3. Broadcast $(2|\mathtt{P}, T_i, U_i, \mathtt{sid}^i, \mathtt{rl})$

Verification. U_i verifies that the incoming T_{i+1} and T_{i+2} are equal to the tags stored in the session state.

[3] To prevent leakage of these confirmation tags, U_i can store fingerprint of these tags. Upon obtaining tags from the alleged peers U_i computes and compares fingerprints of incoming tags with the fingerprints stored in the session state. Thus we can assume that the confirmation tags do not become part of the session state.

In the analysis of many two-party protocols ephemeral public and private keys can be obtained by the adversary only during the session execution. Thus such arguments do not cover pre-computed ephemeral key pairs. In some cases the adversary may be able to recover past ephemeral keys. For this reason in our protocol description the ephemeral key pairs are pre-computed and the adversary can access them before event the session is initialized. Indeed the Initialization stage can be performed long before the Communication stage. Similarly, the protocol description does not explicitly destroy the ephemeral private key (but should be done in practice) to allow the possibility that the adversary obtain the ephemeral key after observing some subsequent actions of the parties. These modifications only increase the power of the adversary and does not decrease it relative to the usual approach where ephemeral keys can be obtained only during the session execution.As mentioned in the introduction, Bresson and Manulis [8] considered leakage of ephemeral secrets from the internal states prior to the execution of a session, thus incorporating pre-computations into the model, and also after the completeness of the session, thus implicitly requiring the erasure of ephemeral secrets from the state. However, their approach disallows leakage of ephemeral secrets during the execution of the session.

4 The Model and Security Definitions

Our model can be seen as an extension of the strong authenticated key exchange model for two-party protocols from [31] to the group setting. It is described using the classical notations and terminology from previous models for GKE protocols, in particular those in [21,8,14].

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.

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*[4] $\mathtt{pid}_U^s \subseteq \mathcal{U}$, which encompasses the identities of all the intended session participants (note that $\mathtt{pid}_U^s$ also includes U). 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.

[4] Invocation may also include other public information such as the protocol name that is invoked, the order of user and so on.

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. We say that U *owns* session $\mathtt{sid}_U^s$ if the instance Π_U^s was invoked at U. Note that the integer s is only a tool to describe the model. The users do not keep track of s, instead sessions are identified via the vector $\mathtt{sid}_U^s$. At the onset of the instance the user that owns the instance may not have enough information to create $\mathtt{sid}_U^s$; until $\mathtt{sid}_U^s$ is created the instance is identified via $\mathtt{pid}_U^s$ and the outgoing ephemeral public key[5] which is unique per user except with negligible probability. Furthermore, we assume that instances that accepted or aborted delete all information in their respective states.

Partnering. Two instances Π_U^s and $\Pi_{U_*}^t$ are called *partnered* or *matching* if $\mathtt{sid}_U^s \subseteq \mathtt{sid}_{U_*}^t$ or $\mathtt{sid}_{U_*}^t \subseteq \mathtt{sid}_U^s$ and $\mathtt{pid}_U^s = \mathtt{pid}_{U_*}^t$. The first condition models the fact that if session ids are computed during the protocol execution, e.g. from the exchanged messages, then their equality should be guaranteed only at the end of the protocol, i.e. upon the acceptance of Π_U^s and $\Pi_{U_*}^t$.

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.

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[6].
- *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*$(U, (\text{'}start\text{'}, U_1, \ldots, U_n))$ 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.
- *RevealKey*(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.
- *RevealStaticKey*(U): This query provides $\mathcal{A}$ with the static private key s_U.
- *RevealState*(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.

[5] Implicitly, this assumes that the first outgoing message contains the ephemeral public key. If necessary this can be modified to accommodate other types of protocols.

[6] In our security argument we will only assume that S_U chosen by $\mathcal{A}$ is checked to be an element of $\mathbb{G}$.

– *Test*(Π^s_U): 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^s_U. The query is requires that Π^s_U has accepted.

Correctness. A GKE protocol is said to be *correct* if, when in the presence of benign[7] adversary all instances invoked for the same protocol session accept with the same session group key.

Freshness. The classical notion of freshness of some instance Π^s_U is traditionally used to define the goal of AKE-security by specifying the conditions for the *Test*(Π^s_U) query. For example, the model in [21] defines an instance Π^s_U that has accepted as fresh if none of the following is true: (1) at some point, $\mathcal{A}$ asked *RevealKey* to Π^s_U or to any of its partnered instances; or (2) a query *RevealStaticKey*(U_*) with $U_* \in \mathtt{pid}^s_U$ was asked before a *Send* query to Π^s_U or any of its partnered instances.

Unfortunately, these restrictions are not sufficient for our purpose since Π^s_U 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^s_U. We fairly remark that [21] does not address (strong) corruptions of ephemeral secrets.

The recent model in [8] defines freshness using the additional *AddUser* and *RevealState* queries as follows. According to [8], an instance Π^s_U that has accepted is fresh if none of the following is true: (1) $\mathcal{A}$ queried *AddUser*(U_M, S_{U_M}) with some $U_* \in \mathtt{pid}^s_U$; or (2) at some point, $\mathcal{A}$ asked *RevealKey* to Π^s_U or any of its partnered instances; or (3) a query *RevealStaticKey*(U_*) with $U_* \in \mathtt{pid}^s_U$ was asked before a *Send* query to Π^s_U or any of its partnered instances; or (4) $\mathcal{A}$ queried *RevealState* to Π^s_U 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 [21] 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 [14] addressed the lack of key compromise impersonation resilience. In particular, it modifies the above condition (3) by requiring that if there exists an instance $\Pi^t_{U_*}$ which is partnered with Π^s_U and $\mathcal{A}$ asked *RevealStaticKey*(U_*) then all messages sent by $\mathcal{A}$ to Π^s_U on behalf of $\Pi^t_{U_*}$ must come from $\Pi^t_{U_*}$ intended for Π^s_U. 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

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

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 give the combined definition of freshness taking into account the previously described problems.

Definition 1. *An instance Π^s_U that has accepted is fresh if none of the following is true:*

1. *$\mathcal{A}$ queried AddUser(U_*, S_{U_*}) with some $U_* \in \mathtt{pid}^s_U$; or*
2. *$\mathcal{A}$ asked RevealKey to Π^s_U or any of its accepted partnered instances; or*
3. *$\mathcal{A}$ queried both RevealStaticKey(U_*) with $U_* \in \mathtt{pid}^s_U$ and RevealState$(\Pi^t_{U_*})$ for some instance $\Pi^t_{U_*}$ partnered with Π^s_U; or*
4. *$\mathcal{A}$ queried RevealStaticKey(U_*) with $U_* \in \mathtt{pid}^s_U$ prior to the acceptance of Π^s_U and there exists no instance $\Pi^t_{U_*}$ partnered with Π^s_U.*

Note that since $U \in \mathtt{pid}^s_U$ 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^s_U 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 $\mathtt{pid}^s_U$. 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 that users during it. On the other hand, as long as $\mathcal{A}$ remains benign with respect to such users their instances will still be considered as fresh.

AKE-Security. We are ready to generalize the strong AKE-security definition from [25,31] to a group setting.

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

$$\textit{We define:} \qquad \mathsf{Adv}^{\mathrm{ake}}_{\mathcal{A},\mathtt{P}}(\kappa) := |2\Pr[\tau = \tau'] - 1|$$

and denote with $\mathsf{Adv}^{\mathrm{ake}}_{\mathtt{P}}(\kappa)$ the maximum advantage over all PPT adversaries $\mathcal{A}$. We say that a GKE protocol $\mathtt{P}$ provides strong AKE-security *if this advantage is negligible.*

5 Security Arguments

In this section, we provide security arguments of the proposed implicitly authenticated tripartite protocol. We need the gap BDH(Bilinear Diffie-Hellman) assumption, where one tries to compute $\mathrm{BDH}(U,V,W)$ accessing the BDDH oracle. Here, we denote $\mathrm{BDH}(U,V,W) = \hat{e}(P,P)^{\log U \log V \log W}$, and the BDDH oracle on input $(uP, vP, wP, \hat{e}(P,P)^x)$ returns the bit 1 if $uvw = x$ and the bit 0 otherwise.

Theorem 1. *If $\mathbb{G}$ is a group where gap Bilinear Diffie-Hellman assumption holds and $\mathtt{H}$ and $\mathtt{H_e}$ are random oracles, the proposed implicitly authenticated tripartite protocol in Section 3 is secure in the sense of Definition 2.*

Outline of proof of Theorem 1 is provided in Appendix A. Here, we give an intuition of the proof. We denote by $(S_0,X_0),(S_1,X_1),(S_2,X_2)$ the static and ephemeral public keys of users U_0,U_1,U_2 in the test session $\mathtt{sid}^t$. Consider the case, where user U_0 is honest, ephemeral public key X_0 is not revealed, and static public keys S_1 and S_2 are not revealed. In this case, solver $\mathcal{S}$ embeds instance (U,V,W) of gap BDH problem as $X_0 = U, S_1 = V, S_2 = W$. Since $\mathtt{H}$ is random oracle, adversary $\mathcal{A}$ need to ask $\sigma_0,\sigma_1,\sigma_2,\sigma_3$ to $\mathtt{H}$, s.t. $\mathrm{BDDH}(X_0 + \mathrm{F}_0 S_0, X_1 + S_1, X_2 + S_2, \sigma_0) = 1$, $\mathrm{BDDH}(X_0 + S_0, X_1 + \mathrm{F}_1 S_1, X_2 + S_2, \sigma_1) = 1$, $\mathrm{BDDH}(X_0 + S_0, X_1 + S_1, X_2 + \mathrm{F}_2 S_2, \sigma_2) = 1$, and $\mathrm{BDDH}(X_0 + \mathrm{F}_0 S_0, X_1 + \mathrm{F}_1 S_1, X_2 + \mathrm{F}_2 S_2, \sigma_3) = 1$, to distinguish the session key. Since user U_0 is honest, solver $\mathcal{S}$ knows $s_0 = \log(S_0)$. By using s_0, solver $\mathcal{S}$ can compute four independent terms w.r.t. $s_1 = \log(S_1)$ and $s_2 = \log(S_2)$: $\sigma_0' = \hat{e}(X_1 + S_1, X_2 + S_2)^{-\mathrm{F}_0 s_0}\sigma_0 = \hat{e}(P,P)^{x_0(x_1+s_1)(x_2+s_2)}$, $\sigma_1' = \hat{e}(X_1 + \mathrm{F}_1 S_1, X_2 + S_2)^{-s_0}\sigma_1 = \hat{e}(P,P)^{x_0(x_1+\mathrm{F}_1 s_1)(x_2+s_2)}$, $\sigma_2' = \hat{e}(X_1 + S_1, X_2 + \mathrm{F}_2 S_2)^{-s_0}\sigma_2 = \hat{e}(P,P)^{x_0(x_1+s_1)(x_2+\mathrm{F}_2 s_2)}$, and $\sigma_3' = \hat{e}(X_1 + \mathrm{F}_1 S_1, X_2 + \mathrm{F}_2 S_2)^{-\mathrm{F}_0 s_0}\sigma_3 = \hat{e}(P,P)^{x_0(x_1+\mathrm{F}_1 s_1)(x_2+\mathrm{F}_2 s_2)}$. By using these four independent terms, solver $\mathcal{S}$ can compute answer of gap BDH problem $((\sigma_0'^{-1}\sigma_1')^{-1}\sigma_2'^{-1}\sigma_3')^{1/((\mathrm{F}_1-1)(\mathrm{F}_2-1))} = \mathrm{BDH}(X_0,S_1,S_2)$. This is why the proposed protocol uses four terms $\sigma_0,\sigma_1,\sigma_2,\sigma_3$.

6 Conclusion

We presented a new 3KE protocol and a more general GKE model that takes into account ephemeral key leakage. In this way we closed the outstanding gap in the modeling of AKE-security for 2KE and GKE protocols. Our implicitly authenticated 3KE protocol does not make use of compilers and proceeds in one round achieving this desired higher level of security. As such it is the first one-round tripartite key exchange protocol having these security properties without complicating the messages of the original protocol by Joux [17,18].

We did not take into account malicious insiders in GKE protocols [8,14] and did not consider the possibility of invoking sessions with destination addresses as done in the so called post-specified peer model [10,31]. It is an interesting open problem to formally consider the post-specified peer setting. Furthermore, it is of independent worth to provide methods for key confirmation and contributiveness for implicitly authenticated protocols that tolerate malicious insiders.

References

1. S. S. Al-Riyami and K. G. Paterson. Tripartite Authenticated Key Agreement Protocols from Pairings. In *9th IMA International Conference*, volume 2898 of *LNCS*, pages 332–359, 2003.
2. M. Bellare and P. Rogaway. Entity Authentication and Key Distribution. In *Advances in Cryptology – CRYPTO'93*, volume 773 of *LNCS*, pages 232–249, 1993.
3. S. Blake-Wilson, D. Johnson, and A. Menezes. Key Agreement Protocols and their Security Analysis. In *6th IMA International Conference*, volume 1355 of *LNCS*, pages 30–45, 1997.
4. E. Bresson, O. Chevassut, and D. Pointcheval. Dynamic Group Diffie-Hellman Key Exchange under Standard Assumptions. In *Advances in Cryptology – EUROCRYPT'02*, volume 2332 of *Lecture Notes in Computer Science*, pages 321–336. Springer, 2002.
5. E. Bresson, O. Chevassut, D. Pointcheval, and J.-J. Quisquater. Provably Authenticated Group Diffie-Hellman Key Exchange. In *Proceedings of the 8th ACM conference on Computer and Communications Security (CCS'01)*, pages 255–264. ACM Press, 2001.
6. E. Bresson and M. Manulis. Malicious Participants in Group Key Exchange: Key Control and Contributiveness in the Shadow of Trust. In *Proceedings of the 4th Autonomic and Trusted Computing Conference (ATC 2007)*, volume 4610 of *Lecture Notes in Computer Science*, pages 395–409. Springer-Verlag, 2007.
7. E. Bresson and M. Manulis. Contributory Group Key Exchange in the Presence of Malicious Participants. *IET Information Security*, 2(3):85–93, 2008.
8. E. Bresson and M. Manulis. Securing Group Key Exchange against Strong Corruptions. In *Proceedings of ACM Symposium on Information, Computer and Communications Security (ASIACCS'08)*, pages 249–260. ACM Press, 2008. Full version in Intl. J. Applied Cryptography in 2008.
9. E. Bresson, M. Manulis, and J. Schwenk. On Security Models and Compilers for Group Key Exchange Protocols. In *Proceedings of the 2nd International Workshop on Security (IWSEC 2007)*, volume 4752 of *Lecture Notes in Computer Science*, pages 292–307. Springer-Verlag, October 2007.
10. R. Canetti and H. Krawczyk. Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In *Advances in Cryptology – EUROCRYPT'01*, volume 2045 of *LNCS*, pages 453–474, 2001.
11. Z. Cheng, L. Vasiu, and R. Comley. Pairing-based one-round tripartite key agreement protocols. Cryptology ePrint Archive, Report 2004/079, 2004.
12. C. Cremers. Session-state reveal is stronger than ephemeral key reveal: Attacking the NAXOS key exchange protocol. In M. Abdalla, D. Pointcheval, P.-A. Fouque, and D. Vergnaud, editors, *Applied Cryptography and Network Security, 7th International Conference, ACNS 2009*, volume 5536 of *LNCS*, pages 20–33. Springer Verlag, June 2009.
13. W. Diffie and M. E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, November 1976.
14. M. C. Gorantla, C. Boyd, and J. M. González-Nieto. Modeling Key Compromise Impersonation Attacks on Group Key Exchange Protocols. In *Public Key Cryptography – PKC 2009*, volume 5443 of *LNCS*, pages 105–123, 2009.
15. M. C. Gorantla, C. Boyd, and J. M. González-Nieto. Universally Composable Contributory Group Key Exchange. In *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security (ASIACCS'09)*, pages 146–156. ACM, 2009.

16. Y. Hitchcock, C. Boyd, and J. M. G. Nieto. Tripartite Key Exchange in the Canetti-Krawczyk Proof Model. In *INDOCRYPT*, volume 3348 of *Lecture Notes in Computer Science*, pages 17–32. Springer, 2004.
17. A. Joux. A one round protocol for tripartite Diffie–Hellman. In W. Bosma, editor, *Algorithmic Number Theory 4th International Symposium, ANTS-IV, Proceedings*, volume 1838 of *LNCS*, pages 385–393. Springer, 2000.
18. A. Joux. A one round protocol for tripartite Diffie–Hellman. *Journal of Cryptology*, 17(4):263–276, 2004.
19. B. S. Kaliski Jr. An unknown key-share attack on the mqv key agreement protocol. *ACM Transaction on Information and System Security*, 4(3):275–288, 2001.
20. J. Katz and J. S. Shin. Modeling Insider Attacks on Group Key-Exchange Protocols. In *Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS'05)*, pages 180–189. ACM Press, 2005.
21. J. Katz and M. Yung. Scalable protocols for authenticated group key exchange. In D. Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *LNCS*, pages 100–125. Springer, 2003. Full version available at http://eprint.iacr.org/2003/171.
22. H. Krawczyk. HMQV: A high-performance secure Diffie-Hellman protocol. Cryptology ePrint Archive, Report 2005/176. Full version of [23].
23. H. Krawczyk. HMQV: A high-performance secure Diffie-Hellman protocol. In R. Cramer, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *LNCS*, pages 546–566. Springer Verlag, 2005.
24. B. LaMacchia, K. Lauter, and A. Mityagin. Stronger security of authenticated key exchange. Cryptology ePrint Archive, Report 2006/073, 2006.
25. B. LaMacchia, K. Lauter, and A. Mityagin. Stronger Security of Authenticated Key Exchange. In *Provable Security: First International Conference, ProvSec 2007*, volume 4784 of *LNCS*, pages 1–16, 2007.
26. L. Law, A. Menezes, M. Qu, J. Solinas, and S. Vanstone. An efficient protocol for authenticated key agreement. *Designs, Codes and Cryptography*, 28(2):119–134, 2003.
27. 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.
28. M.-H. Lim, S. Lee, Y. Park, and H. Lee. An enhanced one-round pairing-based tripartite authenticated key agreement protocol. In O. Gervasi and M. L. Gavrilova, editors, *Computational Science and Its Applications – ICCSA 2007*, volume 4406 of *LNCS*, pages 503–513. Springer, 2007.
29. C.-H. Lin and H.-H. Lin. Secure one-round tripartite authenticated key agreement protocol from Weil pairing. In Y. Shibata and T. K. Shih, editors, *19th International Conference on Advanced Information Networking and Applications – AINA05*, volume 2, pages 135–138. IEEE, 2005.
30. M. Manulis. Survey on Security Requirements and Models for Group Key Exchange. Technical Report 2006/02, Horst-Görtz Institute, Network and Data Security Group, January 2008.
31. A. Menezes and B. Ustaoglu. Comparing the Pre- and Post-specified Peer Models for Key Agreement. In *Information Security and Privacy – ACISP 2008*, volume 5107 of *LNCS*, pages 53–68. Springer, 2008.
32. K. Shim. Efficient one round tripartite authenticated key agreement protocol from Weil pairing. *IET Electronics Letters*, 39(2):208–209, 2003.
33. H.-M. Sun and B.-T. Hsieh. Security Analysis of Shim's Authenticated Key Agreement Protocols from Pairings. Cryptology ePrint Archive, Report 2003/113, 2003.

34. B. Ustaoglu. Comparing *SessionStateReveal* and *EphemeralKeyReveal* for Diffie-Hellman protocols. 2009. to appear in ProvSec09.

A Outline of Proof of Theorem 1

In this section, we provide outline of proof of Theorem 1, because of page limitation. We need the gap BDH(Bilinear Diffie-Hellman) assumption, where one tries to compute BDH(U, V, W) accessing the BDDH oracle. Here, we denote BDH$(U, V, W) = \hat{e}(P, P)^{\log U \log V \log W}$, and the BDDH oracle on input $(uP, vP, wP, \hat{e}(P, P)^x)$ returns the bit 1 if $uvw = x$ and the bit 0 otherwise.

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 $\frac{1}{2} + p(\kappa)$, where $p(\kappa)$ is non-negligible, and the event M denotes a successful $\mathcal{A}$.

Let Π^t be the test instance with session id $\mathtt{sid}^t = (\mathtt{P}, U_A, S_A, X_A, U_B, S_B, X_B, U_C, S_C, X_C)$. 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 $(\sigma_0, \sigma_1, \sigma_2, \sigma_3, \mathtt{sid}^t)$ to $\mathtt{H}$, where $\sigma_0, \sigma_1, \sigma_2, \sigma_3$ 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 $\mathtt{H}$ are different for $\mathtt{sid}$ and $\mathtt{sid}^t$. Since $\mathtt{H}$ is a random oracle, $\mathcal{A}$ cannot obtain any information about the session key of test instance Π^t from the session key of instance Π. Hence $\Pr(M \wedge \overline{H^*}) \leq \frac{1}{2}$ and $\Pr(M) = \Pr(M \wedge H^*) + \Pr(M \wedge \overline{H^*}) \leq \Pr(M \wedge H^*) + \frac{1}{2}$, 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, we denote by $(A, X), (B, Y), (C, Z)$ the static and ephemeral public keys of users U_A, U_B, U_C in the session id $\mathtt{sid}^t$ of the test instance Π^t. Events can be classified not exclusively as in Table 1 when A, B, C are distinct, as in Table 2 when $A = B \neq C$, as in Table 3 when $A = C \neq B$, as in Table 4 when $A \neq B = C$, and as in Table 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. Thus, the gap BDH problem can be solved with non-negligible probability, and that means we shows that the proposed protocol is secure. We will investigate each of these events in the following subsections.

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

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.

Simulation $\mathcal{S}$ activates $\mathcal{A}$ on this set of users and awaits the actions of $\mathcal{A}$. $\mathcal{S}$ simulate oracle queries as follows.

1. *Send*$(U_i, (\textit{'start'}, \mathtt{P}, U_i, U_j, U_k))$: $\mathcal{S}$ selects ephemeral private key x_i randomly, computes ephemeral public key $X_i = g^{x_i}$, returns $(\mathtt{P}, U_i, U_j, U_k, X_i)$, and records it.
2. *Send*$(\Pi^l_{U_i}, (\mathtt{P}, U_i, U_j, U_k, X_j, X_k))$: If $(\mathtt{P}, U_i, U_j, U_k, X_i)$ 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. *RevealKey*$(\Pi^l_{U_i} = (\mathtt{P}, U_i, S_i, X_i, U_j, S_j, X_j, U_k, S_k, X_k))$: $\mathcal{S}$ 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 $(\sigma_0, \sigma_1, \sigma_2, \sigma_3, \mathtt{sid})$ is recorded in L_H, where $\mathtt{sid}$ is the session id of instance $\Pi^l_{U_i}$, and $\text{BDDH}(X_i + \mathrm{F}_i S_i, X_j + S_j, X_k + S_k, \sigma_0) = 1$, $\text{BDDH}(X_i + S_i, X_j + \mathrm{F}_j S_j, X_k + S_k, \sigma_1) = 1$, $\text{BDDH}(X_i + S_i, X_j + S_j, X_k + \mathrm{F}_k S_k, \sigma_2) = 1$, $\text{BDDH}(X_i + \mathrm{F}_i S_i, X_j + \mathrm{F}_j S_j, X_k + \mathrm{F}_k S_k, \sigma_3) = 1$, $\mathcal{S}$ 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. $\mathtt{H}(\sigma_0, \sigma_1, \sigma_2, \sigma_3, \mathtt{sid} = (\mathtt{P}, U_i, S_i, X_i, U_j, S_j, X_j, U_k, S_k, X_k))$: $\mathcal{S}$ maintains list L_H of query $(\sigma_0, \sigma_1, \sigma_2, \sigma_3, \mathtt{sid})$ and answered hash value K.
   (a) If $\mathtt{sid}$ is the session id of the test instance $\Pi^t = (\mathtt{P}, U_A, A, X = U, U_B, B = V, Y, U_C, C = W, Z)$, and $\text{BDDH}(X + DA, Y + B, Z + C, \sigma_0) = 1$, $\text{BDDH}(X + A, Y + EB, Z + C, \sigma_1) = 1$, $\text{BDDH}(X + A, Y + B, Z + FC, \sigma_2) = 1$, $\text{BDDH}(X + DA, Y + EB, Z + FC, \sigma_3) = 1$, and U_A is honest, i.e., $\mathcal{S}$ knows $a = \log(A)$, then $\mathcal{S}$ stops and is successful by outputting answer of gap BDH problem $((\sigma_0'^{-1}\sigma_1')^{-1}\sigma_2'^{-1}\sigma_3')^{1/((E-1)(F-1))} = \text{BDH}(X, B, C)$, where $\sigma_0' = \hat{e}(Y + B, Z + C)^{-Da}\sigma_0$, $\sigma_1' = \hat{e}(Y + EB, Z + C)^{-a}\sigma_1$, $\sigma_2' = \hat{e}(Y + B, Z + FC)^{-a}\sigma_2$, $\sigma_3' = \hat{e}(Y + EB, Z + FC)^{-Da}\sigma_3$, and $D = \mathrm{F}_A = \mathtt{H}_\mathtt{e}(X), E = \mathrm{F}_B = \mathtt{H}_\mathtt{e}(Y), F = \mathrm{F}_C = \mathtt{H}_\mathtt{e}(Z)$.
   (b) Else if $(\sigma_0, \sigma_1, \sigma_2, \sigma_3, \mathtt{sid})$ is recorded in L_H, $\mathcal{S}$ returns recorded hash value K.
   (c) Else if instance $\Pi^l_{U_i}$ is recorded in L_S, where $\Pi^l_{U_i}$ is an instance with session id $\mathtt{sid}$, and $\text{BDDH}(X_i + \mathrm{F}_i S_i, X_j + S_j, X_k + S_k, \sigma_0) = 1$, $\text{BDDH}(X_i + S_i, X_j + \mathrm{F}_j S_j, X_k + S_k, \sigma_1) = 1$, $\text{BDDH}(X_i + S_i, X_j + S_j, X_k + \mathrm{F}_k S_k, \sigma_2) = 1$, $\text{BDDH}(X_i + \mathrm{F}_i S_i, X_j + \mathrm{F}_j S_j, X_k + \mathrm{F}_k S_k, \sigma_3) = 1$, $\mathcal{S}$ returns recorded session key K and records it in L_H.
   (d) Otherwise, $\mathcal{S}$ returns random hash value K, and records it in L_H.
5. $\mathtt{H}_\mathtt{e}(X_i)$: $\mathcal{S}$ simulates random oracle in the usual way.
6. *RevealState*$(\Pi^l_{U_i})$: 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. *RevealStaticKey*(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. *AddUser*(U_i, S): $\mathcal{S}$ responds to the query faithfully.
9. *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.

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 $\frac{1}{n^3 s}$. 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 $\mathtt{H}$ with $\mathrm{BDH}(X+DA, Y+B, Z+C)$, $\mathrm{BDH}(X+A, Y+EB, Z+C)$, $\mathrm{BDH}(X+A, Y+B, Z+FC)$, and $\mathrm{BDH}(X+DA, Y+EB, Z+FC)$. Therefore $\mathcal{S}$ is successful as described in Step 4a and does not abort as in Step 10.

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

A.2 Other Events

Event $E_{1b} \wedge M^*$ Same as the event $E_{1a} \wedge M^*$ in Subsection A.1, except the following points. In Setup, $\mathcal{S}$ embeds gap BDH instance (U, V, W) as $A = U, B = V, C = W$. In Simulation of $\mathtt{H}$, $\mathcal{S}$ extracts $\mathrm{BDH}(U, V, W)$ as follows: $((\sigma_0'^{-1}\sigma_1')^{-1}\sigma_2'^{-1}\sigma_3')^{1/((E-1)(F-1))} = \mathrm{BDH}(A, B, C)$, where $\sigma_0' = (\hat{e}(Y+B, Z+C)^{-x}\sigma_0)^{1/D}$, $\sigma_1' = \hat{e}(Y+EB, Z+C)^{-x}\sigma_1$, $\sigma_2' = \hat{e}(Y+B, Z+FC)^{-x}\sigma_2$, $\sigma_3' = (\hat{e}(Y+EB, Z+FC)^{-x}\sigma_3)^{1/D}$.

Event $E_{2a} \wedge M^*$ Same as the event $E_{1a} \wedge M^*$ in Subsection A.1, except the following points. In Setup, $\mathcal{S}$ embeds gap BDH instance (U, V, W) as $X = U, Y = V, Z = W$. In Simulation of $\mathtt{H}$, $\mathcal{S}$ extracts $\mathrm{BDH}(U, V, W)$ as follows: $((\sigma_0'^{E}\sigma_1'^{-1})^{F}(\sigma_2'^{E}\sigma_3'^{-1})^{-1})^{1/((E-1)(F-1))} = \mathrm{BDH}(X, Y, Z)$, where $\sigma_0' = \hat{e}(Y+B, Z+C)^{-Da}\sigma_0$, $\sigma_1' = \hat{e}(Y+EB, Z+C)^{-a}\sigma_1$, $\sigma_2' = \hat{e}(Y+B, Z+FC)^{-a}\sigma_2$, $\sigma_3' = \hat{e}(Y+EB, Z+FC)^{-Da}\sigma_3$.

Event $E_{2b} \wedge M^*$ Same as the event $E_{1a} \wedge M^*$ in Subsection A.1, except the following points. In Setup, $\mathcal{S}$ embeds gap BDH instance (U, V, W) as $A = U, Y = V, Z = W$. In Simulation of $\mathtt{H}$, $\mathcal{S}$ extracts $\mathrm{BDH}(U, V, W)$ as follows: $((\sigma_0'^{E}\sigma_1'^{-1})^{F}(\sigma_2'^{E}\sigma_3'^{-1})^{-1})^{1/((E-1)(F-1))} = \mathrm{BDH}(A, Y, Z)$, where $\sigma_0' = (\hat{e}(Y+B, Z+C)^{-x}\sigma_0)^{1/D}$, $\sigma_1' = \hat{e}(Y+EB, Z+C)^{-x}\sigma_1$, $\sigma_2' = \hat{e}(Y+B, Z+FC)^{-x}\sigma_2$, $\sigma_3' = (\hat{e}(Y+EB, Z+FC)^{-x}\sigma_3)^{1/D}$.

Event $E_{3a} \wedge M^*$ Same as the event $E_{1a} \wedge M^*$ in Subsection A.1, except the following points. In Setup, $\mathcal{S}$ embeds gap BDH instance (U, V, W) as $X = U, B = V, Z = W$. In Simulation of $\mathtt{H}$, $\mathcal{S}$ extracts $\mathrm{BDH}(U, V, W)$ as follows: $((\sigma_0'^{-1}\sigma_1')^{F}(\sigma_2'^{-1}\sigma_3')^{-1})^{1/((E-1)(F-1))} = \mathrm{BDH}(X, B, Z)$, where $\sigma_0' = \hat{e}(Y+B, Z+C)^{-Da}\sigma_0$, $\sigma_1' = \hat{e}(Y+EB, Z+C)^{-a}\sigma_1$, $\sigma_2' = \hat{e}(Y+B, Z+FC)^{-a}\sigma_2$, $\sigma_3' = \hat{e}(Y+EB, Z+FC)^{-Da}\sigma_3$.

Event $E_{3b} \wedge M^*$ Same as the event $E_{1a} \wedge M^*$ in Subsection A.1, except the following points. In Setup, $\mathcal{S}$ embeds gap BDH instance (U, V, W) as $A = U, B = V, Z = W$. In Simulation of $\mathtt{H}$, $\mathcal{S}$ extracts $\mathrm{BDH}(U, V, W)$ as follows: $((\sigma_0'^{-1}\sigma_1')^{F}(\sigma_2'^{-1}\sigma_3')^{-1})^{1/((E-1)(F-1))} = \mathrm{BDH}(A, B, Z)$, where $\sigma_0' = (\hat{e}(Y+B, Z+C)^{-x}\sigma_0)^{1/D}$, $\sigma_1' = \hat{e}(Y+EB, Z+C)^{-x}\sigma_1$, $\sigma_2' = \hat{e}(Y+B, Z+FC)^{-x}\sigma_2$, $\sigma_3' = (\hat{e}(Y+EB, Z+FC)^{-x}\sigma_3)^{1/D}$.

Event $E_{3'a} \wedge M^*$ and $E_{3'b} \wedge M^*$ Event $E_{3'a} \wedge M^*/E_{3'b} \wedge M^*$ can be handled same as event $E_{3a} \wedge M^*/E_{3b} \wedge M^*$ in Subsection A.2/A.2, because of symmetry of B and C.

A.3 Other Cases

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

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

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

In the case of $A = B = C$, events E_{1b}^3, E_{2a}^3 in Table 5 can be handled same as events E_{1b}, E_{2a} in Table 1, with condition $A = B = C$.

	A	X	B	Y	C	Z	succ. prob.
E_{1a}	r	ok	ok	r/n	ok	r/n	$p_{1a}/n^3 s$
E_{1b}	ok	r	ok	r/n	ok	r/n	p_{1b}/n^3
E_{2a}	r	ok	r	ok	r	ok	$p_{2a}/n^3 s^3$
E_{2b}	ok	r	r	ok	r	ok	$p_{2b}/n^3 s^2$
E_{3a}	r	ok	ok	r/n	r	ok	$p_{3a}/n^3 s^2$
E_{3b}	ok	r	ok	r/n	r	ok	$p_{3b}/n^3 s$
$E_{3'a}$	r	ok	r	ok	ok	r/n	$p_{3'a}/n^3 s^2$
$E_{3'b}$	ok	r	r	ok	ok	r/n	$p_{3'b}/n^3 s$

Table 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. "succ. prob." row shows the probability of success of solver $\mathcal{S}$, where $p_{xy} = Pr(E_{xy} \wedge M^*)$ and n and s are the number of users and instances.

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

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

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

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

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

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

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

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