International Symposium on Information Theory and its Applications, ISITA2008
Auckland, New Zealand, 7-10, December, 2008

A Group-Key Agreement Protocol Secure against The Malleability Attacks

Seiko Arita† and Akihiko Sakuma‡

Institute of Information Security
2-14-1 Tsuruya-cho Kanagawa-ku Yokohama-shi Kanagawa, JAPAN
† E-mail: arita@iisec.ac.jp ‡ E-mail: dgs074106@iisec.ac.jp

Abstract. This paper treats group key agreement protocols (GKAP). In group key agreement, it is needed and important to consider existence of malicious insiders. In this paper, we show a GKAP that is UC-secure even if some of group members behave maliciously and attempt the malleability attacks that is the adversary contorol the session-key as desired. So our group key agreement protocol does not allow the adversary to control the session-key, and maintain not only the key freshness, but also the key authentication and the key integrity. Our protocol is efficient enough for practical uses, being composed of 2 rounds and n messages for n-party groups.

1. Introduction

Background: A group key agreement protocol (GKAP, in short) is one of the most important and primitive cryptographic protocols. Recently, more concerns have been emerging of GKAP [1]. Informally, GKAP has three basic requirements: key freshness, key integrity and key authentication. The key freshness requires a generated session-key by a session of a protocol to be independent from other session-keys generated by other sessions of the protocol. The key integrity means all session-keys of honest parties resulting from a session of a protocol must be the same. And the key authentication requires the secrecy of session-keys, i.e., the outsiders cannot only obtain the value of the session-keys but also any information of session-keys by eavesdropping conversations of GKAP among the parties.

Historically, a key agreement protocol (KAP) is first considered and made use of in a two-party case, such as the famous Diffie-Hellman KAP. The formal security of the two-party KAP is defined and proved by Bellare and Rogaway [2]. [2] defines a notion of Session-key indistinguishability (*SK-security*) of two-party KAP.

Boyd and Nieto [3] showed a natural generalization of the SK-security to the cases of GKAP and showed a SK-secure GKAP (we call the protocol BN), assuming a broadcast channel. However, later it turned out that the SK-security is not sufficient for security of GKAP. We need to consider insider attacks more carefully in the GKAP case. For example, it is possible that malicious insiders can make a selected honest party *alone* in GKAP, resulting violation of the key integrity.

Katz and Shin [6] cast the security of GKAP into the framework of Universal Composability (UC)[4] in order to consider the effects of insider attacks against GKAP. UC-framework defines security of cryptographic protocols by ensuring that the target protocol mimics the behavior of a corresponding ideal trusted third party, called an ideal functionality, under any adversarial environment. [6] defines an ideal functionality $\mathcal{F}_{KS05}$ of GKAP and gives a generic compiler that takes a SK-secure GKAP and transforms it into a UC-secure GKAP. However, If any party is corrupted, the ideal functionality $\mathcal{F}_{KS05}$ allows the malicious insiders to choose a value of the session-key as they want, as long as it is the same for all parties in the group. So, the key freshness is violated.

The Malleability Attacks: Desmedt et al.[5] shows *the malleability attack* that aims to 'cancel' the contributions of the honest parties to and violate the key freshness. and generate the session key as desired. The attack proceeds as follows. In the protocol proceeding, the corrupted parties (i.e., malicious insiders) wait for the honest parties to send messages containing their contribution to the session-key. The corrupted parties use this information to compute their 'bad' protocol messages adaptively as some function of the honest party's messages, in order to 'cancel' the contributions of the honest players and fix the session-key as desired. It means that the key freshness is violated.

Such an attack by the malicious insiders can allow a collaborating outside eavesdropper to know some information on the session key even if the insiders are shielded. Now we describe one scenario of *the collaborating Attacks*[5]. The collaborating attacks proceed

as follows. The adversary $\mathcal{A}$ is a pair of the malicious insider A_I and the outside eavesdropper A_O. Before the GKAP execution, $\mathcal{A} = (A_I, A_O)$ (the insider and the outsider) chooses the session-key as desired. In the protocol proceeding, the insider generates his 'bad' protocol messages in order to compute the session-key as desired. After the GKAP ends, honest parties establish the private channel using the session-key computed by the GKAP. Since the outside eavesdropper A_O knows the session-key, he can open this private channel.

Here it is important that such attacks may be undetectable by the honest parties. The malleability attack may apply to GKAPs that securely realize the Katz and Shin's functionality $\mathcal{F}_{KS05}$, simply because $\mathcal{F}_{KS05}$ does not consider any malleability attacks at all, as noted above.

Our contribution: In this paper, we define an ideal functionality of GKAP that takes the malleability attacks into consideration. Then, we show a GKAP that realizes that ideal functionality. Our GKAP is the first one that maintains UC-security in a meaningful way even if malicious insiders attempt the malleability attacks. It is efficient enough for practical uses, composed of 2 rounds and n messages for n-party groups.

Setup: We assume the following setup throughout the paper. 1. The Broadcast Channel: Parties use cryptographic broadcast channel to send and receive messages. 2. The authenticated-link model (AM): An adversary cannot inject or change messages without being detected. 3. The common reference strings (CRS). 4. A static adversary : The adversary may corrupt some parties only at the beginning of protocols.

2. Adversary Model.

We use the shielded-insider model [5] to capture security against corrupted insiders collaborating with an outside eavesdropper. The insiders can communicate with the outsider only until the protocol begins, after which the insiders are shielded, i.e. prevented from further communication with the outsider. Then insiders can initiate several protocols to exchange session-keys with honest parties. The insiders try to choose their protocol messages in order to correlate the session-keys computed by honest parties. To defeat the freshness, the insider's goal is to cause some honest parties to compute the session-keys as desired.

Attack Model: The attacker $\mathcal{A}$ is modelled by a pair $(\mathrm{A_I}, \mathrm{A_O})$ of the collaborating insiders and outsiders, where $\mathrm{A_I}$ is the shielded-insider and $\mathrm{A_O}$ is the outside eavesdropper, respectively. The attack model is described by the game that has three stages and runs as follow.

The Key-Control Game

1. **Stage 1 (Initialization):** $\mathrm{A_O}$ chooses the session-key κ randomly from the key space (At this point, $\mathrm{A_O}$ dose not know the CRS of the target session). $\mathrm{A_O}$ sends κ to $\mathrm{A_I}$.
2. **Stage 2 (Protocol Executions):** (After this point, $\mathrm{A_O}$ can not communicate with $\mathrm{A_I}$.) The malicious insider $\mathrm{A_I}$ generates and sends 'bad' protocol messages in order to make some honest party computes κ as the session-key.
3. **Stage 3 (Adversary goal):** We say that the adversary $\mathcal{A}(= \mathrm{A_I}, \mathrm{A_O})$ wins, if there exist some honest party computes κ as the session-key. Otherwise (no party computes κ or all of the honest party aborts) $\mathcal{A}$ lose.

Now we define some security notion. First we define the security against the malleability attacks.

Definition 1 : [The security against The Malleability Attacks.] *A GKAP* Π *is secure against the malleability attacks, if any adversary* $\mathcal{A} = (\mathrm{A_I}, \mathrm{A_O})$ *can not win the key-control game excepting with the negligible probability.*

The Key Integrity Game: We also define about *The Key Integrity Game*. Informally, the malicious insider $\mathrm{A_I}$ generates and sends 'bad' protocol messages in order to make some honest party computes the wrong session-key. Adversary goal is to cause some honest parties to compute the different session-keys. We say that the adversary $\mathcal{A} = \mathrm{A_I}$ wins the game, if there exists some party who computes different session-keys from another party. Otherwise (all of the honest party computes the same session key or aborts) $\mathcal{A}$ lose.

Definition 2 : [The security in the sense of The Key Integrity.] *A GKAP* Π *is secure in the sense of the key integrity, if any adversary* $\mathcal{A} = \mathrm{A_I}$ *can not win the key integrity game excepting with the negligible probability.*

The adversary capability: We assume the following adversary capability throughout the paper. $\mathrm{A_O}$ does not know the CRS (include the public keys) at the Stage 1, because of the GKAP not yet start. Note that since we assume the static adversary, when the GKAP start, some party is already corrupted, but we

stress that A_O does not know the secret key (A_I knows this).

3. An Ideal Functionality $\mathcal{F}_{\mathcal{GKE}}$

This section we definen an Ideal Functionality.

Assumptions on PRF: We use pseudorandom function families with additional properties of collision-resistance. The collision-resistance of f means that no efficient adversary can find two different seeds s, s' resulting in the same values $f_s(v_0) = f_{s'}(v_0)$ of f at a samplable element v_0.

Definition 3 : [A Collision-Resistant PRF [6]] *Let $f = \{f_s\}_s$ be a PRF. We say that f is if there is an efficient procedure* Sample *such that the probability*

$$\Pr[v_0 \leftarrow \text{Sample}(1^k); s, s' \leftarrow A(1^k, v_0) : s, s' \in \{0,1\}^k$$

$$\wedge s \neq s' \wedge f_s(v_0) = f_{s'}(v_0)]$$

is negligible in k for all polynomial-time adversaries A.

[6] proves the existence of a collision-resistance PRF assuming the existence of a one-way permutation.

We define an ideal functionality $\mathcal{F}_{\mathcal{GKE}}$ of GKAP that takes the malleability attacks into consideration, as Figure 1. In the functionality $\mathcal{F}_{\mathcal{GKE}}$, if all parties in a group *gid* are uncorrupted, $\mathcal{F}_{\mathcal{GKE}}$ proceeds just like $\mathcal{F}_{KS05}$. However, if any party is corrupted, $\mathcal{F}_{\mathcal{GKE}}$ proceeds as follows. For uncorrupted parties P_i, $\mathcal{F}_{\mathcal{GKE}}$ chooses a random nonce N_i for P_i, and sends N_i to the ideal-process adversary $\mathcal{S}$. For corrupted parties P_j, $\mathcal{F}_{\mathcal{GKE}}$ waits for $\mathcal{S}$ to send a nonce N_j for the corrupted parties P_j. To generate session-keys κ, $\mathcal{F}_{\mathcal{GKE}}$ computes $\kappa = f_{N_1}(V_1) \oplus f_{N_2}(V_2) \oplus \cdots \oplus f_{N_n}(V_n)$ with $V_i = (sid||ssid||P_i)$ using a pseudorandom function f, and delivers $(sid, ssid, gid, \kappa)$.

Note that against that functionality $\mathcal{F}_{\mathcal{GKE}}$, it is difficult even for malicious insiders (who can select N_j) to control the session-key κ using the malleability attacks, because they cannot cancel the effect of nonces N_i of honest parties by the way in which the pseudorandom function f is used to generate κ.

Now we show the GKAP which securely realizes the ideal functionality $\mathcal{F}_{\mathcal{GKE}}$ is secure against the malleability attacks (satisfy the definition 1).

Proposition 1 *Let $\{f_s\}_s$ be a pseudorandom function (PRF) family. A GKAP which securely realizes the ideal functionality $\mathcal{F}_{\mathcal{GKE}}$ is secure against the malleability attacks*

Proof: Assume the GKAP Π securely realizes the ideal functionality $\mathcal{F}_{\mathcal{GKE}}$, there exists a simulator $\mathcal{S}$ and no environment $\mathcal{Z}$ can distinguish between real and ideal-process model with a non-negligible advantage.

First we construct an environment $\mathcal{Z}$ execution. $\mathcal{Z}$ simulates A_O and waits A_O sends κ. $\mathcal{Z}$ executes a GKAP Π with input $(sid, ssid, gid)$ collaborating with an adversary $\mathcal{A}$ with input κ. The uncorrupted party outputs the session-key κ', if $\kappa' = \kappa$ then $\mathcal{Z}$ output real model else ideal-process model.

Suppose there exists a simulator $\mathcal{S}$ described above, we show a inconsistency to construct a seed inverter $\mathcal{B}$ for PRF using such $\mathcal{S}$. A seed inverter $\mathcal{B}$ function as follows. $\mathcal{B}$ sends the value v to the challenger $\mathcal{C}$. $\mathcal{C}$ computes $f_N(v)$ and sends to $\mathcal{B}$. $\mathcal{B}$ guess the seed N.

The seed inverter $\mathcal{B}$ proceeds as follows with input a security parameter k and $w = (sid, ssid, gid)$. $\mathcal{B}$ chooses a currupted party P_j^* from *gid* randomly. $\mathcal{B}$ sends the value $v = (sid||ssid||P_j^*)$ to the challenger $\mathcal{C}$. $\mathcal{C}$ returns $\kappa = f_N(v)$ to $\mathcal{B}$.

$\mathcal{B}$ choose random nonces for uncorrupted parties: $N_i \xleftarrow{R} \{0,1\}^k (i = 1, \ldots, n(\neq j))$. Then $\mathcal{B}$ executes $\mathcal{Z}$ with input $w = (sid, ssid, gid)$ and $\kappa \oplus f_{N_i}(V_i)$ together with the simulator $\mathcal{S}$ with input $\kappa \oplus f_{N_i}(V_i)$. $\mathcal{B}$ sends nonces $N_i (i = 1, \ldots, n(\neq j))$ for uncorrupted parties. If $\mathcal{S}$ outputs a nonce N_j, then $\mathcal{B}$ outputs N_j as the seed else aborts.

Since $\mathcal{S}$ can output N_j s.t. $\kappa = f_{N_i}(V_i) \oplus f_{N_j}(V_j)$ $(i = 1, \ldots, n(\neq j))$ with a non-negligible advantage, if $\mathcal{B}$ succeed to guess corrupted party, $\mathcal{B}$ can find the seed of PRF. It is a inconsistency. □

Proposition 2 *The GKAP which securely realizes the ideal functionality $\mathcal{F}_{\mathcal{GKE}}$ is secure in the sense of the key integrity.*

Note that in the ideal functionality $\mathcal{F}_{\mathcal{GKE}}$ the key integrity is always ensured even if some parities get corrupted. We can prove this same case of $\mathcal{F}_{KS05}$ [6].

Here, we consider realizability of the functionality $\mathcal{F}_{\mathcal{GKE}}$ in short. It is impossible for 1 round, 1 broadcast-message protocol to securely realize $\mathcal{F}_{\mathcal{GKE}}$. (If the initiator is corrupted, there is no way to prevent the malleability attacks.) We can construct 1 round, n broadcast-messages protocol that securely realizes $\mathcal{F}_{\mathcal{GKE}}$ by using some non-interactive zero-knowledge proof, but it is not efficient. In the below, we show there exists an efficient 2 round, n broadcast-messages protocol that securely realizes the functionality $\mathcal{F}_{\mathcal{GKE}}$.

4. A proposed GKAP

This section shows a 2 round, n broadcast-messages GKAP that is secure against the malleability attacks. Recall we assume: 1. The Broadcast Channel, 2. The

Figure 1: The ideal functionality $\mathcal{F}_{\mathcal{GKE}}$ of GKAP.

Let k be a security parameter, and $\{f_s\}_s$ be a pseudorandom function (PRF) family. *sid* and *ssid* denote a *session id* and *sub-session id*, respectively, and $gid = \{P_1, P_2, \ldots, P_n\}$ denotes a group of n parties. The ideal functionality $\mathcal{F}_{\mathcal{GKE}}$ proceeds as follows with an ideal-process adversary $\mathcal{S}$.

1. Initialization: Receiving $(sid, ssid, gid,$ new-subsession$)$ from P_i for the first time on the subsession *ssid*, do:
   - $\mathcal{F}_{\mathcal{GKE}}$ records $(sid, ssid, gid, P_i)$ and sends this to $\mathcal{S}$.
2. Key Generation: $\mathcal{F}_{\mathcal{GKE}}$ sends a message $(sid, ssid, gid,$ ready$)$ to $\mathcal{S}$. Upon receiving a message $(sid, ssid, gid,$ ok$)$ from $\mathcal{S}$, do:
   - If all parties in *gid* are uncorrupted, $\mathcal{F}_{\mathcal{GKE}}$ chooses the session key: $\kappa \xleftarrow{R} \{0,1\}^k$ and stores $(sid, ssid, gid, \kappa)$.
   - If any party in *gid* is corrupted,
     - For uncorrupted parties P_i, $\mathcal{F}_{\mathcal{GKE}}$ chooses random nonces N_i for P_i and sends them to $\mathcal{S}$.
     - For corrupted parties P_j, $\mathcal{F}_{\mathcal{GKE}}$ waits for $\mathcal{S}$ to send nonces N_j for P_j and stores them.

   $\mathcal{F}_{\mathcal{GKE}}$ computes $\kappa = f_{N_1}(V_1) \oplus f_{N_2}(V_2) \oplus \cdots f_{N_n}(V_n)$ with $V_i = (sid||ssid||P_i)$ and stores $(sid, ssid, gid, \kappa)$.
3. Key Delivery: If $\mathcal{S}$ sends a message $(\text{deliver}, sid, ssid, gid, P_i)$, then send $(sid, ssid, gid, \kappa)$ to party P_i.

Figure 2: The proposed Group Key Agreement Protocol 1 in the AM.

Let k be a security parameter, $\pi = \{K, E, D\}$ be a public-key encryption scheme, and $\{f_s\}_s$ be a PRF family with a public element v_0 in the domain of f. *sid* and *ssid* denote a *session id* and *sub-session id*, respectively, and $gid = \{P_1, P_2, \ldots, P_n\}$ denotes a group of n parties. Each P_i generates its public and private key pair: $(e_i, d_i) \leftarrow K(1^k)$. The protocol with input $sid, ssid, gid$ proceeds as follows.

1. The initiator P_1 choose a k-bit random nonce N_1, and encrypt the nonce for each parties in *gid*, respectively: $c_j \leftarrow E_{e_j}(sid||ssid||N_1)$ $(j = 2, \ldots, n)$. Then, P_1 sends $(sid, ssid, gid, P_1, (c_2||\cdots||c_n))$ to parties in *gid*. P_1 computes $\sigma_1 = f_{N_1}(v_0)$ and stores this.
2. The responder P_j $(j = 2, \ldots, n)$ decrypts his parts of ciphertexts c_j to get the nonce N_1: $(sid||ssid||N_1) \leftarrow D_{d_j}(c_j)$. (Here, P_j verifies *sid* and *ssid*.) Then, P_j chooses a random k-bit nonce N_j, and computes $\sigma_j = f_{N_1}(v_0)$. P_j sends $(sid, ssid, gid, P_j, N_j, \sigma_j)$ to parties in *gid*.
3. Each P_k $(k = 1, \ldots, n)$ checks the equality of all of the σ-values: $\sigma_k = \sigma_l$ $(\forall l (\in \{2, \ldots, n\}) \neq k)$. If so, P_k computes and outputs $\kappa = f_{N_1}(V_1) \oplus f_{N_2}(V_2) \oplus \cdots \oplus f_{N_n}(V_n)$ with $V_i = (sid||ssid||P_i)$.

authenticated-link model (AM), 3. The CRS model, 4. An Static adversary.

The basic strategy of our protocol is as follows. 1. We enhance the Boyd and Nieto protocol[3], that is the most efficient SK-secure GKAP assuming a broadcast channel. 2. We use some pseudorandom functions to generate MACs of the initiator's nonce to maintain the key integrity even under malicious insiders. 3. We also use some pseudorandom functions in a way to maintain the effects of nonces of honest parties even under malicious insiders that attempt the malleability attack.

Figure 2 shows the proposed GKAP in the AM. The initiator P_1 selects a k-bit random nonce N_1 and encrypts it for every responders, and sends $(sid, ssid, gid, P_1, (c_2||\cdots||c_n))$ to all parties in *gid*. P_1 computes and stores $\sigma_1 = f_{N_1}(v_0)$.

Upon receiving the ciphertext from P_1, the responders P_j decrypt the corresponding parts to get N_1: $(sid||ssid||gid||N_1) \leftarrow D_{d_j}(c_j)$. Then, P_j choose a random k-bit nonce N_j, computes $\sigma_j = f_{N_1}(v_0)$ and sends $(sid, ssid, gid, P_j, N_j, \sigma_j)$ to parties in *gid*. Receiving all the messages, each party P_k in *gid* verifies the integrity of the nonce N_1 using the σ-values as in Figure 2. Only if verified correctly, P_k computes and outputs the session-key as $\kappa = f_{N_1}(V_1) \oplus f_{N_2}(V_2) \oplus \cdots \oplus f_{N_n}(V_n)$ with $V_i = (sid||ssid||P_i)$.

Here we analyze the security of the GKAP, intuitively. Since the initiator P_1 is encrypting nonce N_1, the generated session-key κ is expected to be kept secret from an eavesdropper (key authentication). Since the integrity of the received nonce N_1 is verified through the σ-values, all honest parties should output the same κ or abort (If $f_{N_1}(v_0) = f_{M_1}(v_0)$ then $N_1 = M_1$ by the collision-resistance of f). So, the GKAP would preserve the key integrity under malicious insiders. The session-key κ is generated using the

nonces N_i of every parties P_i, and the way of combining them as $\kappa = f_{N_1}(V_1) \oplus f_{N_2}(V_2) \oplus \cdots \oplus f_{N_n}(V_n)$ would guarantee that even malicious insiders cannot remove the effect of the nonces of honest parties. Thus, the GKAP would preserve security under malicious insiders who attempt the malleability attacks.

5. UC-security of The proposed GKAP1

In fact, we can prove the following theorem:

Theorem 1 *Assume the broadcast channel. If encryption scheme π is indistinguishable under adaptively chosen-ciphertext attack and $\{f_s\}_s$ is a collision-resistant PRF, then the group key agreement protocol (Figure 2) securely realizes the functionality $\mathcal{F}_{\mathcal{GKE}}$ of GKAP against a static adversary under the authenticated-link model (AM).*

To prove Theorem 1, for an arbitrary static real-life adversary $\mathcal{A}$ against the protocol, we have to show existence of an ideal-process adversary (the simulator) $\mathcal{S}$ such that no environment $\mathcal{Z}$ can tell whether it interacts with $\mathcal{A}$ and parties running the GKAP in the real world, or with $\mathcal{S}$ and (dummy) parties communicating with $\mathcal{F}_{\mathcal{GKE}}$ in the ideal process.

Due to lack of space, we can only sketch the work of the simulator $\mathcal{S}$ in this abstract. The full proof will appear in the full version.

The simulator $\mathcal{S}$: The simulator $\mathcal{S}$ proceeds as follows. $\mathcal{S}$ internally invokes a copy of $\mathcal{A}$. Messages from $\mathcal{Z}$ to $\mathcal{S}$ are forwarded to $\mathcal{A}$, and messages from $\mathcal{A}$ to its environment are forwarded to $\mathcal{Z}$. $\mathcal{S}$ generates public and private key pairs for all parties and an element v_0 in the domain of f. $\mathcal{S}$ gives the resulting $(e_1, \ldots, e_n)$ and v_0 to $\mathcal{A}$.

1. If $\mathcal{A}$ does not corrupt any party in *gid*, $\mathcal{S}$ chooses k-bit random nonces for all parties: $N_1, \cdots, N_n \stackrel{R}{\leftarrow} \{0,1\}^k$. Then, $\mathcal{S}$ simulates messages of initiator P_1 as $c_j \leftarrow E_{e_j}(sid||ssid||N_1)$ for $j = 2, \ldots, n$. $\mathcal{S}$ sends the message $(sid, ssid, gid, P_1, (c_1|| \cdots ||c_n))$ to $\mathcal{A}$ on the name of P_1 and waits for $\mathcal{A}$ sending this message to all parties in *gid*. When $\mathcal{A}$ delivers the ciphertexts of P_1 to P_j, $\mathcal{S}$ computes $\sigma_j = f_{N_1}(v_0)$, and sends the message $(sid, ssid, gid, P_j, N_j, \sigma_j)$ to $\mathcal{A}$ on the name of P_j and waits for $\mathcal{A}$ sending the message to all parties.

2. In the case that some parties in *gid* get corrupted, $\mathcal{S}$ proceeds as follows. If the initiator P_1 is uncorrupted, $\mathcal{S}$ waits an initiator's nonce N_1 from $\mathcal{F}_{\mathcal{GKE}}$. Upon receiving a nonce N_1 from $\mathcal{F}_{\mathcal{GKE}}$, $\mathcal{S}$ computes $c_j \leftarrow E_{e_j}(sid||ssid||N_1)$ for $j = 2, \ldots, n$. Then $\mathcal{S}$ sends a message $(sid, ssid, gid, P_1, (c_1||c_2|| \cdots ||c_n))$ to $\mathcal{A}$ and waits for $\mathcal{A}$ sending this message to all parties in *gid*. If responders P_j are uncorrupted, $\mathcal{S}$ waits a P_j's nonce N_j from $\mathcal{F}_{\mathcal{GKE}}$. Upon receiving a message $(sid, ssid, gid, P_j, N_j)$ from $\mathcal{F}_{\mathcal{GKE}}$, $\mathcal{S}$ computes $\sigma_j = f_{N_1}(v_0)$ and sends a message $(sid, ssid, gid, P_j, N_j, \sigma_j)$ to $\mathcal{A}$ and waits for $\mathcal{A}$ sending this message to all parties in *gid*. In the case that the initiator P_1 gets corrupted and outputs a message $(sid, ssid, gid, P_1, (c_1||c_2|| \cdots ||c_n))$, $\mathcal{S}$ decrypts the ciphertexts with the corresponding secret keys, then checks all of the decrypted messages are $(sid||ssid||N_1)$, and sends the nonce N_1 to $\mathcal{F}_{\mathcal{GKE}}$. When corrupted responder P_j outputs a message $(sid, ssid, gid, P_j, N_j, \sigma_j)$, $\mathcal{S}$ sends the nonce N_j to $\mathcal{F}_{\mathcal{GKE}}$.

Upon receiving a message $(sid, ssid, gid, \text{ready})$ from $\mathcal{F}_{\mathcal{GKE}}$, $\mathcal{S}$ sends a message $(sid, ssid, gid, \text{ok})$ to $\mathcal{F}_{\mathcal{GKE}}$. When $\mathcal{A}$ delivers all of the necessary messages to P_j, $\mathcal{S}$ sends $(\text{deliver}, sid, ssid, gid, P_j)$ to $\mathcal{F}_{\mathcal{GKE}}$ to deliver a session-key to P_j.

Analysis of Simulation: We use some of hybrid game to show the above simulator $\mathcal{S}$ works well.

First we construct the hybrid simulator 1 (Hyb1), in which the simulator $\mathcal{S}_{\text{Hyb1}}$ knows values of nonces $N_1, \ldots, N_n$ where $f_{N_1}(V_1) \oplus f_{N_2}(V_2) \oplus \cdots \oplus f_{N_n}(V_n) = \kappa$ of session-keys of honest sub-sessions by some transcendental means. So, the simulation in honest sub-sessions is trivially perfect: it merely mimics the behavior of honest parties using the knowledge of the session-key κ. In corrupted sub-sessions, $\mathcal{S}_{\text{Hyb1}}$ works in the same way as $\mathcal{S}$ in the ideal-process model, checking the integrity of nonces under initiator's ciphertexts directly using the secret keys of simulated parties. (In the real-life, receivers check the integrity by using σ_i's.) We will show Hyb1 is indistinguishable from the real-life model by using collision-resistance of PRF f.

Second we construct a hybrid model 2 (Hyb2), in which the simulator $\mathcal{S}_{\text{Hyb2}}$ also knows values of nonces $N_1, \ldots, N_n$ of honest sub-sessions. However, $\mathcal{S}_{\text{Hyb2}}$ encrypts a new k-bit random string M_1 instead of N_1 to form a message of initiator in simulating honest sub-sessions. This M_1 is completely independent of the session key κ. We will show Hyb2 is indistinguishable from Hyb1 by using indistinguishability of the encryption scheme $\{K, E, D\}$.

Finally we compare Hyb2 with the ideal-process model. $\mathcal{S}_{\text{Hyb2}}$ computes σ_i using N_1 as the seed of a PRF, where N_1 is ingredient of the session-key κ. On the while, $\mathcal{S}$ in the ideal-process model computes σ_i using M_1 as the seeds, where M_1 is independent from κ. We will show the ideal-process model is indistinguishable from Hyb2 by using a assumption of a PRF f.

Figure 3: The proposed Group Key Agreement Protocol 2 in the UM.

Let k be a security parameter, $\pi = \{K, E, D\}$ be a public-key encryption scheme, $\Sigma = \{Gen, Sgn, Vrfy\}$ be a signature scheme, and $\{f_s\}_s$ be a PRF family with a public element v_0 in the domain of f. *sid* and *ssid* denote a *session id* and *sub-session id*, respectively, and $gid = \{P_1, P_2, \ldots, P_n\}$ denotes a group of n parties. Each P_i generates its public/private key pair and signature/verify key pair: $(e_i, d_i) \leftarrow K(1^k)$, $(SK_i, PK_i) \leftarrow Gen(1^k)$. The protocol with input $sid, ssid, gid$ proceeds as follows.

1. The initiator P_1 choose a k-bit random nonce N_1, and encrypt the nonce for each parties in *gid*, respectively: $c_j \leftarrow E_{e_j}(sid||ssid||N_1)$ $(j = 2, \ldots, n)$. P_1 computes his signature τ_1: $\tau_1 \leftarrow Sgn_{SK_1}(sid||ssid||P_1||c_2||\cdots||c_n)$. Then, P_1 sends $(sid, ssid, gid, P_1, (c_2||\cdots||c_n), \tau_1)$ to parties in *gid*. P_1 computes $\sigma_1 = f_{N_1}(v_0)$ and stores this.
2. The responder P_j $(j = 2, \ldots, n)$ verifie the signature τ_1 and decrypt his parts of ciphertexts c_j to get the nonce N_1: $Vrfy_{PK_1}((sid||ssid||P_1||c_2||\cdots||c_n), \tau_1) = 1$, $(sid||ssid||N_1) \leftarrow D_{d_j}(c_j)$. (Here, P_j verifies *sid* and *ssid*.) P_j chooses a random k-bit nonce N_j, and computes $\sigma_j = f_{N_1}(v_0)$. P_j computes his signature τ_j: $\tau_j \leftarrow Sgn_{SK_j}(sid||ssid||P_j||N_j||\sigma_j)$. Then P_j sends $(sid, ssid, gid, P_j, N_j, \sigma_j, \tau_j)$ to parties in *gid*.
3. Each P_k $(k = 1, \ldots, n)$ verifies signatures τ_k $(k = 2, \ldots, n)$: $Vrfy_{PK_k}((sid||ssid||P_k||N_k||\sigma_k), \tau_k) = 1$, and checks the equality of all of the σ-values: $\sigma_k = \sigma_l$ $(\forall l (\in \{2, \ldots, n\}) \neq k)$. If so, P_k computes and outputs $\kappa = f_{N_1}(V_1) \oplus f_{N_2}(V_2) \oplus \cdots \oplus f_{N_n}(V_n)$ with $V_i = (sid||ssid||P_i)$.

6. A proposed GKAP2 in the UM

This section shows our GKAP in the authenticated-link model (AM). The protocol can be enhanced to be secure also in the unauthenticated-link model (UM) by using digital signature schemes.

Figure 3 shows the proposed GKAP in the UM. The initiator P_1 selects a k-bit random nonce N_1 and encrypts it for every responders, and computes his signature τ_1 then P_1 sends $(sid, ssid, gid, P_1, (c_2||\cdots||c_n), \tau_1)$ to all parties in *gid*. P_1 computes and stores $\sigma_1 = f_{N_1}(v_0)$.

Upon receiving the message from P_1, the responders P_j verifie the signature τ_1 and decrypt the corresponding parts to get N_1: $(sid||ssid||gid||N_1) \leftarrow D_{d_j}(c_j)$. Then, P_j choose a random k-bit nonce N_j, computes $\sigma_j = f_{N_1}(v_0)$ and his signature τ_j. Then P_j sends $(sid, ssid, gid, P_j, N_j, \sigma_j, \tau_j)$ to parties in *gid*. Receiving all the messages, each party P_k in *gid* verifies all responder's signatures and the integrity of the nonce N_1 using the σ-values as in Figure 3. Only if verified correctly, P_k computes and outputs the session-key as $\kappa = f_{N_1}(V_1) \oplus f_{N_2}(V_2) \oplus \cdots \oplus f_{N_n}(V_n)$ with $V_i = (sid||ssid||P_i)$.

Theorem 2 *Assume the broadcast channel. If encryption scheme π is indistinguishable under adaptively chosen-ciphertext attack, signature scheme Σ is unforgeable under adaptively chosen-message attack and $\{f_s\}_s$ is a collision-resistant PRF, then the group key agreement protocol (Figure 3) securely realizes the functionality $\mathcal{F}_{\mathcal{GKE}}$ of GKAP against a static adversary under the unauthenticated-link model (UM).*

Proof Due to lack of space, we can only sketch the proof. The full proof will appear in the full version.

Suppose there exist a UM-Adversary $\mathcal{U}$ against GKAP2. We show that we can construct a AM-Adversary $\mathcal{A}$ using $\mathcal{U}$. For any AM-adversary $\mathcal{A}$ against GKAP2 in the UM, there exists AM-adversary $\mathcal{U}$ and no environment $\mathcal{Z}$ can distinguish between real and ideal-process model with a non-negligible advantage. An adveraseary $\mathcal{A}$ proceed as follow. $\mathcal{A}$ computes $(SK_i, PK_i) \leftarrow Gen(1^k)$. The initiator P_1 sends his first message $(sid, ssid, gid, P_1, (c_2||\cdots||c_n))$, then $\mathcal{A}$ computes his signature τ_1 and sends $(sid, ssid, gid, P_1, (c_2||\cdots||c_n), \tau_1)$ to the adversary $\mathcal{U}$. A_{UM} outputs $(sid, ssid, gid, P_1, (c'_2||\cdots||c'_n), \tau'_1)$ then $\mathcal{A}$ verifies τ'_1. If verified correctly, then $\mathcal{A}$ outputs $(sid, ssid, gid, P_1, (c'_2||\cdots||c'_n))$ else aborts. □

7. Conclusion

We have shown a group key agreement protocol that is secure under malicious insiders. Especially, our group key agreement protocol is the first one that maintains UC-security in a meaningful way even if malicious insiders attempt the malleability attacks. So our group key agreement protocol does not allow the adversary to control the session-key as desired. The protocol is efficient enough for practical uses, composed of 2 rounds and n messages for n-party groups.

References

[1] M. Abdalla E. Bresson O. Chevassut and D. Pointcheval, Password-Based Group Key Exchange in a Constant Number of Rounds, PKC 2006, LNCS 3958, pp. 427–442, 2006.

[2] M. Bellare and P. Rogaway, Entity Authentication and

Key Distribution, CRYPTO '93, LNCS 773, pp. 232–249, 1993.

[3] C. Boyd and J. M. G. Nieto, Round-Optimal Contributory Conference Key Agreement, PKC 2003, LNCS 2567, pp. 161–174, 2003.

[4] R. Canetti, Universally Composable Security, A New Paradigm for Cryptographic Protocols, FOCS 2001, pp. 136–145, 2001.

[5] Y. Desmedt J. Pieprzyk R. Steinfeld and H. Wang, A Non-malleable Group Key Exchange Protocol Robust Against Active Insiders, ISC 2006, LNCS 4176, pp. 459–475, 2006.

[6] J. Katz and J. Shin, Modeling Insider Attacks on Group Key-Exchange Protocols, ACM CCCS 2005, pp. 180–189, 2005.