Flexible Group Key Exchange with On-demand Computation of Subgroup Keys Supporting Subgroup Key Randomization

Keita Emura[1] and Takashi Sato[2]
[1]*Network Security Research Institute, Security Architecture Laboratory, National Institute of Information and Communications Technology (NICT), Koganei, Japan*
[2]*School of Information Science, Japan Advanced Institute of Science and Technology (JAIST), Nomi, Japan*

Keywords: Group Key Exchange, On-demand Computation of Subgroup Keys.

Abstract: In AFRICACRYPT2010, Abdalla, Chevalier, Manulis, and Pointcheval proposed an improvement of group key exchange (GKE), denoted by GKE+S, which enables on-demand derivation of independent secret subgroup key for all potential subsets. On-demand derivation is efficient (actually, it requires only one round) compared with GKE for subgroup (which requires two or more rounds, usually) by re-using values which was used for the initial GKE session for superior group. In this paper, we improve the Abdalla et al. GKE+S protocol to support key randomization. In our GKE+S protocol, the subgroup key derivation algorithm is probabilistic, whereas it is deterministic in the original Abdalla et al. GKE+S protocol. All subgroup member can compute the new subgroup key (e.g., for countermeasure of subgroup key leakage) with just one-round additional complexity. Our subgroup key establishment methodology is inspired by the "essential idea" of the NAXOS technique. Our GKE+S protocol is authenticated key exchange (AKE) secure under the Gap Diffie-Hellman assumption in the random oracle model.

1 INTRODUCTION

In AFRICACRYPT2010 (Abdalla et al., 2010), Abdalla, Chevalier, Manulis, and Pointcheval proposed an improvement of group key exchange (GKE), denoted by GKE+S[1], which enables on-demand derivation of independent secret subgroup key for all potential subsets. It is particularly worth noting that the required round of the subgroup key computation phase is just one by re-using the values $\{y_1,\ldots,y_m\}$. So, the Abdalla et al. GKE+S protocol reduces the round complexity (from two to one) compared with the case that the BD protocol (Burmester and Desmedt, 1994) is directly executed for a subgroup $(U_1,U_2,\ldots,U_m)$. It is notable that their on-demand subgroup key derivation algorithm is deterministic, that is, the subgroup key is uniquely determined by $(y_1,y_2,\ldots,y_m)$.

Here, we considered the case that the subgroup $(U_1,U_2,\ldots,U_m)$ would like to establish the "new" subgroup key[2] (e.g., for countermeasure of subgroup key leakage). To establish the new group key, GKE+S protocol for $(U_1,U_2,\ldots,U_n)$ needs to be executed again, and then the new subgroup key is established by executing the on-demand derivation algorithm for $(U_1,U_2,\ldots,U_m)$. That is, it spoils the significant achievement of the GKE+S concept.

Our Contribution. In this paper, we improve the Abdalla et al. GKE+S protocol to support key randomization. In our GKE+S protocol, the subgroup key derivation algorithm (say $\mathcal{P}.SKE$) is probabilistic, and therefore all subgroup member can establish the new subgroup key with just one-round additional complexity. Our subgroup key establishment methodology is inspired by the "essential idea" of the NAXOS technique (LaMacchia et al., 2007)[3] (not di-

[1]First, the Burmester-Desmedt (BD) protocol (Burmester and Desmedt, 1994) (with certain modification to enable the subgroup key derivation) is executed with a group $(U_1,U_2,\ldots,U_n)$). In this group key computation phase, a user (say U_i) publishes the value (say y_i) for establishing the group key. The end of this phase, all user $(U_1,U_2,\ldots,U_n)$ shares the common group key. Next, in the subgroup key computation phase, a member of subgroup (w.l.o.g., $(U_1,U_2,\ldots,U_m) \subset (U_1,U_2,\ldots,U_n)$) can establish the subgroup key, which is independent of the group key.

[2]We explicitly exculde the case that a GKE protocol is directly executed for $(U_1,U_2,\ldots,U_m)$.

[3]The NAXOS technique is for achieving the ephemeral key leakage resilience. An ephemeral public key is com-

Emura K. and Sato T..
Flexible Group Key Exchange with On-demand Computation of Subgroup Keys Supporting Subgroup Key Randomization.
DOI: 10.5220/0003986003530357
In *Proceedings of the International Conference on Security and Cryptography* (SECRYPT-2012), pages 353-357
ISBN: 978-989-8565-24-2

rect use).

The previous one-round GKE schemes (e.g., (Boyd and Nieto, 2003; Gorantla et al., 2009)) assume that each user U_i has a long-lived secret key LL_i which is generated in the initial phase, and the initial phase is not included in the round complexity. So, according to the GKE fashion, our GKE+S protocol (of subgroup key phase) also can be regarded as a one-round GKE protocol. One-round GKE has a benefit point from the viewpoint of robustness[4]. If a GKE has two or more rounds, robustness is important, since it is impractical the case that the remaining nodes must run GKE of the first round again. On the contrary, one-round GKE does not have to consider robustness from the protocol termination's point of view.

Remark. Note that Cheng and Ma pointed out that the Abdalla et al. GKE+S protocol is vulnerable to malicious insiders attack (Cheng and Ma, 2010). They also give a countermeasure of such attack by adding the key confirmation phase. However, adding a signature-based key confirmation round is a standard approach (which has been introduced in (Katz and Shin, 2005)) for insider security. We make it clear that our proposed scheme can be modified to be secure against insider attack by adding the key confirmation phase.

We should notice that a recent paper (Wu et al., 2011) allows to compute group encryption keys for any subgroups without any extra round of communications. This functionally achieves the same goal of this paper. We would like to thank a reviewer who pointed out this fact.

2 SECURITY MODELS

First, we define the syntax of GKE+S protocol by following (Abdalla et al., 2010). Let $\mathcal{U}$ be a set of at most n users in the universe. We assume that their identities are unique. Any subset of m users $(2 \leq m \leq n)$ invokes a single session of a GKE+S protocol $\mathcal{P}$. Each $U_i \in \mathcal{U}$ holds a long-lived key LL_i. The participation of U_i is expressed by an instance Π_i^s for some $s \in \mathbb{N}$ (i stands for the identity of U_i, and s stands for the number of participations of the GKE+S protocol). An execution of a GKE+S protocol is split in two stages, denoted as *group stage* and *subgroup stage*. An instance Π_i^s is invoked for one GKE+S session with some partner id $\mathsf{pid}_i^s \subseteq \mathcal{U}$ which includes the identities of all (i.e., including U_i also) intended participants in a group stage. Similarly, subgroup partner id $\mathsf{spid}_i^s \subset \mathsf{pid}_i^s$ is also defined. Moreover, Π_i^s holds a session id sid_i^s which uniquely identifies the current protocol session of a group stage. Similarly, subgroup session id ssid_i^s (which uniquely identifies the current protocol session of a subgroup stage). Moreover, each subgroup contains different group members. So, in this paper, when U_i executes the $\mathcal{P}.SKE$ algorithm in ℓ times, we assume that Π_i^s holds pid_i^s, sid_i^s, and $\{(\mathsf{ssid}_i^{s,\ell}, \mathsf{spid}_i^{s,\ell})\}$. In addition, we assume that $(\mathsf{ssid}_i^{s,\ell+1}, \mathsf{spid}_i^{s,\ell+1})$ are added into Π_i^s when U_i executes $\mathcal{P}.SKE$ again. We say that Π_i^s and Π_j^t is partnered if $\mathsf{sid}_i^s = \mathsf{sid}_j^t$ and $\mathsf{pid}_i^s = \mathsf{pid}_j^t$. We call Π_i^s is *accepted* if Π_i^s can compute the session group key k_i^s successfully.

Definition 1 (Syntax of GKE+S Protocols)**.** *$[\mathcal{P}.GKE(U_1, \ldots, U_n)]$: This protocol defines the group stage. For each U_i, a new instance Π_i^s with $\mathsf{pid}_i^s = (U_1, \ldots, U_n)$ is created, and a probabilistic interactive protocol between these instances is executed. Then, every instance Π_i^s computes the session group key k_i^s.*

$[\mathcal{P}.SKE(\Pi_i^s, \mathsf{spid}_i^{s,\ell})]$: This protocol defines the subgroup stage. For an accepted instance Π_i^s and a subgroup partner id $\mathsf{spid}_i^{s,\ell} \subset \mathsf{pid}_i^s$, a probabilistic (possibly interactive) algorithm is executed, and outputs the session subgroup key k_{i,J_ℓ}^s, where J_ℓ is the set of indices of users in $\mathsf{spid}_i^{s,\ell}$.

The correctness is defined as follows. A GKE+S protocol $\mathcal{P}$ is said to be correct if all instances (invoked by the group stage of $\mathcal{P}$.GKE) accept with identical group keys. In addition, for all instances Π_j^t (partnered with Π_i^s), $\mathcal{P}.\mathrm{SKE}(\Pi_i^s, \mathsf{spid}_i^{s,\ell}) = \mathcal{P}.\mathrm{SKE}(\Pi_j^t, \mathsf{spid}_j^{t,u})$ holds if $\mathsf{spid}_i^{s,\ell} = \mathsf{spid}_j^{t,u}$.

Next, we define adversarial models and the authenticated key exchange (AKE) security for both group key and subgroup key. As mentioned by Abdalla et al., the security of GKE+S protocol must ensure independence of the group key and any subgroup key, i.e., both (1) even if any subgroup key is leaked to the adversary, the secrecy of the group key must hold, and (2) the leakage of group key must guarantee the secrecy of any subgroup key. Let $\mathcal{A}$ be a PPT adversary who can issue the following queries:

- $\mathsf{Execute}(U_1, \ldots, U_n)$: $\mathcal{A}$ can obtain the execution

puted by using the hashed value of the static long-lived secret key and the ephemeral secret key. Even if the ephemeral secret key is revealed, the exponent of the ephemeral public key is not revealed as long as the static long-lived secret key is not revealed. We apply this essential idea of the NAXOS technique.

[4]GKE is called robust (Hatano et al., 2011; Jarecki et al., 2007) even if a node is down, the protocol can be successfully terminated by the remaining nodes.

transcript of the group stage between the group member $(U_1,\ldots,U_n)$.

- $\mathsf{Send}(\Pi_i^s, M)$: Through this query, $\mathcal{A}$ can deliver a message M to Π_i^s. Then $\mathcal{A}$ can obtain the protocol message generated by Π_i^s in response to M. As in the Abdalla et al. definition (Abdalla et al., 2010), we consider a special invocation query of the form $\mathsf{Send}(U_i, (\mathit{start}', U_1, \ldots, U_n))$. It creates a new instance Π_i^s with $\mathsf{pid}_i^s = (U_1,\ldots,U_n)$ and provides $\mathcal{A}$ with the first protocol message.
- $\mathsf{SKE}(\Pi_i^s, \mathsf{spid}_i^s)$: Through this query, $\mathcal{A}$ can schedule the on-demand subgroup key computation. If $\mathcal{P}.\mathsf{SKE}$ is an interactive algorithm, then $\mathcal{A}$ obtains the first message for the subgroup stage. Otherwise, Π_i^s computes $k_{i,J}^s$. This query is processed only if Π_i^s has accepted and $\mathsf{spid}_i^s \subset \mathsf{pid}_i^s$. Note that a query $(\Pi_i^s, \mathsf{spid}_i^s)$ can be asked in a (polynomial) number of times.
- $\mathsf{RevealGK}(\Pi_i^s)$: $\mathcal{A}$ can obtain the group key k_i^s only if Π_i^s has accepted in the group stage.
- $\mathsf{RevealSK}(\Pi_i^s, \mathsf{spid}_i^{s,\ell})$: $\mathcal{A}$ can obtain the group key k_{i,J_ℓ}^s. The query is answered only if $\mathcal{P}.\mathsf{SKE}(\Pi_i^s, \mathsf{spid}_i^{s,\ell})$ has been invoked and the subgroup key computed.
- $\mathsf{Corrupt}(U_i)$: $\mathcal{A}$ can obtain the U_i's long-lived secret key LL_i. Note that $\mathcal{A}$ does not gain control over the U_i's behavior, but might be able to communicate on behalf of U_i.
- $\mathsf{TestGK}(\Pi_i^s)$: For a random bit $b \xleftarrow{\$} \{0,1\}$, if $b = 0$ $\mathcal{A}$ is given a random value chosen by the group key space, and if $b = 1$ $\mathcal{A}$ is given the real group key k_i^s. Note that this query can be issued only once, and is answered only if Π_i^s has accepted in the group stage.
- $\mathsf{TestSK}(\Pi_i^s, \mathsf{spid}_i^{s,\ell})$: For a random bit $b \xleftarrow{\$} \{0,1\}$, if $b = 0$ $\mathcal{A}$ is given a random value chosen by the subgroup key space, and if $b = 1$ $\mathcal{A}$ is given the real group key k_{i,J^ℓ}^s. The query is answered only if $\mathcal{P}.\mathsf{SKE}(\Pi_i^s, \mathsf{spid}_i^{s,\ell})$ has been invoked and the subgroup key computed.

A user U is called honest if no $\mathsf{Corrupt}(U)$ has been issued by $\mathcal{A}$. On the contrary, a user U is called corrupted or malicious.

Next, we define two freshness notions by following the definitions from (Abdalla et al., 2010).

Definition 2 (Instance Freshness)**.** *Let Π_i^s be an instance and Π_j^t be a partnered instance of Π_i^s. We say that Π_i^s is fresh if Π_i^s has accepted in the group stage and none of the following is true; (1)* $\mathsf{RevealGK}(\Pi_i^s)$ *or* $\mathsf{RevealGK}(\Pi_j^t)$ *has been asked, or (2)* $\mathsf{Corrupt}(U_i)$ *for some $U_i \in \mathsf{pid}_i^s$ was asked before any* $\mathsf{Send}(\Pi_i^s, \cdot)$.

Definition 3 (Instance-Subgroup Freshness)**.** *Let Π_i^s be an instance and Π_j^t be a partnered instance of Π_i^s and $\mathsf{spid}_j^{t,u} = \mathsf{spid}_i^{s,\ell}$. We say that $(\Pi_i^s, \mathsf{spid}_i^{s,\ell})$ are fresh if Π_i^s has accepted in the group stage and none of the following is true; (1)* $\mathsf{RevealSK}(\Pi_i^s, \mathsf{spid}_i^{s,\ell})$ *or* $\mathsf{RevealSK}(\Pi_j^t, \mathsf{spid}_j^{t,u})$ *has been asked, or (2)* $\mathsf{Corrupt}(U_i)$ *or* $\mathsf{Corrupt}(U_j)$ *was asked before any* $\mathsf{Send}(\Pi_i^s, \cdot)$ *or* $\mathsf{Send}(\Pi_j^t, \cdot)$.

Definition 4 (AKE Security of Group Key)**.** *Let $b \xleftarrow{\$} \{0,1\}$ be a uniformly chosen bit flipped in the* TestGK *oracle. A correct GKE+S protocol $\mathcal{P}$ is said to be satisfying AKE security of group key when for any PPT adversary $\mathcal{A}$ the advantage*

$$Adv_{\mathcal{A},\mathcal{P}}^{\text{ake-g}}(\kappa) := 2|\Pr[b' \leftarrow \mathcal{A}^{O}(\kappa); b' = b \wedge \textit{The instance } \Pi_i^s \textit{ is fresh}] - 1|$$

is negligible, where $O = \{\mathsf{Execute}(\cdot), \mathsf{Send}(\cdot,\cdot), \mathsf{SKE}(\cdot,\cdot), \mathsf{RevealGK}(\cdot), \mathsf{RevealSK}(\cdot,\cdot), \mathsf{Corrupt}(\cdot), \mathsf{TestGK}(\cdot)\}$, and Π_i^s is input of TestGK.

Definition 5 (AKE Security of Subgroup Key)**.** *Let $b \xleftarrow{\$} \{0,1\}$ be a uniformly chosen bit flipped in the* TestSK *oracle. A correct GKE+S protocol $\mathcal{P}$ is said to be satisfying AKE security of subgroup key when for any PPT adversary $\mathcal{A}$ the advantage*

$$Adv_{\mathcal{A},\mathcal{P}}^{\text{ake-s}}(\kappa) := 2|\Pr[b' \leftarrow \mathcal{A}^{O}(\kappa); b' = b \wedge \textit{The instance-subgroup pair } (\Pi_i^s, \mathsf{spid}_i^{s,\ell}) \textit{ is fresh}] - 1|$$

is negligible, where $O = \{\mathsf{Execute}(\cdot), \mathsf{Send}(\cdot,\cdot), \mathsf{SKE}(\cdot,\cdot), \mathsf{RevealGK}(\cdot), \mathsf{RevealSK}(\cdot,\cdot), \mathsf{Corrupt}(\cdot), \mathsf{TestSK}(\cdot,\cdot)\}$, and $(\Pi_i^s, \mathsf{spid}_i^{s,\ell})$ is input of TestSK.

3 PROPOSED GKE+S PROTOCOL

In this section, we propose our GKE+S protocol supporting subgroup key randomization. In our proposal, we apply digital signature $\Sigma := (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ for two purposes.

The Underlying Idea. Briefly, the flow of our GKE+S protocol is described as follows.

[Group Stage]. First, we execute the Abdalla et al. GKE+S protocol with $(U_1,\ldots,U_n)$. Then, $(y_1, y_2, \ldots, y_n) = (g^{x_1}, g^{x_2}, \ldots, g^{x_n})$ are published and intermediate values $(z'_{1,2}, z'_{2,3}, \ldots, z'_{n,1})$ are calculated. Note that these intermediate values can be computed by a group member only.

[Subgroup Stage]. Next, in the (ℓ-th) subgroup key computation phase, a member of subgroup $U_i \in (U_1, U_2, \ldots, U_m)$ computes a signature $\sigma_i \leftarrow \mathsf{Sign}(sk_i, (U_i, z_i, \mathsf{ssid}_i^{\ell}, R_i))$, where R_i is the random value chosen by U_i (and other values are explained in the scheme).

- Again, only a subgroup member can compute intermediate values $(z'_{1,2}, z'_{2,3}, \ldots, z'_{m,1})$. We regard these intermediate values $(z'_{1,2}, z'_{2,3}, \ldots, z'_{m,1})$ as the static[5] long-lived secret key in the NAXOS technique context.
- Moreover, we regard $(R_1, R_2, \ldots, R_m)$ as the ephemeral "secret" key in the NAXOS technique context.

So, the subgroup key is computed by applying a hash function to $(z'_{1,2}, z'_{2,3}, \ldots, z'_{m,1})$ and $(R_1, R_2, \ldots, R_m)$ (and $\mathsf{ssid}_i^{s,\ell}$ also). Here, we pay attention to the fact that the NAXOS technique leads to the ephemeral secret key leakage resilience. That is,

- Even if the ephemeral secret key $(R_1, R_2, \ldots, R_m)$ are revealed, the subgroup key is not revealed, as long as the static long-lived secret key $(z'_{1,2}, z'_{2,3}, \ldots, z'_{m,1})$ are not revealed.
- So, $(R_1, R_2, \ldots, R_m)$ can be published, and this is the reason why we achieve the PKE-free setting (whereas, in the Boyd et al. one-round GKE (Boyd and Nieto, 2003), a random nonce is encrypted by using PKE).

On the contrary of the above discussion, even if the subgroup key is revealed, $(z'_{1,2}, z'_{2,3}, \ldots, z'_{m,1})$ are not revealed since a hash function is modeled as the random oracle. In addition, the validity of R_i can be verified by using pk_i.

Here, we describe our GKE+S protocol. It is particularly worth noting that no additional computational cost is required, compared with the Abdalla et al. one. We just additionally require that each user U_i chooses a random nonce R_i.

Protocol 1 (Proposed GKE+S Protocol)**.** *We assume that each $U_i \in \mathcal{U}$ has the long-lived public and secret key pair $(pk_i, sk_i) \leftarrow \mathsf{KeyGen}(1^\kappa)$ (i.e., $LL_i = sk_i$). $H : \mathbb{G} \times \{0,1\}^* \to \{0,1\}^\kappa$, $H_g : \mathbb{G} \to \{0,1\}^\kappa$, and $H_s : \mathbb{G} \to \{0,1\}^\kappa$ are cryptographic hash functions which are modeled as random oracles.*

[5]That is, $(z'_{1,2}, z'_{2,3}, \ldots, z'_{m,1})$ are uniquely determined by the subgroup member $\mathsf{ssid} = (U_1, U_2, \ldots, U_m)$ and their public values $(y_1, y_2, \ldots, y_n)$. In the Abdalla et al. GKE+S, the subgroup key is the hashed value of $(z'_{1,2}, z'_{2,3}, \ldots, z'_{m,1})$ and ssid. This is the reason why the subgroup key derivation algorithm of the Abdalla et al. GKE+S protocol is deterministic.

$\mathcal{P}.\mathbf{GKE}(U_1, \ldots, U_n)$**:** *This is the Group Stage. Let the group be defined by $\mathsf{pid} = (U_1, U_2, \ldots, U_n)$. We assume that user indices from a cycle such that $U_i = U_{i \bmod n}$ and $U_0 = U_n$. We omit s of sid_i^s for simplicity.*

Round 1. *U_i chooses $x_i \xleftarrow{\$} \mathbb{Z}_p$, computes $y_i = g^{x_i}$, and broadcasts (U_i, y_i).*

Round 2. *For U_i, set $\mathsf{sid}_i^s = (U_1|y_1, \ldots, U_n|y_n)$ (" $|$ " stands for the bit concatenation). U_i computes $k'_{i-1,i} := y_{i-1}^{x_i}$, $k'_{i,i+1} := y_{i+1}^{x_i}$, $z'_{i-1,i} := H(k'_{i-1,i}, \mathsf{sid}_i)$, $z'_{i,i+1} := H(k'_{i,i+1}, \mathsf{sid}_i)$, $z_i := z'_{i-1,i} \oplus z'_{i,i+1}$, and $\sigma_i \leftarrow \mathsf{Sign}(sk_i, (U_i, z_i, \mathsf{sid}_i))$, and broadcasts (U_i, z_i, σ_i).*

Group Key Computation. *U_i computes the group key k_i as follows.*

- *Check whether $z_1 \oplus \cdots \oplus z_n = 0$ and whether all received signatures $\{\sigma_j\}_{U_j \in \mathsf{pid} \setminus \{U_i\}}$ are valid. If these checks fail, then abort.*
- *Iteratively compute $z'_{i+1,i+2} \leftarrow z'_{i,i+1} \oplus z_{i+1}$, $z'_{i+2,i+3} \leftarrow z'_{i+1,i+2} \oplus z_{i+2}$, …, and $z'_{i+n-1,i+n} \leftarrow z'_{i+n-2,i+n-1} \oplus z_{i+n-1}$.*
- *Output $k_i = H_g(z'_{1,2}, z'_{2,3}, \ldots, z'_{n,1}, \mathsf{sid}_i)$.*

$\mathcal{P}.\mathbf{SKE}(\Pi_i, \mathsf{spid}_i)$**:** *This is the Subgroup Stage. Let the subgroup be defined by $\mathsf{spid} = (U_1, U_2, \ldots, U_m) \subseteq \mathsf{pid}$. We assume that user indices from a cycle such that $U_i = U_{i \bmod m}$ and $U_0 = U_m$. Note that $(y_1, \ldots, y_m)$ has been published in the previous group stage. We omit s of ssid_i^s and ℓ of k_{i,J^ℓ} for simplicity.*

Round 1. *For U_i, set $\mathsf{ssid}_i = (U_1|y_1, \ldots, U_m|y_m)$. U_i chooses $R_i \xleftarrow{\$} \{0,1\}^\kappa$, and computes $k'_{i-1,i} := y_{i-1}^{x_i}$, $k'_{i,i+1} := y_{i+1}^{x_i}$, $z'_{i-1,i} := H(k'_{i-1,i}, \mathsf{ssid}_i)$, $z'_{i,i+1} := H(k'_{i,i+1}, \mathsf{ssid}_i)$, $z_i := z'_{i-1,i} \oplus z'_{i,i+1}$, and $\sigma_i \leftarrow \mathsf{Sign}(sk_i, (U_i, z_i, \mathsf{ssid}_i, R_i))$, and broadcasts $(U_i, z_i, \sigma_i, R_i)$.*

Subgroup Key Computation. *U_i computes the group key $k_{i,J}$ as follows.*

- *Check whether $z_1 \oplus \cdots \oplus z_m = 0$ and whether all received signatures $\{\sigma_j\}_{U_j \in \mathsf{spid} \setminus \{U_i\}}$ are valid. If these checks fail, then abort.*
- *Iteratively compute $z'_{i+1,i+2} \leftarrow z'_{i,i+1} \oplus z_{i+1}$, $z'_{i+2,i+3} \leftarrow z'_{i+1,i+2} \oplus z_{i+2}$, …, and $z'_{i+m-1,i+m} \leftarrow z'_{i+m-2,i+m-1} \oplus z_{i+n-1}$.*
- *Output $k_{i,J} = H_s(z'_{1,2}, z'_{2,3}, \ldots, z'_{m,1}, R_1, R_2, \ldots, R_m, \mathsf{ssid}_i)$.*

In the original Abdalla et al. GKE+S, each x_i is the random value for establishing both group key and subgroup key. The most essential point of the

key exchange is that U_i and U_{i+1} can compute $k'_{i,i+1}$ by using either x_i or x_{i+1} such that $k'_{i,i+1} = y_i^{x_{i+1}}$ or $k'_{i,i+1} = y_{i+1}^{x_i}$. So, even if $y_i = g^{x_i}$ and $y_{i+1} = g^{x_{i+1}}$ are re-randomized, e.g., $y'_i := g^{x'_i}$ and $y'_{i+1} := g^{x'_{i+1}}$ by re-selected values $x'_i, x'_{i+1} \xleftarrow{\$} \mathbb{Z}_p$, U_i and U_{i+1} cannot compute either $(y'_{i+1})^{x'_i}$ or $(y'_i)^{x'_{i+1}}$, since both y'_i and y'_{i+1} are not published. So, to realize randomization of keys, our methodology works. As a drawback of our methodology, it totally depends on the random oracle methodology.

The remaining concern is the validity of each R_i, namely, an adversary $\mathcal{A}$ may insert a non-legitimate R into the transcript. We prevent this attack by including R as the signed message. It is particularly worth noting that $\mathcal{A}$ may be an insider (e.g., $\mathcal{A} \in \mathcal{U}$ and $\mathcal{A} \notin \mathsf{pid}$ or $\mathcal{A} \in \mathsf{pid} \setminus \mathsf{ssid}$). In both cases, $\mathcal{A}$ has a legitimate long-lived key pair (pk, sk) generated by the KeyGen algorithm. However, since the member of subgroup is bound by $\mathsf{spid} = (U_1, U_2, \ldots, U_m)$, there is no way that such $\mathcal{A}$ inserts non-legitimate R into the transcript if Σ is EUF-CMA. Note that the random nonce R_i depends on ssid_i^ℓ via $\sigma_i \leftarrow \mathsf{Sign}(sk_i, (U_i, z_i, \mathsf{ssid}_i^\ell, R_i))$. In addition, ℓ is incremented by each session. That is, R_i is not used in the different session.

One may think that what the difference between our protocol and the following simple protocol is: the previous subgroup key (say k_{i,J_ℓ}) is used as the massage authentication code (MAC) key, and broadcast $MAC_{k_{i,J_\ell}}(R_i)$, and compute the new subgroup key $k_{i,J_{\ell+1}} = H(\{R_i\}_{i=1}^m)$ by using certain hash function. The main difference between ours and the simple protocol is explained as follows. In our protocol, even if the subgroup key k_{i,J_ℓ} is revealed, $(z'_{1,2}, z'_{2,3}, \ldots, z'_{m,1})$ are not revealed since a hash function is modeled as the random oracle. So, our protocol is secure against the subgroup key leakage. On the contrary, in the above simple protocol, once k_{i,J_ℓ} is revealed, its security is not guaranteed, i.e., anyone (who is not a subgroup member) can compute $k_{i,J_{\ell+1}}$. Note that, unfortunately, our protocol does not follow forward secrecy (i.e., the long-term secret key leakage), since $(z'_{1,2}, z'_{2,3}, \ldots, z'_{m,1})$ is re-used. There is space for improvement of this point.

Here we only state the theorems describing the security of our GKE+S protocols due to the page limitation. Let q_{E_x}, q_{S_e}, and q_{SKE} be the number of invocation of the Execute oracle, the Send oracle, and the SKE oracle, respectively, and q_H, q_{H_g}, and q_{H_s} be the number of access of H, H_g, and H_s, respectively.

Theorem 1. *Our GKE+S protocol satisfies AKE security of group key under the GDH assumption in the random oracle model as follows.*

$$Adv^{ake\text{-}g}_{\mathcal{A},\mathcal{P}}(\kappa) \leq \frac{2n(q_{E_x}+q_{S_e})^2}{p} + \frac{(q_{H_g}+q_{H_s})^2}{2^{\kappa-1}} + 2nAdv^{EUF\text{-}CMA}_{\Sigma,\mathcal{A}}(\kappa) + 2q_{Se}(nq_H Adv^{GDH}_{\mathbb{G}}(\kappa) + \frac{q_{H_g}}{2^{2\kappa}})$$

Theorem 2. *Our GKE+S protocol satisfies AKE security of subgroup key under the GDH assumption in the random oracle model as follows.*

$$\begin{aligned} Adv^{ake\text{-}s}_{\mathcal{A},\mathcal{P}}(\kappa) \leq{}& \frac{2n(q_{E_x}+q_{S_e})^2}{p} + \frac{(q_{H_g}+q_{H_s})^2}{2^{\kappa-1}} \\ &+ 2nAdv^{EUF\text{-}CMA}_{\Sigma,\mathcal{A}}(\kappa) \\ &+ 2q_{Se}\big((n+(n-1)q_{SKE})q_H Adv^{GDH}_{\mathbb{G}}(\kappa) \\ &+ \frac{q_{SKE}q_{H_s}}{2^{2\kappa}}\big) \end{aligned}$$

REFERENCES

Abdalla, M., Chevalier, C., Manulis, M., and Pointcheval, D. (2010). Flexible group key exchange with on-demand computation of subgroup keys. In *AFRICACRYPT*, pages 351–368.

Boyd, C. and Nieto, J. M. G. (2003). Round-optimal contributory conference key agreement. In *Public Key Cryptography*, pages 161–174.

Burmester, M. and Desmedt, Y. (1994). A secure and efficient conference key distribution system (extended abstract). In *EUROCRYPT*, pages 275–286.

Cheng, Q. and Ma, C. (2010). Security weakness of flexible group key exchange with on-demand computation of subgroup keys. *CoRR*, abs/1008.1221.

Gorantla, M. C., Boyd, C., Nieto, J. M. G., and Manulis, M. (2009). Generic one round group key exchange in the standard model. In *ICISC*, pages 1–15.

Hatano, T., Miyaji, A., and Sato, T. (2011). *T*-robust scalable group key exchange protocol with $O(\log n)$ complexity. In *ACISP*, pages 189–207.

Jarecki, S., Kim, J., and Tsudik, G. (2007). Robust group key agreement using short broadcasts. In *ACM Conference on Computer and Communications Security*, pages 411–420.

Katz, J. and Shin, J. S. (2005). Modeling insider attacks on group key-exchange protocols. In *ACM Conference on Computer and Communications Security*, pages 180–189. ACM.

LaMacchia, B. A., Lauter, K., and Mityagin, A. (2007). Stronger security of authenticated key exchange. In *ProvSec*, pages 1–16.

Wu, Q., Qin, B., Zhang, L., Domingo-Ferrer, J., and Farràs, O. (2011). Bridging broadcast encryption and group key agreement. In *ASIACRYPT*, pages 143–160.