Membership Privacy for Asynchronous Group Messaging

Keita Emura[§], Kaisei Kajita[†], Ryo Nojima[§], Kazuto Ogawa[§], and Go Ohtake[†]

[§]National Institute of Information and Communications Technology (NICT), Japan.
[†]Japan Broadcasting Corporation, Japan.

March 9, 2022

Abstract

The Signal protocol is a secure messaging protocol providing end-to-end encrypted asynchronous communication. In this paper, we focus on a method capable of hiding membership information from the viewpoint of non group members in a secure group messaging (SGM) protocol, which we call "membership privacy". Although Chase et al. (ACM CCS 2020) have considered the same notion, their proposal is an extension of Signal so called "Pairwise Signal" where a group message is repeatedly sent over individual Signal channels. Thus their protocol is not scalable. In this work, we extend the Cohn-Gordon et al. SGM protocol (ACM CCS 2018), which we call the Asynchronous Ratcheting Trees (ART) protocol, to add membership privacy. We employ a key-private and robust public-key encryption (Abdalla et al., TCC2010/JoC2018) for hiding membership-related values in the setup phase. Furthermore, we concentrate on the fact that a group common key provides anonymity. This fact is used to encrypt membership information in the key update phase. Our extension does not affect the forward secrecy and post-compromise security of the original ART protocol. Our modification achieves asymptotically the same efficiency of the ART protocol in the setup phase. Any additional cost for key update does not depend on the number of group members (specifically, one encryption and decryption of a symmetric key-encryption scheme and one execution of a key-derivation function for each key update are employed). Therefore, the proposed protocol can add membership privacy to the ART protocol with a quite small overhead.

1 Introduction

1.1 Background

Since a secure messaging protocol called Signal was analyzed by Cohn-Gordon et al. [15], researches on secure messaging have been much popular. Besides end-to-end encryption, secure messaging protocols provide *asynchronicity* where a sender is not required to aware whether a receiver is online or not, *forward secrecy* (FS) where even if state information (i.e., long-term secret keys or any values used in subsequent operations) is leaked, previously derived session keys are still hidden, and *post-compromise security* (PCS) where if an intermediate stage completes with no corruption by an adversary after all of secrets are compromised, then subsequent stages are still secure. Basically, a key update procedure is crucial for providing FS and PCS. The Signal protocol consists of two subprotocols, X3DH (Extended Triple Diffie-Hellman) and double ratcheting, and Hashimoto et al. [24] and Alwen et al. [3] proposed a generic construction, respectively. In addition to two-party messaging, secure group messaging (SGM) protocols also have been proposed [4, 5, 17].

On Membership Privacy in SGM. As an SGM scenario, let us assume that a user participates in an online symposium (with many sessions) several times; for example, when the symposium consists of many courses held at different days. Then, due to asynchronicity, the organizer can set up the online system, even if the participants are offline. Also, due to FS and PCS, the past and future contents are not disclosed, even if the present secret information is disclosed. This is important for a pay-per-view service. Let us consider a privacy if the symposium is held in offline. In this case, a participant is aware of who participates in the symposium (here, we simply consider that he/she can see other participants in the venue), whereas it would be better to hide the participants in the symposium from the nonparticipants, unless the organizer intentionally discloses the membership information, and all participants agree with it. Moreover, if one knows that a particular person speaks more than the others, then this person might be identified. Thus, it would be better to hide the speakers (according to the anonymity in the usual sense). Therefore, in SGM related to multiple participants, where membership privacy (containing anonymity), is very important. Chase et al. [11] have considered the same notion. In their protocol, a central server manages an encrypted member list, and each group member anonymously authenticates to the server via an anonymous credential system. Because the SGM protocol is an extension of the Signal protocol so called "Pairwise Signal" where each group member needs to run the (one-to-one) Signal protocol for all other group members, respectively, their protocol is not scalable. So, to the best of our knowledge, no such a SGM protocol (with scalability) has been proposed so far.

We remark that anonymity does not enough to provide membership privacy. For example, signatures providing anonymity, such as group signatures [12] and ring signatures [31] can hide a user generating a signature. However, the membership information of a user belonging to the group/ring is not usually hidden. Because of this issue, Backes et al. [6] extended the security definitions of fully dynamic group signatures [8] and proposed a group signature scheme providing membership privacy.[1] In this case study on group signature context, simple anonymity cannot provide membership privacy. Therefore, it is necessary to consider how to provide membership privacy in the SGM context.

1.2 Our Contribution

In this paper, we extend the Cohn-Gordon et al. SGM protocol [17], which we call the Asynchronous Ratcheting Trees (ART) protocol, to add membership privacy, which hides the participants and also the participant who sends a message.To achieve this, we basically do not modify the original key update procedure for directly employing FS and PCS of the ART protocol. Moreover, the ART protocol requires $O(n)$ (resp. $O(\log n)$) computational and communication costs in the setup phase (resp. key update phase), where n is the number of users belonging to the SGM protocol. We consider to keep this efficiency as much as possible even membership privacy is additionally provided.

We employ key-private and robust public-key encryption [1, 2] for hiding membership-related values in the setup phase. This modification achieves asymptotically the same efficiency of the ART protocol in the setup phase for the initiator (See Section 3). Our modification achieves asymptotically the same efficiency of the ART protocol in the setup phase. Furthermore, we concentrate on the fact that a group common key provides anonymity. This fact is used to encrypt membership information in the key update phase. It is noted that the ART protocol also introduces a group common key called a "stage key" to add a message authentication code (MAC). The aim is to authenticate updated public keys (See Section 3). We also introduce a similar common key

[1] As a remark, although basically group-size hiding is not considered in [6], they insisted that it can be hidden by adding dummy group members.

to encrypt membership information in the key update phase. By employing the common MAC key-generation procedure of the ART protocol, the additional key update cost does not depend on the number of group members. Specifically, one encryption and decryption of a symmetric key-encryption scheme and one execution of a key-derivation function for each key update are employed. In this way, membership privacy can be added to the ART protocol with a quite small overhead.

Martiny et al. [29] analyzed the sealed sender functionality (See Section 2.2) and showed that it can be broken by identifying the sender. They assumed that most messages receive a quick response, i.e., when a user device receives a message, the device will automatically send back a delivery receipt to the sender. In our work, such a deanonymization attack due to quick responses and due to communication-related value such as IP address are out of the scope, and it is not considered.

Assuming an Initiator. Cohn-Gordon et al. [17] introduced an initiator who setups a group and generates initial secret keys of group members. We remark that the initiator does not know secret keys of group members after key updating. Thus we assume that the initiator is trusted in the setup phase (as in the ART protocol). In the above mentioned online symposium, the organizer can be an initiator, and we do not consider initiator anonymity since the organizer is obviously a member.

2 Differences between Membership Privacy and Previous Security Definitions

2.1 Anonymity

In membership privacy, the participants and the user who sends a message are hidden. That is, membership privacy implies anonymity (in the sense of group and ring signatures context). On the other hand, as mentioned above, simple anonymity does not provide membership privacy. Actually, Chen et al. [13,14] have considered to add anonymity to the ART protocol that hides the user who sent a message. However, they did not consider hiding membership information. As a result, membership information can be disclosed in their protocol. It is noted that they introduced a trusted third party (TTP) to hide the position information, which is necessary for key updating (See Section 3). However, it is highly undesirable to introduce a TTP in the end-to-end encryption context.

2.2 Sealed Sender in the Signal Protocol

In the Signal protocol, a message can be sent when the Signal server recognizes receiver information, i.e., sender information is not necessary for message sending. Thus, a function called "sealed sender" has been implemented, where an encrypted message contains sender information. Martiny et al. [29] extended the sealed sender functionality to hide receiver information by introducing an additional mailbox. In a two-party communication, hiding both sender and receiver information matches our goal. However, it is not clear whether hiding both sender and receiver information implies hiding membership information when the group (more precisely, more than and not equal to two-party) communication is considered. For example, as in group and ring signatures, anonymity can be realized by hiding the actual signer, without hiding the members of the group. Therefore, it is not clear whether a simple SGM variant of the Signal protocol called "Pairwise Signal" (i.e., a sender sends the same message to all other group members repeatedly over an individual Signal channel),

provides membership privacy. As mentioned above, Chase et al. [11] have added membership privacy to the Pairwise Signal, their protocol is not scalable.

2.3 Deniability

Deniability in key-exchange protocols guarantees that a protocol transcript can be simulated by public values only, namely, no information that used its own secret value to provide the transcript is revealed [30]. Deniability in the Signal protocol has also been considered [24, 33]. Here, we concentrate on the results reported by Hashimoto et al. [24]. Deniability was implemented by employing ring signatures. Ring signatures provide anonymity; a verifier can check whether a signer is a member of a ring (specified by the signer) or not. This means that membership information, i.e., those that are members of the ring, is always public for verification. On the other hand, even if membership information is not infringed, there is no guarantee that a protocol transcript can be simulated by the public values only. Thus, deniability and membership privacy are different notions and are incompatible.

3 Cohn-Gordon et al. SGM Protocol (ART)

In this section, the Cohn-Gordon et al. SGM protocol [17], called Asynchronous Ratcheting Trees (ART), is described. Let $\mathbb{G}$ be a group with prime order p, and $H : \mathbb{G} \to \mathbb{Z}_p$ be a hash function.[2] Furthermore, a key-derivation function (KDF), which is used to derive a stage key, is modeled as a random oracle.[3]

Basically, the ART protocol described in [17] is a tree Diffie–Hellman (DH) key agreement. Each user is assigned to a leaf node, and a secret key of a parent node is defined as the DH key of the children nodes. More specifically, let a secret key of a node be $K_1 \in \mathbb{Z}_p$, and a secret key of the sibling node be $K_2 \in \mathbb{Z}_p$. Then, each public key is defined as g^{K_1} and g^{K_2}, respectively. From the DH key $g^{K_1K_2}$, the secret key and the public key of the parent node are defined as $H(g^{K_1K_2})$ and $g^{H(g^{K_1K_2})}$. A four-user case is shown in Fig. 1. Finally, the secret key of the root node $H(g^{K_5K_6})$ is defined as the current tree key tk, which is used to derive a stage key stgk (this part is explained later).

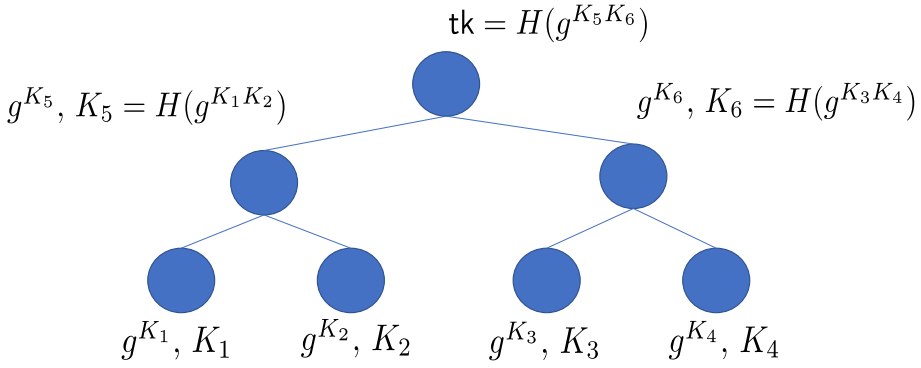

Figure 1: DH Tree

Setup Phase. Next, we explain how to asynchronously setup these node secret keys K. The basic

[2] In [17], H is defined as a mapping from a group element to a $\mathbb{Z}_p$ element that uses the corresponding DH key as the secret key of the parent node. For simplicity, we define H as a hash function.

[3] Cohn-Gordon et al. employed HKDF with SHA256 for key derivation.

concept is to employ a strong one-round authenticated key-exchange protocol. The initiator runs the protocol "locally" using his/her own secret keys and a public key of a group member. Let us assume that each user publishes a key pair $(\mathsf{IK}, \mathsf{EK})$ in a PKI. Here, IK stands for "identity key", and EK stands for "ephemeral prekey". The corresponding secret keys are denoted as ik and ek, respectively. Intuitively (which is not entirely correct), $(\mathsf{IK}, \mathsf{ik})$ is a key pair of a signature scheme used for authentication, $(\mathsf{EK}, \mathsf{ek})$ is a key pair of a key-exchange protocol, and EK is defined as $\mathsf{EK} = g^{\mathsf{ek}}$. For a strong one-round authenticated key-exchange protocol, it is required that

$$\mathsf{KeyExchange}(\mathsf{ik}_I, \mathsf{IK}_R, \mathsf{ek}_I, \mathsf{EK}_R) = \mathsf{KeyExchange}(\mathsf{ik}_R, \mathsf{IK}_I, \mathsf{ek}_R, \mathsf{EK}_I)$$

holds. That is, a key generated by the secret key of an initiator $(\mathsf{ik}_I, \mathsf{ek}_I)$ and a public key of a receiver $(\mathsf{IK}_R, \mathsf{EK}_R)$ is equal to a key generated by a secret key of the receiver $(\mathsf{ik}_R, \mathsf{ek}_R)$ and the secret key of the initiator $(\mathsf{IK}_I, \mathsf{EK}_I)$. Cohn-Gordon et al. stated that although any key-exchange protocol satisfying the above requirement can be employed, they assumed X3DH [28] as the underlying key-exchange protocol. Due to the one-round key exchange, the initiator (and a group member) can locally run the key-exchange protocol if the public key of the partner is obtained. Thus, they can asynchronously (i.e., regardless of the partner being online or offline) exchange a key. Next, the setup procedure is described.

The setup phase run by an initiator is described as follows. Let n be the group size, and the initiator is also a member of the group. For simplicity, the initiator is regarded as the user 1, and $(\mathsf{IK}_1, \mathsf{EK}_1)$ is the initiator's public key managed by the PKI.

[Initiator]:

1. Randomly choose a setup key $\mathsf{suk} \in \mathbb{Z}_p$, and compute $\mathsf{SUK} = g^{\mathsf{suk}}$.
2. Obtain other members' key pairs $n-1$ $\{(\mathsf{IK}_i, \mathsf{EK}_i)\}_{i=2}^{n}$ from PKI.
3. Randomly choose $K_1 \in \mathbb{Z}_p$.
4. For $(\mathsf{IK}_i, \mathsf{EK}_i)$ $(i = 2, 3, \ldots, n)$, locally run the underlying strong one round authenticated key exchange protocol $K_i \leftarrow \mathsf{KeyExchange}(\mathsf{ik}_1, \mathsf{IK}_i, \mathsf{suk}, \mathsf{EK}_i)$. Here, the initiator indicates ik_1 and suk as secret keys.
5. Compute all public keys associated to nodes to the tree $T = (g^{K_1}, \ldots, g^{K_{2n-2}})$. Here, the root node is excluded. Moreover, delete all secret keys (except K_1) $(K_2, \ldots, K_{2n-2})$.
6. Generate a signature σ on $(\{(\mathsf{IK}_i, \mathsf{EK}_i)\}_{i\in[1,n]}, \mathsf{SUK}, T)$ using ik_1.
7. Broadcast $(\{(\mathsf{IK}_i, \mathsf{EK}_i)\}_{i\in[1,n]}, \mathsf{SUK}, T)$ and σ.

Each user $i \in [2, n]$ receives $(\{(\mathsf{IK}_i, \mathsf{EK}_i)\}_{i\in[1,n]}, \mathsf{SUK}, T)$ and σ, and runs the following procedure.

[User i ($i \in [2, n]$)]:

1. Check the validity of σ using the verification key IK_1.
2. If σ is valid, then run $K_i \leftarrow \mathsf{KeyExchange}(\mathsf{ik}_i, \mathsf{IK}_1, \mathsf{ek}_i, \mathsf{SUK})$.
3. Compute the tree key tk using K_i and public keys T
4. Derive the stage key $\mathsf{stgk} \leftarrow \mathsf{KDF}(0, \mathsf{tk}, \{\mathsf{IK}_i\}_{i\in[1,n]}, T)$.

Key Update Phase. Next, we explain how the secret and public keys are updated. When a user updates his/her own secret key K, he/she randomly selects $K' \in \mathbb{Z}_p$ and computes public keys on the path (i.e., from the leaf node associated with the user to the root node). All group members can derive the tree key using their own secret key and updated public keys. However, they need to know which public keys are updated. Thus, the position information, which indicates which leaf node is assigned to the user, is additionally required. Furthermore, these public keys are required to be generated by a group member. Thus, the user computes a MAC for the position information and public keys on the path, and broadcasts the MAC, his/her position information, and the public keys. Here, the MAC key is the current stage key stgk. After receiving the MAC, any group member can check the validity of the MAC since these group members also have a stgk. If the MAC is valid, the group members derive the new tree key and the new stage key. Specifically, each user $i \in [1, n]$ runs the procedure described below. Let the current secret key be K_i and the position information be j.

[User i ($i \in [1, n]$)]:

1. Randomly choose $K'_i \in \mathbb{Z}_p$.
2. Compute public keys on the path (i.e., from the leaf node associated with the user i to the root node). Here, the public key of the root node is excluded because the secret key of the root node is regarded as the tree key. Let T' be a set of updated public keys.
3. Compute a MAC on $j||T'$ using stgk, and broadcast the MAC and $j||T'$.
4. Remove stgk and derive the new stage key using the new tree key.

Each user $k \in [1, n] \setminus \{i\}$ receives the MAC and $j||T'$, and runs the following procedure.

[User k ($k \in [1, n] \setminus \{i\}$)]:

1. Check the validity of the MAC using stgk.
2. If the MAC is valid, then remove stgk and derive the new stage key using the new tree key.

An example of user 2 updating K_2 to K'_2 is shown in Fig. 2. User 2 randomly selects $K'_2 \in \mathbb{Z}_p$ and removes the old K_2. Then, user 2 computes not only $g^{K'_2}$ associated with his/her own leaf node, but also $g^{K'_5}$, where $K'_5 = H((g^{K_1})^{K'_2})$. User 2 also computes a MAC using the current stgk on $2||(g^{K'_2}, g^{K'_5})$ (here "2" is the position information) and derives the new stage key using the new tree key $\mathsf{tk}' = H(g^{K'_5 K_6})$ such that $\mathsf{stgk}' \leftarrow \mathsf{KDF}(\mathsf{stgk}, \mathsf{tk}', \{\mathsf{IK}_i\}_{i \in [1,n]}, T'')$, where T'' is a set of public keys replacing g^{K_2} and g^{K_5} to $g^{K'_2}$ and $g^{K'_5}$, respectively, i.e., $T'' = (g^{K_1}, g^{K'_2}, g^{K_3}, g^{K_4}, g^{K'_5}, g^{K_6})$. stgk' will be used for computing a MAC in the next key update. Finally, user 2 broadcasts the MAC and $(2, g^{K'_2}, g^{K'_5})$, derives tk' and stgk' locally, and removes the previous tk and stgk. When a group member receives $(2, g^{K'_2}, g^{K'_5})$ and the MAC, he/she derives tk' and stgk' and removes the previous tk and stgk if the MAC is valid.

Security: Here, we briefly explain that the ART protocol provides FS and PCS. FS requires that even if long-term secret keys (ik, ek) and random values selected in latter stages are revealed after a user computes a tree key tk, tk still remains secret from nongroup members. Intuitively, the ART protocol provides FS due to the randomness of the updated key K'. That is, K' is independently selected by the previous key K. Thus, knowledge of K' does not affect that of K. We emphasize that K needs to be erased in the key update phase mentioned above, and K is not included in the state information. Here, implicitly assume that a user, whose state information is disclosed,

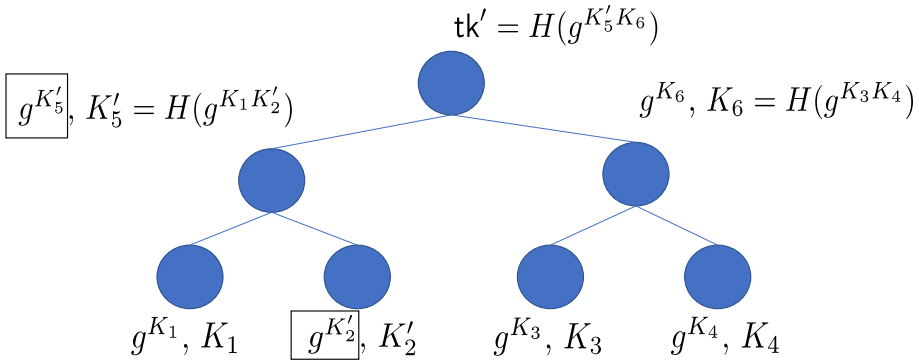

Figure 2: DH Tree (Update)

has updated his/her own key at least once. If no key update is run, the leaf key K is computed by the long-term secret keys (and public keys of the initiator). Thus, the tree key can be derived. Moreover, implicitly assume that state information disclosure occurs after at least two key updates. If only one key update is run, the previous tree key can be computed by the long-term secret keys (and public keys of the initiator), and FS does not hold. In summary, when $n \geq 3$, the ART protocol provides FS if all group members run the key update at least once.

PCS requires that even if all secret values are disclosed, a tree key tk shared in an after stage still remains secret from nongroup members if one no-disclosure stage exists. Since a stateless protocol does not provide PCS [16], a stage key stgk' is derived by the current tree key tk' and the previous stage key stgk, where $\mathsf{stgk}' \leftarrow \mathsf{KDF}(\mathsf{stgk}, \mathsf{tk}', \{\mathsf{IK}_i\}_{i\in[1,n]}, T')$. Let us assume that an adversary obtains a user's secret values at a particular stage, and another user runs the key update in the next stage without the adversary being involved. Then, the adversary can compute the current tree key from the secret values obtained in the previous stage and the current public keys broadcasted by the user. Therefore, the user, whose secret key has been disclosed, is required to run the key update (i.e., the user randomly selects K') without the adversary being involved. In this way, the adversary cannot compute the current tree key since he/she does not know K'. At the same time, the adversary cannot derive the current stage key stgk' since the current tree key is required. However, the adversary knows the previous stage key stgk, and thus, the adversary can send a message with a MAC computed by stgk. Thus, implicitly, the user is required to run the key update and send the updated public keys and his/her position information with a MAC computed by stgk before the adversary sends a message. In summary, the ART protocol provides PCS after a user, whose secret key has been disclosed, runs the key update once without the adversary being involved.

A formal security analysis is explained below. A key-derivation function KDF is modeled as a random oracle. Furthermore, according to Cohn-Gordon et al., the pseudorandom-function oracle-Diffie–Hellman (PRF-ODH) assumption is required to hold over $\mathbb{G}$ [9]. In [17], an unauthorized version of the protocol, where all initial leaf keys K_i are defined by EK_i and SUK (i.e., $K_i = H(g^{\mathsf{ek}_i \mathsf{suk}})$), and no IK_i is involved, is initially introduced. Then, it is proved that stage keys are indistinguishable against a random value under the PRF-ODH assumption in the random oracle model. A model for multistage key exchange defined by Fischlin-Günther [21] is extended in the SGM context as a security model,. Next, the authenticated version, which employs a strong one-round authenticated key-exchange protocol for the initial key setup phase, is introduced, and its security is analyzed by TAMARIN Prover [32]. Note that in the Brende et al. paper [9], the PRF-ODH assumption basically requires $\mathsf{PRF}(g^{xy}, V)$, where g^{xy} is a DH key and V is an input

of the pseudo-random function PRF, look random. Here, an adversary is allowed to obtain g^x, g^y, and V, and access the ODH_x oracle, which takes S and V as input and returns $\mathsf{PRF}(S^x, V)$. Similarly, the adversary is allowed to access the ODH_y oracle. In the Cohn-Gordon et al. paper [17], a "nonoracle" version of the PRF-ODH assumption was introduced, where $(g^x, g^y, H(g^{xy}))$ and $(g^x, g^y, H(g^z))$ are indistinguishable for a random $z \in \mathbb{Z}_p$. However, this assumption is the same as the hashed DH assumption [22].

4 Proposed Protocol

In this section, the proposed SGM protocol, which provides membership privacy by extending the ART protocol, is presented. For simplicity, we assume that a key is updated when a message is sent. Thus, hiding the user who sends a message is the same as hiding the user who updates a key. We assume that a particular user has the role of initiator, and the user always participates in groups (although any user can be an initiator for each group in the ART protocol). Moreover, no membership privacy for the initiator is considered. If a path size, i.e., the number of nodes from a leaf to the root is not the same for all users, the size detracts anonymity. Therefore, we employ a complete binary tree structure (although any binary tree structure can be employed in the ART protocol).

High-level Description. We extract which part of the ART protocol leaks membership privacy, and briefly explain how to provide membership privacy.

1. In the setup phase, a set of public kyes of group members $\{(\mathsf{IK}_i, \mathsf{EK}_i)\}_{i\in[1,n]}$ is contained to a message $(\{(\mathsf{IK}_i, \mathsf{EK}_i)\}_{i\in[1,n]}, \mathsf{SUK}, T)$ sent from the initiator. We do not consider to hide $(\mathsf{IK}_1, \mathsf{EK}_1)$ because these keys are for the initiator.

   - For hiding $\{(\mathsf{IK}_i, \mathsf{EK}_i)\}_{i\in[2,n]}$, we employ key private and robust public key encryption [1, 2]. Briefly, for different two key pairs $(\mathsf{PK}, \mathsf{DK}) \leftarrow \mathsf{KeyGen}(1^k)$ and $(\mathsf{PK}', \mathsf{DK}') \leftarrow \mathsf{KeyGen}(1^k)$, and for a ciphertext $C \leftarrow \mathsf{Ecn}(\mathsf{PK}, M)$ of a message M under PK, it is key private when no information of PK is revealed from C, and it is robust if $\perp \leftarrow \mathsf{Dec}(C, \mathsf{DK}')$. Moreover, it is weak robust when C is generated by the Ecn algorithm.[4] This essentially employs a generic construction of anonymous broadcast encryption proposed by Libert et al. [26].
   - In addition to $(\mathsf{IK}, \mathsf{EK})$, we assume that a public key PK of a key private and robust public key encryption scheme is also managed by PKI. In the setup phase, the initiator chooses a secret key k for a symmetric encryption scheme, encrypts $\{(\mathsf{IK}_i, \mathsf{EK}_i)\}_{i\in[2,n]}$ using k, and encrypts k and position information of a user i using PK_i. Each user can obtain $\{(\mathsf{IK}_i, \mathsf{EK}_i)\}_{i\in[2,n]}$ and own position information only if the user i is a member.
   - Since the ART protocol requires $O(n)$ computation and communication costs for the initiator, additionally employing key private and robust public key encryption in the setup phase does not affect asymptotic complexity of the original protocol. However, if we directly employ key private and robust public key encryption, the efficiency of each user in the setup phase is worsened from $O(\log n)$ to $O(n)$ due to the decryption procedure. That is, if each user does not know which cyphertext can be decrypted owing to

[4] Such a public key encryption scheme can be constructed easily. For example, for a key private public key encryption scheme, e.g., the ElGamal encryption, a random R is contained to a public key and for encryption of M, encrypt $M||R$. The decryption algorithm decrypts a ciphertext, obtains $M||R'$, and outputs M if $R' = R$ and $\perp$, otherwise.

key privacy, then $n/2$-times decryptions are required on average. Thus, we additionally employ a tag-based hint system [26] that provides efficient decryption. Employing the hint system achieves asymptotically the same efficiency of the ART protocol in the setup phase. Briefly, a hint system consists of $(\mathsf{KeyGen}_h, \mathsf{Hint}, \mathsf{Invert})$: the key generation algorithm KeyGen_h takes a security parameter as input, and outputs a key pair $(\mathsf{PK}^h, \mathsf{DK}^h)$. The hint generation algorithm Hint takes a tag t, a public key PK^h, and a random coin r, and outputs a pair (U, H). The inversion algorithm Invert takes DK^h, t, and U as input, and outputs either a hint H or $\perp$. Here, U only depends on r and not on PK^h, and the same r and U are commonly used for all users. Moreover, two security notions hold where, Anonymity: hints generated under two distinct public keys are indistinguishable, and Strong robustness: for two distinct secret keys $\mathsf{DK}_0^h \neq \mathsf{DK}_1^h$ and a common (t, U), the same hint is not generated, i.e., $\mathsf{Invert}(\mathsf{DK}_0^h, t, U) \neq \mathsf{Invert}(\mathsf{DK}_1^h, t, U)$ holds.

We remark that the Libert et al. construction provides recipient anonymity even from the view point of group members. In our purpose, i.e., providing membership privacy from the viewpoint of non group members, outsider anonymity [20, 27], where a group member may break anonymity, is enough and then the ciphertext size can be reduced. However, the initiator needs to generate secret keys of users using a master secret key, and needs to issue the keys to users. This does not match asynchronicity of SGM.

2. In the key update phase, a user who updates the key declares position information. This information distinguishes users that detracts anonymity.

   - It is required that only the group members obtain the position information. We concentrate on the authentication in the key update phase of the ART protocol, which employs a common stage key as a MAC key. Such a common key provides anonymity.[5] Therefore, each user, who updates the key, encrypts his/her position information using a common encryption key.
   - For PCS, this common encryption key needs to be updated. For this purpose, hash chains are employed as in the ART protocol. Let mack be a MAC key, which was originally described as stgk, and let enck be an encryption key. In the setup phase, mack is derived from 0 and the first tree key, whereas enck is derived from 1 and the first tree key. Both are derived from the current tree key and the previous mack and enck, respectively. This structure enables PCS. As in the authentication in the ART protocol, only the group members can derive enck and obtain the position information of the user who updates the key. Moreover, by employing a common encryption key, anonymity is not detracted.
   - Any additional cost for key update does not depend on the number of group members (specifically, one encryption and decryption of a symmetric key-encryption scheme and one execution of a key-derivation function for each key update are employed). In this way, membership privacy can be added to the ART protocol with a quite small overhead.

3. When a stage key is derived, a set of public keys of group members $\{\mathsf{IK}_i\}_{i\in[1,n]}$ is contained to the input of KDF.

[5] For example, if no traceability is required, a simple group signature scheme can be implemented by sharing one signature key pair among the group members. Then, since all members sign a message using the same signing key and the signature is also verified by the same verification key, the actual signer cannot be identified. Thus, anonymity is provided.

- Due to the $\{\mathsf{IK}_i\}_{i\in[1,n]}$ encryption in the setup phase, only the group members know which identity key is included. Thus, one may think that each user locally stores $\{\mathsf{IK}_i\}_{i\in[1,n]}$ and then uses it to derive the stage key. However, if an adversary obtains state information, $\{\mathsf{IK}_i\}_{i\in[1,n]}$ is also disclosed. Thus, we discuss why $\{\mathsf{IK}_i\}_{i\in[1,n]}$ is required to derive the stage key. Cohn-Gordon et al. stated that KDF takes $\{\mathsf{IK}_i\}_{i\in[1,n]}$ as input to satisfy partnering. Intuitively, partnering means that different session and nongroup members cannot derive the same stage key. Since KDF is modeled as a random oracle, if the output is the same, then the input is also the same. Moreover, the probability that a set of public keys T and the current tree key tk are the same as those of another session is negligible due to the randomness of K. Thus, in the proposed protocol, $\{\mathsf{IK}_i\}_{i\in[1,n]}$ is removed from the KDF input.

We give our proposed protocol as follows.

Setup Phase. We give the setup phase as follows. In addition to $(\mathsf{IK}, \mathsf{EK})$ managed by PKI, we introduce a key pair $(\mathsf{PK}, \mathsf{DK})$ of a key private and robust public key encryption, and a key pair $(\mathsf{PK}^h, \mathsf{DK}^h)$ of a hint system.

[Initiator]:

1. Randomly choose a setup key $\mathsf{suk} \in \mathbb{Z}_p$, and compute $\mathsf{SUK} = g^{\mathsf{suk}}$.
2. Obtain other members' key pairs $n-1$ $\{(\mathsf{IK}_i, \mathsf{EK}_i, \mathsf{PK}_i, \mathsf{PK}_i^h)\}_{i=2}^n$ from PKI.
3. Randomly choose $K_1 \in \mathbb{Z}_p$.
4. For $(\mathsf{IK}_i, \mathsf{EK}_i)$ $(i = 2, 3, \ldots, n)$, locally run the underlying strong one round authenticated key exchange protocol $K_i \leftarrow \mathsf{KeyExchange}(\mathsf{ik}_1, \mathsf{IK}_i, \mathsf{suk}, \mathsf{EK}_i)$. Here, the initiator indicates ik_1 and suk as secret keys.
5. Compute all public keys associated with nodes in the tree $T = (g^{K_1}, \ldots, g^{K_{2n-2}})$. Here, the root node is excluded. Furthermore, delete all secret keys (except K_1) $(K_2, \ldots, K_{2n-2})$.
6. Randomly choose a secret key k of a symmetric encryption scheme. Encrypt $\{(\mathsf{IK}_i, \mathsf{EK}_i)\}_{i\in[2,n]}$ using k, and let $\mathsf{C_{pk}}$ be the ciphertext.
7. Encrypt $\mathsf{IK}_1||j||k$ using PK_j $(j = 2, 3, \ldots, n)$ where IK_1 is regarded as a label and j is position information. Let CT_j be the ciphertext.[6]
8. Choose r and run $(U, H_j) \leftarrow \mathsf{Hint}(\mathsf{IK}_1, \mathsf{PK}_j^h, r)$ for $(j = 2, 3, \ldots, n)$. Remark that the same r is used and thus U is a common value for all users.
9. Generate a signature σ on $((\mathsf{IK}_1, \mathsf{EK}_1), \mathsf{C_{pk}}, U, \{(H_j, \mathsf{CT}_j)\}_{j\in[2,n]}, \mathsf{SUK}, T)$ using ik_1.
10. Broadcast $((\mathsf{IK}_1, \mathsf{EK}_1), \mathsf{C_{pk}}, U, \{(H_j, \mathsf{CT}_j)\}_{j\in[2,n]}, \mathsf{SUK}, T)$ and σ.

Each user $i \in [2, n]$ receives $((\mathsf{IK}_1, \mathsf{EK}_1), \mathsf{C_{pk}}, U, \{(H_j, \mathsf{CT}_j)\}_{j\in[2,n]}, \mathsf{SUK}, T)$ and σ, and runs the following procedure.

[User i $(i \in [2, n])$]:

1. Check the validity of σ using the verification key IK_1. If σ is not valid, then abort.

[6]Here, the position j of the user i and the index of the ciphertext do not have to be the same. Here, for the sake of simplicity, we use the same j for both indexes.

2. If σ is valid, run $H \leftarrow \mathsf{Invert}(\mathsf{DK}_i^h, \mathsf{IK}_1, U)$. If $H \neq H_j$ for all $j = 2, 3, \ldots, n$, then abort. Let j be the smallest index such that $H = H_j$. Let $\mathsf{IK}||j||k$ be the decryption result of CT_j. If $\mathsf{IK}||j||k = \perp$, then abort. If $\mathsf{IK} \neq \mathsf{IK}_1$, then abort. Otherwise, decrypt C_{pk} using k and obtain $\{(\mathsf{IK}_i, \mathsf{EK}_i)\}_{i \in [2,n]}$.

3. Run $K_i \leftarrow \mathsf{KeyExchange}(\mathsf{ik}_i, \mathsf{IK}_1, \mathsf{ek}_i, \mathsf{SUK})$.

4. Compute the tree key tk using K_i and public keys T

5. Derive the two stage key $\mathsf{mack} \leftarrow \mathsf{KDF}(0, \mathsf{tk}, T)$ and $\mathsf{enck} \leftarrow \mathsf{KDF}(1, \mathsf{tk}, T)$.

Key Update Phase. In the key update phase, as in the ART protocol, a user randomly selects $K' \in \mathbb{Z}_p$ and computes a set of updated public keys. Since the user has enck and mack, the user encrypts his/her position information using enck (here, any IND-CPA symmetric encryption scheme can be employed) and computes a MAC on the ciphertext and the set of updated public keys using mack. The user broadcasts the MAC, the ciphertext, and the set of updated public keys, locally derives a new tk' and new stage keys enck' and mack', and removes previous tk, enck, and mack. When group members receive the MAC, the ciphertext, and the set of updated public keys, they derive a new tk' and new stage keys enck' and mack' and remove the previous tk, enck, and mack if the MAC is valid under mack. Specifically, each user $i \in [1, n]$ runs the procedure described below. Let the current secret key be K_i, and the position information be j.

[User i ($i \in [1, n]$)]:

1. Randomly choose $K_i' \in \mathbb{Z}_p$.

2. Compute public keys on the path (i.e., from the leaf node associated by the user i to the root node). Here, the public key of the root node is excluded because the secret key of the root node is regarded as the tree key. Let T' be a set of updated public keys.

3. Encrypt j using enck. Let C be the ciphertext.

4. Compute a MAC on $\mathsf{C}||T'$ using mack, and broadcast the MAC and $\mathsf{C}||T'$.

5. Remove enck and mack, and derive the new stage keys using the new tree key.

Each user $k \in [1, n] \setminus \{i\}$ receives the MAC and $\mathsf{C}||T'$, and runs the following procedure.

[User k ($k \in [1, n] \setminus \{i\}$)]:

1. Check the validity of the MAC using mack.

2. If the MAC is valid, then decrypt C and obtain position information j.

3. Remove enck and mack, and derive the new stage keys using the new tree key.

Security: First, we discuss whether or not the proposed modification affects the security of the ART protocol. A stage key for authentication mack is derived by a hash chain, as in the ART protocol. In the proposed protocol, a stage key for encryption enck is also derived from a different hash chain. These stage keys are derived from the previous stage keys (initialized to 0 for mack and 1 for enck, respectively) and the current tree key. Thus, even if they are disclosed, their impact is the same as that of the ART protocol. Moreover, enck is not used for authentication. Thus, its disclosure does not affect the PCS of the ART protocol. Finally, the technique for updating a key

is exactly the same as that of the ART protocol. Therefore, the proposed modification does not affect the FS and PCS in the ART protocol.

Next, we demonstrate that the proposed protocol provides membership privacy. In the setup phase, a set of public keys of group members $\{(\mathsf{IK}_i, \mathsf{EK}_i)\}_{i\in[2,n]}$ is encrypted. Thus, the ciphertext reveals no information on these keys. Moreover, due to the key privacy of the underlying public-key encryption scheme, the ciphertext does not disclose any information, which is used by PK_i for encryption. In addition, due to the anonymity of the hint system, the hint does not disclose any information, which is used by PK_i. In addition, due to the robustness of the underlying public-key encryption scheme and the hint system, a legitimate group member can decrypt a ciphertext, and the proposed modification does not affect the protocol execution. During the key update phase, a user who updates his/her own key, encrypts his/her position information using a common key (a stage key enck for encryption). Thus, the ciphertext does not disclose any information about the user who updates a key. We remark that the updated public keys are randomized due to the randomness of the leaf key. Thus, information about the user who updates a key is not disclosed by the updated public keys. If the length of a user path is different from that of other users, the anonymity breaks because anyone can distinguish whether two users who update the keys are the same or not. Therefore, a complete binary tree structure is assumed in the proposed protocol. If the number of group members is not a power of 2, dummy users are added (i.e., in the setup phase, the initiator selects a leaf secret key K_d for a dummy user d). Alternatively, it might be enough to encrypt T' by enck. However, usually encryption does not hide the plaintext size. In this case, padding is required.

5 Extensions

5.1 Sender-specific Authentication

If a shared common key is used for encryption and authentication, then there is no "cryptographic" proof of which group member sent a particular message. Based on this sender-specific authentication issue [23], Cohn-Gordon et al. [17] stated that "*We remark that it is easy to extend our design with such a system; in particular, by rotating and chaining signature keys, we conjecture that it is possible to achieve this authentication post-compromise.*" Here, they implicitly assume that a verification key and a group member are linked, i.e., they assume that signature verification implies sender identification. We note that they did not explicitly consider how to link verification keys and group members when these verification keys are updated.

In the proposed protocol, as in the ART protocol, the group members share the same stage keys, mack for authentication and enck for encryption. In these protocols, the position information, i.e., which leaf node is assigned to a user, is unique for each member. Although the position information does not identify the user, the group members can distinguish which group member (or more precisely which position in the tree) sent a particular message using the position information. However, this is not a cryptographic proof for sender-specific authentication. Moreover, proposing an SGM protocol with both membership privacy and sender-specific authentication is not a trivial task since they are opposite requirements for each other. As a simple attempt, according to the Cohn-Gordon et al. conjecture, each group member manages a signing key and a verification key, signs on a message to be sent, and encrypts the signature and the verification key in addition to the position information. However, there is still an issue on how to link verification keys and group members when these verification keys are updated.[7] The issues on how to provide sender-specific

[7] Ratcheting digital signatures [18] might be employed.

authentication in the ART protocol and how to simultaneously provide membership privacy and sender-specific authentication in the proposed protocol is left as a future work.

5.2 Dynamic Groups

Usually, users may join or leave the group after the initial setup phase. Cohn-Gordon et al. [17] stated that "*In general, since we build on tree-based ideas, our design can support join and leave operations using standard techniques.*", and they cited [25]. However, they did not provide a way to share a stage key. In the setup phase, the first stage key can be derived if the first tree key can be derived because the initial stage key is 0. However, after the setup phase, a stage key is derived from the current tree key and the previous stage key. Thus, when a user joins the group, it is necessary to consider how to share the last stage key needs. Moreover, in the proposed protocol, we employ a complete binary tree. Therefore, it is necessary to consider how a user joins or leaves the group without modifying the underlying tree structure.

First, we examine how a user joins the group after the setup phase in the proposed protocol is completed. Basically, a key shared by a strong one-round authenticated key-exchange protocol is utilized to encrypt the last stage keys, mack and enck. For simplicity, we set the maximum number of group members in the setup phase, where $n_{\mathsf{max}} = 2^m$ for some positive integer m. Since we employ a complete binary tree structure, we assume that there are dummy users not been assigned to actual users.

[Initiator]:

1. Choose a leaf that a dummy user is assigned.
2. As in the setup phase, locally run a strong one round authenticated key exchange protocol between the initiator and the user to be joined, and generate a key (say K).
3. Encrypt the current mack and enck and position information of the user using the shared key K.
4. Send the ciphertext to the user

[The User to be Joined]:

1. Receive the ciphertext from the initiator.
2. Locally run the strong one round authenticated key exchange protocol between the initiator and the user, and generate a key (say K).
3. Decrypt the ciphertext using K, and obtain mack, enck, and position information.
4. Update the key K as in the key update phase. Since a MAC is generated on the message to be sent using mack, other group members accept the key update.

To provide membership privacy, when a key is shared by a strong one-round authenticated key-exchange protocol and is used for encryption, it is necessary to assume that the ciphertext does not disclose any information on which $(\mathsf{IK}, \mathsf{EK})$ is used for the key-exchange protocol. Since a shared key is indistinguishable from a random value due to the security requirement of a strong one-round authenticated key-exchange protocol, the ciphertext is generated by a random key. Thus, intuitively, the ciphertext does not reveal such information. However, more formal discussion and analysis is required on this issue.

Next, we examine how a user leaves the group. Basically, the initiator updates the key of the user to be leaving the group and assigns a dummy user to the position of the user leaving the group.

[Initiator]:

1. Randomly choose a key K as a secret key of the dummy user.
2. As in the key update phase, compute a set of updated public keys, encrypt position information using enck, compute a MAC on the ciphertext and the set of updated public keys using mack.
3. Broadcasts the MAC, the ciphertext, and the set of updated public keys.
4. Locally derive new tk' and new stage keys enck' and mack', and remove previous tk, enck, and mack.

Note that the user to be leaving the group also has enck. Thus, the user can also decrypt the ciphertext. However, the user does not know the key K selected by the initiator. Thus, the user cannot derive the tree key. Therefore, the user leaves the group.

5.3 Group-size Hiding

We discuss group-size hiding. In the proposed protocol, the initiator broadcasts $((\mathsf{IK}_1, \mathsf{EK}_1), \mathsf{C}_{\mathsf{pk}}, U, \{(H_j, \mathsf{CT}_j)\}_{j\in[2,n]}, \mathsf{SUK}, T)$. Thus, the group size n is disclosed. Since the maximum number of group members is set to $n_{\mathsf{max}} = 2^m$ for some positive integer m, one simple solution is to add ciphertexts for dummy users $\{(H_j, \mathsf{CT}_j)\}_{j\in[2,n_{\mathsf{max}}]}$. Then, the actual group size n is hidden, although n_{max} is disclosed. For dynamic groups, the number of members left the group (e.g., the number of users who quit a service) is negative information.[8] We expect that the number of the members left the group is not revealed since the initiator just updates a key, instead of the users about to leave. More precisely, the information that the number of members left is less than the number of key updates is disclosed. On the other hand, when a user joins a group, the initiator sends a ciphertext to the user for sharing mack, enck, and position information. Thus, the number of users of the group after the setup phase is disclosed by the number of such ciphertexts. This means that just adding dummy users [6] in the group signature context does not work well in the SGM context. Techniques for hiding the group size are left as a future work.

5.4 Membership Privacy for Group Members

Our main goal is to provide membership privacy from the viewpoint of non group members. In addition, it would be better to discuss membership privacy for group members which may be useful in some cases (BTW, Chase et al. [11] do not consider this case). In the proposed protocol, the initiator sends a ciphertext of $\{(\mathsf{IK}_i, \mathsf{EK}_i)\}_{i\in[2,n]}$ to group members, and they know who the other members are from $\{(\mathsf{IK}_i, \mathsf{EK}_i)\}_{i\in[2,n]}$. Since we have removed $\{\mathsf{IK}_i\}_{i\in[1,n]}$ from the input of KDF, and members can generate a key via a strong one round authenticated key exchange protocol if they know $(\mathsf{IK}_1, \mathsf{EK}_1)$, our protocol works well even if $\{(\mathsf{IK}_i, \mathsf{EK}_i)\}_{i\in[2,n]}$ is not sent to group members. Concretely, the initiator encrypts $\mathsf{IK}_1||j$ using PK_j $(j = 2, 3, \ldots, n)$, and let CT_j be the ciphertext, and generates a signature σ on $((\mathsf{IK}_1, \mathsf{EK}_1), U, \{(H_j, \mathsf{CT}_j)\}_{j\in[2,n]}, \mathsf{SUK}, T)$ using ik_1. We remark that anonymity does not hold since a user who updates a key sends position information which can be

[8] In the group signature context, it has been discussed that one may guess the reason behind such circumstances, and this may lead to harmful rumors [19].

used for distinguishing whether two key updates are run by the same user or not. That is, the modified protocol provides "pseudonymity" for group members where each user is linkable via key update, and still provides membership privacy from the viewpoint of non group members.

5.5 How to Adopt our Technique to the MLS protocol

The ART protocol, which is our main target, has been updated and currently is known as the Messaging Layer Security (MLS) protocol. The Internet Engineering Task Force (IETF) has created a working group, and has discussed the MLS protocol. The security of MLS Draft 11 has been analyzed in [10] and currently (October 11, 2021), Draft 12 is the newest version [7]. In this section, we discuss whether our technique to provide membership privacy to the ART protocol can also be adopted for the MLS protocol.

As the main difference, in ART a user who updates a key does not have to know other members (i.e., the user notices own position information and updated public keys), whereas in MLS, or rather TreeKEM, a user who updates a key encrypts secret values assigned to nodes using public keys of other users. That is, the user needs to know other group members explicitly. This seems there is no way to provide membership privacy for group members. However, providing membership privacy from the viewpoint of non group members may be possible. First of all, the underlying Key Encapsulation Mechanism (KEM) scheme is required to be key-private, i.e., no information which public key is used for encryption is revealed from the ciphertext. However, it is not enough to provide membership privacy since if state information is revealed to an adversary, then the adversary knows membership information which is necessary to compute ciphertexts of node secret keys as above. One possible way is to introduce a common encryption key as in our protocol. Let assume that all KEM public keys are managed by the PKI, and each user knows who the group members are (simply, a group member list containing position information). When a user updates a key, then the user picks public keys from the PKI according to the list, and computes ciphertexts as usual. Then, using the common encryption key, the user encrypts the list, and broadcasts ciphertexts. Moreover, the user removes the list as in the previous secret keys, and updates the common encryption key using the new tree key as in our protocol. If a recipient of ciphertexts is a group member, then the member knows the common encryption key. Thus, the member can obtain the list, and know node secret values. If state information is revealed to an adversary, then the list is also revealed. However, if a user updates a key without involving the adversary, and if the group is dynamic (i.e., the list is modified), then still membership privacy is provided in the sense that the current list is not revealed to the adversary. Of course, the difference between the revealed list and the unrevealed list is just hidden, and it seems not enough to provide membership privacy as in the ART protocol. Further discussion and analysis are desired.

6 Conclusion

In this paper, an SGM protocol capable of providing membership privacy, hiding the participants, and also hiding the participant that sends a message was proposed. The proposed protocol is an extension of the ART protocol that maintains the FS and PCS capabilities of the original protocol. Our protocol achieves asymptotically the same efficiency of the ART protocol in the setup phase. Additional costs for key update do not depend on the number of group members (concretely, one encryption and decryption of a symmetric key encryption scheme and one execution of a key derivation function for each key update). Thus, we can add membership privacy to the ART protocol with a quite small overhead.

In the Cohn-Gordon et al. paper, the security of an unauthorized version of the protocol was analyzed. Thus, providing membership privacy by directly extending this model is not a trivial task. In fact, it is difficult to provide formal security models for secure (group) messaging protocols. Recently, Alwen et al. [5] proposed a formal security model for SGM, but no initiator was specified, unlike the ART protocol and our protocol. The extension of the Alwan et al. security model to provide membership privacy is left as a future work.

Cremers et al. [18] analyzed a case when a user participates in multiple groups. They showed that SGM such as ART and MLS provide significantly weaker PCS guarantees than Pairwise Signal, and considered global updates using PCS signatures that allows ART and MLS to achieve substantially stronger PCS guarantees. Considering Cremers et al. technique with membership privacy is also an interesting future work.

References

[1] Michel Abdalla, Mihir Bellare, and Gregory Neven. Robust encryption. In *TCC*, pages 480–497, 2010.

[2] Michel Abdalla, Mihir Bellare, and Gregory Neven. Robust encryption. *J. Cryptol.*, 31(2):307–350, 2018.

[3] Joël Alwen, Sandro Coretti, and Yevgeniy Dodis. The double ratchet: Security notions, proofs, and modularization for the Signal protocol. In *EUROCRYPT*, pages 129–158, 2019.

[4] Joël Alwen, Sandro Coretti, Yevgeniy Dodis, and Yiannis Tselekounis. Security analysis and improvements for the IETF MLS standard for group messaging. In *CRYPTO*, pages 248–277, 2020.

[5] Joël Alwen, Sandro Coretti, Yevgeniy Dodis, and Yiannis Tselekounis. Modular design of secure group messaging protocols and the security of MLS. In *ACM CCS*, pages 1463–1483, 2021.

[6] Michael Backes, Lucjan Hanzlik, and Jonas Schneider-Bensch. Membership privacy for fully dynamic group signatures. In *ACM CCS*, pages 2181–2198, 2019.

[7] Richard Barnes, Benjamin Beurdouche, Raphael Robert, Jon Millican, Emad Omara, and Katriel Cohn-Gordon. The Messaging Layer Security (MLS) Protocol. Internet-Draft draft-ietf-mls-protocol-12, Internet Engineering Task Force, October 2021. Work in Progress.

[8] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, and Jens Groth. Foundations of fully dynamic group signatures. In *ACNS*, pages 117–136, 2016.

[9] Jacqueline Brendel, Marc Fischlin, Felix Günther, and Christian Janson. PRF-ODH: relations, instantiations, and impossibility results. In *CRYPTO*, pages 651–681, 2017.

[10] Chris Brzuska, Eric Cornelissen, and Konrad Kohbrok. Cryptographic security of the MLS RFC, Draft 11. Cryptology ePrint Archive, Report 2021/137, 2021. https://ia.cr/2021/137.

[11] Melissa Chase, Trevor Perrin, and Greg Zaverucha. The signal private group system and anonymous credentials supporting efficient verifiable encryption. In *ACM CCS*, pages 1445–1459, 2020.

[12] David Chaum and Eugène van Heyst. Group signatures. In *EUROCRYPT*, pages 257–265, 1991.

[13] Kaiming Chen and Jiageng Chen. Anonymous end to end encryption group messaging protocol based on asynchronous ratchet tree. In *ICICS*, pages 588–605, 2020.

[14] Kaiming Chen, Jiageng Chen, and Jixin Zhang. Anonymous asynchronous ratchet tree protocol for group messaging. *Sensors*, 21(4):1058, 2021.

[15] Katriel Cohn-Gordon, Cas Cremers, Benjamin Dowling, Luke Garratt, and Douglas Stebila. A formal security analysis of the Signal messaging protocol. In *IEEE EuroS&P*, pages 451–466, 2017.

[16] Katriel Cohn-Gordon, Cas Cremers, and Luke Garratt. On post-compromise security. In *IEEE CSF*, pages 164–178, 2016.

[17] Katriel Cohn-Gordon, Cas Cremers, Luke Garratt, Jon Millican, and Kevin Milner. On ends-to-ends encryption: Asynchronous group messaging with strong security guarantees. In *ACM CCS*, pages 1802–1819, 2018.

[18] Cas Cremers, Britta Hale, and Konrad Kohbrok. The complexities of healing in secure group messaging: Why cross-group effects matter. In *USENIX Security*, pages 1847–1864, 2021.

[19] Keita Emura, Atsuko Miyaji, and Kazumasa Omote. An r-hiding revocable group signature scheme: Group signatures with the property of hiding the number of revoked users. *J. Appl. Math.*, 2014:983040:1–983040:14, 2014.

[20] Nelly Fazio and Irippuge Milinda Perera. Outsider-anonymous broadcast encryption with sublinear ciphertexts. In *Public Key Cryptography*, pages 225–242, 2012.

[21] Marc Fischlin and Felix Günther. Multi-stage key exchange and the case of Google's QUIC protocol. In *ACM CCS*, pages 1193–1204, 2014.

[22] Goichiro Hanaoka and Kaoru Kurosawa. Between hashed DH and computational DH: compact encryption from weaker assumption. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, 93-A(11):1994–2006, 2010.

[23] Eric J. Harder and Debby M. Wallner. Key Management for Multicast: Issues and Architectures. RFC 2627, 1999.

[24] Keitaro Hashimoto, Shuichi Katsumata, Kris Kwiatkowski, and Thomas Prest. An efficient and generic construction for Signal's handshake (X3DH): post-quantum, state leakage secure, and deniable. In *Public-Key Cryptography*, pages 410–440, 2021.

[25] Yongdae Kim, Adrian Perrig, and Gene Tsudik. Communication-efficient group key agreement. In *Trusted Information: The New Decade Challenge*, volume 193 of *IFIP Conference Proceedings*, pages 229–244. Kluwer, 2001.

[26] Benoît Libert, Kenneth G. Paterson, and Elizabeth A. Quaglia. Anonymous broadcast encryption: Adaptive security and efficient constructions in the standard model. In *Public Key Cryptography*, pages 206–224, 2012.

[27] Mriganka Mandal and Koji Nuida. Identity-based outsider anonymous broadcast encryption with simultaneous individual messaging. In *Network and System Security*, pages 167–186, 2020.

[28] Moxie Marlinspike. The X3DH key agreement protocol. Revision 1, 2016-11-04, https://signal.org/docs/specifications/x3dh/.

[29] Ian Martiny, Gabriel Kaptchuk, Adam J. Aviv, Daniel S. Roche, and Eric Wustrow. Improving Signal's sealed sender. In *NDSS*, 2021.

[30] Mario Di Raimondo, Rosario Gennaro, and Hugo Krawczyk. Deniable authentication and key exchange. In *ACM CCS*, pages 400–409, 2006.

[31] Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In *ASIACRYPT*, pages 552–565, 2001.

[32] Benedikt Schmidt, Simon Meier, Cas Cremers, and David A. Basin. Automated analysis of Diffie-Hellman protocols and advanced security properties. In *IEEE CSF*, pages 78–94, 2012.

[33] Nihal Vatandas, Rosario Gennaro, Bertrand Ithurburn, and Hugo Krawczyk. On the cryptographic deniability of the Signal protocol. In *ACNS*, pages 188–209, 2020.