Dependable and Secure Computing, IEEE Transactions on (Volume:PP , Issue: 99),03 February 2015

Group Key Agreement with Local Connectivity

Shaoquan Jiang

Abstract—In this paper, we study a group key agreement problem where a user is only aware of his neighbors while the connectivity graph is arbitrary. In our problem, there is no centralized initialization for users. A group key agreement with these features is very suitable for social networks. Under our setting, we construct two efficient protocols with passive security. We obtain lower bounds on the round complexity for this type of protocol, which demonstrates that our constructions are round efficient. Finally, we construct an actively secure protocol from a passively secure one.

Index Terms—Group key agreement, Diffie-Hellman, lower bound, authentication, protocol.

1 Introduction

Key agreement is a mechanism that allows two or more parties to securely share a secret key (called *a session key*). Starting from Diffie-Hellman [21] for the two-party case, this topic has been extensively studied in the literature [1], [3], [22], [25], [27], [28]. However, almost all the protocols assume a complete connectivity graph: any two users can communicate directly. In the real world, this is not always true. For instance, in social networks such as Facebook, Skype, Wechat and Google+, a user is only connected with his friends. For a group of users (e.g., the faculty union in a university) who wish to establish a session key, it is not necessary that any two of them are friends. But they might still be connected indirectly through the friend network. Of course, we can still regard them as directly connected by regarding the intermediate users as routers. However, this is quite different from a direct connection. First, indirectly connected users may not have the public information of each other (e.g., public-key certificate). Second, indirectly connected users may not know the existence of each other (e.g., in our faculty union example, one professor in one department may not know another professor in a different department). Third, a message between two indirectly connected users travels a longer time than that between directly connected users.

We study the group key agreement with an arbitrary connectivity graph, where each user is only aware of his neighbors and has no information about the existence of other users. Further, he has no information about the network topology. Under this setting, a user does not need to trust a user who is not his neighbor. Thus, if one is initialized using PKI, then he need not trust or remember public-keys of users beyond his neighbors.

• *Shaoquan Jiang is currently with the Institute of Information Security, Mianyang Normal University, Mianyang, China 621000. Email: shaoquan.jiang@gmail.com*

1.1 Related works

Key pre-distribution system (KPS) (a.k.a. non-interactive conference distribution system) [4], [12], [31], [35] can be regarded as a non-interactive group key agreement. In this case, the shared key of a given group is fixed after the setup. If a group is updated, then the group key changes to the shared key of the new group. The drawback of KPS is that the user key size is combinatorially large [12], [36] in the total number of users (if the system is unconditionally secure). Another drawback is that the group key of a given group can not be changed even if it is leaked unexpectedly (e.g., cryptanalysis of ciphertexts bearing this key). The key size problem may be overcome if a computationally secure system is used, while the key leakage problem is not easy. Further, computationally secure KPS is only known for the two-party case [21] and the three-party case [26]. KPS with a group size greater than 3 is still open [9].

A broadcast encryption [23] is a mechanism that allows a sender to send a group key to a selected set of users. This can be regarded as a group key agreement of one message that is sent by the sender. In a symmetric key based broadcast encryption [10], [23], the sender is a fixed authority. In this case, the user key size is combinatorially lower bounded [11]. In addition, it is secure only against a limited number of users. In a public key broadcast encryption [6], the key size problem can be waived. But one still has to set the threshold for the number of bad users. Also the ciphertext size depends on the number of users and hence could be large (e.g., it is $O(\sqrt{n})$ in [6] for n users). Further, users are initialized by a central authority which is not desired in our setting.

Traitor tracing [5], [7], [8] is a special broadcast encryption, where besides the usual broadcast capability, it can trace a pirate user: if a user helps build an illegal decryption device, he will be identified. This primitive inherits the drawbacks of a broadcast encryption.

A rekey scheme [18], [38] for a multicast can be regarded as a centralized dynamic broadcast encryption, where the authority always maintains the group as

the set of dynamically changing privileged users, and updates the group key and some user keys whenever there is a membership change. This mechanism has a drawback that the user key is provided by a centralized authority and has to be updated upon a member leave. If a group key agreement adopts this system, then the user key will be updated whenever the group changes. This is not desired. In addition, in a group key agreement setting, it is possible that several groups might simultaneously require to derive a group key. A rekeying scheme can not handle this case. So we can not adopt a rekeying scheme as a group key agreement.

In all of KPS, broadcast encryption, traitor tracing and a rekey scheme, a user key is set up by a single central authority and there is a dependency between the keys of different users. The first three mechanisms also have a threshold for the number of corruptions. In our key agreement problem, a centralized setup is not convenient and it is also impossible to determine a corruption threshold. Hence, they are not reasonable candidates for a group key agreement in our setting.

Unconditionally secure (interactive) key agreement has been considered in [3], [10], [12], [36]. Beimel and Chor [3] showed that the user key in this setting must be taken from a domain of size at least $|S|^{\tau}$, where S is the domain of the group key and τ is the maximum number of key agreements. If the user key is distributed uniformly (the typical case), it has an entropy of at least $\tau \log |S|$. In real applications, τ is usually large. Hence, this type of scheme does not provide an efficient solution even though it is unconditionally secure.

We now survey the computationally secure group key agreement in a passive model. This started from the Diffie-Hellman protocol [21]. In the following, we use the tuple (a, b, c) to represent a protocol that has a rounds, b elements of messages per user (the unit is a field element in $\mathbb{Z}_p^*$ for a large prime p) and computation cost c. Ingemarsson et al. [25] designed a group key agreement for n users in a ring with an efficiency tuple $(n-1, n-1, n\mathbf{e})$, where $\mathbf{e}$ stands for one exponentiation in $\mathbb{Z}_p^*$. Burmester and Desmedt [17] designed a more efficient protocol with an efficiency tuple $(2, 2n, 4\mathbf{e})$, after ignoring the exponentiations with small exponents and identifying one division with an exponentiation. Their protocol assumes a complete connectivity graph. Steiner et al. [37] proposed three protocols, where the most efficient one has an efficiency tuple $(n+1, 4, 5\mathbf{e})$. Their protocol assumes a complete connectivity graph and user n has a big computation cost of $(n-1)\mathbf{e}$ and a communication cost of $n-1$ messages. Wu et al. [39] proposed a transport-like protocol through a novel aggregate-signature based broadcast from pairing. Their protocol has one round (or 2 rounds if the group setup is counted), 3 elements of message from the initiator and computation cost of two pairings and one division, after the group public key has been setup, while the group public key needs one round and each user needs a cost $(1+n)\mathbf{e}+1\mathbf{p}+[3(n-1)+1]\mathbf{m}$, where $\mathbf{p}$ is a pairing and $\mathbf{m}$ is a multiplication. The strategy of a transport protocol using an aggregate public-key is also implemented by Lv et al. [32] using NTRU, although we feel that it is hard to obtain a provable security as NTRU does not have. Both [32], [39] assume a complete connectivity graph.

Some group key agreements handle a group change through join and leave operations (similar to the strategy of a rekey scheme in a multi-cast); see [1], [2], [14], [15], [22], [28], [29], [30].

We now survey the computationally secure group key agreement in an active model. Tzeng and Tzeng [34] proposed protocols in the random oracle model, where the interesting construction has an efficiency tuple $(2, 3n, 2n+1)$. Bresson et al. [16] formalized a formal model for a group key agreement in the active model and made the protocol in [37] actively secure using a signature based authentication. Katz and Yung [27] implemented the Burmester-Desmedt protocol in the active model with a signature based authenticator. Boyd and González-Nieto [13] proposed a transport-like 2-round protocol in the random oracle model, where one user needs to compute n public-key encryptions and each user has an outgoing message of length $\Omega(n)$. All the works in this paragraph assume a complete connectivity.

We are interested in a protocol without a random oracle. From the discussion above, we can see that the passively secure protocols that are really relevant to us are [17], [25], [37], [39]. They do not rely on a random oracle and the long term secrets (if any) between users do not have dependency. We will compare them with our protocols. The only issue when considered as a solution in our setting is that the connectivity graph in them is either a ring or a complete graph and is known to all users. In our setting, a user is only aware of his neighbors and has no information about others. For actively secure protocols surveyed above, [16], [27] are interesting. But they only implemented passively secure protocols [17], [37]. We will not compare them with us since we are mainly concerned with the key agreement methodology (instead of how to obtain stronger security).

1.2 Contribution

In this paper, we study a group key agreement with a local connectivity where a user is only aware of his neighbors while the connectivity graph is arbitrary. In our problem, there is no central authority to initialize users. Each of them can be initialized independently using PKI. A group key agreement for this setting is very suitable for applications such as a social network. Under our setting, we construct two efficient passively secure protocols. We also prove lower bounds on the round complexity which demonstrates that our protocols are round efficient. Finally, we construct an actively secure protocol from a passively secure one. The efficiency comparison between our protocols and the relevant protocols [17], [25], [37], [39] are presented in Table 1. In this table, $\mathcal{G}$ is the underlying connectivity graph for users, $\mathcal{G}^*$ is

prot	comp	comm	round	$d(\mathcal{G}^*)$
[25]	$(n-1)\mathbf{e}$	$n-1$	n	$n-1$
[17]	$2\mathbf{e}$	$2n$	4	2
[37]	$(n+1)\mathbf{e}$	4	5	2
[39]	$2\mathbf{p}+1\mathbf{e}$	3	1	2
DH-KA (**ours**)	$6\mathbf{e}$ ($4\mathbf{e}$)	6	$2d(\mathcal{G}^*)$	$d(\mathcal{G}^*)$
XO-KA (**ours**)	$3\mathbf{e}$	4	$d(\mathcal{G}^*)+1$	$d(\mathcal{G}^*)$

Fig. 1. Our protocols VS known relevant protocols ($d(\mathcal{G}^*)$ is the maximum distance between nodes in a spanning tree $\mathcal{G}^*$ of the connectivity graph $\mathcal{G}$)

a spanning tree of $\mathcal{G}$ and n is the number of users in $\mathcal{G}$. Further, $d(\mathcal{G}^*)$ is the maximum distance between nodes in $\mathcal{G}^*$. For example, $d(\mathcal{G}^*)=2$ for a star-like $\mathcal{G}^*$. Since the compared protocols [17], [25], [37], [39] are passively secure and assume a special (fixed) connectivity graph, our passively secure protocols in the table assume an arbitrary but fixed connectivity graph, due to which the cost to find $\mathcal{G}^*$ is not included. In our work, a dynamic graph is handled by first preprocessing a procedure to find $\mathcal{G}^*$ and then running our protocols in the table. In the table, Columns **comp**, **comm** and **round** are respectively the average computation cost per user, the average message size per user (with one unit being an element in $\mathbb{Z}_p^*$), and the round complexity. In the comparison, one should replace $d(\mathcal{G}^*)$ in our schemes by the concrete value of the compared protocol (e.g., $d(\mathcal{G}^*)=n-1$ for the protocol in [25]). The computation cost for **DH-KA** has $6\mathbf{e}$ when n is large (in comparison with $\log p$) and $4\mathbf{e}$ when n is small, where the difference comes from different approaches for arithmetic evaluations. We can see that our passively secure protocols are slightly less efficient than [39] but more efficient than others. Finally, it should stressed that all the compared protocols are not suitable for our application as we assume that users are only aware of their neighbors and the graph is arbitrary.

2 Preliminaries

Notations. We will use the following notions.

- For a set S, $x \leftarrow S$ samples x from S uniformly randomly;
- Function $\mu: \mathbb{N} \to \mathbb{R}$ is *negligible* if for any polynomial $p(x)$, $\lim_{n\to\infty} \mu(n)p(n)=0$.
- PPT stands for probabilistic polynomial time.
- $[n]$ denotes the set $\{1,\cdots,n\}$.
- $H(X) = -\sum_x P_X(x)\log P_X(x)$ is the *entropy* of random variable X and

$$H(X|Y) = -\sum_{x,y} P_{XY}(x,y)\log P_{X|Y}(x|y)$$

is the *conditional entropy* of X given Y.

- $I(X;Y)=H(X)-H(X|Y)$ is the *mutual information* between X and Y; $I(X;Y|Z)=H(X|Z)-H(X|YZ)$ is the *conditional mutual information* between X and Y, when Z is given.

2.1 Indistinguishability

Two ensembles are *indistinguishable* if no efficient algorithm can tell them apart. This notion was first proposed by Goldwasser and Micali [24] in case of encryption. Generally, it was due to Yao [40].

Definition 1: Ensembles $X=\{X_\lambda\}_{\lambda\geq 1}$ and $Y=\{Y_\lambda\}_{\lambda\geq 1}$ are *indistinguishable* if for any PPT algorithm $\mathcal{D}$, $|\Pr[\mathcal{D}(X_\lambda)=1]-\Pr[\mathcal{D}(Y_\lambda)=1]|$ is negligible.

In a cryptographic system, λ usually is the security parameter and implicitly defined. For example, in a RSA system, λ is the bit length of the modulus N.

2.2 Decisional Diffie-Hellman assumption

Let p,q be two large primes and $q|(p-1)$. Let $\mathbb{G}$ be the subgroup of $\mathbb{Z}_p^*$ of order q and g be a generator of $\mathbb{G}$. The decisional Diffie-Hellman assumption is as follows.

Definition 2: The *decisional Diffie-Hellman assumption* (DDH) holds if (g,g^x,g^y,g^{xy}) and (g,g^x,g^y,g^z) are indistinguishable when $x,y,z \leftarrow \mathbb{Z}_q$.

The following lemma can be easily proved by a hybrid reduction and it appeared in [15].

Lemma 1: [15] Let $n\in\mathbb{N}$. Then, under the *DDH* assumption, $\{g^{a_ia_j} \mid 1\leq i<j\leq n\}\cup\{g,g^{a_1},\cdots,g^{a_n}\}$ and $\{g^{a_{ij}} \mid 1\leq i<j\leq n\}\cup\{g,g^{a_1},\cdots,g^{a_n}\}$ are indistinguishable, where a_{ij} $(1\leq i<j\leq n)$ and $a_1,\cdots,a_n$ are all uniformly random from $\mathbb{Z}_q$.

3 Formal Model

3.1 Syntax

Let $\mathcal{U}=\{1,\cdots,N\}$ be the universe of users who are connected by an undirected connected graph $\mathcal{G}_\mathcal{U}$. Assume the set of neighbors for $i\in\mathcal{U}$ is $\mathcal{U}_i\subseteq\mathcal{U}$. We assume user i knows $\mathcal{U}_i$. We will define a key agreement on any *undirected* connected subgraph $\mathcal{G}=(V,E)$ of $\mathcal{G}_\mathcal{U}$. The set of neighbors of i in $\mathcal{G}$ is denoted by $\mathcal{N}_i(\mathcal{G})$. The protocol allows users in V to agree on a shared key. Each user i in the protocol can only send messages to his neighbors $\mathcal{N}_i(\mathcal{G})$. Since user i has no knowledge about users other than $\mathcal{U}_i$, we must facilitate him to determine $\mathcal{N}_i(\mathcal{G})$. Toward this, we assume that there is a basic description of $\mathcal{G}$ (denoted by $basic(\mathcal{G})$) such that with $\mathcal{U}_i$ and $basic(\mathcal{G})$, user i can easily determine $\mathcal{N}_i(\mathcal{G})$. $basic(\mathcal{G})$ is determined by the protocol initiator and it will appear in the first incoming message of any user (other than the initiator) in $\mathcal{G}$, which for simplicity will not be mentioned again later. The syntax is as follows.

Definition 3: Let $\mathcal{U}=\{1,\cdots,N\}$ be the universe of users connected by an undirected connected graph $\mathcal{G}_\mathcal{U}$, where user i has a neighbor set $\mathcal{U}_i$. Group key agreement Π with a local connectivity is a mechanism with the following components.

- **Setup**(1^λ). Upon 1^λ, a system parameter sp is generated. For each $i\in\mathcal{U}$, a public key PK_i and a

private key SK_i are generated. $(\mathsf{sp}, PK_1, \cdots, PK_N)$ is public while SK_i is only known to user i.

- **Key Agreement**. For an undirected connected subgraph $\mathcal{G} = (V, E)$ of $\mathcal{G}_\mathcal{U}$, initiated by some $I \in V$ with input $basic(\mathcal{G})$, users in V interact with their neighbors in $\mathcal{G}$ and finally all of them derive a group key sk. The protocol is complete: if users in V follow the protocol, they derive the same sk.

As an example, let $\mathcal{G}_\mathcal{U}$ be a connected social network and $\mathcal{G}$ be a university faculty union who organizes its members on $\mathcal{G}_\mathcal{U}$. Each professor has his own friend list $\mathcal{U}_i$ in $\mathcal{G}_\mathcal{U}$. Given the name "faculty union", professor i can determine $\mathcal{N}_i(\mathcal{G})$, assuming that he knows that who in his friend list is a professor and who is not (this will be a very reasonable assumption). Now if a professor wishes the union to compute a union key. He can send the request "faculty union key" to his union neighbors and interact with them, who then continue the similar interaction with their own union neighbors, and so on. Finally, union members can obtain a group key.

3.2 Security definition

Before formally defining the security, we introduce the following notions.

- $\Pi_i^{\ell_i}$. This is an *instance* (or *session*) in user i and ℓ_i is the instance id that differentiates it from other instances in the same user.
- $\mathsf{stat}_i^{\ell_i}$. This is the *internal state* of instance $\Pi_i^{\ell_i}$.
- $\mathsf{sid}_i^{\ell_i}$. This is the *session identifier* of instance $\Pi_i^{\ell_i}$. Its exact functionality will be discussed in the specification of *partnering* later.
- $\mathsf{pid}_i^{\ell_i}$. This is the set of neighbors that $\Pi_i^{\ell_i}$ supposedly is directly interacting with.
- $sk_i^{\ell_i}$. this is the group key derived by $\Pi_i^{\ell_i}$.

Partnering. By *partnering*, we wish to capture the intuition that two partnered instances must attend the same protocol execution. Formally, two instances $\Pi_i^{\ell_i}$ and $\Pi_j^{\ell_j}$ are **directly partnered** if

1. $i \in \mathsf{pid}_j^{\ell_j}$ and $j \in \mathsf{pid}_i^{\ell_i}$;
2. $\mathcal{S}(\mathsf{sid}_i^{\ell_i}, \mathsf{sid}_j^{\ell_j}) = 1$, where $\mathcal{S}$ is a Boolean function that will be defined w.r.t. the concrete protocol.

Condition (1) intends to say that $\Pi_i^{\ell_i}$ interacts with user j and that $\Pi_j^{\ell_j}$ interacts with user i. This condition implicitly implies that j and i are neighbors. Condition (2) intends to say that $\Pi_i^{\ell_i}$ and $\Pi_j^{\ell_j}$ have consistent session identifiers and hence they are jointly executing the agreement. If uses i and j are not neighbors, we can generalize the partnership as follows. $\Pi_i^{\ell_i}$ and $\Pi_j^{\ell_j}$ are **partnered** if there exists a sequence of instances $\Pi_{i_1}^{\ell_{i_1}}, \cdots, \Pi_{i_t}^{\ell_{i_t}}$ such that $\Pi_{i_k}^{\ell_{i_k}}, \Pi_{i_{k+1}}^{\ell_{k+1}}$ are directly partnered for all $k = 1, \cdots, t-1$, where $i_1 = i, i_t = j$.

In the security model, we wish to capture the concern that an attacker $\mathcal{A}$ can adaptively choose a connected subgraph $\mathcal{G}$ of $\mathcal{G}_\mathcal{U}$ and request to execute the key agreement on it. He can also corrupt users and obtain their long term secrets. He can request to obtain the group key of any session. If he is an active attacker, he can also launch a man-in-the-middle attack. In the following, these adversarial capabilities are formalized by allowing $\mathcal{A}$ to adaptively access the set of oracles that are maintained by a challenger.

Initially, challenger generates sp, (PK_1, SK_1), $\cdots$, (PK_N, SK_N), provides sp and $PK_1, \cdots, PK_N$ to $\mathcal{A}$ and answers his oracle queries as follows.

- **Send**(i, ℓ_i, M). $\mathcal{A}$ can send any M to $\Pi_i^{\ell_i}$ and the oracle will reply according to the specification. If $\Pi_i^{\ell_i}$ does not exist in user i, then it will be immediately created and in this case M should contain $basic(\mathcal{G})$. Allowing $\mathcal{A}$ to invoke a new initiator $\Pi_i^{\ell_i}$ suggests that key agreement events are scheduled by $\mathcal{A}$. Generally, a **Send** query represents an active attack. We assume that when $\Pi_i^{\ell_i}$ receives M, he knows the claimed sender of M. Practically, this sender identity could be stated in M or has been mentioned in the hello messages at the initialization stage of the two parties' communication. Of course, since a **Send** query represents an active attack, this claimed sender could be impersonated by $\mathcal{A}$.
- **Corrupt**(i). Upon this, SK_i is provided to $\mathcal{A}$.
- **Reveal**(i, ℓ_i). Upon this, $sk_i^{\ell_i}$ is provided to $\mathcal{A}$ if $\Pi_i^{\ell_i}$ has successfully completed with $sk_i^{\ell_i}$ defined.
- **Execute**$(i_1, \ell_1, \cdots, i_t, \ell_t)$. Upon this, a key agreement over a connected subgraph $\mathcal{G}$ (formed by $i_1, \cdots, i_t$) is executed. Finally, the transcript tr is returned to $\mathcal{A}$. This presents a passive attack. Although it can be replaced by a sequence of Send queries, it allows us later to simplify the definition of passive security.
- **Test**(s, ℓ_s). This query is for the security test. It can be called only once. Upon this, it is assumed that $\Pi_s^{\ell_s}$ has successfully completed with $sk_s^{\ell_s}$ defined and that $\mathcal{A}$ will never *compromise* $\Pi_s^{\ell_s}$ (see the definition below). In this case, the oracle takes $\alpha_1 \leftarrow \mathcal{K}, b \leftarrow \{0, 1\}$ and defines $\alpha_0 = sk_s^{\ell_s}$. Finally, α_b is provided to $\mathcal{A}$ who will try to output a guess bit b' for b. Toward this, he can continue to query other oracles. Finally, he succeeds if $b' = b$.

$\Pi_i^{\ell_i}$ is *compromised* if a **Reveal** query was issued to $\Pi_i^{\ell_i}$ or one of its partnered instances, or if i or j (who has a partnered instance for $\Pi_i^{\ell_i}$) was corrupted.

Now we are ready to define the security of a group key agreement. We use **Succ**$(\mathcal{A})$ to denote the success of $\mathcal{A}$ in the **Test** query. We will define the passive security by removing the **Send** oracle.

Definition 4: A group key agreement Π for a connected $\mathcal{G}_\mathcal{U}$ is **secure** if for any PPT adversary $\mathcal{A}$, $P(\mathbf{Succ}(\mathcal{A})) \leq 1/2 + negl(\lambda)$. Further, Π is **passively secure** if $P(\mathbf{Succ}(\mathcal{A})) \leq 1/2 + negl(\lambda)$ when **Send** oracle is removed.

Some works (e.g., [13]) emphasized the contributiveness of a key agreement: the randomness of a group key should come from all users. Under this, the randomness of the group key will be guaranteed even if some users have bad random sources. We capture this through strengthening the security definition as follows.

Definition 5: A group key agreement Π for a connected $\mathcal{G}_{\mathcal{U}}$ is **(passively) secure with contributiveness** if it is (passively) secure under Definition 4 and is contributive:

- Let R_j be user j's random input. Let $\mathcal{G} = (V, E)$ (with $|V| \geq 2$) be any connected subgraph. If users in V agree on a group key sk, then, for any $i \in V$,

$$I\Big(sk; \{SK_j, R_j\}_{j \in V \setminus \{i\}} \Big| \pi\Big) = 0, \tag{1}$$

  where $\pi = (\mathsf{sp}, \mathcal{G}, PK_1, \cdots, PK_N)$.

Eq. (1) requires that given the public information, sk is independent of $\{SK_j, R_j\}_{j \in V \setminus \{i\}}$. As sk is deterministic in $\{SK_j, R_j\}_{j \in V}$ and π, sk must depend on (SK_i, R_i).

3.3 Efficiency

Now we consider the efficiency measure of a group key agreement. The widely known measures are computation cost, communication complexity and round complexity. The *computation cost* of a user usually is defined as the number of time-consuming operations such as a modular exponentiation. The *communication complexity* is defined as the total traffic length of the protocol. To define the round complexity, we assume the protocol proceeds in rounds. The number of rounds in which a protocol proceeds is called its *round complexity*.

4 Passively Secure Constructions

In this section, we present two passively secure constructions. We assume that at the beginning of the protocol, all parties in $\mathcal{G}$ are already notified the key agreement event (and so they can start the protocol simultaneously). We call it a *starting assumption*. This assumption is needed only to count the round complexity. It has been implicitly assumed by many protocols in the literature (e.g., [17]). In our constructions, without this assumption, a user will not start until he receives the first message while the whole protocol starts from an initiator. Under this, the passive security of our protocols remains unchanged but the round complexity becomes larger. It might be surprising: if a user in our setting is only aware of his neighbors, how can all users be notified of the key agreement event before the protocol starts? We remark that the protocols in this section are only passively secure and ultimately they need to be made actively secure. In Section 6, this is done through a two-stage protocol: stage 0 is a preprocessing stage which notifies each party of the key agreement event (starting from an initiator) and stage 1 is a real transformation from a passively secure protocol to an actively secure one, where the starting assumption has been implemented in stage-0.

We will first present the constructions for a graph $\mathcal{G}$ that is a tree. Then, we will extend them to a general connected graph. The first construction can be regarded as a group Diffie-Hellman with a local connectivity. The second construction essentially is a private coin tossing protocol protected by a Diffie-Hellman key.

4.1 When $\mathcal{G}$ is a tree: the first scheme

Let p, q be large primes with $q|p-1$ and g be a generator of the group $\mathbb{G}$ of order q in $\mathbb{Z}_p^*$. Assume that p, q, g are all public. Let $(\mathcal{E}_\rho, \mathcal{D}_\rho)$ be a symmetric encryption scheme with a secret key ρ. Let $\mathcal{G} = (V, E)$ be an undirected connected graph for $V \subseteq \mathcal{U}$. Let $\mathcal{N}_i = \mathcal{N}_i(\mathcal{G})$. The protocol is formally described in Fig. 4. However, it might be useful to give more explanations here. The protocol contains three stages.

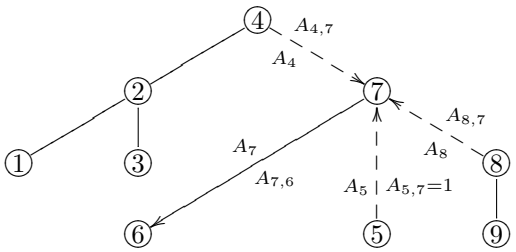

Fig. 2. Message formation from node 7 to 6 in stage one

In Stage one, each ℓ sends to each of his neighbor i: his own temporary Diffie-Hellman (DH) public key as well as the merged temporary DH public key of users in the subtree ℓ (excluding ℓ) of node i. Specifically, $\ell \in V$ sends $(A_{\ell i}, A_\ell)$ to each neighbor i, where he defines $A_\ell = g^{a_\ell}$ by taking a secret $a_\ell \leftarrow \mathbb{Z}_q$ and $A_{\ell i}$ is prepared as follows. If user ℓ is a leaf, $A_{\ell i} = 1$. Generally, $A_{\ell i} = \prod_{j \in \mathcal{N}_\ell \setminus \{i\}} A_j A_{j\ell}$, which is iteratively defined starting from leaf users. For instance, in Fig. 2, $A_{7,6} = A_4 A_{4,7} \cdot A_8 A_{8,7} \cdot A_5 A_{5,7}$ and toward this, node 7 must first receive $(A_{8,7}, A_8)$ from node 8, $(A_{4,7}, A_4)$ from node 4 and $(A_{5,7}, A_5)$ from node 5. This could need several rounds in the protocol. For instance, $(A_{4,7}, A_4)$ is sent to node 7 in the 3th round in Stage one while $(A_{5,7}, A_5)$ is sent to node 7 in the 1st round. Later, we will show in the completeness that $A_{\ell i}$ is actually the product of A_j for all j in the subtree ℓ (excluding ℓ) of node i (where we regard $\mathcal{G}$ as a tree rooted at i). For instance, $A_{7,6} = A_1 A_2 A_3 A_4 A_5 A_8 A_9$ and $A_{6,7} = A_{5,7} = 1$.

In Stage two, each ℓ sends to each of his neighbor i: a partial group secret that is a merged result of DH keys of all indirectly connected user pairs such that each pair has a user in the subtree ℓ of node i, where this partial group secret is sent under the encryption of the pairwise DH key between ℓ and i. Specifically, user ℓ prepares and sends an encrypted $L_{\ell i}$ to each neighbor i, where the encryption uses the Diffie-Hellman key $\rho_{\ell i} = g^{a_\ell a_i}$ between i and ℓ (the encryption is denoted by [ ] in Fig. 3). Here $L_{\ell i}$ is

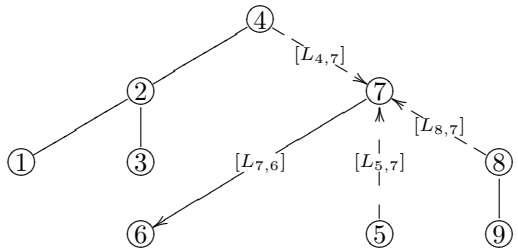

Fig. 3. Message formation from node 7 to 6 in stage two

defined as $L_{\ell i} = (\Pi_{j\in\mathcal{N}_\ell\backslash\{i\}} L_{j\ell}) \cdot (\Pi_{j\in\mathcal{N}_\ell} A_{j\ell})^{a_\ell}$, which again is defined iteratively starting from leaf users. For instance, $L_{7,6} = L_{4,7}L_{8,7}L_{5,7} \cdot (A_{4,7}A_{8,7}A_{5,7}A_{6,7})^{a_7}$ in Fig. 3. Here, $L_{\ell i}$ can be computed only if each product term $L_{j\ell}$ has been received by ℓ. Later, we will show in the completeness that $L_{\ell i}$ is the product of all Diffie-Hellman key $g^{a_j a_u}$, where j is in the subtree ℓ (including ℓ) of node i and u is arbitrary as long as $(j,u) \notin E$ and $j \neq u$. For instance, $L_{8,7} = g^{a_8(a_1+\cdots+a_6)+a_9(a_1+\cdots+a_7)}$ in Fig. 3.

In Stage three, each user computes the group key using his own secret and the partial group secrets he received in stage two. Specifically, each user ℓ computes the group key $sk = \prod_{s\in\mathcal{N}_\ell}(L_{s\ell} \cdot A_{s\ell}^{a_\ell})$. For instance, in Fig. 3, node 2 computes $sk = L_{1,2}A_{1,2}^{a_2} \cdot L_{3,2}A_{3,2}^{a_2} \cdot L_{4,2}A_{4,2}^{a_2}$. Later we will show in the completeness that sk is the product of all $g^{a_j a_u}$ for $(j,u) \notin E$ and $j \neq u$. Since sk does not depend on ℓ, it is a shared key among all users.

With the notions of $A_\ell, A_{\ell i}$ and $L_{\ell i}$, we can now conveniently reveal the design idea of our protocol (Fig. 4). Use $\mathcal{G}$ in Fig. 3 as an example. Roughly, we intend to design the protocol such that sk is the product of all $g^{a_j a_u}$ for any pair of users (j,u) who are not neighbors in $\mathcal{G}$ (here excluding neighboring (j,u) is for the security proof purpose only and will be justified soon). That is, $sk = \prod_{(j,u)\notin E, j\neq u} g^{a_j a_u}$. To admit user ℓ to compute sk, we intend to partition $\Omega = \{(j,u) \mid (j,u) \notin E, j \neq u\}$ according to his neighbors and himself. Take $\ell = 7$ as an example. His neighbors are $4,5,6,8$. Partition $\Omega = \Omega_4 \cup \Omega_5 \cup \Omega_6 \cup \Omega_8 \cup \Omega_7$. Here Ω_4 is defined as the set of all (j,u) for $j \in \{1,2,3,4\}$ and u arbitrary such that $(j,u) \notin E$ and $j \neq u$. That is, the set of j is the subtree 4 (including 4) of node 7 and u is arbitrary as long as $(j,u) \notin E$ and $j \neq u$. If we recall the specific meaning of $L_{4,7}$ in the explanation of Stage two, we will see that $L_{4,7} = \prod_{(j,u)\in\Omega_4} g^{a_j a_u}$. Sets $\Omega_5, \Omega_6, \Omega_8$ are defined similarly. The remaining set Ω_7 in the partition is necessarily the set of $(7,u)$ for all $u \neq 7$, except $u \in \mathcal{N}_7$. Looking at the specific meaning of $A_{j\ell}$ at Stage 1, we know that $\prod_{(7,u)\in\Omega_7} g^{a_7 a_u} = (\prod_{j\in\mathcal{N}_7} A_{j7})^{a_7}$. So in order for node 7 to compute sk, we only need to request each $j \in \mathcal{N}_7$ to send $A_{j,7}, L_{j,7}$ to node 7. Finally, as $L_{j,7}$ must be sent confidentially, we need to encrypt it which can be done using a Diffie-Hellman key $g^{a_j a_7}$.

Finally, one might wish to design the protocol such that sk is the product of $g^{a_j a_u}$ for all pairs (j,u) with $j \neq u$ (instead of additionally requiring $(j,u) \notin E$ in our protocol). We do not choose this way because in this case $g^{a_j a_u}$ is sometimes used as a product term in $L_{\ell i}$ and sometimes is used as a secret key for the encryption. This will complicate the security proof as generally the security of an encryption scheme requires that the key and the message should be independent.

Stage one.

0. Each $i \in V$ takes $a_i \leftarrow \mathbb{Z}_q$ and sets $A_i = g^{a_i}$.
1. Each leaf user s in $\mathcal{G}$ (i.e., $\mathcal{N}_s = \{i\}$) sets $A_{si} = 1$ and sends (A_{si}, A_s) to i.
2. **[Loop]** Each $\ell \in V$ does the following. For $i \in \mathcal{N}_\ell$, if user ℓ has received $(A_{j\ell}, A_j)$ from each $j \in \mathcal{N}_\ell\backslash\{i\}$ and did not send $(A_{\ell i}, A_\ell)$ to i, then he computes $A_{\ell i} = \Pi_{j\in\mathcal{N}_\ell\backslash\{i\}} A_j A_{j\ell}$ and sends $(A_{\ell i}, A_\ell)$ to i.
3. Each user ℓ continues step 2 until he has sent $(A_{\ell i}, A_\ell)$ to user i for each $i \in \mathcal{N}_\ell$, in which case, he proceeds to Stage two. Let $\rho_{\ell i} = g^{a_\ell a_i}$.

Stage two.

1. Each leaf user s (i.e., $\mathcal{N}_s = \{i\}$) computes $L_{si} = A_{is}^{a_s}$ and sends $C_{si} = \mathcal{E}_{\rho_{si}}(L_{si})$ to user i.
2. **[Loop]** Each $\ell \in V$ does the following. For $i \in \mathcal{N}_\ell$, if user ℓ has received $C_{j\ell}$ from each $j \in \mathcal{N}_\ell\backslash\{i\}$ and did not send $C_{\ell i}$ to i, then he decrypts $L_{j\ell} = \mathcal{D}_{\rho_{j\ell}}(C_{j\ell})$, defines $L_{\ell i} = (\Pi_{j\in\mathcal{N}_\ell\backslash\{i\}} L_{j\ell}) \cdot (\Pi_{j\in\mathcal{N}_\ell} A_{j\ell})^{a_\ell}$ and sends $C_{\ell i} = \mathcal{E}_{\rho_{\ell i}}(L_{\ell i})$ to user i.
3. Each user ℓ continues step 2 until he has sent $C_{\ell i}$ to user i for each $i \in \mathcal{N}_\ell$, in which case, he proceeds to Stage three.

Stage three (*group key derivation*).

- Upon $C_{s\ell}$ for all $s \in \mathcal{N}_\ell$, user ℓ decrypts $L_{s\ell} = \mathcal{D}_{\rho_{s\ell}}(C_{s\ell})$ (if not done before) and calculates group key $sk = \prod_{s\in\mathcal{N}_\ell}(L_{s\ell} \cdot A_{s\ell}^{a_\ell}) = \prod_{(u,v)\notin E, u\neq v} g^{a_u a_v}$.

Fig. 4. Our Protocol DH-KA

4.1.1 Completeness

We consider the completeness: if all users follow the protocol, they derive the same group key.

For each edge $(\ell, i) \in E$, define $\mathcal{S}_{\ell i}$ to be the set of users in the subtree ℓ (but excluding ℓ) of node i (regard $\mathcal{G}$ as a tree rooted at i). For instance, $\mathcal{S}_{9,8} = \emptyset$, $\mathcal{S}_{8,9} = \{1,\cdots,7\}$ and $\mathcal{S}_{7,8} = \{1,\cdots,6\}$ in Fig. 3.

The completeness follows from two claims below.

Claim 1. *$A_{\ell i} = \prod_{j\in\mathcal{S}_{\ell i}} A_j$ and $A_\ell \cdot \prod_{i\in\mathcal{N}_\ell} A_i A_{i\ell} = \prod_{j\in V} A_j$.*

Proof. Consider part one first. Use induction. This claim is true for a leaf user ℓ as $\mathcal{S}_{\ell i} = \emptyset$. If the claim is true for all children of ℓ (with parent i in $\mathcal{G}$), then from $A_{\ell i} = \prod_{j\in\mathcal{N}_\ell\backslash\{i\}} A_j A_{j\ell}$ and the definition of $\mathcal{S}_{\ell i}$, the claim is true for user ℓ. For part two, $A_\ell \cdot \prod_{i\in\mathcal{N}_\ell} A_i A_{i\ell} = \prod_{j\in V} A_j$

follows from part one, as $\mathcal{G}$ is a tree and hence any user j for $j \neq \ell$ must be rooted at a unique neighbor of ℓ. □

By Claim 1, the meaning of $A_{\ell i}$ is evident. It is the product of g^{a_j} for all j in the subtree ℓ (excluding ℓ) of node i.

We will show in Claim 2 below that $L_{\ell i}$ is the product of $g^{a_j a_u}$ for all $j \in \mathcal{S}_\ell \cup \{\ell\}$ and u is arbitrary such that $(j,u) \notin E$ and $j \neq u$. We will also show that sk in Stage three is the product of $g^{a_t a_u}$ for all $(t,u) \notin E$ and $t \neq u$. The proof idea is basically to manipulate sets $S_{\ell i}$ for all ℓ, i and watch its specific meaning in the graph $\mathcal{G}$.

Claim 2. $L_{\ell i} = \prod_{j \in \mathcal{S}_{\ell i} \cup \{\ell\}} \prod_{u:(j,u)\notin E, j\neq u} g^{a_u a_j}$ *and* $\prod_{i \in \mathcal{N}_\ell} (L_{i\ell} \cdot A_{\ell i}^{a_\ell}) = \prod_{(t,u)\notin E,\ t\neq u} g^{a_u a_t}$.

Proof. Consider part one first. Notice that $A_{i\ell} = \prod_{j \in \mathcal{S}_{i\ell}} A_j$ by Claim 1. Hence, for a leaf node ℓ with (only) neighbor i, the claim follows easily as $L_{\ell i} = A_{i\ell}^{a_\ell}$, $\mathcal{S}_{i\ell} = V \backslash \{i, \ell\}$ and $\mathcal{S}_{\ell i} = \emptyset$. If the claim is true for all children of user ℓ (in $\mathcal{G}$ with root i), then using $\biguplus_{j \in \mathcal{N}_\ell \backslash \{i\}} (\mathcal{S}_{j\ell} \cup \{j\}) = \mathcal{S}_{\ell i}$, we have

$$\Pi_{j \in \mathcal{N}_\ell \backslash \{i\}} L_{j\ell} = \prod_{j \in \mathcal{S}_{\ell i}} \prod_{u:(j,u)\notin E, j\neq u} g^{a_u a_j}. \quad (2)$$

Further, as $\biguplus_{j \in \mathcal{N}_\ell} \mathcal{S}_{j\ell} = V \backslash (\mathcal{N}_\ell \cup \{\ell\})$, by Claim 1,

$$\begin{aligned} (\Pi_{j \in \mathcal{N}_\ell} A_{j\ell})^{a_\ell} &= \prod_{u \in V \backslash (\mathcal{N}_\ell \cup \{\ell\})} A_u^{a_\ell} \\ &= \prod_{u:\ (\ell,u)\notin E, u\neq \ell} A_u^{a_\ell}, \end{aligned} \quad (3)$$

as $(\ell, u) \in E$ if and only if $u \in \mathcal{N}_\ell$. So Eqs. (2)(3) give

$$\begin{aligned} L_{\ell i} &= (\Pi_{j \in \mathcal{N}_\ell \backslash \{i\}} L_{j\ell}) \cdot (\Pi_{j \in \mathcal{N}_\ell} A_{j\ell})^{a_\ell} \\ &= \prod_{j \in \mathcal{S}_{\ell i} \cup \{\ell\}} \prod_{u:(j,u)\notin E, j\neq u} A_u^{a_j}. \end{aligned} \quad (4)$$

So the claim holds for user ℓ and part one follows. For part two, notice that $\biguplus_{j \in \mathcal{N}_\ell} (\mathcal{S}_{j\ell} \cup \{j\}) = V \backslash \{\ell\}$. Using part one and Eq. (3), we have

$$\begin{aligned} &\prod_{j \in \mathcal{N}_\ell} (L_{j\ell} \cdot A_{j\ell}^{a_\ell}) \\ = &\Big(\prod_{t \in V \backslash \{\ell\}} \prod_{u:(t,u)\notin E, t\neq u} g^{a_u a_t} \Big) \times \\ &\Big(\prod_{u:\ (\ell,u)\notin E, \ell\neq u} g^{a_u a_t} \Big) \\ = &\Big(\prod_{t \in V} \prod_{u:(t,u)\notin E, t\neq u} g^{a_u a_t} \Big) \\ = &\prod_{t,u:\ (t,u)\notin E, t\neq u} g^{a_u a_t}. \end{aligned}$$

The part two follows. □

4.1.2 Efficiency

Computation cost. User ℓ needs to compute A_ℓ, and for each $i \in \mathcal{N}_\ell$, $A_{\ell i} = \prod_{j \in \mathcal{N}_\ell \backslash \{i\}} A_j A_{j\ell}$, $\rho_{\ell i} = A_i^{a_\ell}$ and $L_{\ell i} = (\Pi_{j \in \mathcal{N}_\ell \backslash \{i\}} L_{j\ell}) \cdot (\Pi_{j \in \mathcal{N}_\ell} A_{j\ell})^{a_\ell}$, where $\mathcal{D}$, $\mathcal{E}$ are relatively cheap light weight operations and ignored, and in addition, based on these computations, only one multiplication is further needed to compute sk in stage three and is ignored too. So the cost of user ℓ is

$$3|\mathcal{N}_\ell|^2 \mathbf{m} + (|\mathcal{N}_\ell| + 2)\mathbf{e},$$

where $\mathbf{e}$ is exponentiation, $\mathbf{m}$ is multiplication. When $\max_\ell |\mathcal{N}_\ell| \ll \sqrt{\log p}$, the average user cost is dominated by $4\mathbf{e}$, as $\sum_{\ell \in V} |\mathcal{N}_\ell| = 2|V| - 2$ and $1\mathbf{e}$ needs $\log p \mathbf{m}$.

For a large $|\mathcal{N}_\ell|$, we can use $\prod_{j=1, j\neq i}^n T_j = (T_1 \cdots T_n) \cdot T_i^{-1}$ to save the cost. In this case, the total cost for user ℓ will be $(|\mathcal{N}_\ell| + 2)\mathbf{e} + 4|\mathcal{N}_\ell|\mathbf{m} + 2(|\mathcal{N}_\ell| - 1)\mathbf{I}$, where $\mathbf{I}$ is for an inverse operation. As $1\mathbf{I}$ is at most $1\mathbf{e}$ while $1\mathbf{m}$ is much cheaper, his cost is dominated by $3|\mathcal{N}_\ell|\mathbf{e}$. As there are $|V| - 1$ edges in $\mathcal{G}$, a user has an average cost $6\mathbf{e}$.

Communication complexity. For each neighbor, user ℓ sends two elements in $\mathbb{Z}_p^*$ in stage one and one ciphertext of an element in $\mathbb{Z}_p^*$. Notice that there exists a CCA2 secure cipher [33] that has the similar length as its input. So the communication complexity for user ℓ has $3|\mathcal{N}_\ell|$ elements. So each user in average sends 6 elements.

Round complexity. Recall $d(\mathcal{G})$ is the maximum distance between nodes in $\mathcal{G}$. The round complexity for stage one is $d(\mathcal{G})$. The message order in stage two is identical to stage one and hence has $d(\mathcal{G})$ rounds too. Thus, the round complexity is $2d(\mathcal{G})$. For instance, if $\mathcal{G}$ is a complete binary tree of n nodes, then $d(\mathcal{G}) < 2 \log n$ and hence the round complexity is $4 \log n$. In section 5, we will show that the factor $d(\mathcal{G})$ for any (passively) secure group key agreement in our model is inevitable.

4.1.3 Security

Now we prove the passive security with contributiveness of DH-KA. Toward this, define $\mathsf{sid}_i^{\ell_i} = \{(A_j, j) \mid j \in \mathcal{N}_i \cup \{i\}\}$ and let $\mathcal{S}(\mathsf{sid}_i^{\ell_i}, \mathsf{sid}_j^{\ell_j}) = 1$ if and only if $\{(A_j, j), (A_i, i)\} \subseteq \mathsf{sid}_i^{\ell_i} \cap \mathsf{sid}_j^{\ell_j}$ for some A_j, A_i. So ignoring the probability that user i samples the same A_i twice, $\Pi_i^{\ell_i}$ and $\Pi_j^{\ell_j}$ are partnered in the passive model if and only if they belong to the same **Execute** query.

The idea of the passive security is simple. Since there is no key setup for each user, the protocol executions are completely independent. So **Corrupt** query and **Execute** query are useless. By Lemma 1, given $A_i, i = 1, \cdots, n$, $\{g^{a_i a_j}\}_{1 \leq i < j \leq n}$ are pseudorandom. Notice that $L_{\ell i}$ is the product of $g^{a_\ell a_i}$ for all $(\ell, i) \notin E$ and $\ell \neq i$. So under the protection of $\{g^{a_\ell a_i}\}_{(\ell,i)\in E}$, all $L_{\ell i}$ are confidential by the security of the encryption scheme. So the view of the passive adversary is essentially just $A_1, \cdots, A_n$. Since sk is the product of all $g^{a_\ell a_i}$ for $(\ell, i) \notin E$, the adversary view is computationally independent of sk. The passive security follows. The formal proof is as follows.

Theorem 1: Let $(\mathcal{E}, \mathcal{D})$ be IND-CPA secure. Then, under the DDH assumption, DH-KA is passively secure with contributiveness.

Proof. By Claim 2, each user ℓ derives the same $sk = \prod_{(i,j)\notin E, i\neq j} g^{a_i a_j}$. The completeness follows. For any i, as

a_i is uniformly random over $\mathbb{Z}_q$, sk is uniformly random in $\langle g\rangle$, even when $\{a_j\}_{j\neq i}$ and public information are fixed. The contributiveness follows. It is left to prove the passive secrecy. Notice that users have no long term secrets. So **Corrupt** oracle can be removed. In addition, randomness for different **Execute** oracle queries are completely independent. Notice that instances in one **Execute** query are all partnered by our definition of $\mathsf{sid}_i^{\ell_i}$. Hence, **Reveal** query can not be issued to any instance in the **Execute** query where the test session $\Pi_s^{\ell_s^*}$ is generated. Thus, **Reveal** query can be removed and also all **Execute** queries other than the one corresponding to the test session can be removed. Let $\mathcal{A}$ be an attacker against the secrecy of sk of the test session. Then, the view of $\mathcal{A}$ after the **Execute** query is

$$\mathsf{view} = \langle p, q, g, (V, E), \{A_i\}_{i\in V}, \{C_{\ell i}\}_{(\ell i)\in E}\rangle. \tag{5}$$

Denote the game where view is real by Γ_0. We modify Γ_0 to Γ_1, Γ_2 and build relationships between them.

Game Γ_1. We modify Γ_0 to Γ_1 such that $A_j^{a_i}$ for any i, j is replaced by $g^{a_{ij}}$ for $a_{ij} \leftarrow \mathbb{Z}_q$, due to which all $\rho_{\ell t}$ and $L_{\ell t}$ are revised accordingly. By Lemma 1, we know that $\mathsf{view}(\Gamma_0)$ and $\mathsf{view}(\Gamma_1)$ are indistinguishable.

Game Γ_2. We modify Γ_1 to Γ_2 such that $C_{\ell i} = \mathcal{E}_{\rho_{\ell i}}(\mathbf{1})$ (instead of $C_{\ell i} = \mathcal{E}_{\rho_{\ell i}}(L_{\ell i})$), where $\mathbf{1}$ is the identity of $\mathbb{Z}_p^*$. By Claim 2, all $L_{\ell i}$ is deterministic in $\{a_{uv} \mid (u, v) \notin E\}$ and hence is independent of $\{a_{uv} \mid (u, v) \in E\}$ which deterministically decides all $\rho_{\ell i}$. Hence, by a simple hybrid reduction to the IND-CPA security of $(\mathcal{E}, \mathcal{D})$, $\mathsf{view}(\Gamma_1)$ and $\mathsf{view}(\Gamma_2)$ are indistinguishable.

Now we analyze game Γ_2. Notice that in Γ_2, $\mathsf{view}(\Gamma_2)$ is completely independent of $\{a_{uv} \mid (u, v) \notin E\}$. On the other hand, by Claim 2, we know that sk is deterministic in $\{a_{uv} \mid (u, v) \notin E\}$ and hence is independent of $\mathsf{view}(\Gamma_2)$. Thus, $\mathcal{A}$ can guess b with probability exactly $1/2$. Since $\mathsf{view}(\Gamma_i)$ for $i = 0, 1, 2$ are indistinguishable, $\mathcal{A}$ has a negligible success advantage in Γ_0. ■

4.2 When $\mathcal{G}$ is a tree: the second scheme

Let p, q be large primes with $q|p-1$ and g be a generator of the group $\mathbb{G}$ of order q in $\mathbb{Z}_p^*$. Let $(\mathcal{E}_\rho, \mathcal{D}_\rho)$ be a symmetric encryption scheme with a secret key ρ. Assume p, q, g are known to all users. Let $\mathcal{G} = (V, E)$ be a tree as a subgraph of $\mathcal{G}_\mathcal{U}$. A key agreement over $\mathcal{G}$ is formally described in Fig. 5. To better understand it, we provide more explanations here.

In Stage one, user i will send a temporary DH public key to each of his neighbor j so that they can share a pairwise DH key, and he will also generate a secret as his share for the final group key. Specifically, user i takes a secret $a_i \leftarrow \mathbb{Z}_q$ and sends $A_i = g^{a_i}$ to his neighbors. Later, the communication over edge (ℓ, i) will be protected by the DH key $g^{a_\ell a_i}$. In the protocol, a_i will only be used to generate a DH key. In addition, user i will also generate $c_i \leftarrow \{0, 1\}^k$, which will play as the additive contribution of user i to the group key $sk = \bigoplus_{j\in V} c_j$.

In Stage two, user ℓ sends to each of his neighbor i a partial group secret that is derived from his own share (generated in stage one) and the partial group secrets received from other neighbors, where the partial group secret is encrypted by the pairwise DH key between i and ℓ. Specifically, he defines $c_{\ell i} = c_\ell \bigoplus(\bigoplus_{j\in\mathcal{N}_\ell\backslash\{i\}} c_{j\ell})$ and sends it in an encrypted form (with key $g^{a_\ell a_i}$) to his neighbor i. For instance, $c_{7,6} = c_7 \bigoplus c_{4,7} \bigoplus c_{8,7} \bigoplus c_{5,7}$ in the graph of Fig. 3 (where L should be replaced by c for the message in the graph). Of course, he can do this only after receiving $c_{j\ell}$ from each $j \in \mathcal{N}_\ell\backslash\{i\}$. This can be satisfied iteratively, starting from leaf users who have only one neighbor. We will show in the completeness that the specific meaning of $c_{\ell i}$ under the above definition is the sum of c_j for all j in the subtree ℓ (including ℓ) of node i. For instance, $c_{\ell i} = c_\ell$ for any leaf ℓ. In Fig. 3, $c_{4,7} = c_1 \bigoplus c_2 \bigoplus c_3 \bigoplus c_4$ and $c_{3,2} = c_3$.

In Stage three, each user derives the group key using his own share and the partial group secrets received from all neighbors. Specifically, ℓ computes $sk = c_\ell \bigoplus(\bigoplus_{j\in\mathcal{N}_\ell} c_{j\ell})$. From the specific meaning of $c_{\ell i}$ stated in Stage two, we can see that $sk = \bigoplus_{j\in V} c_j$.

Stage one.

- Each user $i \in V$ takes $a_i \leftarrow \mathbb{Z}_q, c_i \leftarrow \{0, 1\}^k$ and defines $A_i = g^{a_i}$. Then, user i sends A_i to his neighbors $\mathcal{N}_i$ and receives A_j from each $j \in \mathcal{N}_i$. Let $\rho_{ij} = g^{a_i a_j}$.

Stage two.

1. Each leaf user s (with $\mathcal{N}_s = \{i\}$) computes $c_{si} = c_i$ and sends $C_{si} = \mathcal{E}_{\rho_{si}}(c_{si})$ to i.
2. **[Loop]** Each $\ell \in V$ does the following. For $i \in \mathcal{N}_\ell$, if user ℓ has received $C_{j\ell}$ from all $j \in \mathcal{N}_\ell\backslash\{i\}$ and did not send $C_{\ell i}$ to i, then he decrypts $c_{j\ell} = \mathcal{D}_{\rho_{j\ell}}(C_{j\ell})$, computes $c_{\ell i} = c_\ell \bigoplus(\bigoplus_{j\in\mathcal{N}_\ell\backslash\{i\}} c_{j\ell})$ and sends $C_{\ell i} = \mathcal{E}_{\rho_{\ell i}}(c_{\ell i})$ to user i.
3. Each user ℓ continues step 2 until he has sent $C_{\ell i}$ to each $i \in \mathcal{N}_\ell$, in which case, he proceeds to Stage three.

Stage three (*group key derivation*).

- Upon $C_{j\ell}$ for all $j \in \mathcal{N}_\ell$, user ℓ decrypts $c_{j\ell} = \mathcal{D}_{\rho_{j\ell}}(C_{j\ell})$ with $\rho_{j\ell}$ (if not done before) and calculates group key $sk = c_\ell \bigoplus(\bigoplus_{j\in\mathcal{N}_\ell} c_{j\ell}) = \bigoplus_{j\in V} c_j$.

Fig. 5. Our Protocol XO-KA

Completeness. For each $(\ell, i) \in E$, let $\mathcal{S}_{\ell i}^*$ be the set of users in the subtree ℓ (including ℓ) of node i. Thus, $\mathcal{S}_{\ell i}^* = \mathcal{S}_{\ell i} \cup \{\ell\}$. For instance, $\mathcal{S}_{1,2} = \{1\}$, $\mathcal{S}_{4,7} = \{1, 2, 3, 4\}$ in Fig. 3. Completeness follows from the claim below.

Claim 3. $c_{\ell i} = \bigoplus_{j\in\mathcal{S}_{\ell i}^*} c_j$ and $c_\ell \bigoplus(\bigoplus_{j\in\mathcal{N}_\ell} c_{j\ell}) = \bigoplus_{j\in V} c_j$.

Proof. Consider part one first. Use induction on the

depth of ℓ. The claim is obviously true for a leaf node. If it is valid for all children of ℓ (with parent i), then $c_{\ell i} = c_\ell \bigoplus(\bigoplus_{j\in\mathcal{N}_\ell\setminus\{i\}} c_{j\ell}) = c_\ell \bigoplus(\bigoplus_{j\in\mathcal{N}_\ell\setminus\{i\}}(\bigoplus_{s\in\mathcal{S}^*_{j\ell}} c_s))$. Since $\mathcal{G}$ is a tree, sets $\mathcal{S}^*_{j\ell}$ with j over $\mathcal{N}_\ell$ are disjoints. Hence, $c_{\ell i} = c_\ell \bigoplus(\bigoplus_{s\in\cup_{j\in\mathcal{N}_\ell\setminus\{i\}}\mathcal{S}^*_{j\ell}} c_s)$. Notice that $\mathcal{S}^*_{\ell i} = \{\ell\}\cup(\cup_{j\in\mathcal{N}_\ell\setminus\{i\}}\mathcal{S}^*_{j\ell})$. It follows that $c_{\ell i} = \bigoplus_{j\in\mathcal{S}^*_{\ell i}} c_j$. This finishes the part one. Part two is immediate as $\mathcal{S}^*_{j\ell}$ with j over $\mathcal{N}_\ell$ are disjoint and $\cup_{j\in\mathcal{N}_\ell}\mathcal{S}^*_{j\ell} = V\setminus\{\ell\}$. □

Efficiency. *Computation cost.* The cost for user ℓ other than $A_\ell, \rho_{\ell i}$ is negligible and ignored here. Hence, the computation cost of user ℓ is dominated by $(1+|\mathcal{N}_\ell|)$e. As there are $|V|-1$ edges in $\mathcal{G}$, each user in average has a cost of 3e.

Communication complexity. The outgoing messages of ℓ are $A_\ell, C_{\ell i}$ for each $i\in\mathcal{N}_\ell$. There are $2|\mathcal{N}_\ell|$ elements in $\mathbb{Z}^*_p$ for him. So a user in average sends 4 elements.

Round complexity. The round complexity is one in stage one and $d(\mathcal{G})$ in stage two, giving the total of $d(\mathcal{G})+1$. Later we will see that from the lower bound in Theorem 4, XO-KA is nearly round optimal.

Remark. In comparison, we can see that XO-KA is more efficient than DH-KA in all three measures (computation, communication and round complexity). However, DH-KA can be regarded as a generalization of the well-known Diffie-Hellman protocol to the setting where a user does not know anything (such as the total number of users, the network topology) beyond his neighbors. As most existing group key agreements in the literature are variants of a certain generalized Diffie-Hellman with a special connectivity graph, it would be interesting to add DH-KA as a new method into this Diffie-Hellman family with a feature of an arbitrary connectivity graph.

Theorem 2: If $(\mathcal{E},\mathcal{D})$ is IND-CPA secure, then under the DDH assumption, XO-KA is passively secure with contributiveness.

Proof. By Claim 3, each user ℓ derives the same $sk = \bigoplus_{j\in V} c_j$. The completeness follows. For any i, as c_i is uniformly random over $\{0,1\}^k$, sk is uniformly random when $\{c_j\}_{j\neq i}$ are fixed. The contributiveness follows. It is left to prove the passive security. Similar to the proof of Theorem 1, any query other than the **Test** query and its corresponding **Execute** query can be removed. Let $\mathcal{A}$ be an attacker against the secrecy of sk. Then, the adversary view after the **Execute** query is

$$\mathsf{view} = \langle p, q, g, (V,E), \{A_i\}_{i\in V}, \{C_{\ell i}\}_{(\ell,i)\in E}\rangle. \quad (6)$$

Denote the game where view is real by Γ_0. We modify Γ_0 to Γ_1, Γ_2 and build relationships between them.

Game Γ_1. We modify Γ_0 to Γ_1 such that $\rho_{ji} = g^{a_j a_i}$ for $(i,j)\in E$ is replaced by $g^{a_{ij}}$ for $a_{ij}\leftarrow\mathbb{Z}_q$. By Lemma 1, $\mathsf{view}(\Gamma_0)$ and $\mathsf{view}(\Gamma_1)$ are indistinguishable.

Game Γ_2. We modify Γ_1 to Γ_2 such that $C_{\ell i} = \mathcal{E}_{\rho_{\ell i}}(\mathbf{0})$ (instead of $C_{\ell i} = \mathcal{E}_{\rho_{\ell i}}(c_{\ell i})$), where $\mathbf{0}$ has the length of k. By a simple hybrid reduction to the IND-CPA security of $(\mathcal{E},\mathcal{D})$, $\mathsf{view}(\Gamma_1)$ and $\mathsf{view}(\Gamma_2)$ are indistinguishable.

Now we analyze game Γ_2. Notice that in Γ_2, $\mathsf{view}(\Gamma_2)$ is completely independent of $(c_1,\cdots,c_n)$. By Claim 3, sk is independent of $\mathsf{view}(\Gamma_2)$. Thus, the success probability of $\mathcal{A}$ in guessing b is exactly 1/2. Since $\mathsf{view}(\Gamma_i)$ for $i=0,1,2$ are all indistinguishable, it follows that the adversary has only a negligible advantage of success. ■

4.3 The general case

In the previous subsections, we assume that $\mathcal{G}$ is a tree. In the real applications, this is unlikely. To overcome this, we design a protocol to help users determine a spanning tree $\mathcal{G}^*$ of $\mathcal{G}$ before executing the actual key agreement. We remark that $\mathcal{G}^*$ either is determined upon each key agreement event or remains unchanged for a long time (if $\mathcal{G}$ does). This can be decided up to the real applications. The protocol is described in Fig. 6.

The idea is as follows. The protocol starts with an initiator s. He sends a "hello" message to his neighbors. Any user ℓ, who receives the first "hello", will reply a message "new" to its sender and then send a "hello" message to his other neighbors. The protocol continues until all users have done this. In the protocol, each edge (i,j) that carries message "new" will be added into graph $\mathcal{G}^*$. To do this, i,j just add each other as neighbors in $\mathcal{G}^*$ and the list of neighbor sets of all users in $\mathcal{G}$ uniquely defines graph $\mathcal{G}^*$. We remark that $\mathcal{G}^*$ is connected since each node is connected with s through a path of a "new" message. Further, $\mathcal{G}^*$ is acyclic as including each edge e of a "new" message into $\mathcal{G}^*$ implies that the sender of this message was not contacted before and hence was not added as neighbors in $\mathcal{G}^*$ by any user. Formally, we have the following theorem.

Let s be an initiator. Each ℓ in the protocol obtains $\mathcal{N}_\ell(\mathcal{G}^*) \stackrel{def}{=} \mathcal{N}^*_\ell$ that defines $\mathcal{G}^*$.

1. User s sends "hello" to his neighbors $\mathcal{N}_\ell(\mathcal{G})$ and proceeds to Step 3.
2. Upon "hello", user ℓ checks if he has received it before. If yes, he replies "old" to the sender (say, user i); otherwise, he does the following.
   - He replies "new" to sender i and sets $\mathcal{N}^*_\ell = \{i\}$. He then sends "hello" to $\mathcal{N}_\ell(\mathcal{G})\setminus\{i\}$ and proceeds to Step 3.
3. **[Loop]** Upon a reply "new" from user u, user ℓ updates $\mathcal{N}^*_\ell = \mathcal{N}^*_\ell\cup\{u\}$; upon a reply "old", user ℓ does nothing.
4. Each user ℓ continues Step 3 until he has received all responses to his "hello" message, in which case, he outputs $\mathcal{N}^*_\ell$.

Fig. 6. Protocol LocSpan to compute a spanning tree $\mathcal{G}^*$

Theorem 3: If $i\in\mathcal{N}^*_\ell$, then $\ell\in\mathcal{N}^*_i$. In addition, $\mathcal{G}^*$ is a spanning tree of a connected graph $\mathcal{G}$.

Proof. By the protocol description, $\mathcal{N}^*_\ell$ is the subset of $\mathcal{N}_\ell$ who sends "new" or the first "hello" to user ℓ. If

user $i \in \mathcal{N}_\ell^*$ sends the first "hello" to user ℓ, then user ℓ will reply "new" to user i and hence $\ell \in \mathcal{N}_i^*$. If user $i \in \mathcal{N}_\ell^*$ sends "new" to user ℓ, then the first "hello" that user i has received must come from user ℓ and hence $\ell \in \mathcal{N}_i^*$. This completes the first statement. We now consider the second one. Obviously, $\mathcal{G}^*$ is connected. Indeed, as user ℓ is always first activated by a "hello" message (which can be traced back to user s), user ℓ must be connected with user s. As $\mathcal{G}$ is connected, any user can be reached by a "hello" message. Hence, $\mathcal{G}^*$ is a connected subgraph including all users V. If $\mathcal{G}^*$ is not a tree, there must exist a cycle $\ell_1 \cdots \ell_t \ell_1$ in it as $\mathcal{G}^*$ is connected. Notice two facts: (i) for any edge (a, b), either a sends "hello" to b and b replies "new", or vice versa; (ii) any node v in $\mathcal{G}^*$ will send a "new" message at most once. W.O.L.G., assume that in edge (ℓ_1, ℓ_2), ℓ_2 sends "hello" and ℓ_1 replies "new". Then, by fact (ii), ℓ_1 must send "hello" to ℓ_t who then replies with "new". Continue this argument on $(\ell_t, \ell_{t-1}), (\ell_{t-1}, \ell_{t-2}), \cdots, (\ell_3, \ell_2)$. We conclude that ℓ_3 sends "hello" to ℓ_2 and ℓ_2 answers "new" to him. However, a "hello" message is always earlier than its corresponding reply of "new". Hence, the "hello" message of ℓ_2 to ℓ_1 is earlier than "hello" of ℓ_1 to ℓ_t, which is further earlier than that of ℓ_t to ℓ_{t-1}. Continuing this, we conclude that the "hello" message of ℓ_2 is earlier than itself, contradiction! Hence, $\mathcal{G}^*$ has no cycle and is a tree. As it contains all nodes V, it is a spanning tree of $\mathcal{G}$. ■

Efficiency. If $\mathcal{G}$ is not a tree, then our previous constructions will be run over $\mathcal{G}^*$ computed by LocSpan. Hence, the efficiency cost should include the cost in LocSpan. In LocSpan, the computation cost of user ℓ is constant and his message length is also constant. Compared with DH-KA and XO-KA, they can be ignored. In addition, it is easy to verify that LocSpan has a round complexity of $d(\mathcal{G}^*)$. Thus, if DH-KA* and XO-KA* respectively denote the DH-KA and XO-KA with a preprocessing protocol LocSpan, then their computation costs and communication complexities essentially remain unchanged while DH-KA* has a round complexity of $3d(\mathcal{G}^*)$ and XO-KA* has a round complexity of $2d(\mathcal{G}^*) + 1$.

We stress that this efficiency evaluation assumes every key agreement execution will always run LocSpan first. As said before, if graph $\mathcal{G}$ remains unchanged for a long time, then the round complexity due to the preprocessing can be ignored as well.

Security of DH-KA* and XO-KA*. Notice LocSpan does not include any secret computation or secret key. Hence with it as a preprocessing, the passive security of protocol will remain true. Hence, from Theorem 1 and Theorem 2, we conclude the following corollaries.

Corollary 1: If $(\mathcal{E}, \mathcal{D})$ is IND-CPA secure, then under the DDH assumption, DH-KA* is passively secure with contributiveness.

Corollary 2: If $(\mathcal{E}, \mathcal{D})$ is IND-CPA secure, then under the DDH assumption, XO-KA* is passively secure with contributiveness.

5 LOWER BOUND

In this section, we will derive lower bounds on the round complexity of a group key agreement. The lower bounds hold even when a *starting assumption* is made: before the protocol starts, each user is aware of the request to execute a key agreement. As before, assume that $d(\mathcal{G})$ is the maximum distance between nodes in a connected graph $\mathcal{G}$, where the distance between two nodes is the number of edges in a shortest path connecting them. We show that if users' private keys (if any) are not correlated, then the round complexity of a secure group key agreement is lower bounded by $d(\mathcal{G})/2$ and the round complexity of a contributively secure key agreement is lower bounded by $d(\mathcal{G})$. Toward this, we introduce an auxiliary protocol.

This protocol is to compute a set $\mathcal{I}_i^t$ for each $i \in V$ and integer $t \geq 0$. The specific meaning of $\mathcal{I}_i^t$ is the set of users that can be reached by i through a path of length at most t. Initially, of course $\mathcal{I}_i^0 = \{i\}$. At each step t, each user i sends $\mathcal{I}_i^{t-1}$ to his neighbors. In turn, he will receive his neighbors' sets $\mathcal{I}_j^{t-1}, j \in \mathcal{N}_i$. Upon this, he updates $\mathcal{I}_i^{t-1}$ to $\mathcal{I}_i^t = \mathcal{I}_i^{t-1} \cup (\cup_{j \in \mathcal{N}_i} \mathcal{I}_j^{t-1})$. Since $(\cup_{j \in \mathcal{N}_i} \mathcal{I}_j^{t-1})$ contains all nodes that are accessible from one of his neighbors through a path of length at most $t-1$, $\mathcal{I}_i^t$ is really the set of all nodes reachable from i through a path of length at most t.

Let d be any positive integer.

0. User $i \in V$ initializes $\mathcal{I}_i^0 = \{i\}$. Let $t = 1$.
1. **[Loop]** At stage t, each user i sends $\mathcal{I}_i^{t-1}$ to his neighbors $\mathcal{N}_i$. Upon $\mathcal{I}_j^{t-1}$ for all $j \in \mathcal{N}_i$, user i computes $\mathcal{I}_i^t = \mathcal{I}_i^{t-1} \cup (\cup_{j \in \mathcal{N}_i} \mathcal{I}_j^{t-1})$.
2. Continue step 1 for $t = 1, \cdots, d$ and end.

Fig. 7. Protocol IndCom for a connected graph $\mathcal{G}$

In the following, we prove the specific meaning in our idea description is true.

Lemma 2: Let $d(i, j)$ be the distance between i and j in a connected graph $\mathcal{G}$. Then, $\mathcal{I}_i^t = \{j \mid d(i, j) \leq t\}$ in protocol IndCom, for any $t \leq d$ and any $i \in V$.

Proof. First of all, if $d(i, j) = t$ (witnessed by a path $i_0 (= j) i_1 \cdots i_t (= i)$), then $j \in \mathcal{I}_{i_\ell}^\ell$ for any $\ell = 0, \cdots, t$ (as j is iteratively transmitted on this path through stage $0, 1, \cdots, \ell$, starting from j and moving one step in each stage). Thus, $j \in \mathcal{I}_i^t$. Hence, $\{j \mid d(i, j) \leq t\} \subseteq \mathcal{I}_i^t$. We show the other direction by induction on t. It holds for $t = 0$ obviously. Assume it holds for $t - 1$. Consider the case t. As $\mathcal{I}_i^t = \mathcal{I}_i^{t-1} \cup (\cup_{j \in \mathcal{N}_i} \mathcal{I}_j^{t-1})$, by induction, we have $\mathcal{I}_i^t \subseteq \{s \mid d(i, s) \leq t-1\} \cup (\cup_{j \in \mathcal{N}_i} \{s \mid d(s, j) \leq t-1\})$. Notice $d(s, j) \leq t-1$ for $j \in \mathcal{N}_i$ implies that $d(s, i) \leq t$. Thus, $\mathcal{I}_i^t \subseteq \{s \mid d(s, i) \leq t\}$. The lemma follows. □

For a given protocol, we use view_i^t to denote the view of user i till the end of round t which consists of global public information, the random tape of user i and the messages received so far by user i.

Lemma 3: Let Π be a d-round key agreement for a connected graph $\mathcal{G}_\mathcal{U}$ with system parameter sp. Let user

i $(1 \le i \le N)$ have a public key PK_i, secret key SK_i and random tape R_i. Then, view_i^0 is determined by $\{\mathsf{sp}, SK_i, R_i, PK_1, \cdots, PK_N, \mathcal{G}\}$ and view_i^t is determined by $\{\mathsf{view}_j^0 \mid j \in \mathcal{I}_i^t\}$ for any $t \le d$.

Proof. Let Π be run on a connected graph $\mathcal{G} = (V, E)$. Use induction on t. The conclusion obviously holds for $t = 0$. Assume it holds for round $t-1$. Consider round t. In fact, view_i^t consists of view_i^{t-1} and messages m_{ji}^t from user $j \in \mathcal{N}_i$ in the tth round. As m_{ji}^t is determined by view_j^{t-1}, view_i^t is determined by $\{\mathsf{view}_i^{t-1}\} \cup \{\mathsf{view}_j^{t-1} \mid j \in \mathcal{N}_i\}$. By induction, view_i^t is determined by

$$\begin{aligned}&\{\mathsf{view}_s^0 \mid s \in \mathcal{I}_i^{t-1}\} \cup (\cup_{j\in\mathcal{N}_i}\{\mathsf{view}_s^0 \mid s \in \mathcal{I}_j^{t-1}\})\\ =&\{\mathsf{view}_s^0 \mid s \in \mathcal{I}_i^{t-1} \cup (\cup_{j\in\mathcal{N}_i}\mathcal{I}_j^{t-1})\}\\ =&\{\mathsf{view}_s^0 \mid s \in \mathcal{I}_i^t\},\end{aligned}$$

where the last "=" follows from the definition of $\mathcal{I}_i^t$. □

Now we prove the lower bounds of the round complexity d. The idea is that any message and also the final session key sk_i generated by user i is deterministic in his secret key SK_i, random tape R_i and his list tr_i of all incoming messages. Also a message only travels one edge in one round. So if (SK_j, R_j) has contributed to compute sk_i, then j must be within distance d from user i. So for u, v with $d(u, v) = d(\mathcal{G}) > 2d$, the set of users $\mathcal{U}_u$ that has contributed to sk_u must be disjoint with the set of users $\mathcal{U}_v$ that has contributed to sk_v. Consequently, if all $SK_1, \cdots, SK_n$ are independent, then $\{(SK_j, R_j)\}_{j\in\mathcal{U}_u}$ and $\{(SK_j, R_j)\}_{j\in\mathcal{U}_v}$ are independent. Hence, sk_u and sk_v are independent too, contradicting to the completeness $sk_u = sk_v$. This concludes $d \ge \frac{d(\mathcal{G})}{2}$. If Π is contributive, then any user u has contributed to sk_v for any other v. As argued above, this takes at least $d(u, v)$ rounds. This especially holds for u, v with $d(u, v) = d(\mathcal{G})$ and hence the round complexity $d \ge d(\mathcal{G})$.

It should be pointed out that although the intuition of lower bounds is very clear, we do not found a statement for this in the literature. Besides that the proof details are not completely trivial, these lower bounds are also important to show the round efficiency of our protocols. Thus, we now formally state and prove them.

Theorem 4: Let Π be a d-round key agreement on a connected subgraph $\mathcal{G} = (V, E)$ of $\mathcal{G}_\mathcal{U}$. For any $i \in V$, assume $I(SK_i; \{SK_j\}_{j\in V\setminus\{i\}} | \mathsf{sp}, \mathcal{G}, PK_1, \cdots, PK_N) = 0$. Then, $d \ge \frac{d(\mathcal{G})}{2}$ if Π is passively secure; $d \ge d(\mathcal{G})$ if Π is passively secure with contributiveness.

Proof. Assume u, v achieves $d(u, v) = d(\mathcal{G})$ in $\mathcal{G}$. Let sk_i be the group key computed by user i in an execution of Π, where user i has a random tape R_i. Let $\pi = \{\mathsf{sp}, PK_1, \cdots, PK_N, \mathcal{G}\}$. As sk_i is deterministic in view_i^d for any $i \in V$, we have

$$I(sk_u; sk_v|\pi) \le I(\mathsf{view}_u^d; \mathsf{view}_v^d|\pi). \tag{7}$$

By Lemma 3, view_i^d is determined by $\{\mathsf{view}_s^0 \mid s \in \mathcal{I}_i^d\}$. Hence,

$$\begin{aligned}&I(sk_u; sk_v|\pi) \qquad (8)\\ \le&I\Big(\{\mathsf{view}_s^0 \mid s \in \mathcal{I}_u^d\}; \{\mathsf{view}_s^0 \mid s \in \mathcal{I}_v^d\}\Big|\pi\Big)\\ =&I\Big(\{(SK_s, R_s) \mid s \in \mathcal{I}_u^d\}; \{(SK_s, R_s) \mid s \in \mathcal{I}_v^d\}\Big|\pi\Big).\end{aligned}$$

proof of part one Now if $d < d(\mathcal{G})/2$, then by Lemma 2, $\mathcal{I}_v^d \cap \mathcal{I}_u^d = \emptyset$. As R_i is independent of $(\{SK_j\}_{j\in V}, \{R\}_{j\ne i}, \pi)$, Eq. (8) reduces to

$$\begin{aligned}&I(sk_u; sk_v|\pi)\\ \le&I\Big(\{SK_s \mid s \in \mathcal{I}_u^d\}; \{SK_s \mid s \in \mathcal{I}_v^d\}\Big|\pi\Big)\\ \le&\sum_{s\in\mathcal{I}_u^d} I\Big(SK_s; \{SK_j \mid j \ne s\}\Big|\pi\Big) \qquad (9)\\ =&0, \qquad (10)\end{aligned}$$

where Eq. (9) uses the fact:

$$\begin{aligned}&I(A_1, \cdots A_n; A_{n+1}, \cdots, A_m|B)\\ =&\sum_{i=1}^n I(A_i; \{A_j\}_{j=n+1}^m | A_1 \cdots A_{i-1} B)\\ \le&\sum_{i=1}^n I(A_i; \{A_j\}_{j=1, j\ne i}^m | B).\end{aligned}$$

Here both (in)equalities follow from the chain rule of the mutual information.

By completeness of Π, $sk_u = sk_v$. Hence, $H(sk_u|\pi) = I(sk_u; sk_v|\pi) = 0$ by Eq. (10). That is, sk_u is deterministic in π and remains unchanged for different executions. Hence, an attacker can obtain it through an **Execute** query followed by a **Reveal** query. Thus, Π is impossible to be passively secure, contradiction!

proof of part two Let $W_u = \{\pi\} \cup \{\mathsf{view}_j^0 | j \ne u\}$. As sk_v is deterministic in view_v^d,

$$I(\mathsf{view}_u^0; sk_v|W_u) \le I(\mathsf{view}_u^0; \mathsf{view}_v^d|W_u).$$

If $d < d(\mathcal{G}) = d(u, v)$, then $u \notin \mathcal{I}_v^d$ by Lemma 2. Hence,

$$\begin{aligned}&I(\mathsf{view}_u^0; sk_v|W_u)\\ \le&I(\mathsf{view}_u^0; \{\mathsf{view}_j^0 \mid d(j, v) \le d\}|W_u)\\ \le&I(\mathsf{view}_u^0; \{\mathsf{view}_j^0 \mid j \ne u\}|W_u)\\ =&0.\end{aligned}$$

Hence, $H(sk_v|W_u) = H(sk_v|\mathsf{view}_u^0, W_u)$. Further, sk_v is determined by $\{\mathsf{view}_i^0 \mid i \in V\}$. Thus, $H(sk_v|W_u) = H(sk_v|\mathsf{view}_u^0, W_u) = 0$. On the other hand, $H(sk_v|\pi) > 0$ (otherwise, sk_v is constant and can be obtained by an **Execute** query followed by a Reveal query). Hence, $I(sk; \{SK_i, R_i\}_{i\ne u}|\pi) = H(sk_v|\pi) - H(sk_v|W_u) > 0$, where we use the completeness $sk_v = sk$. This contradicts the contributiveness. ■

6 AN ACTIVELY SECURE CONSTRUCTION

We present a construction of an actively secure protocol Π' from a passively secure one Π. Our construction consists of two stages. Stage 0 is to set up the session information and satisfy the starting assumption. Stage 1 is the actual transformation of Π that essentially authenticates each message in Π using a signature. The formal description is in Fig. 8. To better understand the protocol, we provide some explanations as follows.

In Stage 0, besides satisfying the starting assumption, we will establish a global session identifier and the session identifier between any two neighboring users. This is necessary as a user can only access his neighbors. Toward this, an initiator I first takes a random $\theta_I \leftarrow \{0,1\}^\kappa$ and then sends $\theta_I|\theta_I|I$ to his neighbors. His neighbor i will also take $\theta_i \leftarrow \{0,1\}^\kappa$ and send $\theta_I|\theta_i|i$ to his own neighbors. Generally, when a user j is first contacted, he will take $\theta_j \leftarrow \{0,1\}^\kappa$ and send $\theta_I|\theta_j|j$ to his own neighbors. Here θ_I essentially plays as a global session identifier. The session between two neighbors i, j can be identified using $\theta_I|\theta_i|\theta_j$.

In Stage 1, the purpose is to execute the protocol Π authentically. Specifically, if user i wishes to send m to his neighbor j, he sends $\theta_I|m|\mathsf{sig}_{s_i}(\theta_I|i|j|\theta_i|\theta_j|m)$. Here θ_I allows j to find the session to process the message and $\mathsf{sig}_{s_i}(\theta_I|i|j|\theta_i|\theta_j|m)$ allows user j to confirm that m is authenticated: if i is corrupted, no security is possible; if i is uncorrupted, the freshness of $j|\theta_j$ (as user j chooses θ_j randomly) implies that the signature is fresh.

In the remaining of this section, we will prove the security of Π'. Toward this, we need to formally define the session identifier. We define $\mathsf{sid}_i^{\ell_i} = \{\theta_I\} \cup \{(\theta_j, j) \mid j \in \mathcal{N}_i \cup \{i\}\}$. From our protocol description, θ_I is well-defined for $\Pi_i^{\ell_i}$ upon the first activation. In Π', any message M to user i will start with θ_I and will be directed to $\Pi_i^{\ell_i}$ with $\theta_I \in \mathsf{sid}_i^{\ell_i}$ (only one $\Pi_i^{\ell_i}$ in user i with this property exists). This is important as we must coordinate different neighbor instances with $\Pi_i^{\ell_i}$. Finally, we say that $\Pi_i^{\ell_i}$ and $\Pi_j^{\ell_j}$ are directly partnered if $\{\theta_I\} \cup \{(\theta_i, i), (\theta_j, j)\} \subseteq \mathsf{sid}_i^{\ell_i} \cap \mathsf{sid}_j^{\ell_j}$. Note that a repeated θ_I will cause a user to normally reject. However, if a normal initiator samples a repeated θ_I, this occurs with probability only $2^{-\kappa}$, which can be ignored; if an attacker reuses θ_I, the reject means that the attack fails.

The security idea of our construction is as follows. Essentially, we want to argue that if Π is passively secure, then Π' is actively secure. First of all, in any execution of Π with all users uncorrupted, we can assume that users see the same θ_I and that any two neighbors i, j see the same $\theta_i|\theta_j$. This is true as each message at Stage 1 in Π' is accompanied with a signature containing input $\theta_I|\theta_i|\theta_j|i|j$. Under this assumption, if there is an adversary $\mathcal{A}'$ breaking Π', we show how to build an adversary $\mathcal{A}$ breaking Π. The strategy of $\mathcal{A}$ is to simulate the execution of Π' and run $\mathcal{A}'$ against it. In turn, $\mathcal{A}$ mimics the action of $\mathcal{A}'$ to attack Π. Specifically, whenever $\mathcal{A}'$ requests a new execution of Π', $\mathcal{A}$ issues an Execute query in Π and obtains a transcript tr. He tries to simulate Π' such that the transcript of Π in stage 1 in Π' is exactly tr. If this is true, the group key in Π' and the group key in Π are identical. So $\mathcal{A}$ can break the privacy of Π if $\mathcal{A}'$ does this for Π'. To embed tr into Π'. The main task for $\mathcal{A}$ is to answer the Send queries from $\mathcal{A}'$ for a Stage-1 message. To do this, each Send oracle generates the output $\theta_I|m|\sigma$ normally except that the Π message m is taken from tr. Upon a query $\mathsf{Send}(j, \ell_j, \theta_I^*|m|\sigma)$ from user i, $\mathcal{A}$ verifies whether (σ_I^*, σ, m) is consistent with his record $(\sigma_I, \theta_i, \theta_j)$ and m in tr. If yes, it is assured that i and j are in the same session and m is not changed. So again, $\mathcal{A}$ simulates the oracle output normally except the Π message is taken from tr. If no, the attack of $\mathcal{A}'$ is detected and so $\mathcal{A}$ can safely reject. As a result, $\mathcal{A}$ can smoothly simulate a Π' execution for $\mathcal{A}'$ and inherit his success. We present this formally in the following.

Theorem 5: Let Π be a passively (contributively) secure group key agreement with $(PK_i, SK_i) = nil$. Assume that $(\mathsf{sig}, \mathsf{ver})$ is existentially unforgeable. Then, Π' is an actively (contributively) secure group key agreement.

Proof. We prove that if there exists adversary $\mathcal{A}'$ that breaks the active security of Π', then we can construct adversary $\mathcal{A}$ that breaks the passive security of Π. Upon parameter sp and the description of $\mathcal{G}_\mathcal{U}$, $\mathcal{A}$ does as follows. He takes (v_i, s_i) normally for each $i \in \mathcal{U}$. Then, he provides $\mathsf{sp}, \mathcal{G}_\mathcal{U}$ and $(v_1, \cdots, v_N)$ to $\mathcal{A}'$ and simulates the execution of Π' with $\mathcal{A}'$ as follows. First of all, we assume $\mathcal{A}'$ never makes an Execute query as it can be replaced a sequence of Send queries. Let the number of initiating Send queries by $\mathcal{A}'$ be bounded by ν. Then, $\mathcal{A}$ takes $t \leftarrow [\nu]$. Denote the tth initiation Send query by Q.

1. Upon $\mathsf{Send}(i, \ell_i, M)$ query with a stage-0 message M, it processes normally. In addition, if this is query Q on $\Pi_s^{\ell_s^*}$ with graph $\mathcal{G}^*$, $\mathcal{A}$ in Π makes an Execute query on $\mathcal{G}^*$, then $\mathsf{Test}(s, \ell_s^*)$ query. He then receives a transcript tr^* with a test key α_b.
2. Upon $\mathsf{Send}(j, \ell_j, M)$ query with a stage-1 message M from user i (note as stated in the definition of Send oracle, we assume that i and j know the identity of each other at the initialization stage of their communication), $\mathcal{A}$ does as follows. If $\Pi_j^{\ell_j}$ is not equal to or partnered with $\Pi_s^{\ell_s^*}$, he proceeds normally with (v_i, v_j, s_j) and stat_j. Otherwise, he parses $M = \theta_s^*|m|\sigma$ and verifies if σ is valid using (stat_j^*, v_i) and if m is equal to the corresponding message in tr^*. If not, $\Pi_j^{\ell_j}$ rejects; otherwise, he follows user j in tr^* to process normally, except that when he needs to send m' in tr^* to u, he instead sends $\theta_s^*|m'|\mathsf{sig}_{s_j}(\theta_s^*|j|u|\theta_j|\theta_u|m')$.
3. Upon $\mathsf{Corrupt}(i)$ query, $\mathcal{A}$ sends s_i to $\mathcal{A}'$ if $i \notin \mathcal{G}^*$; otherwise, he aborts with Fail.
4. Upon a $\mathsf{Reveal}(\ell_i, i)$ query, if $\Pi_i^{\ell_i}$ is equal to or partnered with $\Pi_{s^*}^{\ell_s^*}$, then $\mathcal{A}$ aborts with Fail; otherwise, he returns sk (if it is defined).
5. Upon $\mathsf{Test}(\ell_i, i)$ query, if $\Pi_i^{\ell_i}$ is equal to or partnered with $\Pi_{s^*}^{\ell_s^*}$, he provides α_b to $\mathcal{A}'$; otherwise, $\mathcal{A}$ aborts

Let (v_i, s_i) be user i's signing/verification key of a signature scheme $(\mathsf{sig}, \mathsf{ver})$. Let user I be the initiator.

Stage 0. i. Initially, user I takes $\theta_I \leftarrow \{0,1\}^\kappa$, sends $\theta_I|\theta_I|I$ to $\mathcal{N}_I$ and sets $\mathsf{stat}_I = \{\theta_I\} \cup \{(\theta_I, I)\}$.

ii. **[Loop]** Upon $\theta_I|\theta_j|j$ from $j \in \mathcal{N}_i$, if the instance in user i is new, then he takes $\theta_i \leftarrow \{0,1\}^\kappa$, sets $\mathsf{stat}_i = \{\theta_I\} \cup \{(\theta_j, j), (\theta_i, i)\}$ and sends $\theta_I|\theta_i|i$ to each $\ell \in \mathcal{N}_i$; otherwise, he sets $\mathsf{stat}_i = \mathsf{stat}_i \cup \{(\theta_j, j)\}$ only if $\theta_I|*|j$ was not received from user j before.

iii. User i remains in Steps (ii) until he received $\theta_I|\theta_j|j$ from all $j \in \mathcal{N}_i$, in which case, he now has $\mathsf{stat}_i = \{\theta_I\} \cup \{(\theta_j, j) \mid j \in \mathcal{N}_i \cup \{i\}\}$ and then moves to Stage 1.

Stage 1. a. Whenever user i (in Π) sends m to $j \in \mathcal{N}_i$, he sends $\theta_I|m|\mathsf{sig}_{s_i}(\theta_I|i|j|\theta_i|\theta_j|m)$ instead.

b. Whenever user j receives $\theta_I|m|\sigma$ from $i \in \mathcal{N}_j$, he rejects if σ is not consistent with $\theta_I|i|j|\theta_i|\theta_j|m$; otherwise, he processes m normally as an incoming message in Π from j, except that an outgoing message m' in Π is prepared according to Step a.

Fig. 8. Actively secure construction Π' from a passively secure protocol Π

with Fail.

Finally, $\mathcal{A}$ outputs whatever $\mathcal{A}'$ does.

This completes the description of $\mathcal{A}$. Now we analyze the success probability of $\mathcal{A}$. We start with a claim.

Claim 4. *$\Pi_i^{\ell_i}$ has at most one partnered instance in user j for a given j.*

Proof. This is obvious as each user i will keep only one instance consistent with θ_I. □

Note if an instance that is equal to or partnered with $\Pi_s^{\ell_s^*}$ is the Test session, then abortion events in item 3 will not occur, as a Test session will not have a corrupted group member. So before an abortion event, the view of $\mathcal{A}'$ differs from the real game only in the following:

- In item 2, after $\mathcal{A}$ confirms that $\Pi_j^{\ell_j}$ is equal to or partnered with $\Pi_s^{\ell_s^*}$ and has verified σ, he will further check if m in M is equal to the corresponding message in tr^* while in the real game this is not needed. Denote the inconsistency of m by Bad_1.

We now prove the following claim.

Claim 5. $\Pr(\mathsf{Bad}_1)$ is negligible.

Proof. Note $M = \theta_s^*|m|\sigma$ from user i supposedly was sent from some $\Pi_i^{\ell_i}$. By the validity of σ and the security of $(\mathsf{sig}, \mathsf{Ver})$, we can assume that user i ever sent $M' = \theta_s^*|m|sig'$ to user j s.t. $\sigma' = \mathsf{sig}_{s_i}(\theta_s^*|i|j|\theta_i^*|\theta_j^*|m)$, where θ_i^*, θ_j^* is taken from $\mathsf{stat}_j^{\ell_j}$. Hence, $\mathsf{stat}_i^{\ell_i}$ must also contain $\theta_s^*, (\theta_j^*, j), (\theta_i^*, i)$. Thus, $\Pi_i^{\ell_i}$ and $\Pi_j^{\ell_j}$ are directly partnered. Since $\Pi_j^{\ell_j}$ in Bad_1 is equal to or partnered with $\Pi_s^{\ell_s^*}$, so is $\Pi_i^{\ell_i}$. By Claim 4, there is at most one such instance in user i with this property. So under our description of $\mathcal{A}$, the simulation of $\Pi_i^{\ell_i}$ in Π will use the messages in tr^*. So m is the corresponding message in tr^*. This contradicts the definition of Bad_1. □

Now we come back to our theorem proof. By Claim 5, before an abortion event occurs, the view of $\mathcal{A}'$ is identical to his view in a real game. As $\Pi_s^{\ell_s^*}$ is taken uniformly random from all possible initiator instances, it is identical to the initiator instance that is equal to or partnered with Test session with probability $1/\nu$. Given this, the view of $\mathcal{A}'$ is identical to the real game and hence has a non-negligible advantage ϵ. This implies that the advantage of $\mathcal{A}$ is at least ϵ/ν, non-negligible. This contradicts the passive security of Π. ■

7 Conclusion

We studied a group key agreement problem, where a user is only aware of his neighbors while the connectivity graph is arbitrary. In addition, users are initialized completely independent of each other. A group key agreement in this setting is very suitable for applications such as social networks. We constructed two passively secure protocols with contributiveness and proved lower bounds on a round complexity, demonstrating that our protocols are round efficient. Finally, we constructed an actively secure protocol from a passively secure one. In our work, we did not consider how to update the group key more efficiently than just running the protocol again, when user memberships are changing. We are not clear how to do this. One can either propose algorithms to our current protocols (as Dutta and Barua [22] did for [17]) or construct a completely new key agreement with these features. We leave it as an open question.

Acknowledgments

The author would like to thank anonymous reviewers and the associate editor for valuable comments. This work is supported by National 973 Program of China (No. 2013CB834203).

References

[1] Y. Amir, Y. Kim, C. Nita-Rotaru and G. Tsudik, "On the Performance of Group Key Agreement Protocols", *ACM Trans. Inf. Syst. Secur.*, vol. 7, no. 3, pp. 457-488, Aug. 2004.

[2] D. Augot, R. Bhaskar, V. Issarny and D. Sacchetti, "An Efficient Group Key Agreement Protocol for Ad Hoc Networks", *Proc. 6th IEEE Int'l Symp. on a World of Wireless Mobile and Multimedia Networks (WOWMOM 2005)*, pp. 576-580, 2005.

[3] A. Beimel and B. Chor, "Communication in Key Distribution Schemes", *Proc. Advances in Cryptology (CRYPTO'93)*, vol. 773, pp. 444-455, 1994.
[4] R. Blom, "An Optimal Class of Symmetric Key Generation Systems", *Proc. Advances in Cryptology-EUROCRYPT'84*, vol. 209, pp. 335-338, 1984.
[5] D. Boneh and M. K. Franklin, "An Efficient Public-key Traitor Tracing Scheme", *Proc. Advances in Cryptology (CRYPTO'99)*, vol. 1666, pp. 338-353, 1999.
[6] D. Boneh, C. Gentry and B. Waters, "Collusion Resistant Broadcast Encryption with Short Ciphertexts and Private Keys", *Proc. Advances in Cryptology (CRYPTO'05)*, vol. 3621, pp. 258-275, 2005.
[7] D. Boneh, A. Sahai and B. Waters, "Fully Collusion Resistant Traitor Tracing with Short Ciphertexts and Private Keys", *Proc. 25th Int'l Conf. Theory and Application of Cryptographic Techniques (EUROCRYPT'06)*, vol. 4004, pp. 573-592, 2006.
[8] D. Boneh and M. Naor, "Traitor Tracing with Constant Size Ciphertext", *Proc. 15th ACM Conf. Computer and Comm. Security*, pp. 501-510, 2008.
[9] D. Boneh and A. Silverberg, "Applications of Multilinear Forms to Cryptography", *Contemporary Mathematics*, Vol. 324, American Mathematical Society, pp. 71-90, 2003.
[10] C. Blundo, L. A. Mattos and D. R. Stinson, "Generalized Beimel-Chor Schemes for Broadcast Encryption and Interactive Key Distribution", *Theor. Comp. Sci.*, vol. 200, no. 1-2, pp. 313-334, 1998.
[11] C. Blundo and A. Cresti, "Space Requirements for Broadcast Encryption", *Proc. Advances in Cryptology - EUROCRYPT 1994*, vol. 950, pp. 287-298, 1995.
[12] C. Blundo, A. De Santis, A. Herzberg, S. Kutten, U. Vaccaro and M. Yung, "Perfectly Secure Key Distribution for Dynamic Conferences", *Inf. Comput.*, vol. 146, no. 1, pp. 1-23, 1998.
[13] C. Boyd and J. M. González-Nieto, "Round-Optimal Contributory Conference Key Agreement", *Proc. Public Key Cryptography (PKC'03)*, vol. 2567, pp. 161-174, 2003.
[14] E. Bresson, O. Chevassut and D. Pointcheval, "Provably Authenticated Group Diffie-Hellman Key Exchange The Dynamic Case", *Proc. 7th Int'l Conf. Theory and Application of Cryptology and Information Security (ASIACRYPT'01)*, vol. 2248, pp. 290-309, 2001.
[15] E. Bresson, O. Chevassut and D. Pointcheval, "Dynamic Group Diffie-Hellman Key Exchange under Standard Assumptions", *Proc. 21th Int'l Conf. Theory and Application of Cryptographic Techniques (Eurocrypt'02)*, vol. 2332, pp. 321-336, 2002.
[16] E. Bresson, O. Chevassut, D. Pointcheval and J. J. Quisquater, "Provably Authenticated Group Diffie-Hellman Key Exchange", *Proc. 8th ACM Conf. Computer and Comm. Security (CCS'01)*, pp. 255-264, 2001.
[17] M. Burmester and Y. Desmedt, "A Secure and Efficient Conference Key Distribution System", *Proc. Advances in Cryptology-EUROCRYPT'94*, vol. 950, pp. 275-286, 1994.
[18] R. Canetti, J. A. Garay, G. Itkis, D. Micciancio, M. Naor and B. Pinkas, "Multicast Security: a Taxonomy and Some Efficient Constructions", *Proc. IEEE INFOCOM 1999*, vol. 2, pp. 708-716, 1999.
[19] S. Cimato, A. Cresti and P. D'Arco, "A Unified Model for Unconditionally Secure Key Distribution", *Journal of Computer Security*, vol. 14, no. 1, pp. 45-64, 2006.
[20] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, Wiley, New York, 2006.
[21] W. Diffie and M. Hellman, "New Directions in Cryptography", *IEEE Trans. Information Theory*, vol. 22, pp. 644-654, 1976.
[22] R. Dutta and R. Barua, "Provably Secure Constant Round Contributory Group Key Agreement in Dynamic Setting", *IEEE Trans. Information Theory*, vol. 54, no. 5, pp. 2007-2025, 2008.
[23] A. Fiat and M. Naor, "Broadcast Encryption", *Proc. Advances in Cryptology (CRYPTO'93)*, vol. 773, pp. 480-491, 1994.
[24] S. Goldwasser and S. Micali, "Probabilistic encryptions", *Journal of Computer and System Science*, vol. 28, no. 2, pp. 270-299, 1984.
[25] I. Ingemarsson, D. T. Tang and C. K. Wong, "A Conference Key Distribution System", *IEEE Trans. Information Theory*, vol. 28, no. 5, pp. 714-719, 1982.
[26] A. Joux, "A One Round Protocol for Tripartite Diffie-Hellman", *Proc. 4th Int'l Symp. Algorithmic Number Theory (ANTS'00)*, pp. 385-394, 2000.
[27] J. Katz and M. Yung, "Scalable Protocols for Authenticated Group Key Exchange", *Proc. Advances in Cryptology (Crypto'03)*, vol. 2729, pp. 110-125, 2003.
[28] Y. Kim, A. Perrig and G. Tsudik, "Tree-based Group Key Agreement", *ACM Trans. Inf. Syst. Secur.*, vol. 7, no. 1, pp. 60-96, 2004.
[29] K. Yongdae, P. Adrian and G. Tsudik, "Group Key Agreement Efficient in Communication", *IEEE Trans. Computers*, vol. 53, no. 7, pp. 905-921, 2004.
[30] H. Kurnio, R. Safavi-Naini and H. Wang, "A Group Key Distribution Scheme with Decenteralized User Join", *Proc. 3rd Int'l Conf. Security in Communication Networks (SCN'02)*, vol. 2576, pp. 146-163, 2003.
[31] T. Matsumoto and H. Imai, "On the Key Predistribution System: A Practical Solution to the Key Distribution Problem", *Proc. Advances in Cryptology (CRYPTO'87)*, vol. 239, pp. 185-193, 1987.
[32] X. Lv, H. Li and B. Wang, "Group Key Agreement for Secure Group Communication in Dynamic Peer Systems", *J. Parallel Distrib. Comput.*, vol. 72, no. 10, pp. 1195-1200, 2012.
[33] D. H. Phan, D. Pointcheval, "About the Security of Ciphers (Semantic Security and Pseudo-Random Permutations)", *Proc. 11th Int'l Workshop on Selected Areas in Cryptography (SAC'04)*, vol. 3357, pp. 182-197, 2004.
[34] W. G. Tzeng and Z. J. Tzeng, "Round Efficient Conference Key Agreement Protocols with Provable Security", *Proc. 6th Int'l Conf. Theory and Application of Cryptology and Information Security (ASIACRYPT'00)*, vol. 1976, pp. 614-627, 2000.
[35] R. Safavi-Naini, S. Jiang, "Non-interactive Conference Key Distribution and Its Applications", *Proc. the 2008 ACM Symposium on Information, Computer and Communications Security (ASIACCS'08)*, pp. 271-282, 2008.
[36] R. Safavi-Naini and S. Jiang, "Unconditionally Secure Conference Key Distribution: Security Notions, Bounds and Constructions", *International Journal of Foundations of Computer Science*, vol. 22, no. 6, pp. 1369-1393, 2011.
[37] M. Steiner, G. Tsudik and M. Waidner, "Diffie-Hellman Key Distribution Extended to Group Communication", *Proc. 3rd ACM Conf. Computer and Comm. Security (CCS'96)*, pp. 31-37, 1996.
[38] C. K. Wong, M. Gouda and S. S. Lam, "Secure Group Communication Using Key Graphs", *Proc. ACM SIGCOMM'98*, pp. 68-79, 1998.
[39] Q. Wu, Y. Mu, W. Susilo, B. Qin and J. Domingo-Ferrer, "Asymmetric Group Key Agreement", *Proc. 28th Int'l Conf. Theory and Application of Cryptographic Techniques (EUROCRYPT'09)*, vol. 5479, pp. 153-170, 2009.
[40] A. Yao, "Theory and Applications of Trapdoor Functions", *Proc. 23th Ann. Symp. Foundations of Computer Science (FOCS'82)*, pp. 80-91, 1982.

Shaoquan Jiang received the B.S. and M.S. degrees in mathematics from the University of Science and Technology of China, Hefei, China, in 1996 and 1999, respectively. He received the Ph.D degree in Electrical and Computer Engineering from the University of Waterloo, Waterloo, ON, Canada, in 2005.

From 1999 to 2000, he was a research assistant at the Institute of Software, Chinese Academy of Sciences, Beijing; from 2005 to 2013, he was a faculty member at the University of Electronic Science and Technology of China, Chengdu, China; from 2013 to now, he is a faculty member at Mianyang Normal University, Mianyang, China. He was a postdoc at the University of Calgary from 2006 to 2008 and a visiting research fellow at Nanyang Technological University from Oct. 2008 to Feb. 2009. His research interests are public-key based secure systems and secure protocols.