Scalable Compilers for Group Key Establishment : Two/Three Party to Group

S. Sree Vivek[1,⋆], S. Sharmila Deva Selvi[1], Deepanshu Shukla[2,], C. Pandu Rangan[1,⋆]

[1]Department of Computer Science and Engineering,
Indian Institute of Technology Madras.
sharmila@cse.iitm.ac.in,svivek@cse.iitm.ac.in, prangan@iitm.ac.in.
[2]Institute of Technology Banaras Hindu University,
deepanshus.itbhu@gmail.com

Abstract. This work presents the first scalable, efficient and generic compilers to construct group key exchange (GKE) protocols from two/three party key exchange (2-KE/3-KE) protocols. We propose three different compilers where the first one is a 2-KE to GKE compiler (2-TGKE) for tree topology, the second one is also for tree topology but from 3-KE to GKE (3-TGKE) and the third one is a compiler that constructs a GKE from 3-KE for circular topology. Our compilers 2-TGKE and 3-TGKE are first of their kind and are efficient due to the underlying tree topology. For the circular topology, we design a compiler called 3-CGKE. 2-TGKE and 3-TGKE compilers require a total of $\mathcal{O}(n \lg n)$ communication, when compared to the existing compiler for circular topology, where the communication cost is $\mathcal{O}(n^2)$. By extending the compilers 2-TGKE and 3-TGKE using the techniques in [18], scalable compilers for tree based authenticated group key exchange protocols (2-TAGKE/3-TAGKE), which are secure against active adversaries can be constructed. As an added advantage our compilers can be used in a setting where there is asymmetric distribution of computing power. Finally, we present a constant round authenticated group key exchange (2-TAGKE) obtained by applying Diffie-Hellman protocol and the technique in [18] to our compiler 2-TGKE. We prove the security of our compilers in a stronger Real or Random model and do not assume the existence of random oracles.

Keywords. Group Key Exchange, Compilers, Tree Based Group Key Exchange, Circular topology, Real or Random Model, Scalability.

1 Introduction

Secure communication over an insecure channel which may be under the control of an adversary, is one of the fundamental goals of cryptography. Encryption and authentication are the main tools for achieving the aforementioned goal. Public key encryption and signature schemes can be used to achieve this but with an overhead of very high cost for the basic operations, which deteriorates the main aim of the goal and may not fit for most of the real time applications. As an alternative, concerned parties may establish a secret key using key exchange(KE) protocols and then use this key to derive keys for symmetric encryption and message authentication schemes. Group key exchange protocols(GKE) aim at establishment of a secret key among a group of n users over an insecure channel. With the increase in use of applications like encrypted group communication for audio-video conferences, chat systems, computer supported collaborative workflow systems etc., group key exchange protocols are gaining more and more attention. Naive approach for designing these GKE is to have a group leader who will choose the secret key and exchange it with the next user, who will exchange with the next etc., but the round complexity for this is $\mathcal{O}(n)$, which is very inefficient and hence non-scalable. Burmester-Desmedt [5][7][6] gave constant round group key exchange protocols, but the protocols given are unauthenticated.

Compilers are tools used for adding extra features to existing KE schemes. As two-party KE protocols are quite well studied in literature [20–29] and there exist efficient three-party KE protocols [30], compilers for transforming 2-KE and 3-KE to GKE can be quite useful in practice. The protocol in [1] gives one such

⋆ Work supported by Project No. CSE/05-06/076/DITX/CPAN on Protocols for Secure Communication and Computation sponsored by Department of Information Technology, Government of India

compiler for transforming secure 2-KE to secure GKE which works on circular topology. Tree topology is more common in practice than circular topology (e.g.: Consider the case of an organization with a CEO at the top, having some managers under his control and further each manager have many subordinates and so on) and also tree based group key exchange protocols are more efficient as they have lesser communication and computation complexity. To our knowledge, no efficient compiler for transforming a 2-KE/3-KE to tree based GKE exists. In this paper, we present the compilers for transforming 2-KE/3-KE to tree based GKE as well as 3-KE to GKE in the circular topology.

Related work. Several Diffie-Hellman based [5][6][7][8][9] and pairing based [11][12][13][14] group key exchange protocols have been proposed. Bresson et. al. [19] gave a GKE (BCPQ) that required n rounds and $\mathcal{O}(n)$ communication and computation. Boyd-Nieto [32] gave constant round group key exchange protocols (BN PK and BN DH) but the length of messages communicated in their protocols is $\mathcal{O}(n)$. Burmester-Desmedt [5], [7] presented a constant round GKE (BD-I) with constant message size for circular topology. They have also given a protocol (BD-II) for tree based settings in [6] but both the protocols are unauthenticated also BD-II is not contributory (i.e., the secret key obtained finally does not involve the contribution of intermediate values all the users in the group). Later, Katz and Yung in [15] have given a compiler for transforming any secure GKE into an authenticated group key exchange (AGKE) protocol, which requires $\mathcal{O}(n)$ signature verifications per user and adds one more round to the protocol. Following that, Katz and Shin [16] gave a compiler for adding universal composibility (UC) and mutual authentication to any secure GKE protocol. Also, Bresson et. al.[17] have proposed a compiler which provides both authentication and mutual authentication to any secure GKE protocol. In a recent work by Desmedt et. al. [18], they obtain authentication for a GKE protocol in tree structure more efficiently than [15]. Most of the tree based key exchange protocols have complexity $\mathcal{O}(\lg n)$, but [15] adds $\mathcal{O}(n)$ computation as each user has to perform $\mathcal{O}(n)$ verifications, while [18] adds only $\mathcal{O}(\lg n)$ computation, thereby keeping the complexity class intact. Mayer and Yung [31] first considered the expansion of authenticated 2-party key transport to authenticated group key transport, but the length of messages communicated in [31] is $\mathcal{O}(n)$. Hwang et. al.[1] proposed a compiler for transforming any secure 2-KE protocol to a secure GKE assuming that group members are arranged in a circular fashion and each member knows the relative position of all other members. Compiler in [1] requires $\mathcal{O}(n^2)$ communication, besides this the resulting protocol is not authenticated. Authors of [1] suggest the use of compiler in [15] for making the resulting protocol authenticated, which in turn adds $\mathcal{O}(n)$ verification per user. In a similar work for password authenticated key exchange(PAKE), Abdalla et.al. [2] gave a compiler from 2 party PAKE to password authenticated group key exchange (GPAKE). They have proved the security of their scheme in a stronger model -'real or random model'(ROR). Wu et. al. [3] extended the work in [2] to consider group dynamism also. Although the protocols resulting from [3] are authenticated, they do not introduce any means for construction of unique session identifier and hence [3] is prone to replay attack, which we show in our paper out of interest. For avoiding this attack a unique session identifier should be established for every new instance of the protocol similar to [15].

Our Contribution: In this paper, we present a replay attack, which is prevalent on any GPAKE protocol obtained by applying the compiler of Wu et. al.[3]. We also present the first compilers for transformation of secure 2/3 KE to TGKE secure against passive eavesdroppers. In the existing works only circular topology was considered but it is important to consider such compilers for tree based protocols also, as tree based topology is very common in practice. Further, we extend the compiler in [1] for converting any 3-KE into GKE for circular topology. We prove the security of our compilers in ROR model and do not use any random oracle assumption. Our compiler requires only $\mathcal{O}(n \lg n)$ communication and computation. For achieving authentication we use techniques of Desmedt et. al. [18] (it keeps the overall complexity $\mathcal{O}(n \lg n)$ only). We also present a constant round authenticated tree based group key exchange protocol(2-TAGKE) obtained by applying our compiler to Diffie-Hellman protocol. Finally, we compare the performance of our compilers and protocols with the existing ones.

Paper Organization: Rest of the paper is organized as follows. In section 2, we discuss security model and efficiency requirements of group key exchange. In section 3, we give a replay attack on the compiler in [3]. In section 4, we present compilers for transformation of 2/3 KE to tree based GKE(2-TGKE, 3-TGKE and 3-CGKE). In section 5, we prove the security of our compilers in ROR model. In section 6, we present a constant round authenticated tree based group key exchange protocol obtained by applying our compiler to

Diffie-Hellman key exchange protocol. Section 7, gives efficiency comparison of our compilers and protocol with the existing ones. Finally we conclude the paper in section 8.

2 Security models for Group key exchange

In this section we give security models and efficiency goals for group key exchange protocols. We follow the model of Katz and Yung [15] who have used the security model for GKE due to Bresson et al. [19]. However our model differs from [15] as we allow multiple test queries. Allowing multiple test queries is an extensively used technique in password authenticated group key exchange(PAKE) and the model is commonly referred to as real or random (ROR) model. It should be noted that ROR model is a stronger model as it considers a stronger adversary, who is capable of asking many test queries. It can be proved that ROR model is equivalent to the prevalent model with a loss of factor r, which is to the number of protocol instances given to adversary. First, we discuss the notations and then we briefly discuss the various oracles which an adversary of GKE has access to. Finally, we recall the efficiency concerns for the GKE.

Participants and Initialization: We assume that set of participants $\mathcal{P}$ is a polynomial size set of users and any subset of $\mathcal{P}$ can establish a session key. For the authenticated version of our protocol, we further assume that during the initialization phase each participant runs an algorithm $\mathcal{G}(1^k)$ to generate a pair of public and private keys (PK, SK). The secret key is stored by the user and public keys are made available to all the members.

Adversarial Model: Each participant is given access to unlimited number of instances of the protocol. We denote the instance i of user U as Π_U^i, each instance can be used only once. As in [15] each instance Π_U^i has various variables $state_U^i$, $term_U^i$, acc_U^i, $used_U^i$, partner id pid_U^i, session id sid_U^i and session key sk_U^i associated with it. Similar to [18], we define gid and rel_U^i where, gid is the group identifier contained in pid_U^i which identifies all the partners involved in the current execution of the protocol and the set $rel_U^i = \{V_1, V_2, \ldots, V_t\}$ is the set of users whose input is processed by U (U is also an element of the set rel_U^i).

The adversary is assumed to have full control over the communication channel. As in [15] and [18], we do not consider malicious insiders. Different adversarial capabilities are captured by giving the adversary access to following oracles:-

1. **Execute**$(U_1, U_2, \ldots, U_n)$**:** This oracle models a passive eavesdropper. GKE is executed between the unused instances of $U_1, U_2, \ldots, U_n \in P$ and the transcript of the execution is returned as the output. Adversary has control over the number of players and their identities.

2. **Send**(U_1, i, M)**:** This oracle models an active adversary. It sends message M to the instance $\Pi_{U_1}^i$ and outputs the reply generated by the instance. This oracle can also be used to prompt $\Pi_{U_1}^i$ to initiate protocol with the unused instances of the users $U_2, U_3, \ldots, U_n$ by calling $Send(U_1, i, (U_2, \ldots, U_n))$.

3. **Reveal**(U, i)**:** Secret key sk_U^i is returned as the output.

4. **Corrupt**(U)**:** Long term secret key SK_U of user U is given as output.

5. **Test**(U, i)**:** This query is allowed only if the session key is defined (i.e. $acc_U^i = true$ and $sk_U^i \neq NULL$) and instance Π_U^i is fresh (we define freshness of instance below). Challenger selects a random bit $b \in \{0, 1\}$ prior to the first call. It returns the session key sk_U^i, if $b = 0$. Otherwise, a uniformly chosen random session key is returned. Similar to [2] we allow an arbitrary number of test queries, but once the test oracle returned a value for an instance Π_U^i, it will return the same value for all instances partnered with Π_U^i(see the definition of partnering below).

A passive adversary is given access to *Execute*, *Reveal*, *Corrupt* and *Test* oracles, while an active adversary is additionally given access to *Send* oracle.

Partnering. As in [15] the session ID sid_U^i equals the concatenation of all messages sent and received by Π_U^i during the course of its execution. While for partner ID, we follow the approach of [18]. Partner ID

pid_U^i consists of the group identifier and the identities of the player in the group with which Π_U^i exchanges messages during the protocol execution. Two instances Π_U^i and Π_U^j are said to be partnered if (1) session ID of both the instances are equal, (2) each of the player belongs to the partner ID of the other and (3) *gid* for both the instances is same.

Correctness. For correctness we require that for all partnered instances Π_U^i, $\Pi_{U'}^j$ such that $acc_U^i = acc_{U'}^j = TRUE$, same valid session key $sk_U^i = sk_U^j$ is established.

Freshness. An instance Π_U^i is fresh unless one of the following is true (1) at some point, the adversary queried $Reveal(U, i)$ or $Reveal(U', i')$ for any $\Pi_{U'}^{i'}$ partnered with Π_U^i or (2) a query $Corrupt(V)$ was asked before a query of the form $Send(U', i', *)$ by V, where V and U' are in pid_U^i.

Security. Let *Succ* be the event that the adversary queries $Test$ oracle only on fresh instances and guesses correctly the bit b used by the $Test$ oracle. The advantage of adversary $\mathcal{A}$ against protocol P is defined as:

$$Adv_{\mathcal{A},P}\left(1^k\right) = |2.Pr\left[Succ\right] - 1|.$$

A protocol P is a secure GKE if it is secure against a passive adversary, i.e. for any PPT adversary $\mathcal{A}$ the advantage $Adv_{\mathcal{A},P}\left(1^k\right)$ is negligible. Protocol P is a secure authenticated GKE(AGKE) if it is secure against an active adversary. We use $Adv_P^{KE}\left(t, q_{ex}, q_t\right)$ to denote the maximum advantage of any passive adversary attacking P, running in time t, asking q_{ex} *Execute* queries and q_t $Test$ queries. For the AKE we use $Adv_P^{AKE}\left(t, q_{ex}, q_s, q_t\right)$ where q_s is number of *Send* queries.

Efficiency Concerns. Number of players in the GKE can be quite large, therefore efficiency is a major concern. For GKE to be scalable it should be constant round, should have minimum possible communication(number of messages exchanged) and computational complexity. We consider both overall complexity and average complexity per user(It should be noted that our protocols are more valuable in the case when a subset of users has access to lesser computational resources).

3 A replay attack:

In this section, we give a replay attack on the protocols produced by [3] upon application of the compiler to some base 2-party PAKE protocols (although they do not claim the resilience against replay attacks, it is an important requirement and can easily be achieved by establishing a unique session ID for every different run of the protocol). For the description of the protocol we refer the reader to [3]. The compiler in [3] is an extended work on 2-party PAKE to GPAKE due to Abdalla et. al. [2] inorder to consider group dynamism (i.e. *Join* and *Leave*) but they do not use the unique randomness as used in [2]. The protocol due to [2] is secure against replay attacks while [3] is not because of the unique randomness used in [2] throughout the scheme.

Let P be the underlying 2-KE(or 2-PAKE) which is authenticated and secure. The scheme in [3] do not give any description of the unique session identifier for the GKE protocol, therefore the protocol obtained is prone to replay attack. Let $\mathcal{A}$ be an active adversary. First $\mathcal{A}$ observes the honest run of the protocol between the members $U_1, U_2, \ldots, U_n$ and saves the values $X_1, X_2, \ldots, X_n$ broadcasted by the users. If every member has broadcasted the right value and the protocol is successful then, $X_1 \oplus X_2 \oplus \ldots \oplus X_n = 0$. $\mathcal{A}$ waits for the initiation of another run of the protocol between the same set of users and allows the first round to complete without any interruption. During the second round when U_1 broadcasts $X_1^{'}$ to all other users, $\mathcal{A}$ replays $X_2^{'}, X_3^{'}, \ldots, X_n^{'}$ on user U_1 as if it were broadcasted by all others to U_1 and lets all other members operate according to the protocol. Here,

$$X_2^{'} = X_2 \oplus X_1,\ X_n^{'} = X_n \oplus X_1^{'} \text{ and } X_i^{'} = X_i \text{ for all other } i.$$

As $X_1^{'} \oplus X_2^{'} \ldots \oplus X_n^{'} = 0$, replayed values pass the verification test of U_1. U_1 computes the secret key according to replayed values, while all others compute the intended secret key. We show that $\mathcal{A}$ has a non negligible advantage in the security game in ROR model. First $\mathcal{A}$ asks the $Execute(U_1, U_2, \ldots, U_n)$ and stores the transcript which contains all the X_i for $i = 1, \ldots, n$. Then it starts another run of the protocol and replays the messages as above using the proper *Send* queries. $\mathcal{A}$ then asks $Test$ oracle to U_1 and another fresh instance(Note that instance corresponding to U_1 is also fresh as no *Reveal* or *Corrupt* query has been asked on this instance). If the randomly chosen bit b of the simulator is 0 then both of them output different value because session keys computed by both of them are different. Otherwise they output the same random value. Therefore, comparing the output of the $Test$ oracles, $\mathcal{A}$ can guess the correct value of bit b and thus can win the game in all the cases.

This attack can be avoided by having a unique session identifier(session ID) for each new run of the protocol and sending it with all the messages so that messages of one session cannot be replayed in the succeeding sessions.

4 Efficient Compilers from 2/3 KE to GKE

In this section we present compilers for transforming a secure 2/3 KE into secure GKE. First, we present a compiler (2-TGKE) for 2-KE to tree based GKE transformation, then we present a similar compiler(3-TGKE) which uses 3-KE as the basic protocol. Finally, we give a compiler(3-CGKE) for converting 3-KE to GKE in circular topology. Protocols obtained by our compilers are not authenticated but authenticated protocols can be obtained by using the techniques in [18] for tree based protocols (Note: It should be noted that using the techniques in [18] keeps the complexity of the protocol intact). For circular geometry, compiler in [15] can be used. Our compiler follows the design similar to [1],[2],[3] but it should be noted that we are the first to consider such compilers for 3-party key exchange protocols as well as for tree topology. We follow the approach of [3] to consider group dynamism. For tree based protocols we assume that users join and leave at lowermost level only so as to maintain the almost complete structure of the binary tree. Due to space constraint we give $Join$ and $Leave$ for 2-TGKE only, which can easily be adapted to work for 3-TGKE and 3-CGKE also.

4.1 2-KE to TGKE (2-TGKE)

Let the members who wish to share the key be arranged in an almost complete binary tree structure (refer **Fig. 1.**) We denote user i by U_i and use $parent(i)$, $left(i)$, $right(i)$ to denote parent of U_i, left child of U_i and right child of U_i respectively. We assume that 1 and 2 are parents of each other. Let 2-P be the underlying two party key exchange protocol secure against passive adversary. Let κ be the security parameter then the group key established belongs to $\{0,1\}^\kappa$. Let $G = \langle g \rangle$ be a cyclic group with prime order p. Similar to [3] we choose a collision resistant pseudo random function $\mathcal{F} : \{0,1\}^* \longrightarrow \{0,1\}^l$ and an injective mapping $f : G \longrightarrow \mathbb{Z}_q($ f is introduced to consider the group dynamism). Let $P = \{U_1, U_2, \ldots U_n\}$ be the set of participants. Our compiler works as follows:

1. ***Round 1:*** Each non-leaf user U_i establishes 2-P keys with its parent, left child and right child independently with 3 runs of the 2-P protocol, while a leaf node establishes 2-P key with its parent only. Thus non-leaf nodes have $K_{i,parent(i)}, K_{i,left(i)}$ and $K_{i,right(i)}$, while leaf nodes have $K_{i,parent(i)}$.
2. ***Round 2:*** Each non-leaf user U_i computes and broadcasts
   $X_{left} = f(g^{K_{i,parent(i)}}) \oplus f(g^{K_{i,left(i)}})$ to its left descendants and
   $X_{right} = f(g^{K_{i,parent(i)}}) \oplus f(g^{K_{i,right(i)}})$ to its right descendant

3. Each user U_i computes

$$N_i = f(g^{K_{i,parent(i)}})$$

$$N_{parent(i)} = X_i \oplus N_i$$

$$N_{parent(parent(i))} = X_{parent(i)} \oplus N_{parent(i)}$$

$$\vdots$$

$$N_1 = X_3 \oplus N_3 = f(g^{K_{1,2}}),\ if\ U_i\ is\ a\ descendent\ of\ U_3$$

$$OR$$

$$N_1 = X_4 \oplus N_4 = f(g^{K_{1,2}}),\ if\ U_i\ is\ a\ descendent\ of\ U_4$$

$$OR$$

$$N_2 = X_5 \oplus N_5 = f(g^{K_{1,2}}),\ if\ U_i\ is\ a\ descendent\ of\ U_5$$

$$OR$$

$$N_2 = X_6 \oplus N_6 = f(g^{K_{1,2}}),\ if\ U_i\ is\ a\ descendent\ of\ U_6$$

Now, the shared secret key can be obtained by U_i as $\mathcal{F}\left(f(g^{K_{1,2}}) \parallel P\right)$.

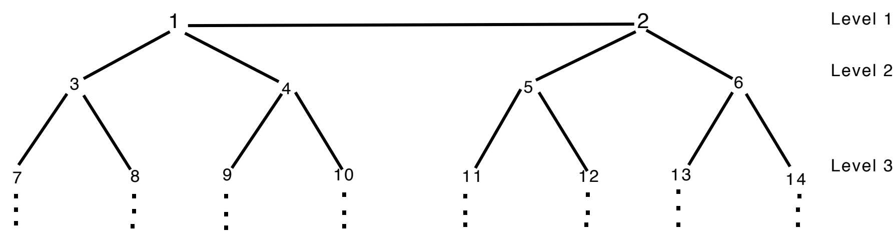

Fig. 1.

Now we give *Join* and *Leave* algorithms for considering group dynamism for 2-TGKE.

Join: Let $\mathcal{J} = \{U_{n+1}, U_{n+2}, \ldots, U_{n+n'}\}$ $(n' \geq 1)$ be the set of new members who wish to join the existing group P. It is required that none of these new members should be able to know the previously established keys. Algorithm works as follows:

- ***Round 1:*** Members of $\mathcal{J}$ are arranged at the bottom of tree so as to maintain the almost complete binary tree structure. Each member $U_{n+j}(j = 1, \ldots, n')$ of $\mathcal{J}$ executes 2-P with its parent and with both children also independently if it is a non-leaf node to get the corresponding keys. All previously existing members update their old 2-P keys by squaring them to obtain their new 2-P keys ($K^2_{i,parent(i)}, K^2_{i,left(i)}, K^2_{i,right(i)}$, for $i = 1$ to n).
- ***Round 2:*** Same as **Round 2** of 2-TGKE with group size $n + n'$.
- The shared secret key is computed similar to **step 3** of 2-TGKE with group size $n + n'$.

Intuitively, the security depends on the inability of a newly joining member to compute g^x from g^{x^2}.

Leave: Let $\mathcal{L} = \{U_{n-n'+1}, \ldots, U_n\}$ $(n' \geq 1)$ be the set of n' members who wish to leave from the existing group P. It is required that none of these old members should be able to know the future keys of the reduced group. We assume that the members leave from bottom only, maintaining the almost complete binary tree structure. Algorithm works as follows:

- ***Round 1:*** Old 2-P keys established with the members of $\mathcal{L}$ are expired and the remaining members update their new keys to the square of their old 2-P keys($K^2_{i,parent(i)}, K^2_{i,left(i)}, K^2_{i,right(i)}$ etc.).
- ***Round 2:*** Same as **Round 2** of 2-TGKE with group size $n - n'$.
- The shared secret key is calculated similar to **step 3** of 2-TGKE with group size $n - n'$.

Security here depends on the inability of a leaving member to compute g^{x^2} from g^x.

Remark 1: Note that leaf nodes don't have to do any broadcast, also leaf nodes have to participate only in one run of the 2-P protocol while all non-leaf nodes have to participate in 3-runs of the 2-P protocol. The protocol obtained can be used for the settings where, some users have lower computational power and can't do highly expensive broadcasts. Such a scene can occur quite frequently, for example consider the case of security agency of some country having a head office in the capital where enough computational power is available, it is also having zonal offices in state capitals and regional offices in other cities with in the state, with computing powers in decreasing order. Finally, it is having some local agents in different cities of various countries with mobiles or other hand held devices. Here, we require (1) Tree structure as an implicit requirement, also (2) users at the lowest level don't have enough computational power so a GKE given by above compiler will be well suited for this kind of applications.

Remark 2: Note that the protocol obtained is secure against passive adversary only. To make it secure against an active adversary we use the technique presented in [18].

4.2 3-KE to TGKE(3-TGKE)

The tree in **Fig. 2**. gives a clean picture of the setting up of users in a binary tree format. Let 3-P be a three party key exchange protocol secure against passive adversary. We show that it can be extended to obtain a GKE secure against passive adversary in the following way:

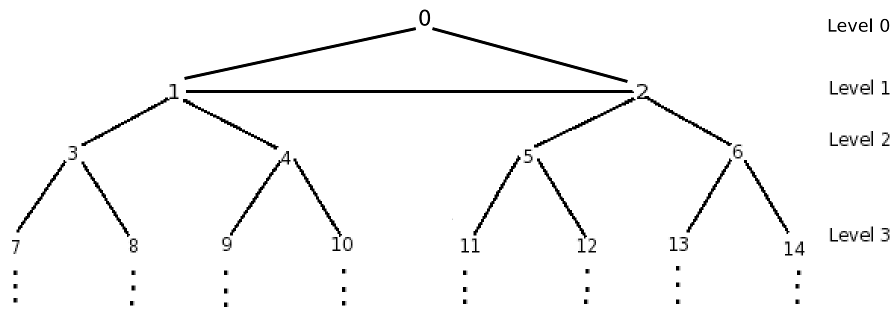

Fig. 2.

1. ***Round 1:*** Each non-leaf user U_i (from level 1 onwards) establishes 3-P keys with (1) both children and (2) its parent and sibling independently by running two instances of 3-P, while each leaf node establishes a 3-P with its parent and sibling alone. Thus, non-leaf nodes compute $K_{i,left(i),right(i)}$ and $K_{i,parent(i),sibling(i)}$ while leaf nodes compute only $K_{i,parent(i),sibling(i)}$ (note: we use $sibling(i)$ to represent the sibling of U_i with respect to its parent $parent(i)$).
2. ***Round 2:*** Each non-leaf user U_i(from level 1 onwards) computes and broadcasts to all its descendants:

$$X_i = f(g^{K_{i,parent(i),sibling(i)}}) \oplus f(g^{K_{i,left(i),right(i)}})$$

3. Each user i ($i \geq 2$) computes,

$$N_i = f(g^{K_{i,parent(i),sibling(i)}})$$

$$N_{parent(i)} = X_{parent(i)} \oplus N_i$$

$$N_{parent(parent(i))} = X_{parent(parent(i)) \oplus N_{parent(i)}}$$

$$\vdots$$

$$N_1 = X_1 \oplus N_3 = f(g^{K_{0,1,2}}),\ if\ U_i\ is\ a\ descendent\ of\ U_3$$

$$OR$$

$$N_1 = X_1 \oplus N_4 = f(g^{K_{0,1,2}}),\ if\ U_i\ is\ a\ descendent\ of\ U_4$$

$$OR$$

$$N_2 = X_2 \oplus N_5 = f(g^{K_{0,1,2}}),\ if\ U_i\ is\ a\ descendent\ of\ U_5$$

$$OR$$

$$N_2 = X_2 \oplus N_6 = f(g^{K_{0,1,2}}),\ if\ U_i\ is\ a\ descendent\ of\ U_6$$

Now, $\mathcal{F}\left(f(g^{K_{0,1,2}}) \parallel P\right)$ is the shared secret key of the group that is computed by each U_i.

Remark 3: Inorder to achieve group dynamism, *Join* and *Leave* can be done similar to 2-TGKE.

4.3 3-KE to GKE for circular topology(3-CGKE)

Consider n users $U_1, U_2, \ldots U_n$ arranged in a circular fashion (i.e. $U_{n+1} = U_1$, $U_{n+2} = U_2 \ldots$). Assuming that the number of users in the group is even ($n = 2m$), the compiler works as follows:

1. ***Round 1:*** Each user U_i (for $i = 1, 3, 5, \ldots, 2m-1$) establishes 3-P key with $i-2, i-1$ and $i+1, i+2$ to get $K_{i,i-1,i-2}$ and $K_{i,i+1,i+2}$.
2. ***Round 2:*** Each user U_i for ($i = 1, 3, 5, \ldots, 2m-1$) broadcasts $X_i = f(g^{K_{i,i-1,i-2}}) \oplus f(g^{K_{i,i+1,i+2}})$ to all other users in the group.

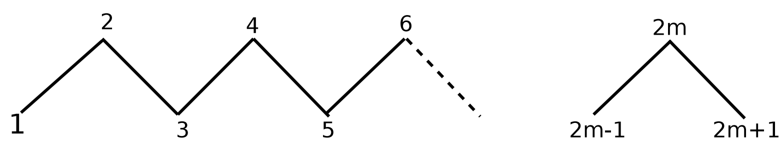

Fig. 3.

3. Each user checks whether $X_1 \oplus X_3 \oplus \ldots X_{2m-1} \stackrel{?}{=} 0$. If the check fails it sets $acc^i_U = 0$ and terminates the protocol. Otherwise, it evaluates (assuming that user U is from the triplet U_i, U_{i+1}, U_{i+2})

$$N_i = f(g^{K_{i,i+1,i+2}})$$
$$N_{i+2} = X_{i+2} \oplus N_i$$
$$\vdots$$
$$N_{i+2m-2} = X_{i+2m-2} \oplus N_{i+2m-4}$$

In this way each user evaluates the secret key of all the triplets and sets $\mathcal{F}(f(g^{K_{1,2,3}}) \parallel f(g^{K_{3,4,5}}) \ldots f(g^{K_{2m-1,2m,1}}) \parallel P)$ as the shared secret key of the group.

5 Security Proof:

In this section we prove the security of the compilers given in section 4. We prove that, if the underlying protocol is secure against a passive adversary then the resulting GKE is secure against a passive adversary. We consider security in ROR model where multiple $Test$ queries are allowed. ROR model is apparently stronger than traditional model where only one $Test$ query is allowed, although both the model can be shown to be equivalent with a loss of factor r, which is the number of protocol instances that the adversary is given access to. Our proof is inline with [4][2] except that we do not give access to $Send$ oracles as we are dealing with a passive adversary. We give the proof of security of our first compiler (2-TGKE) only, which can easily be adapted for other two compilers. Due to space constraint we do not give proof of security for $Join$ and $Leave$, we have given intuition for security with respect to $Join$ and $Leave$ algorithms in section 4.1, for full proof we refer the reader to $Theorem 2$ of [3].

Theorem 1. *Let $\mathcal{A}^{ror-KE}_{GKE}$ be a passive adversary having non-negligible advantage against the security of group key exchange produced by $2-TGKE$ compiler, running in time t and asking at most q_{exe}, q_{reveal} and q_t Execute, Reveal and Test queries respectively then there exists a passive adversary $\mathcal{A}^{ror-KE}_{2-P}$ against the security of the underlying 2-KE protocol 2-P such that*

$$Adv^{ror-KE}_{\mathcal{A}_{GKE}}(t, q_{exe}, q_{reveal}, q_t) \leq 2Adv^{ror-KE}_{\mathcal{A}_{2-P}}(t, 2nq_{exe}, q_{reveal}, nq_{exe} + 2q_t) +$$

$$\frac{q^2_{exe}}{2q} + \frac{q^2_{exe}}{2^{l+1}}$$

Proof: We prove the security using a sequence of hybrid games, starting with the real attack and ending in a game where adversary's advantage is zero. We use Adv_{G_i} to denote the advantage of $\mathcal{A}_{GKE}^{ror-KE}$ in *Game i*.
Game 0: This game corresponds to the real attack. By definition we have

$$Adv(G_0) = Adv_{\mathcal{A}_{GKE}}^{ror-KE}$$

Game 1: In this game we replace all the 2-P session keys by random session keys. We show below that difference between the advantages of adversary in *Game* 0 and *Game* 1 is at most that of the advantage of $\mathcal{A}_{2-P}^{ror-KE}$ against the security of underlying 2-KE protocol 2-P

$$|Adv(G_1) - Adv(G_0)| = 2Adv_{\mathcal{A}_{2-P}}^{ror-KE}(t, 2nq_{exe}, q_{reveal}, nq_{exe} + 2q_t)$$

We show this by constructing an adversary $\mathcal{A}_{2-P}^{ror-KE}$ using an adversary $\mathcal{A}$, distinguishing *Game* 0 from *Game* 1.

$\mathcal{A}_{2-P}^{ror-KE}$ is given access to the simulation of 2-P. To answer its queries, $\mathcal{A}_{2-P}^{ror-KE}$ first associates three instances of 2-P with each non-leaf and a single instance of 2-P with each leaf user according to the specification of 2-TGKE in section 5. Now, whenever $\mathcal{A}$ queries a *Corrupt* query, $\mathcal{A}_{2-P}^{ror-KE}$ answers it by querying its own *Corrupt* oracle. To answer an *Execute* query it first queries the *Execute* oracles of 2-P instances to obtain the transcript for Round 1. To simulate the following rounds, $\mathcal{A}_{2-P}^{ror-KE}$ first queries the $Test$ oracles of instances of 2-P and uses the returned values as the 2-P keys of Round 1 ($K_{i,parent(i)}, K_{i,left(i)}, K_{i,right(i)}$ etc.). It uses these values to construct the $X_i's$ broadcasted in Round 2 and returns the transcript. To simulate a *Reveal* query $\mathcal{A}_{2-P}^{ror-KE}$ asks its *Reveal* oracle to the corresponding instance of the user 1 or 2 if it is non-fresh, otherwise, it asks its $Test$ oracle for that instance and uses the 2-P key obtained to construct the session key and returns it. In a similar way it can simulate the $Test$ oracle.

It is easy to observe that view of $\mathcal{A}$ corresponds to *Game* 0, if $Test$ query reveals the actually exchanged key and to *Game* 1 if $Test$ returns a random element from the key space. Thus, $\mathcal{A}$ distinguishes the games *Game* 0 from *Game* 1 with probability at most $2Adv_{\mathcal{A}_{2-P}}^{ror-KE}(t, 2nq_{exe}, q_{reveal}, nq_{exe} + 2q_t)$

Game 2: Now the simulation of *Execute* is modified at the point of computing the session key. Simulator keeps a list of assignments $(N_1, N_2, \ldots, N_n, i_1, i_2, \ldots, i_n, U_1, \ldots, U_n, sk)$. When a *Reveal* query is asked simulator first checks whether the secret key for that session was already established and returns that key. Otherwise, a key $sk \in \{0,1\}^l$ is selected uniformly at random. The output of pseudo random function $\mathcal{F}$ is indistinguishable from the random secret key sk except in case of a collision, which occurs with probability at most $\frac{q_{exe}^2}{2^{l+1}}$. Also a collision might occur in the records $(N_1, N_2, \ldots, N_n)$ but that happens with a probability at most $\frac{q_{exe}^2}{2q}$. Therefore

$$|Adv(G_2) - Adv(G_1)| = \frac{q_{exe}^2}{2q} + \frac{q_{exe}^2}{2^{l+1}}$$

In *Game* 2, all session keys are chosen uniformly at random and the adversary has no advantage i.e. $Adv(G_2) = 0$. We get the result as,

$$Adv_{\mathcal{A}_{GKE}}^{ror-KE}(t, q_{exe}, q_{reveal}, q_t) = |Adv(G_0) - Adv(G_2)|$$

$$\leq |Adv(G_1) - Adv(G_0)| + |Adv(G_2) - Adv(G_1)|$$

6 Constant round authenticated tree based group key exchange protocol (2-TAGKE) - An instantiation

In this section, we give a constant round authenticated tree based group key exchange protocol(2-TAGKE) obtained by applying a combination of our 2-TGKE compiler and techniques of Desmedt et.al.[18] on the Diffie-Hellman key exchange protocol for two parties. Let $P = \{U_1, U_2, \ldots U_n\}$ be the subset of users who wish to establish a common group key and $rel_U = \{V_1, V_2, \ldots, V_{tu}\}$ be the set of ancestors of U(including

U) whose broadcast will be processed by U. Let G be a group of prime order p and $\phi : G \longrightarrow \{0,1\}^\kappa$ be an injective mapping. The set of users proceed in the following way:-

During the initialization phase each party $U \in P$ generates the signing / verification key pair (PK_U, SK_U) using the algorithm $\mathcal{G}(1^\kappa)$.

Round 0:

- Each user U_i chooses a random nonce $r_i \in \{0,1\}^\kappa$ and broadcasts it to its descendants $U_i|0|r_i$. Each instance Π_U^j stores the identities and their per round randomness together with the group ID in $direct_U^j$. Where $direct_U^j = (gid|V_1|r_1 \ldots V_{tu}|r_{tu})$ and stores this as a part of its state information.

Round 1:

- Each non-leaf member U_i sends to its parent and children $U_i|1|g^{a_i}|\sigma$ where $\sigma = Sign_{SK_{U_i}}(1|g^{a_i}|direct_{U_i}^j)$. While leaf members send only to their parent node.
- Each non-leaf member U_i on receiving the messages $U_s|k|g^{a_s}|\sigma_s$ for $s = \{parent(i), left(i), right(i)\}$ checks that (1) $U_s \in pid_{U_i}^j$, (2) $k = 1$ and (3) $Verify_{PK_{U_s}}(k|g^{a_s}|direct_{U_s}^j, \sigma_s) = 1$. If any of these checks fails, $\Pi_{U_i}^j$ aborts the protocol and sets $acc_{U_i}^j$ = FALSE and $sk_{U_i}^j$ = NULL. Otherwise, it computes $K_{i,s} = \phi((g^{a_s})^{a_i}) = \phi(g^{a_s \cdot a_i})$ for $s = \{parent(i), left(i), right(i)\}$. While leaf users proceed in a similar manner to compute $K_{i,parent(i)}$ only.

Round 2:

- Each non-leaf member computes and broadcasts $(U_i|2|X_{left}|\sigma_{left})$, $(U_i|2|\ X_{right}|\sigma_{right})$ to its left descendants and right descendants respectively. Where $X_{left} = f(g^{K_{i,parent(i)}}) \oplus f(g^{K_{i,left(i)}})$, $X_{right} = f(g^{K_{i,parent(i)}}) \oplus f(g^{K_{i,right(i)}})$, $\sigma_{left} = Sign_{SK_{U_i}}(2|X_{left}|direct_{U_i}^j)$ and σ_{right} is equal to $Sign_{SK_{U_i}}(2|X_{right}|direct_{U_i}^j)$.
- Each member U_i on receiving the messages $U_s|k|X_s|\sigma_s$ where $s \in \left\{rel_{U_i}^j\right\}$ first goes through the verification process as in ***Round 1*** and then computes

$$N_i = f(g^{K_{i,parent(i)}})$$

$$N_{parent(i)} = X_i \oplus N_i$$

$$N_{parent(parent(i))} = X_{parent(i)} \oplus N_{parent(i)}$$

$$\vdots$$

$$N_1 = X_3 \oplus N_3 = f(g^{K_{1,2}}),\ if\ U_i\ is\ a\ descendent\ of\ U_3$$

$$OR$$

$$N_1 = X_4 \oplus N_4 = f(g^{K_{1,2}}),\ if\ U_i\ is\ a\ descendent\ of\ U_4$$

$$OR$$

$$N_2 = X_5 \oplus N_5 = f(g^{K_{1,2}}),\ if\ U_i\ is\ a\ descendent\ of\ U_5$$

$$OR$$

$$N_2 = X_6 \oplus N_6 = f(g^{K_{1,2}}),\ if\ U_i\ is\ a\ descendent\ of\ U_6$$

Now, each user computes the shared secret key of the group as $\mathcal{F}\left(f(g^{K_{1,2}}) \parallel P\right)$.

Remark 1: 2-TAGKE runs in 3 rounds. For each run of the protocol $\frac{n}{2}$ non-leaf users require 6 exponentiations (E) and $\frac{n}{2}$ leaf users require only 4 E. (Note: two extra exponentiations in **Round 2** are to consider dynamism). Therefore on an average 5 exponentiations are to be computed per user(3 if dynamism is not considered). On an average each user has to compute 2 signatures(S), $\lg n$ XORs and has to do at most $\mathcal{O}(\lg n)$ signature verifications(V).

Remark 2: 2-TAGKE is efficient communication wise. On an average each user needs to do 1 broadcast(b) to $\mathcal{O}(\lg n)$ users and 2 point to point communication(Overall $\mathcal{O}(n \lg n)$ communication overhead is there as compared to $\mathcal{O}(n^2)$ for [1]). Users at the bottom don't have to do any broadcast, therefore 2-TAGKE is useful in applications where leaf users have lower computational power.

Remark 3: Security of 2-TAGKE follows from the security proof in Section 5 and security proof of [18].

Remark 4: Note that 2-TAGKE is non contributory. In fact, no protocol with computation and communication complexity lower than $\mathcal{O}(n)$ can be fully contributory [18].

In a similar fashion we can obtain 3-TAGKE and 3-CAGKE by application of 3-TGKE and 3-CGKE protocol on joux's 3-party protocol [30].

7 Comparison:

In this section, we compare our compilers with the existing protocols. In **Table 1** we compare the various properties of GKE protocols such as $Topology$, whether they are *Dynamic* or not, the $Work\ Distribution$ among the members of a group and whether the schemes are *Contributory*. We are the first to consider the compiler for transformation of 2/3 KE to GKE in tree based settings. Mayer et.al. [31] considered the transformation of any two party authenticated key transport to group key transport but it is not scalable.

The tree based authenticated group key exchange protocols obtained from 2-TGKE and 3-TGKE by applying the techniques in [18] are named 2-TAGKE and 3-TAGKE respectively. We compare the efficiency of protocols obtained by applying Diffie-Hellman 2-party protocol and Joux's 3-party protocol to our tree based compilers 2-TAGKE and 3-TAGKE respectively with the authenticated version of few existing GKE protocols, based on tree topology and arbitrary structures. We compare the number of rounds (*Rounds*) required to complete the protocol, message length (*Length*) with respect to the security parameter, message complexity ($Message$), communication complexity (*Comm*) and computational complexity (*Computation*). The various notations and parameters which are considered for comparison of computation overhead are 'PM' that denotes scalar multiplication of points on elliptic curve, 'pa' for pairing computation, 'M' for multiplication of field elements, 'E' for exponentiation of a group element, 'S' for signature generation and 'V' for signature verification. The parameters, $Message$ and *Comm* in **Table 2** are compared by the number of point-to-point communication per user represented by '**p**' and the number of broadcasts per user which is represented by '**b**'. In particular, we would like to compare 2-TAGKE with BD-II because both are non-contributory and have $\mathcal{O}(\lg n)$ computation and communication complexity. But 2-TAGKE is dynamic while BD-II is not, also BD-II requires $\lg n$ multiplications while 2-TAGKE requires $\lg n$ XORs which are easier to compute. For 3-TAGKE, if the parameters are chosen appropriately such that pairing computation is fast then the pairing based protocols can give performance comparable to discrete logarithm based protocols.

Compiler of Hwang et. al. [1] converts a two party authenticated KE protocol to GKE assuming the circular topology as the base structure. We compare the authenticated versions of BD-I [5, 7], the GKE obtained by applying Diffie-Hellman 2-party protocol to the compiler in [1] (HLGKE) and 3-CAGKE (applying the authentication compiler given in [15] and 3-party Joux's protocol to 3-CGKE we obtain 3-CAGKE) in **Table 3**. It can be observed from **Table 3** that 3-CAGKE is far more efficient than BD-I and HLGKE. It should be noted that BD-I and HLGKE are contributory but 3-CAGKE is not.

Table 1: Comparison of the properties of few authenticated GKE schemes

	Topology	Dynamic	Work Distribution	Contributory
BCPQ[19]	Any	-	Asymmetric	√
BN PK[32]	Any	-	Asymmetric	√
BN DH[32]	Any	-	Asymmetric	√
BD-I[5, 7]	Circular	-	Symmetric	√
BD-II[6]	Tree	-	Symmetric	-
BD-II seq [6]	Tree	-	Asymmetric	-
HLGKE [1]	Circular	-	Symmetric	√
2-TAGKE	Tree	√	Asymmetric	-
3-TAGKE	Tree	√	Asymmetric	-
3-CAGKE	Circular	√	Asymmetric	√

Table 2: Comparison of the costs of few authenticated GKE schemes in arbitrary and tree topology

	Rounds	Message	Comm	Length	Computation
BCPQ[19]	n	2**b**	$(n-1)$**p**, 2**b**	$\mathcal{O}(1)$	nS, nV, nE, nM
BN PK[32]	1	1**b**	1**b**	$\mathcal{O}(n)$	1S, nV , $2(n-1)$E, $(n-1)$M
BN DH[32]	2	1**p**, 1**b**	$(n1)$**p**, 1**b**	$\mathcal{O}(n)$	2S, nV, nE, $(n-1)$M
BD-II[6]	3	3**p**, 1**b**	6**p**, $(log_2 n)$**b**	$\mathcal{O}(1)$	2S, $(log_2 n)$V, 3E, $(log_2 n)$M
BD-II seq [6]	$(log_2 n)$	5**p**	6**p**	$\mathcal{O}(1)$	3S, 4V , 4E, 2M
2-TAGKE	3	1**b**, 3**p**	$\mathcal{O}(\lg n)$**b**, 3**p**	$\mathcal{O}(1)$	2S, $\lg n$V, $5(3)^*E$, $\lg n$ XOR
3-TAGKE	3	1**b**, 4**p**	$\mathcal{O}(\lg n)$**b**, 4**p**	$\mathcal{O}(1)$	1.5S, $\lg n$V, 3E(1.5E), 1 PM, 1.5 pa, $\lg n$ XOR

Table 3: Comparison of the costs of authenticated GKE in circular topology

	Rounds	Message	Comm	Length	Computation
BD-I[5, 7]	3	2**p**, 1**b**	4**p**, n**b**	$\mathcal{O}(1)$	2S, nV , 3E, $(2n-1)$M
HLGKE [1]	3	1**b**, 2**p**	$(n-1)$**b**, 2**p**	$\mathcal{O}(1)$	2S, $(n+1)$V, 3E, $(n-1)$ XOR
3-CAGKE	3	1**b**, 4**p**	$n-1$**b**, 4**p**	$\mathcal{O}(1)$	1.5S, $\frac{n}{2}$V, 3E(1.5E), 1 PM, 1.5 pa, $(\frac{n}{2}-1)$ XOR

8 Conclusion:

We have presented efficient and scalable compilers for transformation of secure 2/3 KE to secure GKE, both in tree (2-TGKE/3-TGKE) and circular (3-CGKE) topologies. We proved the security of our compilers in ROR model without assuming the existence of random oracles. Protocols produced by our tree based compilers have $\mathcal{O}(\lg n)$ computational and communication complexity which is very efficient when compared to the existing compiler due to [1] which requires $\mathcal{O}(n)$ computational and communication complexity in the circular topology. We have also compared the authenticated versions of the protocols obtained by applying 2-party Diffie-Hellman KE to 2-TGKE (2-TAGKE) and 3-party Joux's protocol to 3-TGKE (3-TAGKE) with few other existing schemes that are in tree as well as arbitrary topologies. We have also demonstrated a weakness in the protocol in [3] due to replay attack. Finally, we presented a constant round authenticated tree based group key exchange protocol with performance better than BD-II. It is an interesting open problem to design a contributory protocol for tree based setting which is efficient like the protocols presented in this paper.

References

1. Hwang, J.Y., Lee, S.M. and Lee, D.H.: *Scalable key exchange transformation: from two-party to group.* In: Electronics Letters, Volume: 40, Issue: 12, Page: 728-729. (2004)
2. Michel Abdalla, Jens-Matthias Bohli, Maria Isabel Gonzalez Vasco and Rainer Steinwandt: *(Password) Authenticated Key Establishment: From 2-Party to Group.* In: Theory of Cryptography, 4th Theory of Cryptography Conference, TCC 2007, Proceedings in Springer, Lecture Notes in Computer Science, Volume: 4392, Page: 499-514. (2007)
3. Shuhua Wu and Yuefei Zhu: *Constant-Round Password-Based Authenticated Key Exchange Protocol for Dynamic Groups.* In: Financial Cryptography and Data Security, 12th International Conference, FC 2008, Revised Selected Papers, Proceedings in Springer, Lecture Notes in Computer Science, volume: 5143, Page: 69-82. (2007)
4. Michel Abdalla, Pierre-Alain Fouque and David Pointcheval.: *Password-Based Authenticated Key Exchange in the Three-Party Setting.* In: Public Key Cryptography, PKC 2005, 8th International Workshop on Theory and Practice in Public Key Cryptography, Proceedings in Springer, Lecture Notes in Computer Science, Volume: 3386, Page: 65-84. (2005)
5. Burmester, M., Desmedt, Y.: *A Secure and Efficient Conference Key Distribution System (Extended Abstract).* In: EUROCRYPT 94, Proceedings in Springer, Lecture Notes in Computer Science, Volume: 950, Page: 275286. (1994)
6. Burmester, M., Desmedt, Y.: *Efficient and secure conference key distribution.* In: Security Protocols 1996, Proceedings in Springer, Lecture Notes in Computer Science, Volume: 1189, Page: 119130. (1996)
7. Burmester, M., Desmedt, Y.: *A secure and scalable group key exchange system.* In: Information Processing Letters, Volume: 94(3), Page: 137143.

8. Just, M., Vaudenay, S.: *Authenticated multi-party key agreement.* In: Advances in Cryptology - ASIACRYPT 1996, International Conference on the Theory and Applications of Cryptology and Information Security, Volume: 1163, Page: 36-49. (1996)
9. Ingemarsson, I., Tang, D.T., Wong, C.W..: *A conference key distribution system.* In: IEEE Transaction Information Theory, Volume: 28, Page: 714720. (1982)
10. Rana Barua, Ratna Dutta and Palash Sarkar.: *Extending Joux's Protocol to Multi Party Key Agreement (Extended Abstract).* In: Progress in Cryptology, INDOCRYPT 2003, 4th International Conference on Cryptology in India, Proceedings in Springer, Lecture Notes in Computer Science, Volume: 2904, Page: 205-217. (2003)
11. Rana Barua, Ratna Dutta and Palash Sarkar.: *Provably secure authenticated tree based group key agreement protocol using pairing.* In: Information and Communications Security, ICICS 2004, 6th International Conference, Proceedings in Springer, Lecture Notes in Computer Science, Volume: 3269, Page: 92-104. (2004)
12. Kyu Young Choi, Jung Yeon Hwang and Dong Hoon Lee.: *Efficient ID-based Group Key Agreement with Bilinear Maps.* In: Public Key Cryptography, PKC 2004, 7th International Workshop on Theory and Practice in Public Key Cryptography, Proceedings in Springer, Lecture Notes in Computer Science, Volume: 2947, Page: 130-144. (2004)
13. Xinjun Du, Ying Wang, Jianhua Ge and Yumin Wang.: *An Improved ID-based Authenticated Group Key Agreement Scheme.* In: Cryptology ePrint Archive, Report 2003/260. (2003)
14. Yvo Desmedt and Tanja Lange.: *Revisiting Pairing Based Group Key Exchange.* In: Financial Cryptography and Data Security, 12th International Conference, FC 2008, Proceedings in Springer, Lecture Notes in Computer Science, Volume: 5143, Page: 53-68. (2008)
15. Jonathan Katz and Moti Yung.: *Scalable Protocols for Authenticated Group Key Exchange.* In: Advances in Cryptology, CRYPTO 2003, 23rd Annual International Cryptology Conference, Proceedings in Springer, Lecture Notes in Computer Science, Volume: 2729, Page: 110-125. (2003)
16. Jonathan Katz and Ji Sun Shin.: *Modeling insider attacks on group key-exchange protocols.* In: Proceedings of the 12th ACM Conference on Computer and Communications Security, CCS 2005, ACM, Page: 180-189. (2005)
17. Emmanuel Bresson, Mark Manulis and Jörg Schwenk.: *On Security Models and Compilers for Group Key Exchange Protocols.* In: Advances in Information and Computer Security, Second International Workshop on Security, IWSEC 2007, Proceedings in Springer, Lecture Notes in Computer Science, Volume: 4752, Page: 292-307. (2007)
18. Yvo Desmedt, Tanja Lange and Mike Burmester.: *Scalable Authenticated Tree Based Group Key Exchange for Ad-Hoc Groups.* In: Financial Cryptography and Data Security, 11th International Conference, FC 2007, Proceedings in Springer, Lecture Notes in Computer Science, Volume: 4886, Page: 104-118. (2007)
19. Bresson, E., Chevassut, O., Pointcheval, D., Quisquater, J.-J.: *Provably authenticated group Diffie-Hellman key exchange.* In: Proceedings of 8th Annual ACM Conference on Computer and Communications Security, Page: 255-264. (2001)
20. Ran Canetti and Hugo Krawczyk: *Security Analysis of IKE's Signature-Based Key-Exchange Protocol* In: Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Volume: 2442, Page: 143-161. (2002)
21. Ran Canetti and Hugo Krawczyk: *Universally Composable Notions of Key Exchange and Secure Channels* In: Advances in Cryptology - EUROCRYPT 2002, International Conference on the Theory and Applications of Cryptographic Techniques, Volume: 2332, Page: 337-351. (2002)
22. Ran Canetti and Hugo Krawczyk: *Key-Exchange Protocols and Their Use for Building Secure Channels* In: Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques, Volume: 2045, Page: 453-474. (2001)
23. Victor Shoup: *On Formal Models for Secure Key Exchange* In: http://eprint.iacr.org/1999/012. (1999)
24. Mihir Bellare, Ran Canetti and Hugo Krawczyk: *A Modular Approach to the Design and Analysis of Authentication and Key Exchange Protocols (Extended Abstract)* In: STOC-1998, Page: 419-428. (1998)
25. Hugo Krawczyk: *SKEME: A Versatile Secure Key-Exchange Mechanism for the Internet* In: Proceedings of the Internet Society Symposium on Network and Distributed System Security, Page: 114127. (1996)
26. Mihir Bellare and Phillip Rogaway: *Entity Authentication and Key Distribution* In: Advances in Cryptology - CRYPTO '93, 13th Annual International Cryptology Conference, Volume 773, Page: 232-249. (1993)
27. W. Diffie, P. van Oorschot, and M. Wiener: *Authentication and Authenticated Key Exchanges* In: Designs, Codes, and Cryptography, Page: 107125. (1992)
28. Ray Bird, Inder S. Gopal, Amir Herzberg, Philippe A. Janson, Shay Kutten, Refik Molva and Moti Yung: *Systematic Design of Two-Party Authentication Protocols* In: Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Volume: 576, Page: 44-61. (1991)
29. W. Diffie and M. Hellman: *New Directions in Cryptography* In: IEEE Transactions on Information Theory, Page: 644654. (1976)
30. Antoine Joux: In: Algorithmic Number Theory, 4th International Symposium, ANTS-IV, Volume 1838, Page: 385-394. (2000)

31. Alain J. Mayer and Moti Yung: *Secure Protocol Transformation via "Expansion": From Two-Party to Groups* In: ACM Conference on Computer and Communications Security, Page: 83-92. (1999)
32. Colin Boyd and Juan Manuel González Nieto: *Round-Optimal Contributory Conference Key Agreement* In: Public Key Cryptography - PKC 2003, 6th International Workshop on Theory and Practice in Public Key Cryptography, Volume: 2567, Page: 161-174. (2003)