Provably Secure Authenticated Tree Based Group Key Agreement Protocol using Pairing

Rana Barua, Ratna Dutta and Palash Sarkar
Cryptology Research Group
Stat-Math and Applied Statistics Unit
203, B.T. Road, Kolkata
India 700108
e-mail:{rana,ratna_r,palash}@isical.ac.in

Abstract

We present a provably secure authenticated tree based key agreement protocol. The protocol is obtained by combining Boneh *et al.*'s aggregate signature with an unauthenticated ternary tree based multi-party extension of Joux's key agreement protocol. The security is in the standard model as formalized by Bresson *et al.*. The proof is based on the techniques used by Katz and Yung in proving the security of their key agreement protocol.

Keywords : group key agreement, authenticated key agreement, bilinear pairing, provable security.

1 Introduction

Key agreement protocols are important cryptographic primitives. These protocols allow two or more parties to exchange information among themselves over an insecure channel and agree upon a common key. Diffie-Hellman proposed the first two-party single-round key agreement protocol in their seminal paper [22]. In one of the breakthroughs in key agreement protocols, Joux [25] proposed a single-round three party key agreement protocol that uses bilinear pairings. These protocols are unauthenticated in the sense that an active adversary who has control over the channel can mount a man-in-the-middle attack to agree upon separate keys with the users without the users being aware of this. Burmester and Desmedt [19] proposed a multi-party two-round key agreement protocol, a variant of which has shown to be secure against a passive adversary in the standard model under decision Diffie-Hellman assumption by Katz and Yung [26].

Tree based key agreement protocols have applications in cryptography. They provide modularity and also enable different user sets to have different sets of keys. Achieving multiple keys and modularity makes tree based authenticated key agreement protocols desirable in cryptography. Barua, Dutta and Sarkar [5] presented a ternary tree key agreement protocol by extending the basic Joux protocol to multi-party setting with a proof of security against a passive adversary.

Authenticated key agreement protocols are cryptographic protocols by which two or more parties that communicate over an adversarially controlled network can generate a common secret key (session key). These protocols are essential for enabling the use of symmetric key cryptography to protect transmitted data over insecure network and are crucial to construct secure communications in modern cryptography. Authenticated group key agreement protocols are the basic tools for group-oriented and collaborative applications such as, distributed simulation, multi-user games, audio or video-conferencing, electronic notebooks and also peer-to-peer applications that are likely to involve a large number of users.

Recently, Katz and Yung [26] proposed the first scalable, constant round, authenticated group key agreement protocol with forward secrecy. The protocol is a variant of Burmester and Desmedt(BD) [19] key agreement protocol. Katz and Yung [26] provided a detailed proof of security in the security model formalized by Bresson *et al.* [17].

1.1 Our contribution

The main contribution of this paper is to obtain a provably secure authenticated tree based group key agreement protocol from the unauthenticated protocol of Barua, Dutta and Sarkar [5]. This unauthenticated protocol is a multi-party extension of Joux's three-party key agreement protocol. We use the bilinear pairing based aggregate signature scheme introduced by Boneh *et al.* [12] to provide authentication. The security model formalized by Bresson *et al.* [17] is adopted. Modifying the proof technique used by Katz and Yung [26] and making use of bilinear aggregate signature [12], we authenticate the unauthenticated tree based group key agreement protocol specified in [5]. This completes the work to extend Joux's three-party key agreement protocol to multi-party setting with a concrete security analysis against active adversaries in a formal security model. Moreover, a similar construction can be used to make any tree based unauthenticated group key agreement protocol to authenticated group key agreement.

1.2 Related Work

Till a few years ago, a trial and error approach has been taken to provide informal security analysis of key agreement protocols. An extensive work have been considered to extend the two-party key agreement protocols to multi-party setting [3, 6, 19, 24, 27, 28, 32] and consequently, a number of group key agreement protocols have been suggested. However, some of these protocols have flaws that come to light years after its proposal. There are very few key agreement protocols that have concrete security proofs against active adversaries in a well defined formal security model.

Bellare and Rogaway [9] proposed a formal model for proving provable security of protocols in two-party setting. A modular approach is presented by Bellare, Canetti and Krawczyk [7] to design and analyze key agreement protocols. The modularity is achieved by applying a protocol translation tool, called an authenticator/compiler to protocols proven secure in a much simplified adversarial setting where authentication of communication links is not required.

Based on these works, Bresson *et al.* [17] defined a sound formalization for the authenticated group Diffie-Hellman key agreement and provide provably secure protocols within this model. This is an important step and has been used to analyze group key agreement protocols [16, 17, 18].

Boyd and Nieto [15] proposed a constant round authenticated group key agreement protocol that is proven to be secure in the random oracle model. This protocol does not achieve forward secrecy. More recently, Katz and Yung [26] presented a rigorous proof of security of two round group key agreement protocol (unauthenticated) of Burmester and Desmedt (BD) [19] in the standard model under the decision Diffie-Hellman (DDH) assumption. They also provide a compiler construction, application of which makes the unauthenticated BD protocol to a provably secure constant round authenticated group key agreement that achieves forward secrecy. Their security model is in the standard model of Bresson *et al.* [17].

The literature has a vast collection of key agreement protocols most of which does not have a proper security proof in formal security model. The discussion of all of them is beyond the scope of this paper.

2 Preliminaries

2.1 Cryptographic Bilinear Maps

Let G_1, G_2 be two groups of the same prime order q. We view G_1 as an additive group and G_2 as a multiplicative group. Let P be an arbitrary generator of G_1. (*aP denotes P added to itself a times*). Assume that discrete logarithm problem (DLP) is hard in both G_1 and G_2. A mapping $e : G_1^2 \rightarrow G_2$

satisfying the following properties is called a cryptographic bilinear map:
Bilinearity : $e(aP, bQ) = e(P, Q)^{ab}$ for all $P, Q \in G_1$ and $a, b \in Z_q^*$.
Non-degeneracy : If P is a generator of G_1, then $e(P, P)$ is a generator of G_2. In other words, $e(P, P) \neq 1$.
Computable : There exists an efficient algorithm to compute $e(P, Q)$ for all $P, Q \in G_1$.
Modified Weil Pairing [11] and Tate Pairing [4, 23] are examples of cryptographic bilinear maps.

2.2 Decision Hash Bilinear Diffie-Hellman (DHBDH) problem in (G_1, G_2, e)

Instance : (P, aP, bP, cP, r) for some $a, b, c, r \in Z_q^*$ and a one way hash function $H : G_2 \to Z_q^*$.
Solution : Output yes if $r = H(e(P, P)^{abc}) \bmod q$ and output no otherwise.

The DHBDH problem [5] in (G_1, G_2, e) is a combination of bilinear Diffie-Hellman(BDH) problem [11] and hash Diffie-Hellman(HDH) problem [1]. The advantage of any probabilistic, polynomial time, 0/1-valued algorithm $\mathcal{A}$ in solving DHBDH problem in (G_1, G_2, e) is defined to be :
$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{DHBDH}} = |\mathsf{Prob}[\mathcal{A}(P, aP, bP, cP, r) = 1] - \mathsf{Prob}[\mathcal{A}(P, aP, bP, cP, H(e(P, P)^{abc})) = 1] : a, b, c, r {\in_R} Z_q^*|$.

DHBDH assumption : There exists no probabilistic, polynomial time, 0/1-valued algorithm which can solve the DHBDH problem with non-negligible probability of success. In other words, for every probabilistic, polynomial time, 0/1-valued algorithm $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{DHBDH}} < \frac{1}{M^L}$ for every fixed $L > 0$ and sufficiently large integer M.

3 Adversarial Model

Let $\mathcal{P} = \{U_1, \ldots, U_n\}$ be a set of n (fixed) users. At any point, any subset of $\mathcal{P}$ may decide to establish a session key. A user can execute the protocol several times with different partners. The adversarial model consists of allowing each user an unlimited number of instances with which it executes the protocol. Throughout the paper, we will use some notations defined below :

– Π_U^i : i-th instance of user U
– sk_U^i : session key after execution of the protocol by user U in it's i-th instance.
– pid_U^i : partner identity for instance Π_U^i which consists of the users, including U itself, with whom Π_U^i intends to establish a session key.
– sid_U^i : session identity for instance Π_U^i which consists of all instances $\Pi_{U'}^j$, $U' \in \mathsf{pid}_U^i$ and U' has j-th instance while U is executing the protocol with its i-th instance Π_U^i.
– acc_U^i : 0/1-valued variable which is set to be 1 by Π_U^i upon normal termination of the session and 0 otherwise.

We assume that the adversary has complete control over all communications in the network. All information that the adversary gets to see is written in the transcript. So a transcript consists of all the public information flowing across the network. The following oracles model an adversary's interaction with the users in the network :

– $\mathsf{Send}(U, i, m)$: This sends message m to instance Π_U^i and outputs the reply generated by this instance. The adversary is allowed to prompt the unused instance Π_U^i to initiate the protocol with partners $U_2, \ldots, U_l$, $l \leq n$, by invoking $\mathsf{Send}(U, i, \langle U_2, \ldots, U_l \rangle)$.

– $\mathsf{Execute}(V_1, \ldots, V_l)$: Here $\{V_1, \ldots, V_l\}$ is a non empty subset of $\mathcal{P}$. This executes the protocol between unused instances of players $V_1, \ldots, V_l \in \mathcal{P}$ and outputs the transcript of the execution. The adversary is allowed to choose the group members and their identities.

– $\mathsf{Reveal}(U, i)$: This outputs session key sk_U^i.
– $\mathsf{Corrupt}(U)$: This outputs the long-term secret key (if any) of player U.

– $\mathsf{Test}(U, i)$: This query is allowed only once, at any time during the adversary's execution. A random $\mathsf{coin} \in \{0, 1\}$ is generated; the adversary is given sk_U^i if $\mathsf{coin} = 1$, and a random session key if $\mathsf{coin} = 0$.

The adversary given access to the Execute, Reveal, Corrupt and Test oracles, is considered to be passive while an active adversary is given access to the Send oracle in addition. We say that an instance Π_U^i is *fresh* unless either the adversary, at some point, queried $\mathsf{Reveal}(U, i)$ or $\mathsf{Reveal}(U', j)$ with $U' \in \mathsf{pid}_U^i$ or the adversary queried $\mathsf{Corrupt}(V)$ (with $V \in \mathsf{pid}_U^i$) before a query of the form $\mathsf{Send}(U, i, *)$ or $\mathsf{Send}(U', j, *)$ where $U' \in \mathsf{pid}_U^i$ having j-th instance.

Let Succ denote the event that an adversary $\mathcal{A}$ queried the Test oracle to the protocol XP on a fresh instance Π_U^i for which $\mathsf{acc}_U^i = 1$ and correctly predicted the coin used by the Test oracle in answering this query.

We define

$$\mathsf{Adv}_{\mathcal{A},\mathsf{XP}} := |2 \; \mathsf{Prob}[\mathsf{Succ}] - 1|$$

to be the advantage of the adversary $\mathcal{A}$ in attacking the protocol XP. The protocol XP is said to be a *secure unauthenticated group key agreement* (KA) protocol if there is no polynomial time passive adversary with non-negligible advantage. In other words, for every probabilistic, polynomial-time, 0/1 valued algorithm $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A},\mathsf{XP}} < \frac{1}{M^L}$ for every fixed $L > 0$ and sufficiently large integer M. We say that protocol XP is a *secure authenticated group key agreement* (AKA) protocol if there is no polynomial time active adversary with non-negligible advantage. For the sake of convenience, we use the notations UP and AP for the unauthenticated and authenticated protocol respectively. Next we define

$\mathsf{Adv}_{\mathsf{XP}}^{\mathsf{KA}}(t, q_E)$:= the maximum advantage of any passive adversary attacking protocol XP, running in time t and making q_E calls to the Execute oracle.
$\mathsf{Adv}_{\mathsf{XP}}^{\mathsf{AKA}}(t, q_E, q_S)$:= the maximum advantage of any active adversary attacking protocol XP, running in time t and making q_E calls to the Execute oracle and q_S calls to the Send oracle.

The unauthenticated key agreement protocol UP proposed in [5] executes a single Execute oracle. Since it does not involve any long term public/private keys, Corrupt oracles may simply be ignored and thus the protocol achieves the forward secrecy. This protocol UP has been proved to be secure against passive adversary [5] under DHBDH assumption. All communications in the protocol UP are done by representatives. This proof can be extended for the case of multiple Execute oracles without affecting the tightness of the security reduction making use of standard hybrid arguments : if $\mathsf{Adv}_{\mathsf{UP}}^{\mathsf{KA}}(t, 1)$ is the advantage of the protocol UP for single Execute oracle, then with $q_E(> 1)$ Execute oracles, the advantage of UP is

$$\mathsf{Adv}_{\mathsf{UP}}^{\mathsf{KA}}(t, q_E) \leq q_E \; \mathsf{Adv}_{\mathsf{UP}}^{\mathsf{KA}}(t, 1).$$

4 Protocol

4.1 Basic Protocol Requirements

Suppose a set of n users $\{1, 2, \ldots, n\}$, with user i having secret key $s_i \in Z_q^*$, wish to agree upon a common key. Let US be a subset of $\{1, 2, \ldots, n\}$ with consecutive integers. Let us call US to be a user set. Each user set US has a representative. We denote this representative by $\mathsf{Rep(US)}$. For convenience, we take $\mathsf{Rep(US)} = \min(\mathsf{US})$. Let us use the notation $A[1, \ldots, n]$ for an array of n elements $A_1, \ldots, A_n$ and write $A[i]$ and A_i interchangeably.

We take two groups G_1, G_2 of some large prime order q and a cryptographic bilinear map $e : G_1 \times G_1 \to G_2$. Let P be an arbitrary generator of G_1. We choose a hash function $H : G_2 \to Z_q^*$. The public parameters are $\mathsf{params} = (G_1, G_2, e, q, P, H)$.

4.2 Aggregate Signature

In the construction of our authenticated protocol AP, any aggregate signature can be used. To be specific, we use the *bilinear aggregate signature scheme* introduced by Boneh *et al.* [11], secure against existential forgery in the aggregate chosen key model under computational co-Diffie-Hellman assumption. This scheme works in Gap Diffie-Hellman (GDH) groups. We denote this scheme by ASig.

Formally the aggregate signature scheme $\mathsf{ASig} = (\mathcal{G}, \mathcal{K}, \mathcal{S}, \mathcal{V}, A\mathcal{S}, A\mathcal{V})$ consists of six algorithms, where $\mathsf{DSig} = (\mathcal{G}, \mathcal{K}, \mathcal{S}, \mathcal{V})$ is a standard digital signature scheme, called the base signature scheme. Here $\mathcal{G}$ is the randomized system parameters generator algorithm, $\mathcal{K}$ is the randomized key generation algorithm, $\mathcal{S}$ is the (possibly) randomized signing algorithm and $\mathcal{V}$ is the deterministic verification algorithm. The aggregation algorithm and the aggregation verification algorithm are respectively $A\mathcal{S}$ and $A\mathcal{V}$.

The base signature scheme DSig used for this aggregate signature is the co-GDH signature scheme [12], which allows to produce short signatures existentially unforgeable under an adaptive chosen message attack in the random oracle model.

Let $\mathsf{PK}_i, \mathsf{SK}_i$ be the public and private key respectively for player $i, 1 \leq i \leq n$, for the scheme DSig. The aggregate signature is generated as follows :

$$A\mathcal{S}(\mathsf{PK}_1, \ldots, \mathsf{PK}_n, m_1, \ldots, m_n, \sigma_1, \ldots, \sigma_n) = \sigma$$

where $\sigma = \prod_{i=1}^{n} \sigma_i$. Each signature σ_i is valid for message m_i relative to public key PK_i and σ is the single aggregate signature. The messages m_i are all distinct. The verification is done by checking whether $A\mathcal{V}(\mathsf{PK}_1, \ldots, \mathsf{PK}_n, m_1, \ldots, m_n, A\mathcal{S}(\mathsf{PK}_1, \ldots, \mathsf{PK}_n, m_1, \ldots, m_n, \mathcal{S}(\mathsf{SK}_1, m_1), \ldots, \mathcal{S}(\mathsf{SK}_n, m_n))) = 1$.

Note that the aggregate signature verification holds if and only if the individual signature verification in DSig holds. *i.e.* $\mathcal{V}(\mathsf{PK}_i, m_i, \mathcal{S}(\mathsf{SK}_i, m_i)) = 1$ holds for all i, $1 \leq i \leq n$.

4.2.1 Security of Aggregate Signature

Next we describe the security model of aggregate signature : the *aggregate chosen key model.* Informally, the goal of the aggregate forger $\mathcal{F}$ in attacking ASig is to existentially forge an aggregate signature given access to a single challenge public key and a signing oracle on this challenge key. The other public keys are of adversary's choice. Consider the following game that formalizes this idea :

– A public key PK_1, generated at random, is provided to the aggregate forger $\mathcal{F}$ which in addition has also given the power to access the signing oracle corresponding to PK_1.

– $\mathcal{F}$ requests adaptively signatures with PK_1 on messages of his choice.

– At the end, $\mathcal{F}$ outputs $n-1$ additional public keys $\mathsf{PK}_2, \ldots, \mathsf{PK}_n$ where $n \geq 1$ is the number of users participating in the generation of aggregate signature, a sequence of distinct messages $m_1, \ldots, m_n$ and an aggregate signature σ for these messages.

The forger $\mathcal{F}$ wins the game if σ is a valid aggregate signature on messages $m_1, \ldots, m_n$ under public keys $\mathsf{PK}_1, \ldots, \mathsf{PK}_n$ with the restriction that $\mathcal{F}$ did not submit m_1 to the signing oracle. The advantage of the forger $\mathcal{F}$ is defined to be the probability of success of $\mathcal{F}$ in the above game where probability is over coin tosses of the key generation and of $\mathcal{F}$. The aggregate signature scheme ASig is said to be secure if there is no probabilistic, polynomial-time forger with non-negligible advantage.

We denote by $\mathsf{Succ}_{\mathsf{DSig}}(t)$ the maximum advantage of any adversary running in time t in forging a new message/signature pair for the signature scheme DSig and $\mathsf{Succ}_{\mathsf{ASig}}(t)$ is the same for the aggregate signature scheme ASig.

4.3 Description of the Unauthenticated Key Agreement Protocol

First we describe an informal idea of the n-party key agreement protocol of Barua, Dutta and Sarkar [5] which is an extension of Joux's three-party single-round key agreement protocol to multi-party setting. The multi-party protocol KeyAgreement recursively invokes two subroutines CombineThree and CombineTwo – a three group Diffie-Hellman protocol and a two group Diffie-Hellman protocol respectively.

Let $p = \lfloor \frac{n}{3} \rfloor$ and $r = n \bmod 3$. Informally the set of users $\{1, 2, \ldots, n\}$ is partitioned into three user sets $\mathsf{US}_1, \mathsf{US}_2, \mathsf{US}_3$ with cardinality p, p, p respectively if $r = 0$ or with cardinality $p, p, p+1$ respectively if $r = 1$ or with cardinality $p, p+1, p+1$ respectively if $r = 2$. This top down procedure is used recursively for further partitioning. Essentially a ternary tree structure is obtained. The lower level 0 consists of singleton users having a secret key. Key agreement is done by invoking CombineTwo for two user sets and CombineThree for three user sets in the key tree.

With this tree structure, CombineTwo is never invoked above level 1. The formal description of the three procedures are given below. The representative for a user set US_i is denoted by $\mathsf{Rep}(\mathsf{US}_i)$ which is set to be $\min(\mathsf{US_i})$. *The symbols $\$*$, describing the steps needed to authenticate the unauthenticated protocol, are presented in the next subsection. These are executed for the authenticated version.*

```
procedure CombineThree($\mathsf{US}[1,2,3], s[1,2,3]$)
    $i = 1$ to 3 do
        Let $\{j, k\} = \{1, 2, 3\} \setminus \{i\}$;
        $\mathsf{Rep}(\mathsf{US}_i)$ sends $s_i P$ to all members of both $\mathsf{US}_j, \mathsf{US}_k$;
    end do;
    $i = 1$ to 3 do
        Let $\{j, k\} = \{1, 2, 3\} \setminus \{i\}$;
        each member of $\mathsf{US}_i$ computes $H(e(s_j P, s_k P)^{s_i})$;
    end do;
end CombineThree
```

Common agreed key of user sets $\mathsf{US}_1, \mathsf{US}_2, \mathsf{US}_3$ is $H(e(P, P)^{s_1 s_2 s_3})$;

```
procedure CombineTwo($\mathsf{US}[1,2], s[1,2]$)
```

```
    Rep($\mathsf{US}_1$) generates $\overline{s} \in Z_q^*$ at random and sends $\overline{s}P$ to the rest of the users;
    Rep($\mathsf{US}_1$) sends $s_1P$ to all members of $\mathsf{US}_2$;
    Rep($\mathsf{US}_2$) sends $s_2P$ to all members of $\mathsf{US}_1$;
    each member of $\mathsf{US}_1$ computes $H(e(s_2P, \overline{s}P)^{s_1})$;
    each member of $\mathsf{US}_2$ computes $H(e(s_1P, \overline{s}P)^{s_2})$;
end CombineTwo
```

Common agreed key of user sets $\mathsf{US}_1, \mathsf{US}_2$ is $H(e(P, P)^{s_1 s_2 \overline{s}})$;

```
procedure KeyAgreement($l, \mathsf{US}[i+1, \dots, i+l]$)
    if ($l = 1$) then
        $KEY = s[i+1]$;
    end if
    if ($l = 2$) then
        call CombineTwo($\mathsf{US}[i+1, i+2], s[i+1, i+2]$);
        $1;
        $5;
        Let $KEY$ be the agreed key between user sets $\mathsf{US}_{i+1}, \mathsf{US}_{i+2}$;
    end if
    $n_0 = 0$; $n_1 = \lfloor \frac{l}{3} \rfloor$; $n_3 = \lceil \frac{l}{3} \rceil$; $n_2 = l - n_1 - n_3$;
    $j = 1$ to 3 do
        call KeyAgreement($n_j, \mathsf{US}[i + n_{j-1} + 1, \dots, i + n_{j-1} + n_j]$);
        $\widehat{\mathsf{US}}_j = \mathsf{US}[i + n_{j-1} + 1, \dots, i + n_{j-1} + n_j]$; $\widehat{s}_j = KEY$; $n_j = n_{j-1} + n_j$;
    end do;
    call CombineThree($\widehat{\mathsf{US}}[1, 2, 3], \widehat{s}[1, 2, 3]$);
    $1;
    $5;
    Let KEY be the agreed key among user sets $\widehat{\mathsf{US}}_1, \widehat{\mathsf{US}}_2, \widehat{\mathsf{US}}_3$;
end KeyAgreement
```

The start of the recursive protocol KeyAgreement is made by the following statements:

```
1.  $2;
    $\mathsf{US}_j = j$ for $1 \leq j \leq n$;
    $3;
    $4;
2. call KeyAgreement($n, \mathsf{US}[1, \dots, n]$);
```

4.4 Description of the Authenticated Key Agreement Protocol

We provide the description by describing the annotations **$1**, . . . **$5** in the algorithms of section 4.3.

$1 The j-th message m sent by an instance Π_U^i has the form $U|j|m|\mathsf{US}_F|\mathsf{US}_T$ where US_F denotes the user set with representative U at this instance who sends the message m as its j-th message; US_T denotes the collection of the user sets to whom user U sends its jth message m as a representative of the user set US_F.

$2 $\mathcal{P} = \{1, \dots, n\}$.

$**3** Each user $U \in \mathcal{P}$ generates the verification, signing keys $\mathsf{PK}_U, \mathsf{SK}_U$ respectively according to key-generation of DSig.

$**4** Each user $U \in \mathcal{P}$, in every session, generates its *nonce* value $r_U \in \{0,1\}^p$ for a fixed integer p.

$**5** The members of the group then execute UP with the following modifications :

– Suppose at instance Π_U^i the message $U|j|m|\mathsf{US}_F|\mathsf{US}_T$ (U being the representative of the user set US_F at this instance) is sent as part of protocol UP. Note that according to the protocol UP, the message m is known to all users in US_F. Instead of this action, the following sequence of actions are to be performed.

– each user U_k in US_F computes $\sigma_{U_k} = \mathcal{S}(\mathsf{SK}_{U_k}, k|j|m|r_{U_k})$;
– each user $U_k \neq U$ in US_F sends $U_k|\sigma_{U_k}|r_{U_k}|\{U_k\}|\{U\}|j|m$, m being the j-th message of user U;
– user U (representative) computes the aggregate signature $\sigma = A\mathcal{S}(\{\mathsf{PK}_{U_k}, k|j|m|r_{U_k}, \sigma_{U_k} : U_k \in \mathsf{US}_F\})$ and sends $U|j|m|\sigma|\mathsf{US}_F|\mathsf{US}_T|\{r_{U_k} : U_k \in \mathsf{US}_F\}$. The signature from user $U_k \neq U$ is not verified by U. If such signature is invalid, this will be detected at the time of aggregate signature verification. Note that the message signed by $U_k \in \mathsf{US}_F$ is $k|j|m|r_{U_k}$. Hence for $k_1 \neq k_2$, the messages signed by U_{k_1} and U_{k_2} are distinct.

So in the protocol AP, two different forms of messages are transmitted depending on the role of the user : whether it is representative or not in the specified instance.

– When instance Π_V^i ($V \in \mathsf{US}_T$) receives message $U|j|m|\sigma|\mathsf{US}_F|\mathsf{US}_T|\{r_{U_k} : U_k \in \mathsf{US}_F\}$, it checks that :

– $U \in \mathsf{pid}_V^i$;
– j is the next expected sequence number (in the unauthenticated protocol) for message from U;
– $A\mathcal{V}(\{\mathsf{PK}_{U_k} : U_k \in \mathsf{US}_F\}, \{k|j|m|r_{U_k} : U_k \in \mathsf{US}_F\}, \sigma) = 1$.

Π_V^i aborts the protocol if any of these are violated and sets $\mathsf{acc}_U^i = 0$, $\mathsf{sk}_U^i = \mathsf{null}$. Otherwise, Π_V^i continues as it would in UP upon receiving message $U|j|m|\mathsf{US}_F|\mathsf{US}_T$.

– Each non-aborted instance computes the session key as in UP.

This completes the description of line $**5** and the description of the authenticated protocol.
Note that in the protocol AP, upon receiving a message of the form $U_k|\sigma_{U_k}|r_{U_k}|\{U_k\}|\{U\}|j|m$, an instance Π_V^i ($V \in \mathsf{US}_T$) need not verify the co-GDH signature σ_{U_k} on $k|j|m|r_{U_k}$. If such a signature is invalid, then the corresponding aggregate signature verification will fail and accordingly instance Π_V^i will abort the protocol as above by setting $\mathsf{acc}_U^i = 0$ and $\mathsf{sk}_U^i = \mathsf{null}$.

5 Security Analysis

A secure key agreement should regist both passive and active attacks. The scheme is shown to be secure against a passive adversary in [5] under DHBDH assumption. Here we will analyze the security against an active adversary.

The goal is to show that the protocol UP, secure against a passive adversary, can be transformed into a group key agreement protocol AP secure against an active adversary. For this the first modification used in the protocol UP is $**1**. The security of the modified protocol against a passive adversary follows from that of the protocol UP. We denote this modified protocol by UP itself. Then we have the following theorem.

Theorem 5.1 *The protocol* AP *given above is a group key agreement protocol secure against active adversary achieving the following inequality :*
$\mathsf{Adv}^{\mathsf{AKA}}_{\mathsf{AP}}(t, q_E, q_S) \le \mathsf{Adv}^{\mathsf{KA}}_{\mathsf{UP}}(t_1 + t_2, q_E + q_S/2) + \frac{q_S^2 + q_E q_S}{2^p} + |\mathcal{P}|\ \mathsf{Succ}_{\mathsf{DSig}}(t_1) + |\mathcal{P}|\ \mathsf{Succ}_{\mathsf{ASig}}(t_2)$
where $t_1 + t_2 \le t + (|\mathcal{P}|q_E + q_S)t_{\mathsf{AP}}$, t_{AP} *being the time required for execution of* AP *by any party.*

Proof : Given an active adversary $\mathcal{A}'$ attacking AP, we will construct a passive adversary $\mathcal{A}$ attacking UP. Before defining $\mathcal{A}$, let us first define two events R and F and bound their respective probabilities:

Claim 1 : Let R be the event that a *nonce* repeats. Then

$$\mathsf{Prob}[R] \le \frac{q_S q_E + q_S^2}{2^p}.$$

Proof of Claim 1 : A *nonce* used by any user in response to Send query was used previously by that user in response to either an $\mathsf{Execute}$ query or a Send query. The claim follows immediately from this statement. ■(of Claim 1)

Claim 2 : Let F be the event that a signature is forged. Then

$$\mathsf{Prob}[F] \le |\mathcal{P}|\ \mathsf{Succ}_{\mathsf{ASig}}(t_2) + |\mathcal{P}|\ \mathsf{Succ}_{\mathsf{DSig}}(t_1).$$

Proof of Claim 2 : Let E_1 be the event that $\mathcal{A}'$ makes a query of the type $\mathsf{Send}(V, i, X)$ where X has the form :

$$X = U|j|m|\sigma|\mathsf{US}_F|\mathsf{US}_T|\{r_{U_k} : U_k \in \mathsf{US}_F\}$$

with $A\mathcal{V}(\{\mathsf{PK}_{U_k}, k|j|m|r_{U_k} : U_k \in \mathsf{US}_F\}, \sigma) = 1$

and E_2 be the event that $\mathcal{A}'$ makes a query of the type $\mathsf{Send}(V, i, Y)$ where Y has the form

$$Y = U_k|\sigma_{U_k}|r_{U_k}|\{U_k\}|\{U\}|j|m$$

with $\mathcal{V}(\mathsf{PK}_{U_k}, k|j|m|r_{U_k}, \sigma_{U_k}) = 1$.

Then clearly $F = E_1 \vee E_2$.

First consider the event E_2. Using $\mathcal{A}'$, we may construct an algorithm $\mathcal{F}$ that forges a signature with respect to the standard digital signature scheme DSig as follows :

Given a public key PK, algorithm $\mathcal{F}$ chooses a random $U \in \mathcal{P}$ and sets $\mathsf{PK}_U = \mathsf{PK}$. The other public keys and private keys for the system are generated honestly by $\mathcal{F}$. The forger $\mathcal{F}$ simulates all oracle queries of $\mathcal{A}'$ by executing protocol AP itself, obtaining the necessary signatures with respect to PK_U, as needed, from its signing oracle. Thus $\mathcal{F}$ provides a perfect simulation for $\mathcal{A}'$. So if $\mathcal{A}'$ ever outputs a new valid message/signature pair with respect to $\mathsf{PK}_U = \mathsf{PK}$, then $\mathcal{F}$ outputs this pair as its forgery. The success probability of $\mathcal{F}$ is equal to $\frac{\mathsf{Prob}[E_2]}{|\mathcal{P}|}$ which immediately implies that

$$\mathsf{Prob}[E_2] \le |\mathcal{P}|\ \mathsf{Succ}_{\mathsf{DSig}}(t_1)$$

Next consider the event E_1. Using $\mathcal{A}'$, we may construct an algorithm $\mathcal{F}$ that forges an aggregate signature in the aggregate chosen key model with respect to the aggregate signature scheme ASig as follows :

Given a public key PK_1, algorithm $\mathcal{F}$ chooses a random $U \in \mathcal{P}$, sets $\mathsf{PK}_U = \mathsf{PK}_1$ and honestly generates all other public/private keys for the system. The forger $\mathcal{F}$ simulates all oracle queries of $\mathcal{A}'$ in the natural way by executing protocol AP itself, obtaining the necessary signatures with respect to PK_U, as needed, from its signing oracle and thus providing a perfect simulation for $\mathcal{A}'$. Now, if $\mathcal{A}'$, in some session ever outputs a valid aggregate signature σ on distinct messages $m_1, \ldots, m_l$, $l \leq |\mathcal{P}|$ under public keys $\mathsf{PK}_1, \ldots, \mathsf{PK}_l$ where m_1 is a new message with respect to PK_1, then $\mathcal{F}$ outputs messages $m_1, \ldots, m_l$, public keys $\mathsf{PK}_2, \ldots, \mathsf{PK}_l$ and σ as its forgery. The success probability of $\mathcal{F}$ is equal to $\frac{\mathsf{Prob}[E_1]}{|\mathcal{P}|}$; this immediately implies that

$$\mathsf{Prob}[E_1] \leq |\mathcal{P}|\ \mathsf{Succ}_{\mathsf{ASig}}(t_2).$$

Then $\mathsf{Prob}[F] = \mathsf{Prob}[E_1 \vee E_2] \leq \mathsf{Prob}[E_1] + \mathsf{Prob}[E_2] \leq |\mathcal{P}|\ \mathsf{Succ}_{\mathsf{ASig}}(t_2) + |\mathcal{P}|\ \mathsf{Succ}_{\mathsf{DSig}}(t_1)$, yielding the result of Claim 2. ■(of Claim 2)

Now we describe the construction of the passive adversary $\mathcal{A}$ attacking UP that uses adversary $\mathcal{A}'$ attacking AP as a subroutine and simulates the oracle queries of $\mathcal{A}'$. Adversary $\mathcal{A}$ generates the verification/signing keys $\mathsf{PK}_U, \mathsf{SK}_U$ for each user $U \in \mathcal{P}$ and gives $\{\mathsf{PK}_U\}_{U \in \mathcal{P}}$ to $\mathcal{A}'$. Then $\mathcal{A}$ runs $\mathcal{A}'$ as a subroutine and simulates the oracle queries of $\mathcal{A}'$ as follows using its own queries to the $\mathsf{Execute}$ oracle. $\mathcal{A}$ aborts and outputs a random bit if F or R occur. Otherwise, $\mathcal{A}$ outputs whatever bit is eventually output by $\mathcal{A}'$. (During this simulation, $\mathcal{A}$ maintains a list Nlist) :

Execute queries : Suppose $\mathcal{A}'$ makes a query $\mathsf{Execute}(U_1, \ldots, U_l)$, $l \leq |\mathcal{P}|$. Adversary $\mathcal{A}$ sends the same query to its $\mathsf{Execute}$ oracle, receives in turn a transcript T of an execution of UP. Next, $\mathcal{A}$ chooses $r_1, \ldots, r_l \in \{0,1\}^p$, sets $nonces = ((U_1, r_1), \ldots, (U_l, r_l))$ and stores $(nonces, T)$ in Nlist.
To simulate the transcript T' of an execution of AP, for each message $U|j|m|\mathsf{US}_F|\mathsf{US}_T$ in transcript T, adversary $\mathcal{A}$ computes signatures

$$\sigma_{U_k} = \mathcal{S}(\mathsf{SK}_{U_k}, k|j|m|r_{U_k})$$

for each user $U_k \in \mathsf{US}_F$ and the aggregate signature

$$\sigma = \mathcal{AS}(\{\mathsf{PK}_{U_k}, k|j|m|r_{U_k}, \sigma_{U_k} : U_k \in \mathsf{US}_F\}).$$

$\mathcal{A}$ places $\{U_k|\sigma_{U_k}|r_{U_k}|\{U_k\}|\{U\}|j|m : U_k \in \mathsf{US}_F, U_k \neq U\}$ concatenated with
$U|j|m|\sigma|\mathsf{US}_F|\mathsf{US}_T|\{r_{U_k} : U \in \mathsf{US}_F\}$ in T'. When done, the complete transcript T' is given to $\mathcal{A}'$.

Send queries : Since $\mathcal{A}'$ can not forge signatures with respect to any other users and *nonces* do not repeat with high probability, $\mathcal{A}'$ is "limited" to send messages already contained in T'.

– The initial Send queries to be sent to each participants. For any particular instance Π_U^i, the initial send query has the form $\mathsf{Send}_0(U, i, \langle U_2, \ldots, U_l \rangle)$. On a query $\mathsf{Send}_0(U, i, *)$ of $\mathcal{A}'$, $\mathcal{A}$ chooses at random $r_U \in \{0,1\}^p$. After completion of all Send_0 queries in a particular session, $\mathcal{A}$ sets

$$nonces_U^i = ((U, r_U), (U_2, r_{U_2}), \ldots, (U_l, r_{U_l}))$$

where r_{U_k} is the nonce generated by user $U_k, 2 \leq k \leq l$. Send_0 query has no output to $\mathcal{A}'$.

– In response to a Send_1 query to an instance $\Pi_{U_k}^i$, the value $\mathsf{pid}_{U_k}^i$ and *nonces* have already been defined. $\mathcal{A}$ looks in Nlist for an entry of the form $(nonces, T)$; if no such entry, then $\mathcal{A}$ queries $\mathsf{Execute}(\mathsf{pid}_{U_k}^i)$, receives in return a transcript T, and stores $(nonces, T)$ in Nlist. In either case, $\mathcal{A}$ now has a transcript T associated with the value *nonces*. Next $\mathcal{A}$ finds the (unique) message of the form $U_k|1|m|\{U_k\}|\mathsf{US}_T$ in T, computes the signature $\sigma_{U_k} = \mathcal{S}(\mathsf{SK}_{U_k}, k|1|m|r_{U_k})$ and returns $U_k|1|m|\sigma_{U_k}|\{U_k\}|\mathsf{US}_T|r_{U_k}$ to $\mathcal{A}'$.

– In response to any other Send query to an instance Π_U^i, $\mathcal{A}$ first verifies the correctness of the current incoming message(s) as in the specification of AP and terminates the instance if verification fails. Assuming verification succeeds, $\mathcal{A}$ finds an entry $(nonces, T)$ in Nlist, locates the appropriate message $U|j|m|\mathsf{US}_F|\mathsf{US}_T$ in transcript T, computes the signature $\sigma_{U_k} = \mathcal{S}(\mathsf{SK}_{U_k}, k|j|m|r_{U_k})$ for each user $U_k \in \mathsf{US}_F$ and the aggregate signature

$$\sigma = A\mathcal{S}(\{\mathsf{PK}_{U_k}, k|j|m|r_{U_k}, \sigma_{U_k} : U_k \in \mathsf{US}_F\})$$

and replies to $\mathcal{A}'$ with

$$\{U_k|\sigma_{U_k}|r_{U_k}|\{U_k\}|\{U\}|j|m : U_k \in \mathsf{US}_F, U_k \neq U\}$$

concatenated with

$$U|j|m|\sigma|\mathsf{US}_F|\mathsf{US}_T|\{r_{U_k} : U_k \in \mathsf{US}_F\}.$$

Reveal/Test queries : Transcript T' is defined when $\mathcal{A}'$ makes the query $\mathsf{Reveal}(U, i)$ or $\mathsf{Test}(U, i)$ for an instance for which $\mathsf{acc}_U^i = 1$. We "strip" the signatures, *nonce* values and "omit" the messages of the form $\{U_k|\sigma_{U_k}|r_{U_k}|\{U_k\}|\{U\}|j|m : U_k \in \mathsf{US}_F$ does not represent the user set $\mathsf{US}_F\}$ and denote the resulting transcript by T. Assuming that events R and F do not occur, at least one query must exist which $\mathcal{A}$ made to its own $\mathsf{Execute}$ oracle resulting in transcript T. Then $\mathcal{A}$ makes the appropriate Reveal or Test query to one of the instances involved in this query and returns the result to $\mathcal{A}'$.

Let $\mathsf{Invd} = F \vee R$. As long as Invd do not occur, the above simulation for $\mathcal{A}'$ is perfect. Whenever Invd occurs, adversary $\mathcal{A}$ aborts and outputs a random bit. So

$$\mathsf{Prob}_{\mathcal{A}',\mathsf{AP}}[\mathsf{Succ}|\mathsf{Invd}] = \frac{1}{2}.$$

Now

$$\begin{aligned}
\mathsf{Adv}_{\mathcal{A},\mathsf{UP}} &:= 2\ |\mathsf{Prob}_{\mathcal{A},\mathsf{UP}}[\mathsf{Succ}] - \frac{1}{2}| \\
&= 2\ |\mathsf{Prob}_{\mathcal{A}',\mathsf{AP}}[\mathsf{Succ} \wedge \overline{\mathsf{Invd}}] + \mathsf{Prob}_{\mathcal{A}',\mathsf{AP}}[\mathsf{Succ} \wedge \mathsf{Invd}] - \frac{1}{2}| \\
&= 2\ |\mathsf{Prob}_{\mathcal{A}',\mathsf{AP}}[\mathsf{Succ} \wedge \overline{\mathsf{Invd}}] + \mathsf{Prob}_{\mathcal{A}',\mathsf{AP}}[\mathsf{Succ}|\mathsf{Invd}]\ \mathsf{Prob}_{\mathcal{A}',\mathsf{AP}}[\mathsf{Invd}] - \frac{1}{2}| \\
&= 2\ |\mathsf{Prob}_{\mathcal{A}',\mathsf{AP}}[\mathsf{Succ} \wedge \overline{\mathsf{Invd}}] + \frac{1}{2}\mathsf{Prob}_{\mathcal{A}',\mathsf{AP}}[\mathsf{Invd}] - \frac{1}{2}| \\
&= 2\ |\mathsf{Prob}_{\mathcal{A}',\mathsf{AP}}[\mathsf{Succ}] - \mathsf{Prob}_{\mathcal{A}',\mathsf{AP}}[\mathsf{Succ} \wedge \mathsf{Invd}] + \frac{1}{2}\mathsf{Prob}_{\mathcal{A}',\mathsf{AP}}[\mathsf{Invd}] - \frac{1}{2}| \\
&\geq |2\ \mathsf{Prob}_{\mathcal{A}',\mathsf{AP}}[\mathsf{Succ}] - 1| - |\mathsf{Prob}_{\mathcal{A}',\mathsf{AP}}[\mathsf{Invd}] - 2\ \mathsf{Prob}_{\mathcal{A}',\mathsf{AP}}[\mathsf{Succ} \wedge \mathsf{Invd}]| \\
&\geq \mathsf{Adv}_{\mathcal{A}',\mathsf{AP}} - \mathsf{Prob}[\mathsf{Invd}]
\end{aligned}$$

The adversary $\mathcal{A}$ can make at most $q_E + q_S/2$ queries to its $\mathsf{Execute}$ oracle and since $\mathsf{Adv}_{\mathcal{A},\mathsf{UP}} \leq \mathsf{Adv}_{\mathsf{UP}}^{\mathsf{KA}}(t_1 + t_2, q_E + q_S/2)$ by assumption, we obtain :

$$\mathsf{Adv}_{\mathsf{AP}}^{\mathsf{AKA}} \leq \mathsf{Adv}_{\mathsf{UP}}^{\mathsf{KA}}(t_1 + t_2, q_E + q_S/2) + \mathsf{Prob}[F] + \mathsf{Prob}[R].$$

This yields the statement of the theorem. ■

6 Efficiency

Efficiency of a protocol is measured by communication and computation cost. Communication cost involves counting total number of rounds needed and total number of messages transmitted through the network during a protocol execution. Computation cost counts total scalar multiplications, pairings, group exponentiations *etc.*.

Let Y be the total number of singleton user sets in level 1 and set $R(n) = \lceil \log_3 n \rceil$.

For the unauthenticated protocol [5], number of rounds required is $R(n)$ and total messages sent is $< \frac{5}{2}(n-1)$. The computation cost of this protocol is as follows :
total number of scalar multiplications in G_1 is $< \frac{5}{2}(n-1)$, total pairings required is $nR(n)$ and total group exponentiations in G_2 is $nR(n)$.

Note that in the authenticated protocol, the representative of a user set with more than one user creates an aggregate signature after it collects the basic signatures from the other users in that user set. This makes the representative to wait for accumulating the basic signatures. In the first round of the authenticated protocol, no aggregation of signatures is required because each user set is a singleton set with a user itself being the representative and only a basic signature on the transmitted message is sent by the representative. The authenticated protocol additionally requires the followings :

The number of rounds increases by $R(n) - 1$.
Total number of basic signatures computed is $nR(n)$.
Total number of additional messages (basic signatures) communicated is $n[R(n) - 1] - \frac{3}{2}(3^{R(n)-1} - 1)$.
Total bilinear aggregate signatures computed, communicated and verified is $\frac{3}{2}(3^{R(n)-1} - 1) - Y$.

This protocol involves no basic signature verification, only verification of bilinear aggregate signatures are required.

7 Conclusion

We have described an authentication mechanism to authenticate the protocol proposed by Barua, Dutta and Sarkar [5] in a standard fomalized security model. The bilinear pairing based aggregate signature by Boneh *et al.* [12] is used and the formal security model of Bresson *et al.* [17] is adopted. Using the proof technique of Katz and Yung [26], we get a provably secure authenticated tree based key agreement protocol.

References

[1] M. Abdalla, M. Bellare and P. Rogaway. *DHIES : An encryption scheme based on the Diffie-Hellman problem*, CT-RSA 2001 : 143-158.

[2] G. Ateniese, M. Steiner, and G. Tsudik. *Authenticated Group Key Agreement and Friends.* In ACM CCS 98[1], pages 17-26.

[3] G. Ateniese, M. Steiner, and G. Tsudik. *New Multiparty Authenticated Services and Key Agreement Protocols*, Journal of Selected Areas in Communications, 18(4):1-13, IEEE, 2000.

[4] P. S. L. M. Barreto, H. Y. Kim and M. Scott. *Efficient algorithms for pairing-based cryptosystems.* Advances in Cryptology - Crypto '2002, LNCS 2442, Springer-Verlag (2002), pp. 354-368.

[5] R. Barua, R. Dutta, P. Sarkar. *Extending Joux Protocol to Multi Party Key Agreement.* Indocrypt 2003, Also available at http://eprint.iacr.org/2003/062.

[6] K. Becker and U. Wille. *Communication Complexity of Group Key Distribution.* ACMCCS '98.

[7] M. Bellare, R. Canetti, and H. Krawczyk. *A Modular Approach to the Design and Analysis of Authentication and Key Exchange Protocols.* In Proceedings of the 30th Annual Symposium on the Theory of Computing, pages 419-428. ACM, 1998. http://www.cs.edu/users/mihir/papers/key-distribution.html/.

[8] M. Bellare, D. Pointcheval, and P. Rogaway. *Authenticated Key Exchange Secure Against Dictionary Attacks.* Advances in Cryptology - Eurocrypt 2000, LNCS vol. 1807, B. Preneel ed., Springer-Verlag, 2000, pp. 139-155.

[9] M. Bellare and P. Rogaway. *Entity Authentication and Key Distribution.* Advances in Cryptology - CRYPTO '93, LNCS Vol. 773, D. Stinson ed., Springer-Verlag, 1994, pp. 231-249.

[10] S. Blake-Wilson and A. Menezes. *Security Proofs for Entity Authentication and Authenticated Key Transport Protocols Employing Asymmetric Techniques.* In B. Christianson, B. Crispo, T. Lomas, and M. Roe, editors, *Proceedings of the 5th International Workshop on Security Protocols*, volume 1361 of *Lecture Notes in Computer Science*, pages 137-158. Springer-Verlag, 1997.

[11] D. Boneh and M. Franklin. *Identity-Based Encryption from the Weil Pairing.* In Advances in Cryptology - CRYPTO '01, LNCS 2139, pages 213-229, Springer-Verlag, 2001.

[12] D. Boneh, C. Gentry, B. Lynn and H. Shacham. *Aggregate and Verifiably Encrypted Signature from Bilinear Maps.* Eurocrypt 2003, LNCS 2248, pp. 514-532, Springer-Verlag, 2003.

[13] D. Boneh, B. Lynn, and H. Shacham. *Short Signature from Weil Pairing*, Proc. of Asiacrypt 2001, LNCS, Springer, pp. 213-229, 2001.

[14] D. Boneh and A. Silverberg. *Applications of Multilinear forms to Cryptography*, Report 2002/080, http://eprint.iacr.org, 2002.

[15] C. Boyd and J. M. G. Nieto. *Round-Optimal Contributory Conference Key Agreement.* Public Key Cryptography, LNCS vol. 2567, Y. Desmedt ed., Springer-Verlag, 2003, pp. 161-174.

[16] E. Bresson, O. Chevassut, and D. Pointcheval. *Provably Authenticated Group Diffie-Hellman Key Exchange - The Dynamic Case.* Advances in Cryptology - Asiacrypt 2001, LNCS vol. 2248, C. Boyd ed., Springer-Verlag, 2001, pp. 290-309.

[17] E. Bresson, O. Chevassut, and D. Pointcheval. *Dynamic Group Diffie-Hellman Key Exchange under Standerd Assumptions.* Advances in Cryptology - Eurocrypt '02, LNCS 2332, L. Knudsen ed., Springer-Verlag, 2002, pp. 321-336.

[18] E. Bresson, O. Chevassut, D. Pointcheval, and J. J. Quisquater. *Provably Authenticated Group Diffie-Hellman Key Exchange.* Proc. 8th Annual ACM Conference on Computer and Communications Security, ACM, 2001, pp. 255-264.

[19] M. Burmester and Y. Desmedt. *A Secure and Efficient Conference Key Distribution System.* In A. De Santis, editor, Advances in Cryptology EUROCRYPT '94, Workshop on the theory and Application of Cryptographic Techniques, LNCS 950, pages 275-286, Springer-Verlag, 1995.

[20] R. Canetti and H. Krawczyk. *Analysis of Key-Exchange Protocols and their use for Building Secure Channels.* In B. Pfitzmann, editor, Advances in Cryptographic Techniques, Volume 2045 of LNCS, pages 453-474. Springer-Verlag, 2001.

[21] C. Cocks. *An Identity Based Encryption Scheme based on Quadratic Residues.* In Cryptography and Coding, LNCS 2260, pp. 360-363, Springer-Verlag, 2001.

[22] W. Diffie and M. Hellman. *New Directions In Cryptography.* IEEE Transactions on Information Theory, IT-22(6) : 644-654, November 1976.

[23] S. Galbraith, K. Harrison and D. Soldera. *Implementing the Tate Pairing.* Algorithm Number Theory Symposium - ANTS V, LNCS 2369, Springer- Verlag (2002), pp. 324-337.

[24] I. Ingemarsson, D. T. Tang, and C. K. Wong. *A Conference Key Distribution System.* IEEE Transactions on Information Theory 28(5) : 714-720 (1982).

[25] A. Joux. *A One Round Protocol for Tripartite Diffie-Hellman.* ANTS IV, LNCS 1838, pp. 385-394, Springer-Verlag, 2000.

[26] J. Katz and M. Yung. *Scalable Protocols for Authenticated Group Key Exchange.* In Advances in Cryptology - CRYPTO 2003.

[27] Y. Kim, A. Perrig, and G. Tsudik. *Simple and Fault-tolerant Key Agreement for Dynamic Collaborative Groups.* In S. Jajodia, editor, 7th ACM Conference on Computation and Communication Security, pages 235-244, Athens, Greece, Nov. 2000, ACM press.

[28] Y. Kim, A. Perrig, and G. Tsudik. *Tree based Group Key Agreement.* Report 2002/009, http://eprint.iacr.org, 2002.

[29] Y. Kim, A. Perrig, and G. Tsudik. *Communication-efficient Group Key Agreement.* In information systems security, Proceedings of the 17th International Information Security Conference, IFIP SEC '01, 2001.

[30] A. Shamir. *Identity-based Cryptosystems and Signature Schemes.* In Advances in Cryptology - CRYPTO '84, LNCS 196, pages 47-53, Springer-Verlag, 1984.

[31] V. Shoup. *On Formal Models for Secure Key Exchange.* IBM Technical Report RZ 3120, 1999. http://shoup.net/papers.

[32] M. Steiner, G. Tsudik, M. Waidner. *Diffie-Hellman Key Distribution Extended to Group Communication*, ACM Conference on Computation and Communication Security, 1996.