Asymmetric Group Key Agreement Protocol for Open Networks and Its Application to Broadcast Encryption[☆]

Lei Zhang[1,2*], Qianhong Wu[2,3], Bo Qin[2,4], Josep Domingo-Ferrer[2], Úrsula González-Nicolás[2]

[1] *East China Normal University, Software Engineering Institute, Shanghai, China*
[2] *Universitat Rovira i Virgili, Department of Computer Engineering and Mathematics*
UNESCO Chair in Data Privacy
Av. Països Catalans 26, E-43007 Tarragona, Catalonia
[3] *Wuhan University, School of Computer*
Key Lab. of Aerospace Information Security and Trusted Computing
Ministry of Education, China
[4] *Xi'an University of Technology, School of Science, Department of Maths, China*
leizhang@sei.ecnu.edu.cn,{qianhong.wu,bo.qin,josep.domingo,ursula.gonzaleznicolas}@urv.cat

Abstract

Asymmetric group key agreement is a recently introduced versatile cryptographic primitive. It allows a group of users to negotiate a common encryption key which is accessible to any entities, and each user only holds her respective secret decryption key. This concept not only enables confidential communications among group users but also permits any outsider to send encrypted messages to the group. The existing instantiation is only secure against passive adversaries. In this paper, we first propose an authenticated asymmetric group key agreement protocol which offers security against active attacks in open networks. Based on this protocol, we then proposes a broadcast encryption system without relying on a trusted dealer to distribute the secret keys to the users. Our system is equipped with the *perfect forward security* property and has short ciphertexts. Improved systems are also proposed to allow a sender to select receivers for broadcast encryption and to balance the transmission overhead against the ciphertext size.

Keywords: group communication; asymmetric group key agreement;

[☆]Parts of this paper appeared in [27].
[*]Corresponding author.

broadcast encryption; bilinear map.

1. Introduction

Group Key Agreement (GKA) protocols allow a group of users to come up with a common secret from which a session key can be derived. Therefore, the GKA protocols are likely to be used in numerous group-oriented communication scenarios, such as audio/video conferences, priced VCD/DVD distribution, collaborative computations, secure replicated databases, etc., in order to achieve secure broadcasting at the network layer among the parties. In the conventional definition of GKA, all the group users establish a shared secret key. Hence, only the group users can securely broadcast to others. Recently, the concept of Asymmetric Group Key Agreement (ASGKA) was introduced by Wu *et al.* [25]. In their definition, unlike regular GKA where a common shared secret key is established, the group users merely negotiate a common encryption key which is accessible to any entities, and each user holds a different secret decryption key. Therefore, besides the group users, any outsider can also broadcast encrypted messages to the group, provided that the sender knows the negotiated public encryption key.

Compared with conventional group key protocols, another attractive advantage of ASGKA is the round efficiency. By far, the most popular conventional GKA protocols requires two or more rounds to establish a secret key, among which the Burmester-Desmedt protocol [8] is one of the best-known. In [25], Wu *et al.* proposed a static one-round ASGKA protocol from *scratch* in the sense that the participants do not hold any secret values prior to the execution of the protocol. One-round key agreement protocols have several advantages [16] over key agreement protocols in two or more rounds. For instance, exchanging an email message key among a group of users with a two-round key agreement protocol would require all of them to be connected concurrently. Another scenario described in [25] is a group of friends wishing to share their private files via the insecure Internet; doing so with a two-round key agreement protocol would require all of them to be online at the same time. In practice, it is difficult for a group of distributed users to be online concurrently (especially if they live in different time zones). In these scenarios, the bandwidth is not a problem but the round efficiency is critical. In addition, the bandwidth overhead can be mitigated by the hardware development but the round overhead can only be addressed by more efficient protocols. Recently, Liu *et al.* [19] report that using their new technologies, the bandwidth of networks could be improved by about 10,000 times. This is a powerful argument to support that it is more important to reduce rounds of protocol, which cannot be mitigated by any new technologies but elegant designs.

1.1. Motivation and Contribution

The protocols from scratch are efficient, but they are only secure against passive adversaries who just eavesdrop the communication channel. The attackers in the real world are usually active adversaries who can control the communication channel to mount more powerful attacks. For instance, the attackers may relay, delay, modify, interleave, insert or delete messages during the execution of the protocol. In particular, an active adversary is able to launch well-known man-in-the-middle attacks. Hence, it is vital for an ASGKA protocol to resist the attacks from active adversaries.

Authenticated key agreement protocols [29] guarantee that no entities other than the intended ones can possibly compute the secret keys agreed, no matter whether the adversaries are active or not. In the preliminary version [28] of this paper, we proposed an authenticated version of the static ASGKA protocol [25] *without* requiring additional rounds. Analysis shows that the authenticated ASGKA protocol satisfies all the desirable security properties (*i.e.*, *known-key security*, *unknown key-share*, *key-compromise impersonation*, *perfect forward security* and *key control* defined in Section 2.3) of key agreement protocols. To the best of our knowledge, this is the first ASGKA protocol that captures those attributes. In our basic protocol, similarly to the protocol in [25], for a group of n participants, each participant should publish $\mathcal{O}(n)$ group elements. When the group size is very large, the transmission overhead is very heavy. To reduce the transmission overhead, we use a technique to trade off communication against ciphertext size. In this paper, we compare our ASGKA protocol with two of the most popular conventional authenticated GKA protocols. Results show that our protocol is more efficient in rounds and has comparable computational overhead with other two protocols. We also carry out several simulations to determine the practical performance of our protocol. The experimental results suggest that our protocol performs well in practice.

The problem of broadcast encryption was first studied by Berkovits in [4], and later a more formal study of this primitive was published by Fiat and Naor [13]. Broadcast encryption systems [4, 7, 13, 14, 15] allow a sender to efficiently and securely broadcast information to a dynamically changing group of users over open networks. It is preferable if the system is *public key* (anybody can encrypt) and permits *stateless receivers* (users do not need to update their private keys). We notice that conventional GKA protocols only allow group members to securely broadcast to other group members. Therefore, conventional GKA is not suitable to construct broadcast encryption systems that are *public-key.* It seem that ASGKA protocols can be used directly to construct broadcast encryption systems. However, the ASGKA protocol in [25] as well as our basic authenticated ASGKA protocol are static ASGKA protocols which cannot deal with member changes. In the preliminary version [28] of this paper, based on our authenticated static ASGKA protocol, we proposed a broadcast encryption system that is *public-key* and permits *stateless receivers.* Our new system is efficient and has short ciphertexts. As to security, in addition to the basic property of broadcast encryption systems (*i.e.*, only the group users can decrypt the ciphertexts), our system offers the perfect forward security property. Hence, even if the private keys of some system users are leaked, the secrecy of messages encrypted under previously established group encryption keys is guaranteed. Despite the merits of the basic broadcast encryption system, it has several limitations. In this paper, we investigate the limitations of our basic broadcast encryption system. The limitations are that 1) it only allows a sender to send the messages to *all* the members in the system, which is not always what the sender wants; 2) a third party (system maintainer) may have the ability to decrypt the group messages; 3) since the scale of a broadcast encryption system is hard to anticipate, the transmission overhead may be very high. We propose improved systems to eliminate the above limitations. With the improved systems, a sender may adaptively chose her receivers (even the system maintainer), and, the transmission overhead can be traded-off against ciphertext size. Therefore, the improved systems are more practical for real world applications. We note that, in fact, the broadcast encryption systems proposed are essentially dynamic ASGKA protocols which overcome the limitation of member changes in static ASGKA protocols.

1.2. Related Work

The first solution to key agreement is the Diffie-Hellman protocol [10]. Later Joux [16] proposed a tripartite key agreement protocol. Both protocols are one-round. Many attempts have been made to extend the Diffie-Hellman and Joux protocols to n parties. Among them, one of the most famous GKA protocols is the Burmester-Desmedt protocol [8] which requires two rounds and is the most efficient existing GKA protocol in round efficiency without constraints on n. We note that all of these extensions require more than one round, which leaves the open problem of how to extend the Diffie-Hellman or Joux protocols to more parties without increasing round complexity.

For a key agreement protocol to be usable in open networks, it is essential for the protocol to resist attacks from active adversaries. However, the basic Diffie-Hellman and Joux protocols, as well as the Burmester-Desmedt protocol, do not authenticate the communicating entities. Hence they are not suited for hostile networks where man-in-the-middle attacks may happen. Many protocols are designed to add authentication to these protocols [9, 12, 17, 18]. In [17], Katz and Yung proposed a scalable compiler which transforms a secure unauthenticated GKA protocol into a secure authenticated GKA protocol. Employing that technique, they propose a modified Burmester-Desmedt protocol to achieve active security by adding one round. By exploiting a technique similar to the one in [17], Choi *et al.* [9] proposed a GKA protocol which is a bilinear version of the Burmester-Desmedt protocol in the context of identity-based public-key cryptosystems [22]. More recently, Dutta and Barua [12] presented a two-round authenticated GKA protocol which may be viewed as a variant of the Burmester-Desmedt [8] protocol with considerably better efficiency.

2. Preliminaries

2.1. Bilinear Maps

We review some basic notions of bilinear maps [6, 26] underlying our proposal. Let $\mathbb{G}_1, \hat{\mathbb{G}}_1$ and $\mathbb{G}_2$ be three multiplicative groups of the same order q. $\mathbb{G}_1$ and $\hat{\mathbb{G}}_1$ are generated by g and $\hat{g}$ respectively. A map $\hat{e} : \mathbb{G}_1 \times \hat{\mathbb{G}}_1 \longrightarrow \mathbb{G}_2$ is called a bilinear map if it satisfies the following properties:

1. Bilinearity: $\hat{e}(g^a, \hat{g}^b) = \hat{e}(g, \hat{g})^{ab}$ for $a, b \in \mathbb{Z}_q$.
2. Non-degeneracy: There exist $u \in \mathbb{G}_1, v \in \hat{\mathbb{G}}_1$ such that $\hat{e}(u, v) \neq 1$.
3. Computability: There exists an efficient algorithm to compute $\hat{e}(u, v)$ for any $u \in \mathbb{G}_1, v \in \hat{\mathbb{G}}_1$.

It is called symmetric bilinear map if $\mathbb{G}_1 = \hat{\mathbb{G}}_1$ and $g = \hat{g}$. For easy of presentation, in this paper we will use the symmetric bilinear map. However, our systems can be also designed by using asymmetric bilinear map.

A bilinear map instance generator is defined as a probabilistic polynomial-time algorithm $\mathcal{IG}(\ell)$ that takes as input a security parameter ℓ and returns a uniformly random tuple $(\mathbb{G}_1, \mathbb{G}_2, \hat{e}, g, q)$ of bilinear parameters, where $\mathbb{G}_1$ and $\mathbb{G}_2$ are two multiplicative groups of order q, $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \longrightarrow \mathbb{G}_2$ is a bilinear map, and g is a generator of $\mathbb{G}_1$.

2.2. Computational Assumptions

The security of our protocol is based on the computational assumptions that the following well-studied problems are hard:

Computational Diffie-Hellman (CDH) Problem in G_1: Given (g^α, g^β) for unknown $\alpha, \beta \in \mathbb{Z}_q$, compute $g^{\alpha\beta}$.

The CDH assumption is that there is no polynomial-time algorithm that can solve the CDH problem with non-negligible probability.

Divisible Computational Diffie-Hellman (DCDH) Problem in G_1: Given (g^α, g^β) for unknown $\alpha, \beta \in \mathbb{Z}_q$, compute $g^{\alpha/\beta}$.

The DCDH assumption is that there is no polynomial-time algorithm that can solve the DCDH problem with non-negligible probability. The DCDH problem was proven equivalent to the CDH problem [3].

k-**Bilinear Diffie-Hellman Exponent (BDHE) Problem** [6]: Given g, h, and $y_i = g^{\alpha^i}$ in $\mathbb{G}_1$ for $i = 1, 2, ..., k, k+2, ..., 2k$ as input, compute $\hat{e}(g, h)^{\alpha^{k+1}}$. Since the input vector lacks the term $g^{\alpha^{k+1}}$, the bilinear map does not seem to help to compute $e(g, h)^{\alpha^{k+1}}$.

The k-BDHE assumption is that there is no polynomial-time algorithm that can solve the k-BDHE problem with non-negligible probability.

2.3. Desirable Attributes

Some desirable security attributes of key agreement protocols are identified in [5, 8, 11, 20, 23], *i.e.*, *known-key security*, *unknown key-share*, *key-compromise impersonation*, *perfect forward security* and *key control*. We emphasize that the definitions of those attributes are for evaluating the security of conventional key agreement protocols. In order to evaluate the security of ASGKA protocols, we accordingly re-define those attributes as follows:

1. *Known-key security*: Each run of the protocol should establish a unique group decryption key for each participant and a unique common group encryption key. The leakage of the group decryption key(s) of one session should not compromise the group decryption key(s) of other sessions.

2. *Unknown key-share*: An active adversary must not be able to make one participant believe that the key is shared with one party when it is in fact shared with another party.
3. *Key-compromise impersonation*: The compromise of A's private key will allow an adversary to impersonate A, but it should not enable the adversary to impersonate other entities in the presence of A.
4. *Perfect forward security*: If the private keys of all the participants are compromised, the secrecy of previously established group decryption keys should not be affected. A weaker notion is *partial forward security*: the compromise of the private keys of some participants does not jeopardize the secrecy of previously established group decryption keys.
5. *Key control*: None of the participants is able to force the group encryption or decryption key to a preselected value.

3. Authenticated Asymmetric Group Key Agreement Protocols

In this section, we propose authenticated ASGKA protocols motivated by [25].

3.1. The Basic Protocol

We begin with a basic authenticated ASGKA protocol. In the protocol, each participant is armed with a private-public key pair for authentication purposes; and a trusted authority CA is employed to issue certificates for the participants. Our protocol can be described in terms of the following seven stages.

[System Setup]

CA generates the system-wide parameters at this stage. It runs $\mathcal{IG}(\ell)$ to generate a uniformly random tuple $(\mathbb{G}_1, \mathbb{G}_2, \hat{e}, g, q)$ of a bilinear map instance, and selects $g_1, ..., g_\omega \in \mathbb{G}_1$, where ω is the largest group size that the system can support. Finally, CA publishes the system parameters $\Psi = (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, g, q, g_1, ..., g_\omega)$.

[Certification]

At this stage, each participant generates her private-public key pair and requests a certificate for her public key. The concrete steps come as follows:

1. A user $\mathcal{U}_i$ generates her private-public key pair (s_i, p_i), and sends her information to CA, which includes her public key p_i and any necessary additional identifying information, such as her name and evidences that she is the owner of p_i.

2. CA verifies $\mathcal{U}_i$'s information. If it is valid, CA issues and sends a certificate $cert_i$ to $\mathcal{U}_i$.

[Key Establishment]

At this stage, a set of participants generate and publish the messages which will be used to generate the group encryption and decryption keys. Without loss of generality, let the participants be $\mathcal{U}_1, ..., \mathcal{U}_n, n \leq \omega$. A participant $\mathcal{U}_i$ holding private key s_i performs the following steps:

1. Randomly choose $h_i \in \mathbb{G}_1, r_i \in \mathbb{Z}_q$ and compute $x_i = g^{-r_i}, A_i = \hat{e}(h_i, g)$.
2. For $1 \leq j \leq n$, compute $\sigma_{i,j} = h_i g_j^{r_i}$.
3. Using the private key s_i, generate a signature ρ_i on $\mathbb{M}_i = \{\sigma_{i,1}, ..., \sigma_{i,i-1}, null, \sigma_{i,i+1}, ..., \sigma_{i,n}, x_i, A_i, cert_i\}$.
4. Publish $(\sigma_{i,1}, ..., \sigma_{i,i-1}, null, \sigma_{i,i+1}, ..., \sigma_{i,n}, (x_i, A_i, \rho_i, cert_i))$.

When this stage is completed, each participant can get the messages as shown in Table 1, where $\underline{\sigma_{i,i}} = \sigma_{i,i} = h_i g_i^{r_i}$ is only known to $\mathcal{U}_i$ and will not be published.

Table 1: Message retrieval by participants

Required for	$\mathcal{U}_1$	$\mathcal{U}_2$	$\mathcal{U}_3$	$\cdots$	$\mathcal{U}_n$	All
$\mathcal{U}_1 \Rightarrow$	$\underline{\sigma_{1,1}}$	$\sigma_{1,2}$	$\sigma_{1,3}$	$\cdots$	$\sigma_{1,n}$	$(x_1, A_1, \rho_1, cert_1)$
$\mathcal{U}_2 \Rightarrow$	$\sigma_{2,1}$	$\underline{\sigma_{2,2}}$	$\sigma_{2,3}$	$\cdots$	$\sigma_{2,n}$	$(x_2, A_2, \rho_2, cert_2)$
$\mathcal{U}_3 \Rightarrow$	$\sigma_{3,1}$	$\sigma_{3,2}$	$\underline{\sigma_{3,3}}$	$\cdots$	$\sigma_{3,n}$	$(x_3, A_3, \rho_3, cert_3)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
$\mathcal{U}_n \Rightarrow$	$\sigma_{n,1}$	$\sigma_{n,2}$	$\sigma_{n,3}$	$\cdots$	$\underline{\sigma_{n,n}}$	$(x_n, A_n, \rho_n, cert_n)$
Key	d_1	d_2	d_3	$\cdots$	d_n	(x, A)

[Encryption Key Derivation]

Any user can compute the group encryption key (x, A), where

$$x = \prod_{i=1}^{n} x_i, A = \prod_{i=1}^{n} A_i.$$

The group encryption key (x, A) is accepted if the signatures $\rho_1, ..., \rho_n$ are all valid.

[Decryption Key Derivation]

To get the group decryption key, $\mathcal{U}_i$ computes $d_i = \prod_{l=1}^{n} \sigma_{l,i}$ and accepts d_i if the signatures $\rho_1, ..., \rho_n$ are valid.

[Encryption]

Knowing the public parameters Ψ and the group encryption key (x, A), anyone can encrypt any message $m \in \mathbb{G}_2$ by performing the following steps:

1. Select a random $t \in \mathbb{Z}_q$.
2. Compute $c_1 = g^t, c_2 = x^t, c_3 = m \cdot A^t$.
3. Output the ciphertext $c = (c_1, c_2, c_3)$.

[Decryption]

Each participant $\mathcal{U}_i$ can decrypt

$$m = \frac{c_3}{e(d_i, c_1)e(g_i, c_2)}.$$

The correctness of the decryption procedure follows from a direct verification.

3.2. Security Analysis

In this section, we show that above ASGKA protocol is equipped with all the security attributes defined in Section 2.3.

1. *Known-key security*: In each run of the protocol, h_i and r_i are randomly selected by each participant $\mathcal{U}_i$. As a result, the group encryption key and the group decryption key for one session are distributed uniformly and independently of the group encryption and decryption keys for other sessions. Therefore, leakage of the group decryption key(s) for one session should not compromise the group decryption key(s) for other sessions. Known-key security follows.
2. *Unknown key-share*: In our protocol, each participant $\mathcal{U}_i$ should generate a signature ρ_i on $\mathbb{M}_i$. Therefore, any other entities can verify that ρ_i and $\mathbb{M}_i$ are truly generated by $\mathcal{U}_i$. This implies that, in a successful run of our protocol, all participants must be the real participants as claimed and no participant can be impersonated. The negotiated key must be shared by the true parties. In addition, the ASGKA protocol in [25] is proven secure against passive adversaries, which means that the decryption keys are only known by the corresponding participants and the negotiated public common encryption key is unique. Hence, even

an active adversary cannot make one participant believe that the key is shared with one party when it is in fact shared with another party. The unknown key-share property holds, thereby achieving security against man-in-the-middle attacks.

3. *Key-compromise impersonation*: Due to the unforgeability of signatures, without knowing the private key of $\mathcal{U}_j$, the adversary cannot generate a valid signature on behalf of $\mathcal{U}_j$. If $\mathcal{U}_i$'s private key is compromised by an adversary, the adversary cannot impersonate another participant $\mathcal{U}_j$ with the private key of $\mathcal{U}_i$. This implies the key-compromise impersonation property.
4. *Perfect forward security*: As shown in [25], to extract σ_{ii} without knowledge of the random values h_i and r_i, $\mathcal{U}_i$ should break the k-BDHE assumption. Hence, even if an adversary knows the private keys of all participants, the adversary cannot recover σ_{ii}. Since to compute the previously established group decryption keys, the adversary needs input σ_{ii}, the adversary cannot derive the previous decryption keys and the protocol enjoys perfect forward security.
5. *Key control*: In our protocol, all participants have an input. It is easy to see that only the participant who is last to publish her message may have the chance to control the keys. Without loss of generality, we assume that this participant is $\mathcal{U}_n$. However, $\mathcal{U}_n$ cannot force a group encryption key to be a preselected one (x', A') either. This is because, knowing (x', A') and $(\prod_{i=1}^{n-1} x_i, \prod_{i=1}^{n-1} A_i)$, to compute (x_n, A_n), she should solve the DCDH problem. As to a group decryption key, not even $\mathcal{U}_n$ can preselect her own one, since this is equivalent to solving the DCDH problem.

3.3. Performance Evaluation

Here we summarize the performance of above authenticated ASGKA protocol and two other authenticated Burmester-Desmedt [8] based protocols in the same cryptosystem setting. In what follows, we consider a group of n participants.

Round Efficiency. To establish a session key, the most efficient existing secure GKA protocols (*e.g.*, [12, 17]) require two or more rounds. Therefore, in these protocols, all participants should be connected concurrently. In our protocol, however, each participant only needs to transmit the message once. Hence, like the protocol in [25], our protocol still requires one round,

Table 2: Comparison of protocols in terms of computational complexity

	Exponentiation	Multiplication	Verification	Rounds
Protocol in [12]	3	$2n-2$	$n+1$	2
Protocol in [17]	3	$\frac{n^2}{2}+\frac{3n}{2}-3$	$2n-2$	3
Our Protocol	0	$2n^{\dagger}, 2n^{\ddagger,\S}$	$n^{\dagger}, n-1^{\ddagger}$	1

†: For a sender. ‡: For a participant. §: With pre-computation.

although we achieve stronger security, *i.e.*, against active attacks. The one-round feature in our protocol implies that each participant can simply send and then leave. This is very desirable in practice.

Computational Overhead. In our protocol, for a sender to generate the group encryption key, she only needs to verify n signatures and compute $2n$ group multiplication operations. For a participant, she needs to verify $n-1$ signatures, and when the pre-computation is considered, she only needs to compute $2n$ group multiplication operations. From Table 2, it is easy to see that our protocol has much lower computational overhead than the protocol in [17] and has a comparable overhead with the protocol in [12]. One may notice that the protocols in [12, 17] only support broadcast from within the group. However, our protocol allows anyone to broadcast secret messages to the group.

3.4. Simulation

In this section, we carry out several simulations to determine the practical performance of above protocol. In the simulations, we use the Pairing-Based Cryptography (PBC) library [2] run on a laptop using the Intel®Core™2 Duo P8800 chip at a frequency of 2.67 GHz; the security parameter ℓ is set to be 160; the length of a group element in $\mathbb{G}_1$ and $\mathbb{G}_2$ are 171 bits and 1024 bits respectively; the number of protocol participants is chosen from 3 to 100; the elliptic curve version of Schnorr's signature scheme is selected to secure the protocol, and the length of a Schnorr's signature is 342 bits; the certificates used are of size 121 bytes and signed by Schnorr's signature scheme. The computations which can be pre-computed are omitted in the simulations, *e.g.*, the computations at Steps 1 and 2 in Key Establishment stage.

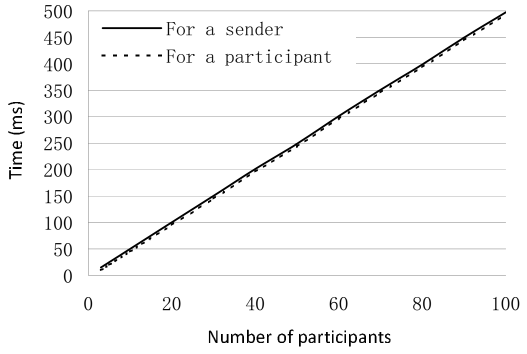

Figure 1: Average running time of Encryption Key Derivation

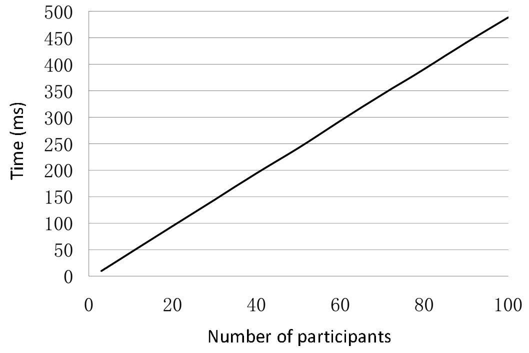

Figure 2: Average running time of Decryption Key Derivation

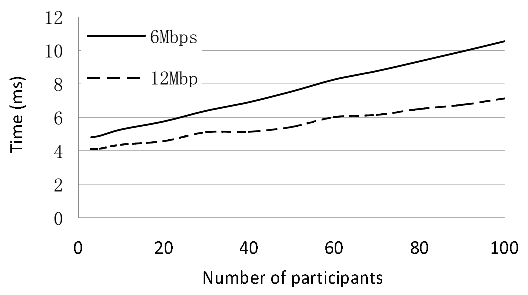

Figure 3: End-to-end delay in the wireless network environment

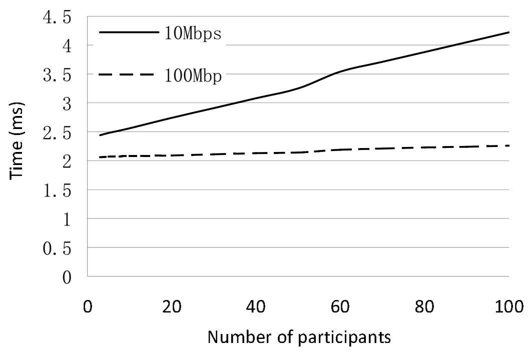

Figure 4: End-to-end delay in the wired network environment

Figure 1 shows the relationship between the number of participants and the running time to generate a group encryption key for a sender and a participant. From this figure, one can see that the time costs for a sender and a participant to generate a group encryption key are almost the same and grow linearly as the number of participants grows. However, the time costs are still not high when the number of participants is 100. They are 497.29 ms and 492.4 ms for a sender and for a participant, respectively. Figure 2 shows the relationship between the number of participants and the running time to generate a group decryption key. From this figure, one can see that the time cost for a participant to generate a group decryption key also grows linearly with the number of participants. This time cost ranges from 9.8 ms to 488.18 ms. The time costs to generate a ciphertext and decrypt a ciphertext are constant regardless the number of protocol participants. They are 6 ms and 36 ms, respectively.

By using NS-3 [1], we carry out the simulations to evaluate the end-to-end delay[1] of our protocol. In the simulations, we evaluate our protocol in wireless and wired network environments[2]. In the wireless network environment, we do the simulation using the IEEE 802.11p protocol, which is common in vehicular *ad hoc* networks (VANETs) for vehicle-to-vehicle communication [24, 28]; ASGKA protocols can be used in VANETs to provide location privacy for the on-the-fly groups of vehicles. The channel bandwidth bounds used are 6 Mbps and 12 Mbps. The result of this simulation is shown in Figure 3, where the relationship between the number of participants and the end-to-end delay is depicted. The end-to-end delay grows linearly with the number of protocol participants. The delays range from 4.81 ms to 10.54 ms and from 4.1 ms to 7.13 ms for channel bandwidth bounds 6 Mbps and 12 Mbps, respectively. The simulation implies that our protocol is practical for VANETs and other *ad hoc* networks. In the wired network environment, the channel bandwidth bounds are 10 Mbps and 100 Mbps. The result of the simulation is shown in Figure 4. The end-to-end delay also grows linearly with the number of protocol participants. When the channel bandwidth bound is 10 Mbps, the delay ranges from 2.44 ms to 4.22 ms. When the channel bandwidth bound is 100 Mbps, the number of participants does not affect the delay a lot: the delay ranges from 2.06 ms to 2.26 ms.

3.5. Trade-Off Between Communication and Ciphertext Size

In above basic protocol, for a group of n participants, each participant should publish $\mathcal{O}(n)$ group elements during key establishment. When the group size is very large, the transmission overhead is very heavy. To reduce the transmission overhead, we can break the group into several smaller subgroups. For instance, we can set the subgroup size to be $\sqrt{n}$. The group is thus separated into $\sqrt{n}$ subgroups. Then the participants in each subgroup run the above protocol to generate their subgroup encryption and decryption keys. Finally, $\sqrt{n}$ subgroup encryption keys will be generated and each participant will have a subgroup decryption key. To send an encrypted message to all the group users, a sender separately encrypts for each subgroup and

[1]It refers to the time taken for ρ_i to be generated and the packet $(\mathbb{M}_i, \rho_i)$ to be transmitted across a network in the Key Establishment stage.

[2] In the wired network, the Bus (point-to-multipoint) topology can be chosen, and, in the wireless network, the Independent Basic Service Set (IBSS) topology can be chosen which enables each wireless device to connect to any other wireless device within range.

generates the final ciphertext by concatenating all of the underlying individual ones. In this way, each participant only needs to transmit $\mathcal{O}(\sqrt{n})$ group elements to enable a sender to broadcast a message to n group users, at the cost of the sender sending a ciphertext of $\mathcal{O}(\sqrt{n})$ group elements during encryption.

We note that the above technique is similar to that in [7]. However, the technique in [7] is focused on a broadcast encryption scheme and usually not suited for GKA protocols. In a conventional GKA protocol, if we separate a group into several subgroups and the participants in each subgroup run the conventional GKA protocol, then all the subgroups will have different session keys and only the participants in the same subgroup will be able to communicate with each other. However, in an ASGKA protocol, anyone who knows the (sub)group encryption key can send a encrypted message which can be decrypted by the participants in this (sub)group. Hence, adapting a technique similar to that in [7] to ASGKA as we do in this paper is interesting in its own right.

Figure 5 shows the simulation result of the relationship between the number of participants and the running time to generate a group encryption key for a sender and a participant after trade-off. In the simulation, n is chosen from 9 to 400. If n is not a square number, the group is separated into $\lceil\sqrt{n}\rfloor$ subgroups, where $\lceil\sqrt{n}\rfloor$ is the closest integer of $\sqrt{n}$. Except one subgroup, the number of participants in other subgroups are set to be $\lfloor\sqrt{n}\rfloor$, where $\lfloor\sqrt{n}\rfloor$ is the smallest integer close to $\sqrt{n}$. With the trade-off, the running time to generate a group encryption key is slight lower that of our basic protocol. Therefore, the computational overhead for a user to generate a group encryption key cannot be lowered down a lot by the trade-off. As to the running time for a participant to generate a group decryption key, it is much lower than that of the basic protocol. For simplicity, in our simulation, if n is not a square number, we only consider the running time of a participant in the subgroups with $\lfloor\sqrt{n}\rfloor$ participants. Figure 6 shows the simulation result of the relationship between the number of participants and the running time for a participant to generate a group decryption key. The running time is about $\frac{1}{\lfloor\sqrt{n}\rfloor}$ of that in the basic protocol. As to the time cost to generate a ciphertext, it's related to the number of subgroups. It's about $6\lceil\sqrt{n}\rfloor$ ms. The time cost to decrypt a ciphertext is the same as that in our basic protocol. It's about 36 ms, which is constant.

Here we will not test the end-to-end delay of our protocol in various

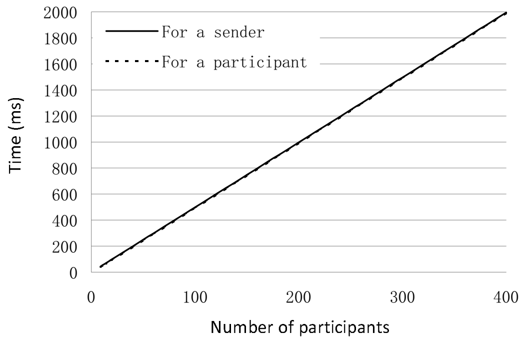

Figure 5: Average running time of Encryption Key Derivation after trade-off

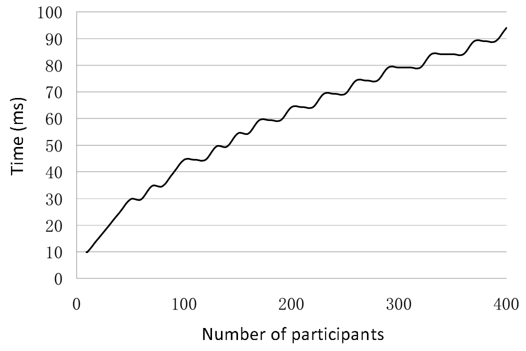

Figure 6: Average running time of Decryption Key Derivation after trade-off

network environments. The end-to-end delay of our protocol is determined by the number of participants in a subgroup. Therefore, we will have the similar results as those shown in Figures 3 and 4, where the number of participants is replaced by the number of participants in the subgroup.

From above results, one may find that with the trade-off, the running time for a participant to generate her group decryption key and the end-to-end delay of our protocol are lowered down, and, the time cost to generated a group encryption key is slightly affected, and, the time cost to decrypt a ciphertext is constant and the same as that of the basic protocol. However, the time cost to generate a ciphertext is much higher that the basic protocol. Since Encryption Key Derivation and Decryption Key Derivation only need to be run once, for the most cases, they will not affect the whole efficiency of the protocol. However, to securely send a message to a group, the time cost to generate a ciphertext is increased to $6\lceil\sqrt{n}\rfloor$ ms, which is increased by $\lceil\sqrt{n}\rfloor$ times. Therefore, according to above analysis, it's not worth using the above trade-off when the bandwidth is not a problem or the number of total participants is small (*e.g.*, < 100).

4. Extensions to Broadcast Encryption

Broadcast encryption is a cryptographic primitive in which the encrypted message can be decrypted only by qualified users. In this section, based on our authenticated ASGKA protocol, we propose novel broadcast encryption systems which can be viewed as a dynamic ASGKA protocols. Our systems are especially suited for groups whose users do not change frequently.

4.1. The Basic Broadcast Encryption System

There are three main roles in our system: CA, billboard maintainer and system users.

- CA is a trusted authority whose main responsibility is to issue certificates for the billboard maintainer and system users.
- The billboard maintainer (a special user in the system) has a private-public key pair and maintains a billboard which has two parts. The first part of the billboard is initialized by the billboard maintainer (or other parties) and the second part of the billboard is set to be empty at the beginning.

- Each user in our system has a private-public key pair and publishes a message on the billboard second part.

Next we begin to describe our system, which consists of the following nine stages.

[System Setup]

CA runs $\mathcal{IG}(\ell)$ to generate a uniformly random tuple $(\mathbb{G}_1, \mathbb{G}_2, \hat{e}, g, q)$ of a bilinear map instance, selects $g_1, ..., g_\omega \in \mathbb{G}_1$ and a symmetric key encryption scheme $\mathscr{E}_K(.)/\mathscr{D}_K(.)$ (*e.g.*, AES), where ω is the largest group size that the system can support and K is a key which specifies the particular transformation of plaintext into ciphertext during encryption, or vice versa during decryption. Finally, CA publishes the system-wide parameters $\Psi = (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, g, q, g_1, ..., g_\omega, \mathscr{E}_K(.)/\mathscr{D}_K(.))$.

[Certification]

At this stage, $\mathcal{U}_i$ (the billboard maintainer or a user of the broadcast system) selects a private-public key pair and submits her information to CA to request a certificate. This stage consists of the following two steps:

1. $\mathcal{U}_i$ first chooses her private-public key pair (s_i, p_i), then sends her information to CA, which includes her public key p_i and any necessary additional identifying information.
2. CA verifies $\mathcal{U}_i$'s information. If valid, CA generates and sends a certificate $cert_i$ to $\mathcal{U}_i$.

[Billboard Initialization]

Assume that the broadcast encryption system is of size $n \leq \omega$. Let the billboard maintainer be $\mathcal{U}_0$ holding private key s_0 and certificate $cert_0$. $\mathcal{U}_0$ initializes the billboard part I (Table 3) as follows:

1. For $1 \leq i \leq n$, randomly choose $h_i^0 \in \mathbb{G}_1, r_i^0 \in \mathbb{Z}_q$ and compute $x_i^0 = g^{-r_i^0}, A_i^0 = \hat{e}(h_i^0, g)$.
2. For $1 \leq i, j \leq n$, compute $\sigma_{i,j}^0 = h_i^0 g_j^{r_i^0}$.
3. Set $\mathbb{M}_i = \{\sigma_{i,1}^0, ..., \sigma_{i,i-1}^0, null, \sigma_{i,i+1}^0, ..., \sigma_{i,n}^0, x_i^0, A_i^0, cert_0\}$ and, with s_0, generate a signature ρ_i^0 on $\mathbb{M}_i$.
4. Publish $\mathbb{L}_i = \{\sigma_{i,1}^0, ..., \sigma_{i,i-1}^0, null, \sigma_{i,i+1}^0, ..., \sigma_{i,n}^0, (x_i^0, A_i^0, \rho_i^0, cert_0)\}$.

For the billboard part II, as shown in Table 4, $\mathcal{U}_0$ sets each row to be $\mathbb{L}'_i = \{\overbrace{null, null, ..., null}^{n+1}\}$.

Table 3: Billboard Part I

Column	1	2	3	$\cdots$	n	$n+1$
$\mathbb{L}_1$	$null$	$\sigma^0_{1,2}$	$\sigma^0_{1,3}$	$\cdots$	$\sigma^0_{1,n}$	$(x^0_1, A^0_1, \rho^0_1, cert_0)$
$\mathbb{L}_2$	$\sigma^0_{2,1}$	$null$	$\sigma^0_{2,3}$	$\cdots$	$\sigma^0_{2,n}$	$(x^0_2, A^0_2, \rho^0_2, cert_0)$
$\mathbb{L}_3$	$\sigma^0_{3,1}$	$\sigma^0_{3,2}$	$null$	$\cdots$	$\sigma^0_{3,n}$	$(x^0_3, A^0_3, \rho^0_3, cert_0)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
$\mathbb{L}_n$	$\sigma^0_{n,1}$	$\sigma^0_{n,2}$	$\sigma^0_{n,3}$	$\cdots$	$null$	$(x^0_n, A^0_n, \rho^0_n, cert_0)$

Table 4: Billboard Part II

Column	1	2	3	$\cdots$	n	$n+1$
$\mathbb{L}'_1$	$null$	$null$	$null$	$\cdots$	$null$	$null$
$\mathbb{L}'_2$	$null$	$null$	$null$	$\cdots$	$null$	$null$
$\mathbb{L}'_3$	$null$	$null$	$null$	$\cdots$	$null$	$null$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
$\mathbb{L}'_n$	$null$	$null$	$null$	$\cdots$	$null$	$null$

[Joining]

When a user $\mathcal{U}_i$ wants to join the broadcast system, she first checks whether there is a row with all *null* elements on billboard part II. If there is no such row, it means the broadcast system is full; otherwise, assuming that the index of that row is l, she does the following:

1. Randomly choose $h_l \in \mathbb{G}_1, r_l \in \mathbb{Z}_q$ and compute $x_l = g^{-r_l}, A_l = \hat{e}(h_l, g)$.
2. For $1 \leq j \leq n$, compute $\sigma_{l,j} = h_l g_j^{r_l}$.
3. Using the private key s_i, generate a signature ρ_l on $\mathbb{M}'_l = \{\sigma_{l,1}, ..., \sigma_{l,l-1}, null, \sigma_{l,l+1}, ..., \sigma_{l,n}, x_l, A_l, cert_i\}$.
4. Submit $\{\sigma_{l,1}, ..., \sigma_{l,l-1}, null, \sigma_{l,l+1}, ..., \sigma_{l,n}, (x_l, A_l, \rho_l, cert_l)\}$ to the billboard maintainer, where $cert_l = cert_i$.

When the billboard maintainer receives $\{\sigma_{l,1}, ..., \sigma_{l,l-1}, null, \sigma_{l,l+1}, ..., \sigma_{l,n}, (x_l, A_l, \rho_l, cert_l)\}$, she first checks whether ρ_l and $cert_l$ are valid; if they are valid, the billboard maintainer sets $\mathbb{L}'_l = \{\sigma_{l,1}, ..., \sigma_{l,l-1}, null, \sigma_{l,l+1}, ..., \sigma_{l,n}, (x_l, A_l, \rho_l, cert_l)\}$; otherwise, she drops that message.

[Leaving]

Suppose that a user $\mathcal{U}_i$ leaves the system and her published message is in the l-th row; the billboard maintainer sets $\mathbb{L}'_i = \{\overbrace{null, null, ..., null}^{n+1}\}$.

[Group Encryption Key Derivation]

Let $\mathbb{S}$ be the set of indices such that $\mathbb{L}'_i = \{\overbrace{null, null, ..., null}^{n+1}\}$ and $\bar{\mathbb{S}}$ be the set of indices such that $\mathbb{L}'_i \neq \{\overbrace{null, null, ..., null}^{n+1}\}$ on the billboard part II, respectively. Any user can compute the group encryption key (x, A), where

$$x = \prod_{j\in\mathbb{S}} x_j^0 \prod_{j\in\bar{\mathbb{S}}} x_j, A = \prod_{j\in\mathbb{S}} A_j^0 \prod_{j\in\bar{\mathbb{S}}} A_j.$$

The user accepts (x, A), if the certificates and signatures in $\{\mathbb{L}_i\}_{i\in\mathbb{S}}$ and $\{\mathbb{L}'_i\}_{i\in\bar{\mathbb{S}}}$ are all valid.

[Group Decryption Key Derivation]

To derive the group decryption key, the user with index l computes $d_l = \prod_{i\in\mathbb{S}} \sigma_{i,l}^0 \prod_{i\in\bar{\mathbb{S}}} \sigma_{i,l}$ and accepts this group decryption key if the certificates and signatures in $\{\mathbb{L}_i\}_{i\in\mathbb{S}}$ and $\{\mathbb{L}'_i\}_{i\in\bar{\mathbb{S}}}$ are all valid.

[Encryption]

Knowing the public parameters Ψ and the group encryption key (x, A), any one can encrypt any plaintext m as follows:

1. Select a random $t \in \mathbb{Z}_q, \kappa \in \mathbb{G}_2$.
2. Compute $c_1 = g^t, c_2 = x^t, c_3 = \kappa \cdot A^t, c_4 = \mathscr{E}_\kappa(m)$.
3. Output the ciphertext $c = (c_1, c_2, c_3, c_4)$

[Decryption]

To decrypt the ciphertext $c = (c_1, c_2, c_3, c_4)$, a user $\mathcal{U}_i$ in the system does the following:

1. Extract $\kappa = \frac{c_3}{e(d_i, c_1)e(g_i, c_2)}$.
2. Output the plaintext $m = \mathscr{D}_\kappa(c_4)$.

To improve the efficiency of the above system, the billboard maintainer can verify the signatures on the billboard for other users and then generate an additional signature ϱ on the message-signatures on the whole billboard. By using this approach, to obtain the group encryption or decryption key, an entity does not need to verify the signatures generated by the system users; it just needs to verify whether ϱ is a valid signature by the billboard maintainer.

4.2. Evaluation

This section evaluates our broadcast encryption system. In addition to the advantages similar to those of our ASGKA protocol, we show that our system has low storage overhead, and enjoys the perfect forward security property.

Storage Overhead. In our system, a sender or a user do not need to store all the information on the billboard. As an example, for the basic system a sender only needs to store the $n+1$-th column on billboard parts I and II. For the l-th user in the system, she needs to store the information of the l-th and $n+1$-th columns on billboard parts I and II, respectively.

Security. A secure broadcast encryption system requires that only the group users can decrypt the ciphertexts. This property is guaranteed by our ASGKA protocol. In addition to the basic security requirement, our broadcast system has also perfect forward security, which can be proven similarly as for the authenticated ASGKA protocol. Forward security implies that, if some users' decryption keys are occasionally compromised, the secrecy of messages encrypted under previously established group encryption keys can still be guaranteed. This is very useful but rarely achieved in existing broadcast systems.

4.3. Selecting Receivers for Broadcast Encryption

In the above basic broadcast encryption system, all the users involved in the system can decrypt the ciphertexts of the group. However, sometimes, this property is not desired by the senders. For instance, a sender may not want the l-th user to decrypt the ciphertext. To exclude the l-th user from the sender's broadcast list, the sender only needs to compute the group encryption key (x, A) as

$$x = \prod_{j \in \{\mathbb{S}, l\}} x_j^0 \prod_{j \in \bar{\mathbb{S}}, j \neq l} x_j, A = \prod_{j \in \{\mathbb{S}, l\}} A_j^0 \prod_{j \in \bar{\mathbb{S}}, j \neq l} A_j.$$

Then the sender encrypts the plaintext under this group encryption key to generate the ciphertext $c = (c_1, c_2, c_3, c_4)$ as in the $\mathsf{Encryption}$ stage. Finally, the sender sends (l, c) to the group, which means that the l-th user does not have the right to decrypt the message. To decrypt the above ciphertext, an entitled user with index i first computes her temporary decryption key as

$$d_i = \prod_{j \in \{\mathbb{S}, l\}} \sigma^0_{j,i} \prod_{j \in \bar{\mathbb{S}}, j \neq l} \sigma_{j,i},$$

and then decrypts the ciphertext as in the Decryption stage. To exclude more users from the sender's broadcast list, the method is similar.

In our basic broadcast system, if the messages on billboard part I are generated by the billboard maintainer and the billboard part II is not full, then the maintainer always has the ability to decrypt the group messages. This property is required in some scenarios (*e.g.*, the head of a company wishes to monitor messages received by his employees during the working time or the public security department needs to read communications in order to prevent organized crime or terrorist attacks). If this property is not desirable in other scenarios, the messages on billboard part I can be generated by several parties using a (t, n) threshold scheme [21]. In this way, to decrypt the message, at least t out of n parties are required to collaborate.

4.4. An Extended System

We notice that, usually, the scale of a broadcast encryption system is hard to anticipate. Therefore, if the basic system is chosen, we should set n to be large. However, in this case, similarly to our authenticated ASGKA protocol, each user should publish $\mathcal{O}(n)$ group elements. If n is too large, the transmission overhead of our system will be very high. In this section, we propose an extended system to trade off the transmission overhead and the ciphertext size. Our method is similar to that in Section 3.5. In the following, we propose our new system. However, we do not consider the case of selecting receivers: such a selection can be achieved by using a method similar to that in Section 4.3.

The new system also has nine stages as our basic system. The first 2 stages, System Setup and Certification, are the same as those in our basic system. As to the Billboard Initialization stage, we also use a billboard which has two parts. Before initializing the billboard, the billboard maintainer $\mathcal{U}_0$ first chooses a suitable $n \leq \omega$, then initializes the billboard part I like in Section 4.1. As to the billboard part II, it dynamically contains several tables like Table 4, each row of which is initially set to $\mathbb{L}'_i = \{\overbrace{null, null, ..., null}^{n+1}\}$.

Each table corresponds to a subgroup. The Joining and Leaving stages are similar to those in our basic system. Hence, we omit the details. The Group Encryption Key Derivation stage is slightly different from that in our basic system. Instead of a single group encryption key, in this stage we have several subgroup encryption keys. The detailed description of this stage follows:

[Group Encryption Key Derivation]

Let $\mathbb{S}$ be the set of indices such that $\mathbb{L}'_i = \{\overbrace{null, null, ..., null}^{n+1}\}$ and $\bar{\mathbb{S}}$ be the set of indices such that $\mathbb{L}'_i \neq \{\overbrace{null, null, ..., null}^{n+1}\}$ on the τ-th table of billboard part II, respectively. Any user can compute the subgroup encryption key (x_τ, A_τ) in a way parallel to that in our basic system. In the end, we have several subgroup encryption keys $(x_1, A_1), (x_2, A_2), ...$.

The Group Decryption Key Derivation stage is similar to that in our basic system. Hence, we omit the description here. As to the Encryption stage, a sender will encrypt a plaintext m under all the subgroup encryption keys $(x_1, A_1), (x_2, A_2), ...$ to generate $c^1, c^2, ...$, where c^1 is a ciphertext as generated in our basic system. Finally, the full ciphertext is the concatenation of all the underlying individual ciphertexts $c^1||c^2||...$. In the Decryption stage, a user in the τ-th subgroup first extracts c^τ from $c^1||c^2||...$, then uses her decryption key to decrypt the ciphertext as in our basic system.

5. Conclusion

We have proposed a one-round authenticated ASGKA protocol which captures all the desirable security attributes of key agreement protocols. Evaluation shows that our protocol has a low computational overhead and optimal round efficiency. Based on our authenticated ASGKA protocol, a broadcast encryption system was proposed as well. Our system achieves perfect forward security and has short ciphertexts.

Acknowledgments and disclaimer

The help of Michal Sramka in proofreading this manuscript is gratefully acknowledged. This work is partly supported by the Nature Science Foundation of China through projects 61021004, 60970114, 60970115, 60970116 and 61003214, by the Spanish Government under projects TSI2007-65406-C03-01 “E-AEGIS”, TIN2009-11689 “RIPUP”, “eVerification” TSI-020100-2009-720, SeCloud TSI-020302-2010-153 and CONSOLIDER INGENIO 2010

CSD2007-00004 “ARES”, by the Fundamental Research Funds for the Central Universities of China through projects 3103004, 78210044, by the Scientific Research Program Funded by Shaanxi Provincial Education Department (Program No. 2010JK727) and by the Shanghai Natural Science Foundation under Grant No. 11ZR1411200. The fourth author is partially supported as an ICREA-Acadèmia researcher by the Catalan Government. The views of the author with the UNESCO Chair in Data Privacy do not necessarily reflect the position of UNESCO nor commit that organization.

References

[1] NS-3 Network Simulator. http://www.nsnam.org/

[2] Pairing-Based Cryptography Library. http://crypto.stanford.edu/pbc/

[3] F. Bao, R. Deng, H. Zhu. Variations of Diffie-Hellman Problem. ICICS 2003, LNCS 2836, pp. 301-312, Springer, Heidelberg, 2003.

[4] S. Berkovits. How to Broadcast a Secret. EUROCRYPT 1991, LNCS 547, pp. 535-541, Springer, Heidelberg, 1991.

[5] S. Blake-Wilson, D. Johnson, A. Menezes. Key Agreement Protocols and Their Security Analysis. Cryptography and Coding, LNCS 1355, pp. 30-45, Springer, Heidelberg, 1997.

[6] D. Boneh, X. Boyen, E. Goh. Hierarchical Identity Based Encryption with Constant Size Ciphertext. EUROCRYPT 2005, LNCS 3494, pp. 440-456, Springer, Heidelberg, 2005.

[7] D. Boneh, C. Gentry, B. Waters. Collusion Resistant Broadcast Encryption with Short Ciphertexts and Private Keys. CRYPTO 2005, LNCS 3621, pp. 258-275, Springer, Heidelberg, 2005.

[8] M. Burmester, Y. Desmedt. A Secure and Efficient Conference Key Distribution System. EUROCRYPT 1994, LNCS 950, pp. 275-286, Springer, Heidelberg, 1995.

[9] K. Choi, J. Hwang, D. Lee. Efficient ID-based Group Key Agreement with Bilinear Maps. PKC 2004, LNCS 2947, pp. 130-144, Springer, Heidelberg, 2004.

[10] W. Diffie, M. Hellman. New Directions in Cryptography. IEEE Transactions on Information Theory 22(6), pp. 644-654, 1976.

[11] W. Diffie, P. Oorschot, M. Wiener. Authentication and Authenticated Key Exchanges. Designs, Codes and Cryptography 2(2), pp. 107-125, 1992.

[12] R. Dutta, R. Barua. Constant Round Dynamic Group Key Agreement. ISC 2005, LNCS 3650, pp. 74-88, Springer, Heidelberg, 2005.

[13] A. Fiat, M. Naor. Broadcast Encryption. CRYPTO 1993, LNCS 773, pp. 480-491, Springer, Heidelberg, 1994.

[14] C. Gentry, B. Waters. Adaptive Security in Broadcast Encryption Systems (with Short Ciphertexts). EUROCRYPT 2009, LNCS 5479, pp. 171-188, Springer, Heidelberg, 2009.

[15] D. Halevy and A. Shamir. The LSD Broadcast Encryption Scheme. CRYPTO 2002, LNCS 2442, pp. 47-60, Springer, Heidelberg, 2002.

[16] A. Joux. A One Round Protocol for Tripartite Diffie-Hellman. Journal of Cryptology 17(4), pp. 263-276, 2004.

[17] J. Katz, M. Yung. Scalable Protocols for Authenticated Group Key Exchange. CRYPTO 2003, LNCS 2729, pp. 110-125, Springer, Heidelberg, 2003.

[18] H. Kim, S. Lee, D. Lee. Constant-Round Authenticated Group Key Exchange for Dynamic Groups. ASIACRYPT 2004, LNCS 3329, pp. 245-259, Springer, Heidelberg, 2004.

[19] M. Liu, X. Yin, E. Ulin-Avila, B. Geng, T. Zentgraf, L. Ju, F. Wang, X. Zhang. A Graphene-Based Broadband Optical Modulator, Nature, 2011. doi:10.1038/nature10067

[20] C. Mitchell, M. Ward, P. Wilson. Key Control in Key Agreement Protocols. Electronic Letters 34(10), pp. 980-981, 1998.

[21] A. Shamir. How to Share a Secret. Communications of the ACM 22(11), pp. 612-613, 1979.

[22] A. Shamir. Identity-Based Cryptosystems and Signature Schemes. CRYPTO 1984, LNCS 196, pp. 47-53, Springer, Heidelberg, 1985.

[23] Y. Yacobi and Z. Shmuely. On Key Distribution Systems. CRYPTO 1989, LNCS 435, pp. 344-355, Springer, Heidelberg, 1990.

[24] Q. Wu, J. Domingo-Ferrer and U. González-Nicolás. Balanced trustworthiness, safety and privacy in vehicle-to-vehicle communications. IEEE Transactions on Vehicular Technology 59(2), pp. 559-573, 2010.

[25] Q. Wu, Y. Mu, W. Susilo, B. Qin, J. Domingo-Ferrer. Asymmetric Group Key Agreement. EUROCRYPT 2009, LNCS 5479, pp. 153-170, Springer, Heidelberg, 2009.

[26] L. Zhang, B. Qin, Q. Wu, F. Zhang. Efficient Many-to-One Authentication with Certificateless Aggregate Signatures. Computer Networks, 54(14), pp.2482-2491, 2010.

[27] L. Zhang, Q. Wu, B. Qin. Authenticated Asymmetric Group Key Agreement Protocol and Its Application. International Communications Conference (ICC 2010), pp. 1-5, IEEE, 2010.

[28] L. Zhang, Q. Wu, A. Solanas, J. Domingo-Ferrer. A Scalable Robust Authentication Protocol for Secure Vehicular Communications. IEEE Transactions on Vehicular Technology 59(4), pp. 1606-1617, 2010.

[29] L. Zhang, F. Zhang, Q. Wu, J. Domingo-Ferrer. Simulatable Certificateless Two-Party Authenticated Key Agreement Protocol. Information Sciences, 180(6), pp. 1020-1030, 2010.