Fast Transmission to Remote Cooperative Groups: A New Key Management Paradigm

Qianhong Wu, *Member, IEEE,* Bo Qin, Lei Zhang,
Josep Domingo-Ferrer, *Fellow, IEEE,* Jesús A. Manjón

***Abstract*—The problem of efficiently and securely broadcasting to a remote cooperative group occurs in many newly emerging networks. A major challenge in devising such systems is to overcome the obstacles of the potentially limited communication from the group to the sender, the unavailability of a fully trusted key generation center and the dynamics of the sender. The existing key management paradigms cannot deal with these challenges effectively. In this paper, we circumvent these obstacles and close this gap by proposing a novel key management paradigm. The new paradigm is a hybrid of traditional broadcast encryption and group key agreement. In such a system, each member maintains a single public/secret key pair. Upon seeing the public keys of the members, a remote sender can securely broadcast to any intended subgroup chosen in an *ad hoc* way. Following this model, we instantiate a scheme which is proven secure in the standard model. Even if all the non-intended members collude, they cannot extract any useful information from the transmitted messages. After the public group encryption key is extracted, both the computation overhead and the communication cost are independent of the group size. Further, our scheme facilitates simple yet efficient member deletion/addition and flexible rekeying strategies. Its strong security against collusion, its constant overhead, and its implementation friendliness without relying on a fully trusted authority render our protocol a very promising solution to many applications.**

***Index Terms*—*Ad hoc* networks, broadcast, cooperative computing, access control, information security, key management.**

I. Introduction

In many newly emerging networks, there is a need to broadcast to remote cooperative groups using encrypted transmission. Examples can be found in access control in remote group communication arising in wireless mesh networks (WMNs), mobile *ad hoc* networks (MANETs), vehicular *ad hoc* networks (VANETs), etc.

WMNs have been recently suggested as a promising low-cost approach to provide last-mile high-speed Internet access. A typical WMN is a multihop hierarchical wireless network [1]. The top layer consists of high-speed wired Internet entry points. The second layer is made up of stationary mesh routers serving as a multi-hop backbone to connect to each other and Internet via long-range high-speed wireless techniques. The bottom layer includes a large number of mobile network users. The end users access the network either by a direct wireless link or through a chain of other peer users leading to a nearby mesh router; the router further connects to remote users through the wireless backbone and Internet. Security and privacy issues are of utmost concern in pushing the success of WMNs for their wide deployment and for supporting service-oriented applications [2]. For instance, a manager on his way to holiday may want to send a confidential email to some staff of her company via WMNs, so that the intended staff members can read the email with their mobile devices (laptops, PDAs, smartphones, etc.). Due to the intrinsically open and distributed nature of WMNs, it is essential to enforce access control of sensitive information to cope with both eavesdroppers and malicious attackers.

A MANET is a system made up of wireless mobile nodes. These nodes have wireless communication and networking characteristics. MANETs have been proposed to serve as an effective networking system facilitating data exchange between mobile devices even without fixed infrastructures. In MANETs, it is important to support group-oriented applications, such as audio/video conference and one-to-many data dissemination in battlefield or disaster rescue scenarios [3]. In general, users working for the same mission form a cooperation domain; any particular application or interest in a network may lead to the establishment of a corresponding community. Since communication in wireless networks is broadcast and a certain amount of devices can receive transmitted messages, the risk of unsecured sensitive information being intercepted by unintended recipients is a real concern [4]. For instance, a commander may issue secret commands to soldiers in battlefield via satellite-to-MANET communication. Consequently, efforts to secure group communications in MANETs are essential.

As the first commercial version of MANETs, VANETs are expected to be deployed in the near future. A VANET consists of on-board units (OBUs) embedded in vehicles serving as mobile computing nodes and road-side units (RSUs) working as the information infrastructure located in the critical points on the road. Mobile vehicles form many cooperative groups in their wireless communication range in the roads, and through roadside infrastructures, vehicles can access other networks such as Internet and satellite communication. VANETs are designed with the primary goal of improving traffic safety and the secondary goal of providing value-added services to vehicles. A substantial body of studies has been devoted to making the primary goal secure and private, by guaranteeing

Q. Wu is with School of Computer, Wuhan University, Wuhan, China. B. Qin is with School of Science, Xi'an University of Technology, Xi'an, China. Q. Wu, B. Qin, J. Domingo-Ferrer and J. A. Manjón are with Universitat Rovira i Virgili, Department of Computer Engineering and Mathematics, UNESCO Chair in Data Privacy, Tarragona, Catalonia.
E-mail: {qianhong.wu, bo.qin, josep.domingo,jesus.manjon}@urv.cat
L. Zhang is with the Software Engineering Institute, East China Normal University, Shanghai, China
E-mail: leizhang@sei.ecnu.edu.cn

the trustworthiness of vehicle-generated traffic reports and the privacy of vehicles (*e.g.* [5], [6]). Only very recently, making the secondary goal secure by securing value-added services in VANETs has been considered [7]. In a typical scenario of this kind of applications, only subscribers among an on-the-fly cooperative group of vehicles can enjoy/decrypt the value-added services (*e.g.* multi-player video games) from remote service providers. Hence, secure group access control is essential to extensively deploy such services in VANETs.

In the above group communication scenarios, the common problem is to enable a sender to securely transmit messages to a remote cooperative group. A solution to this problem must meet several constraints. First, the sender is remote and can be dynamic. Second, the transmission may cross various networks including open insecure networks before reaching the intended recipients. Third, the communication from the group members to the sender may be limited. Also, the sender may wish to choose only a subset of the group as the intended recipients. Further, it is hard to resort to a fully trusted third party to secure the communication. In contrast to the above constraints, mitigating features are that the group members are cooperative and the communication among them is local and efficient. This paper exploits these mitigating features to facilitate remote access control of group-oriented communications without relying on a fully trusted secret key generation center.

A. Related Work

The major security concern in group-oriented communications with access control is key management. Existing key management systems in these scenarios are mainly implemented with two approaches referred to as group key agreement (or group key exchange by some authors) and key distribution systems (or the more powerful notion of broadcast encryption). Both are active research areas having generated large respective bodies of literature.

Group key agreement allows a group of users to negotiate a common secret key via open insecure networks. Then any member can encrypt any confidential message with the shared secret key and only the group members can decrypt. In this way, a confidential intragroup broadcast channel can be established without relying on a centralized key server to generate and distribute secret keys to the potential members. A large number of group key agreement protocols have been proposed [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19]. The earlier efforts [8], [9] focused on efficient establishment of the initial group key. Later studies [10] enable efficient member joins but the cost for a member leave is still comparatively high. A tree key structure has been further proposed and improved to achieve better efficiency for member joins and leaves [11], [13], [17]. The theoretical analysis in [20] proves that, for any tree-based group key agreement scheme, the lower bound of the worst-case cost is $O(\log n)$ rounds of interaction for member join or leave, where n is the number of group members. This optimal round efficiency was recently achieved in [18]. By using a ring-based key structure, the up-to-date proposal in [19] breaks this round barrier because only a constant number of rounds is required for member changes.

In a key distribution system, a trusted and centralized key server presets and allocates the secret keys to potential users, such that only the privileged users can read the transmitted message. The early key distribution protocol [21] does not support member addition/deletion after the system is deployed. This notion was subsequently evolved to allow the sender to freely choose the intended receiver subset of the initial group, which is usually referred to as broadcast encryption. Broadcast encryption is essential for key management [22] in priced media distribution [23] and digital rights management [24]. Broadcast encryption schemes in the literature can be classified in two categories: symmetric-key broadcast encryption and public-key broadcast encryption. In the symmetric-key setting, only the trusted center generates all the secret keys and broadcasts messages to users. Hence, only the key generation center can be the broadcaster or the sender. In the public-key setting, in addition to the secret keys for each user, the trusted center also generates a public key for all the users so that any one can play the role of a broadcaster or sender. Fiat and Naor [25] first formalized broadcast encryption in the symmetric-key setting and proposed a systematic method of broadcast encryption. Similarly to the group key agreement setting, tree-based key structures were subsequently proposed to improve efficiency in symmetric-key based broadcast encryption systems [26], [27]. The state of the art along this research line is presented in [28]. In the public-key setting, Naor and Pinkas presented in [29] the first public-key broadcast encryption scheme in which up to a threshold of users can be revoked. If more than this threshold of users are revoked, the scheme will be insecure and hence not fully collusion-resistant. Subsequently, by exploiting newly developed bilinear paring technologies, a fully collusion-resistant public-key broadcast encryption scheme was presented [30] which has $O(\sqrt{N})$ complexity in key size, ciphertext size and computation cost, where N is the maximum allowable number of potential receivers. A recent scheme [31] reduces the size of the key and the ciphertexts, although it has the same asymptotical sub-linear complexity as [30]. An up-to-date scheme was presented in [32] which strengthens the security concept of public-key broadcast encryption schemes while keeping the same $O(\sqrt{N})$ complexity as [30].

B. Contribution

Our contribution includes three aspects. First, we formalize the problem of secure transmission to remote cooperative groups, in which the core is to establish a one-to-many channel securely and efficiently under certain constraints. We observe that the existing key management approaches do not provide effective solutions to this problem. On one hand, group key agreement provides an efficient solution to secure intragroup communication but, for a remote sender, it requires the sender to simultaneously stay online with the group members for multiple rounds of interactions to negotiate a common secret session key before transmitting any secret contents. This is impractical for a remote sender who may be in a different time zone. This situation is further deteriorated if the sender is mobile or otherwise dynamic.

On the other hand, broadcast encryption enables external senders to broadcast to non-cooperative members of a preset group without requiring the sender to interact with the receivers before transmitting secret contents, but it relies on a centralized key server to generate and distribute secret keys for each group member. This implies that, (i) before a confidential broadcast channel is established, numerous confidential unicast channels from the key server to each potential receiver have to be constructed, and (ii) the key server holding the secret key of each receiver can read all the communications and has to be fully trusted by any potential sender and the group members. The former requirement incurs extra costs while the latter is somewhat unrealistic in open networks. Indeed, only very recently specific efforts were performed to secure communications from a remote sender to a cooperative group when asymmetric group key agreement was proposed by the authors [33] at Eurocrypt 2009. In asymmetric group key agreement, the group members *first negotiate a common public key* but hold different secret keys. Then any sender knowing the group public key can securely encrypt to the group and only the group members can decrypt. The concept of asymmetric group key agreement is theoretically attractive. The instantiated protocols so far [33] have an $O(N)$ size public/secret key per member and does not support member deletion or addition. Subsequently, one-round asymmetric group key agreement protocols were extended [34] to contributory broadcast encryption in which some members can be excluded but new members cannot join. The new functionality of member exclusion is at the cost of a $O(N^2)$ key size, although the ciphertext size remains constant and short. The authors illustrated an efficient tradeoff with the ciphertext size so that both the size of the ciphertext and the size of the keys are $O(N^{2/3})$, which is still large for applications in *ad hoc* networks. Hence, this paper further investigates a new key management paradigm and pursues protocols which are more realistic from the viewpoint of security practitioners.

Second, we propose a new key management paradigm allowing secure and efficient transmissions to remote cooperative groups by effectively exploiting the mitigating features and circumventing the constraints discussed above. The new approach is a hybrid of group key agreement and public-key broadcast encryption. In our approach, each group member has a public/secret key pair. By knowing the public keys of the members (*e.g.*, by retrieving them from a public key infrastructure which is widely available in existing network security solutions), a remote sender can securely broadcast a secret session key to any intended subgroup chosen in an *ad hoc* way, and, simultaneously, any message can be encrypted to the intended receivers with the session key. Only the selected group members can jointly decrypt the secret session key and hence the encrypted message. In this way, the dependence on a fully trusted key server is eliminated. Also, the dynamics of the sender and the group members are coped with, because the interaction between the sender and the receivers before the transmission of messages is avoided and the communication from the group members to the remote sender is minimized.

Third, we present a provably secure protocol in the new key management paradigm and perform extensive experiments in the context of mobile *ad hoc* networks. In the proposed protocol, after extraction of the public group encryption key in the first run, the subsequent encryption by the sender and the decryption by each receiver are both of constant complexity, even in the case of member changes or system updates for rekeying. The initial decryption requires a one-round interaction among receivers. Although the subsequent decryptions in some cases may also require one-round interactions, only few, less than four members will be involved in the interaction. As to security, the proposal is shown secure against an attacker colluding with all the non-intended members. Even such an attacker cannot get any useful information about the messages transmitted by the remote sender. The proof is given under a variant of the standard Decision Diffie-Hellman (DDH) assumption.

To evaluate the practicality of our protocol, we provide a detailed theoretical performance analysis and implement the protocol in the context of MANETs, one of the motivating applications. Both the theoretical analysis and the experimental results show that our proposal is promising for many distributed computing applications.

C. Paper Organization

The rest of this paper is organized as follows. We present the system model in Section II. Section III realizes our protocol and proves its security. We discuss the implementation aspects of our protocol in Section IV. A detailed performance analysis is given in Section V. Section VI concludes the paper.

II. Problem Statement and System Model

A. Problem Statement

We consider a group composed of N users, indicated by $\{\mathcal{U}_1, \cdots, \mathcal{U}_N\}$. A sender would like to transmit secret messages to a receiver subset $\mathbb{S}$ of the N users, where the size of $\mathbb{S}$ is $n \leq N$. The problem is how to enable the sender to efficiently and securely finish the transmission with the following constraints:

1) It is hard to deploy a key generation authority fully trusted by all users and potential senders in open network settings.
2) The communication from the receivers to the sender is limited, *e.g.* in the battlefield communication setting.
3) N might be very large and up to millions, for instance, in vehicular *ad hoc* networks.
4) Both the sender and the receiver sets are dynamic due to *ad hoc* communication.

According to the application scenarios, there are also some mitigating features that may be exploited for solving the problem:

1) n is usually a small or medium value, *e.g.* less than 256.
2) The receivers are cooperative and communicated via efficient local (broadcast) channels.
3) A partially trusted authority, *e.g.* a public key infrastructure, is available to authenticate the receivers (and the senders).

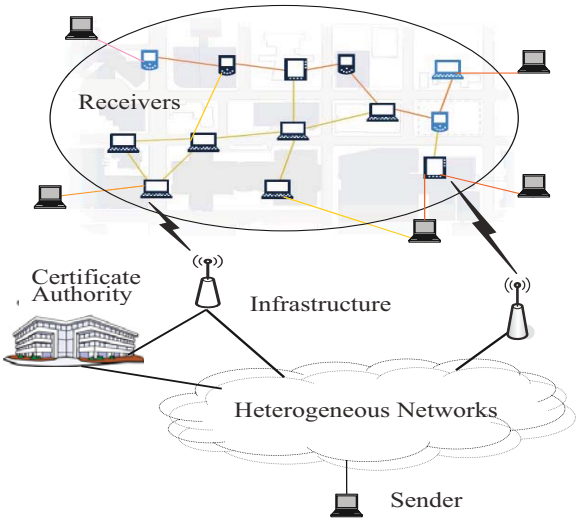

Fig. 1. System architecture

B. System Model

We address the above problem by formalizing a new key management paradigm referred to as group key agreement based broadcast encryption. The system architecture is illustrated in Figure 1. The potential receivers are connected together with efficient local connections. Via communication infrastructures, they can also connect to heterogeneous networks. Each receiver has a public/secret key pair. The public key is certified by a certificate authority but the secret key is kept only by the receiver. A remote sender can retrieve the receiver's public key from the certificate authority and validate the authenticity of the public key by checking its certificate, which implies that no direct communication from the receivers to the sender is necessary. Then the sender can send secret messages to any chosen subset of the receivers.

We next formally define the model of group key agreement based broadcast encryption. The definition incorporates the up-to-date definitions of group key agreement [19] and public-key broadcast encryption [32]. Since the core of key management is to securely distribute a session key to the intended receivers, it is sufficient to define the system as a session key encapsulation mechanism. Then the sender can simultaneously encrypt any message under the session key and only the intended receivers can decrypt. Specifically, our key management system consists of the following (probabilistic) polynomial-time algorithms:

- $\texttt{KeyGen}(i, n, N)$: This key generation algorithm is run by each user $\mathcal{U}_i \in \{\mathcal{U}_1, \cdots, \mathcal{U}_N\}$ to generate her public/private key pair. A user takes as input the system parameters n, N and her index $i \in \{1, \cdots, N\}$, and outputs $\langle pk_i, sk_i\rangle$ as her public/secret key pair. Denote $\{\langle pk_i, sk_i\rangle | \mathcal{U}_i \in \mathbb{S} \subseteq \{\mathcal{U}_1, \cdots, \mathcal{U}_N\}\}$ by $\langle pk_i, sk_i\rangle_{\mathbb{S}}$ and similarly, $\{\langle pk_i\rangle | \mathcal{U}_i \in \mathbb{S} \subseteq \{\mathcal{U}_1, \cdots, \mathcal{U}_N\}\}$ by $\langle pk_i\rangle_{\mathbb{S}}$. Here, we implicitly omit the input security parameter λ: actually, n, N are polynomials in λ.
  We assume that each user's public key is certified by a publicly accessible certificate authority so that any one can retrieve the public keys and verify their authenticity. This is plausible as public key infrastructures have been a standard component in many systems supporting security services. The key generation and the registration to the certificate authority can be done offline before the online message transmission by the sender.
- $\texttt{Encryption}(\mathbb{S}, \langle pk_i\rangle_{\mathbb{S}})$: It is run by any sender who may or may not be in $\{\mathcal{U}_1, \cdots, \mathcal{U}_N\}$, provided that the sender knows the public keys of the potential receivers. It takes as input a recipient set $\mathbb{S} \subseteq \{\mathcal{U}_1, \cdots, \mathcal{U}_N\}$ and the public key pk_i for $\mathcal{U}_i \in \mathbb{S}$. If $|\mathbb{S}| = n$, it outputs a pair $\langle Hdr, k\rangle$ where Hdr is called the header and k is the message encryption key. $(\mathbb{S}, Hdr)$ is sent to the receivers. This algorithm incorporates the functionality of the encryption procedure in traditional broadcast encryption systems.
- $\texttt{Decryption}(\mathcal{U}_j(sk_j)_{\mathbb{S}}, Hdr, \langle pk_i\rangle_{\mathbb{S}})$: This algorithm is jointly run by the intended receivers to extract the secret session key k hidden in the header. Each receiver $\mathcal{U}_j$ privately inputs her secret key sk_j. The common inputs are the header Hdr and the public keys of receivers in the recipient set $\mathbb{S}$. If $|\mathbb{S}| = n$, each receiver in $\mathbb{S}$ outputs the same session key k. This procedure incorporates a traditional group key agreement protocol. It exploits the cooperation of the receivers with efficient local connections.

We next justify the assumptions on trusted authorities and limited communication from the receivers to the sender in our key management paradigm. At a first look, the new paradigm seems to require a trusted third party as its counterpart in traditional broadcast encryption systems. A closer look shows there is a difference. In a traditional broadcast encryption system, the third party has to be *fully* trusted, that is, the third party knows the secret keys of all group members and can read any transmission to any subgroup of the members. This kind of fully trusted third party is hard to implement in open networks. In contrast, the third party in our key management model is only *partially* trusted. In other words, the third party only knows and certifies the public key of each member. This kind of partially trusted third party has been implemented and is known as public key infrastructure (PKI) in open networks. Second, the new key management paradigm ostensibly requires a sender to know the keys of the receivers, which may need communications from the receivers to the sender as in traditional group key agreement protocols. However, some subtleties must be pointed out here. In traditional group key agreement protocols, the sender has to simultaneously stay online with the receivers and direct communications from the receivers to the sender are needed. This is difficult for a remote sender. On the contrary, in our key management paradigm, the sender only needs to obtain the receivers' public keys from a third party and no direct communication from the receivers to the sender is required, which is implementable with exactly the existing PKIs in open networks. Hence, this is feasible for a remote sender. Furthermore, a sender does not need to frequently contact the third party or keep a large number

of keys since a sender usually communicates to a relatively fixed group in practice. For instance, a department manager usually communicates with her subordinates, superiors and other department managers, but rarely needs to send secret messages to all staff members.

The above discussions show that our key management paradigm addresses the first two constraints of secure transmission to remote cooperative groups, listed in Section II-A. We further show that the rest of constraints are also addressed. From the definition, only the sender and the intended receivers are involved in the Encryption and Decryption procedures. Hence, the complexity of the system does not depend on the size N of the full group but on the size of the receiver subset. The same analysis applies to the dynamics of the sender and the receivers. This implies that our approach is particularly efficient in the case when the full group is very large but the actual receiver set is small. Hence, the last two constraints of Section II-A are also addressed. Indeed, our protocol enjoys almost constant complexity when coping with the change of the sender or the receivers. This is especially attractive for mobile networks.

C. Security Definitions

When focusing on the confidentiality of the session key transmitted by the sender, we implicitly assume that the public keys of users are authentic, that is, we assume that they have been previously authenticated.

We start by defining the correctness of our system as the property that any user in the receiver set can decrypt a valid header. A formal definition follows.

Definition 1 (Correctness)**.** *Assume the model described in the previous section. A group key agreement based broadcast encryption scheme is correct if for* $\{\langle pk_i, sk_i\rangle\} \leftarrow \mathsf{KeyGen}(i, n, N)$*, all* $\mathbb{S} \subseteq \{\mathcal{U}_1, \cdots, \mathcal{U}_N\}$ *(with* $|\mathbb{S}| = n$*) and all* $\mathcal{U}_i \in \mathbb{S}$*, if* $\langle Hdr, k\rangle \leftarrow \mathsf{Encryption}(\mathbb{S}, \langle pk_i\rangle_{\mathbb{S}})$*, then it holds that* $\mathsf{Decryption}(\mathcal{U}_j(sk_j)_{\mathbb{S}}, Hdr, \langle pk_i\rangle_{\mathbb{S}}) = k$ *for any* $U_j \in \mathbb{S}$*.*

The above correctness definition can be satisfied by a trivial implementation consisting of encrypting the message with each user's public key, but this implies substantial encryption complexity and ciphertext size linear in the size of the receiver set. Our requirement is that encryption should be efficient and the ciphertext should be short, regardless of the number of receivers.

We now define the main security property, *i.e.* the secrecy of a group key agreement based broadcast encryption scheme. In Section II-B, to achieve better practicality, our key management paradigm is modeled as a key encapsulation mechanism (KEM) in which a sender sends a (short) secret session key to the intended receivers and, simultaneously, (long) messages can be encrypted under the session key using a secure symmetric encryption algorithm. Hence, we define secrecy as the indistinguishability of the encrypted session key from a random element in the session key space. Since there exist standard conversions (*e.g.*, [35]) from KEMs secure against chosen-plaintext attacks (CPA) to encryption secure against adaptively chosen-ciphertext attacks (CCA2), it suffices to define the CPA secrecy of group key agreement based broadcast encryption. However, noting that this encryption is designed for distributed applications where the users are likely to be corrupted, we include full collusion resistance into our secrecy definition. That is, the adversary is allowed to see the public keys of all users and corrupt some of them to obtain their secret keys. It is required for such an attacker that he cannot distinguish a session key hidden in the header to non-corrupted users from a random element in the session key space. Formally, secrecy is defined by means of the following game between an attacker $\mathcal{A}$ and a challenger $\mathcal{CH}$. Both $\mathcal{CH}$ and $\mathcal{A}$ are given (λ, N, n) as input, where N, n are polynomials in the security parameter λ.

- **Setup**. The challenger runs $\mathsf{KeyGen}(i, n, N)$ to obtain the users' public keys. The challenger gives the public keys and public system parameters to the attacker.
- **Corruption**. Attacker $\mathcal{A}$ adaptively issues private key queries for some indices $i \in \{1, \cdots, N\}$.
- **Challenge**. At some point, the attacker specifies a challenge set $\mathbb{S}^*$, satisfying that $|\mathbb{S}^*| = n$ and, for the private key of any user $\mathcal{U}_i$ queried in the corruption step, $\mathcal{U}_i \notin \mathbb{S}^*$. The challenger sets $\langle Hdr^*, k_0\rangle \leftarrow \mathsf{Encryption}(\mathbb{S}^*, \langle pk_i\rangle_{\mathbb{S}^*})$ and $k_1 \leftarrow \mathbb{K}$. It sets $b \leftarrow \{0, 1\}$ and gives (Hdr^*, k_b) to attacker $\mathcal{A}$.
- **Observation**. After receiving the challenge header, the attacker $\mathcal{A}$ can access the public transcripts from users in $\mathbb{S}^*$ during the decryption interactions.
- **Guess**. Attacker $\mathcal{A}$ outputs a guess bit $b' \in \{0, 1\}$ for b and wins the game if $b = b'$.

We define $\mathcal{A}$'s advantage in attacking the group key agreement based broadcast encryption system with security parameter λ as

$$Adv_{\mathcal{A},n,N}(1^\lambda) = |\Pr[b = b'] - \frac{1}{2}|.$$

Definition 2 (Secrecy)**.** *We say that a group key agreement based broadcast encryption scheme is collusion-resistant against adaptive attacks if for any polynomial-time attacker* $\mathcal{A}$ *we have that* $Adv_{\mathcal{A},n,N}(1^\lambda)$ *is negligible in* λ*, and the scheme is collusion-resistant against static attacks if the attacker* $\mathcal{A}$ *has to commit to the challenge set* $\mathbb{S}^*$ *before the setup stage.*

III. Key Management for Transmission to Remote Cooperative Groups

A. Mathematical Background

Our scheme is built from bilinear groups [36]. Let `PairGen` be an algorithm that, on input a security parameter 1^λ, outputs a tuple $\Upsilon = (p, \mathbb{G}, \mathbb{G}_T, e)$, where $\mathbb{G}$ and $\mathbb{G}_T$ have the same prime order p, and $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is an efficient non-degenerate bilinear map such that $e(g, g) \neq 1$ for any generator g of $\mathbb{G}$, and for all $u, v \in \mathbb{Z}$, it holds that $e(g^u, g^v) = e(g, g)^{uv}$.

At Crypto 2007 [37], Bresson *et al.* formalized the (P, Q)-Decision Diffie-Hellman ((P, Q)-DDH) assumption to simplify the security proofs for group key agreement protocols that use various extensions of the Decision Diffie-Hellman assumption. Our scheme relies on the (P, Q)-DDH, which is next reviewed.

Definition 3. ***((P,Q)-DDH)*** *Assume that g is a generator of a finite cyclic group $\mathbb{G}$ of prime order p. Let P, Q be two sets of polynomials in $\mathbb{Z}_p[\alpha_0, \alpha_1, \cdots, \alpha_n]$, where the polynomials in Q are not a linear combination of those in P. The (P,Q)-DDH states that, given $\{g^{p_i(x_0,x_1,\cdots,x_n)}\}_{p_i \in P}$ and g^y for randomly chosen $y, x_i \in \mathbb{Z}_p$, it is hard for a polynomial-time attacker to distinguish $\{g^{y(q_i(x_0,x_1,\cdots,x_n))}\}_{q_i \in Q}$ from randomly sampled elements in $\mathbb{G}$.*

B. The Proposal

The proposed key management scheme incorporates the ideas of broadcast encryption systems [32] and GKA protocols [8].

KeyGen. Assuming the above bilinear group setting, each user i for $i = 1, \cdots, N$ randomly chooses $x_i \in \mathbb{Z}_p^*$ and computes

$$X_i = g^{x_i} \in \mathbb{G}.$$

User i keeps x_i secret as her secret key, and registers X_i to the certificate authority as her public key. The registered public keys are supposed to be organized in a certain order.

Encryption. Assume that a sender wishes to broadcast to users indexed by $\{i_1, \cdots, i_n\} \subseteq \{1, \cdots, N\}$. The sender runs the following algorithm.

1) Randomly select $r, x_{i_0} \in \mathbb{Z}_p^*$ and compute:
$$X_{i_0} = g^{x_{i_0}}, Y_{i_0} = (X_{i_1}/X_{i_n})^{x_{i_0}}, c = g^r.$$
2) Extract the public group encryption key for $\mathbb{S}$:
$$K = e(X_{i_1}, X_{i_2}) e(X_{i_2}, X_{i_3}) \cdots e(X_{i_{n-1}}, X_{i_n}) \quad (1)$$
Note that $K = e(g,g)^{x_{i_1}x_{i_2} + x_{i_2}x_{i_3} + \cdots + x_{i_{n-1}}x_{i_n}}$.
3) Compute
$$S = K e(X_{i_n}, X_{i_0}) e(X_{i_0}, X_{i_1})$$
$$= e(g,g)^{x_{i_1}x_{i_2} + x_{i_2}x_{i_3} + \cdots + x_{i_n}x_{i_0} + x_{i_0}x_{i_1}} \stackrel{\text{Define}}{=} e(g,g)^x.$$
4) Compute the secret session key
$$k = S^r = e(g,g)^{xr}.$$
5) Broadcast the header
$$Hdr = (X_{i_0}, Y_{i_0}, c)$$
as well as the receiver set $\mathbb{S}$ to the receivers.

Using the session key k, the sender can encrypt any message to the receivers with any secure symmetric encryption algorithm, *e.g.* AES. The encrypted message can be simultaneously sent to the receivers with the header.

Decryption. The intended receivers run this algorithm as follows.

1) For $j = 1, \cdots, n$, each receiver $\mathcal{U}_{i_j} \in \mathbb{S}$ publishes
$$Y_{i_j} = (X_{i_{j+1}}/X_{i_{j-1}})^{x_{i_j}} = g^{(x_{i_{j+1}} - x_{i_{j-1}})x_{i_j}} \in \mathbb{G}$$
where the subscript j of i_j is computed modulo $n+1$. That is, $n+1 \equiv 0 \mod (n+1)$.
2) Each receiver indexed by i_j can compute the secret decryption key
$$d = X_{i_{j-1}}^{(n+1)x_{i_j}} Y_{i_j}^n Y_{i_{j+1}}^{n-1} \cdots Y_{i_{j-2}} \quad (2)$$
Similarly, the subscript j of i_j is also computed modulo $n+1$ here.
3) Using d, each receiver extracts the session key k from c by computing
$$k = e(d, c).$$

Finally, the receiver can read messages encrypted with this session key.

The correctness of the scheme follows from the following direct verification:

$$\begin{aligned} d &= X_{i_{j-1}}^{(n+1)x_{i_j}} Y_{i_j}^n Y_{i_{j+1}}^{n-1} \cdots Y_{i_{j-2}} \\ &= g^{(n+1)x_{i_{j-1}}x_{i_j}} g^{n(x_{i_{j+1}} - x_{i_{j-1}})x_{i_j}} \\ &\quad \times g^{(n-1)(x_{i_{j+2}} - x_{i_j})x_{i_{j+1}}} \cdots g^{(x_{i_{j-1}} - x_{i_{j-3}})x_{i_{j-2}}} \\ &= g^{(n+1)x_{i_{j-1}}x_{i_j}} g^{nx_{i_j}x_{i_{j+1}} - nx_{i_{j-1}}x_{i_j}} \\ &\quad \times g^{(n-1)x_{i_{j+1}}x_{i_{j+2}} - (n-1)x_{i_j}x_{i_{j+1}}} \cdots g^{x_{i_{j-2}}x_{i_{j-1}} - x_{i_{j-3}}x_{i_{j-2}}} \\ &= g^{x_{i_{j-1}}x_{i_j}} g^{x_{i_j}x_{i_{j+1}}} g^{x_{i_{j+1}}x_{i_{j+2}}} \cdots g^{x_{i_{j-2}}x_{i_{j-1}}} \\ &= g^{x_{i_0}x_{i_1} + x_{i_1}x_{i_2} + x_{i_2}x_{i_3} + \cdots + x_{i_n}x_{i_0}} = g^x \end{aligned}$$

Hence, $k = e(d,c) = e(g,g)^{xr}$. This completes the correctness proof of the scheme.

The security of our scheme relies on the (P,Q)-DDH assumption. Bresson *et al.* formalized a family of assumptions which can be instantiated by setting the polynomial sets P and Q. For our proposal, one can set $P = \{\alpha_i | i = 0, \cdots, n\} \cup \{\alpha_i \alpha_{i+1} - \alpha_{i-1}\alpha_i | i = 0, \cdots, n\}$ and $Q = \{\sum_{i=0}^n \alpha_i \alpha_{i+1}\}$, where the subscripts i are computed modulo $n+1$. Based on the instantiated (P,Q)-DDH assumption, we have the following claim.

Theorem 1. *Let $\mathcal{A}$ be a static attacker committing to a target set $\mathbb{S}^* \subseteq \{1, \cdots, N\}$ satisfying $|\mathbb{S}^*| = n$ as in Definition 2. If the attacker $\mathcal{A}$ can distinguish the session key to receivers in $\mathbb{S}^*$ from a random element in $\mathbb{G}_T$ with advantage ϵ in time τ, then there exists an algorithm $\mathcal{B}$ breaking an instance of the (P,Q)-DDH assumption in $\mathbb{G}$ allowing a bilinear map with the same advantage ϵ in time at most $\tau + (N-n)\tau_E + 1\tau_P$, where τ_E is the time to compute an exponentiation in $\mathbb{G}$ and τ_P is the time to compute a bilinear map.*

Since the (P,Q)-DDH assumption is believed to hold, no such polynomial-time algorithm $\mathcal{B}$ exists, and hence no polynomial-time attacker can distinguish the session key to any receiver set from a random string in the session key space $\mathbb{G}_T$. Therefore, our scheme is secure against polynomial-time bounded attackers.

Proof: We first outline the proof. For any $\mathbb{S}^* \subseteq \{1, \cdots, N\}$ satisfying $|\mathbb{S}^*| = n$, we can construct an algorithm $\mathcal{B}$ to break an instance of the (P,Q) assumption. $\mathcal{B}$ is given the corresponding instance of the (P,Q)-DDH challenge. With it, $\mathcal{B}$ simulates the system parameters, the public keys of the users, and the secret keys of the users whom the attacker may corrupt. The simulated data are indistinguishable from those generated in a real scheme from the viewpoint of the attacker, so that the attacker does not know he is interacting with a simulator. Then $\mathcal{B}$ uses $\mathcal{A}$'s guess to solve the (P,Q)-DDH challenge. This contradicts the (P,Q)-DDH assumption. Therefore, such an attacker against our key management scheme does not exist and our scheme is secure.

Assume that $\mathbb{S}^* = \{i_1, \cdots, i_n\}$. The corresponding (P, Q)-DDH is instantiated with polynomial sets $P = \{\alpha_{i_j} | j = 0, \cdots, n\} \cup \{\alpha_{i_j}\alpha_{i_{j+1}} - \alpha_{i_{j-1}}\alpha_{i_j} | j = 0, \cdots, n\}$ and $Q = \{\sum_{j=0}^{n} \alpha_{i_j}\alpha_{i_{j+1}}\}$, where the subscripts j's are computed modulo $n+1$. Clearly, the only polynomial $\sum_{j=0}^{n} \alpha_{i_j}\alpha_{i_{j+1}}$ in Q is not a linear combination of polynomials in P.

$\mathcal{B}$ is given the (P, Q)-DDH challenge $\{g^{x_{i_j}} | j = 0, \cdots, n\} \cup \{g^{x_{i_j}x_{i_{j+1}} - x_{i_{j-1}}x_{i_j}} | j = 0, \cdots, n\}$ and g^y, where $x_{i_j}, y \in \mathbb{Z}_p^*$ are randomly chosen and unknown to $\mathcal{B}$. $\mathcal{B}$ is additionally given $Z \in \mathbb{G}$ and required to answer whether $Z = g^{y\sum_{j=0}^{n} x_{i_j}x_{i_{j+1}}}$ or not.

After obtaining the (P, Q)-DDH challenge instance, $\mathcal{B}$ initializes the game in Definition 2 with $\mathcal{A}$ as follows.

In the Setup phase, $\mathcal{A}$ can request the system parameters and the public keys of all users. For $i \in \{1, \cdots, N\} \setminus \mathbb{S}^*$, $\mathcal{B}$ randomly selects $x_i \in \mathbb{Z}_p^*$ and computes $X_i = g^{x_i}$, and sets $\mathcal{U}_i$'s public-secret key pair as (X_i, x_i). For $i \in \mathbb{S}^*$, $\mathcal{B}$ sets $\mathcal{U}_i$'s public key as X_i from the (P, Q)-DDH challenge. In this case, $\mathcal{B}$ does not know the corresponding user's secret key. $\mathcal{B}$ forwards X_i as user i's public key and other system parameters to the attacker $\mathcal{A}$. This simulation is perfect since both $\{x_i\}_{i \notin S}$ and $\{x_i\}_{i \in S}$ are randomly sampled from $\mathbb{Z}_p^*$.

In the Corruption phase, the attacker $\mathcal{A}$ can adaptively query the secret key of any user indexed by $\{1, \cdots, N\} \setminus \mathbb{S}^*$. Since $\mathcal{B}$ generates the secret keys for these users, $\mathcal{B}$ can correctly answer the corruption request for these users and $\mathcal{B}$ simulates the corrupted users perfectly.

In the Challenge phase, $\mathcal{A}$ can request (Hdr, k_b) from $\mathcal{A}$, where $b \in \{0, 1\}$ is unknown, indeed a question to be answered by the attacker. From Definition 2, the requirement on b is that, if $b = 0$, Hdr is a header of $k_b \in \mathbb{G}_T$ encrypted by the public keys of users indexed by $\mathbb{S}^*$; else if $b = 1$, k_b is randomly chosen from $\mathbb{G}_T$. $\mathcal{B}$ can compute (Hdr, k_b) from the (P, Q)-DDH challenge: $X_{i_0} = g^{x_{i_0}}, Y_{i_0} = g^{x_{i_0}x_{i_1} - x_{i_n}x_{i_0}} = g^{(x_{i_1} - x_{i_n})x_{i_0}}, c = g^y$ and $k_b = e(Z, g)$. Set $Hdr = (X_{i_0}, Y_{i_0}, c)$. Clearly, Hdr is well formed and it has the same distribution as in the real scheme.

In the Observation phase, $\mathcal{A}$ can request the transcripts Y_{i_j} from users indexed by $\mathbb{S}^*$. $\mathcal{B}$ simulates these transcripts from the (P, Q)-DDH challenge: $Y_{i_j} = g^{x_{i_j}x_{i_{j+1}} - x_{i_{j-1}}x_{i_j}} = g^{(x_{i_{j+1}} - x_{i_{j-1}})x_{i_j}}$ for $j = 1, \cdots, n$. They are exactly the same as those in the real scheme.

In the Guess phase, $\mathcal{A}$ is required to answer whether k_b is the session key hidden in Hdr or is independent from Hdr. Note that, if and only if $Z = g^{y\sum_{j=0}^{n} x_{i_j}x_{i_{j+1}}}$, $k_b = e(g, g)^{y\sum_{j=0}^{n} x_{i_j}x_{i_{j+1}}}$ is the session key hidden in Hdr. Else, k_b is independent of Hdr. Hence, $\mathcal{B}$ can conclude $Z = g^{y\sum_{j=0}^{n} x_{i_j}x_{i_{j+1}}}$ if $\mathcal{A}$ answers that k_b is the session key hidden in Hdr. $\mathcal{B}$ answers correctly if $\mathcal{A}$ does.

Note that algorithm $\mathcal{B}$ perfectly answers the queries from attacker $\mathcal{A}$ in the Setup, Corruption, Challenge and Observation stages. Hence, from the viewpoint of $\mathcal{A}$, the interactions between her and $\mathcal{B}$ are the same as the interactions between her and the sender together with the receivers in the real world. Further, in the Guess stage, $\mathcal{B}$ will correctly answer the (P, Q)-DDH challenge if $\mathcal{A}$ correctly distinguishes the session key hidden in Hdr from a random string sampled from the session key space. Therefore, $\mathcal{B}$'s success probability is at least as large as the attacker's success probability.

As to time complexity, the extra overhead for $\mathcal{B}$ is $N - n$ exponentiations to generate the public keys of non-corrupted users and one pairing to compute k_b. Hence, $\mathcal{B}$ needs time at most $\tau + (N - n)\tau_{Exp} + 1\tau_{Pair}$. This completes the proof. ■

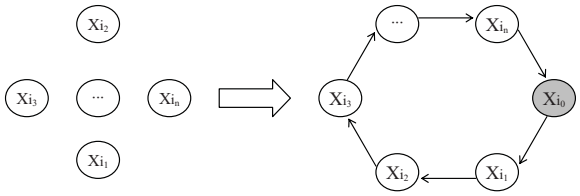

Fig. 2. Member organization

IV. Implementation Issues

In this section, we consider the practical aspects of our key management scheme.

A. Member Organization

Many key management (*i.e.*, group key agreement or broadcast encryption) schemes organize the users in a tree-based structure. However, for our scheme, it is preferable to organize them in a chain and then use the sender to close the chain to form a logical ring. The chain can be formed by ordering the users lexicographically by the least important bits of their unique public keys, and then a ring is formed by closing the chain with the sender as illustrated in Figure 2, where the public keys $\{X_{i_1}, \cdots, X_{i_n}\}$ of the receivers and the temporary public key X_{i_0} of the sender appear as the corresponding nodes in the ring, respectively.

Compared with the tree-based structure, the above structure allows better performance for receiver and sender changes. Without loss of generality, assume that the sorted chain is $X_{i_1} \prec \cdots \prec X_{i_n}$. In this way, if the sender changes, only receivers $\mathcal{U}_{i_1}$ and $\mathcal{U}_{i_n}$ need to communicate with other receivers during the decryption procedure if the receiver set does not change (functionally, this is equivalent to updating the group decryption key of the receivers, so that the previous sender cannot read the message transmitted by the current sender; see the following Section IV-C). This is very desirable if the sender may change frequently while the local cooperative group is relatively static. Further, it is preferable that the certificate authority form the list of users and corresponding public keys in the same order, so that no specific sorting efforts are required from each sender.

B. Member Deletion/Addition and Group Partition/Merging

In existing group key agreement based key management protocols, to exclude a group member or enroll a new member, multiple rounds of communication among the members are required *before* the sender can securely broadcast to the new receiver set. In our scheme, it is almost free of cost for a sender to exclude a group member by deleting the public key

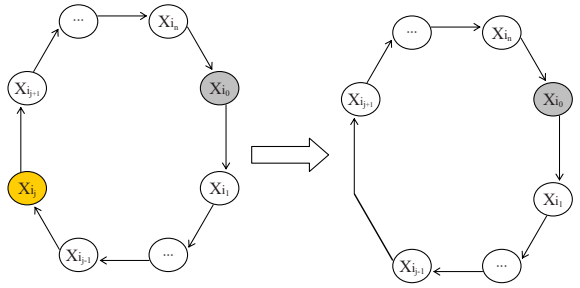

Fig. 3. Member deletion

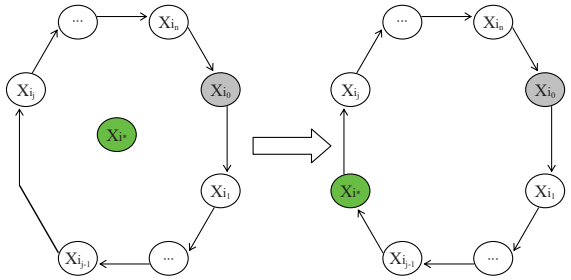

Fig. 4. Member addition

of the member from the public key chain, or, similarly, to enroll a user as a new member by inserting that user's public key into the proper position of the public key chain of the receivers. After the deletion/addition of certain member, a new logical public-key ring naturally forms. Hence, a trivial way to enable this change is to run the protocol independently with the new key ring. We illustrate in the following an alternative implementation equivalent to the trivial way, but such that much cost is saved by exploiting the values computed in the last run of the protocol.

Member deletion. Figure 3 shows the deletion of member $\mathcal{U}_{i_j}$ from the receiver group. Then the sender and the remaining receivers need to apply this change to their subsequent encryption and decryption procedures.

`Encryption.` The sender runs this algorithm as follows.

1) Randomly select $r', x'_{i_0} \in \mathbb{Z}_p^*$ and compute:

$$X'_{i_0} = g^{x'_{i_0}}, Y'_{i_0} = (X_{i_1}/X_{i_n})^{x'_{i_0}}, c = g^{r'}.$$

In this step, the sender indexed by i_0 re-inserts herself into the ring and connects to receivers i_1 and i_n. Hence, the operation is the same as that of the basic protocol but the sender has to choose new random values r', x'_{i_0}.

2) Compute the new public group encryption key:

$$K' = \frac{Ke(X_{i_{j-1}}, X_{i_{j+1}})}{e(X_{i_{j-1}}, X_{i_j})e(X_{i_j}, X_{i_{j+1}})}.$$

Since member i_j is deleted, receiver i_{j+1} then plays the role of the deleted receiver and connects to receiver i_{j-1}. According to Equation (1), the new group encryption key is $K' = e(X_{i_1}, X_{i_2})e(X_{i_2}, X_{i_3}) \times \cdots \times e(X_{i_{j-2}}, X_{j_{j-1}})e(X_{i_{j-1}}, X_{j_{j+1}}) \times \cdots \times e(X_{i_{n-1}}, X_{i_n})$. Note that the sender has known K computed in Equation (1). Then the computation of K' can be simplified and K' can be obtained by computing $K' = \frac{Ke(X_{i_{j-1}}, X_{i_{j+1}})}{e(X_{i_{j-1}}, X_{i_j})e(X_{i_j}, X_{i_{j+1}})}$ as above. The following three steps are to compute the session key and the header using K' to replace K. Accordingly, we only need literal modifications of the remaining steps of the basic protocol.

3) Compute $S' = K'e(X_{i_n}, X'_{i_0})e(X'_{i_0}, X_{i_1})$.
4) Compute the new secret session key $k' = (S')^{r'}$.
5) Broadcast to the receivers the new header $Hdr = (X'_{i_0}, Y'_{i_0}, c')$.

`Decryption.` The receivers run this algorithm as follows.

1) According to Step 1 of the Decryption procedure of the basic protocol, it is easy to see that only receivers $\mathcal{U}_{i_{j-1}}$ and $\mathcal{U}_{i_{j+1}}$ need to respond to the change in this step. They respectively publish

$$Y'_{i_{j-1}} = (X_{i_{j+1}}/X_{i_{j-2}})^{x_{i_{j-1}}}, Y'_{i_{j+1}} = (X_{i_{j+2}}/X_{i_{j-1}})^{x_{i_{j+1}}}.$$

2) For $t = 1, \cdots, j-1$, denote $X'_{i_t} = X_{i_t}, x'_{i_t} = x_{i_t}$; for $t = j$, denote $x'_{i_t} = x_{i_{t+1}}$; for $t = j+1, \cdots, n-1$, denote $X'_{i_t} = X_{i_{t+1}}, X'_{i_t} = X_{i_{t+1}}$. For $t = 1, \cdots, j-2$, set $Y'_{i_t} = Y_{i_t}$; for $t = j+2, \cdots, n-1$, set $Y'_{i_t} = Y_{i_{t+1}}$. Each receiver indexed by $i_t (t = 1, \cdots n-1)$ can compute the new group decryption key

$$d' = (X'_{i_{t-1}})^{nx_{i_t}} (Y'_{i_t})^{n-1} (Y'_{i_{t+1}})^{n-2} \cdots Y'_{i_{t-2}}.$$

In the above, due to deletion of receiver i_j, only $n-1$ receivers remain in the receiver set. Further, to employ Equation (2) in a clearer way, we use updated indices and corresponding notions to represent the remaining receivers' public keys and secret keys.

3) Using d', each receiver extracts the new session key k' from c' by computing $k' = e(d', c')$.

Member addition. If the sender would like to include a new member $\mathcal{U}_{i^*}$, the sender just needs to retrieve the public key X_{i^*} of this user and insert it into the public key chain of the current receiver set. Figure 4 shows the addition of member $\mathcal{U}_{i^*}$ to the receiver group, where we assume that $X_{i_{j-1}} \prec X_{i^*} \prec X_{i_j}$. Then the sender and receivers in the new receiver set need to apply this change to their subsequent encryption and decryption procedures.

`Encryption.` The sender runs this algorithm as follows.

1) Randomly select $r', x'_{i_0} \in \mathbb{Z}_p^*$ and compute:

$$X'_{i_0} = g^{x'_{i_0}}, Y'_{i_0} = (X_{i_1}/X_{i_n})^{x'_{i_0}}, c' = g^{r'}.$$

2) Compute the new public group encryption key:

$$K' = \frac{Ke(X_{i_{j-1}}, X_{i^*})e(X_{i^*}, X_{i_j})}{e(X_{i_{j-1}}, X_{i_j})}.$$

3) Compute $S' = K'e(X_{i_n}, X'_{i_0})e(X'_{i_0}, X_{i_1})$.
4) Compute the new secret session key $k' = (S')^{r'}$.
5) Broadcast the new header $Hdr = (X'_{i_0}, Y'_{i_0}, c')$.

`Decryption.` The intended receivers run this algorithm as follows.

1) Only receivers $\mathcal{U}_{i_{j-1}}$, $\mathcal{U}_{i^*}$ and $\mathcal{U}_{i_{j+1}}$ need to respond to the change in this step. They respectively publish

$$Y'_{i_{j-1}} = (X_{i^*}/X_{i_{j-2}})^{x_{i_{j-1}}}, Y'_{i^*} = (X_{i_{j+1}}/X_{i_{j-1}})^{x^*},$$
$$Y'_{i_{j+1}} = (X_{i_{j+2}}/X_{i^*})^{x_{i_{j+1}}}.$$

2) For $t = 1, \cdots, j-1$, denote $X'_{i_t} = X_{i_t}, x'_{i_t} = x_{i_t}$; for $t = j$, denote $X'_{i_t} = X_{i^*}, x'_{i_t} = x_{i^*}$; for $t = j+1, \cdots, n+1$, denote $X'_{i_t} = X_{i_{t-1}}, x'_{i_t} = x_{i_{t-1}}$. For $t = 1, \cdots, j-2$, set $Y'_{i_t} = Y_{i_t}$; for $t = j$, set $Y'_{i_t} = Y'_{i^*}$; for $t = j+2, \cdots, n+1$, denote $Y'_{i_t} = Y_{i_{t-1}}$. Each receiver indexed by $i_t (t = 1, \cdots n+1)$ can compute the new secret decryption key

$$d' = (X'_{i_{t-1}})^{(n+2)x'_{i_t}} (Y'_{i_t})^{n+1} (Y'_{i_{t+1}})^n \cdots Y'_{i_{t-2}}.$$

3) Using d', each receiver extracts the new session key k' from c' by computing $k = e(d', c')$.

By repeatedly invoking the member addition operation, a sender can merge two receiver sets into a single group. Similarly, by repeatedly invoking the member deletion operation, a sender can partition one receiver set into two groups. Both merging and partitioning can be done efficiently.

C. Rekeying

The above refers to the change of members. Even if the receiver group does not change, various scenarios may require key update. This is a complex issue in most key management schemes. On the contrary, our protocol can provide three levels of key update, which facilitates flexible rekeying strategies.

Session key update. This first level is to update the session key k. This key is used to encrypt digital contents to the receivers and it expires after each session. To update the session key, the sender just needs to partially run Steps 1, 4 and 5 in the `Encryption` procedure:

1) Randomly select a new value $r' \in \mathbb{Z}_p^*$ and compute $c' = g^{r'}$.
4) Compute the new session key $k' = S^{r'}$.
5) Broadcast a new header $Hdr' = c'$ to the receivers.

Receivers only need to execute Step 3 in the `Decryption` procedure:

3) Compute $k' = e(d, c')$.

Note that Step 1 of `Decryption` is not necessary as the receivers have obtained d. This implies that updating the session key does not require the receivers to communicate with each other. Hence, the session key can be updated frequently.

Group decryption key update. The second level is to update the secret decryption key d used by the receivers to compute the session key $k = e(d, c)$. Due to the difficulty of computing the inverse of the bilinear map, it is hard for an attacker who knows the session key k to deduce d. Hence, d needs to be updated less frequently than the session key k. To update the shared decryption key d, the sender only needs to run Steps 1), 3), 4) and 5) in the `Encryption` procedure:

1) Randomly select $r', x'_{i_0} \in \mathbb{Z}_p^*$ and compute:

$$X'_{i_0} = g^{x'_{i_0}}, Y'_{i_0} = g^{(x_{i_1} - x_{i_n})x'_{i_0}}, c' = g^{r'}.$$

3) Compute $S' = Ke(X_{i_n}, X'_{i_0}) e(X'_{i_0}, X_{i_1})$.
4) Compute the new session key $k' = (S')^{r'}$.
5) Broadcast the new header $Hdr' = (X'_{i_0}, Y'_{i_0}, c')$.

Receivers only need to partially execute the three steps in the `Decryption` procedure:

1) Receiver $\mathcal{U}_{i_1}$ and $\mathcal{U}_{i_n}$ respectively publish

$$Y'_{i_1} = (X_{i_2}/X'_{i_0})^{x_{i_j}}, Y'_{i_n} = (X'_{i_0}/X_{i_{n-1}})^{x_{i_j}}.$$

2) According to Equation (2), for each receiver indexed by i_j, $\mathcal{U}_{i_1}$ can obtain the new group decryption key d' by computing

$$d' = \frac{d(X'_{i_0})^{(n+1)x_{i_1}} (Y'_{i_1})^n Y'_{i_n}}{X_{i_0}^{(n+1)x_{i_1}} (Y_{i_1})^n Y_{i_n}},$$

and the other receiver $\mathcal{U}_{i_j}$ can obtain the new group decryption key d' by computing

$$d' = \frac{d(Y'_{i_n})^j (Y'_{i_0})^{j-1} (Y'_{i_1})^{j-2}}{Y_{i_n}^j Y_{i_0}^{j-1} Y_{i_1}^{j-2}}.$$

To validate the above computations, one just needs to literally implement Equation (2) by noting the changes from $X_{i_0}, Y_{i_0}, Y_{i_1}, Y_{i_n}$ to $X'_{i_0}, Y'_{i_0}, Y'_{i_1}, Y'_{i_n}$, respectively.

3) Using d', each receiver extracts the new session key k' from c' by computing $k' = e(d', c')$.

Note that in this case only members $\mathcal{U}_1$ and $\mathcal{U}_n$ need to communicate with other members. The computation for all members and the sender is of constant complexity.

Long-term secret key update. The third level is to update the secret key x_i of user $\mathcal{U}_i$. This is needed if the user's public key expires or is compromised. Assume that user $\mathcal{U}_{i_j}$ would like to update her secret key x_{i_j}. Then $\mathcal{U}_{i_j}$ needs to run the `KeyGen` algorithm to generate a new public-secret key pair (X'_{i_j}, x'_{i_j}) and register her new public key at the certificate authority. The sender replaces $\mathcal{U}_{i_j}$'s old public key X_{i_j} with the new one X'_{i_j}. Then the sender and all the receivers can run `Encryption` and `Decryption`, respectively, exactly as shown in Section III-B. However, by exploiting previously computed values, the encryption and decryption algorithms can be equivalently implemented in a simplified way for better performance without full repetition of each step.

`Encryption`. The sender does the following.

1) Randomly select $r', x'_{i_0} \in \mathbb{Z}_p^*$ and compute:

$$X'_{i_0} = g^{x'_{i_0}}, Y'_{i_0} = (X_{i_1}/X_{i_n})^{x'_{i_0}}, c' = g^{r'}.$$

2) According to Equation (1), compute for $\mathbb{S}$ the public broadcast encryption key:

$$K' = \begin{cases} \frac{Ke(X_{i_{j-1}}, X'_{i_j}) e(X'_{i_j}, X_{i_{j+1}})}{e(X_{i_{j-1}}, X_{i_j}) e(X_{i_j}, X_{ij+1})} & \text{if } j \neq 1, n \\ \frac{Ke(X'_{i_1}, X_{i_2})}{e(X'_{i_1}, X_{i_2})} & \text{if } j = 1 \\ \frac{Ke(X_{i_{n-1}}, X'_{i_n})}{e(X_{i_{n-1}}, X_{i_n})} & \text{if } j = n \end{cases}$$

3) Compute

$$S' = \begin{cases} K'e(X_{i_n}, X'_{i_0})e(X'_{i_0}, X_{i_1}) & \text{if } j \neq 1, n \\ K'e(X_{i_n}, X'_{i_0})e(X'_{i_0}, X'_{i_1}) & \text{if } j = 1 \\ K'e(X'_{i_n}, X'_{i_0})e(X'_{i_0}, X_{i_1}) & \text{if } j = n \end{cases}$$

4) Compute the new secret session key $k' = (S')^r$.
5) Broadcast the header $Hdr' = (X'_{i_0}, Y'_{i_0}, c')$ to the receivers.

`Decryption.` The intended receivers run this algorithm as follows.
1) Receiver $\mathcal{U}_{i_j}$ and her two neighboring receivers $\mathcal{U}_{i_{j-1}}, \mathcal{U}_{i_{j+1}}$ respectively publish

$$Y'_{i_j} = \begin{cases} (X_{i_{j+1}}/X_{i_{j-1}})^{x'_{i_j}} & \text{if } j \neq 1, n \\ (X_{i_2}/X'_{i_0})^{x'_{i_1}} & \text{if } j = 1 \\ (X'_{i_0}/X_{i_{n-1}})^{x'_{i_n}} & \text{if } j = n \end{cases}$$

$$Y'_{i_{j-1}} = \begin{cases} (X'_{i_j}/X_{i_{j-2}})^{x_{i_{j-1}}} & \text{if } j \neq 1, 2 \\ (X'_{i_2}/X'_{i_0})^{x_{i_1}} & \text{if } j = 2 \\ Y'_{i_0} & \text{if } j = 1 \end{cases}$$

$$Y'_{i_{j+1}} = \begin{cases} (X_{i_{j+2}}/X'_{i_j})^{x_{i_{j+1}}} & \text{if } j \neq n, n-1 \\ (X'_{i_0}/X'_{i_j})^{x_{i_n}} & \text{if } j = n-1 \\ Y'_{i_0} & \text{if } j = n \end{cases}$$

Receiver $\mathcal{U}_{i_1}$ and $\mathcal{U}_{i_n}$ respectively publish

$$Y'_{i_1} = (X_{i_2}/X'_{i_0})^{x_{i_j}}, Y'_{i_n} = (X'_{i_0}/X_{i_{n-1}})^{x_{i_j}}.$$

In the above, the subscript j of i_j is computed modulo $n+1$.
2) According to Equation (2), the receiver (indexed by i_j) who updates her long-term secret key can compute the new group decryption key as

$$d' = \frac{dX_{i_{j-1}}^{(n+1)x'_{i_j}}(Y'_{i_j})^n(Y'_{i_{j+1}})^{n-1}(Y'_{i_n})^j(Y'_{i_0})^{j-1}(Y'_{i_1})^{j-2}}{X_{i_{j-1}}^{(n+1)x_{i_j}}Y_{i_j}^n Y_{i_{j+1}}^{n-1}Y_{i_n}^j Y_{i_0}^{j-1}Y_{i_1}^{j-2}}.$$

Receiver $\mathcal{U}_{i_{j+1}}$ can compute d' as

$$d' = \frac{d(X'_{i_j})^{(n+1)x_{i_{j+1}}}(Y'_{i_{j+1}})^n(Y'_{i_n})^{j+1}(Y'_{i_0})^j Y'_{i_{j-1}}}{X_{i_j}^{(n+1)x_{i_{j+1}}}Y_{i_{j+1}}^n Y_{i_n}^{j+1}Y_{i_0}^j Y_{i_{j-1}}}.$$

Receiver $\mathcal{U}_{i_1}$ can compute d' as

$$d' = \frac{d(X'_{i_1})^{(n+1)x_{i_0}}(Y'_{i_1})^n(Y'_{i_{j-1}})^{n-j+2}(Y'_{i_j})^{n-j+1}(Y'_{i_{j+1}})^{n-j}Y'_{i_n}}{X_{i_0}^{(n+1)x_{i_1}}Y_{i_1}^n Y_{i_{j-1}}^{n-j+2}Y_{i_j}^{n-j+1}Y_{i_{j+1}}^{n-j}Y_{i_n}}.$$

Other receivers $\mathcal{U}_{i_t}(t \neq 1, j, j+1)$ can compute the new group decryption key d' as

$$d' = \frac{d(Y'_{i_{j-1}})^{n+t-j+1}(Y'_{i_j})^{n+t-j}(Y'_{i_{j+1}})^{n+t-j-1}(Y'_{i_n})^t(Y'_{i_0})^{t-1}}{Y_{i_{j-1}}^{n+t-j+1}Y_{i_j}^{n+t-j}Y_{i_{j+1}}^{n+t-j-1}Y_{i_n}^t Y_{i_0}^{t-1}}.$$

Similarly, the subscript j of i_j is computed modulo $n+1$.
3) Using d', each receiver extracts the session key k' from c' by computing $k' = e(d', c')$.

From the above, it can be seen that updating the long-term secret key of a member causes more overhead than updating her session key or her group decryption key, although the long-term secret key update process described is still much more efficient than a completely new run of the protocol. This is reasonable, because the long-term secret key is the one that should be changed least often; each member should keep its long-term key secure to reduce unwanted burden to other members.

V. Performance Evaluation

A. Theoretical Analysis

The key generation requires only one exponentiation for each user and is very efficient. The following considers the cost for encryption and decryption in various settings.

We first consider the cost for the first run of our protocol. Let M denote the cost of a multiplication, D the cost of a division, P the cost of a bilinear map, and E and E_T the costs of an exponentiation in $\mathbb{G}$ and $\mathbb{G}_T$, respectively; for the multiplication and division costs, no distinction is made between $\mathbb{G}$ and $\mathbb{G}_T$. The sender needs $3E + 1E_T + \lceil \frac{n+1}{2} \rceil P + 1D + (n+1)M$ in computation and three elements ($O(\log p)$ bits) in $\mathbb{G}$ in communication, noting that $e(X_{i_{j-1}}, X_{i_j})e(X_{i_j}, X_{i_{j+1}}) = e(X_{i_j}, X_{i_{j-1}}X_{i_{j+1}})$. An exponentiation in a group of order p needs about $1.5 \log p$ multiplications. In our motivated applications, n is usually less than $2^8 = 256$. Here p is typically suggested to be at the level of 2^{160} in bilinear map based cryptography[1]. In the suggested security parameter setting, $1E$ requires about $240M$ on average. Then $(n+1)M$ have the same complexity as $1E$. A bilinear map is more time-consuming than an exponentiation but in the same order in complexity. Similarly, a division is also less efficient than a multiplication but still in the same order. Accordingly, the sender needs about $O(n)E$ in computation and $O(\log p)$ bits in communication for the first run of the protocol.

At the receiver side, each receiver's secret key is one group element of about $O(\log p)$ bits. Each receiver requires $1E+1D$ in Step 1, $1E + (1 + 1.5 \log n)nM$ in Step 2 and $1P$ in Step 3, respectively. Each receiver needs to broadcast $O(\log p)$ bits in Step 1. Accordingly, each receiver needs about $O(\log n)E$ in computation and $O(\log p)$ bits in communication.

In the case that a member is deleted, the sender needs $3E + 1E_T + 3P + 5M + 1D$ in computation and $O(\log p)$ bits in communication. At the receiver side for decryption, $\mathcal{U}_{i_{j-1}}$ and $\mathcal{U}_{i_j}$, respectively, need $1E + 1D$ in computation and $\log p$ bits in communication in Step 1. Each member needs at most $1E + (1 + 1.5 \log(n-1))nM$ in computation in Step 2 and $1P$ in Step 3. Hence, both the sender and each member need $O(1)E$ in computation, and the sender and only two members need additionally $O(\log p)$ bits in communication.

To enroll a new member, the sender needs $3E+1E_T+3P+7M+1D$ in computation and $O(\log p)$ bits in communication. At the receiver side for decryption, only $\mathcal{U}_{i_{j-1}}$, $\mathcal{U}_{i^*}$ and $\mathcal{U}_{i_{j+1}}$, respectively, need $1E + 1D$ in computation and $O(\log p)$ bits

[1]This setting provides a security level of 2^{80}, *i.e.*, the security level of RSA-1024, which is affordable in mobile devices. A higher level of security is possible by setting larger p and $\mathbb{G}_T$, which implies less efficiency in both computation and bandwidth consumption but does not affect the round efficiency. A detailed trade-off between security and computation/bandwidth cost can be found in [38].

TABLE 1.
COMPLEXITY OF OUR PROTOCOL IN DIFFERENT SCENARIOS

	Computation		Communication		Round
	Sender	Member	Sender	Member	Member
FR	$O(n)$	O(log n)	$O(\log p)$	$O(\log p)$	1
MD	$O(1)$	$O(1)$	$O(\log p)$	$O(\log p)$ or 0	1
MA	$O(1)$	$O(1)$	$O(\log p)$	$O(\log p)$ or 0	1
SKU	$O(1)$	$O(1)$	$O(\log p)$	0	0
GDKU	$O(1)$	$O(1)$	$O(\log p)$	$O(\log p)$ or 0	1
LTKU	$O(1)$	$O(1)$	$O(\log p)$	$O(\log p)$ or 0	1

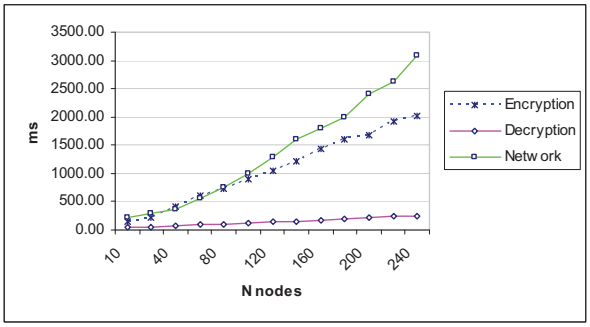

Fig. 5. Cost of the first run of the protocol

in communication. Each member needs at most $1E + (1 + 1.5\log(n+1))nM$ in Step 2 and $1P$ in Step 3. By summing all factors, both the sender and each receiver need $O(1)E$ in computation. Only the sender and three members need an extra $O(\log p)$ bits in communication. The other members do not need any interaction.

We then consider the overhead to update a session key for a fixed group. The sender needs $2E$ in computation and about $O(\log p)$ bits in communication. Each receiver needs $1P$ in computation and no communication with other receivers is necessary.

The group decryption key may be updated after several sessions. In this case, the sender needs $3E+1E_T+1P+1M$ in computation, and $O(\log p)$ bits in communication. At the receiver side, $\mathcal{U}_1$ and $\mathcal{U}_n$, respectively, need $1E$ and $O(\log p)$ bits in Step 1. In Step 2, $\mathcal{U}_1$ needs $1E+3D+(3+1.5\log n)M$, and the other receivers need $(4+3\log n)M$. In Step 3, each member needs $1P$. By summing them, $\mathcal{U}_1$ needs $2E + 3D + (3 + 1.5\log n)M + 1P$; $\mathcal{U}_n$ needs $1E + (3+1.5\log n)M + 1P$ and other members need $(3 + 1.5\log n)M + 1P$. Approximately, both the sender and each member need $O(1)$ exponentiations in computation; the sender and only two members $\mathcal{U}_1$ and $\mathcal{U}_n$, respectively, need $O(\log p)$ bits in communication.

Finally, we consider the cost of updating some member's long-term secret key. In this case, the sender needs at most $3E+1E_T+2P+2D+2M$ in computation and about $O(\log p)$ bits in communication. At the receiver side, $\mathcal{U}_{i_{j-1}}$, $\mathcal{U}_{i_j}$, $\mathcal{U}_{i_{j+1}}$, $\mathcal{U}_{i_1}$ and $\mathcal{U}_{i_n}$ respectively need $1E + 1D$ and $O(\log p)$ bits in Step 1. All members respectively need at most $5D+1E+(5+6\log n)M$ in Step 2 and $1P$ in Step 3. Taking all factors into account, during the decryption procedure each member needs $O(1)$ exponentiations in computation and only members $\mathcal{U}_{i_{j-1}}$, $\mathcal{U}_{i_j}$, $\mathcal{U}_{i_{j+1}}$, $\mathcal{U}_{i_1}$ and $\mathcal{U}_{i_n}$ need $O(\log p)$ bits in communication.

The complexity in various scenarios is summarized in Table 1 using asymptotical notations with a distinction of E and E_T for clarity, where FR represents the first run of our protocol, MD a member deletion, MA a member addition, SKU the session key update, GDKU the group decryption key update and LKTU the long-term key update for one member. The computation and communication are respectively measured in exponentiations and bits. From the table, it is seen that, after the first run of the protocol, the subsequent runs have almost constant complexity independently of the group size. This is very desirable in *ad hoc* networks where members may join and leave, or some member's key might be compromised. Also, one should note that although our protocol needs one-round interaction for decryption in the cases of member changes or update of the group decryption key or long-term keys of members, only very few (less than four) members are involved in the interaction. This is also convenient in practice.

B. Experimental Results

In this section, we encode our new key management protocol and perform simulations in the context of MANETs. The simulations are run on a laptop using an Intel Core i7-2620M at a frequency of 2.7 GHz. Nevertheless, we simulate the system in a virtual machine that uses only 2 of the 4 cores of the chip. The experimental results are highly consistent with the theoretical analysis and show that our protocol is especially efficient in coping with member changes and the rekeying issues usual in various MANETs.

By using NS-3 [39], we carried out the simulations to evaluate the network delay of our protocol. We have to differentiate between remote communication and local communication. We situated a sender in an American server (remote communication) and measured the time that a 192 bytes packet, *i.e.*, the header Hdr, takes from the remote sever to our university in Tarragona. In local communication, we evaluated the protocol in the wireless environment. We used the IEEE 802.11b protocol, very common in mobile ad hoc networks. The channel bandwidth bound used were 11 Mbps. The number of protocol participants was chosen from 10 to 240. The nodes were scattered in a grid layout, where each pair of nodes were separated by 100 meters.

In the simulation, we used the Pairing-Based Cryptography (PBC) library [40]. We generated a symmetric pairing constructed on the curve $y^2 = x^3 + x$ with characteristic a 512-bit prime and embedding degree 2, *i.e.*, the Type A pairings suggested in [40]. The order of $\mathbb{G}$ over the curve is a prime of 20 bytes. Here we did not optimize the underlying pairing-related parameters or operations, e.g., by choosing the large prime characteristic of the base field and the prime order p with most bits 0 (or 1), and by accelerating multi-base exponentiations/multi-base pairings with pre-computation [41]. Hence, the practical performance of our protocol can be even better than the illustrated experimental results.

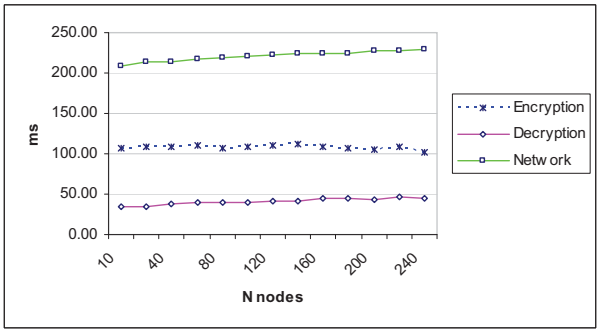

Fig. 6. Cost of member deletion

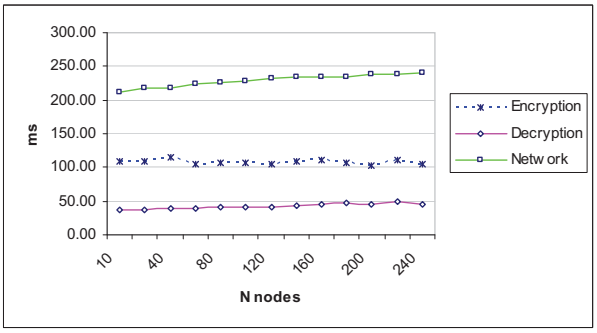

Fig. 7. Cost of member addition

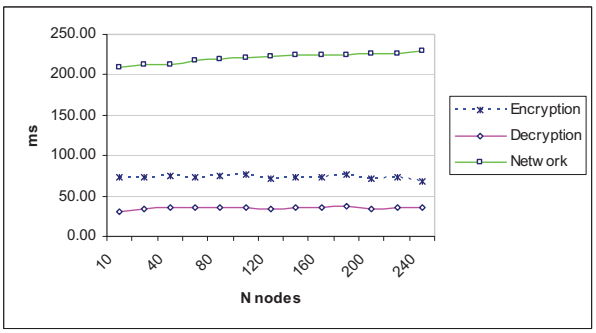

Fig. 8. Cost of group decryption key update

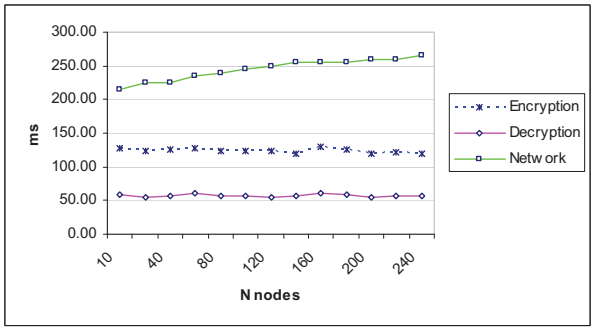

Fig. 9. Cost of long-term secret key update

The experimental results are illustrated in Figures 5-9. It can be seen in Figure 5 that the time delay introduced by group decryption (excluding the interactions for decryption) is really low. The cost of the encryption to the group grows linearly with the number of the receivers due to the linear number of bilinear map operations. The network delay grows faster than the delay incurred by encryption and decryption when there are more than 100 members. However, even for a remote group with 240 members, the total delay is about only five seconds. This is bearable for a sender to transmit to a remote group organized and connected in an *ad hoc* way. Note that the greatest delay is caused by the network; this result highlights the importance of reducing the number of communication rounds to cope with member changes in key management protocols designed for MANETs, as we have done in this work.

Experiments show that our protocol can cope with member changes and key updates in an efficient way. According to Figures 6 and 7, member deletion and addition have very similar cost, much smaller than the cost of the basic protocol. This feature is desirable in practice, since members may leave and join a MANET. From Figures 8 and 9, updating a group decryption key or the long-term key of a member has a similar cost as deleting or adding a member. Among these two update operations, group decryption key update is slightly more efficient due to less time spent in group encryption and decryption. The network time of these two operations is similar and accounts for the most substantial delay. However, in both cases, the total delay is less than 500ms, which is affordable in practice.

In addition to the above remarkable performance, our new key management paradigm has also structural advantages over existing paradigms. Compared with group key agreement, our approach does not require a remote sender to simultaneously stay online with the receivers. This makes possible the desirable send-and-leave pattern for the senders. Compared with broadcast encryption, our approach does not require a fully trusted key server and is easy to be deployed in practice.

VI. Conclusion

We have proposed a new key management paradigm to enable send-and-leave broadcasts to remote cooperative groups without relying on a fully trusted third party. Our scheme has been proven secure in the standard model. A thorough complexity analysis and extensive experiments show that our proposal is also efficient in terms of computation and communication. These features render our scheme a promising solution to group-oriented communication with access control in various types of *ad hoc* networks.

Acknowledgments

This paper is partly supported by the EU FP7 through projects “DwB” and “Inter-Trust”, the Spanish Government through projects CTV-09-634, PTA2009-2738-E, TSI-020302-2010-153, TIN2009-11689, TIN2011-27076-C03-01, CONSOLIDER INGENIO 2010 “ARES” CSD2007-0004 and

TSI2007-65406-C03-01, by the Catalonia Government under grant SGR2009-1135, and by the NSF of China under grants 60970114, 60970115, 91018008, 60970116, 61173154, 61003214 and 61173192. We also acknowledge support by the Fundamental Research Funds for the Central Universities of China through Project 3103004, and Shaanxi Provincial Education Department through Scientific Research Program 2010JK727. The fourth author is partially supported as an ICREA-Acadèmia researcher by the Catalan Government. The authors are with the UNESCO Chair in Data Privacy, but this paper does not necessarily reflect the position of UNESCO nor does it commit that organization.

REFERENCES

[1] Y. Zhang and Y. Fang, "ARSA: An Attack-Resilient Security Architecture for Multi-Hop Wireless Mesh Networks," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 10, pp. 1916-1928, Oct. 2006.

[2] K. Ren, S. Yu, W. Lou and Y. Zhang, "PEACE: A Novel Privacy-Enhanced Yet Accountable Security Framework for Metropolitan Wireless Mesh Networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 2, pp. 203-215, Feb. 2010.

[3] B. Rong, H.-H. Chen, Y. Qian, K. Lu, R. Q. Hu and S. Guizani, "A Pyramidal Security Model for Large-Scale Group-Oriented Computing in Mobile Ad Hoc Networks: The Key Management Study," *IEEE Trans. Veh. Technol.*, vol. 58, no. 1, pp. 398-408, Jan. 2009.

[4] Y-M. Huang, C.-H. Yeh, T.-I. Wang and H.-C. Chao, "Constructing Secure Group Communication over Wireless Ad Hoc Networks Based on a Virtual Subnet Model," *IEEE Wireless Comm.*, vol. 14, no. 5, pp. 71-75, Oct. 2007.

[5] Q. Wu, J. Domingo-Ferrer and U. González-Nicolás, "Balanced Trustworthiness, Safety and Privacy in Vehicle-to-vehicle Communications," *IEEE Trans. Veh. Technol.*, vol. 59, no. 2, pp. 559-573, Feb. 2010.

[6] L. Zhang, Q. Wu, A. Solanas and J. Domingo-Ferrer, "A Scalable Robust Authentication Protocol for Secure Vehicular Communications," *IEEE Trans. Veh. Technol.*, vol. 59, no. 4, pp. 1606 - 1617, May 2010.

[7] K. Sampigethaya, M. Li, L. Huang and R. Poovendran, "AMOEBA: Robust Location Privacy Scheme for VANET," *IEEE J. Sel. Areas Commun.*, vol. 25, no. 8, pp. 1569-1589, Oct. 2007.

[8] M. Burmester and Y. Desmedt, "A Secure and Efficient Conference Key Distribution System," in *Advances in Cryptology–EUROCRYPT'94*, LNCS, vol. 950, pp. 275-286, 1995.

[9] M. Waldvogel, G. Caronni, D. Sun, N. Weiler and B. Plattner, "The VersaKey Framework: Versatile Group Key Management," *IEEE J. Sel. Areas Commun.*, vol. 17, no. 9, pp. 1614-1631, Sept. 1999.

[10] M. Steiner, G. Tsudik and M. Waidner, "Key Agreement in Dynamic Peer Groups," *IEEE Trans. Parallel Distrib. Syst.*, vol. 11, no. 8, pp. 769-780, Aug. 2000.

[11] A. Sherman and D. McGrew, "Key Establishment in Large Dynamic Groups Using One-way Function Trees," *IEEE Trans. Software Eng.*, vol. 29, no. 5, pp. 444-458, May 2003.

[12] Y. Amir, Y. Kim, C. Nita-Rotaru, J. L. Schultz, J. Stanton, and G. Tsudik, "Secure Group Communication Using Robust Contributory Key Agreement," *IEEE Trans. Parallel Distrib. Syst.*, vol. 15, no. 5, pp. 468-480, May 2004.

[13] Y. Kim, A. Perrig and G. Tsudik, "Tree-Based Group Key Agreement," *ACM Trans. Inf. Syst. Security*, vol. 7, no. 1, pp. 60-96, Feb. 2004.

[14] Y. Sun, W. Trappe and K.J.R. Liu, "A Scalable Multicast Key Management Scheme for Heterogeneous Wireless Networks," *IEEE/ACM Trans. Netw.*, vol. 12, no. 4, pp. 653-666, Aug. 2004.

[15] W. Trappe, Y. Wang and K.J.R. Liu, "Resource-Aware Conference Key Establishment for Heterogeneous Networks," *IEEE/ACM Trans. Netw.*, vol 13, no 1, pp.134-146, Feb. 2005.

[16] P. P. C. Lee, J. C. S. Lui and D. K. Y. Yau, "Distributed Collaborative Key Agreement and Authentication Protocols for Dynamic Peer Groups," *IEEE/ACM Trans. Netw.*, vol. 14, no. 2, pp. 263-276, April 2006.

[17] Y. Mao, Y. Sun, M. Wu and K. J. R. Liu, "JET: Dynamic Join-Exit-Tree Amortization and Scheduling for Contributory Key Management," *IEEE/ACM Trans. Netw.*, vol 14, no 5, pp.1128-1140, Oct. 2006.

[18] W. Yu, Y. Sun and K. J. R. Liu, "Optimizing the Rekeying Cost for Contributory Group Key Agreement Schemes," *IEEE Trans. Dependable and Secure Computing*, vol. 4, no. 3, pp. 228 - 242, July-Sep. 2007.

[19] R. Dutta and R. Barua, "Provably secure constant round contributory group key agreement in dynamic setting," *IEEE Trans. Inf. Theory*, vol. 54, no. 5, pp. 2007-2025, May 2008.

[20] J. Snoeyink, S. Suri and G. Varghese, "A Lower Bound for Multicast Key Distribution," in *Proc. INFOCOM'01*, 2001, pp. 422-431.

[21] I. Ingemarsson, D.T. Tang and C.K. Wong, "A Conference on Key Distribution System," *IEEE Trans. Inf. Theory*, vol. 28, no. 5, pp. 714-720, Sep. 1982.

[22] M. Abdalla, Y. Shavitt and A. Wool, "Key Management for Restricted Multicast Using Broadcast Encryption," *IEEE/ACM Trans. Netw.*, vol. 8, no. 4, pp. 443-454, Aug. 2000.

[23] B. M. Macq and J.-J. Quisquater, "Cryptology for Digital TV Broadcasting," *Proc. IEEE*, vol. 83, no. 6, pp. 944-957, Jun. 1995.

[24] J. Lotspiech, S. Nusser and F. Pestoni, "Anonymous Trust: Digital Rights Management Using Broadcast Encryption," *Proc. IEEE*, vol. 92, no. 6, pp. 898-909, June 2004.

[25] A. Fiat and M. Naor, "Broadcast Encryption," in *Advances in Cryptology–CRYPTO'93*, LNCS, vol. 773, pp. 480-491, 1993.

[26] C. K.Wong, M. Gouda and S. Lam, "Secure Group Communications Using Key Graphs," *IEEE/ACM Trans. Netw.*, vol. 8, no. 1, pp. 16-30, Feb. 2000.

[27] D. Halevi and A. Shamir, "The LSD Broadcast Encryption Scheme," in *Advances in Cryptology–CRYPTO'02*, LNCS, vol. 2442, pp. 47-60, 2002.

[28] J. H. Cheon, N.-S. Jho, M.-H. Kim and E. S. Yoo, "Skipping, Cascade, and Combined Chain Schemes for Broadcast Encryption," *IEEE Trans. Inf. Theory*, vol. 54, no. 11, pp. 5155-5171, Nov. 2008.

[29] M. Naor and B. Pinkas, "Efficient Trace and Revoke Schemes," in *Proc. 4th International Conf. on Financial Cryptography (FC'00)*, LNCS, vol. 1962, pp. 1-20, 2001.

[30] D. Boneh, C. Gentry and B. Waters, "Collusion Resistant Broadcast Encryption with Short Ciphertexts and Private Keys," in *Advances in Cryptology–CRYPTO'05*, LNCS, vol. 3621, pp. 258-275, 2005.

[31] J.-H. Park, H.-J. Kim, M.-H. Sung and D.-H. Lee, "Public Key Broadcast Encryption Schemes With Shorter Transmissions," *IEEE Trans. on broadcasting*, vol. 54, no. 3, pp. 401-411, Sep. 2008.

[32] C. Gentry and B. Waters, "Adaptive Security in Broadcast Encryption Systems (with Short Ciphertexts)," *Advances in Cryptology–EUROCRYPT'09*, LNCS, vol. 5479, pp. 171-188, Springer-Verlag, 2009.

[33] Q. Wu, Y. Mu, W. Susilo, B. Qin and J. Domingo-Ferrer, "Asymmetric Group Key Agreement," in *Advances in Cryptology–EUROCRYPT'09*, LNCS, vol. 5479, pp. 153-170, 2009.

[34] Q. Wu, B. Qin, L. Zhang and J. Domingo-Ferrer, O. Farràs, "Bridging Broadcast Encryption and Group Key Agreement," in *Advances in Cryptology–ASIACRYPT'11*, LNCS, vol. 7073, pp. 143-160, 2011.

[35] E. Fujisaki and T. Okamoto, "Secure Integration of Asymmetric and Symmetric Encryption Schemes," in *Advances in Cryptology–CRYPTO'99*, LNCS, vol. 1666, pp. 537-554, 1999.

[36] D. Boneh and M. Franklin, "Identity Based Encryption from the Weil Pairing," *SIAM J. of Computing*, vol. 32, no. 3, pp. 586-615, 2003.

[37] E. Bresson, Y. Lakhnech, L. Mazaré and B. Warinschi, "A Generalization of DDH with Applications to Protocol Analysis and Computational Soundness," in *Advances in Cryptology–CRYPTO'07*, LNCS, vol. 4622, pp. 482-499, 2007.

[38] N. Koblitz and A. Menezes, "Pairing-Based Cryptography at High Security Levels," in *Cryptography and Coding–IMA 2005* LNCS, vol. 3796, pp. 13-36, 2005.

[39] NS-3 Simulator, http://www.nsnam.org/, 2011.

[40] PBC Library, http://crypto.stanford.edu/pbc/, 2011.

[41] M. Scott, "On the Efficient Implementation of Pairing-Based Protocols," http://eprint.iacr.org/2011/334.pdf, 2011.

Qianhong Wu received his M.Sc in applied mathematics from Sichuan University in 2001 and his Ph.D. in Cryptography from Xidian University in 2004. Since then, he has been with Wollongong University (Australia) as an associate research fellow, with Wuhan University (China) as an associate professor, and with Universitat Rovira i Virgili (Catalonia) as a research director. His research interests include cryptography, information security and privacy, and *ad hoc* network security. He has been a holder/co-holder of 6 China/Australia/Spain funded projects. He has authored 3 patents and over 85 publications. He has served in the program committee of several international conferences in information security and privacy. He is a member of IACR and IEEE.

Bo Qin received her Ph.D. degree in Cryptography from Xidian University in 2008 in China. Since then, she has been with Xi'an University of Technology (China) as a lecturer, and with Universitat Rovira i Virgili (Catalonia) as a postdoctoral researcher. Her research interests include pairing-based cryptography, data security and privacy, and VANET security. She has been a holder/co-holder of 5 China/Spain funded projects. She has authored over 45 publications and served in the program committee of several international conferences in information security.

Lei Zhang is an associate research fellow in the Software Engineering Institute, East China Normal University. Before this, he had been a postdoctoral researcher at Universitat Rovira i Virgili (Catalonia). He got his Ph.D. degree in Computer Engineering from Universitat Rovira i Virgili in 2010. His fields of activity are information security, cryptography, e-commence security, data privacy and network security. He has authored over 30 publications.

Josep Domingo-Ferrer is a Full Professor of Computer Science and an ICREA-Acadèmia Researcher at Universitat Rovira i Virgili, Tarragona, Catalonia, where he holds the UNESCO Chair in Data Privacy. His research interests are in data privacy and data security. He received his M. Sc. and Ph. D. degrees in Computer Science from the Autonomous University of Barcelona in 1988 and 1991, respectively. He also holds an M. Sc. in Mathematics. He has won several research and technology transfer awards, including the IEEE Fellow Grade, as well as the "Narcís Monturiol" Medal to the scientific merit and the 1st Edition of the ICREA Acadèmia Prize 2008, both awarded by the Government of Catalonia. He has authored 5 patents and over 280 publications. He has been the co-ordinator of projects funded by the European Union and the Spanish government, among which the CONSOLIDER ARES project on security and privacy, one of Spain's 34 strongest research teams. He has been the PI of US-funded research contracts. He has held visiting appointments at Princeton, Leuven and Rome. He is a co-Editor-in-Chief of *Transactions on Data Privacy*.

Jesús A. Manjón is a computer engineer with the UNESCO Chair in Data Privacy and the CRISES Research Group at the Department of Computer Engineering and Maths at Universitat Rovira i Virgili of Tarragona, Catalonia. He got his B.Sc. in Computer Engineering in 2004 and his M.Sc. in Computer Security in 2008. He has participated in several Spanish-funded projects and he is a co-author of several research publications on security and privacy.