
Volume 30, issues 11–12, 10 September 2007
ISSN 0140-3664
ELSEVIER
computer
communications
Available online at
ScienceDirect
www.sciencedirect.com
Special Issue:
Security on Wireless Ad Hoc and Sensor Networks
Guest Editors: Sghaier Guizani
Hsiao-Hwa Chen
Peter Mueller

Available online at www.sciencedirect.com
ScienceDirect
Computer Communications 30 (2007) 2428–2441
computer communications
www.elsevier.com/locate/comcom

A DoS and fault-tolerant authentication protocol for group communications in ad hoc networks

Yixin Jiang [a], Chuang Lin [a], Minghui Shi [b], Xuemin (Sherman) Shen [b,*], Xiaowen Chu [c]

[a] *Department of Computer Science and Technology, Tsinghua University, Beijing, PR China*
[b] *Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Canada N2L 3G1*
[c] *Department of Computer Science, Hong Kong Baptist University, Hong Kong, PR China*

Available online 1 May 2007

Abstract

In this paper, a novel authentication protocol is proposed, which satisfies both security and reliability requirements for group communications in ad hoc networks. The security features include identity anonymity and location intracability, periodic one-way session key and pseudonym identity refreshment with implicit authentication, dynamic joining and leaving an in-progress communication session, and data encryption. The reliability features include efficient Denial of Service tolerance for broadcasting refreshment messages, fault-tolerance for recovering lost refreshment messages, robustness for resisting the clock skews among member nodes and seamless key switch without disrupting ongoing data transmissions. The performance and security analysis show that the communication and computation overhead of the proposed protocol is similar to the existing one, while the security can be enhanced significantly. The simulation results demonstrate the robustness of the proposed protocol under severe Denial of Service attack and poor wireless channel quality.

Keywords: Authentication protocol; Forward secrecy; DoS-tolerant; Fault-tolerant; Ad hoc group communications

1. Introduction

Wireless ad hoc networks have attracted great intension in both academia and industry due to their unique characteristics and wide application scenarios [1]. They consist of mobile nodes which communicate with each other through wireless medium without fixed infrastructure. The key advantages include easy and fast deployment and decreased dependence on infrastructure, etc. Therefore, wireless ad hoc networks are widely used in emergency operations, such as search and rescue, policing and firefighting, and military use, such as on the battle field, etc. In those applications, group communications, as a growing application area in mobile communications, are preferred in many cases to keep the privacy of information for each onsite units and reduce the wireless traffic load. As shown in Fig. 1, the mobile nodes from two units in the ad hoc network form two communication groups. In such aforementioned applications, there is usually at least one officer leading each unit. We define the corresponding node as commander (*CMD*) node, which takes charge of issuing secret certificate to group communication members.

A secure group communication session guarantees that only legal members share a common key which can be used in the session. The concept of traditional conference key distribution was first proposed in [2], and further studied in [3–8], which is not quite suitable for ad hoc group communications scenario. The protocol in [3] provides a basic secure key distribution protocol for mobile networks. The schemes in [4,5] are for active members to dynamically join or leave an in-progress group session. Two cryptosystems used in the schemes are not friendly for the mobile devices.

* Corresponding author. Tel.: +1 519 888 4567x32691.
E-mail addresses: yxjiang@csnet1.cs.tsinghua.edu.cn (Y. Jiang), clin@csnet1.cs.tsinghua.edu.cn (C. Lin), mshi@bbcr.uwaterloo.ca (M. Shi), xshen@bbcr.uwaterloo.ca (Xuemin (Sherman) Shen), chxw@comp.hkbu.edu.hk (X. Chu).

0140-3664/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2007.04.012

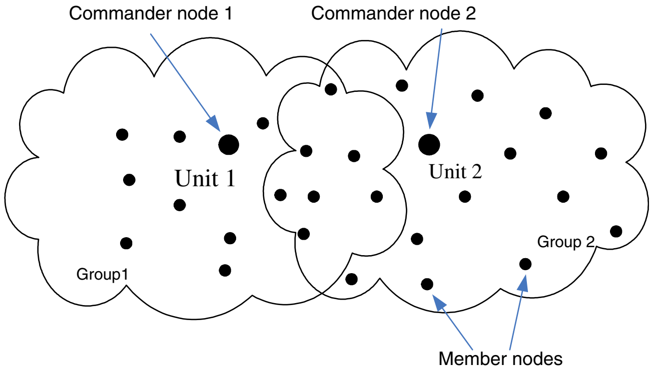


Fig. 1. Ad hoc group communication architecture.

In [6], the protocol does not offer identity anonymity so that an intruder can easily obtain real identity of a member by message interception and trace its mobility and current location. The light-weight protocol [7] lacks key refreshment mechanism so that the communication may be compromised by using a stale key. The impaired ad hoc communication environment and other various attacks from the Internet, such as DoS attacks, need be considered carefully, which otherwise may lead to protocol failure if members cannot communicate with the *CMD* due to communication interruption.

It is important that the confidentiality and authenticity mechanism is available in ad hoc group communications to prevent various illegal intrusions [1,9,10]. The intrusions include traditionally known attacks, such as impersonation, conversation eavesdropping, mobile user's mobility tracing, etc., and newly appeared and more severe attacks, such as Denial of Services (DoS) attack, which can diminish or black out a network's capacity. The main plausible ways for DoS attacks [11] include signal jamming in the physical layer and packet collision/exhaustion in the link layer. In this paper, we focus on the DoS attacks in the link layer.

In this paper, a DoS and fault-tolerant authentication protocol for ad hoc group communications is proposed. Besides resisting common attacks, the proposed protocol features several notable properties:

- identity anonymity to protect a legal member's identity and mobility information from tracking by deploying dynamic identity replacement-pseudo-identity (*PID*);
- forward secrecy so that the communication key (*CK*) and member's *PID* can be refreshed with implicit authentication in a one-way manner;
- dynamic joining or leaving an in-progress group communication;
- DoS-tolerance for broadcasting *CK* renewal message without relying on message retransmissions or acknowledgement (ACKs);
- fault-tolerance by recovering the lost *CK*s;
- seamless *CK&PID* renewal without disrupting ongoing data transmissions;
- robustness to the clock skews among member nodes and the *CMD*.

The proposed protocol also takes into account the resource constraints in the mobile devices by minimizing the computation overhead. Because of its implicit authentication capability of the *CK&PID* refreshment mechanism, the proposed protocol can work well under impaired wireless environment, without using message retransmission or ACKs. Therefore, the communication overhead is lightweight. Demonstrated by the performance analysis and simulation results, the proposed protocol can effectively tolerate high channel loss rate and DoS attacks, which are of particular importance in the emergency and military applications.

The rest of the paper is organized as follows. In Section 2, the authentication protocol with forward secrecy for ad hoc group communications is proposed. In Sections 3 and 4, the security and the performance analysis are presented, respectively, followed by conclusion given in Section 5.

2. Proposed DoS- and fault-tolerant authentication protocol

Fig. 2 shows the messages used in the protocol between the member nodes and the *CMD*. The *InitConfKey* message initiates or re-initiates refreshment parameters. It is sent to all member nodes in the initial phase. The *CMD* uses the *RefreshKey* message to periodically broadcast the next *CK* in the key sequence to member nodes. The member nodes employ the *RequestKey* message to explicitly request the current *CK* in the key sequence. This message is generated by a node when it fails to receive *CK*s over t key renewal intervals. We assume the nodes may also receive forged *CMD* messages sent by attackers.

For the convenience, we list the related notations used in the rest of the paper in Table 1.

2.1. Architecture

The architecture of the proposed protocol consists of four modules: DoS-tolerant module, fault-tolerant module, *CK&PID* switch module, and the stream encryption/

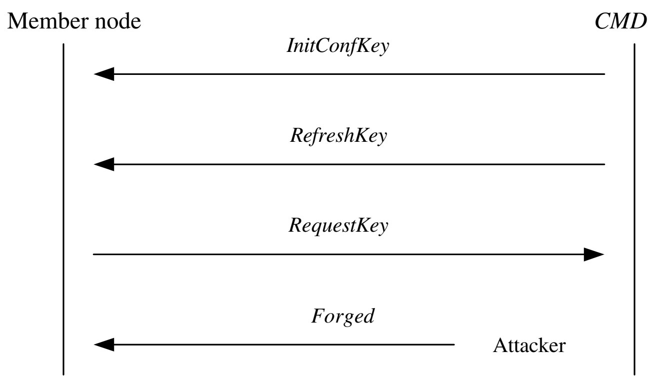


Fig. 2. Message flows between *CMD* and group members.

Table 1
Notations

T_k	Member node
CMD	Commander node
t	Timestamp
ID_k	Identity of mobile member T_k
PID_k	Pseudo-identity of mobile member T_k
CK	Group Communication Key
CK_I	Integrity key derived from CK
CK_E	Data encryption key derived from CK
$\oplus$	Bitwise XOR operation
$\|$	Concatenation operation
$f(.)$	Key generating function
$H(.)$	One-way hash function
$E_k(.)$	Symmetric encryption with key k
$D_k(.)$	Symmetric decryption with key k
MIC	Message integrity code generated by integrity key CK_I
MK	Master key used to encrypt *InitConfKey* and *RefreshKey*

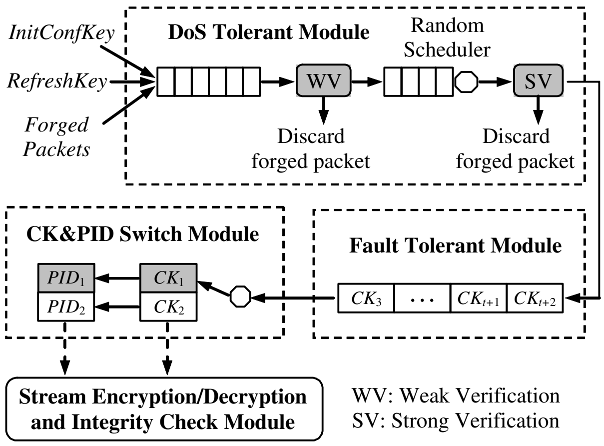


Fig. 3. Architecture of proposed protocol.

decryption and integrity check module. As shown in Fig. 3, the DoS-tolerant module uses two-phase DoS-tolerant authentication: WV (weak verification) and SV (strong verification) to filter out forged packets efficiently. The*CMD* pre-computes key sequence of *CK*s by utilizing a one-way hash function, which is similar to that of S/KEY [12]. Each *CK* is distributed to all member nodes before it is used for encryption or decryption. The authenticity of the received *CK* is verified by using the other pre-stored *CK*s, and the missed *CK*s can be recovered from the new *CK*s. The new *PID* is computed based on the *CK* and its previous *PID*. The detailed description of each module is list as follows.

The DoS-tolerant module protects the *RefreshKey* message from DoS attack by using two-stage verification. *Weak verification* filters out a large number of forged messages by executing fast authentication with simple computation. And *strong verification* executes strict authentication with a little more complex computation to drop a few forged messages which have passed the weak verification phase.

The fault-tolerant module provides a robust and reliable mechanism for tolerating the packet loss in the impaired wireless channel. On receiving an authentic *RefreshKey* message, each member node can automatically recover the lost *CK&PID* without requesting the *CMD* to retransmit the lost message. The fault-tolerant feature relies on the distinctive property of the cryptographic one-way hash function, which is also used in TESLA [13–15] and LiSP [16]. The proposed protocol can improve: (1) efficiency since each member node only buffers the constant number of keys, whereas TESLA is required to buffer all the received control messages until the node receives an authentic message; and (2) reliability since DoS-tolerance mechanism is offered while it is not considered in LiSP protocol.

The *CK&PID* switch module computes the new *PID* and seamlessly refreshes *CK&PID* without disrupting ongoing data transmission. To accomplish the functions, two key-slots, which can be operated concurrently are set up in each member node. When the *CK&PID* in one key-slot is being used for data encryption or decryption, the received new *CK* in the key sequence will be stored in the other key-slot. At the middle point of the refreshment interval, the member node switches to the other key-slot to use the new *CK* key.

Finally, the stream encryption/decryption and integrity check module guarantees data privacy. By considering the dynamic or periodic *CK&PID* refreshment and the fast stream cipher, the proposed protocol provides enhanced security to resist key-stream reuse attacks.

2.2. Forward secrecy

Forward secrecy is used to assure the refreshment of *CK&PID* and offer a base for implementing DoS- and fault-tolerant mechanisms. As shown in Fig. 4, forward secrecy is ensured in three aspects: one-way *CK* refreshment, one-way *PID* refreshment, and one-way data privacy. To ensure the forward secrecy in *CK* refreshment, the proposed architecture offers an *MK* used by the *CMD* to encrypt the *InitConfKey* or the *RefreshKey* message containing the temporal *CK*, which is used to encrypt or decrypt data. Similarly, to assure the forward secrecy in the *PID* renewal, it also defines a master *PID* and a temporal *PID*. The temporal *PID* is derived from master *PID* and its corresponding temporal *CK*. The data privacy is also endowed with the forward secrecy, since we uses the temporal *CK&PID* as the seeds to compute the block cipher and message integrity code. Hence, key-stream collisions can be efficiently avoided due to the forward secrecy.

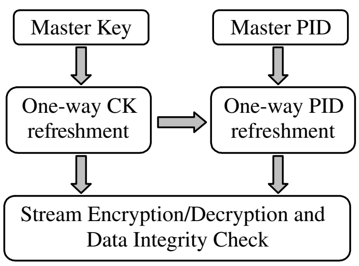


Fig. 4. Forward secrecy: one-way *CK&PID* renewal and data privacy.

Forward secrecy is based on a one-way hash chain, which is generated by a one-way function H. H satisfies two properties: (1) given x, it is easy to compute y such that $y = H(x)$; and (2) given y, it is computationally infeasible to compute x such that $y = H(x)$. A one-way hash chain is a sequence of hash values, $x_n, x_{n-1}, \ldots, x_0$, such that $\forall j{:}0 < j \leqslant n$, $x_{j-1} = H(x_j)$. Thus, there exists the following linear derivative relation:

$$x_1 = H(x_2) = \cdots = H^{j-1}(x_j) = \cdots = H^{n-2}(x_{n-1}) \\ = H^{n-1}(x_n) \quad (1 < j \leqslant n).$$

In the proposed protocol, all temporal *CK*s are also derived from a one-way hash function H and belong to one key sequence. In the initial phase, the *CMD* needs to pre-compute a one-way key sequence $\{CK_i | i = 1, 2, \ldots, n\}$, where n is reasonably large. The *CMD* selects CK_n as the last key in the key chain and repeatedly performs the hash function H to compute all the rest of keys as $CK_i = H(CK_{i+1})$, $0 \leqslant i \leqslant n-1$. Each key CK_i will be distributed to all members by the *CMD* at ith time interval. With this one-way function, given CK_j in the key chain, anybody can compute the previous keys CK_i, $0 \leqslant i \leqslant j$, but cannot compute any of other keys CK_i, $j+1 \leqslant i \leqslant n$. Similarly, all temporal *CK*s also form the following linear derivative relation:

$$CK_1 = H(CK_2) = \cdots = H^{j-1}(CK_j) = \cdots = H^{n-2}(CK_{n-1}) \\ = H^{n-1}(CK_n) \quad (1 < j \leqslant n).$$

Given that the temporal $PID_{k,j}(1 \leqslant j \leqslant n)$ is defined as the function of $PID_{k,j-1}$ and CK_j, for member T_k, the corresponding one-way *PID* chain can be derived as follows:

$$PID_{k,1} \Leftarrow \cdots \Leftarrow PID_{k,j} \Leftarrow \cdots \Leftarrow PID_{k,n-1} \Leftarrow PID_{k,n} \quad (1 < j \leqslant n).$$

Thus, the *PID* can be renewed with *CK* synchronously. Based on the mentioned linear derivative relations, the forward secrecy in the proposed protocol provides three significant security properties: (1) the identity anonymity mechanism is enhanced, since a dynamic *PID* can protect a member node's location and mobility information from being tracked more efficiently than a long-term static *PID*; (2) the key-stream reuse attacks are avoided, in which the *CK*&*PID*, used to compute the stream cipher, are renewed by the *CMD* both periodically or dynamically; and (3) forward secrecy leads to a solution to implement the important DoS- and fault-tolerance feature in our protocol, which will be discussed in detail in the following sections.

2.3. Mutual authentication protocol

When a member node (it becomes chairman in this case) intends to start on a group session, it firstly initiates mutual authentication protocol (MAP). In this phase, the *CMD* setups an *MK*, and then uses the *MK* to encrypt the *InitConfKey* message which includes the length t of key buffer for *CK*s, an initial *CK*, and the key refreshment period $T_{refresh}$. The message is securely broadcasted to each node. Then, at each interval $T_{refresh}$, the *CMD* uses *MK* to encrypt a *RefreshKey* message that contains the next *CK* in the pre-computed key sequence. All the refreshment messages will be securely broadcasted or unicasted to each node.

The MAP offers basic identity anonymity. When a member T_i registers with the *CMD*, it submits its identity ID_i to the *CMD*. The *CMD* generates a secret sufficiently long, e.g., 256 bits, random number N_i for each T_i, computes a pseudonym identity PID_i for T_i using Eq. (1), and records the mapping relation of PID_i and N_i ($PID_i \leftrightarrow N_i$).

$$PID_i = h(N_i \| ID_{CMD}) \oplus ID_i \oplus ID_{CMD}, \tag{1}$$

where "$\oplus$" denotes XOR operation, ID_{CMD} is the identity of the *CMD*, and $h()$ is a one-way hash function. Then, the *CMD* delivers PID_i to T_i through a secure channel. With this secret-splitting mechanism, the real identity ID_i is concealed in PID_i and the identity anonymity is ensured. The *CMD* also shares a long-term secret key $s_i = f(ID_i)$ with T_i, where f is a key generating function.

In the following, we describe the MAP according to the order of message exchanges, and discuss the security goals which can be achieved in each message (Fig. 5).

Step 1. The chairman T_1 chooses a random r_1 and computes its long-term key s_1 by $s_1 = f(ID_1)$. Then, T_1 uses s_1 to encrypt $(t_1 \| s_1 \| r_1 \| ID_2 \| \cdots \| ID_m)$ and sends $\{PID_1, E_{s_1}(t_1 \| s_1 \| r_1 \| ID_2 \| \cdots \| ID_m)\}$ to the *CMD*.

Step 2. On receiving the message from T_1, the *CMD* derives the real identity of member T_1 by computing

$$ID_1 = PID_1 \oplus h(N_1 \| ID_{CMD}) \oplus ID_{CMD}. \tag{2}$$

The *CMD* can retrieve corresponding shared key s_1 and decrypt $E_{s_1}(t_1 \| s_1 \| r_1 \| ID_2 \| \cdots \| ID_m)$. Then, the *CMD* verifies the authenticity of s_1 and the timestamp t_1. If it is true, the *CMD* calls the other user ID_i $(i = 2, \ldots, m)$. All the keys s_i $(i = 1, \ldots, m)$ are pre-computed by the *CMD*.

Step 3. Each member T_i, $i = 2, 3, \ldots, m$, does the same as T_1 in Step 1. The member T_i chooses a random r_i, computes the long-term key s_i as $s_i = f(ID_i)$, uses s_i to encrypt $\{t_i \| s_i \| r_i\}$, and sends the message $\{PID_i, E_{s_i}(t_i \| s_i \| r_i)\}$ to the *CMD*.

Step 4. On receiving the message from T_i, the *CMD* extracts the real identity ID_i of member T_i by computing

$$ID_i = PID_i \oplus h(N_i \| ID_{CMD}) \oplus ID_{CMD}. \tag{3}$$

Then, the *CMD* can retrieve corresponding shared key s_i, and further decrypt $E_{s_i}(t_i \| s_i \| r_i)$. Next, the *CMD* checks the authenticity of key s_i and t_i. If it is true, the *CMD* pre-computes a key sequence $\{CK_i | i = 0, 1, 2, \ldots, n\}$ by using a one-way hash function H, where n is chosen to be reasonably large (e.g., 256) and each CK_i satisfies $CK_i = H(CK_{i+1})$, or $CK_i = H^{n-i}(CK_n)$. The

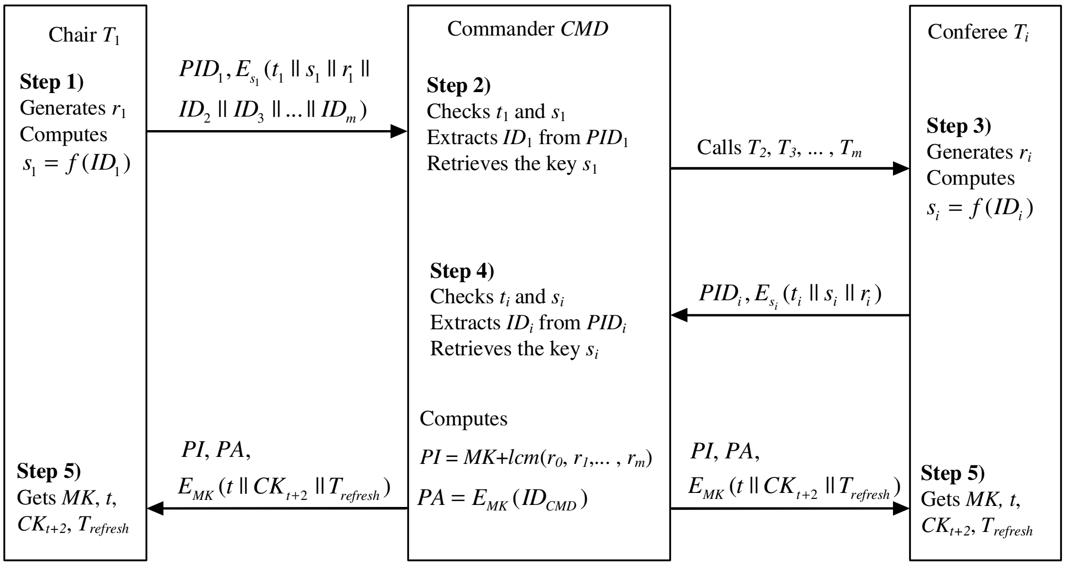


Fig. 5. Mutual authentication protocol for the proposed teleconference.

CMD selects a nonzero random r_0, and computes PI and PA by

$$PI = MK + lcm(r_0, r_1, \ldots, r_m), \quad (4)$$

$$PA = E_{MK}(ID_{CMD}), \quad (5)$$

where lcm(r_0,r_1,…,r_m) denotes the least common multiple function and $MK(=CK_0)$ is the master key of the group session. Then, at time t_{start}, the CMD broadcasts this message to nodes T_i $(i = 1, 2, \ldots, m)$.

$$CMD \rightarrow T_i : \{PI, PA, InitConfKey\},$$

where $InitConfKey$ denotes $E_{MK}(t \| CK_{t+2} \| T_{refresh})$. Note that MK and CK_{t+2} satisfy $MK = CK_0 = H^{t+2}(CK_{t+2})$.

Step 5. According to the received message, T_i gets MK which is given by

$$MK = PI \bmod (r_i). \quad (6)$$

Then T_i verifies the validity of MK by checking if $PA = E_{MK}$ (ID_{CMD}). If it holds, T_i gets $\{t, CK_{t+2}, T_{refresh}\}$ by decrypting $E_{MK}(t \| CK_{t+2} \| T_{refresh})$. The detailed corresponding procedures are given by Algorithm 1.

Fig. 6 shows how the member node copies CK sequence into its key buffer and key-slots, and switches the active-key after receiving CK_{t+2}. Algorithm 2 gives the right-shift process of automatic key renewal at the midpoint of each interval $T_{refresh}$. Each member node maintains two variables e and CK_W. Sentry CK_W tracks the most recently outdated CK, and e traces the number of CK that the node failed to receive correctly.

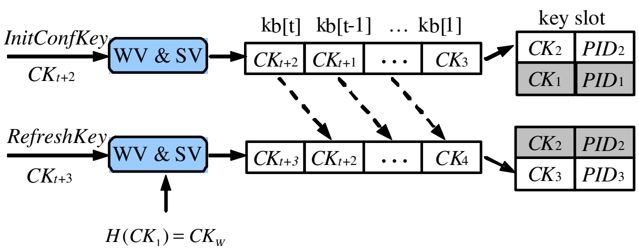


Fig. 6. Initial setup and CK&PID refreshment mechanisms.

Algorithm 1. Initial group communication session parameters

```
function Init_Conf_Key ( ) {
   if (InitConfKey message received) {
      Compute MK = PI mod (r_i), E_MK (ID_CMD);
      if (PA ≠ E_MK(ID_CMD)) return ERROR;
      Decrypt InitConfKey to get {CK_{t+2}, t, T_refresh};
      Allocate a key buffer of length t(kb[1], ..., kb[t]),
        and two key-slots (ks[1], ks[2]);
      for (i = 1; i <= t − 1; i++) do
        kb[i] = H^{t−i}(CK_{t+s});
      ks[2] = H^t(CK_{t+s}), ks[1] = H^{t+1}(CK_{t+s});
      CK_W = H^{t+2}(CK_{t+s});
      Set key ks[1] for data encryption;
      Set RefreshKeyTimer to T_refresh/2;
   }
}
```

Algorithm 2. Refresh key timer

```
function Refresh_Key_Timer ( ) {
   if (RefreshKeyTimer triggered) {
      Right-shift the key buffer and key-slot;
      e++; CK_W = {the inactive key in key-slots};
   }
   Set active CKs in key-slots;
   Set RefreshKeyTimer to T_refresh;
   If (e = = t) send RequestKey message to CMD
}
```

2.4. One-way CK&PID renewal mechanism

Forward secrecy requires that the *CK&PID* be refreshed in a one-way manner. One-way *CK* renewal guarantees that the *CMD* can update the *CK* at regular intervals. One-way *PID* renewal allows each member to renew its *PID* frequently and reduces the risk that it uses a compromised *PID* to communicate with the *CMD*.

If the *RefreshKey* message is broadcast every interval $T_{refresh}$, an intruder may predict when to launch an attack. Thus, it is much easier for the intruder to disrupt such messages by initiating the DoS attacks. To refrain from such attack, the *CMD* should send the *RefreshKey* packets randomly or in a pseudorandom ways that cannot be predicted by an attacker.

Assume that the MAP phase completes at time t_{init}. To resist DoS attack, the *CMD* broadcasts the *RefreshKey* message with CK_{i+t+2} $(i = 0,\ldots,n-t-2)$ for the ith *CK&PID* renewal to all member nodes at the time randomly chosen from the interval $[t_{init} + i \cdot T_{refresh} - \delta, t_{init} + i \cdot T_{refresh} + \delta]$, $\delta < T_{refresh}/4$, i.e.,

$$CMD \rightarrow T_j(j = 1,\ldots,m) : \{CK_i, E_{MK}(CK_{i+t+2} \| CK_{i+1})\},$$

where CK_{i+1} is the active encryption key at the time when the *RefreshKey* message is broadcast, and CK_i is the outdated *CK* in the *CK* sequence. To provide the DoS-tolerant functionality, CK_i is used for weak verification (WV) and $E_{MK}(CK_{i+t+2}\|CK_{i+1})$ is used for strong verification (SV).

On receiving the *RefreshKey* message, each participant deals with this message according to Algorithm 3. Fig. 6 illustrates how to initialize and refresh the *CK&PID*. Due to one-way property of the *CK* sequence, the *RefreshKey* message does not need message authentication code, since the receiver can verify if the received *CK* belongs to the same key sequences as those stored in the key buffer. Such implicit authentication notably decreases the message size.

Algorithm 3. *CK&PID* refreshment for member node

function *Refresh_CK&PID* ( ) {
  while (*RefreshKey* message received) {
    if ($CK_i \neq CK_W$) {/* weak Verification*/
    Discard this message; **continue**;
    }
    Decrypt *RefreshKey* to get CK_{i+t+2};
    $CK_W = \{\textit{the inactive key in key-slots}\}$;
    Right-shift $kb[1] = CK_{i+3}$ to the inactive key-slot;
    Computing $PID_{k,i+3} = H(PID_{k,i+2}\|CK_{i+3})$;
    for $(i = 2; i <= t; i{+}{+})$ **do** $kb[i] \rightarrow kb[i-1]$;
    if $(e \neq 0)$ {/* there are lost *CK*s */
    Recover the lost *CK*s by Algorithm 4;
    $CK_{i+t+2} \rightarrow kb[t]$; $e = 0$;
    }
  }
}

In the following, we discuss the *PID* renewal mechanism. Though in the MAP phase, basic identity anonymity is provided by using PID_i for T_i instead of its real identity ID_i, there are still some security issues to be concerned. For instance, even T_i never reveals ID_i to parties other than the *CMD*, it does reveal its long-term PID_i during the session. Hence, illegal parties can track a member's location by PID_i, although they cannot obtain ID_i.

In the proposed protocol, the *PID* renewal is in progress with the *CK* renewal synchronously. For the jth refreshment, a member T_k can compute its new pseudonym identity $PID_{k,j}$ as

$$PID_{k,j} = H(PID_{k,j-1}\|CK_j), \quad j = 1,2,\ldots,n. \tag{7}$$

Evidently, it will vary with CK_j. Note that $PID_{k,0}$ of T_k is equal to the initial *PID* in the MAP phase, i.e., $PID_{k,0} = PID_k$, $k = 1,2,\ldots,m$. Hence, the *PID* of each member is updated with forward secrecy due to the one-way *CK* renewal.

The computation complexity of the refreshment algorithm is light-weight, since it is only requisite to broadcast the *RefreshKey* messages and perform low-cost hash operations. Periodically refreshing *CK&PID* can also improve the system scalability.

2.5. DoS-tolerant authentication mechanism

The *CK&PID* refreshment scheme relies on the authenticity of the *RefreshKey* messages, which makes the *RefreshKey* messages attractive targets for the DoS attack. An attacker may send a large amount of forged messages to exhaust the nodes' buffer before they can verify the messages, and force them to drop some authentic messages.

An efficient way for an attacker to disrupt the *RefreshKey* message is to jam the communication channel when the *RefreshKey* messages are transmitted. If the attacker can predict the schedule of such messages, it would be much easier for the attacker to disrupt such message transmissions. Thus, the *CMD* is required to send the *RefreshKey* packets randomly or in a pseudorandom ways so that prediction is not feasible.

In the proposed protocol, a packet filter is designed to efficiently verify the *RefreshKey* message, $\{CK_i, E_{MK}(CK_{i+t+2}\|CK_{i+1})\}$. As shown in Fig. 3, in the WV phase, the member nodes perform a fast check to identify the forged messages, and try to discard most unintended forged messages. The "unintended" refers to those random packets used for jam purpose only, and the "intended" refers to those fake packets used for both fraud and jam. Upon receiving the *RefreshKey* message, each node first checks the authenticity of clear-text CK_i in $\{CK_i, E_{MK}(CK_{i+t+2}\|CK_{i+1})\}$. The messages that fail this test are discarded. The computation overhead for the WV is very low. Those intended forged messages that slip through the WV are removed in the SV phase. Compared with the WV, the SV performs strict check with hash computation.

To further improve the possibility that a member node receives authentic *RefreshKey* packets, the node uses a random selection policy to store and authenticate the incoming packets that pass the above weak verification.

Without loss of generality, assume that the length of the buffer at each member node is m. During each time interval $T_{refresh}$, a node can save the first m copies of *RefreshKey* packets that pass the WV. Then, if a new copy is to be kept, the member node randomly selects one of the m buffers and replaces the corresponding copy. For the kth copy ($k > m$), the node keeps it with the probability m/k. It is easy to verify that if a node receives n copies of *RefreshKey* packets, all copies have the same probability m/n to be kept in one of the buffers. The key issue is to make sure that all *RefreshKey* copies have the equal probability to be selected. Otherwise, an attacker who knows the refreshment rule may exploit the unequal probabilities and make a forged *RefreshKey* be chosen with high possibility. Therefore, each member node verifies $E_{MK}(CK_{i+t+2} \| CK_{i+1})$ for at most m times and $(m-1)/2$ on average. With random selection strategy, the probability that a member node receives an authentic *RefreshKey* copy can be estimated as

$$P[\text{RefreshKey packet is authentic}] = 1 - p^m, \tag{8}$$

where

$$p = \frac{\#\text{forged copies}}{\#\text{total copies}}. \tag{9}$$

This indicates that the longer the buffers are, the more effective the random selection algorithm is. Due to the exponential form of (8), a little longer buffer can remarkably improve the reliability of broadcasting the *RefreshKey* messages. To maximize the successful DoS attack, an attacker has to send as many forged copies as possible. Hence, the DoS-tolerant method makes the DoS attack so difficult that the attacker would rather use signal jamming than directly attacking the member nodes.

2.6. Fault-tolerant key recovery mechanism

The *RefreshKey* message can also be used to recover the lost *CK*s for fault-tolerant and key recovery mechanism shown in Algorithm 4. Suppose that there are $r(\leqslant t)$ *CK*s reserved in the key buffer due to previous lost messages, i.e., there are $e = t - r$ empty slots in the key buffer. Let $\{CK'_r, \ldots, CK'_1\}$ denote these r *CK*s in the key buffer $\{kb[r], \ldots, kb[1]\}$, respectively. They also belong to the same key sequence, and satisfy $H(CK'_r) = CK'_{r-1}, \ldots, H(CK'_2) = CK'_1$.

Upon receiving a *RefreshKey* message with CK_k, each node checks if $H(CK_{i+1}) = CK_W$, where CK_W tracks the most recently outdated *CK*. If it is true, the node uses CK_{i+t+2} to recover the lost *CK*s in the same key sequence. Fig. 7 illustrates the recovery of the lost CK(s). Assume that a node receives a *RefreshKey* message with CK_{t+2}. Since $H(CK_{t+2}) = CK_{t+1}$ and $e = 0$, there is no message loss. However, the next two renewal messages are discarded because $H(CK^*_1) \neq CK_W$ and $H(CK^*_2) \neq CK_W$. Thus, there are $t-2$ *CK*s in the key-buffer. The member receives an authentic *RefreshKey* message with CK_{t+5}. Since $H(CK^*_3) = CK_W$, the member can recover the previous two lost *CK*s as $CK_{t+3} = H^2(CK_{t+5})$ and $CK_{t+2} = H^3(CK_{t+5})$.

Algorithm 4. Strong verification & *CK* recovery

function *Recover_CK*( ) {
  if ($H(CK_{i+1}) \neq CK_W$) {/* Strong Verification*/
    Discard *RefreshKey* message $\{CK_i, E_{MK}(CK_{i+t+2} \| CK_{i+1})\}$;
    return FALSE;
  }
  if ($e \geqslant 1$) **for** ($i = 1; i <= e; i++$) **do**
    $\{H^i(CK_k) \to kb[t-i+1];\ e = 0;\}$
  return TRUE;
}

2.7. Dynamic participation mechanism

The dynamic participation mechanism, as a basic security requirement, is that any active participant can join or leave an in-progress group session while assuring the freshness of the key.

Member joining: When a participant T_{m+1} joins an in-progress session, T_{m+1} is required to obtain the permission from the chairman T_1 to join the session. The corresponding actions are described as follows.

Step 1. T_1 sends the *CMD* the message $J = E_{s_1}(ID_{m+1} \| t' \| JOIN)$ with its current PID_1, where t' is timestamp and ID_{m+1} is the identity of the participant T_{m+1}.

Step 2. The *CMD* decrypts J with $s_1(=f(ID_1))$ to obtain t' and ID_{m+1}, and then it checks the validity of

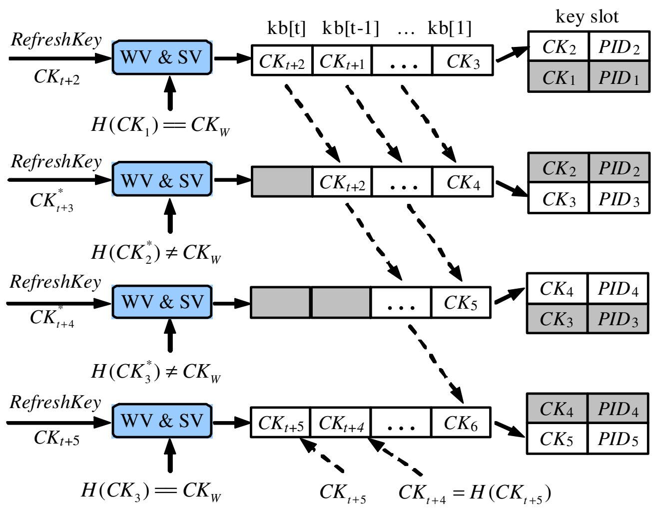


Fig. 7. Fault-tolerant and key recovery mechanisms.

the timestamp t' and ID_{m+1}. If it is true, the CMD calls T_{m+1}.

Step 3. T_{m+1} chooses a random r_{m+1}, and computes the secret key s_{m+1} as $s_{m+1} = f(ID_{m+1})$. It uses s_{m+1} to encrypt $\{t_{m+1}\|s_{m+1}\|r_{m+1}\}$ and sends the message $\{PID_{m+1}, E_{s_{m+1}}(t_{m+1}\|s_{m+1}\|r_{m+1})\}$ to the CMD.

Step 4. On receiving the message from T_{m+1}, the CMD extracts ID_{m+1} as $ID_{m+1} = PID_{m+1} \oplus h(N\|ID_{CMD}) \oplus ID_{CMD}$, computes s_{m+1}, and decrypts $E_{s_{m+1}}(t_{m+1}\|s_{m+1}\| r_{m+1})$ with s_{m+1}. Later, the CMD checks the authenticity of s_{m+1} and t_{m+1}. If it is true, the CMD calculates PI' by

$$PI' = MK + r_{m+1} \cdot s_{m+1},$$

where MK is the main key of the session. Finally, at time $t_{init} + i \cdot T_{refresh}$ (assume that the MAP phase ends at time t_{init}), the CMD sends following message to T_{m+1}

$$CMD \rightarrow T_{m+1} : \{PI', PA, InitConfKey'\},$$

where $InitConfKey' = E_{MK}(t\|CK_{i+t+2}\|T_{refresh})$.

Step 5. T_{m+1} processes the $RefreshKey$ message according to Algorithm 1. That is, T_{m+1} computes MK as $MK = PI' \mathrm{mod}(r_{m+1})$ and verifies its validity by checking if PA is equal to $E_{MK}(ID_{CMD})$. If it holds, T_{m+1} decrypts $E_{MK}(t\|CK_{i+t+2}\|T_{refresh})$ with MK. T_{m+1} joins the session.

Member leaving: When a participant wants to leave an in-progress session, the CMD must update all of the previous CKs to assure the freshness of CK. Assume that member T_q has exited the session. The procedure of updating CK can be depicted as follows:

Step 1. T_1 sends the message $Q = E_{s_1}(ID_q\|t''\|QUIT)$ to the CMD, where ID_q is the identity of T_q.

Step 2. The CMD obtains t'' and ID_q by decrypting Q with s_1, and then it checks the validity of t''. If it is true, the CMD selects a new MK' and further calculates PI' and PA' as:

$$PA' = E_{MK'}(ID_{CMD}),$$
$$PI' = CK'_0 + lcm(r'_0, r'_1, \ldots, r'_{q-1}, r'_{q+1}, \ldots, r'_m),$$

where $r'_i = r_i + t'$, and t' denotes the current time. Then, the CMD broadcasts following message to the remaining members $T_i(i \neq q)$.

$$CMD \rightarrow T_i(i \neq q) : \{PI', InitConfKey'\},$$

where $InitConfKey' = E_{MK'}(t\|CK_{t+s}\|T_{refresh})$, and MK' and CK_{t+2} satisfy $MK' = H^{t+2}(CK_{t+2})$.

Step 3. The rest of members $T_i(i \neq q)$ attain the MK' as $MK' = PI'$ mod $(r_i + t')$ and verify the authenticity of MK' by checking $PA' = E_{MK'}(ID_{CMD})$. If it is true, they get $\{t, CK_{t+2}, T_{refresh}\}$ by decrypting $E_{MK'}(t\|CK_{t+2}\|T_{refresh})$ and then execute the Algorithm 1 to re-initiate the system. The CMD updates the CKs and makes all previous CKs obsolete.

2.8. Re-initialization mechanism

The CMD needs to re-initialize the group session system, if all n CKs in the CK sequence have been used up, or existing member nodes have been compromised, or a node has definitely requested CK since it has missed more than t CKs. In the former two scenarios, all nodes are forced to be re-initialized, while in the third scenario only the requested node needs to be re-initialized.

Specifically, for the first case, the CMD re-computes a new CK sequence $\{CK'_i | i = 1, 2, \ldots, n\}$ and then broadcasts a new $InitConfkey$ message with CK'_{t+s} to all the nodes. For the second case, the procedures of re-initialization are similar to those when a participant leaves an in-progress session. For the third case, theCMD only sends the requesting node an$InitConfkey$ message with the current configuration parameter $\{t\|CK_{i+t+2}\|T_{refresh}\}$. Subsequently, this node can periodically renew the $CK\&PID$ by receiving the $RefreshKey$ message.

2.9. Robustness for clock skews

The proposed protocol is robust to clock skew among the member nodes and the CMD. Let $m_i(T)$ denote the mapping from clock time to real time at node T_i. Then the clock skew between node A and B is denoted as $\lambda = |m_A(T) - m_B(T)|$, where T is clock time. To seamlessly renew the $CK\&PID$, λ should satisfy $\lambda < T_{refresh}/2$, since each member will switch the active key to the new one at the midpoint of the renewal interval.

Assume that MAP ends at time t_{init}. Then at the ith renewal period $[t_{init} + (i - 1/2) \cdot T_{refresh}, t_{init} + (i + 1/2) \cdot T_{refresh}]$, node A uses CK_{i+2} for data encryption while nodes B still uses CK_{i+1} due to the clock skew between A and B. However, they can still successfully communicate with each other during this period, since both A and B hold the same decryption key pair, $\{CK_{i+1}, CK_{i+2}\}$. Therefore, the proposed protocol can ensure the worst case clock skew of $T_{refresh}/2$, For any two member nodes A and B, the timing margin against clock skews should satisfy

$$\max\{|m_A(T) - m_B(T)|, \forall A, B\} < T_{refresh}/2. \quad (10)$$

The proposed protocol can also tolerate the clock skew between the CMD and the member node. For the ith $CK\&PID$ renewal, to resist DoS attacks, the CMD broadcast $RefreshKey$ message at a time randomly chosen from the interval $[t_{init} + i \cdot T_{refresh} - \delta, \quad t_{init} + i \cdot T_{refresh} + \delta]$, $\delta < T_{refresh}/4$. δ should satisfy $\delta < T_{refresh}/4$ for refreshing the $CK\&PID$ seamlessly. Under this constraint, the timing margin against clock skews between the CMD and any node B is denoted as

$$\max\{|m_{NC}(T) - m_B(T)|, \forall B\} < T_{refresh}/2 - 2\delta. \quad (11)$$

2.10. Message encryption/decryption and integrity mechanism

We also propose a privacy mechanism for the session between any two member nodes (sender or receiver), which offers data confidentiality via encryption and data integrity via an integrity checker, Message Integrity Code (*MIC*).

Fig. 8 shows the frame format of a message, in which the *PID* identifies the pseudo-identity of the sender, *KeyID* indexes which *CK* in the two key-slots is active, and *IV* denotes the initialization vector. They are all sent unencrypted, only the data is encrypted and denoted by shaded part of the frame. Once the sender generates such a frame, it sends the frame to the receiver via the wireless link.

$$sender \rightarrow receiver$$
$$: \{Header\|KeyID\|PID\|IV\|Ciphertext\|MIC\}.$$

IV, *PID*, and *KeyID* offer seeds for computing the block cipher and the *MIC*, and thus make encryption and decryption self-synchronous between the sender and receiver. Because of the key renewal mechanism, the length of the *IV* field can be small, e.g., 32 bits. Typically, the *IV* is varied with each frame. The *KeyID* field is used to identify the *CK*, which is used to derive the integrity key *CK_I* and encryption key *CK_E*, respectively.

As shown in Fig. 9, the *MIC* field is used to provide integrity mechanisms, which is computed from *KeyID*, *PID*, *IV*, and the cipher-text data by CBC (Cipher Block Chaining) mode. The integrity key is *CK_I*. The *MIC* is created by using an IV that is fed into a cipher block and its output is XOR'd with selected elements from the frame header which is then fed into the next cipher block. The process is continued over the remainder of the frame header until a 128 bit *MIC* is obtained.

To assure data confidentiality, data encryption involves bitwise module 2 addition of the output of a block stream

Header	*KeyID*	*PID*	*IV*	Ciper-text	*MIC*

Fig. 8. Message frame format.

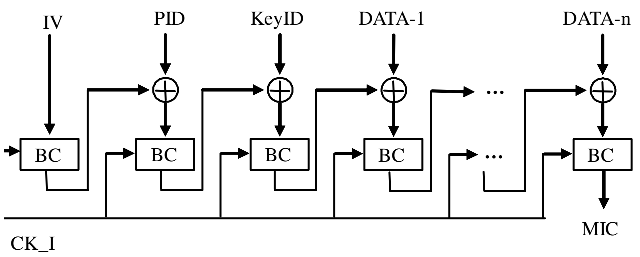


Fig. 9. Message integrity check mechanisms (BC: Block Cipher).

cipher with the transmitted data. Fig. 10 shows how the data is encrypted/decrypted between the sender and the receiver. The block cipher is seeded by the encryption key *CK_E*, *PID*, and *IV*. The output key-stream is fed back to the block cipher. This process is repeated until the entire frame has been encrypted.

The block cipher takes the concatenation of *PID*, *IV*, *CK_E*, and previous key-stream $KeyStream_{i-1}$ as input. As a result, it outputs a new stream block $KeyStream_i$.

$$KeyStream_i = BlockCipher(KeyStream_{i-1}, CK\_E, PID, IV).$$

Thus, if the plain-text is equally divided into n blocks, $\{PlainText_i | i = 1, 2, \ldots, n\}$, the ith cipher text is generated as $CipherText_i = KeyStream_i \oplus PlainText_i$.

The proposed protocol ensures that the key-steam will never be reused with the following measures: (1) a sender blends its own *PID* into the key-steam to ensure that all the member sharing the *CK* use different key-streams; (2) a sender increases its own *IV* by 1 for each message to avoid any repetition of key-stream; and (3) updating *CK* periodically also guarantees that none of member nodes reuse *IV*. Therefore, the proposed protocol addresses data integrity with a *MIC* value and confidentiality with symmetric CBC encryption.

3. Security analysis

We demonstrate that the proposed protocol satisfies security requirements for ad hoc group communications.

3.1. Identity anonymity and intracability analysis

The security requirement for concealing members' location information is achieved by introducing a simple identity anonymity mechanism. This feature makes an intruder unable to trace a particular user's location by intercepting the conversation. Our protocol provides identity anonymity in all phases by replacing members' real identity with a pseudonym identity.

Case 1. In the MAP phase, the real identity ID_i of T_i is replaced with $PID_i(=h(N_i\|ID_{CMD}) \oplus ID_i \oplus ID_{CMD})$. Since only the *CMD* knows the secret N_i and $h(N_i\|ID_{CMD}) \oplus ID_{CMD}$, nobody except the *CMD* can

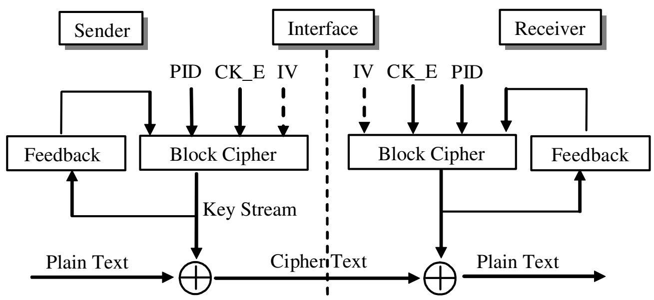


Fig. 10. Message encryption and decryption mechanisms.

obtain ID_i from PID_i by computing $ID_i = PID_i \oplus h(N_i \| ID_{CMD}) \oplus ID_{CMD}$. Given that a tracker does not know $h(N_i \| ID_{CMD}) \oplus ID_{CMD}$, it cannot get ID_i from the transmitted messages and then trace the location of a mobile member. Since each T_i's PID_i is computed using unique N_i, the legitimateT_i cannot compute another member T_k's ID_k by intercepting PID_k and further impersonate T_k.

Case 2. In one-way *CK&PID* renewal phase, the identity anonymity is enhanced by one-way *CK* refreshment mechanism. A member T_k can renew its *PID* as $PID_{k,j} = H(PID_{k,j-1} \| CK_j)$, $j = 1, 2, \ldots, n$.

3.2. Resistance to relay attack

A replay attack is a method that an intruder stores "stale" intercepted messages and retransmits them at a later time. An efficient measure against a replaying attack is to introduce timestamp t and lifetime L into the transmitted messages and set an expected valid time interval Δt for transmission delay.

All transmitted messages in each step of the proposed protocol scheme contain timestamps. According to the timestamp t and the interval Δt, the receiver can efficiently verify the validity of these messages by checking if $t - t_i < \Delta t$ is true, where t_i is the timestamp of a message while t is the current time when it is received. If this inequality holds, the message is valid. Otherwise, the *CMD* regards the message as a replaying message. This mechanism resists replaying attacks to a large extent.

3.3. Privacy of group conversation

The conversation information will be encrypted with *CK*. Hence, an intruder cannot know the conversation content without *CK*. To obtain *CK* in *InitConfkey* or *RefreshKey* message, an intruder must get the secret random r_i and then use it to calculate the *MK* as in Eq. (6). However, r_i $(i = 1,2,\ldots,m)$ is generated secretly by T_i. Nobody except T_i itself and the *CMD* know r_i. Hence, even $\{PID_1, E_{s_1}(t_1 \| s_1 \| r_1 \| ID_2 \| \cdots \| ID_m)\}$ and $\{Q, y, R, PA\}$ can be intercepted, the intruder cannot obtain r_i $(i = 1,2,\ldots,m)$ and compute $MK = (Q \cdot 2^y + R) \bmod r_i$, since it is impossible for him to get the key $s_i(s_i = f(ID_i))$ unless it knows ID_i of T_i. Hence, the intruder is prohibited from taking *CK* and eavesdropping any communication content.

3.4. Prevention of fraud

To prevent fraud, the *CMD* and members should mutually authenticate each other. Consider the following impersonation attack scenarios in MAP. This security requirement can be achieved by verifying the correctness of the member's identity ID_i and its secret key s_i.

Case 1. An intruder cannot impersonate the *CMD* to cheat T_i. Since the shared key s_i is only known to T_i and the *CMD*, and an intruder cannot send member T_i the valid response $\{PI, PA\}$ which is generated by the *CMD*. Once T_i receives the pair, it computes $MK = \mathrm{PI} \bmod r_i$ and verifies the validity of *MK* by checking if $PA = E_{MK}(ID_{CMD})$.

Case 2. An intruder cannot impersonate T_i to deceive the *CMD* since it cannot know the real identity of T_i. If the intruder uses a phony identity ID_i', the corresponding spurious pseudonym identity PID_i' can be identified by the *CMD*, since the *CMD* can obtain ID_i' by $ID_i' = PID_i' \oplus h(N_i \| ID_{CMD}) \oplus ID_{CMD}$ and detect the fake ID_i'. Since the real identity ID_i is kept secret, nobody except T_i itself and the *CMD* know the real identity.

Therefore, the MAP can efficiently prevent an intruder from impersonating attacks become of the mandatory mutual authentication mechanism between T_i and the *CMD*.

Similarly, in the *CK&PID* renewal phase, T_i and the *CMD* are also compulsorily authenticated to each other. Due to the one-way property of the *CK* sequence, the receivers can verify whether the new *CK* in the *RefreshKey* message belongs to the same key sequences as those stored in the key buffer, thus verifying its authenticity by an implicit authentication way.

3.5. Forward secrecy mechanism

The proposed protocol meets the security requirement for forward secrecy, since its key distribution mechanism can assure the one-way *CK&PID* refreshment by periodically or dynamically re-configuring the protocol, when (1) a member joins or leaves an in-progress group communication session; (2) the lifetime of the keys is overdue; (3) all n *CK*s in the *CK* sequence have been used up; (4) existing member nodes have been compromised; and (5) a node has definitely requested the *CK*, because more than t *CK*s is lost.

4. Performance analysis

In this session, the computation and communication overhead of the proposed protocol is analyzed. We quantify the cost of the communication and computation overhead when member nodes renew the *CK&PID*, and the performance improvement because to the robust and reliable mechanism.

4.1. Steady Markov state distribution

Fig. 11 shows the distribution state of a member node with a Markov chain. We assume that occurrence of the *CK* loss or authentication failure is random and mutually independent, and each node can finish the operation (*RequestConfKey*) within the interval $T_{refresh}$, if the keybuffer of a node is full. Let state S_i denote that there are

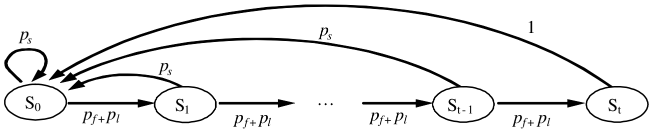


Fig. 11. State transition diagram of each member node.

iCK authentication failures, and thus there are *i* empty slots in the key-buffer. The state transition is triggered by three events: packet loss, *CK* authentication failure and *CK* authentication success. Let $p_f = \Pr\{CK$ authentication fails $|CK$ is received$\}$, and $p_s = \Pr\{CK$ authentication succeeds $|CK$ is received$\}$.

Without loss of generality, we also assume that all transmitted messages (including both legal and forged packets) via the wireless channel have the same loss probability p_l, which is defined as $p_l = \Pr\{\text{Message is lost}\}$. The assumption is reasonable since the wireless channel cannot distinguish the different packets. According to Eqs. (8) and (9), we have $p_s = p_l \cdot (1 - p^m)$ and $p_f = p_l \cdot p^m$, where $p_f + p_s + p_l = 1$. p_l represents the channel condition. A high p_l means that a wireless channel is with high loss or error rate. p_f is imposed by the forged packets, which leads to successful DoS attacks.

Let $P(k)$ denote the steady-state probability of state S_i that there are exactly k empty slots. According to the global balance equation, we have

$$\begin{cases} P(i) \cdot (p_s + p_f + p_l) = P(i-1) \cdot (p_f + p_l) \quad (i = 1, \ldots, t). \\ P(0) \cdot (p_f + p_l) = P(t) + \sum_{i=1}^{t-1} P(i) \cdot p_s. \end{cases} \tag{12}$$

Considering $\sum_{k=0}^{t} P(k) = 1$, each $P(k)$ is derived as

$$\begin{cases} P(0) = (1-\theta)/(1-\theta^{t+1}). \\ P(i) = P(0) \cdot \theta^k \quad (k = 1, 2, \ldots, t). \end{cases} \tag{13}$$

where $\theta = p_f + p_l$.

4.2. Communication overhead

To evaluate the communication overhead between the *CMD* and a member node, we normalize the expected communication overhead C_{Comm} by the cost of transmitting *RefreshKey* messages. Let C_{init} and $C_{refresh}$ denote communication costs for sending the *InitConfKey* and the *RefreshKey* message, respectively. Let $\alpha = C_{init}/C_{refresh}$ be the ratio of communication cost of *InitConfKey* to that of *RefreshKey*. Clearly, $\alpha > 1$ since the *InitConfKey* message needs more bandwidth or resources than the *RefreshKey* message.

It is necessary for the *CMD* to transmit the *InitConfKey* message when the following events occur:

Case 1. When a participant joins an in-progress group communication session, the *CMD* sends this new member the current configuration via an *InitConfkey* message.

Case 2. When a participant leaves a group communication session, the *CMD* will revoke this member by broadcasting the *InitConfkey* message to all the other member nodes.

Case 3. When all *n CK*s have been used, the *CMD* recomputes a new key sequence $\{CK'_i | i = 1, 2, \ldots, n\}$ and broadcasts the *InitConfkey* message to all members.

Case 4. A member node has definitely requested the *CK*, since it missed more than *t RefreshKey* messages. For this event, the *CMD* sends an *InitConfkey* message containing the configuration parameters to the node.

Note that in cases 2 and 3, all member nodes are required to be re-initialized, while in cases 1 and 4 the requesting node needs to be re-initialized by sending the *InitConfkey* message. Except for these cases, the *CMD* broadcasts the *RefreshKey* message periodically. So the expected communication cost of a node is

$$E[C_{Comm}] = C_{init} \cdot \left[\frac{1}{n} + P(t) + p_e + p_j\right] + C_{refresh} \cdot \sum_{k=0}^{t-1} P(k),$$

where p_e and p_j denote the probability of a member joining or leaving a group session, respectively. According to Eq. (13), the communication cost can be normalized with $C_{refresh}$ as:

$$C_{comm} = \alpha \cdot \left[\frac{1}{n} + \theta^t \cdot P(0) + p_e + p_j\right] + \sum_{k=0}^{t-1} \theta^k \cdot P(0). \tag{14}$$

If the value of C_{Comm} is close to 1, it indicates that most *RefreshKey* messages should work well. If C_{Comm} is close to α, *RefreshKey* messages works less efficiently. To analyze the dynamics of the *CK* refreshment, we assume that the frequency of joining or leaving a group session is low so that Eq. (14) can be approximated as

$$C_{comm} = \alpha \cdot \left[\frac{1}{n} + \theta^t \cdot P(0)\right] + \sum_{k=0}^{t-1} \theta^k \cdot P(0).$$

Fig. 12 shows the function relation between C_{Comm} and the key buffer length t, where $n = 500$, $p_l = 0.05$–0.45, and $\alpha = 10$. The choice of α implies that the cost of transmitting and dealing with *InitConfKey* message is higher than that of transmitting *RefreshKey* message, since the packet size of *InitConfKey* is larger than that of *RefreshKey*. It can be seen that the key-buffer length in each member node determines the communication cost. A small *t* will lead to a high communication overhead while a large *t* can remarkably reduce the communication overhead. Fig. 12 also shows that the proposed protocol is efficient in terms of communication overhead even under serious DoS attacks ($p_f \geqslant 0.25$) and packet loss ($p_l \geqslant 0.25$), since the normalized communication cost C_{Comm} approaches 1 if $t \geqslant 10$.

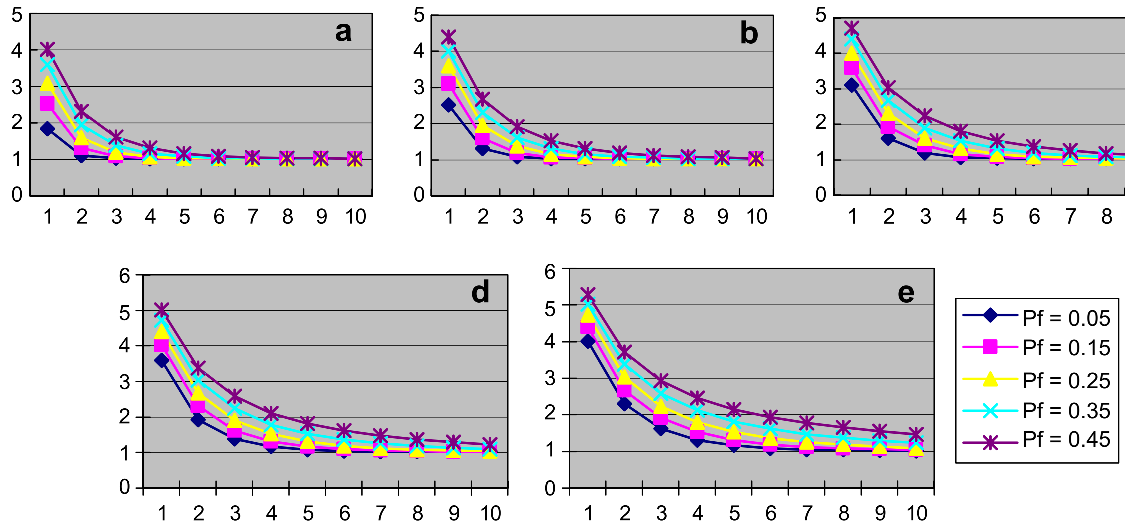


Fig. 12. Normalized communication costs C_{Comm} vs. the key buffer length t at member node: $p_f = 0.05$–0.45. (a) $p_l = 0.05$, $n = 500$, $\alpha = 10$. (b) $p_l = 0.15$, $n = 500$, $\alpha = 10$. (c) $p_l = 0.25$, $n = 500$, $\alpha = 10$. (d) $p_l = 0.35$, $n = 500$, $\alpha = 10$. (e) $p_l = 0.45$, $n = 500$, $\alpha = 10$.

4.3. Computation overhead

The main computation overhead in member nodes is the modular arithmetic per *InitConfkey* message and the hash computation per *RefreshKey* message. Let N_m and N_h denote the number of modular arithmetic per *InitConfKey* message and hash computations per *RefreshKey* message, respectively. If there are exactly $k < t$ empty slots, N_h can be computed as

$$N_h = \begin{cases} 0, & \text{if } CK \text{ message is lost;} \\ 1, & \text{if } CK \text{ strong authentication fails;} \\ k+1, & \text{if } CK \text{ authentication succeeds.} \end{cases} \quad (15)$$

If all t slots in the key-buffer are empty due to the CK authentication failure or message loss, the member node can explicitly initiate a *RequestConfKey* message to obtain the new CK. Thus, it needs to do $(t+1)$ extra hash computations according to the received CK, and we can have

$$N_h = \begin{cases} t+1, & \text{if } CK \text{ message is lost;} \\ t+2, & \text{if } CK \text{ strong authentication fails;} \\ t+1, & \text{if } CK \text{ authentication succeeds.} \end{cases} \quad (16)$$

Therefore, if there are k empty slots, the corresponding conditional expected value of N_h, can be derived as

$$E[N_h|k\ slots] = \begin{cases} p_f + (k+1)\cdot p_s & (k<t), \\ (t+2)\cdot p_f + (t+1)\cdot(p_s+p_l) & (k=t), \end{cases} \quad (17)$$

and the expected value of N_h is calculated as

$$E[N_h] = \sum_{k=0}^{t} E[N_h|k\ slots]\cdot P(k) = (t+1+p_f)\cdot P(0)\cdot\theta^t + \sum_{k=0}^{t-1}\{(k+1)\cdot(1-p_l) - k\cdot p_f\}\cdot P(0)\cdot\theta^k. \quad (18)$$

Let that C_{Hash} and $C_{Modular}$ denote the cost of calculating a single modular arithmetic and hash operation, respectively. Let $\beta = C_{Modular}/C_{Hash}$. Similar to the evaluation of the communication costs, the expected computation costs of a member node can be computed as follows

$$E[C_{Comp}] = C_{hash}\cdot E[N_h] + C_{Modular}\cdot\left[\frac{1}{n} + P(t) + p_e + p_j\right].$$

According to Eq. (13), the computation cost of member nodes can be normalized with $C_{Modular}$ as

$$C_{comp} = E[N_h] + \beta\cdot\left(\frac{1}{n} + P(0)\cdot\theta^t + p_e + p_j\right). \quad (19)$$

To reduce the analysis complexity, we also assume that the frequency of joining or exiting a group session is low. So, p_e and p_j in Eq. (19) can be ignored. Figs. 13 and 14 show the normalized computation cost C_{comp} as the function of p_l and p_f under the assumption of $n = 500$ and $\beta = 5$, respectively. The computation cost of each node is low, since each node only computes less than three hash functions per *CK&PID* renewal even in the worst case, e.g., where $p_l = 0.45$ and $p_f = 0.45$.

Fig. 15 depicts C_{comp} as the function of the key buffer length t, where both p_f and p_l vary from 0.05 to 0.45. It can be seen that the desirable number of key buffer is $t \geqslant 15$ to keep communication and computation cost low, i.e., the normalized communication or computation cost is within the range of 1–1.5. Therefore, the proposed protocol is efficient in terms of communication and computation overhead, even under heavy DoS attacks and high packet loss rate.

4.4. Comparison with existing protocol

The performance comparison between the proposed protocol and the one in [6] is listed in Table 2. The number of modular arithmetic, symmetric encryption/decryption operations, and message transmissions are compared. The

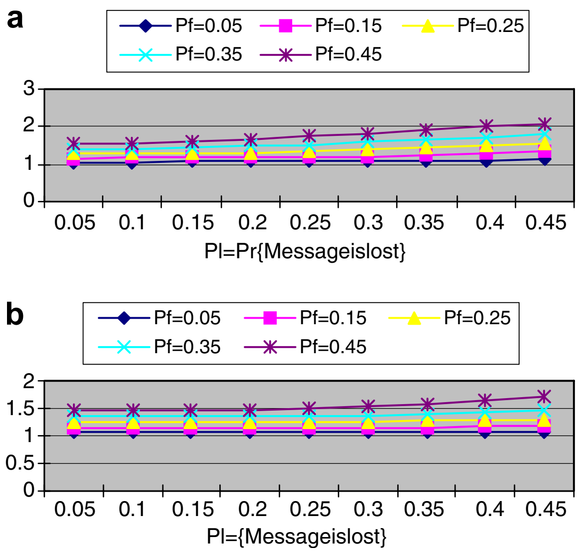


Fig. 13. Normalized computation costs of Nodes vs. $p_l = \Pr\{\text{Message is lost}\}$: $n = 500$, $\beta = 5$. (a) $t = 5$, $\beta = 5$. (b) $t = 10$, $\beta = 5$.

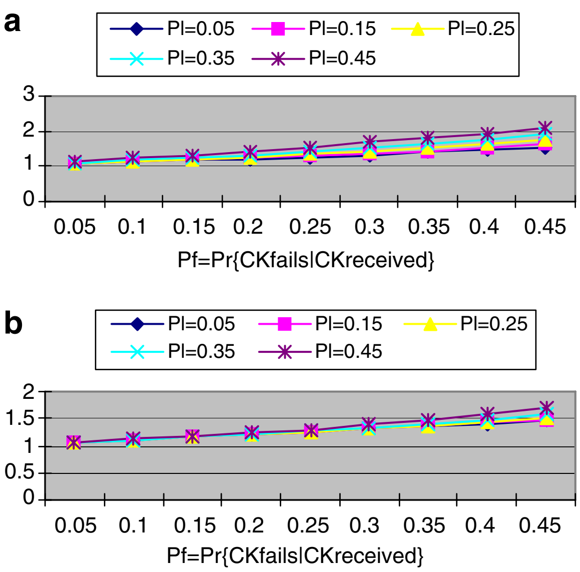


Fig. 14. Normalized computation costs C_{Comm} at member node vs. $p_f = \Pr\{CK \text{ fails} \mid CK \text{ received}\}$: $n = 500$, $\beta = 5$. (a) $t = 5$, $\beta = 5$. (b) $t = 10$, $\beta = 5$.

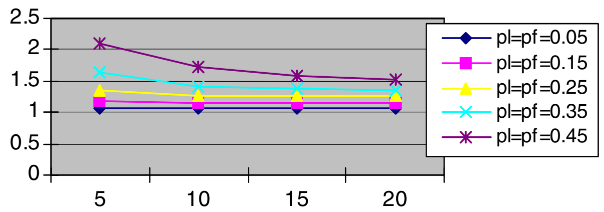


Fig. 15. Normalized computation costs of member nodes vs. key buffer length t: $n = 500$, $\beta = 5$.

security features of the two protocols are also listed side by side in the table. The differences between them are shown in shaded rows. It can be seen that the communication and computation complexity required for the proposed protocol is similar to that in [6], but the proposed protocol achieves the salient features, such as identity anonymity and location intracability, periodically one-way *CK&PID* refreshment, dynamically joining and leaving an in-progress session, forward secrecy, data privacy, DoS-tolerance for broadcasting refreshment message, fault-tolerance for recovering the lost refreshment message, robustness for resisting the clock skews among member nodes, and seamless key switch without disrupting ongoing data transmissions.

Table 2
Performance and security comparison

Comparison item		Protocol in [6]	Our protocol
Modular arithmetic	*T*	1 (step 5)	1 (step 5)
	CMD	1 (step 4)	1 (step 4)
Symmetric encryption	*T*	1 (step 1 or 5)	1 (steps 1 or 5)
	CMD	N/A	N/A
Symmetric decryption	*T*	N/A	N/A
	CMD	m (step 2 and 5)	m (steps 2 and 5)
Transmission messages		$2 + 3(m - 1)$	$2 + 3(m - 1)$
Identity anonymity		N/A	Yes
Location intracablity		N/A	Yes
One-way *CK&PID* renewal		N/A	Yes
Data privacy		N/A	Yes
Dynamic participant		Yes	Yes
Forward secrecy		N/A	Yes
DoS-tolerance		N/A	Yes
Fault-tolerance		N/A	Yes
Resistance for clock skews		N/A	Yes

T, Group member; *CMD*, Commander node.

5. Conclusion

In this paper, a novel secure and reliable authentication protocol has been developed for group communications in ad hoc networks. The protocol has several attractive features, such as identity anonymity and location intracability, one-way *CK&PID* refreshment, dynamically joining and leaving an in-progress session, forward secrecy, data privacy, etc. In addition, reliability enhancement features, such as DoS- and fault-tolerance, clock skews resistance, and seamless key switch, can improve the robustness, and two-phase packet filters with random selection strategies can tolerate the DoS attack and improve the system survivability.

Acknowledgements

This research has been supported in part by the NSFC under Contracts No. 60573144, 60218003, 60429202, 60673187, 60432030, and 90412012, Intel IXA University Research Plan, and a grant from NSERC (Natural Sciences and Engineering Research Council of Canada) Postdoctoral Fellowship.

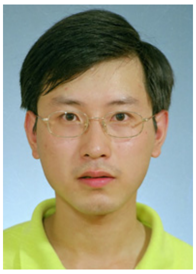
References

[1] D. Djenouri, L. Khelladi, A.N. Badache, A survey of security issues in mobile ad hoc and sensor networks, IEEE Commun. Surv. Tutorials 7 (2005) 2–28.

[2] I. Ingemarson, D.T. Tang, C.K. Wong, A conference key distribution system, IEEE Trans. Inf. Theory IT-28 (1982) 714–720.
[3] M.-S. Hwang, W.-P. Yang, Conference key distribution schemes for secure digital mobile communications, IEEE J. Select. Areas Commun. 13 (1995) 416–420.
[4] M.-S. Hwang, Dynamic participation in a secure conference scheme for mobile communications, IEEE Trans. Veh. Tech. 48 (1999) 1469–1474.
[5] S.-L. Ng, Comments on dynamic participation in a secure conference scheme for mobile communications, IEEE Trans. Veh. Tech. 50 (2001) 334–335.
[6] K.-F. Hwang, C.-C. Chang, A self-encryption mechanism for authentication of roaming and teleconference services, IEEE Trans. Wireless Commun. 2 (2003) 400–407.
[7] X. Yi, C.K. Siew, C.H. Tan, Y. Ye, A secure conference scheme for mobile communication, IEEE Trans. Wireless Commun. 2 (2003) 1168–1177.
[8] Y. Jiang, C. Lin, M. Shi, X. Shen, A self-encryption authentication protocol for teleconference services, Globecom (2005) 1706–1710.
[9] D. Brown, Techniques for privacy and authentication in personal communication system, IEEE Pers. Commun. Mag. 2 (1995) 6–10.
[10] Y. Jiang, C. Lin, M. Shi, X. Shen, Hash-binary-tree based group key distribution with time-limited node revocation, in: Y. Xiao, Y. Pan, (Eds.), Security in Distributed and Networking Systems, 2007.
[11] A.D. Wood, J.A. Stankovic, Denial of service in sensor networks, Computer 35 (2002) 54–62.
[12] N. Haller, The s/key one-time password system, RFC1760, IETF, 1995.
[13] A. Perrig, R. Szewczyk, V. Wen, D. Culler, J.D. Tygar, SPINS: security protocols for sensor networks, in: Proceedings of IEEE/ACM MobiCom'01, pp. 189–199, 2001.
[14] A. Perrig, J.D. Tygar, D. Song, R. Canetti, Efficient authentication and signing of multicast streams over lossy channels, in: Proceedings of the 2000 IEEE Symposium on Security and Privacy, pp. 56–65, 2000.
[15] D. Liu, P. Ning, Multilevel μTESLA: broadcast authentication for distributed sensor networks, ACM Trans. Embed. Comput. Syst. 3 (2004) 800–836.
[16] T. Park, K.G. Shin, LiSP: a lightweight security protocol for wireless sensor networks, ACM Trans. Embed. Comput. Syst. 3 (2004) 634–660.

Yixin Jiang received a Ph.D. degree (2006) from Tsinghua University, China and an M.E. degree (2002) from Huazhong University of Science and Technology both in Computer Science. In 2005, he was a Visiting Scholar with the Department of Computer Sciences, Hong Kong Baptist University. His current research interests include security and performance evaluation in wireless communication and mobile computing. He has published more than 20 papers in research journals and IEEE conference proceedings in these areas.

Chuang Lin is a professor and the head of the Department of Computer Science and Technology, Tsinghua University, Beijing, China. He received the Ph.D. degree in Computer Science from Tsinghua University in 1994. In 1985–1986, he was a Visiting Scholar with the Department of Computer Sciences, Purdue University. In 1989–1990, he was a Visiting Research Fellow with the Department of Management Sciences and Information Systems, University of Texas at Austin. In 1995–1996, he visited the Department of Computer Science, Hong Kong University of Science and Technology. His current research interests include computer networks, performance evaluation, network security, logic reasoning, and Petri net and its applications. He has published more than 200 papers in research journals and IEEE conference proceedings in these areas and has published three books. Professor Lin is an IEEE senior member and the Chinese Delegate in IFIP TC6. He serves as the General Chair, ACM SIGCOMM Asia workshop 2005; the Associate Editor, IEEE Transactions on Vehicular Technology; and the Area Editor, Journal of Parallel and Distributed Computing.



Minghui Shi received a B.S. degree (1996) from Shanghai Jiao Tong University, China, and an MASc. degree (2002) and a Ph.D. degree (2006) from the University of Waterloo, Ont., Canada, all in electrical and computer engineering. He is currently with McMaster University, Ont., Canada as an NSERC (Natural Sciences and Engineering Research Council of Canada) Postdoctoral Fellow and a research associate with the Centre for Wireless Communications, University of Waterloo. His current research interests include wireless LAN/cellular network integration, vehicular communications networks and relevant network security issues.

Xuemin (Sherman) Shen received the B.Sc. (1982) degree from Dalian Maritime University (China) and the M.Sc. (1987) and Ph.D. degrees (1990) from Rutgers University, New Jersey (USA), all in electrical engineering. Dr. Shen is with the Department of Electrical and Computer Engineering, University of Waterloo, Canada, where he is a Professor and the Associate Chair for Graduate Studies. His research focuses on mobility and resource management in interconnected wireless/wire line networks, UWB wireless communications systems, wireless security, and ad hoc and sensor networks. He is a coauthor of two books, and has published more than 200 papers and book chapters in wireless communications and networks, control and filtering. Dr. Shen serves as the Technical Program Committee Chair for IEEE Globecom'07, General Co-Chair for Chinacom'07 and QShine'06, the Founding Chair for IEEE Communications Society Technical Committee on P2P Communications and Networking. He also serves as a Founding Area Editor for IEEE Transactions on Wireless Communications; the Editor-in-Chief for Peer-to-Peer Networking and Application; Associate Editor for IEEE Transactions on Vehicular Technology; KICS/IEEE Journal of Communications and Networks, Computer Networks; ACM/Wireless Networks; Wireless Communications and Mobile Computing (Wiley), etc. He has also served as Guest Editor for IEEE JSAC, IEEE Wireless Communications, and IEEE Communications Magazine. Dr. Shen received the Excellent Graduate Supervision Award in 2006, and the Outstanding Performance Award in 2004 from the University of Waterloo, the Premier's Research Excellence Award (PREA) in 2003 from the Province of Ontario, Canada, and the Distinguished Performance Award in 2002 from the Faculty of Engineering, University of Waterloo. Dr. Shen is a registered Professional Engineer of Ontario, Canada.

Xiaowen Chu received a B.E. degree in the Computer Science from Tsinghua University, Beijing, PR China, in 1999, and a Ph.D. degree in the Computer Science from the Hong Kong University of Science and Technology in 2003. He is currently an assistant professor with the Department of Computer Science, Hong Kong Baptist University. His main research interests are on Wireless Mesh Networks, Optical WDM Networks, Internet Security, Peer-to-Peer network, Multimedia Networking, and Web Server Performance.