2008

Mobile ad-hoc network key management with certificateless cryptography

Zhenfei Zhang
*University of Wollongong*

Willy Susilo
*University of Wollongong, wsusilo@uow.edu.au*

Raad Raad
*University of Wollongong, raad@uow.edu.au*

**Publication Details**

Mobile ad-hoc network key management with certificateless cryptography

Abstract
In this paper, we present an idea of adopting certificateless public key encryption (CL-PKE) schemes over mobile ad hoc network (MANET), which has not been explored before. In current literature, essentially there exists two main approaches, namely the public key cryptography and identity-based (ID-based) cryptography. Unfortunately, they both have some inherent drawbacks. In the public key cryptography system, a certificate authority (CA) is required to issue certificates between users' public keys and private keys to ensure their authenticity, whilst in an ID-based cryptography system, users' private keys are generated by a key generation center (KGC), which means the KGC knows every users' keys (the key escrow problem). To avoid these obstacles, Al-Riyami and Paterson proposed certificateless cryptography systems where the public keys do not need to be certified and the KGC does not know users' keys. Essentially, certificateless cryptography relies between the public key cryptography and ID-based cryptography. In this work, we adopt this system's advantage over MANET. To implement CL-PKE over MANET and to make it practical, we incorporate the idea of Shamir's secret sharing scheme. The master secret keys are shared among some or all the MANET nodes. This makes the system self-organized once the network has been initiated. In order to provide more flexibility, we consider both a full distribution system and a partial distribution system. Furthermore, we carry out two simulations to support our schemes. We firstly simulate our scheme to calculate our encryption, decryption and key distribution efficiency. Then we also simulate our scheme with AODV to test the network efficiency. The simulations are performed over OPNET.

Disciplines
Physical Sciences and Mathematics

Publication Details

This conference paper is available at Research Online: http://ro.uow.edu.au/infopapers/3199
Mobile Ad-hoc Network Key Management with Certificateless Cryptography

Zhenfei Zhang, Willy Susilo and Raad Raad

1School of Electrical, Computer and Telecommunications Engineering (SECTE)
University of Wollongong
Email: {zz920, raad}@uow.edu.au

2Centre for Computer and Information Security Research (CCISR)
School of Computer Science and Software Engineering (SCSSE)
University of Wollongong
Email: wsusilo@uow.edu.au

Abstract

In this paper, we present an idea of adopting certificateless public key encryption (CL-PKE) schemes over mobile ad hoc network (MANET), which has not been explored before. In current literature, essentially there exists two main approaches, namely the public key cryptography and identity-based (ID-based) cryptography. Unfortunately, they both have some inherent drawbacks. In the public key cryptography system, a certificate authority (CA) is required to issue certificates between users’ public keys and private keys to ensure their authenticity, whilst in an ID-based cryptography system, users’ private keys are generated by a key generation center (KGC), which means the KGC knows every users’ keys (the key escrow problem). To avoid these obstacles, Al-Riyami and Paterson proposed certificateless cryptography systems where the public keys do not need to be certified and the KGC does not know users’ keys. Essentially, certificateless cryptography relies between the public key cryptography and ID-based cryptography. In this work, we adopt this system’s advantage over MANET. To implement CL-PKE over MANET and to make it practical, we incorporate the idea of Shamir’s secret sharing scheme. The master secret keys are shared among some or all the MANET nodes. This makes the system self-organized once the network has been initiated. In order to provide more flexibility, we consider both a full distribution system and a partial distribution system. Furthermore, we carry out two simulations to support our schemes. We firstly simulate our scheme to calculate our encryption, decryption and key distribution efficiency. Then we also simulate our scheme with AODV to test the network efficiency. The simulations are performed over OPNET.

Keywords: certificateless cryptography, MANET, AODV, OPNET, public key cryptography, identity-based cryptography, secret sharing

1 Introduction

The mobile ad hoc network (MANET) is a network that is merely comprised of mobile devices without any pre-established infrastructures. In this type of network, routing is an essential problem, since unlike a traditional network, MANET has no access point for the nodes to connect to and communicate. With the fast development of MANET technology, security becomes an important issue. With more and more applications developed over MANET devices, the need for MANET security has increased significantly over the last few years. One of the main discussion points in this topic is the key distribution schemes. Several solutions have been proposed in the literature, but nevertheless they have raised several drawbacks, such as the reliance on a single online authority [8, 5] or the unconditional trust to a trusted authority in the ID-based system [4]. When the schemes are built from public key cryptography, it will require certificate authorization that makes it impractical. Nonetheless, an ID-based system requires a significant amount of trust to a single entity which also makes it impractical.

Our Contribution

In this paper, we consider a different approach to the existing solutions, namely to incorporate the certificateless cryptography into MANET. As we shall show in this paper, the adoption of certificateless cryptography to the MANET scenario is not very straightforward. Nonetheless, by combining the secret sharing schemes with the certificateless cryptography, we ob-
tain an efficient and secure MANET scheme. Our contribution is to apply the existing certificateless cryptography into MANET using a threshold secret sharing scheme. We firstly create a generic model based on the above ideas and then we proposed our scheme that comprises of a combination of certificateless cryptography and secret sharing scheme. To support our idea, we implement our schemes in OPNET to analyze its efficiency and practicality.

**Paper Organization**
The rest of the paper is as follows. In Section 2, we will review the background required throughout this paper that includes the background on cryptography, MANET and some existing MANET key management schemes. In Section 3, we will propose our scheme that comprises of a combination of certificateless cryptography and secret sharing scheme. In Section 4, we present the result of our simulation in both C language and OPNET. Finally, the last Section concludes the paper.

## 2 Background

### 2.1 Certificateless Cryptography

**Public Key Cryptography** The concept of the public key cryptography scheme was put forth by Diffie and Hellman in their seminal paper in [11] and the first realization of the public key cryptography was proposed by Rivest, Shamir and Adleman in 1978 [1]. In a public key cryptography system, there are two separate keys involved: the public key and the private key. In an encryption scheme scenario, the public key is used for encrypting the message and the private key is used to decrypt the message. The main idea of this system relies on the fact that if the private key is known, then it is easy to compute the public key, but not vice versa. Therefore, the public key can be made public and known by anyone. This method makes it possible for a user to deliver some messages without any pre-established shared keys.

Nonetheless, the key management is the main stumbling block in the public key scenario, since it is not possible for anyone who obtains someone’s public key from a public place, such as the Internet, to verify the authenticity of this public key. Therefore, there is a necessity to authenticate this public key and hence, an adversary cannot replace a genuine public key with any other public key of its choice. Henceforth, a trusted third party called the certification authority (CA) is required. The role of the CA is to issue certificates on public keys for users. Then, anyone who obtains any user’s public key can verify its authenticity by verifying whether the certificate attached is indeed valid. This is the main drawback of this system.

**Identity-based Cryptography** The concept of identity-based (ID-based) cryptography was introduced by Shamir in [13] to solve the main drawback of public key cryptography by removing the necessity of the certificates. In an ID-based system, the identity of users are used as their public keys and therefore there is no need to have this public keys (i.e. the users’ identity) certified. The secret key is derived from the user’s identity together with the trusted authority, called the Private Key Generator (PKG)’s secret key. Nonetheless, this makes the system impractical since the PKG will know all the secret keys that the users have and therefore, the PKG can always impersonate any users. This inherent problem in ID-based cryptography is known as the key escrow problem, which makes the ID-based system only practical in a closed organization. An unconditional trust to the PKG is required and it is assumed that the PKG will not be malicious.

**Certificateless Cryptography** In 2003, Al-Riyami and Paterson [3] proposed a new system known as certificateless cryptography. The idea of certificateless cryptography is to gather the strength of both the public key cryptography and ID-based cryptography and to avoid the drawbacks that these two systems have. In this system, there is a trusted authority called the Key Generation Centre (KGC) that will need to generate a partial secret key for the users, given the users’ identity. Nonetheless, each user also needs to generate his/her own partial secret key and based on these two pieces of information (partial secret keys), the user can generate the public key that needs to be published. Although this system incorporates a public key, this public key does not need to be certified as this public key has been ‘implicitly’ certified by the partial secret key issued by the KGC. Hence, to verify the authenticity of the public key, the KGC’s public key needs to be involved. We note that there is no key escrow problem in this model as the KGC does not know the user’s secret key. The KGC can only know the partial secret key but not the complete secret key as some part of the secret key is generated by the user himself/herself.

### 2.2 Mobile Ad Hoc Network

**MANET Overview** Mobile Ad-hoc Network (MANET) is one of the most widely discussed and researched areas in the field of wireless communications. In a traditional network, mobile devices connect to each other via an access point. If the access point fails, users cannot communicate to each other. In the MANET scenario, no access point or node is required. MANET is a network that only consists of mobile
devices such as personal digital assistants (PDAs) and laptops. It requires no centralized infrastructure like basic switch centers or wireless routers. Nodes connect to each other via the ad hoc model. Nodes work not only as a host but also as a router, joining or leaving the network at any moment, making the network highly dynamic.

Because of MANET’s non-centralized infrastructure and highly dynamic characteristics, routing is an essential part of this network. Without routing, devices are unable to connect to each other, and the network becomes crippled. Routing protocols for the Internet do not perform very well in MANET. Routes may become invalid at any second, which may be caused by a slight movement of one node. In this case, dynamic adaptive routing protocols must be applied.

AODV Ad-hoc on-demand distance vector (AODV) routing protocol is an on-demand routing protocol in MANET proposed by Perkins, Belding-Royer and Das [10, 7]. In this protocol, nodes do not perform routing until a request is generated or received. It uses three types of control messages: Route Request (RREQ), Route Reply (RREP) and Route Error (RERR) to control the whole network.

In order to discover a Destination Node (DN), the source node (SN) broadcasts a RREQ message. A sequence number is given to each node which has received a RREQ message. When this RREQ message finds its way to the DN, a RREP message is generated, sending back to the SN the same way the RREQ came from, and thus a route is established. After this, this route will be assigned with a lifetime. Every time a message is transferred via this route, the lifetime is refreshed. When the lifetime is expired, the route becomes invalid.

MANET Security MANET is equipped with some security aspects. It has neither pre-existing infrastructure nor predictable network topology, so data is transferred through nodes in a multi-hop way. The resources of a MANET device, such as battery life or transmitting range, are always limited.

However its physical security is poor. This means that MANET devices can be easily stolen and the physical signals that a MANET device uses may also be vulnerable to a spoofing attack.

When discussing MANET security, we mainly consider two approaches: the key management and routing security. In this paper, we merely consider the key management schemes. The key management schemes handle the generation, establishment and distribution and revocation of keys. We will elaborate the three main key management schemes in MANET in the following section.

2.3 Existing Key Management Schemes

Partially distributed authority scheme Partially distributed authority scheme was firstly proposed by Zhou and Hass [8]. In their scheme it is assumed that there is an Offline Trust Third Party (OTTP) constructing and distributing keys for all the nodes. Firstly, this OTTP generates a pair of master public/secret keys. The master public key (mpk) is known by every node in the MANET, while the master secret key (msk) is divided into n parts, where each part is presented by $S_i (i = 0, 1, 2...n)$. Then OTTP picks n arbitrary nodes, randomly distributed with msk parts. These n nodes collectively form the Distributed Certificate Authority (DCA).

The OTTP then generates certificates for all of the nodes and distributes them respectively. In Zhou and Hass’s scheme, those certificates are fully stored in each DCA node as well. This provides authentication from potential threads of unauthorized nodes. Any unauthorized node does not have valid certificate, thus will not get key shares from DCA nodes.

Assuming the threshold of the system is $t$, node $i$ needs to obtain at least $t + 1$ msk shares to retrieve the msk. Node $i$ will send out requests to $t$ DCA nodes, with a certificate of its own. Once the certificate is verified by a DCA node, which is achieved by comparing with DCA’s certificate database, the DCA node will reply with a share of msk. After successfully obtain $t$ valid key shares, node $i$ will retrieve the msk.

This brings an imbalanced load to the DCA nodes, because those DCA nodes are in charge of the whole network. This scheme also requires pre-establishment before the initiation. Certificates of each node are pre-stored in the DCA nodes.

In order to solve these problems, Yi and Kraves proposed a modified model [6]. It makes use of the broadcast certification request (CREQ) and the certification reply (CREP) packets. It allows nodes to broadcast the certification request (CREQ) packets using a flooding method. Any DCA which gets this packet answers with a certification reply (CREP). If the node successfully collects $t + 1$ CREPs, it will be able to reconstruct the full certificate. If the certificate is valid, the certification is successful; otherwise, the node will generate another CREQ packet.

Fully distributed authority scheme A fully distributed authority scheme is a modification of partially distributed certificate authority scheme firstly proposed by Luo et al. [5]. This scheme also makes use of the $(n, t)$ threshold secret sharing scheme [12]. The difference between Luo et al.’s model and Zhou
and Hass’ model relies on the following: In Zhou and Hass’ model, DCA nodes are randomly selected from all the nodes while Luo et al’s model uses all of the nodes in the MANET to form the DCA. The msk is shared among all the nodes and for this reason, this scheme is called ‘fully distributed’.

Firstly, an offline trusted third party (OTTP) generates an RSA pair mpk/msk. The mpk is shared in the MANET. The msk is divided into n shares; each part is a Secret Key (sk) for every node. Nodes’ Public Keys (pk) are created from those sks.

Then the OTTP creates certificates signed with the msk for each node, in order to bind nodes’ unique ID with nodes’ public key. These certificates are unforgeable and are stored in every node in the network.

When a node, namely, node A, needs to get the msk, it sends out requests to all its one hop neighbouring nodes. If one of the neighbour nodes, namely, node B, gets the request, it compares node A’s ID and certificate pair with the information B stored in its database. If the result is positive, node B will send back its own share of the msk, as well as the certificate of itself. If the number of the nodes which replied with valid certificates and key shares is more than t, the node A obtains the msk.

**ID-Based distribution scheme** One of the Identity-based authority schemes was proposed by Boneh and Franklin [4], which is an upgraded solution to Zhou and Hass’s scheme. It replaced the DCA with a threshold private key generator (PKG). Initially, users in the network will collectively form the PKG. This PKG will generate a pair of mpk/msk, and the msk is divided and shared among all the initial nodes. It is not stated in [4] how this PKG is formed nor how the msk is distributed. In [2], Van Der Merwe, Dawoud and McDonald designed an OTTP which is called centralized PKG to generate and distribute keys. After the initiation, the user’s identity is used as the user’s public key, while each PKG node will generate a part of this user’s private key, which is based on the user’s identity. In this way, each user needs to obtain t + 1 parts of private key to retrieve the private key.

### 3 CL-PKE over MANET

#### 3.1 Generic Model

We assume that at the beginning of the network there is a Key Generator Center (KGC) which generates partial secret keys for all the users. We also denote n to be the number of original nodes and t to be the pattern of security level of the threshold system. Those n nodes collectively form a Distributed Key Generator Center (DKGC). After the initiation, the KGC will go offline, and the network becomes self-organized. We define those nodes that get partial secret keys from the KGC to be the original nodes, those nodes that get partial secret keys from DKGC to be the new-joint nodes and those nodes that collectively form the DKGC to be DKGC nodes.

- **Setup:**
  This algorithm takes as input a security parameter $1^k$ and returns the master private key $msk$ and master public key $mpk$. This algorithm is run by the KGC, in order to setup a certificateless ad hoc system.

- **Extract-partial-secret-key:**
  This algorithm takes as input the master public key $mpk$, the master private key $msk$ and an identity ID=$i \in \{0,1\}^*$. It outputs a partial private key $d_i$. This algorithm runs by KGC once at the initiation of the network.

- **Extract-master-secret-key-shares:**
  This algorithm takes as input the master private key $msk$ and an identity ID=$i \in \{0,1\}^*$. It outputs a master secret key shares $msks_i$. This algorithm runs by KGC once at the initiation of the network.

- **Extract-partial-secret-key-share-and-master-secret-key-share:**
  This algorithm takes as input the master public key $mpk$, the master private key share $msks_i$ from a DKGC node and an identity $new$ of a new-jointly node. It outputs a share of partial user private key $ds_{new,i}$ and a share of master secret key share $msks_{new,i}$, $i \in \{0,1...n\}$. This algorithm runs by DKGC nodes.

- **Extract-master-secret-key-shares-DKGC:**
  This algorithm takes as input the master public key $mpk$, an identity ID=$new \in \{0,1\}^*$, and $t$ shares of master private key share $msks_{new,i}$, $i \in \{0,1...n\}$. It outputs a master secret key share $msks_{new}$. This algorithm runs by the new-joint node.

- **Extract-partial-secret-key-DKGC:**
  This algorithm takes as input the master public key $mpk$, a user identity ID=$i$, a partial private key $d_{i}$ and a secret value $x_i$. It outputs a user...
public/private key pair \((pk_i, sk_i)\) or an error symbol. This algorithm runs by all the nodes.

- **Encryption:**
  This algorithm takes as input the master public key \(msk\), a user’s identity \(ID=i\), a user’s public key \(pk\), and a message \(msg\). It outputs a cipher text \(c\).

- **Decryption:**
  This algorithm takes as input the master public key \(msk\), a user’s private key \(sk\), and a cipher text \(c\). It outputs a message \(msg\).

**Fully Distributed System** In the fully distributed system, all the nodes will have a share of \(msk\). They together maintain the stability of the system. At the initiation stage, the KGC generates a master public/private key pair \((mpk/msk)\) using Setup algorithm. It then generates user partial keys using Extract-partial-secret-key algorithm and divides \(msk\) with Extract-master-secret-key-share. The user partial keys \(d_{ID}\) and master secret key shares \(msks_{ID}\) are distributed to all the origin nodes. Once this is done, the KGC goes offline, and all the original nodes become DKGC nodes.

We use the threshold cryptography to provide authentication for new jointly nodes. A new-joint nodes need to successfully contact at least \(t\) DKGC nodes. Those DKGC nodes will run Extract-partial-secret-key-share-and-master-secret-key-share algorithm for the new-joint node. Once this new-joint node obtains \(t\) shares of \(msks_{new,i}\) and \(t\) shares of \(d_{new,i}\), it will be able to derive a master secret key share \(msks_{new}\) and a partial secret key \(d_{new}\) by Extract-master-secret-key-shares-DKGC and Extract-partial-secret-key-DKGC respectively, and it becomes a DKGC node. The number of DKGC nodes rises with the increase of node numbers.

DKGC nodes use Set-user-keys algorithm to calculate their own public/private keys. The public keys will be broadcasted all through the network so that nodes can communicate to each other with Encryption and Decryption algorithms.

**Partially Distributed System** In a partially distributed system, a certain number of nodes will become DKGC nodes. The \(msk\) is only shared between these nodes. They are responsible for issuing partial secret key for new coming nodes. This system differs from fully distribution system that:

1. For a new-joint node, the DKGC nodes only issue partial secret key shares \(d_{new,i}\), without any master secret key shares \(msks_{new,i}\).
2. Once a DKGC node goes offline, a random non-DKGC node will be picked. Other DKGC nodes will give this node master secret key shares \(msks_{new,i}\) so that this chosen one will become a new DKGC node. In this model, the number of DKGC nodes does not increase.

In our model, we pick all the initiation nodes to be the DKGC nodes. The relationship among the number of DKGC nodes, the total number of nodes and threshold of the system will be further discussed in Section 3.3.

### 3.2 Proposed Scheme

The first certificateless public key encryption scheme was proposed by Al-Riyami and Paterson. We incorporate their work and adopt it to MANET key management with CL-PKE. The scheme is as follows:

- **Setup:**
  We assume \(IG\) is a Bilinear Diffie-Hellman parameter generator and \(k\) is the security parameter for the system. This algorithm has four steps.

  1. Run the \(IG\) generator on an input \(k\), it outputs \(\langle G_1, G_2, e \rangle\) where \(G_1\) and \(G_2\) are groups of prime order \(q\), \(e\): \(G_1 \times G_1 \rightarrow G_2\) is a pairing.
  2. Choose an arbitrary generator \(P \in G_1\).
  3. Select a master private key \(msk\) uniformly at random from \(Z_q^*\) and set \(P_0 = msk \times P\).
  4. Choose four cryptographic hash functions \(H_1: \{0,1\}^* \rightarrow G_1\), \(H_2: G_2 \rightarrow \{0,1\}\), \(H_3: \{0,1\}^m \times \{0,1\}^m \rightarrow Z_q^*\) and \(H_4: \{0,1\}^m \rightarrow \{0,1\}^m\), here \(m\) will be the bit-length of plaintexts.

The master public key \(mpk = \langle G_1, G_2, e, m, P, P_0, H_1, H_2, H_3, H_4 \rangle\). The master private key is \(msk \in Z_q^*\). The message space is \(M = \{0,1\}^m\) and the ciphertext space \(C = \{0,1\}^{2m} \times G_1\).

- **Extract-partial-secret-key:**
  This algorithm takes as input an \(ID \in \{0,1\}^*\) and carries out the following steps.

  1. Compute \(Q_{ID} = H_1(ID) \in G_1\).
  2. Output the partial private key \(d_{ID} = msk \times Q_{ID} \in G_1^t\).

Any user can verify its partial secret key by checking \(e(d_{ID}, P) = e(Q_{ID}, P_0)\).
We assume a polynomial $f(x)$ can be defined as

$$f(x) = msk + \sum_{i=1}^{t} (a_i x^i)$$

Where $a_1, a_2...a_t$ are uniformly distributed over a finite field $\mathbb{F}$. This algorithm takes as input an ID $\in \{0, 1\}^*$ and outputs a master secret key share $msks_i = f(ID_i)$. From this formula we can compute $msk$ by

$$f(0) = msk = \sum_{i=1}^{t+1} [\hat{L}(0, ID_i) \times f(ID_i)] \in \mathbb{Z}_q^*$$

and we also have

$$f(x) = \sum_{i=1}^{t+1} [\hat{L}(x, ID_i) \times f(ID_i)]$$

where $\hat{L}(\alpha, \beta)$ is the appropriate Lagrangian coefficients. Assuming $S = \{ID_1, ID_2, ID_3...ID_{t+1}\}$, then

$$\hat{L}(\alpha, \beta) = \prod_{\gamma \in S, \gamma \neq \alpha} (\alpha - \gamma) \prod_{\gamma \in S, \gamma \neq \beta} (\beta - \gamma)$$

**Extract-partial-secret-key-share-and-master-secret-key-share:**

Giving a master secret key share of node $i$ $msks_i$ and a new-joint node's ID $new$, this algorithm takes the following steps.

1. A partial secret key share is calculated by
   
   $$ds_{new,i} = \hat{L}(0, ID_i) \times msksi \times Q_{new} = \hat{L}(0, ID_i) \times f(ID_i) \times Q_{new} \in \mathbb{G}_1$$

2. A master secret key share is calculated by
   
   $$msks_{new,i} = \hat{L}(ID_{new}, ID_i) \times msksi \in \mathbb{Z}_q^*$$

**Extract-partial-secret-key-DKGC:**

This algorithm takes as input $t$ partial secret key shares $ds_{new,i}$, the partial secret key $dnew$ can be calculated by

$$dnew = \sum_{i=1}^{t+1} ds_{new,i} = \sum_{i=1}^{t+1} \hat{L}(0, ID_i) \times f(ID_i) \times Q_{new} = msk \times Q_{new} \in \mathbb{G}_1$$

**Extract-master-secret-key-shares:**

This algorithm takes as input $t$ master secret shares $msks_{new,i}$ and the $msk_{new}$ can be calculated by

$$msk_{new} = \sum_{i=1}^{t+1} \hat{L}(ID_{new}, ID_i) \times msksi = f(ID_{new}) \in \mathbb{Z}_q^*$$

**Set-user-keys:**

This algorithm takes as select a user's secret value $x_{ID} \in \mathbb{Z}_q^*$, input the master public key $mpk$ and user’s partial secret key $d_{ID}$. It outputs user’s secret key $sk_{ID} = x_{ID} \times d_{ID}$ and user’s public key $pk_{ID} = (X_{ID}, Y_{ID})$, where $X_{ID} = x_{ID} \times Q_{ID}$ and $Y_{ID} = x_{ID} msk_{ID}$.

**Encryption:**

For a message $msg \in M$ and an identity $ID \in \{0, 1\}^*$ with its public key $pk_{ID} = (X_{ID}, Y_{ID})$, the encryption algorithm takes as follows:

1. Check the public key by $e(X_{ID}, P_0) = e(Y_{ID}, P)$. If the result is negative, abort the encryption and output an error symbol.
2. Compute $Q_{ID} = H_1(ID) \in \mathbb{G}_1$.
3. Choose a random number $r \in \{0, 1\}^m$.
4. Set $r = H_3(sigma, msg)$
5. Compute and output ciphertext:

$$c = <r \times \mathbb{P}, \sigma \oplus H_2(e(Q_{ID}, Y_{ID})^r), msg \oplus H_4(sigma) >$$

**Decryption:**

Suppose $c = <U, V, W > \in C$. To decrypt this ciphertext with private key $sk_{ID}$:

1. Compute $V \oplus H_2(e(s_{k_{ID}}, U)) = \sigma'$
2. Compute $W \oplus H_4(\sigma') = msg'$
3. Set $r' = H_3(\sigma', msg')$ and test if $U = r' \times \mathbb{P}$. If not, output an error symbol and reject the ciphertext.
4. Output $msg'$ as the decryption of $c$.

**Correctness:**

$$\sigma' = V \oplus H_2(e(s_{k_{ID}}, U))$$

$$= V \oplus H_2(e(x_{ID} \times ID_{ID}, r \times \mathbb{P}))$$

$$= V \oplus H_2(e(x_{ID} \times msk_{Q_{ID}} \times \mathbb{P}'))$$

$$= V \oplus H_2(e(Q_{ID} \times x_{ID} msk_{\mathbb{P}})^r)$$

$$= V \oplus H_2(e(Q_{ID}, Y_{ID})^r)$$

$$= \sigma$$

$$msg' = W \oplus H_4(\sigma')$$

$$= msg \oplus H_4(\sigma') \oplus H_4(\sigma')$$

$$= msg$$

$$r' \times \mathbb{P} = H_3(\sigma', msg') \times \mathbb{P}$$

$$= H_3(\sigma, msg) \times \mathbb{P}$$

$$= U$$
3.3 Issues and design principles

We incorporate a distributed system to replace the KGC, so that the network becomes self-organized. This fully distributed system is based on the threshold cryptography with two patterns \((t, n)\). The pattern \(t\) represents the threshold of the model, which means any \(t+1\) malicious users can break the system (hence, the system is upperbounded by \(t+1\), which means that as long as there are at most \(t\) malicious users, then the system is considered to be at the ‘secure’ state). The pattern \(n\) represents the total number of users. We denote \(n'\) to be the maximum number of users, and \(t'\) to be the number of malicious users in the network at the initiation state. \(t'\) should be less than \(t\) to get the network initiated.

Unfortunately, we cannot anticipate if a new-joint node is malicious or not. If the system is based on fully distributed model, then in the worst case, all the new-joint nodes are malicious, which add up to \(n'-n+t'\) malicious DKGC nodes. In order to keep the system running well, this \(n'-n+t'\) should be smaller than \(t\). The system becomes vulnerable when \(t-t'\) nodes join the network.

If the system is based on the partially distributed model, every DKGC sends its data to a random non DKGC node before it goes offline. When \(t-t'\) original nodes goes offline, and they all replicate themselves to new-joint node, the system becomes vulnerable.

Fully distributed systems are more efficient, but only allow a small number of new-joint nodes. Partially distributed system can be secure as long as certain amount of origin nodes stay online, but it requires cooperation between DKGC nodes and new-joint nodes, and it brings along with extra communication overhead searching for DKGC nodes. Different systems should be chosen over different scenarios.

4 Simulation

4.1 Simulation with C

**Setup** In this simulation, we implement our scheme with C codes. The programming is based on Pairing Based Cryptography library (PBC) and GNU MP library (GMP), which define a large amount of efficient functions over pairing calculations. The programming environment is showing as follows:

<table>
<thead>
<tr>
<th>CPU</th>
<th>Intel T2250 1.73GHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ram</td>
<td>1GB</td>
</tr>
<tr>
<td>Hard Disk</td>
<td>80GB at 5400rpm</td>
</tr>
<tr>
<td>OS</td>
<td>Ubuntu 7.01</td>
</tr>
<tr>
<td>PBC lib version</td>
<td>0.4.17</td>
</tr>
<tr>
<td>GMP lib version</td>
<td>4.2.2</td>
</tr>
</tbody>
</table>

Table 1: Programming Environment

<table>
<thead>
<tr>
<th>Number of nodes</th>
<th>5</th>
<th>10</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keys from KGC</td>
<td>142.756</td>
<td>142.756</td>
<td>142.756</td>
</tr>
<tr>
<td>Key shares</td>
<td>13.165</td>
<td>11.315</td>
<td>10.189</td>
</tr>
<tr>
<td>Keys from DKGC</td>
<td>156.739</td>
<td>224.295</td>
<td>313.790</td>
</tr>
</tbody>
</table>

Table 2: Result of Simulation I

As shown in table 2, if the partial secret key comes from the KGC, it takes 142.7ms for a node to get its key. This time is consist of the time partial secret key generated by the KGC and the time a node generates its secret key/public key based on this partial secret key. On the contrary, if the partial secret key comes from DKGC nodes, the total generating time increases to 156.7ms for a network with 5 nodes, 224.3ms for a network with 10 nodes and 313.8ms for a network with 20 nodes. This time is comprised of the time for each DKGC node to generate the partial secret key shares (10-13ms) and the time the node generates the key based on these shares.

Note that this time will not change too much because all DKGC nodes generate partial secret key shares separately and parallel. The reason that key generating time is much higher than partial secret key generating time is that the key generating process involves a few pairing calculation over groups, while the partial secret key generating process only involves calculations over the infinite field.

4.2 Simulation with OPNET

**Scenarios** The second simulation runs over six scenarios:

1. 10 nodes in total running in partially distribution system, consist of 5 DKGC nodes, 1 type I attacker, 1 type II attacker and 3 normal nodes.
2. 10 nodes in total running in fully distribution system, all of them are DKGC nodes, consist of 1 type I attackers, 1 type II attackers and 6 normal nodes.
<table>
<thead>
<tr>
<th>Maximum Velocity</th>
<th>10m/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mobility Model</td>
<td>RWM</td>
</tr>
<tr>
<td>Pause Time</td>
<td>1 second</td>
</tr>
<tr>
<td>Dimensions of Space</td>
<td>100m × 100m</td>
</tr>
<tr>
<td>Radio Range</td>
<td>35m</td>
</tr>
<tr>
<td>Initiation Time</td>
<td>100 seconds</td>
</tr>
<tr>
<td>Background Traffic</td>
<td>1 packet per second</td>
</tr>
<tr>
<td>Packet Size</td>
<td>1024 bits</td>
</tr>
</tbody>
</table>

Table 3: The AODV Parameters

3. 10 nodes running in pure AODV system, with 1 type I attacker and 1 type II attacker.

4. 20 nodes in total running in partially distribution system, consist of 10 DKGC nodes, 2 type I attackers, 2 type II attackers and 6 normal nodes.

5. 20 nodes in total running in fully distribution system, all of them are DKGC nodes, consist of 2 type I attackers, 2 type II attackers and 16 normal nodes.

6. 20 nodes running in pure AODV system, with 2 type I attackers and 2 type II attackers.

The attackers are defined as follows:

- Type I attacker does not forward any packets. It works simply as a sink.

- Type II attacker does wrong routing. It sends packets to any node other than the correct node.

During the simulation, all the type II attackers forwards their packets to type I attackers.

AODV parameters The parameters of AODV are shown in Table 3.

In the simulation, all the nodes’ movement follows the random waypoint model [9] with a pause time of 1 second and a maximum velocity of 10m/s. This mobility model defines that node will pick some random waypoint in the wireless domain and move towards the waypoint with a velocity randomly picked between 0m/s(exclusive) and 10m/s(inclusive). Once a node gets to its destination, it will pause for 1 second and then move to the next waypoint. The movement repeats till the end of simulation.

The space of the wireless domain is 100m × 100m, and the propagation range for each node is 35 meters. When the simulation starts, there is an initiation time for 100 seconds, during which time, no traffic is generated, except that between nodes and the KGC. After that stage, the KGC goes offline and each normal node (including DKGC nodes) will generate a background traffic, which is 1 packet per second in our simulation. Once a packet received/generated, it take 0.04 second for a node to process it. This 0.04 second is the OPNET standard average propagation and processing delay. This delay increases to 0.055 second for DKGC nodes, which is because DKGC nodes need to have some extra time (10-13ms) to calculate partial secret key shares and validate public keys. The extra 10-13ms comes from the result of simulation 1.

Result As we can see from the figures, in a network with 10 nodes, our scheme generates around 30 percent more traffic than a pure AODV network, but the packet drop rate decreases to one quarter of pure network. The average route discovery time (0.38s) is a little higher than pure AODV network (0.32) at first but than decreases to 0.13s which is 60 percent of the pure AODV network(0.20s).
In a network with 20 nodes, our scheme contributes to the average route discovery time as well, around 0.41s with CL-PKE while 0.71s without CL-PKE. Nevertheless, the packet drop rate is higher than pure AODV network. This is probably because our scheme produces a lot more traffic overhead and some of them are dropped because of the Type I attacker.
5 Conclusion

This paper presented the design and the simulation of a key distribution scheme over mobile ad hoc network, based on the certificateless cryptography and threshold secret sharing scheme. In this work, we have successfully issued public/secret keys for users without providing certificates. Our scheme also ensures that system can work on self-organized networks after the initiation. From the simulation we found out that our scheme works extremely well in a small size of MANET. It reduces both packet drop rate and route discovery time for around 30 per cent, compared with pure AODV networks.

References


