南　　華　　大　　學

資訊管理學系

碩士論文

以Bilinear Pairings運算技術為基礎之協商金鑰生產程序在NTDR網路架構及智慧卡系統之運用

An Efficient Session Key Generation for NTDR Networks and Smart Card System Based On ID-based Bilinear Paring

研 究 生：陳宗亨

指導教授：周志賢

中華民國　97 年　6 月　30　日

南　華　大　學

資訊管理所

碩 士 學 位 論 文

以 Bilinear Pairings 運算技術為基礎之協商金鑰生產程序在 NTDR 網路架構及智慧卡系統之運用

An Efficient Session Key Generation for NTDR Networks and Smart Card System Based On ID-based Bilinear Paring

研究生：陳宗亨

經考試合格特此證明

口試委員：

指導教授：周志賢

系主任(所長)：

口試日期：中華民國　97 年　6 月　16 日

誌　　謝

本篇論文可以完成主要是仰賴指導教授周志賢老師以及陳雅玲學姐不辭辛苦的細心指導，在每次的討論中給予我寫作本篇論文的正確觀念及方向，在論文修改的期間更是細心地給予我指教及他們過去寶貴的寫作經驗，還有在百忙之中抽空為啟峰和我擔任口試委員的莊振村教授及邱宏彬教授所提出的修改意見，有了這些師長的細心教導才能使學生我可以有更縝密的思考來完成此篇論文。

在南華大學資訊管理所這兩年的生活，我相當地感謝我的父母及弟弟柏仲，在我低潮的時後給予我鼓勵及肯定，使我感受到家人的溫暖，在學校的生活中，首先要感謝我的直屬學長 晉城及仲儒教導我正確的 Bilinear pairings 運算技術的正確觀念，再來就是我的搭檔啟峰了，因為有同樣研究領域的搭檔可以相互討論問題才能快速找出問題的癥結所在以便快速解決問題，並且建立了更深入的觀念來加速本篇論文的生成，最後當然不能忘記的還有碩士班的學長 育弘和同學明勳、光南、宣均及淳淳，有了你們我這兩年的研究所生活才更加的有意義及豐富的色彩。

最後，在這兩年的生活中雖然辛苦，但也是因為有了家人、師長、同學及朋友的相互鼓勵及扶持我才能完成這兩年來的學業以及體驗到不同的學生生活，謝謝你們。

以 Bilinear Pairings 運算技術為基礎之協商金鑰生產程序在 NTDR 網路架構及智慧卡系統之運用

學生：陳宗亨　　指導教授：周志賢

南　華　大　學　資訊管理學系碩士班

摘　　要

由於近年來隨意型無線網路（例如：Mobile Ad Hoc Networks）的發展及成長快速為人們帶來了許多的便利性，但是大多缺乏了相對應的網路安全機制，導致資料有外洩的危險，因此我們以此為動機發展了利用 ID-based bilinear pairing 運算技術的安全協商金鑰產生程序來達到 NTDR（Near-Term Digital Radio networks）網路環境中的通訊安全。 基於 NTDR 的網路環境是將無線網路中節點（nodes）來做分群的叢聚型（cluster based）無線網路中的一種，因此適合在範圍大的無線環境中運作。在本研究中，我們所提出的方法不僅可以適用在大範圍的無線網路環境中，並且適合兩個節點間的安全通訊。最後，

我們更進一步的將上述的研究方法做進一步的研究，進而發展出另一套以 ID-based bilinear pairing 運算技術為基礎的智慧卡安全認證機制。

關鍵字：隨意型無線網路，bilinear pairing 運算技術，叢聚型無線網路，智慧卡。

An Efficient Session Key Generation for NTDR Networks and Smart Card System Based On ID-based Bilinear Paring

Student: Chen Tsung-Heng Advisor: Dr. Chou Jue-Sam

Department of Information Management
The M.I.M Program
Nan-Hua University

ABSTRACT

Near-Term Digital Radio (NTDR) network is a kind of mobile ad hoc network (MANET) in which mobile nodes are assigned into different clusters. Therefore, it can let the nodes to communicate with each other efficiently in a large area. Despite several NTDR protocols have been proposed, there still lacks an efficient secure one. Accordingly, in this paper, we propose a new method based on ID-based bilinear pairings to overcome the unsolved security problems nowadays. After our analysis, we conclude that our scheme is the first protocol for NTDR network that is not only secure but also very efficient. We also propose the authentication scheme for smart card system and show that the traditional smart card authentication techniques inevitably suffer the parallel session attack. Henceforth, we propose a novel scheme based on ID-based bilinear pairing which not only can avoid this attack.

Keyword: MANET, NTDR networks, bilinear pairings, smart card.

目　　錄

表　　目　　錄

圖　　目　　錄

1. Introduction

A mobile ad hoc network (MANET) does not need (or require less) any fixed infrastructure and can be constructed quickly. Moreover, the member nodes it encompasses can change frequently. Hence, MANET is suitable for some applications such as military missions, emergency handling, or rescue processing. However, due to the limitation of bandwidth and resource of MANET, designing a secure and efficient routing protocol in such a network is a great challenge. Despite this, many studies for MANET [1-7, 10, 12-20] have been proposed.

There are three types for the study of MANET routing protocols during 1999 to 2004. We list three proposals for a representation of each type: (1) Ad hoc on-demand distance vector routing (AODV) [1], (2) The dynamic source routing protocol (DSR) [2], and (3) Authenticated routing for ad hoc networks (ARAN) [3]. The routing protocols [1] and [2] do not take security into consideration while the routing protocol [3] intends to satisfy all of the security requirements. However, [12] indicated that [3] still has security flaws because the source node can not authenticate all intermediate nodes in the routing path. In 2005, Liaw et al. [19] proposed a secure key exchange protocol for MANET. However, we found that there is a weakness in their protocol, i.e. the session key can be compromised by the intercepted messages. We will describe this in Section 3. In 2007, Zhou et al. [20] proposed an access control in wireless sensor networks, but we found that the CA in their scheme may become a

bottleneck when there are enough pairs of nodes waiting to communicate with each other. It may happen that the corresponding life times of the nodes' signatures are overdue. Moreover, the negotiated session key n_{ij} of any pair, for nodes n_i and n_j in Zhou et al.'s scheme, has no forward and backward secrecy.

NTDR network is a kind of MANET in which mobile nodes are assigned into different clusters. Therefore, nodes can communicate with each other efficiently in a large area. In 1997, Ruppe R. et al. [17] first proposed a NTDR network system for military missions but it lacks security considerations. Recently, several NTDR related studies [4, 5, 6] are proposed. In 2004, Varadharajan et al. [4] proposed a scheme using public key infrastructure (PKI). However, in 2005, Chang et al. [5] pointed out that using PKI would be a heavy burden for the computation of each mobile node. Hence, they proposed a protocol based on Diffie-Hellman method. In 2007, Lee and Chang [6] proposed an identity-based NTDR scheme. However after our analysis, we found that both of the protocols [5] and [6] have some security weaknesses. In [5], the communicating parties do not verify the certificates of each other. In [5] and [6], the hash value of the session key is clearly transferred over the communication line. This makes the session key become weaker. We will describe this in Section 3.

Moreover, we also develop a password-based authentication scheme for smart card system which demands a system to verify the legality of a user for preventing any malicious depredations. Under this requirement, a

smart card with storage and computation ability [3] is integrated into the system for its convenience and portability. It stores a user's ID together with his password and usually allows password to be changed freely.

There are many password authentication schemes using smart cards proposed [1-14, 24, 25]. In 1999, Yang and Shieh [1] proposed both of a timestamp-based and a nonce-based password authentication schemes with smart cards. They claim that their scheme needs not to hold a verified table of passwords and allow the users to select or change passwords freely. However, in 2002, Chan and Cheng [2] pointed out that Yang-Shieh's schemes [1] are vulnerable to both of the given-ciphertext and forgery attacks. In 2003, Cheng and Zhong [5] also proposed an attack on their scheme [1]. But in 2003, Sun and Yeh [4] indicated that Chan and Cheng's attack was unreasonable since the client's ID forged doses not exist in the ID table as mentioned in [9]. They also showed that Yang-Shieh's schemes were subject to the forgery attack. In 2005, Yang and Wang [11] improved Yang-Shieh's schemes to resist Sun-Yeh's attack. However, in 2005, Kim et al.' [24] points out Yang-Wang time-based password authentication scheme suffers forgery attack. In 2005, Das et al. [25] proposed a novel remote user authentication scheme using bilinear pairings. But in 2005, we [26] pointed out their scheme suffer the forgery attack. In 2004, Das et al. [12] proposed a dynamic ID-based remote user authentication scheme. Yet, in 2006, Misbahuddin et al. [13] pointed out that [12] can not resist both of an insider attack and an impersonation attack. In 2004, Ku et al.[22] proposed an efficient password based remote server authentication scheme using smart card. However, in 2007,

Wang et al. [14] showed Ku et al.s' scheme were vulnerable to the guessing, forgery and denial of service (DoS) attacks. Hence, they proposed an efficient scheme to improve the scheme. In 2007, Cheng et al. [23] proposed security enhancement of an IC-card-based remote login mechanism to improve the scheme [27] proposed in 2004.

In this paper, we propose a novel ID-based password authentication scheme to resolve this inherent problem.

This paper is organized as follows. The introduction is presented in Section 1 and the background is shown in Section 2. In Section3, we review the two protocols, [5, 19]. After that, we show our protocols in Section 4 and Section 5. Then the security analyses are described in Section 6 and Section 7. Then performance comparisons are made in Section 8 and Section 9 respectively. Finally, a conclusion is given in Section 10.

2. Background

2.1 Bilinear pairings

In 1984, Shamir [21] proposed an ID-based encryption and signature schemes, in which each user uses his identity as his public key. This makes the key distribution easier than the conventional ones. In 2001, Boneh and Franklin [25] first proposed a practical ID-based cryptosystem using bilinear pairing on elliptic curve.

Bilinear pairings, such as Weil pairing and Tate pairing, defined on elliptic curves are proved efficient [23] and thus applied to cryptosystem gradually. Moreover, studies shows that the computational cost of both Weil pairing and Tate pairing, each is 5 to 10 times the cost of scalar multiplication of a point on an elliptic curve [30, 32, 33, 34]. Many protocols have been designed based on the Weil pairing [8, 9, 11, 26, 27, 28, 29]. Now, we briefly introduce Weil pairing which will be applied in our study as follows.

Let P be a generator of group G_1 over a elliptic curve with order q and G_2 be a multiplicative group of the same order. It is assumed that solving the discrete logarithm problem (DLP) in both G_1 and G_2 is difficult [25]. Let e: $G_1 \times G_1 \rightarrow G_2$ be a Weil pairing which has the following properties [23].

(1) Identity: For all $P \in G_1$, $e(P, P) = 1$,

(2) Alternation: For all $P_1, P_2 \in G_1$, $e(P_1, P_2) = e(P_2, P_1)$,

(3) Bilinearity: For all $P_1, P_2, P_3 \in G_1$, $e(P_1 + P_2, P_3) = e(P_1, P_3)e(P_2, P_3)$, and $e(aP_1, bP_2) = e(P_1, P_2)^{ab}$,

(4) Non-degeneracy: For all $P_1, P_2 \in G_1$, $e(P_1, P_2) \neq 1$.

Moreover, some well-known assumptions related to our study are listed as follows.

(1) Discrete logarithm problem (DLP): The DLP is to compute a when given aP, where $a \in Z_q^*$.

(2) Computational Diffie-Hellman problem (CDHP): The CDHP is to compute abP when given P, aP and bP, where $a, b \in Z_q^*$.

(3) Bilinear computational Diffie-Hellman problem (BCDHP): The BCDHP is to compute $e(P, P)^{abc}$ when given P, aP, bP and cP, where a, b and $c \in Z_q^*$.

2.2 The NTDR network system

The NTDR network is designed to efficiently use the limited resources for a large environment in which mobile nodes are classified into clusters. Each cluster is composed of a clusterhead which handles and manages the cluster, and mobile nodes controlled by the clusterhead. In the cluster, any two authorized members within one hop apart, e.g. nodes A and B in Figure 1, can directly communicate to each other while nodes B and C depart more than one hop only can communicate via the clusterhead. This case is a so-called intra-cluster communication. The other case is the inter-cluster communication, in which the two communicating nodes belongs two different clusters separately. They

should communicate to each other through their own clusterheads, e.g, nodes *X* and *Y* shown in Figure 2. Such a design philosophy has the following two advantages: (1) it can use limited network resources efficiently due to the necessity of communicating via the clusterhead when nodes are not within one hop, and (2) a clusterhead can monitor all the nodes well when they transmit messages through it.

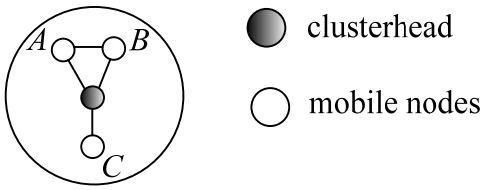


Fig. 1. Nodes *B* and *C*, beyond one hop apart in the same cluster, must communicate through the clusterhaed.

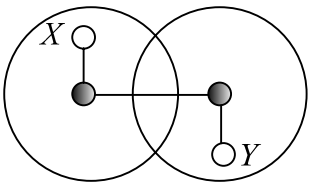


Fig. 2. Nodes *X* and *Y*, in different clusters, must communicate with each other through their own clusterhead, respectively.

2.3 Security requirements in a NTDR network

For NTDR network is a kind of MANET. In this section, we will review the requirements of a secure communication for MANETs. The security requirements are not only for MANETs but also for traditional wired or infrastructure-based wireless networks. We delineate them as follows.

(1) Authentication: Only authorized users can be allowed to communicate with each other.

(2) Confidentiality: Only authorized users can decrypt the encrypted messages.

(3) Data-integrity: The messages transmitted in the network should not be maliciously interpolated.

(4) Non-repudiation: A user can not deny that he had sent the message

sent by him before.

(5) Non-impersonation: Malicious users should not be able to impersonate any authorized user to send or obtain valid information.

(6) Against key-compromise impersonation (KCI) attack: The KCI attack means if the private key of user A is compromised, then an adversary can impersonate the other user to communicate with him. Thus, a secure protocol should resist against such an attack.

(7) Against man-in-the-middle attack: The man-in- the-middle attack means that an adversary E intercepts the transmitted messages between A and B and modifies them to make two session keys which can enable E to impersonate A to B and impersonate B to A, respectively.

(8) The forward secrecy: When a user is revoked by the group manager or leaves the group, he should not learn any future messages in the group.

(9) The backward secrecy: When a user becomes a new member of a group, he should not get any valid messages transmitted before in the group.

3. Review of Chang et al.s' protocols [5], Liaw et al.s' methods [19]

In this section, we first show the definitions of used notations in the authentication phase proposed by Chang et al. in 2005 [5] in Section 3.1, then we briefly review the authentication phase of their scheme in Section 3.2. In Section 3.3, we also review the scheme proposed by Liaw et al. [19].

3.1 Definitions of used notations in [5]

The notations used in [5] and [6] are as follows:

mh_i: the identity of mobile node i,

$CERT_X$: the certificate of node X,

CID_j: the identity of cluster j,

$CHID_j$: the identity of the clusterhead that dominates cluster j,

$E_K[M]/D_K[M]$: the encryption/decryption result of the message M encrypted/decrypted by the key K,

T: a timestamp,

AUC: an authentication token,

K_{ij}: the session key shared by mobile node i and clusterhead j.

3.2 Review of the authentication phase of [5]

In 2005, Chang et al. proposed a DH-based secure communication method for cluster-based ad hoc networks, but we found that there are two weaknesses in the authentication phase. First, when a mobile host mh_i

enters the radio range of a cluster CID_j and is detected by the clusterhead $CHID_j$, $CHID_j$ and mh_i both will transmit their corresponding certificates to each other for authentication as shown in Figure 3. But neither of them checks the validity of the certificate of the other party. Thus, an adversary can easily impersonate one party to the other. In other words, their scheme can not achieve the goal of mutual authentication as claimed. Secondly, the authentication token AUC ($=H(K_{ij})$) is the hash value of the session key K_{ij} and is transferred clearly over the communication line. This may make the session key K_{ij} insecure. As we know, collision finding attack significantly threats the security of a hash function [35]. Additionally, due to progress in computer hardware, the speed of computation under network-computer cooperation is fast-growing and thus the attacker may have a non-negligible probability to find the preimage from several collisions of the hash $H(K_{ij})$. The same security vulnerability of the session key is also found in their later work [6]. We denote this vulnerability as weak session key.

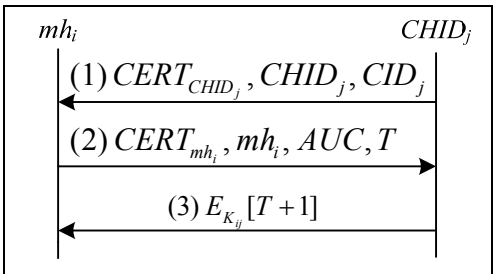


Fig. 3. The authentication phase in [5]

3.3 Review of Liaw et al.s' protocol [19]

In 2005, Liaw et al. proposed a secured key exchange protocol for a MANET. However we find that, in their protocol, an adversary can easily obtain the session key shared between any two nodes. In the following,

we list the definitions of used notations in their scheme then review their method, after that, we present our attack.

3.3.1 Definitions of used notations

The definitions of used notations in Liaw et al.s' protocol are listed as follows:

KGC: the key generation center,

ID_i: the identification of user i,

p, q: two large strong primes,

n: the product of p and q; $n = pq$,

$\phi(n) = (p-1)(q-1)$,

e: the public key of KGC,

d: the private key of KGC; $d = e^{-1} \bmod \phi(n)$,

α : a primitive element of $GF(p)$ and $GF(q)$,

f( ): a one-way hash function, and

g_i: the signature for user i computed by KGC,

T_i: a timestamp.

3.3.2 The four phases of Liaw et al.s' protocol

We describe the four phases of Liaw et al.'s protocol as follows:

(a) *Initialization phase*

In this phase, the KGC calculates his public key as (n, e) and private key as (p, q, $d, \phi(n)$). KGC also public parameters (G, g, α), where G is a multiplicative group, g is its generator and $\alpha \in G$.

(b) ***Registration phase***

In this phase, when user *i* wants to register to the KGC, he sends his identification ID_i to KGC and obtains a KGC's signature $g_i = ID_i^d \bmod n$. KGC can then be closed or off-line. (However, we consider that this assumption is not practical. Since nodes in MANETs may change very frequently, KGC had better keep on-line for any node's registration).

(c) ***user verification phase***

Whenever two users, *i* and *j*, want to communicate with each other, they need to verify each other using the following steps.

Step1: User *i* chooses a random number r_i and calculates two ephemeral public keys as $y_i = g_i \cdot \alpha^{r_i} \bmod n$ and $t_i = r_i^e \bmod n$. Then he uses the identity of user *j*, ID_j, and timestamp T_i to generate $f(y_i, t_i, T_i, ID_j)$ and then computes $s_i = g_i \cdot r_i^{f(y_i, t_i, T_i, ID_j)} \bmod n$. Finally, he sends his identity ID_i, y_i, t_i, s_i, and T_i to user *j*.

Step2: Similarly, user *j* sends ID_j, $y_j (= g_j \cdot \alpha^{r_{ji}} \bmod n)$, $t_j (= r_j^e \bmod n)$, $s_j (= g_j \cdot r_j^{f(y_{ji}, t_{ji}, T_{ji}, ID_i)} \bmod n)$ and timestamp T_j to user *i*.

Step3: After receiving the messages from each other, user *i* checks to see whether $s_j^e = ID_j \cdot t_j^{f(y_j, t_j, T_j, ID_i)} \bmod n$ holds. If it holds, then he authenticates that user *j* is valid. Similarly, user *j* verifies user *i* by checking to see whether the equation $s_i^e = ID_i \cdot t_i^{f(y_i, t_i, T_i, ID_j)} \bmod n$ holds.

(d)*key exchanging phase*

After completing the user verification phase, user i and j can compute their session key as $SK_i = \left(\frac{y_j^e}{ID_j}\right)^{r_i} \bmod n = \left(\frac{y_i^e}{ID_i}\right)^{r_j} \bmod n = SK_j = \alpha^{er_ir_j} \bmod n$.

3.3.3 Our attack

We now describe our attack on Liaw et al.'s scheme to obtain the session key SK_i (= SK_j) shared between user i and j. We describe it as follows:

Assume that an adversary E has intercepted the public transmitted information t_i and t_j, and set his identifier as $ID_E = t_i \cdot t_j \bmod n$. He then sends his identity ID_E to KGC. After receiving ID_E from E, KGC computes

$$g_E = ID_E^d \bmod n = (t_i \cdot t_j)^d \bmod n = (r_i \cdot r_j)^{e \cdot d} \bmod n = (r_i \cdot r_j)^{e \cdot e^{-1}} \bmod n = r_i \cdot r_j \bmod n$$

and then sends g_E to E. After obtaining g_E, by using the public parameters α and e, E can compute the session key shared between user i and j as $SK_E = \alpha^{er_ir_j} \bmod n = SK_i (= SK_j)$. Accordingly, E can decrypt any encrypted messages transmitted between user i and j. That is, we have a successful attack.

4. Our proposed protocols

In this section, we first describe the environment of our protocol in Section 4.1 and then list the definitions of used notations in Section 4.2. Finally, we present our scheme in Section 4.3. Our protocol bases on the NTDR network model without using PKI and includes three phases as follows: (1) session key generation phase for nodes in a cluster, (2) group key generation phase for: (a) all members in a cluster, (b) all clusterheads in the system, and (3) session key generation phase for nodes in different clusters. These phases can make a member node generate a session key to securely communicate with his clusterhead, or a member, within or beyond one-hop apart, in the same cluster or in different cluster. Moreover, we also develop an authentication protocol based on ID-based bilinear pairings for securing smart card systems in section 4.4.

4.1 Definitions of used notations

We define the notations used in our protocol as follows:

G_1: a cyclic additive group with order q,

G_2: a cyclic multiplicative group with order q,

e: $G_1 \times G_1 \rightarrow G_2$ be a bilinear paring,

P: a generator of G_1,

$H(.)$: an one-way hash function which maps a point in G_1 to a bit string,

level 0-1: a level composed of level 0 and level 1 in which the public/private key pair for mobile node i at level 0 is $Q_{M_i} / S_{M_i} (= s_j Q_{M_i})$

and the public/private key pair for clusterhead j (CH_j) at level 1 is $P_{CH_j}(=s_jP)/s_j$,

level 1-2: a level composed of level 1 and level 2 in which the public/private key pair for clusterhead j at level 1 is $Q_{CH_j}/S_{CH_j}(s_{root}Q_{CH_j})$ and the public/private key pair for *rootTA* at level 2 is P_{root} (= $s_{root}P$)/s_{root},

key pair architecture: a key hierarchy composed of level 0-1 and level 1-2,

M_i: the identity of mobile node i,

CH_j: the identity of clusterhead j which manages cluster j,

CID_j: the identity of cluster j,

s_j: the private key of CH_j at level 1 which is also a TA of CID_j in level 0-1,

$P_{CH_j}(=s_jP)$: the public key of CH_j at level 1 in level 0-1,

$S_{M_i}(=s_jQ_{M_i})$: the long-term private key of M_i at level 0 issued by CH_j at level 1 in level 0-1,

$Q_{M_i}(=H(M_i))$: the long-term public key of M_i at level 0 in level 0-1,

s_{root}: the private key of *rootTA* at level 2 which is a TA for all clusterheads at level 1 in level 1-2,

P_{root} (= $s_{root}P$): the public key of *rootTA* at level 2 in level 1-2,

$S_{CH_j}(=s_{root}Q_{CH_j})$: the long-term private key of CH_j at level 1 issued by *rootTA* at level 2 in level 1-2,

Q_{CH_j}: the long-term public key of CH_j at leve l in level 1-2,

r_i: the short-term private key of M_i which is a random number chosen by

M_i,

$P_{M_i} (= r_i P)$: the short-term public key of M_i,

CP_{CH_j} : the short-term public key of CH_j,

$K_{M_i H_j}$: the session key shared between M_i and CH_j,

SK_{AB}: the session key shared between mobile nodes A and B,

$SK_{H_1 H_2}$: the session key shared between CH_1 and CH_2,

CGK_j: the group key of cluster j,

$CHGK$: the group key of all clusterheads in the clusterhead group managed by *rootTA*,

RS: remote server,

U_n: the user n,

IP_n: the IP address of U_n,

IP_{RS}: the IP address of RS,

ID_n: the identity of user n,

CID_n: the identity of smart card (of user n),

PW_n: the password of user n,

s: the secret key chosen by RS,

$P_{RS} = sP$: the corresponding RS public key of s,

$Q_n = H(ID_n)$: the public key of smart card (of user n),

$S_n = sQ_n$: the private key of smart card,

y: a secret number stored in the smart card (of each registered user),

T_i: a timestamp of the message i,

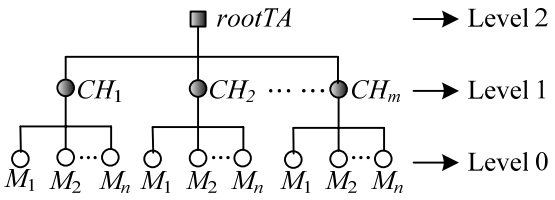
ΔT_{ij} : the valid time interval between T_i and T_j,

$E_K[M]/D_K[M]$: the encryption/decryption result of message M

en/decrypted by key K.

4.2 Two-level hierarchy environment

In our protocol, the nodes in the system are classified into clusters. Each cluster has a unique clusterhead, which is a trust party (TA) for all other nodes in the cluster. We assume that when a node joins to a cluster, he would not move to another cluster like a soldier would not leave his troop in the battlefield as indicated in [17]. Besides, all clusterheads are managed by a TA named *rootTA*. In other words, our protocol is a 2-level structure in hierarchy as illustrated in Figure 6. In the figure, the clusterhead CH_j at level 1 plays a role as a group manager for level 0, choosing his private key s_j, publishing his public key P_{CH_j}, and distributing a valid private key $s_j Q_{M_i}$ for each group member M_i whose public key is Q_{M_i} via secure channel in the initialization phase.



Key pair architecture

Level 0 – 1:

The public/private key pair for mobile node i at level 0 is $Q_{M_i}/S_{M_i}(=s_j Q_{M_i})$, and the public/private key pair for clusterhead j (CH_j) at level 1 is $P_{CH_j}(=s_j P)/s_j$.

Level 1 – 2:

The public/private key pair for clusterhead j at level 1 is $Q_{CH_j}/S_{CH_j}(=s_{root} Q_{CH_j})$, and the public/private key pair for *rootTA* at leve2 1 is $P_{root}(=s_{root} P)/s_{root}$.

Fig. 4. Two-level environment in our protocol

Similarly, the *rootTA* at level 2 plays as a group manager for level 1, controlling all clusterheads, choosing his private key s_{root}, publishing his public key P_{root}, and distributing a valid private key $s_{root}Q_{CH_j}$ for each group member CH_j whose public key is Q_{CH_j} via secure channel in the initialization phase.

4.3 Our Proposed Scheme

In this Section, we describe the three phases in our schemes of NTDR network and smart card system as follows. They are session key generation phase in a cluster, group key generation phase, and session key generation phase in different clusters, as described in section 4.3.1, 4.3.2, and 4.3.3 respectively. Finally, we describe authentication protocol for smart card system in section 4.4.

4.3.1 Session key generation phase in a cluster

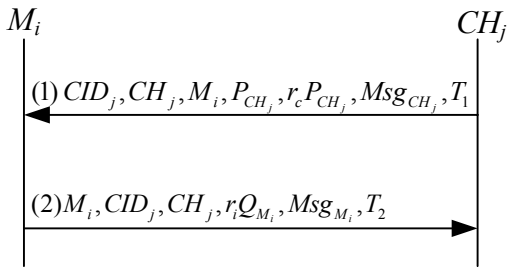
In this phase, we describe the following three cases:

(a) ***For a node entering a cluster to communicate with his clusterhead***

In this case, when a mobile node M_i enters the radio range of cluster j and is detected by the clusterhead CH_j, CH_j (whose private key is s_j and public key is P_{CH_j}) and M_i (whose private key is S_{M_i} and public key is Q_{M_i}) will generate their session key as illustrated in Figure 7 which is also described by the following steps.

Step1: CH_j chooses a random number r_c, computes $r_c P_{CH_j}$ and $Msg_{CH_j} (= H(e(h_{ji} \times Q_{M_i}, r_c s_j P_{CH_j}))$ where $h_{ji} = H(e(s_j P_{CH_j}, Q_{M_i})))$.

As we know h_{ji} is equal to $h_{ij}(=H(e(s_jQ_{M_i},P_{CH_j})))$, which is the default pre-shared secret information between M_i and CH_j. When having prepared $r_cP_{CH_j}$ and Msg_{CH_j}, CH_j sends the beacon message composed of CID_j, CH_j, M_i, P_{CH_j}, $r_cP_{CH_j}$, Msg_{CH_j} and timestamp T_1 to M_i.



$$Msg_{CH_j} = H(e(h_{ji} \times Q_{M_i}, r_c s_j P_{CH_j}))$$

$$Msg_{M_i} = H(e(s_jQ_{M_i}, h_{ij} \times r_i r_c P_{CH_j}))$$

Fig. 5. Session key generation phase for a node with his clusterhead

Step2: After receiving the beacon message from CH_j, M_i checks the validity of timestamp T_1. If T_1 is valid, he computes $Msg_{CH_j}'(=H(e(s_jQ_{M_i}, h_{ij} \times r_cP_{CH_j})))$, where h_{ij} is equal to h_{ji} and checks to see if Msg_{CH_j}' is equal to Msg_{CH_j}. If they are not equal, M_i drops the communication. Otherwise, M_i chooses a random number r_i and computes $Msg_{M_i}(=H(e(s_jQ_{M_i}, h_{ij} \times r_i r_c P_{CH_j})))$. Having prepared $r_iQ_{M_i}$ and Msg_{M_i}, M_i sends M_i, CID_j, CH_j, $r_iQ_{M_i}$, Msg_{M_i} and

timestamp T_2 to CH_j. Finally, M_i can expect a session key shared with CH_j: he first computes the pre-session key $K_{M_i-CH_j}(=e(S_{M_i}, r_i \times P_{CH_j}) = e(s_j Q_{M_i}, r_i s_j P) = e(Q_{M_i}, P)^{r_i s_j^2})$; and then obtains the session key by computing $K_{M_i H_j}(=H(K_{M_i-CH_j}||M_i||CH_j))$.

Step3: After receiving the messages from M_i, CH_j checks to see if the timestamp T_2 is valid, if it is not valid, he terminates the communication, else he computes $Msg_{M_i}'(=H(e(h_{ji} \times r_i Q_{M_i}, r_c \times s_j P_{CH_j})))$. If Msg_{M_i}' equals Msg_{M_i}, CH_j accepts and identifies M_i as valid. Then, CH_j can compute the session key shared with M_i by first computing the pre-session key $K_{CH_j-M_i}(=e(r_i Q_{M_i}, s_j P_{CH_j}) = e(Q_{M_i}, P)^{r_i s_j^2})$ and then obtains the session key by computing $K_{M_i H_j}(=H(K_{CH_j-M_i}||M_i||CH_j))$.

(b) ***For nodes to communicate with each other within one hop in the same cluster***

Before nodes can communicate with each other, we assume that all nodes have done the mutual authentication with their corresponding clusterheads. When two nodes are within one hop in the same cluster CID_j, they can communicate to each other directly. We delineate the session key generation under this situation in Figure 8 and describe it using the following steps.

Step1: M_A chooses a random number r_A, computes $r_A Q_{M_A}$ and sends M_A, CID_j, M_B, $r_A Q_{M_A}$ and timestamp T_1 to M_B.

Step2: After receiving the message from M_A, M_B checks the validity of T_1. If it is valid, M_B selects a random number r_B, computes $r_B Q_{M_B}$ and then sends M_B, CID_j, M_A, $r_B Q_{M_B}$ and timestamp T_2 to M_A. After that, he computes the pre-session key as $K_{BA}(= H(e(S_{M_B}, h_{BA} \times r_B r_A Q_{M_A})))$ and computes the session key as $SK_{BA}(= H(K_{BA}||M_A||M_B))$, where the pre-computed value h_{BA} in K_{BA} is equal to $H(e(S_{M_B}, Q_{M_A}))$.

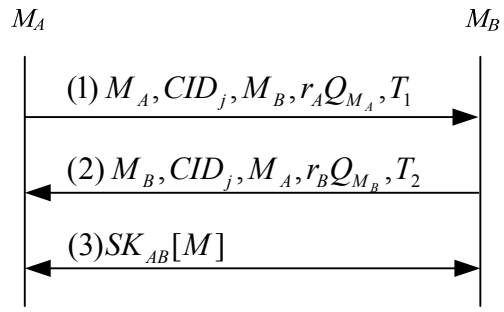


Fig. 6. Session key generation for nodes within one hop apart to communicate directly in the same cluster

Step3: When obtaining the message sent from M_B, M_A checks the validity of T_2. If T_2 is invalid, M_A drops the communication; otherwise, he computes the pre-session key as $K_{AB}(= H(e(S_{M_A}, h_{AB} \times r_A r_B Q_{M_B})))$ and then computes the session key as $SK_{AB}(= H(K_{AB}||M_A||M_B))$, where the

pre-computed value h_{AB} in K_{AB} is equal to $H(e(S_{M_A}, Q_{M_B}))(= h_{BA})$.

After completing above steps, the two nodes, M_A and M_B, each can obtain his session key SK_{AB}.

Similarly, we can use the same method to generate the session key between any two clusterheads, CH_1 and CH_2, by replacing M_A with CH_1 and M_B with CH_2, respectively. We show the computation of the session key $SK_{H_1H_2}$ for CH_1 and CH_2 as follows:

For CH_1, he computes $SK_{H_1H_2}(= H(e(S_{CH_1}, h_{12} \times r_{CH_1} r_{CH_2} Q_{CH_2}) || CH_1 || CH_2))$, and for CH_2, he computes $SK_{H_1H_2}(= H(e(h_{21} \times r_{CH_2} r_{CH_1} Q_{CH_1}, S_{CH_2}) || CH_1 || CH_2))$. Where Q_{CH_1} and Q_{CH_2} are the corresponding public key of CH_1 and CH_2, h_{12} (= h_{21}) is the pre-computed value, r_{CH_1} and r_{CH_2} are two random numbers chosen by CH_1 and CH_2, respectively and s_{root}, used in both S_{CH_1} and S_{CH_2}, is the private key of *rootTA* who is a TA of all clusterheads.

(c) ***For nodes, beyond one-hop apart in the same cluster, to communicate with each other through the clusterhead***

In this case, as a symmetric key cryptosystem is more efficient than asymmetric one, our protocol uses symmetric approach among

intermediate nodes and asymmetric approach between end-to-end nodes which is a similar method used in cases (a) and (b). We assume that there are two nodes, M_A and M_B, in the same cluster but not within one-hop apart, want to transmit messages to each other through the clusterhead CH_j. Under this situation, we delineate how they can get their session key in Figure 9 and also describe it using the following steps.

Step1: M_A selects a random number r_A to compute $r_A Q_{M_A}$. He then transmits M_A, CID_j, CH_j, M_B, $r_A Q_{M_A}$ and timestamp T_1 to CH_j.

Step2: When receiving the message sent by M_A, CH_j checks the validity of T_1. If the message is in time, CH_j chooses a random number r_C and uses the session key $K_{M_B H_j}$, shared with M_B, to encrypt r_C and a timestamp T_2. He then sends CH_j, CID_j, M_B, M_A, $r_A Q_{M_A}$ together with the encrypted message $E_{K_{M_B H_j}}[r_C, T_2]$ to M_B.

Step3: After obtaining the encrypted message from CH_j, M_B decrypts it using the session key shared with CH_j, obtaining r_C and T_2. M_B then checks the validity of timestamp T_2. If T_2 is overdue, he rejects the communication; otherwise, he chooses a random number r_B and computes $r_B Q_{M_B}$. He then sends M_B, CID_j, CH_j, M_A, $r_B Q_{M_B}$ and timestamp T_3 to CH_j. After this, M_B can compute the pre-session key

$K_{BA}(= H(e(r_C S_{M_B},\ h_{BA} \times r_B r_A Q_{M_A})) = H(e(Q_{M_A}, Q_{M_B})^{h_{AB} r_A r_B r_C s_j}))$, and then computes the session key (shared with M_A) as $SK_{BA}(= H(K_{BA}||M_A||M_B))$, where h_{AB} in K_{AB} is equal to $H(e(S_{M_A}, Q_{M_B}))$.

Step4: After receiving the encrypted message sent by M_B, CH_j checks the validity of T_3. If T_3 is not valid, CH_j stops communicating with M_B; otherwise, he uses the session key $K_{M_A H_j}$ shared with M_A to encrypt the message including r_C and timestamp T_4. Then CH_j sends this encrypted message along with CH_j, CID_j, M_A, M_B and $r_B Q_{M_B}$ to M_A.

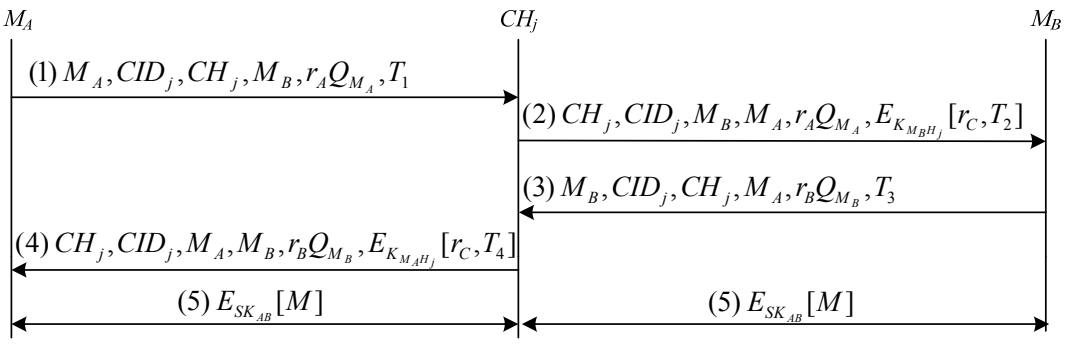


Fig. 7. Session key generation for nodes communicating through the clusterhead in the same cluster

Step5: When receiving the encrypted message sent by CH_j, M_A uses the session key $K_{M_A H_j}$ to decrypt this encrypted message, obtaining r_C and timestamp T_4. Then M_A checks the validity of T_4. If T_4 is valid, he computes the pre-session key $K_{AB}(= H(e(h_{AB} \times r_A r_B Q_{M_B},\ r_C S_{M_A})) = K_{BA})$ and then computes

the session key as $SK_{AB}(=(K_{AB} \| M_A \| M_B))$, where h_{AB} is equal to $H(e(S_{M_B}, Q_{M_A}))(= h_{BA})$.

4.3.2 Group key generation phase for: (a) a cluster, and (b) the group of all clusterheads

In this phase, we describe the group key generation phase in two cases: (a) group key generation for a cluster, and (b) group key generation for the group of all clusterheads.

(a) *Group key generation for a cluster*

Here, we assume that there are n - 1 mobile nodes in the cluster. They are $M_1, M_2, \ldots, M_{n-1}$. When a new node M_n joins the cluster j and finishes the authentication protocol (as in 4.3.1 (a)), the clusterhead CH_j starts to launch the group key generation phase, also known as re-keying process. We delineate the group key generation phase for a cluster in Figure 10 and describe it using the following steps.

Step1: CH_j first sends his identity CH_j, CID_j, M_i, the encryption of a r_G (randomly chosen by CH_j) and CH_j by using $K_{M_iH_j}$ (the session key built between M_i and CH_j), and a timestamp T to M_i, (for i = 1 to n).

Step2: After each M_i obtaining the message from CH_j, he checks the validity of timestamp T. If it is not reasonable, M_i drops the communication. Otherwise, he chooses a random number r_i, computes $P_{M_i}(= r_iP)$, and lets P_{M_i} be a

short-term public key. Then he broadcasts (M_i, P_{M_i}, $H(K_{M_iH_j}, P_{M_i}, M_i)$, T+1) to all nodes in the cluster, where $H(K_{M_iH_j}, P_{M_i}, M_i)$ functions as a commitment of value P_{M_i}.

Step3: On receiving M_i, P_{M_i}, $H(K_{M_iH_j}, P_{M_i}, M_i)$ and timestamp T+1 from each M_i, CH_j engages for checking if $H(K_{M_iH_j}, P_{M_i}, M_i)$ consists with P_{M_i}. If they are not, CH_j will warn all member nodes to ignore the invalid message.

Step4: On receiving message (1) from CH_j, node M_i in the cluster decrypts the encryption of r_G and CH_j, obtaining r_G and CH_j. And on receiving the broadcast messages (2) from other nodes in the cluster, node M_i uses r_G and all P_{M_j} for all $j \neq i$, in the corresponding broadcast message together with his own to calculate the cluster group key CGK using the following equation.

$$\begin{aligned} CGK &= e(P_{M_1}, r_G P_{CH_j}) \cdot e(P_{M_2}, r_G P_{CH_j}) \cdots\cdots e(P_{M_n}, r_G P_{CH_j}) \\ &= e(r_1 P, r_G s_j P) \cdot e(r_2 P, r_G s_j P) \cdots\cdots e(r_n P, r_G s_j P) \\ &= e(P, P)^{r_G s_j (r_1 + r_2 + \cdots + r_n)} \end{aligned}$$

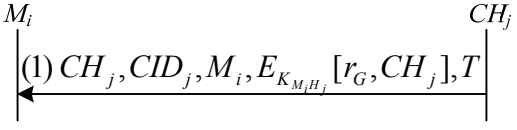


Fig. 8. (a) CH_j sends r_G to M_i, for i = 1 to n, for generating group key

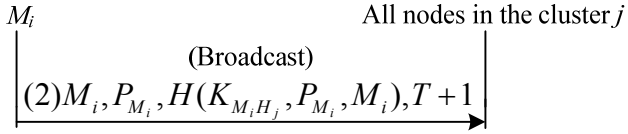


Fig. 8. (b) M_i broadcasts the message (2) to all nodes in the cluster j for generating group key

(b) ***Group key generation for the group of all clusterheads***

The computation of clusterhead group key (*CHGK*) for the group of all clusterheads is similar to the computation of the cluster group key (*CGK*) in a cluster as mentioned above in case (a), just by replacing CH_j with *rootTA* and M_i with CH_i. We list the calculation of the *CHGK* by the following equation.

$$CHGK = e(CP_{CH_1}, r_{CH} s_{root} P) \cdot e(CP_{CH_2}, r_{CH} s_{root} P) \cdot \cdots \cdot e(CP_{CH_m}, r_{CH} s_{root} P)$$
$$= e(P,P)^{r_{CH} s_{root} (c_1 + c_2 + \cdots + c_m)},$$

where $CP_{CH_1}, ..., CP_{CH_m}$ are the short-term public keys of clusterhead 1 to clusterhead *m* and r_{CH} is a random number chosen by *rootTA*.

For in a cluster-based ad hoc network, nodes in a cluster may change frequently. Therefore, for consideration of the forward and backward secrecy, the corresponding computation, of the cluster group key in (a) or the clusterhead group key in (b), needs to be recalculated once a member in a cluster, or the member itself is a clusterhead in the group of all clusterheads, has changed.

4.3.3 Session key generation phase for nodes in different clusters

After completing (b) in Section 4.3.2, we now describe how two nodes in different clusters can compute their session key. Here, we also adopt both the symmetric and asymmetric approaches for efficiency to design our protocol. We assume that mobile nodes M_A and M_B are in two different clusters, CID_1 and CID_2, respectively but M_A does not know which cluster M_B belongs to. We depict the process in Figure 11 and

describe the protocol using the following steps.

Step1: M_A in cluster CID_1 chooses a random number r_A and computes $r_A S_{M_A}$. Then he uses the session key $K_{M_A H_1}$, shared with clusterhead CH_1, to encrypt M_B, CH_1, $r_A S_{M_A}$ and timestamp T_1, obtaining message Msg1. Then, M_A sends M_A, CID_1, CH_1 and Msg1 to CH_1.

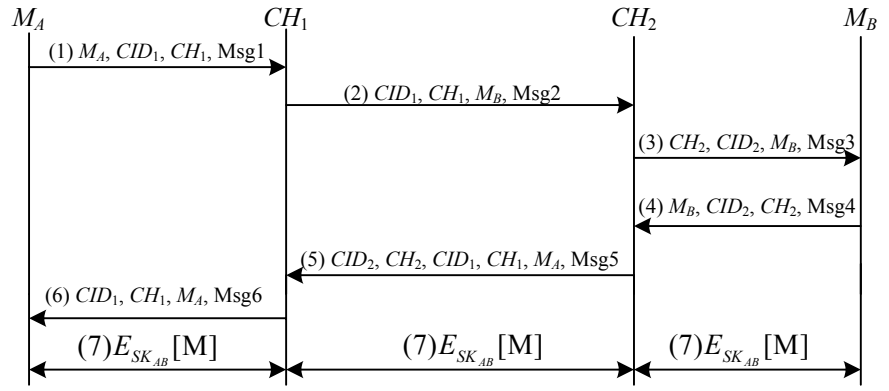
Step2: After receiving Msg1, CH_1 uses the session key $K_{M_A H_1}$ to decrypt Msg1, obtaining M_B, CH_1, $r_A S_{M_A}$ and T_1. He then checks the validity of timestamp T_1. If T_1 is not valid, CH_1 terminates the communication with M_A; else, he chooses a random number r_x and uses the clusterhead group key $CHGK$, shared with all other clusterheads, to encrypt CID_1, M_A, M_B, $r_A S_{M_A}$, r_x and timestamp T_2, obtaining Msg2. CH_1 then broadcasts Msg2 together with CID_1, CH_1, M_B to all clusterheads in the network.

Step3: When all clusterheads in the network receiving the broadcast message from CH_1, they each can examine their database to see if M_B is in his cluster. If M_B is his member, say CH_2, CH_2 can use the clusterhead group key $CHGK$ to decrypt Msg2 and then checks the validity of the timestamp T_2 and confirms that M_B is really in the decryption of Msg2. If both are valid, CH_2 selects a random number r_y and uses session key $K_{M_B H_2}$, shared with M_B, to encrypt CID_1, M_A,

$r_A S_{M_A}$, r_x, r_y, and timestamp T_3, obtaining Msg3, then CH_2 sends CH_2, CID_2, M_B and Msg3 to M_B.

Step4: After receiving Msg3 from CH_2, M_B uses the session key $K_{M_B H_2}$ to decrypt it, obtaining CID_1, M_A, $r_A S_{M_A}$, r_x, r_y, and timestamp T_3. He then checks the validity of T_3. If T_3 is not correct, he terminates the communication with CH_2; otherwise, he randomly chooses a number r_B and then encrypts CID_1, M_A, M_B, $r_B S_{M_B}$ and timestamp T_4 by using the session key $K_{M_B H_2}$, obtaining Msg4. Then M_B sends M_B, CID_2, CH_2 and Msg4 to CH_2. After this, M_B can compute the pre-session key $K_{BA}(= e(S_{M_B}, r_B r_x r_y r_A S_{M_A}) = e(Q_{M_A}, Q_{M_B})^{r_A r_B r_x r_y s_1 s_2})$ and thereafter computes the session key as $SK_{BA}(= H(K_{BA} \| M_A \| M_B))$.

Step5: After receiving Msg4 from M_B, CH_2 decrypts it using session key $K_{M_B H_2}$. Then, he checks the validity of timestamp T_4. If T_4 is overdue, he terminates the communication; else, he uses the session key $SK_{H_1 H_2}$ to encrypt CID_2, M_B, M_A, $r_B S_{M_B}$, r_x, r_y, and timestamp T_5, obtaining Msg5. Then CH_2 sends CID_2, CH_2, CID_1, CH_1, M_A and Msg5 to CH_1.



$\text{Msg1} = E_{K_{M_A H_1}}[M_B, CH_1, r_A S_{M_A}, T_1]$

$\text{Msg2} = E_{CHGK}[CID_1, M_A, M_B, r_A S_{M_A}, r_x, T_2]$

$\text{Msg3} = E_{K_{M_B H_2}}[CID_1, M_A, r_A S_{M_A}, r_x, r_y, T_3]$

$\text{Msg4} = E_{K_{M_B H_2}}[CID_1, M_A, M_B, r_B S_{M_B}, T_4]$

$\text{Msg5} = E_{SK_{H_1 H_2}}[CID_2, M_B, M_A, r_B S_{M_B}, r_x, r_y, T_5]$

$\text{Msg6} = E_{K_{M_A H_1}}[CID_2, M_B, r_B S_{M_B}, r_x, r_y, T_6]$

r_A: random number chosen by M_A

r_B: random number chosen by M_B

r_x: random number chosen by CH_1

r_y: random number chosen by CH_2

Fig. 9. Nodes communicate in two different cluster

Step6: After receiving Msg5, CH_1 uses session key $SK_{H_1 H_2}$ to decrypt it, obtaining CID_2, M_B, M_A, $r_B S_{M_B}$, r_x, r_y and T_5. Then he checks the validity of T_5 and confirms that M_A is really in the decryption of Msg5. If both are valid, he uses the session key $K_{M_A H_1}$ to encrypt CID_2, M_B, $r_B S_{M_B}$, r_x, r_y and timestamp T_6, obtaining Msg6. Then he sends CID_1, CH_1, M_A and Msg6 to M_A.

Step7: When receiving Msg6 from CH_1, M_A uses session key $K_{M_A H_1}$ to decrypt it and checks the validity of the timestamp T_6. If T_6 is valid, then he computes the pre-session key $K_{AB} (= e(S_{M_A}, r_A r_x r_y r_B S_{M_B}) = e(Q_{M_A}, Q_{M_B})^{r_A r_B r_x r_y s_1 s_2})$ then computes session key SK_{AB} as

$$SK_{AB}(= H(K_{AB} \| M_A \| M_B) = H(K_{BA} \| M_A \| M_B) = SK_{BA}).$$

4.4 Our authentication protocol for smart card system

In this section, we proposed our authentication protocol for smart card system. They are (1) setup phase, (2) registration phase, (3) login phase (4) authentication phase and (5) password change phase respectively.

(1) Setup phase

RS chooses his private key *s*, computes his public key as P_{RS} (= sP) and publishes it.

(2) Registration phase:

When a user U_n wants to register to *RS* through a secure channel, he does following steps which are also depicted in Figure 12.

Step 1: First, he chooses his password PW_n and submits his ID_n and $h(PW_n)$ to the *RS*.

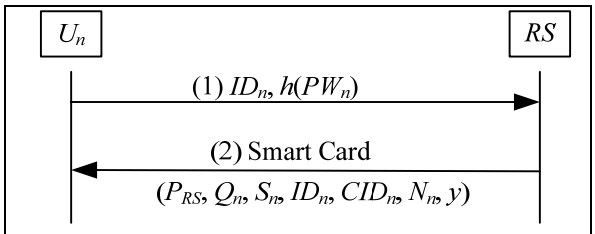


Fig. 10. The registration phase of our scheme

Step 2: After receiving ID_n and $h(PW_n)$, *RS* computes a key pair for U_n which are public key Q_n (= $H(ID_n)$) and private key S_n (= sQ_n). Then he computes $N_n = h(h(PW_n) \oplus y)$. Finally, he

personalizes the smart card by storing P_{RS}, Q_n, S_n, ID_n, CID_n, N_n and y to it through a secure channel, where y is a secret serial number chosen by *RS* for each registered user.

(3) Login phase

When a user wants to login to *RS*, he inserts his smart card into the input device and keys in his ID_n and PW_n. The smart card will compute as follows. We also depict it in Figure 3.

Step1: It chooses a random number r_n, computes r_nQ_n, $c_1 = h(y \cdot h(PW_n)) \cdot IP_n$ and $Msg_1 = e(r_nS_n, c_1P_{RS})$.

Step 2: Then it sends the login message consisting of ID_n, CID_n, N_n, r_nQ_n, Msg_1, IP_n and T_1 to *RS*, where T_1 is the current device time of U_n.

(4) Authentication phase

(a) *RS* authenticate U_n

After receiving login message from U_n at time T_2, *RS* executes the following steps which are also depicted after step 3 in Figure 13.

Step 3: *RS* checks the validity of ID_n and CID_n, if one of them is not valid then *RS* rejects the login message. Else, *RS* checks the validity of time interval between T_1 and T_2. If $(T_2 - T_1) \le \Delta T_{12}$ dose not hold, *RS* rejects the login request, else he computes $c_1' = h(y \times h(PW_n)) \cdot IP_n$ and verifies the validity of Msg_1 by computing $Msg_1' = e(c_1'r_nQ_n, sP_{RS})$. If Msg_1' is equal to Msg_1,

RS accepts U_n's login request.

Step 4: After accepting the login from U_n, *RS* chooses a random number r_R, computes $r_R P_{RS}$, $Y = h(N_n \| y \| ID_n \| CID_n \| IP_{RS})$ and $Msg_2 = e(r_R Y Q_n, sP_{RS})$.

Step 5: He sends $r_R P_{RS}$, Msg_2, IP_{RS} and T_3 to the smart card, where T_3 is the current time of *RS*.

(b) U_n authenticates *RS*

Step 6: After receiving $r_R P_{RS}$, Msg_2, IP_{RS} and T_3 at time T_4 from *RS*, smart card checks to see whether $(T_4 - T_3) \le \Delta T_{34}$ holds or not. If they hold, he computes $Y^* = h(N_n \| y \| ID_n \| CID_n \| IP_{RS})$ then verifies to see whether $Msg_2 = Msg_2' = e(S_n, Y^* r_R P_{RS})$ holds or not. If it holds, smart card authenticates the *RS*.

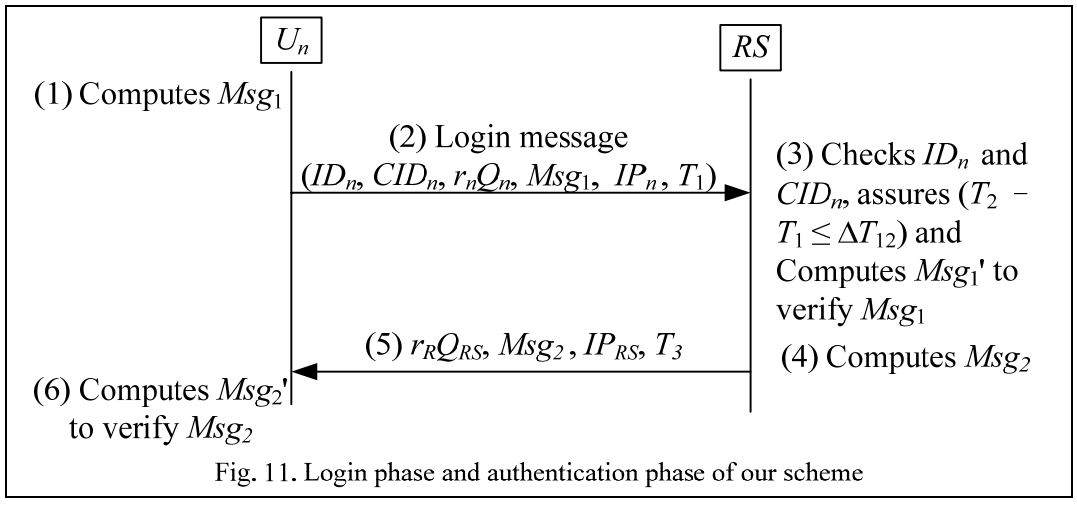


Fig. 11. Login phase and authentication phase of our scheme

(5) Password changing phase

In this phase, U_n can change his password at his will by doing the following steps:

Step 1: U_n inserts his smart card into the device then keys in his ID_n and PW_n.

Step 2: U_n submits his new password PW_n^* to the smart card. The smart card then computes $N_n^* = h(h(PW_n^*) \oplus y) \oplus N_n \oplus h(h(PW_n) \oplus y)$.

Step 3: By this way, the password can be changed from PW_n to PW_n^* successfully.

5. Security analysis

In this section, we describe security analysis. They are security analysis of protocols for NTDR networks and smart card systems in section 5.1 and section 5.2 respectively.

5.1 Security analysis of our protocol for NTDR network

In this section, we discuss the security of our proposed protocols. We show that our protocol can satisfy all the security requirements in the session key establishment including: (1) mutual authentication (2) against KCI attack (3) against man in the middle attack (4) the forward/backward secrecy. We describe them as follows.

(1) Mutual authentication

Here, we claim that only two intended members can communicate with each other in our protocol. Because if a user wants to become a new member of cluster j, he must register to CH_j in the initialization phase (as described in Section 4.1). TA (CH_j) then distributes each member M_i with a private key through a secure channel. As we know that TA has private key s_j and public key $P_{CH_j}(=s_jP)$; M_i has private key $S_{M_i}(=s_jQ_{M_i})$ and public key Q_{M_i}. In the following, we describe how our protocol can achieve mutual authentication in different cases.

(a) ***Session key generation phase for a node to communicate with his clusterhead***

When CH_j detects M_i entering his radio range, CH_j triggers the mutual authentication protocol (as described in Section 4.3.1 (a)). In the protocol, the first message, Msg_{CH_j} made by CH_j, can act as an identification token for CH_j to prove himself to M_i. We now analyze the computation of $Msg_{CH_j} = H(e(h_{ji} \times Q_{M_i}, r_c s_j P_{CH_j}))$ as follows: (1) $h_{ji} = H(e(s_j P_{CH_j}, Q_{M_i}))$ is the pre-shared secrecy between M_i and CH_j, and intuitively if M_i can verify that the other party has the same pre-shared secrecy as his own, he then believes that the other party he is talking to is the true CH_j; (2) r_c is a random challenge number to assure every session key being fresh, preventing from replay attack, (3) the two factors h_{ji} and r_c in (1) and (2) are combined and protected by the bilinear pairing form, which had been proved computationally infeasible based on Bilinear Diffie-Hellman assumption [25]. According to the property of bilinear pairing, M_i can compute $Msg_{CH_j}'(= H(e(s_j Q_{M_i}, h_{ij} \times r_c P_{CH_j})))$, by using his private key and the pre-shared secrecy $h_{ij}(= H(e(s_j Q_{M_i}, P_{CH_j})) = h_{ji})$, to verify the validity of Msg_{CH_j}. Similarly, the second message, Msg_{M_i} computed by M_i, also plays the role of identification proof for M_i to be authenticated by CH_j. The security analysis is similar to the above mentioned.

(b) ***Session key generation for nodes to communicate directly with each other in the same cluster***

For session key generation protocol in case (b) of Section 4.3.1, we claim that it can achieve implicit mutual authentication. The implicit mutual authentication means two parties can not succeed to generate the same session key except that they are the ones whom are believed by each other mutually. We analyze the computation of session key $SK_{AB} = H(K_{AB}||M_A||M_B)$, where $K_{AB} = H(e(S_{M_A}, h_{AB} \times r_A r_B Q_{M_B}))$, as follows: (1) h_{AB} is the pre-shared secrecy between M_A and M_B, and intuitively if M_A can verify that the other party can compute the same session key, M_A then confirms the party he is talking to is the true M_B. (2) r_A is a random challenge number to assure every session key being fresh, preventing from the replay attack, (3) the two factors h_{AB} and r_A in (1) and (2) are combined and protected by the bilinear form which had been proved to be computationally infeasible, (4) M_A provides his private key S_{M_A} in computing K_{AB}. According to the property of bilinear pairing, M_B can compute the same session key $SK_{BA}(= H(e(S_{M_B}, h_{BA} \times r_B r_A Q_{M_A})||M_A||M_B) = SK_{AB})$ by using his private key S_{M_B}, the pre-shared secrecy h_{BA} $(= H(e(S_{M_B}, Q_{M_A})) = h_{AB})$, and M_A's $r_A Q_{M_A}$. If SK_{BA} computed by M_B is equal to SK_{AB} computed by M_A, M_B implicitly authenticates M_A. Similarly, M_A can implicitly authenticate M_B.

(c) ***Session key generation phase for nodes in the same cluster to communicate through their clusterhead***

For the same reason, we also claim that our protocol can achieve implicit mutual authentication in case (c) of Section 4.3.1. We analyze the computation of $SK_{AB} = H(e(h_{AB} \times r_A r_B Q_{M_B}, r_C S_{M_A}) || M_A || M_B)$ as follows: (1) h_{AB} is the pre-shared secrecy between M_A and M_B, r_A chosen by M_A are two random challenge numbers to ensure every session key being fresh, preventing replay attack, (2) r_C chosen by CH_j, and (3) h_{AB}, r_A, r_C are combined and protected by the bilinear form. According to the property of bilinear pairing, M_B can compute the same session key $SK_{BA} (= H(e(r_C S_{M_B}, h_{BA} \times r_B r_A Q_{M_A}) || M_A || M_B) = SK_{AB})$ by using his private key S_{M_B}, r_C transmitted from clusterhead, the pre-shared secrecy h_{BA}, and $r_A Q_{M_A}$ transmitted from M_A. Thus, if SK_{BA} computed by M_B is equal to SK_{AB} computed by M_A, M_B implicitly authenticates M_A. Similarly, M_A can implicitly authenticate M_B by the same reason.

(d) ***Session key generation phase for nodes to communicate in different clusters***

In Section 4.3.3, M_A and M_B communicate in two different clusters. Thus, they need to communicate through their own clusterheads, CH_1 and CH_2. The session key $SK_{AB} (= H(K_{AB} \| M_A \| M_B) = H(K_{BA} \| M_A \| M_B))$ has the similar construction as above protocols, and thus can achieve implicit mutual

authentication for the same inference. The difference is that M_A and M_B's identities and other information are transferred via CH_1 and CH_2 using symmetric encryption. The random values of r_x and r_y are chosen by CH_1 and CH_2 respectively and contributed to SK_{AB} (SK_{BA}) for ensuring each session key's freshness.

(2) Against KCI attack

KCI attack means that when a node has been compromised, an adversary can impersonate any other node to communicate with the compromised node. In the following, we describe how our protocol can resist such KCI attack.

(a) *Session key generation phase for a node to communicate with his clusterhead* (*Section 4.3.1* (*a*))

Here, we assume that the private key $S_{M_i}(=s_jQ_{M_i})$ of M_i had been compromised to an adversary E. We want to show that E still can not impersonate any node, says CH_j, to communicate with M_i.

For message (1) shown in Figure 5, E can easily forge a valid Msg_{CH_j} by computing $h_{ji}=H(e(S_{M_i},P_{CH_j})(=H(e(Q_{M_i},s_jP_{CH_j}))$ and then $H(e(h_{ji}\times S_{M_i},r_cP_{CH_j}))(=H(e(h_{ji}\times Q_{M_i},r_cs_jP_{CH_j}))$ if he knows M_i's private key S_{M_i}. For message (2), E can just accept it. But E still fails to compute the valid session key for he still needs to compute the pre-session key $K_{CH_j-M_i}(=e(r_iQ_{M_i},s_jP_{CH_j})$. However, he only knows

S_{M_i} and P_{CH_j} (public key of CH_j), he can not figure out r_i which is protected by the encryption form $r_i Q_{M_i}$. According to DLP assumption, it is computationally infeasible to extract r_i from $r_i Q_{M_i}$. Therefore, E fails to generate a shared session key with M_i. In other words, E can not launch a KCI attack.

(b) ***Session key generation phase for nodes to communicate directly with each other in the same cluster* (*Section 4.3.1* (*b*))**

For the protocol in Section 4.3.1 (b), if an adversary E compromises M_A's private key $S_{M_A} (= s_j Q_{M_A})$ and therefore obtains h_{AB} $(= H(e(S_{M_A}, Q_{M_B})))$. We want to show that adversary E still can not successfully compute the valid session key shared with M_A. Because he needs to compute the pre-session key $K_{BA} (= H(e(S_{M_B}, h_{BA} \times r_B r_A Q_{M_A})) = H(e(Q_{M_B}, Q_{M_A})^{s_j \times h_{BA} \times r_B \times r_A}))$. However, he only knows S_{M_A}, h_{AB} (= h_{BA}), r_B (chosen by E) and Q_{M_B}, he can not figure out r_A which is protected by the encryption form $r_A Q_{M_A}$. According to DLP assumption, it is computationally infeasible for extracting r_A from $r_A Q_{M_A}$. Therefore, E eventually fails to generate a shared session key with M_A. That is, E can not launch a KCI attack.

(c) ***Session key generation phase for nodes in the same cluster to communicate through their clusterhead*** **(*Section 4.3.1 (c)*)**

For the protocol in Section 4.3.1 (c), if E compromises M_A, and thus knows $S_{M_A}(=s_J Q_{M_A})$, h_{AB} and $K_{M_A H_j}$. E still can not successfully compute the valid session key shared with M_A for he needs to compute the pre-session key $K_{BA}(=H(e(r_C S_{M_B}, h_{BA} \times r_B r_A Q_{M_A})) = H(e(Q_{M_B}, Q_{M_A})^{s_J \times h_{BA} \times r_A \times r_B})$. However, he only knows r_C (decrypted by $K_{M_A H_j}$), S_{M_A}, h_{AB} (= h_{BA}), r_B (chosen by E) and Q_{M_B}, he can not figure out r_A which is protected by the encryption form $r_A Q_{M_A}$. Therefore, E eventually fails to generate a shared session key with M_A and hence can not succeed in such a KCI attack.

(d) ***Session key generation phase for nodes to communicate in different clusters*** **(*Section 4.3.3*)**

For the protocol in Section 4.3.3, if an adversary E compromises M_A, and thus knows $S_{M_A}(=s_1 Q_{M_A})$, h_{AB} and $K_{M_A H_1}$. We want to show that E still can not successfully compute the valid session key shared with M_A. For he needs to compute the pre-session key $K_{BA}(=e(S_{M_B}, r_B r_x r_y r_A S_{M_A}) = e(Q_{M_A}, Q_{M_B})^{r_A r_B r_x r_y s_1 s_2})$. However, he only knows r_x, r_y (decrypted by $K_{M_A H_1}$), S_{M_A}, h_{AB} (= h_{BA}) and Q_{M_B}, he can not figure out r_A which is protected by the encryption form $r_A S_{M_A}$. Therefore, the adversary E fails to launch such a KCI attack.

(3) Against man-in-the-middle attack (MIMA)

MIMA is an attack that an adversary M_E slinkingly intercepts the communication line between the two communication parties and uses some means to make them believe that they each other were talking to the intended party. For illustrating the MIMA resistance of our scheme in the following, we first take a classical MIMA on a scheme slightly modified from our protocol in Section 4.3.1 (b). Let the scheme lacks h_{AB} (h_{BA}) when the two parties compute pre-session key, i.e., the pre-session key is computed as $K_{AB} = H(e(S_{M_A}, r_A r_B Q_{M_B}))$ (K_{BA} be $H(e(S_{M_B}, r_B r_A Q_{M_A}))$). The MIMA on the scheme is illustrated in Figure 14. As shown in the figure, M_E can generate a pre-session key K_{EA} which is same as K_{AE} computed by M_A. Similarly, he also can generate a pre-session key K_{EB} which is same as K_{BE} computed by M_B. Hence, M_E has a successful MIMA.

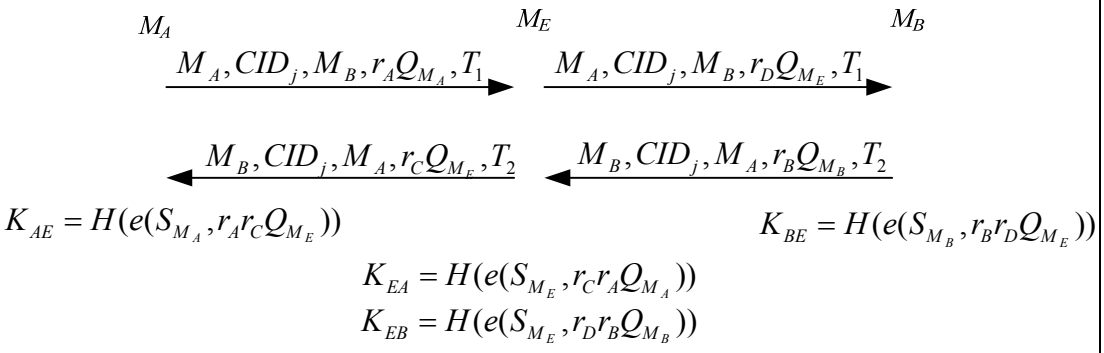


Fig. 12. The MIMA in Section 4.3.1 (b) if it lacks h_{AB} and h_{BA}

For resisting such a MIMA, the general suggestion is to add the IDs of the two parties' into the session key; however, it is not yet provable secure. So, except adding IDs into the session key, we also add h_{AB} (h_{BA}) which is the pre-shared secrecy between the two communicating parties,

and is known to M_E only with a negligible probability. That is, in our scheme, we add h_{AB} in the computation of pre-session key $K_{AB} = H(e(S_{M_A}, h_{AB} \times r_A r_B Q_{M_B}))$ and hence in the computation of session key $SK_{AB} = (K_{AB} \| M_A \| M_B)$. It is computationally infeasible for M_E to compute h_{AB} in K_{AB} based on BCDH assumption [25]. Therefore, we conclude that our scheme can resist against MIMA.

(4) The backward secrecy

Backward secrecy means that when a node becomes a new member of a cluster, it can not learn any past transmitted messages. We assume that a group key $CGK = e(P,P)^{r_G s_j (r_1 + r_2 + \cdots r_{n-1})}$ is shared by the cluster *j* which includes a clusterhead CH_j and *n* - 1 member nodes M_k, where *k* = 1 to *n* – 1. When a new node M_n joins into cluster *j*, he must be first authenticated by CH_j as described in Section 4.3.2 (a), and then CH_j will launch a rekeying process. In the process, each node in cluster *j* including CH_j itself will compute a new group key $CGK' = e(P,P)^{r_G' s_j (r_1' + r_2' + \cdots + r_{n-1}' + r_n)}$.

Apparently, *CGK′* is not equal to the old group key *CGK* and it is impossible for M_n to figure out the old group key *CGK* from the new *CGK′*. In other words, M_n can not use this new cluster group key *CGK′* to decrypt any messages encrypted by key *CGK*. Therefore, our protocol can achieve the backward secrecy property.

(5) The forward secrecy

Forward secrecy means that when a user is revoked by the group manager or leaves the group, he can not learn any future messages in the group. We assume that a group key $CGK = e(P,P)^{r_G s_j (r_1 + r_2 + \cdots r_n)}$ is shared by the cluster j which includes a clusterhead CH_j and n member nodes M_i, for i = 1 to n. When node M_n leaves cluster j, he is first authenticated by CH_j as described in Section 4.3.2 (a), and then CH_j will launch a re-keying process. In the process, each node in cluster j including CH_j itself will compute a new group key $CGK'' = e(P,P)^{r_G'' s_j (r_1'' + r_2'' + \cdots + r_{n-1}'')}$.

Apparently, CGK'' is not equal to CGK and it is impossible for M_n to figure out the new group key CGK'' from the old CGK. In other words, M_n can not use the old cluster group key CGK to decrypt any future messages encrypted by CGK''. Therefore, our protocol possesses the forward secrecy property.

5.2 Security analysis of our protocol for smart card system

In this section, we discuss the security of our proposed scheme for smart card systems in the following aspects including: (1) Mutual authentication and replay attack, (2) man-in-the-middle-attack, and (3) KCI attack.

(1) Mutual authentication and replay attack

Here, we first show that our scheme can achieve mutual authentication and then show its resistance of replay attack.

In the login and authentication phases (as described in Sections 5.2 (3) and (4)), the login message Msg_1, sent from smart card to *RS*, can act as an identification token for the smart card to prove himself to *RS*. The computation of c_1 in $Msg_1 = e(r_n S_n, c_1 P_{RS})$ is equal to $h(y \cdot h(PW_n)) \cdot IP_n$ which can be viewed as a pre-shared secrecy between smart card and *RS* multiplied by the user's IP, IP_n. Since y and $h(PW_n)$ are sent through a secure channel in the registration phase (as described in Section 5.2 (2)), if *RS* can verify the other party has the same c_1 as his own by assuring that Msg_1 is equal to Msg_1', he then believes the other party is the intended party. Similarly, the message Msg_2 computed by *RS* also plays the role of identification proof for the smart card to authenticate *RS*. From the analysis, we can see that our scheme can achieve mutual authentication.

Since r_n and r_R are two random challenge numbers chosen by smart card and *RS* respectively to assure every login request being fresh and are protected by the computations of Msg_1 and Msg_2 which are based on the bilinear pairing, we can see that a replay attack to our system is hence infeasible.

(2) Man-in-the-middle-attack (MIMA)

MIMA is an attack that an adversary *E* intercepts the communication line between two communicating parties and makes them believe that they each other are talking to the intended party, but indeed both of them are talking to him (*E*). In the following, we describe why our scheme can

avoid this type of attack.

In our scheme, the smart card uses the pre-shared secrecies (shared with *RS*), y and $h(PW_n)$, to compute $c_1 (= h(y \cdot h(PW_n)) \cdot IP_n)$, which is then protected by the computation of $Msg_1 = e(r_n S_n, c_1 P_{RS})$. Since any one of the pre-shared secrecies is known to *E* only with a negligible probability and it is computationally infeasible for *E* to compute c_1 from Msg_1 due to the BCDH assumption. It is impossible for *E* to compute a valid Msg_1 to pass the verification of *RS* without knowing y, $h(PW_n)$ and r_n. The security of the second message Msg_2 made by *RS* can be analyzed in a similar fashion shown as the above mentioned. Hence, we can conclude that our scheme can resist against MIMA.

(3) KCI attack

KCI attack means that when the secrecy of a party has been compromised, an adversary can impersonate any party to communicate with the compromised party. In the following, we describe how our scheme can against such a KCI attack.

Here, we only show the case that even if the private key S_n (= sQ_n) of U_n had been compromised to an adversary *E*. *E* still can not impersonate *RS* to communicate with the smart card. Because *E* can not know the hashed password $h(PW_n)$, the secret serial number y which is chosen by *RS* and the random number r_n protected by the encryption form r_nQ_n. It is computationally infeasible for *E* to extract r_n from r_nQ_n according to the BDLP assumption. Therefore, *E* can not generate the

correct Msg_1 to launch such a KCI attack. The other case (If the private key s of RS is compromised) can be analyzed in a similar fashion.

6. Performance and security comparisons

In this section, we describe performance and security comparisons. They are comparisons of protocols for NTDR networks and smart card systems in section 6.1 and section 6.2 respectively.

6.1 Comparisons of protocols for NTDR networks

In this section, we compare our protocol with recent NTDR-related schemes [4, 5, 6]. Due to schemes [5] and [6] proposed by the same research team and [5] has a outstanding mistake that the certificates of both parties are not checked by each other, we only compare the newest one [6]. The performance comparisons of the four phases in the three protocols are shown in Table 1 through Table 4 and the security comparison is shown in Table 5.

For performance comparison, we first list three facts that will be used to estimate the computational cost of schemes [4, 6] and ours.

(a) From [36], we know that a g^k mod p (where p is a 1024-bit prime) operation is estimated as $1.5\times|k|$ times the cost of a 1024-bit modular multiplication (1024-MM in brief) by using square-and-multiply algorithm. We denote this operation as $|k|$-Exp.

(b) According to Koblitz et al.'s study [31], an operation of kP (where $k \in Z_q^*$, $P \in G$, and G is a group over an elliptic curve with order q) is estimated as 29 times the cost of a 1024-MM. We denote this scalar multiplication as SM.

(c) According to studies [30, 32, 33, 34], a bilinear pairing is estimated as 5 to 10 times the cost of a SM. If on the average, we use 7.5 times as estimation, a bilinear pairing is approximately 218 times the cost of a 1024-MM. we denote this operation as BP.

Now, considering scheme [4], the computational cost of M_i in the case (a) is 3 public key operations (PKO). For public key cryptosystems, RSA and ElGamal are two most popular schemes and the later is more efficient than the former in general. Hence, we use ElGamal cryptosystem to estimate the cost of 3 PKO. As an ElGamal public key encryption is about 2 times the cost of a 160-Exp (i.e., two 160-Exps in ciphertext (g^k, $m\beta^k$) where k is a random integer, m is the message to be encrypted and β is the receiver's public key) and thus is 480 (= 2×1.5×160) times the cost of a 1024-MM. Therefore, the cost of 3 PKO is 1440 (= 3×480) times the cost of a 1024-MM. On the other hand, the computational cost of CH_j in case (a) of scheme [4] can be estimated in the same way and the result is also 1440 times the cost of a 1024-MM.

For scheme [6], the computational cost of M_i in the case (a) is one time the cost of a 320-Exp (= 1.5×320 = 480 times the cost of a 1024-MM), one hash operation and one symmetric encryption/decryption; it has the same computational cost for CH_j in the case (a).

For our scheme in case (a), we can pre-compute the pairing h_{ij} and thus the computational cost of M_i is one BP (218 times the cost of a 1024-MM), two SMs (58 times the cost of a 1024-MM) and two hash

operations. Hence, the total computational cost is approximately 276 times the cost of a 1024-MM. Similarly, the cost of CH_j is the same. We list all above results in table 1.

Moreover, the cost for two nodes building a session key, which are within one hop in the same cluster, can be estimated by using the same method described above. The result of comparisons is listed in Table 2.

Table 1: The cost comparison of the session key building for a node with his clusterhead (case (a) in Section 4.3.1)

Nodes	*Protocols*		
	Varadharajan et al.s' protocol [4]	Lee et al.s' protocol [6]	Our protocol
M_i	*1440MM	480MM + 1Sym + 1H	276MM + 2H
CH_j	*1440MM	480MM + 1Sym + 1H	276MM + 2H

*(1) We use an ElGamal encryption to estimate one PKO operation, (2) MM: a 1024-bit modular multiplication operation.

For other cases, the cost of two nodes building a session key can be compared by only calculating the cost of symmetric key encryption/decryption since each node has shared a session key with his clusterhead. We list the comparison results in Table 3 and Table 4.

Table2: The cost comparison of the session key building for two nodes within one hop in the same cluster (case (b) in Section 4.3.1)

Nodes	*Protocols*		
	Varadharajan et al.s' protocol [4]	Lee et al.s' protocol [6]	Our protocol
M_A	X	480MM + 1Sym + 1H	276MM + 2H
M_B	X	480MM + 1Sym + 1H	276MM + 2H

X: [4] lacks related schemes.

From Table 1 to Table 3, we can see that our scheme is the most efficient when compared with schemes [4] and [6] in the session key building for the three cases, (a), (b) and (c) in Section 4.3.1 respectively. From Table 4, our scheme has the same performance with scheme [6] but is more efficient than scheme [4] in the session key building for two nodes in different clusters.

Table3: The cost comparison of the session key building for two nodes via the clusterhead in the same cluster (case (c) in Section 4.3.1)

Nodes	*Protocols*		
	Varadharajan et al.s' protocol [4]	Lee et al.s' protocol [6]	Our protocol
$\boldsymbol{M_A}$	8Sym + 1H	2Sym + 1H	1Sym + 1H
$\boldsymbol{CH_j}$	8Sym	4Sym	2Sym
$\boldsymbol{M_B}$	8Sym + 1H	2Sym + 1H	1Sym + 1H

Table4: The cost comparisons of the session key building via two clusterheads for two nodes in different clusters (in Section 4.3.3)

Nodes	*Protocols*		
	Varadharajan et al.s' protocol [4]	Lee et al.s' protocol [6]	Our protocol
$\boldsymbol{M_A}$	8Sym + 1H	2Sym + 1H	2Sym + 1H
$\boldsymbol{CH_1}$	8Sym	4Sym	4Sym
$\boldsymbol{CH_2}$	4Sym	4Sym	4Sym
$\boldsymbol{M_B}$	4Sym + 1H	2Sym + 1H	2Sym + 1H

For the security comparison as shown in Table 5, schemes [4] and [6] do not consider the mechanism of group key rekeying when a node enters or leaves the group and thus they do not have the properties of group key forward and backward secrecy. Moreover, scheme [6] suffers the vulnerability of weak session key that we have mentioned in Section 3.

Our scheme can avoid this vulnerability for it does not transmit the hash value of the session key over the net. Therefore, our protocol is more secure than scheme [4] and [6].

Table5: Comparisons of security attributes

Security attributes	*Protocols*		
	Varadharajan et al.s' protocol [4]	Lee et al.s' protocol [6]	Our protocol
Mutual authentication	Yes	Yes	Yes
MIMA resistance	Yes	Yes	Yes
Group key forward secrecy	No	No	Yes
Group key backward secrecy	No	No	Yes
Weak session key resistance	Yes	No	Yes

6.2 Comparisons of protocols for smart card systems

In this section, we make performance comparisons of our scheme with protocols [22] and [14] as shown in table 6. The definitions of used notations are described as follows.

H: denotes the operation of one way hash function,

XOR: denotes the operation of exclusive OR,

BP: denotes the operation of bilinear pairings,

PM: denote the point multiplication,

PSA: denote the parallel session attack,

SM: denote the scalar multiplication computation computed by the smart card.

The r_nQ_n and Msg_1 computed by smart card in step 2 and Msg_2 computed by *RS* in step 4 all can be pre-computed. In our analysis, we ignore the one which can be pre-computed. After analyzing step 3, 4 for

RS to authenticate U_n, we know that it needs 2BP + 2PM + 1SM + 1H. After analyzing step 6 for U_n to authenticate *RS*, we know that it needs only 1BP + 1PM + 1H computations. Therefore, our scheme needs 3BP + 3PM + 1SM + 2H in the authentication phase. Note that the client end in our scheme needs only 1BP + 1PM + 1H. It does not cost too much compared to the other two schemes. But its security is pointed greatly.

Table 6: performance and security comparisons of our scheme with protocols [13] and [14]

	Protocols		
Computation cost	Ku et al.s' protocol [22]	Wang et al.s' protocol [14]	Our ID-based bilinear pairings protocol
Login	2H + 2XOR	4H + 2XOR	0
Authentication	4H + 3XOR	4H + 5XOR	3BP + 3PM + 1SM + 1H
Total	6H + 5XOR	8H + 7XOR	3BP + 3PM + 1SM + 2H
Resist PSA	NO	NO	YES
Resist KCI attack	NO	NO	YES
Basis of security	Hash functions	Hash functions	Bilinear pairings

7. Conclusions

The architecture of NTDR network is especially suitable for an ad hoc network in a large communication area due to the necessity of nodes transmitting message through the local clusterhead when they are beyond one-hop apart. And each clusterhead can monitor the communication messages to ensure the system's safety. It can greatly reduce the power consumption due to its cluster-based hierarchy. But, till now, there does not exist a secure and efficient protocol which can really satisfy all the security requirements for such a network. In this paper, we propose a novel two-level architecture for the session key generation by using ID-based bilinear pairings. We have described and proved the correctness of our protocol. Up to now, this is the first scheme which can really be implemented securely and efficiently. Moreover, we also propose a novel password authentication scheme based on ID-based bilinear pairing and give its security analysis and comparisons. From the security analysis, we can conclude that our protocol is really secure and from the comparisons, we can see that our scheme outperforms the other two in security strength only with a limited additional computation overhead for the client end.

References

1. C. E. Perkins, E. M. Royer, Ad hoc on-demand distance vector routing, in Proc. WMCSA, New Orleans, LA, Feb. 1999, pp. 90-100.

2. D. Johnson, D. Maltz, Y.-C. Hu, The dynamic source routing protocol for mobile ad hoc networks (DSR), IEEE Internet Draft, Apr. 2003.

3. K. Sanzgiri, D. LaFlamme, B. Dahill, B. N. Levine, C. Shields, E.M. Belding-Royer, Authenticated routing for ad hoc networks Selected Areas in Communications, IEEE Journal on Vol. 23, Issue 3, March 2005, pp. 598-610.

4. V. Varadharajan, R. Shankaran, M. Hitchens, Security for cluster based ad hoc networks, Computer Communications, Vol. 27, Issue 5, 20 March 2004, pp. 88-501.

5. C. C. Chang, K. C. Lin, J. S. Lee, DH-based communication method for cluster-based ad hoc networks, in Proceedings of the 2nd International Conference of Mobile Technology, Applications and Systems, 15-17 Nov. 2005, pp. 8-15.

6. J. S. Lee, C. C. Chang, Secure communications for cluster-based ad hoc networks using node identities, Journal of Network and Computer Applications, Vol. 30, Issue 4, November 2007, pp. 1377-1396.

7. H. Y. Chien, R. Y. Lin, Identity-based Key Agreement Protocol for Mobile Ad-hoc Networks Using Bilinear Pairing, in Proceedings of IEEE

International Conference of Sensor Networks, Ubiquitous, and Trustworthy Computing, Vol. 1, 05-07 June 2006, pp. 520-529.

8. F. Zhang, X. Chen, Attack on an ID-based authenticated group key agreement scheme from PKC 2004, Information Processing Letters, Vol. 91, Issue 4, 31 August 2004, pp. 191-193.

9. K. Shim, S. Woo, Weakness in ID-based one round authenticated tripartite multiple-key agreement protocol with pairings, Applied Mathematics and Computation, Vol. 166, Issue 3, 26 July 2005, pp. 523-530.

10. C. Castelluccia, N. Saxena, Jeong Hyun Yi, Robust self-keying mobile ad hoc networks, Computer Networks, Vol. 51, Issue 4, 14 March 2007, pp. 1169-1182

11. C. Popescu, A secure authenticated key agreement protocol, Electrotechnical Conference, MELECON 2004. Proceedings of the 12th IEEE Mediterranean Vol. 2, 12-15 May 2004, pp. 783-786.

12. Y. H. Lee, H. Kim, B. Chung, J. Lee, H. Yoon, On-demand secure routing protocol for ad hoc network using ID based cryptosystem, Parallel and Distributed Computing, Applications and Technologies, 2003, PDCAT′2003, Proceedings of the Fourth International Conference on 27-29 Aug. 2003, pp. 211-215.

13. A. Durresi, V. Bulusu, V. Paruchuri, L. Barolli, Secure emergency communication of cellular phones in ad hoc mode, Ad Hoc Networks, Vol. 5, Issue 1, January 2007, pp. 126-133.

14. H. Y. Chien, R. Y. Lin, Improved ID-based security framework for ad hoc network, Ad Hoc Networks, Vol. 6, Issue 1, January 2008, pp. 47-60.

15. S. Capkun, L. Buttyan, J.P. Hubaux, Self-organized public-key management for mobile ad hoc networks, Mobile Computing, IEEE Transactions on Vol. 2, Issue 1, Jan.-March 2003, pp. 52-64.

16. H. Deng, W. Li, D.P. Agrawal, Routing security in wireless ad hoc networks, Communications Magazine, IEEE Vol. 40, Issue 10, Oct. 2002,pp. 70-75.

17. R. Ruppe, S. Grisward, P. Walsh, R. Martin, Near term digital radio (NTDR) system, proceedings of the IEEE military communication conference, California, USA: 1997.

18. J. Chandra, L.L. Singh, A cluster based security model for mobile ad hoc networks, Personal Wireless Communications, 2005. ICPWC 2005. 2005 IEEE International Conference on 23-25 Jan. 2005, pp. 413-416.

19. S. H. Liaw, P. C. Su, H. K. C. Chang, E. H. Lu, S. F. Pon, Secured key exchange protocol in wireless mobile ad hoc networks, in proceedings of 39th Annual 2005 International Carnahan Conference, Security Technology,2005, CCST'05, on 11-14 Oct. 2005, pp. 171-173.

20. Y. Zhou, Y. Zhang, Y. Fang, Access control in wireless sensor networks, Ad Hoc Networks Vol. 5, Issue: 1, January, 2007, pp. 3-13.

21. A. Shamir, Identity based cryptosystems & signature schemes, Advances in Cryptology, CRYPTO’84, Lecture Notes-Computer Science, 1984, pp. 47-53.

22. G. Frey, H. Ruck, A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves, Mathematics of

Computation 62, 1994, pp. 865-874.

23. A. Menezes, T. Okamoto, S. Vanston, Reducing elliptic curve logarithms to logarithms in a finite field, IEEE Transaction on Information Theory 39, 1993, pp. 1639-1646.

24. A. Joux, A one round protocol for tripartite Diffie–Hellman, in Proceedings of Algorithmic Number Theory Symposium Lecture Notes in Computer Science, Vol. 1838, Springer-Verlag, Berlin, 2000, pp. 385-394.

25. D. Boneh, M. Franklin, Identity-based encryption from the Weil pairing, in: Advances in Cryptology—Crypto_01, LNCS 2139, Springer-Verlag, 2001, pp. 213-229.

26. N.P. Smart, An identity based authentication key agreement protocol based on pairing, Electron. Lett. 38, 2002, pp. 630-632.

27. K.G. Paterson, ID-based signature from pairings on elliptic curves, Electron. Lett. 38 (18), 2002, pp. 1025-1026.

28. R. Dupont, A. Enge, Provably secure non-interactive key distribution based on pairings, Discrete Applied Mathematics, Vol. 154, Issue 2, 1 February 2006, pp. 270-276.

29. N. Y. Lee, C. N. Wu, C. C. Wang, Authenticated multiple key exchange protocols based on elliptic curves and bilinear pairings, Computers & Electrical Engineering, Vol. 34, Issue 1, January 2008, pp. 12-20.

30. K. Y. Choi, J. Y. Hwang, D. H. Lee, I. S. Seo, ID-based Authenticated Key Agreement for Low-Power Mobile Devices, Conference on Information Security and Privacy-ACISP 2005, LNCS 3574, pp. 494-505.

31. N. Koblitz, AJ. Menezes, SA. Vanstone, The state of elliptic curve cryptography, Design Codes Crypto 2000,19:173–93.

32. P. S. L. M. Barreto, H. Y. Kim, B. Lynn, M. Scott, Efficient algorithms for pairing-based cryptosystems, Proc. of Crypto '02, LNCS 2442, 2002, pp. 354-368.

33. P. S. L. M. Barreto, B. Lynn, M. Scott, Efficient implementation of pairing-based cryptosystems, Journal of Cryptology, 2004, pp. 321-334.

34. S. D. Galbraith, K. Harrison, D. Soldera, Implementing the Tate pairing, Proc. of ANTS'02, LNCS 2369, Springer-Verlag, 2002, pp.324-337.

35. E. Thompson, MD5 collisions and the impact on computer forensics, Digital investigation 2005, pp. 36-40.

36. D. R. Stinson, Cryptography Theory and Practice, 3rd, 2006, p.177.

37. W.H. Yang, S.P. Shieh, Password authentication schemes with smart cards, Computers & Security, 18 (8), 1999, pp.727-733.

38. C. K. Chan, L. M. Cheng, Cryptanalysis of a timestamp based password authentication scheme, Computers &Security, 21(1), 2002, pp.74-76.

39. C.C. Lee, M.S. Hwang, W.P. Yang, A flexible remote user authentication scheme using smart cards, ACM Operating Systems Review 36 (3), 2002, pp.46-52.

40. H.M. Sun, H.T. Yeh, Further cryptanalysis of a password authentication scheme with smart cards, IEICE Transactions and Communications E86-B(4), 2003, pp.1412-1415.

41. K.F. Chen, S. Zhong, Attacks on the Yang-Shieh authentication, Computers & Security, 22(8), 2003, pp.725-727.

42. J.J. Shen, C.W. Lin, Security enhancement for the timestamp based password authentication scheme using smart cards, Computers& Security, 22 (7), 2003, pp.591-595.

43. S.T. Wu, B.C. Chieu, A user friendly remote authentication scheme with smart cards, Computers & Security,22 (6), 2003, pp.547-550.

44. W.S. Juang, Efficient password authenticated key agreement using smart card, Computers & Security (23), 2004, pp.167-173.

45. W.J. Tsaur, C.C. Wu, A smart card-based remote scheme for password authentication in multi-server Internet services, Computer Standards & Interfaces (27), 2004, pp.39–51.

46. R. Jiang, Li. Pan, Further analysis of password authentication schemes-based on authentication tests, Computers & Security (23), 2004, pp. 469-477.

47. C.C. Yang, R.C. Wang, An improvement of the Yang-Shieh password authentication schemes, Applied Mathematics and Computation, Vol. 162, issue:3, 2005, pp.1391-1396.

48. M.L. Das, A. Saxena, V.P.A. Gulati, A Dynamic ID-based remote user authentication scheme, Consumer Electronics, IEEE Transactions on Vol. 50, Issue 2, May 2004 Page(s):629 – 631.

49. M. Misbahuddin, M.A. Ahmed, A.A. Rao, C.S. Bindu, M.A.M. Khan, A Novel Dynamic ID-Based Remote User Authentication Scheme, Annual

India Conference, 2006 Sept. 2006 Page(s):1 - 5

50. X.M. Wang, W.F. Zhang, J.S. Zhang, M.K. Khan, Cryptanalysis and improvement on two efficient remote user authentication scheme using smart cards, Computer Standards & Interfaces, Volume 29, Issue 5, July 2007, pp. 507-512

51. W.C. Ku, S.M. Chen, Weaknesses and improvements of an efficient password based remote user authentication scheme using smart cards, Consumer Electronics, IEEE Transactions on Volume 50, Issue 1, Feb 2004 pp. 204 -207.

52. T.F. Cheng, J.S. Lee, C.C. Chang, Security enhancement of an IC-card-based remote login mechanism, Computer Networks, Volume 51, Issue 9, 20 June 2007, pp. 2280-2287

53. K.W. Kim, J.C. Jeon, K.Y. Yoo, An improvement on Yang et al.'s password authentication schemes, Applied Mathematics and Computation 170, 2005 pp. 207–215.

54. M. L. Das, A. Saxena, A novel remote user authentication scheme using bilinear pairings, Computers & Security, Volume 25, Issue 3, May 2006, pp. 184-189

55. J.S. Chou, Y. Chen, J.Y. Lin, Improvement of Manik et al.'s remote user authentication scheme, http://eprint.iacr.org/2005/450.pdf.

56. C.T. Wang, C.C. Chang, C.H. Lin, Using IC cards to remotely login passwords without verification tables, in: Proceedings of the 18th International Conference on Advanced Information. Networking and

Applications, Fukuoka, Japan, vol. 1, 2004, pp. 321 – 326.

57. D.S. Wong, A.H. Chan, Efficient and mutually authenticated key exchange for low power computing devices, Advances in Cryptology—ASIACRYPT'2001, 2001 – Springer pp. 1-18.