Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.12785/amis/070414

A novel remote user authentication and key agreement scheme for mobile client-server environment

*Haiyan Sun**, *Qiaoyan Wen*, *Hua Zhang and Zhengping Jin*

State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China

Received: 27 Jun. 2012, Revised: 26 Aug. 2012, Accepted: 20 Sep. 2012
Published online: 1 Jul. 2013

Abstract: Recently, many identity (ID)-based user authentication and key agreement schemes for mobile client-server environment were proposed. However, these schemes are subjected to an inherent design weakness, namely, the server knows all users' private keys. Under this problem, these schemes cannot provide insider attack resistance or mutual authentication. Furthermore, some of these schemes cannot simultaneously provide user anonymity, perfect forward secrecy, or leakage of session temporary secrets resistance. In this paper, we propose a strongly secure remote user authentication and key agreement scheme to solve these security weaknesses. Security proof shows that the proposed scheme can achieve mutual authentication and key agreement, and provide perfect forward secrecy. Further security analysis shows that the proposed scheme can provide user anonymity, insider attack resistance and leakage of session temporary secrets resistance. In addition, the proposed scheme possesses low computation cost and low power consumption. Thus the proposed scheme is more suitable for mobile client-server environment.

Keywords: Mutual authentication; Key agreement; Insider attack resistance; Anonymity; Provably security

1. Introduction

Remote user authentication allows a remote user and a server authenticate the identity with each other over insecure networks. Mobile devices (e.g., smart phones) are widely and popularly used in many electronic transactions, such as online shopping, Internet banking, e-payment, e-voting and pay-TV. Considering the limited energy resources and computing ability of mobile devices, it is inappropriate for remote user authentication schemes to be realized in traditional public key cryptography since most cryptographic algorithms require many expensive computations and it suffers from heavy certificate burden.

To avoid the above problems, identity (ID)-based public key cryptography (IB-PKC) was introduced by Shamir [2]. Compared with traditional public key cryptography, IB-PKC eliminates certificate management burden. Since then, many ID-based remote user authentication schemes [3,4,5,6,7,8,9,10,11,12] were proposed. These identity-based schemes can be divided into two types, namely authentication schemes from pairings and authentication schemes without pairings. The former authentication schemes from pairings are described below. In 2006, Das et al. [3] proposed a pairing-based remote client authentication scheme with smart cards. However, this scheme suffered from a forgery attack. To overcome this attack, an improved scheme [4] was presented, however, it suffered from a computation burden. In 2008, Tseng et al. [5] presented a provably secure and efficient user authentication scheme for wireless clients with smart cards. However, these schemes [3,4,5] cannot provide mutual authentication and key agreement between the mobile client and the server. In 2010, Wu and Tseng [6] presented a user authentication and key exchange scheme and also claimed their scheme was provably secure in the random oracle model. However, their scheme is not efficient. To further improve efficiency, He [7] proposed a novel user authentication and key exchange protocol from pairings in 2012.

Authentication schemes without pairings also have been proposed successively. In 2009, Yang and Chang [8] proposed an ID-based remote mutual authentication with key agreement scheme without pairings. Later, Yoon and Yoo [9] pointed out that Yang and Chang's scheme suffered from an impersonation attack and did not provide

* Corresponding author e-mail: wenzhong2520@gmail.com

perfect forward secrecy, and also proposed an improved scheme. In 2011, Islam and Biswas [10] found these two schemes [8,9] still suffered from replay attack and clock synchronization problem, and cannot provide user anonymity, perfect forward secrecy, or leakage of session temporary secrets resistance. To solve these weaknesses, they also proposed an improved scheme. Later, Truong et al. [11] pointed that this scheme still cannot provide leakage of session temporary secrets resistance. However, these schemes [8,9,10,11] were inefficient due to a special hash function called MapToPoint function. To improve efficiency, He et al. [12] proposed an efficient user authentication and key exchange scheme with provable security in 2012.

In this paper, we find that the above ID-based remote user authentication and key agreement schemes [6,7,8,9,10,11,12] are subjected to an inherent design weakness, i.e., the server knows all users' private keys, and under this problem these schemes cannot provide insider attack resistance or mutual authentication. Furthermore, we find that some of these schemes cannot provide user anonymity [6,7,12], perfect forward secrecy [6,7], or leakage of session temporary secrets resistance [6,7,12]. To improve security, we propose a novel remote user authentication and key agreement scheme. Security analysis shows that the proposed scheme does not suffer from these security weaknesses. We also provide a formal security proof for the proposed scheme's basic security properties including mutual authentication and key agreement. Compared with several latest schemes, the proposed scheme is more secure, practical and suitable for mobile client-server environment.

The rest of this paper is organized as follows. Some preliminaries are given in Section 2. A brief cryptanalysis of some schemes is provided in Section 3. Our scheme is proposed in Section 4, its security is proved in Section 5, and a comparison with some schemes is given in Section 6. Finally, some conclusions are drawn in Section 7.

2. Preliminaries

2.1. Notations

Some important notations to be used in this paper are described as follows.

- q: a large prime number;
- G: a cyclic additive group of order q;
- P: the generator of G;
- Z_q^*: $\{1,2,\cdots,q-1\}$;
- ID_C: the identity of the client C;
- CID_C: the dynamic identity of the client C;
- (s_C, x_C): the private key of the client C;
- (PK_C, X_C): the public key of the client C;
- s: the private key of the server S;
- P_{pub}: the system public key;
- H_1, H_2, H_3, H_4: collision-free one-way hash functions;
- $[]_x$: the x-coordinate.

2.2. Complexity assumptions

Let G be a cyclic additive group generated by point P, whose order is a prime q. We review the following well-known problems to be used in the security analysis of our scheme.

Elliptic Curve Discrete Logarithm (ECDL) Problem: Given two group elements P and Q, find an integer $a \in Z_q^*$, such that $Q = aP$.

Computational Diffie-Hellman (CDH) Problem: For $a, b \in Z_q^*$, given P, aP, bP, compute abP.

Up to now, there is no efficient algorithm to be able to solve any of the above problems [13].

2.3. Forking lemma

Let $\mathscr{F}$ be a probabilistic polynomial time Turing machine whose input only consists of public data. If $\mathscr{F}$ can produce a valid signature $(m, \sigma_1, h, \sigma_2)$ with non-negligible probability, then a replay of this machine, with the same random tape and a different oracle, outputs two valid signatures $(m, \sigma_1, h, \sigma_2)$ and $(m, \sigma_1, h', \sigma_2')$ such that $h \neq h'$ with non-negligible probability.

For the details of this lemma, please refer to [14].

3. Weaknesses of some ID-based remote user authentication and key agreement schemes

Recently, many remote user authentication and key agreement schemes [6,7,8,9,10,11,12] were presented. These schemes generally consist of three phases, i.e., setup phase, client registration phase, and user authentication and key agreement phase. The detail description of these schemes can refer to [6,7,8,9,10,11,12]. In this section we analyze these schemes' security weaknesses from the common security weaknesses, user anonymity, and security of session keys.

3.1. Common security weaknesses

Observing the *Client registration phase* of these schemes [6,7,8,9,10,11,12], the server can know all clients' private keys since all private keys only depend on the master key s of the server. Thus these schemes [6,7,8,9,10,11,12] are subjected to an inherent design weakness, i.e., the server knows all users' private keys. In the following, we show that under this problem these schemes cannot provide insider attack resistance or mutual authentication.

3.1.1. Insider attack

Insider attack is that the client's private key is derived by the privileged insider of the server. Observing the *Client registration phase* of these schemes [6,7,8,9,10,11,12], the privileged insider of the server who usually owns the master key s of the sever can easily compute the client's private key. Hence, these schemes fail to resist insider attack.

3.1.2. Failure to provide mutual authentication

From the analysis in Section 3.1.1, a privileged insider of the server can know all clients' private keys. Once the attacker derives the client C's private key, he can impersonate C to access the server just selecting a random number $r_C \in Z_q^*$ and performing the following steps of these schemes in *user authentication and key agreement phase*. Therefore, a privileged insider of the server can impersonate any client to access the server, that is, these schemes cannot provide the client-to-server authentication. Hence, these schemes fail to provide mutual authentication.

3.2. User anonymity

For schemes [6,7,12], the identity ID_C of the client C is transferred by plaintext, so an attacker can easily obtain the identity of the client. Thus user anonymity cannot be protected.

3.3. Security of the session key

There are two important security properties about the session key, including perfect forward secrecy and leakage of session temporary secrets resistance.

3.3.1. Perfect forward secrecy

A protocol can provide perfect forward secrecy, if private keys of the server and the client are compromised, the secrecy of previous session keys is not affected.

The session key of scheme [7] is $sk = H_4(P_{pub}, ID_C, \alpha, U, R_2)$ where P_{pub} is the system public key and $R_2 = sU = r_C P_{pub}$. Assume that an adversary collects all previous messages $M_1 = \{ID_C, U\}$ and $M_2 = \{\alpha, Auth\}$. Now if the adversary obtains the master key s of the server S, the adversary can compute $R_2 = sU$. Therefore previous session key sk of scheme [7] can be compromised to the adversary.

The analysis of [6] is similar to that of [7]. Thus schemes [6,7] cannot provide perfect forward secrecy.

3.3.2. Leakage of session temporary secrets resistance

A protocol can provide leakage of session temporary secrets resistance, if session ephemeral secrets of the server and the client are leaked, the secrecy of the session key is not affected.

Assume that an adversary have intercepted messages $M_1 = \{ID_C, U\}$ and $M_2 = \{\alpha, Auth\}$ of scheme [7]. Now if the adversary obtains the random number r_C of the client C, the adversary can compute $R_2 = r_C P_{pub}$. Therefore the session key of scheme [7] can be compromised to the adversary.

The analysis of scheme [6] is similar to that of scheme [7].

The session key of scheme [12] is $sk = H_3(ID_C, T_C, T_S, M, W, K)$ where $M = r_C P, W = r_S P, K = r_C r_S P$. Assume that an adversary have intercepted messages $\{ID_C, T_C, M\}$ and $\{ID_C, T_S, W\}$. Now if the adversary obtains the random number r_C of the client C, the adversary can compute $K = r_C W$. Therefore the session key of scheme [12] can be compromised to the adversary.

Thus schemes [6,7,12] cannot provide leakage of session temporary secrets resistance.

4. Our proposed scheme

Our proposed scheme consists of the following three phases including system setup phase, client registration phase, and user authentication and key agreement phase.

4.1. System setup phase

Given a security parameter k, the server S generates the system parameters as follows.

(1) Choose a finite field F_p, where p is a k-bit prime.
(2) Define an elliptic curve $E : y^2 \equiv x^3 + ax + b \bmod p$ over F_p, where $a, b \in F_p, p \geq 3, 4a^3 + 27b^2 \neq 0 \bmod p$.
(3) Choose a public point P with prime order q over E, and generate a cyclic additive group G of order q by point P.
(4) Choose a random number $s \in Z_q^*$ as the master key and set $P_{pub} = sP$ as the system public key.
(5) Choose four cryptographic hash functions $H_1 : \{0,1\}^* \times G \to Z_q^*$, $H_2 : \{0,1\}^* \times G^3 \to \{0,1\}^k$, $H_3 : \{0,1\}^* \times G^4 \to \{0,1\}^k$ and $H_4 : \{0,1\}^* \times G^5 \to \{0,1\}^k$.
(6) Publish the system parameters $params = (F_q, E, G, P, P_{pub}, H_1, H_2, H_3, H_4)$ while keeping s secret.

4.2. Client registration phase

When a low-power computing client C with identity ID_C wants to register to the server S, S generates the private key of C. C and S do as follows.

(1) The client C chooses a secret random number $x_C \in Z_q^*$, computes $X_C = x_C P$, and then sends (ID_C, X_C) to the server S over secure channels.
(2) Once receiving (ID_C, X_C), the server S selects a random number $y_C \in Z_q^*$, computes $W_C = X_C + y_C P$ and $d_C = (H_1(ID_C, W_C)s - y_C) \bmod q$, and then sends (W_C, d_C) to the client C over a secure channel.
(3) The client C computes $s_C = (d_C - x_C) \bmod q$ and $PK_C = s_C P$. Finally, C sets (s_C, x_C) and (PK_C, X_C) as his private key and public key, respectively. Everyone who receives W_C can compute $PK_C = H_1(ID_C, W_C)P_{pub} - W_C$.

PK_C is correct since

$$\begin{aligned} PK_C &= s_C P = (d_C - x_C)P \\ &= (H_1(ID_C, W_C)s - y_C - x_C)P \\ &= H_1(ID_C, W_C)sP - (y_C P + x_C P) \\ &= H_1(ID_C, W_C)P_{pub} - W_C. \end{aligned}$$

4.3. User authentication and key agreement phase

In this phase, the low-power client C communicates with the server S. The phase is depicted in Figure 1. The detail is illustrated as follows.

(1) The client C chooses random numbers $r_C, z_C \in Z_q^*$, and computes $R_C = r_C P$, $k_1 = r_C P_{pub}$, $CID_C = ID_C \oplus [k_1]_x$, $Z_C = z_C P$, $h = H_2(ID_C, Z_C, P_{pub}, W_C, X_C)$, and $v = (z_C - hs_C) \bmod q$. Then C sends $M_1 = (CID_C, R_C, W_C, X_C, h, v)$ to the server S.
(2) Upon receiving M_1, the server S computes $k_2 = sR_C$. Then S extracts the client C's identity by doing $ID_C = CID_C \oplus [k_2]_x$, and checks the validity of ID_C. If ID_C is valid, then S continues to go next step; otherwise S rejects C's login request.
(3) Next, the server S computes $PK_C = H_1(ID_C, W_C)P_{pub} - W_C$, $Z'_C = vP + hPK_C$ and $h' = H_2(ID_C, Z'_C, P_{pub}, W_C, X_C)$, and then verifies whether h' is equal to h. If it does not hold, then the server S rejects C's login request; otherwise, S randomly chooses $r_S \in Z_q^*$, and computes $R_S = r_S P_{pub}$, $Auth = H_3(ID_C, R_S, Z'_C, P_{pub}, k_2)$, $k_3 = sr_S(R_C + PK_C - X_C)$ and $sk = H_4(ID_C, R_S, R_C, W_C, P_{pub}, k_3)$. Finally, S sends $M_2 = (Auth, R_S)$ to the client C.
(4) Upon receiving M_2, the client C verifies whether $Auth$ is equal to $H_3(ID_C, R_S, Z_C, P_{pub}, k_1)$. If it holds, then C computes $k_4 = (r_C + s_C - x_C)R_S$ and $sk = H_4(ID_C, R_S, R_C, W_C, P_{pub}, k_4)$.

The scheme is correct because $k_1 = r_C P_{pub} = r_C sP = sr_C P = sR_C = k_2$, $Z'_C = vP + hPK_C = (v + hs_C)P = z_C P = Z_C$, and $k_3 = sr_S(R_C + PK_C - X_C) = sr_S(r_C + s_C - x_C)P = (r_C + s_C - x_C)r_S sP = (r_C + s_C - x_C)R_S = k_4$. Thus the client C and the server S establish a common session key $sk = H_4(ID_C, R_S, R_C, W_C, P_{pub}, k_3) = H_4(ID_C, R_S, R_C, W_C, P_{pub}, k_4)$.

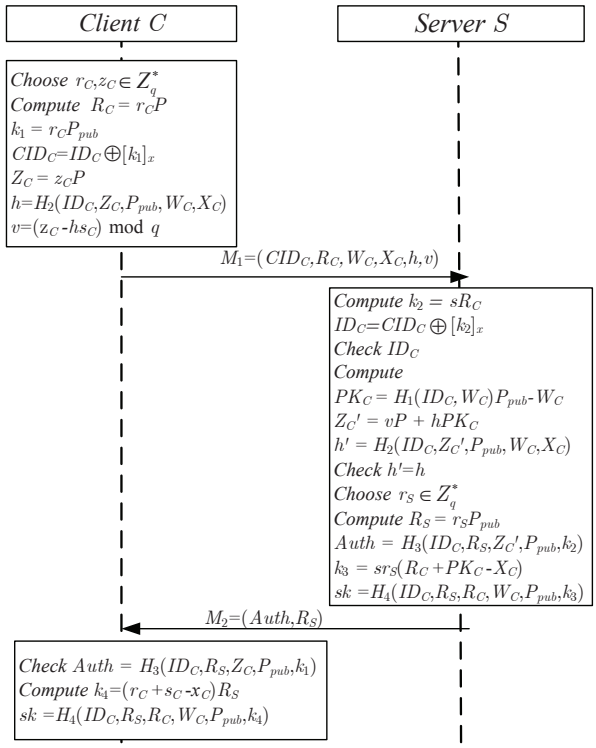

Figure 1: User authentication and key agreement phase

5. Security analysis

In this section, we discuss the security analysis of the proposed scheme. For a mutual authentication and key agreement protocol, mutual authentication and key agreement are basic security properties. We discuss the basic security properties in the security model (described later) and provide a full security proof. Then we show that the proposed scheme can provide other security properties including user anonymity, insider attack resistance, perfect forward secrecy and leakage of session temporary secrets resistance.

5.1. Security model

The model is defined by the following game which is run between a challenger $\mathscr{CH}$ and an adversary $\mathscr{A}$. $\mathscr{A}$ controls all communications from and to the protocol participants via accessing to a set of oracles as described below. Every participant involved in a session is treated as an oracle. We denote an instance i of the participant U as $\prod_U^i$, where $U \in \{C_1, \cdots, C_n\} \bigcup S$. Each client C has an identity ID_C. An adversary $\mathscr{A}$ is modeled by a probabilistic polynomial time Turing machine. All the communications go through the adversary $\mathscr{A}$. Participants only respond to the following oracles by $\mathscr{A}$

and do not communication directly among themselves. $\mathscr{A}$ can replay, delete, modify, interleave or delete all messages follows in the system.

- *Create(C)*: In this query, every client C with identity ID_C is created. Specially, C's private key (s_C, x_C) is established. All clients should be created by this query. Thus $\mathscr{A}$ knows the identities of all clients.
- *Corrupt(C)*: This query outputs the private key (s_C, x_C) of the client C.
- *Send*$(\prod_U^i, M)$: $\mathscr{A}$ sends a message M to the oracle $\prod_U^i$, and obtains the replay generated by this oracle.
- $H_j(m)$: When $\mathscr{A}$ makes this hash query with the message m, the oracle $\prod_U^i$ returns a random number r and records (m,r) into an initially empty list $L_{H_j}(j = 1,2,3,4)$.
- *Reveal*$(\prod_U^i)$: This query outputs the session key sk of the oracle $\prod_U^i$ if the oracle has accepted; otherwise, it returns *null* to the adversary.
- *Test*$(\prod_U^i)$: $\mathscr{A}$ can send a single *Test* query to $\prod_U^i$. Upon receiving this query, the oracle $\prod_U^i$ chooses a bit $b \in \{0,1\}$ uniformly at random. If $b = 1$, then it returns the session key. Otherwise, it returns a random string. This query measures the semantic security of the session key.

For a mutual authentication and key agreement scheme, we can divide it into three parts: client to server authentication (i.e., the server verifies whether the client is legal), server to client authentication (i.e., the client verifies whether the server is legal), and the session key agreement (i.e., both the client and the server agree a common session key). For authentication, the adversary can make Corrupt, $H_j(j = 1,2,3,4)$, and Send queries. For a key agreement, the adversary can make Corrupt, $H_j(j = 1,2,3,4)$, Send, Reveal and Test queries.

In an execution of our protocol P, we say adversary $\mathscr{A}$ violates client-to-server authentication if $\mathscr{A}$ can fake the authenticator "h, v". We denote this event and its probability by $C2S$ and $\Pr[C2S]$ respectively. We say protocol P can provide client to server authentication if $\Pr[C2S]$ is negligible. Similarly, we say adversary $\mathscr{A}$ violates server to client authentication if $\mathscr{A}$ can fake the authenticator "*Auth*". We denote this event and its probability by $S2C$ and $\Pr[S2C]$ respectively. We say protocol P can provide server-to-client authentication if $\Pr[S2C]$ is negligible. We say adversary $\mathscr{A}$ violates key agreement, if $\mathscr{A}$ sends a single *Test* query to an oracle $\prod_U^i$ ($\mathscr{A}$ can ask this oracle a Corrupt query after Test query), and correctly guesses the value b. We denote $\mathscr{A}$'s advantage by $Adv^{KA}(\mathscr{A})$, which is defined as $|Pr[\mathscr{A}\ wins] - 1/2|$ [15]. We say protocol P can provide key agreement if $Adv^{KA}(\mathscr{A})$ is negligible.

5.2. Basic security properties

In this subsection, we show that our scheme can achieve client-to-server authentication, server-to-client authentication and key agreement in the random oracle model.

Theorem 1. *Let $C2S$ and $Pr[C2S]$ denote the event that an adversary $\mathscr{A}$ violates client-to-server authentication and its probability respectively. Then $Pr[C2S]$ is negligible and our scheme can provide client to server authentication.*

Proof. Event $C2S$ means that there is some oracle $\prod_S^j$, which accepts but has no legal partner $\prod_{C_i}^m$. In the following, we will show that if an adversary $\mathscr{A}$ can violate client-to-server authentication with a non-negligible advantage ε, then we can use $\mathscr{A}$ to construct a challenger $\mathscr{CH}$ to solve the ECDL problem with a non-negligible advantage.

Given an ECDL instance $\{P, Q = aP\}$, $\mathscr{CH}$ simulates the system setup algorithm to generate system parameters $params = (F_q, E, G, P, P_{pub}, H_1, H_2, H_3, H_4)$ where $P_{pub} = sP$ and s is the master key. $\mathscr{CH}$ gives s and $params$ to $\mathscr{A}$. Four hash functions are simulated as random oracles controlled by $\mathscr{CH}$. Let q_C, q_S and q_{H_i} be the number of Create query, Send query and H_i query, where $i = 1,2,3,4$, respectively. $\mathscr{CH}$ picks $I \in \{1,2,...,q_C\}$ and $J \in \{1,2,...,q_S\}$, guesses $\mathscr{A}$ can violate client-to-server authentication against the oracle $\prod_S^J$ and the client with identity ID_I, and interacts with $\mathscr{A}$ as follows.

- H_1-*Query*: $\mathscr{CH}$ maintains a list L_{H_1} of tuples (ID_i, W_i, h_1). When a query on (ID_i, W_i) is submitted, $\mathscr{CH}$ outputs h_1 if it has appeared on L_{H_1}; otherwise, $\mathscr{CH}$ picks $h_1 \in Z_q^*$, adds (ID_i, W_i, h_1) to L_{H_1} and outputs h_1.
- *Create-Query*: $\mathscr{CH}$ maintains a list L_C of tuples $(ID_i, x_i, y_i, d_i, X_i, W_i, s_i)$. On receiving this query, $\mathscr{CH}$ does as follows:
  (1) If $ID_i = ID_I$, set $W_i = Q$, choose $y_i, h_1 \in Z_q^*$ and compute $d_i = sh_1 - y_i$ and $X_i = W_i - y_iP$. Tuples (ID_i, W_i, h_1) and $(ID_i, null, y_i, d_i, X_i, W_i, null)$ are added to L_{H_1} and L_C respectively.
  (2) Otherwise, randomly choose $x_i, y_i, h_1 \in Z_q^*$, compute $X_i = x_iP, W_i = X_i + y_iP, d_i = sh_1 - y_i, s_i = d_i - x_i$, then add (ID_i, W_i, h_1) and $(ID_i, x_i, y_i, d_i, X_i, W_i, s_i)$ to L_{H_1} and L_C respectively.
  Without loss of generality, we assume that, before asking the following queries, $\mathscr{A}$ has already asked some Create queries on related clients.
- H_2-*Query*: $\mathscr{CH}$ maintains a list L_{H_2} of tuples $(ID_i, Z_i, P_{pub}, W_i, X_i, h_2)$. When a query on $(ID_i, Z_i, P_{pub}, W_i, X_i)$ is submitted, $\mathscr{CH}$ outputs h_2 if it has appeared on L_{H_2}; otherwise, $\mathscr{CH}$ picks $h_2 \in Z_q^*$ at random, adds $(ID_i, Z_i, P_{pub}, W_i, X_i, h_2)$ to L_{H_2} and outputs h_2.
- H_3-*Query*: $\mathscr{CH}$ maintains a list L_{H_3} of tuples $(ID_i, R_S, Z_i, P_{pub}, k_2, h_3)$. When a query on $(ID_i, R_S, Z_i, P_{pub}, k_2)$ is submitted, $\mathscr{CH}$ outputs h_3 if it has appeared on L_{H_3}; otherwise, $\mathscr{CH}$ picks $h_3 \in Z_q^*$ at random, adds $(ID_i, R_S, Z_i, P_{pub}, k_2, h_3)$ to L_{H_3} and outputs h_3.
- H_4-*Query*: $\mathscr{CH}$ maintains a list L_{H_4} of tuples $(ID_i, R_S, R_i, W_i, P_{pub}, k_3, h_4)$. When a query on

$(ID_i, R_S, R_i, W_i, P_{pub}, k_3)$ is submitted, $\mathscr{CH}$ outputs h_4 if it has appeared on L_{H_4}; otherwise, $\mathscr{CH}$ picks $h_4 \in Z_q^*$ at random, adds $(ID_i, R_S, R_i, W_i, P_{pub}, k_3, h_4)$ to L_{H_4} and outputs h_4.

- *Corrupt-Query*: If $ID_i = ID_I$, it cancels the game. Otherwise, it returns (s_i, x_i).
- *Send-Query*:
  (1) When $\mathscr{A}$ makes a $Send(\prod_{C_i}^m, \text{“}Start\text{”})$ query, $\mathscr{CH}$ responds as follows. If $ID_i = ID_I$, $\mathscr{CH}$ first obtains $W_i = Q$ and X_i from L_C, then obtains $h_1$from L_{H_1}. Next, $\mathscr{CH}$ randomly chooses $r_i, v, h_2 \in Z_q^*$, and computes $Z_i = vP + h_2(h_1 P_{pub} - W_i)$, $R_i = r_i P$, $k_1 = sR_i$, and $CID_i = ID_i \oplus [k_1]_x$. Then $\mathscr{CH}$ sets $H_2(ID_i, Z_i, P_{pub}, W_i, X_i) = h_2$, and adds $(ID_i, Z_i, P_{pub}, W_i, X_i, h_2)$ to L_{H_2}. Finally, $\mathscr{CH}$ responds with $M_1 = (CID_i, R_i, W_i, X_i, h_2, v)$. If $ID_i \neq ID_I$, then $\mathscr{CH}$ responds with M_1 according to protocol description.
  (2) When $\mathscr{A}$ makes a $Send(\prod_S^j, M_1)$ query, $\mathscr{CH}$ responds with $M_2 = (Auth, R_S)$ according to protocol description.
  (3) When $\mathscr{A}$ makes a $Send(\prod_{C_i}^m, M_2)$ query, $\mathscr{CH}$ verifies whether $Auth$ is equal to $H_3(ID_i, R_S, Z_i, P_{pub}, k_1)$. If it holds, then $\prod_{C_i}^m$ accepts and terminates. Otherwise, $\prod_{C_i}^m$ terminates without accepting.

If $\mathscr{A}$ violates client-to-server authentication, it means that $\mathscr{A}$ has made a valid forgery (W_I, v, h_2) on ID_I. By the Forking lemma, $\mathscr{CH}$ can make two valid signatures (ID_I, W_I, Z_I, h_2, v) and $(ID_I, W_I, Z_I, h_2^*, v^*)$, where $h_2 \neq h_2^*$ and $W_I = Q$. Since $Z_I = vP + h_2(h_1 sP - W_I)$ and $Z_I = v^*P + h_2^*(h_1 sP - W_I)$, $\mathscr{CH}$ can compute $a = h_I s - \frac{v - v^*}{h_2^* - h_2}$. Thus $\mathscr{CH}$ can solve the ECDL problem.

$\mathscr{CH}$ succeeds if and only if $\mathscr{CH}$ does not abort in the simulation and $\mathscr{A}$ succeeds. If $\mathscr{A}$ indeed chooses $\prod_S^J$ and the client C_I as the challenge oracle which is correct with probability $\frac{1}{q_S q_C}$, then $\mathscr{CH}$ does not abort in the simulation. Thus $\mathscr{CH}$'s success probability is $\frac{\varepsilon}{q_S q_C}$. Therefore, if ε is non-negligible, then $\mathscr{CH}$'s success probability is also non-negligible. This contradicts the difficulty of the ECDL problem. Hence, our scheme can provide client to server authentication.

Theorem 2. *Let S2C and Pr[S2C] denote the event that an adversary $\mathscr{A}$ violates server-to-client authentication and its probability respectively. Then Pr[S2C] is negligible and our scheme can provide server to client authentication.*

Proof. Event *S2C* means that there is some oracle $\prod_{C_i}^m$, which accepts but has no legal partner $\prod_S^j$. In the following, we will show that if an adversary $\mathscr{A}$ can violate server-to-client authentication with a non-negligible advantage ε, then we can use $\mathscr{A}$ to construct a challenger $\mathscr{CH}$ to solve the CDH problem with a non-negligible advantage.

Given a CDH instance $\{P, Q_1 = aP, Q_2 = bP\}$, $\mathscr{CH}$ simulates the system setup algorithm to generate system parameters $params = (F_q, E, G, P, P_{pub}, H_1, H_2, H_3, H_4)$ where $P_{pub} = Q_1$. $\mathscr{CH}$ gives $params$ to $\mathscr{A}$. Four hash functions are simulated as random oracles controlled by $\mathscr{CH}$. Let q_C, q_S and q_{H_i} be the numbers of Create query, Send query and H_i query, where $i = 1, 2, 3, 4$, respectively. $\mathscr{CH}$ picks $I \in \{1, 2, ..., q_C\}$ and $J \in \{1, 2, ..., q_S\}$, guesses $\mathscr{A}$ can violate server-to-client authentication against the oracle $\prod_{C_I}^J$, and interacts with $\mathscr{A}$ as follows.

- *Hash-Query*: All hash queries are simulated the same as described in Theorem 1.
- *Create-Query*: $\mathscr{CH}$ maintains a list L_C of tuples $(ID_i, x_i, y_i, d_i, X_i, W_i, s_i)$. On receiving this query, $\mathscr{CH}$ chooses $x_i, h_1, d_i \in Z_q^*$ at random, computes $X_i = x_i P, Y_i = h_1 P_{pub} - d_i P, W_i = Y_i + x_i P, s_i = d_i - x_i$, then adds (ID_i, W_i, h_1) and $(ID_i, x_i, null, d_i, X_i, W_i, s_i)$ to L_{H_1} and L_C respectively.
- *Corrupt-Query*: It returns the private key (s_i, x_i) of C_i.
- *Send-Query*:
  (1) When $\mathscr{A}$ makes a $Send(\prod_{C_i}^m, \text{“}Start\text{”})$ query, $\mathscr{CH}$ responds as follows. If $\prod_{C_i}^m = \prod_{C_I}^J$, $\mathscr{CH}$ sets $R_i = Q_2$, chooses $z_i \in Z_q^*, k_1 \in G$, and computes $CID_i = ID_i \oplus [k_1]_x$, $Z_i = z_i P, h_2 = H_2(ID_i, Z_i, P_{pub}, W_i, X_i)$ and $v = z_i - h_2 s_i \bmod q$. Finally, $\mathscr{CH}$ responds with $M_1 = (CID_i, R_i, W_i, X_i, h_2, v)$. Otherwise, $\mathscr{CH}$ responds with M_1 according to protocol description.
  (2) When $\mathscr{A}$ makes a $Send(\prod_S^j, M_1)$ query, $\mathscr{CH}$ computes $ID_i = CID_i \oplus [k_1]_x$, $Z_i' = vP + h_2(H_1(ID_i, W_i)P_{pub} - W_i)$ and $h' = H_2(ID_i, Z_i', P_{pub}, W_i, X_i)$, and then verifies whether h' is equal to h. If it holds, $\mathscr{CH}$ randomly chooses $r_S \in Z_q^*$, computes $R_S = r_S P_{pub}, k_3 = r_S k_1 + r_S(s_C - x_C)P_{pub}, Auth = H_3(ID_i, R_S, Z_i', P_{pub}, k_1)$ and $sk = H_4(ID_i, R_S, R_i, W_i, P_{pub}, k_3)$, and responds with $M_2 = (Auth, R_S)$.
  (3) When $\mathscr{A}$ makes a $Send(\prod_{C_i}^m, M_2)$ query, $\mathscr{CH}$ verifies whether $Auth$ is equal to $H_3(ID_i, R_S, Z_i, P_{pub}, k_1)$. If it holds, then $\prod_{C_i}^m$ accepts and terminates. Otherwise, $\prod_{C_i}^m$ terminates without accepting.

If $\mathscr{A}$ violates server-to-client authentication, it means that $\mathscr{A}$ has made the corresponding H_3 query on $(ID_i, R_S, Z_i, P_{pub}, k_2)$. $\mathscr{CH}$ chooses a tuple $(ID_i, R_S, Z_i, P_{pub}, k_2, h_3)$ from L_{H_3}, and then outputs k_2 as the solution to the CDH problem.

$\mathscr{CH}$ succeeds if and only if $\mathscr{A}$ chooses $\prod_{C_I}^J$ as the challenge oracle and finds the corresponding item from L_{H_3} and $\mathscr{A}$ succeeds. The probability that $\mathscr{A}$ chooses $\prod_{C_I}^J$ as the challenge oracle is $\frac{1}{q_S q_C}$. Thus the success probability of $\mathscr{CH}$ is $\frac{\varepsilon}{q_S q_C q_{H_3}}$. Therefore, if ε is non-negligible, then the success probability of $\mathscr{CH}$ is also non-negligible. This contradicts the hardness of the

CDH problem. Hence, our scheme can provide server to client authentication.

Theorem 3. *Assume an adversary $\mathscr{A}$ can correctly guess the value b involved in the Test-query with a non-negligible advantage ε, then we can construct a challenger $\mathscr{CH}$ to solve the CDH problem with a non-negligible probability. Thus Our scheme can provide key agreement.*

Proof. Given a CDH instance $\{P, Q_1 = aP, Q_2 = bP\}$, $\mathscr{CH}$ simulates the system setup algorithm to generate parameters $params = (F_q, E, G, P, P_{pub}, H_1, H_2, H_3, H_4)$ where $P_{pub} = sP$ and s is the master key. $\mathscr{CH}$ gives $params$ and s to $\mathscr{A}$. All hash functions are simulated as random oracles controlled by $\mathscr{CH}$. Let q_C, q_S and q_{H_i} be number of the Create query, Send query and H_i query, where $i = 1,2,3,4$, respectively. $\mathscr{CH}$ picks $I \in \{1,2,...,q_C\}$ and $J, L \in \{1,2, ...,q_S\}$, guesses $\mathscr{A}$ will select the oracle $\prod_{C_I}^J$ to ask Test query after $\prod_S^L$ has had a matching conversation to $\prod_{C_I}^J$, and interacts with $\mathscr{A}$ as follows.

- *Hash-Query*: All hash queries are simulated the same as described in Theorem 1.
- *Create-Query*: $\mathscr{CH}$ maintains a list L_C of tuples $(ID_i, x_i, y_i, d_i, X_i, W_i, s_i)$. On receiving this query, $\mathscr{CH}$ chooses $x_i, y_i, h_1 \in Z_q^*$ at random, computes $X_i = x_i P, W_i = X_i + y_i P, d_i = sh_1 - y_i, s_i = d_i - x_i$, then adds $(ID_i, x_i, y_i, d_i, X_i, W_i, s_i)$ and (ID_i, W_i, h_1) to L_C and L_{H_1} respectively.
- *Corrupt-Query*: It returns the private key (s_i, x_i) of C_i.
- *Send-Query*:
  (1) When $\mathscr{A}$ makes a $Send(\prod_{C_i}^m, \text{“}Start\text{”})$ query, $\mathscr{CH}$ responds as follows. If $\prod_{C_i}^m = \prod_{C_I}^J$, $\mathscr{CH}$ chooses $z_i \in Z_q^*$, sets $R_i = Q_1$, and computes $k_1 = sR_i, CID_i = ID_i \oplus [k_1]_x$, $Z_i = z_i P, h_2 = H_2(ID_i, Z_i, P_{pub}, W_i, X_i)$ and $v = z_i - h_2 s_i \bmod q$. Finally, $\mathscr{CH}$ responds with $M_1 = (CID_i, R_i, W_i, X_i, h_2, v)$. Otherwise, $\mathscr{CH}$ responds with M_1 according to protocol description.
  (2) When $\mathscr{A}$ makes a $Send(\prod_S^j, M_1)$ query, $\mathscr{CH}$ does as follows. If $\prod_S^j = \prod_S^L$, $\mathscr{CH}$ computes $CID_i = ID_i \oplus [k_1]_x$, $Z_i' = vP + h_2(H_1(ID_i, W_i)P_{pub} - W_i)$ and $h' = H_2(ID_i, Z_i', P_{pub}, W_i, X_i)$, and then verifies whether h' is equal to h. If it holds, $\mathscr{CH}$ sets $R_S = Q_2$, chooses $k_3 \in G, sk \in Z_q^*$, computes $Auth = H_3(ID_i, R_S, Z_i', P_{pub}, k_1)$, adds $(ID_i, R_S, R_i, W_i, P_{pub}, k_3, sk)$ to L_{H_4}, and responds with $M_2 = (Auth, R_S)$. If $\prod_S^j \neq \prod_S^L$, then $\mathscr{CH}$ responds with M_2 according to protocol description.
  (3) When $\mathscr{A}$ makes a $Send(\prod_{C_i}^m, M_2)$ query, $\mathscr{CH}$ verifies whether $Auth$ is equal to $H_3(ID_i, R_S, Z_i, P_{pub}, k_1)$. If it holds, then $\prod_{C_i}^m$ accepts and terminates. Otherwise, $\prod_{C_i}^m$ terminates without accepting.
- *Reveal-Query*: If $\prod_{C_i}^m = \prod_{C_I}^J$, $\mathscr{CH}$ aborts. Otherwise, $\mathscr{CH}$ returns the session key sk.
- *Test-Query*: If $\prod_{C_i}^m \neq \prod_{C_I}^J$, $\mathscr{CH}$ aborts. Otherwise, $\mathscr{CH}$ randomly picks $\xi \in \{0,1\}^k$ and returns ξ as the answer.

If $\mathscr{A}$ violates key agreement, it means that $\mathscr{A}$ has made the corresponding H_4 query on $(ID_i, R_S, R_i, W_i, P_{pub}, k_3)$. $\mathscr{CH}$ chooses a tuple $(ID_i, R_S, R_i, W_i, P_{pub}, k_3, h_4)$ from L_{H_4}, obtains s_i and x_i from L_C and outputs $abP = k_3 + (x_i - s_i)Q_2$ as the solution to the CDH problem.

$\mathscr{CH}$ succeeds if and only if $\mathscr{CH}$ does not abort in the simulation and finds the corresponding item from L_{H_4} and $\mathscr{A}$ succeeds. If $\mathscr{A}$ indeed chooses $\prod_{C_I}^J$ who has a matching conversation to $\prod_S^L$ as the test oracle which is correct with probability $\frac{1}{q_S^2 q_C}$, then $\mathscr{CH}$ does not abort in the simulation. Thus $\mathscr{CH}$'s success probability is $\frac{\varepsilon}{q_S^2 q_C q_{H_4}}$. Therefore, if ε is non-negligible, then $\mathscr{CH}$'s success probability is also non-negligible. This contradicts the difficulty of the CDH problem. Hence, our scheme can provide key agreement.

5.3. Other security properties

In the following, we first show that the proposed scheme can provide perfect forward secrecy by using Theorem 3. Then we show that the proposed scheme can resist insider attack and provide user anonymity and leakage of session temporary secrets resistance.

Theorem 4. *Our scheme can provide perfect forward secrecy if the CDH problem is intractable.*

Proof. From the proof of Theorem 3, we know that the master key s of the server S are known by adversary $\mathscr{A}$. In Theorem 3, when $\mathscr{A}$ makes a corrupt query on ID_C, $\mathscr{CH}$ returns (s_C, x_C). Since the single Test query is asked before the Corrupt-query, Theorem 3 still holds. Hence our scheme can provide perfect forward secrecy.

Theorem 5. *Our scheme can resist insider attack if the ECDL problem is intractable.*

Proof. Assume that the client C's private key is (s_C, x_C) where $s_C = d_C - x_C$. From the client registration phase, the server S can know the value (d_C, X_C) where $X_C = x_C P$. To derive (s_C, x_C) from (d_C, X_C), the adversary must know x_C which is as hard as solving the ECDL problem. Therefore, Theorem 5 holds.

Theorem 6. *Our scheme can provide user anonymity if the CDH problem is intractable.*

Proof. In our scheme, the identity ID_C is protected by CID_C. The computation of $CID_C = ID_C \oplus [k_1]_x$ requires the knowledge of k_1, but R_C is sent to the server rather than k_1, and $k_1 = r_C P_{pub} = sR_C$. Without s and r_C, computing k_1 from R_C and $P_{pub} = sP$ is as hard as solving the CDH problem. Therefore, Theorem 6 holds.

Theorem 7. *Our scheme can provide leakage of session temporary secrets resistance if the CDH problem is hard.*

Proof. The session key of our scheme is $sk = H_4(ID_C, R_S, R_C, W_C, P_{pub}, k)$, where $R_C = r_C P, R_S = r_S P_{pub}, k = sr_S(R_C + PK_C - X_C) = (r_C + s_C - x_C)r_S P_{pub}$, $P_{pub} = sP, X_C = x_C P, PK_C = s_C P$. Assume the session ephemeral secrets r_C and r_S are disclosed, now we show that from this disclosure the adversary cannot compute sk. Obviously, the computation of sk requires the knowledge of k. To compute k, the adversary must know s or (s_C, x_C). However, without s and (s_C, x_C), computing k from P_{pub} and $PK_C - X_C$ is as hard as solving the CDH problem. Therefore, Theorem 7 holds.

6. Comparison with competitive schemes

In this section, we compare our scheme with several remote user authentication and key agreement schemes in terms of computation efficiency and security.

To measure the computation efficiency, we use the following symbols.

- P: a pairing;
- M_P: a pairing-based scalar multiplication;
- M_E: an ECC-based scalar multiplication;
- H_{M2P}: a map-to-point hash;
- H: a one-way hash;
- Inv: a modular inversion;
- PA: a point addition.

We adopt different hard platforms for the server and the client with low-power computing devices, due to their different computation abilities. Specifically, as in [16], the hard platforms for the client and the server are a 36MHZ Philips HiPer smart card with 16KB RAM memory and a PIV 3GHZ processor with 512MB memory and the Windows XP operation system, respectively. For the pairing-based protocols, to achieve 1024-bit RSA level security, we use the Ate pairing defined over a non-supersingular curve over a finite field F_p, with $p = 512$ bits and a large prime order $q = 160$ bits. For the ECC-based protocols without pairings, to achieve the same security level, we employ the ECC group on Koblitz elliptic curve $y^2 = x^3 + ax^2 + b$ defined on $F_{2^{163}}$ with $a = 1$ and b a 163-bit random prime. The running time of operations for the server and the client by using MIRACL library [17] are summarized in Table 1.

Table 1: Cryptographic operation time

	P	M_P	M_E	Inv	PA	H	H_{M2P}
Server (ms)	3.16	1.17	0.83	< 0.3	< 0.1	< 0.01	< 1
Client (s)	0.38	0.13	0.09	< 0.03	< 0.01	< 0.001	< 0.1

We judge these schemes' security by checking whether they can provide insider attack resistance, mutual authentication, perfect forward security, leakage of session temporary secrets resistance and user anonymity. Furthermore, we also judge these schemes' security by checking whether the security is formally proven.

In Table 2, we demonstrate comparisons among our scheme and five recently proposed remote user authentication and key agreement schemes for mobile client-server environment [6,7,10,11,12], where execution times are measured using Table 1. In Table 2, we know that the server side of our scheme requires $5M_E + 3PA + 4H$, since $vP + hPK_C$ can be computed simultaneously through the simultaneous multiple scalar multiplication [18].

From Table 2, it is easy to draw the following conclusions: (1) our scheme has stronger security and higher efficiency than schemes [6,7,11], and (2) our scheme has stronger security than schemes [10,12], but only loss a little efficiency. To sum up, our scheme is more secure, practical and suitable for mobile client-server environment.

7. Conclusion

In this paper, we have shown some identity-based user authentication and key agreement schemes cannot provide insider attack resistance or mutual authentication, and some of these schemes cannot provide user anonymity, perfect forward secrecy or leakage of session temporary secrets resistance. Then we have proposed an improved remote user authentication and key agreement scheme. Security analysis shows that our scheme can avoid these security problems. Under the ECDL problem and the CDH problem and in the random oracle model, we have shown that our scheme can achieve mutual authentication and key agreement. Protocol comparison shows that our scheme is more secure, piratical and suitable for mobile client-server environment.

Acknowledgement

This work is supported by National Natural Science Foundation of China (Grant Nos. 61202434, 61170270, 61121061), the Fundamental Research Funds for the Central Universities (Grant Nos. 2012RC0612, 2011YB01).

References

[1] M. Burmester and Y. Desmedt, A secure and efficient conference key distribution system, Eurocrypt 1994, 275-286 (1995).
[2] A. Shamir, Identity-based cryptosystems and signature schemes, Crypto 1984, 47-53 (1984).

Table 2: Protocol comparison

Schemes	Computational efficiency				Security					
	Client		Server		IAR	MA	PFS	LSTSR	UA	Provable
	Cost	Time	Cost	Time						
WT [6]	$4M_P+1PA+3H$	0.533s	$2P+2M_P+1PA+3H$	8.77ms	N	N	N	N	N	Y
He[7]	$3M_P+1Inv+3H$	0.423s	$1P+2M_P+2PA+4H$	5.72ms	N	N	N	N	N	Y
IB [10]	$3M_E+2PA+4H$	0.294s	$4M_E+2PA+1H_{M2P}+4H$	4.56ms	N	N	Y	N	Y	N
TTD [11]	$3M_E+1PA+1H_{M2P}+5H$	0.385s	$4M_E+1PA+1H_{M2P}+5H$	4.47ms	N	N	Y	Y	Y	N
HCH[12]	$3M_E+2H$	0.272s	$3M_E+1Inv+3H$	2.82ms	N	N	Y	N	N	Y
Our scheme	$4M_E+3H$	0.363s	$5M_E+4PA+4H$	4.57ms	Y	Y	Y	Y	Y	Y

IAR: insider attack resistance; MA: mutual authentication; PFS: perfect forward security;
LSTSR: leakage of session temporary secrets resistance. UA: User anonymity

[3] M. Das, A. Saxena, V. Gulati and D. Phatak, A novel remote user authentication scheme using bilinear pairings, Computer & Security **25(3)**, 184-189 (2006).

[4] D. Giri and P. Srivastava, An improved remote user authentication scheme with smart cards using bilinear pairings, http://eprint.iacr.org/2006/274.pdf.

[5] Y. Tseng, T. Wu and J. Wu, A pairing-based client authentication protocol for wireless clients with smart cards, Informatica **19(2)**, 285-302 (2008).

[6] T. Wu and Y. Tseng, An efficient client authentication and key agreement protocol for mobile client-server environment, Computer Networks **54**, 1520-1530 (2010).

[7] D. He, An efficient remote user authentication and key agreement protocol for mobile client-server environment from pairings, Ad Hoc Networks **10(6)**, 1009-1016 (2012).

[8] J. Yang and C. Chang, An ID-based remote mutual authentication with key agreement scheme for mobile devices on elliptic curve cryptosystem, Computers & Security **28(3-4)**, 138-143 (2009).

[9] E. Yoon and K. Yoo, Robust ID-based remote mutual authentication with key agreement scheme for mobile devices on ecc, IEEE International Conference on Computational Science and Engineering, 633-640 (2009).

[10] S. Islam and G. Biswas, A more efficient and secure ID-based remote mutual authentication with key agreement scheme for mobile devices on elliptic curve cryptosystem, Journal of Systems and Software **84(11)**, 1892-1898 (2011).

[11] T. Truong, M. Tran and A. Duong, Improvement of the more efficient and secure ID-based remote mutual authentication with key agreement scheme for mobile devices on ECC, The 26th Conference on Advanced Information Networking and Applications Workshops, 698-703 (2012).

[12] D. He, J. Chen and J. Hu, An ID-based client authentication with key agreement protocol for mobile client-server environment on ECC with provable security, Information fusion **13(3)**, 223-230 (2012).

[13] F. Li, X. Xin and Y. Hu, Indentity-based broadcast signcryption, Computer Standards and Interfaces **30(1-2)**, 89-94 (2008).

[14] D. Pointcheval and J. Stern, Security arguments for digital signatures and blind signatures, Journal of Cryptology **13(3)**, 361-369 (2000).

[15] X. Cao, W. Kou and X. Du, A pairing-free identity-based authenticated key agreement protocol with minimal message exchanges, Information Sciences **180(15)**, 2895-2903 (2010).

[16] M. Scott, N. Costigan and W. Abdulwahab, Implementing cryptographic pairings on smartcards, in: Cryptographic Hardware and Embedded Systems. CHES 2006, 134-147 (2006).

[17] Ltd. CertiVox, Miracl library, http://certivox.com/index.php/solutions/miracl-crypto-sdk/.

[18] D. Hankerson1, J. Hernandez and A. Menezes. Software Implementation of Elliptic Curve Cryptography over Binary Fields. CHES 2000, 1-24 (2000).

Haiyan Sun is currently a Ph.D. candidate in State Key Laboratory of Networking and Switch Technology, Beijing University of Posts and Telecommunications. Her present research interests include cryptography, information security, and security in IoT (Internet of Things) and cloud computing.

Qiaoyan Wen received the B.S. and M.S. degrees from Shaanxi normal University in 1981 and 1984, respectively, and the Ph.D. degree from Xidian University in 1997. Now, she is a professor of Beijing University of Posts and Telecommunications. Her present research interests include cryptography and information security.

Hua Zhang received the B.S. and M.S. degrees from Xidian University in 2002 and 2005, respectively, and the Ph.D. degree from Beijing University of Posts and Telecommunications in 2008. Now she is an associate professor of Beijing University of Posts and Telecommunications. Her research interests include cryptographic protocols, and security in IoT, cloud computing, industrial control system and Mobile Internet.

Zhengping Jin received the B.S. and M.S. degrees from Anhui Normal University in 2004 and 2007, respectively, and the Ph.D. degree from Beijing University of Posts and Telecommunications in 2010. Now he is a lecturer of Beijing University of Posts and Telecommunications. His research interests include design and analysis of cryptographic protocols, and security in IoT.