Kuwait J. Sci. **42 (1)** pp. 91-108, 2015

Strongly secure certificateless one-pass authenticated key agreement scheme

BAOJUN HUANG * AND HANG TU**

**School of Mathematics and Statistics, Wuhan University, Wuhan 430072, Hubei, China*

*** School of Computer, Wuhan University, Wuhan 430072, Hubei, China*

Email: tuhang2013@163.com

ABSTRACT

As certificateless public key cryptography (CLPKC) could solve the problem of key escrow, recently it has been studied more and more. Certificateless authenticated key agreement (CLAKA), as an important part of the CLPKC, also attracts considerable attention. So far, many kinds of provably secure one-pass CLAKA schemes have been proposed. However, only few of them are given security proof in a formal model. In this paper, we propose a novel one-pass CLAKA scheme and demonstrate that it is provably secure under the gap bilinear computational Diffie-Hellman (GBCDH) assumption and the Gap Computational Diffie-Hellman (GCDH) assumption in the extended Canetti-Krawczyk (eCK) model. And as far as we know, our scheme is the first provably secure CLAKA scheme in the eCK model.

Keywords: Authenticated key agreement; certificateless cryptography; eCK model; one-pass; provable security.

INTRODUCTION

Now public key cryptography (PKC) is a widely used technique in our life, such as e-bank and pretty good privacy (PGP). In traditional PKC, a certificate with an identity and a public key binding together is issued by a trusted authority. But it costs too much to manage the certificates, as a complex public key cryptography infrastructure is needed. In order to overcome the problem, Shamir (1984) proposed the identity-based (ID-based) public key cryptosystem. But a trusted key generation centre (KGC) is needed to generate the entity's private key with his/her identity in his scheme. So the KGC knows the private key of any user. This also leds to the problem of key escrow. To avoid this problem, Al-Riyami & Paterson (2003) proposed the certificateless public key cryptography (CLPKC), considered as a concept of an intermediate between the traditional public key infrastructure (PKI) and identity-based cryptography (IBC).

Authenticated key agreement (AKA) scheme as a fundamental cryptographic primitive becomes more and more important. Al-Riyami & Paterson (2003) firstly proposed a CLAKA scheme. Since then, several scholars proposed some two-pass CLAKA schemes (He *et al*., 2011; He *et al*., 2012). In these schemes, two or more parities intend to communicate over the insecure network controlled by an adversary, and negotiate a session key for the next secure communication.

Compared with the two-pass scheme, without interaction with Bob (as a receiver), Alice (as a sender) is possible to produce the session key alone in the one-pass scheme. So the scheme suits the scene more where the receiver may be not online (e.g. in a secure email system) (Huang & Cao, 2011). Later, Chen *et al*. (2009) proposed the first one-pass CLAKA scheme. However, they didn't provide a CLAKA security model or a secure proof in their paper.

Bellare & Rogaway (1993) were the first to propose a formal security model which is used for authentication and key distribution. From then on, their model was extended and modified several times by many researchers (Bellare & Rogaway, 1995; Canetti & Krawczyk, 2001; Choo *et al*., 2005). One of the most promising extended models is the Canetti-Krawczyk (CK) model (Canetti & Krawczyk, 2001). After comparing with security models, Choo *et al*. (2005) found that all these kinds of models try to cover the general security attribute asmuch as possible. LaMacchia *et al*. (2007) proposed a relatively strong security model for AKA schemes, which is known as the extended Canetti-Krawczyk (eCK) model. This eCK model contains many satisfying security attributes, such as key-compromise impersonation (KCI) resilience, weak perfect forward secrecy (wPFS) and ephemeral secrets reveal resistance. However, the original CK model (Canetti & Krawczyk, 2001) does not contain KCI attacks. Recently, Lippold *et al*. (2009) presented the first eCK model for CLAKA scheme based on LaMacchia *et al*.'s work.

In this paper, a formal security model is proposed for one-pass CLAKA scheme based on Lippold *et al*.'s work (Lippold *et al*., 2007). And we also propose a novel one-pass CLAKA scheme which is provably secure in the random oracle model. The rest of paper is organized as follows. Next section describes several preliminaries. Section following the next shows a one-pass CLAKA scheme. And the security analysis of the proposed scheme is presented separately, followed by the performance analysis. Finally, in the last section we draw a conclusion.

PRELIM INARIES

BACKGROUND OF BILINESR PAIRINGS

Let G_1 and G_2 be an additive cyclic group and a multiplicative cyclic group of prime order q. Let P be a G_1's arbitrary generator. Let a bilinear pairing be $e: G_1 \times G_1 \to G_2$, which satisfies the three conditions:

1. Bilinearity

$e(aQ,bR)=e(Q,R)^{ab}$, where $Q,R\in G_1$, $a,b\in Z_q^*$.

2. Non-degeneracy

$e(P,P)\neq 1_{G_2}$.

3. Computability

There is a high efficient algorithm to compute $e(Q,R)$ for all $Q,R\in G_1$.

Associated with super singular elliptic curves or abelian varieties, the Weil and Tate pairings are both able to be modified to create such admissible pairings (Galbraith *et al*., 2002; Boneh & Franklin, 2003). Next, we present several computational problems, which are usually used in the scheme security analysis. The several computational problems are all defined based on the bilinear group (G_1,G_2,q,P,e).

Computational Diffie-Hellman (CDH) problem. Uniformly choose random $a,b\in Z_q^*$ and give (aP, bP), compute abP.

Bilinear Computational Diffie-Hellman (BCDH) problem. Uniformly choose random $a,b,c\in Z_q^*$ and give (aP, bP, cP), compute $e(P,P)^{abc}$.

For convenience, we define the function cdh and $bcdh$ as $cdh(aP,bP)=abP$ and $bcdh(aP,bP,cP)=e(P,P)^{abc}$ separately.

Decision Diffie-Hellman (DDH) problem. Uniformly choose random $a,b\in Z_q^*$, $X\in G_1$ and give (aP, bP, Z), decide whether the equation $cdh(aP,bP)=X$ holds.

Gap Computational Diffie-Hellman (DCDH) problem. Uniformly choose random $a,b\in Z_q^*$ and give (aP, bP, $\mathcal{O}_{ddhp}$), compute abP, where $\mathcal{O}_{ddhp}$ is a decision oracle that inputs (aP, bP, X), returns true if $cdh(aP,bP)=X$; else returns false.

The GCDH assumption states that any polynomial-time algorithm to overcome the GCDH problem's probability is negligible. It is easy to say that map e is a decision oracle of DCDH problem.

Decision Bilinear Diffie-Hellman (DBDH) problem. Uniformly choose random $a,b,c\in Z_q^*$, $X\in G_2$ and give (aP, bP, cP, X), answer whether the equation $bcdh(aP,bP,cP)=X$ holds.

Gap Bilinear Diffie-Hellman (GBCDH) problem. Uniformly choose random $a,b,c\in Z_q^*$, compute $e(P,P)^{abc}$ and give (aP, bP, cP, $\mathcal{O}_{dbdhp}$), where $\mathcal{O}_{dbdhp}$ is a decision oracle that inputs (aP, bP, cP, X), returns true if $bcdh(aP,bP,cP)=X$; else returns false.

The GBCDH assumption states that any polynomial-time algorithm to overcome the GBCDH problem's probability is negligible.

Security model for one-pass CLAKA schemes

In CLAKA scheme, there are six polynomial-time algorithms (Al-Riyami & Paterson, 2003; Lippold *et al*., 2007): $Setup$, $Partial\text{-}Private\text{-}Key\text{-}Extract$, $Set-Secret-Value$, $Set\text{-}Private\text{-}Key$, $Set-Public-Key$ and $Key-Agreement$. There are two types of attackers in CLAKA schemes, i.e., Type 1 and Type 2 (Al-Riyami & Paterson, 2003; Lippold *et al*., 2007). We denote $\mathscr{A}1$ as Type 1 adversary representing a dishonest user, $\mathscr{A}2$ as Type 2 adversary representing a malicious KGC. $\mathscr{A}1$ cannot query the master key, but he/she can replace the public key of an arbitrary entity with any value of his/her choice, without certificate contained in CLPKC. $\mathscr{A}2$ is able to query the master key and can obtain the partial keys, but he/she is not able to take the place of the user's public key.

Let $U=\{U_1,U_2,\sim,U_n\}$ denote as the set of participants. And the network is not security over which scheme messages are exchanged. The $\prod_{i,j}^{s}$ symbol represents party i runs at the s th scheme session with intended partner party j. The session $\prod_{i,j}^{s}$ enters an $accepted$ state when it computes a session key $SK_{i,j}^{s}$. It is also possible that a session may be over without an $accepted$ state. It is assumed to be public information that a session ends with acceptance or not. The $pid=(ID_i,ID_j)$ is specified for the session $\prod_{i,j}^{s}$ partner ID. The session ID sid of $\prod_{i,j}^{s}$ at party i is the recode of the information exchanged with party j in the session. The session $\prod_{i,j}^{s}$ and $\prod_{j,i}^{t}$ are thought to be matching only if they hold the same pid (and sid).

Lippold *et al*. (2009) convert the original eCK model (LaMacchia *et al*., 2007) from the PKI-based setting to the CLPKC setting. The eCK model in the CLPKC setting is defined as the game between a challenger $\mathcal{C}$ and an adversary $\mathscr{A}\in\{\mathscr{A}1,\mathscr{A}2\}$. In the model, $\mathscr{A}$ is considered as a probabilistic polynomial-time Turing machine. The adversary $\mathscr{A}$ is able to obtain all the communications. Participant does not communicate directly with others, only responds to $\mathscr{A}$'s queries. $\mathscr{A}$ holds the authority to relay, modify, delay, interleave or delete all the message flows in the system. $\mathscr{A}$ may ask a polynomial number of the following queries.

The game consists of two stages. In the first one, the adversary $\mathscr{A}$ is permitted to ask the several queries arbitrarily:

$Create(i)$: $\mathscr{A}$ asks $\mathcal{C}$ to create a new participant with identity ID_i. Once accepting this query, $\mathcal{C}$ generates a key pair for the new participant i, which consists of private and public keys.

$RevealSessionKey(\prod_{i,j}^{s})$: If the session is not accepted, return □, else reveal the accepted session key.

$RevealPartialPrivateKey(i)$: respond with participant i 's partial private key.

$RevealSecretValue(i)$: return secret value x_i corresponding to the public key. If the $ReplacePublicKey(i, pk)$ query has been asked before, it responds $\perp$.

$ReplacePublicKey(i, pk)$: choose pk to replace the party i 's public key. And then party i will use pk for all the next communication and computation.

$RevealEphemeralKey(\prod_{i,j}^{s})$: the ephemeral secret is responded by party i , which is used in session $\prod_{i,j}^{t}$.

$Send(\prod_{i,j}^{s}, m)$: If there is no session $\prod_{i,j}^{s}$, it will be created as initiator at party i if $m = \lambda$, else as a responder at party j . If the participants have not been initiated before, all the public and private keys are generated respectively. Once accepting the message m , this scheme is carried out. After party i has sent and received the finally specified messages in the scheme, a decision will be output to indicate acceptance or rejection of the session. In one-round scheme, party i runs as following:

$m = \lambda$: Party i selects an ephemeral value and only replies an outgoing message.

$m \neq \lambda$: If party i is a responder, it selects an ephemeral value for the session and replies an outgoing message m' and gives a decision indicating acceptance or rejection of the session. Else, if party i is an initiator, it replies a decision to accept or reject the session. In this scheme, we request $i \neq j$, i.e. any party will not perform a session with itself.

Once $\mathscr{A}$ decides that the first phase is finished, the next phase is started by selecting a fresh session $\prod_{i,j}^{s}$ and issuing a $Test(\prod_{i,j}^{s})$ query. The definition lists like following.

Definition 1 (Freshness for One-Pass AKE Scheme). Let case $\prod_{i,j}^{s}$ be a completed session run by honest parties i and j . $\prod_{i,j}^{s}$ is defined fresh if the three properties are all not met:

- The adversary $\mathscr{A}$ reveals the session $\prod_{i,j}^{s}$'s key or $\prod_{i,j}^{s}$'s matching session (if exists).
- j is concerned with $\prod_{j,i}^{t}$, which matches to $\prod_{i,j}^{s}$ and:
  - If $\prod_{i,j}^{s}$ is as an initiator session, the adversary $\mathscr{A}$ either reveals all of i 's partial private key and secret value and $\prod_{i,j}^{s}$'s ephemeral secret or both of partial private key and secret value of j .
  - If $\prod_{i,j}^{s}$ is as a responder session, $\mathscr{A}$ either reveals both of i 's partial private key and secret value or all of j 's partial private key and secret value and the ephemeral secret of $\prod_{i,j}^{s}$.

- No session matches to $\prod_{i,j}^{s}$ and :
  - If $\prod_{i,j}^{s}$ is as an initiator session, $\mathscr{A}$ either reveals all of i 's partial private key and secret value and the ephemeral secret of $\prod_{i,j}^{s}$ or both of j 's partial private key and secret value.
  - If $\prod_{i,j}^{s}$ is as a responder session, $\mathscr{A}$ either reveals both of i 's partial private key and secret value or both of j 's secret value and partial private key.

$Test(\prod_{i,j}^{s})$: in some cases, A may choose an oracle, says $\prod_{i,j}^{s}$, to take a single Test query, where $\prod_{i,j}^{s}$ has to be a fresh oracle. And it makes a fair coin flip $b \in \{0,1\}$ to answer the query. If $b = 0$, the session key is replied held by $\prod_{i,j}^{s}$, else a sample randomly choosen is replied as the session key.

After issuing the $Test(\prod_{i,j}^{s})$ query, A can continue querying unless the test session $Test(\prod_{i,j}^{s})$ remains fresh. We note that partial corruption is allowed, because our security model has to benefit from it. Moreover, after the test session completes, any party could issue the replace public key query.

When the game is over, $\mathscr{A}$ has to reply a guess bit b'. Only if $b' = b$ holds, the adversary $\mathscr{A}$ wins. The $Adv_{\mathscr{A}}(k)$ is denoted as $\mathscr{A}$'s advantage of winning this game, defined as: $Adv_{\mathscr{A}}(k) = \left| \Pr[b' - b] - \frac{1}{2} \right|$, where k is a security parameter.

Definition 2. A CLAKA scheme is considered to be secure if:

(1) In the presence of a benign adversary on $\prod_{i,j}^{s}$ and $\prod_{j,i}^{t}$, the two oracles always negotiate the same session key, which is uniformly random distribution.

(2) For any adversary $\mathscr{A} \in \{\mathscr{A}1, \mathscr{A}2\}$, $Adv_{\mathscr{A}}(k)$ is negligible.

OUR SCHEME

In this section, we will propose a novel one-pass CLAKA scheme. Our scheme contains six polynomial-time algorithms, i.e. $Setup$, $Partial\text{-}Private\text{-}Key\text{-}Extract$, $Set-Secret-Value$, $Set\text{-}Private\text{-}Key$, $Set-Public-Key$, and $Key-Agreement$.

$Setup$: Upon receiving a security parameter k, the KGC performs an algorithm to produce the master key and system parameters $params$.

1) KGC selects a prime number q, an additive group G_1 and a multiplicative group G_2, of which both are the order q, a bilinear map $e : G_1 \times G_1 \rightarrow G_2$, G_1's generator P .

2) KGC selects randomly a number $z \in Z_n^*$ as the master key and computers the public key $P_{pub} = zP$.
3) KGC selects two secure hash functions H_1 and H_2, where $H_1 : \{0,1\}^* \to G_1$ and $H_2 : \{0,1\}^* \times \{0,1\}^* \times G_1 \times G_2 \times G_1 \to \{0,1\}^k$.
4) KGC publics the system parameters $params = \{G_1, G_2, q, P, e, P_{pub}, H_1, H_2\}$ and stores the mater key z secretly.

$Partial\text{-}Private\text{-}Key\text{-}Extract$: Upon receiving a user's ID, KGC performs the algorithm to compute ID_i's partial private key.

1) KGC computes $Q_i = H_1(ID_i)$.
2) KGC computes $D_i = zQ_i$, and transmits to the user via a secure channel.

$Set - Secret - Value$: With the system parameters, the user with identity ID_i performs the algorithm to generate his secret.

1) The user selects a random number $x_i \in Z_n^*$.
2) The user sets x_i as his secret value and keeps it secret.

$Set\text{-}Private\text{-}Key$: With the partial private key D_i and the secret value x_i, the user executes the algorithm to generate his private key. The user sets $sk_i = (x_i, D_i)$ as his private key.

$Set - Public - Key$: With the secret value x_i, the user performs the algorithm to produce the public key.

1) The user computes $P_i = x_i P$.
2) The user sets $pk_i = (P_i)$ as his public key.

$Key - Agreement$: The algorithm will be executed between the user A and the user B when they intend to generate a session key. We set A's identity, private key and public key as ID_A, $sk_A = (x_A, D_A)$ and $pk_A = (P_A)$ separately. Let B's identity, private key and public key be ID_B, $sk_B = (x_B, D_B)$ and $pk_B = (P_B)$ separately. The algorithm is displayed in Figure 1, and the detail is shown as follows.

1) A selects a random number $t_A \in Z_n^*$, computes $T_A = t_A \cdot P$ and sends the message $M_1 = \{ID_A, T_A\}$ to B. A also computes $K_{AB}^1 = e(t_A P_{pub} + D_A, Q_B)$, $K_{AB}^2 = (t_A + x_A)P_B$ and the session key $sk = H_2(ID_A \| ID_B \| T_A \| K_{AB}^1 \| K_{AB}^2)$.
2) Upon receiving the message $M_1 = \{ID_A, T_A\}$, B computes $K_{BA}^1 = e(T_A + Q_A, D_B)$, $K_{BA}^2 = x_B(T_A + P_A)$ and the session key $sk = H_2(ID_A \| ID_B \| T_A \| K_{BA}^1 \| K_{BA}^2)$.

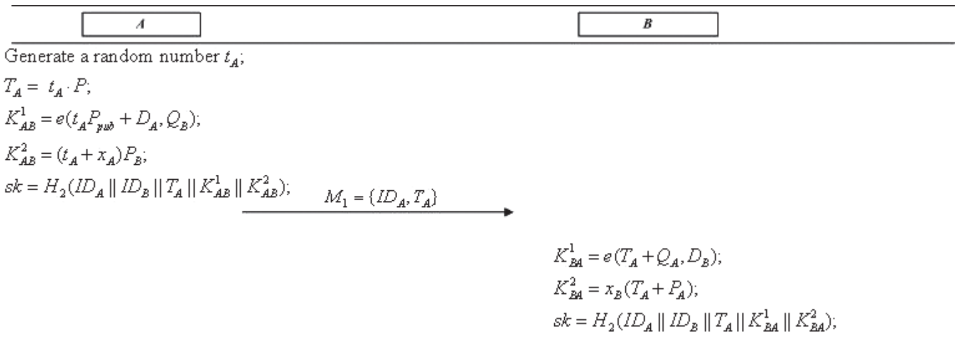
A
B
Generate a random number t_A;
$T_A = t_A \cdot P$;
$K_{AB}^1 = e(t_A P_{pub} + D_A, Q_B)$;
$K_{AB}^2 = (t_A + x_A) P_B$;
$sk = H_2(ID_A \| ID_B \| T_A \| K_{AB}^1 \| K_{AB}^2)$;
$M_1 = \{ID_A, T_A\}$
$K_{BA}^1 = e(T_A + Q_A, D_B)$;
$K_{BA}^2 = x_B(T_A + P_A)$;
$sk = H_2(ID_A \| ID_B \| T_A \| K_{BA}^1 \| K_{BA}^2)$;

Fig. 1. Key agreement of our scheme

Since $T_A = t_A \cdot P$, $P_A = x_A \cdot P$, $D_A = sQ_A$, $T_B = t_B \cdot P$, $P_B = x_B \cdot P$ and $D_B = sQ_B$, then we have

$$\begin{aligned} K_{AB}^1 &= e(t_A P_{pub} + D_A, Q_B) \\ &= e(t_A zP + zQ_A, Q_B) = e(zT_A + zQ_A, Q_B) \\ &= e(T_A + Q_A, zQ_B) = e(T_A + Q_A, D_B) \\ &= K_{BA}^1 \end{aligned} \tag{1}$$

and

$$\begin{aligned} K_{AB}^2 &= (t_A + x_A) P_B = (t_A + x_A) x_B P \\ &= x_B (t_A P + x_A P) = x_B (T_A + P_A) \\ &= K_{BA}^2 \end{aligned} \tag{2}$$

The session key generated by A is equal to B's. Therefore, the proposed scheme is correct.

SECURITY ANALYSIS

In this section, we will demonstrate that the one-pass CLAKA scheme is provably secure in the random oracle model, which treats two hash functions H_1 and H_2 as two random oracles. To prove security of the proposed scheme, the following lemmas and theorem are necessary.

Lemma 1. Suppose $\prod_{i,j}^s$ and $\prod_{i,j}^t$ are two matching oracles and the state of them is accepted, then they will randomly generate the same session key distributing uniformly in sample space.

Proof. Based on the above correctness analysis of the proposed one-pass CLAKA, both of $\prod_{i,j}^s$ and $\prod_{i,j}^t$ could generate the same session key

$sk = H_2(ID_A \| ID_B \| T_A \| e(T_A + Q_A, zQ_B) \| x_B(t_A P + x_A P))$. Besides, the user will generate a new random number t_A for a new session. Then, the generated session key is in the sample key space of random uniform distribution.

Lemma 2. Suppose that the GBCDH problem is intractable, then the scheme we proposed is secure against the Type I adversary.

Proof. If there is a Type I adversary $\mathscr{A}1$ shown in Section 2, who wins the game with a non-negligible advantage $Adv_{\mathscr{A}1}(k)$. With the adversary, the algorithm $\mathscr{C}$ could be built to overcome the GBCDH problem.

Suppose that a party could have n_0 session at most. Let n_1 and n_2 be the number of parties and hash queries in the game. The H_1 and H_2 are viewed as two random oracles, then $\mathscr{A}1$ only uses the following three ways to differ the session key from the random number.

Guessing attack: The adversary $\mathscr{A}1$ could guess the session key correctly.

Key-replication attack: $\mathscr{A}1$ could forge a non-matching session which has the same session key.

Forging attack: $\mathscr{A}1$ queries the random oracle H_2 with the message $(ID_I, ID_J, T_I, K_{IJ}^1, K_{IJ}^2)$ in the tested session $\prod_{I,J}^S$. Then, $\mathscr{A}1$ could get K_{IJ}^1 and K_{IJ}^2.

It is easy to say that the adversary $\mathscr{A}1$ could guess the output of H_2 correctly with probability $O(1/2^k)$ since H_2 is considered to be a random oracle. Besides, the session identity is included when a session key is derived and any two non-matching sessions hold different identities. The success probability of the **guessing attack** and the **key-replication attack** is negligible. Hence, we just need to consider the success probability of the forging attack.

As H_2 is modeled as a random oracle, it is negligible to guess H_2's output, of which probability is $O(1/2^k)$. All information are input to the key derivation function H_2, which is able to uniquely distinguish the matching session. Because any two non-matching sessions hold different identities and ephemeral public keys, the probability of successful **Key-replication attack** is also negligible. Thus the probability of **Guessing attack** and **Key-replication attack** is negligible, the analysis of Forging attack becomes the main of the rest proof. As $\mathscr{A}1$ performs the **Forging attack**, he/she cannot obtain advantages to win the game against the scheme unless it queries the session key from the H_2 oracle.

$\mathscr{C}$ choses two random indexes I and J from the set $\{1, \sim, n_1\}$, where $I \neq J$. $\mathscr{C}$ also chooses a random number S from the set $\{1, \sim, n_0\}$ and determines $\prod_{I,J}^S$ as the $Test$ session. Suppose $\prod_{J,I}^T$ be the matching session of the $Test$ session $\prod_{I,J}^S$

. It is easy to say that $\mathcal{A}1$ chooses $\prod_{I,J}^{S}$ as the $Test$ session with the probability $\frac{1}{n_0 n_1^2}$. We will consider the following two case for $\mathcal{A}1$'s attack.

CASE 1: No honest parties own the matching session.

CASE 2: There exists an honest one, who holds the matching session.

- The analysis of CASE 1

It is easy to say that the $\prod_{I,J}^{S}$ is either initiator session or responder session. Then, we should analyze both of the two cases.

- *Initiator session*

Since $\mathcal{A}1$ is strong Type 1 adversary, $\mathcal{A}1$ is able to possess all parties' secret key x_i through $ReplacePublicKey$ queries. By definition of the fresh session, there are four choices to $\mathcal{C}$ for the strategy of $\mathcal{A}1$:

CASE 1.1: The adversary $\mathcal{A}1$ gets I 's partial private key but does not get the Test session's ephemeral private key.

Given a GBCDH case ($U = uP$, $Z = zP$, $W = wP$, $\mathcal{O}_{dbdhp}$), we will construct an algorithm to solve the problem using $\mathcal{A}1$, where $\mathcal{O}_{dbdhp}$ denotes a decision oracle. For input (uP, zP, wP, X), the decision oracle outputs 1 if the equation $bcdh(uP, zP, wP) = X$ holds; otherwise, it outputs 0. $\mathcal{C}$ sets $Z = zP$ as the system public key P_{pub}, sends the parameters $params = \{G_1, G_2, q, P, e, P_{pub} = Z, H_1, H_2\}$ to $\mathcal{A}1$, and replies $\mathcal{A}1$'s queries as following.

$Create(i)$: Upon accepting the query, $\mathcal{C}$ checks whether the tuple $(ID_i, Q_i, D_i, x_i, P_i)$ is included in the list L_C. If it is included, $\mathcal{C}$ returns Q_i to $\mathcal{A}1$; otherwise, the following steps will be executed.

If $i = J$, $\mathcal{C}$ chooses a random number $x_i \in Z_n^*$, calculates $P_i = x_i P$, and sets $Q_i = H_1(ID_i) \leftarrow W$ and $D_i \leftarrow \perp$. Then, $\mathcal{C}$ stores ($ID_i, Q_i, \perp, x_i, P_i$) and (ID_i, Q_i) into L_C and L_{H_1} separately. $\mathcal{C}$ returns Q_i to $\mathcal{A}1$;

Otherwise ($i \neq J$), $\mathcal{C}$ selects two random numbers $l_i, x_i \in Z_n^*$, computes $P_i = x_i P$, $D_i = l_i P_{pub}$ and sets $Q_i = H_1(ID_i) \leftarrow l_i P$. Then, $\mathcal{C}$ stores $(ID_i, Q_i, D_i, x_i, P_i)$ and (ID_i, Q_i) in L_C and L_{H_1} separately. $\mathcal{C}$ returns Q_i to $\mathcal{A}1$;

$H_1(ID_i)$: Upon receiving the query, $\mathcal{C}$ checks the tuple (ID_i, Q_i) is included in the list L_{H_1} or not; if exist, $\mathcal{C}$ returns Q_i to $\mathcal{A}1$; otherwise, $\mathcal{C}$ makes $Create(i)$ query and then $\mathcal{C}$ returns Q_i to $\mathcal{A}1$.

$H_2(ID_i, ID_j, T_i, Z_1, Z_2, h)$: Upon receiving the query, $\mathcal{C}$ checks whether the tuple ($ID_i, ID_j, T_i, Z_1, Z_2, h$) is included in the list $H_2(ID_i, ID_j, T_i, Z_1, Z_2, h)$. If it is include, $\mathcal{C}$ returns h to $\mathscr{A}1$; otherwise, the following steps will be executed.

(1) If $ID_i = ID_J$, $\mathcal{C}$ checks whether a tuple ($ID_i, ID_j, T_i, *$) is include in the list L_S. If L_S does not contain such tuple, $\mathcal{C}$ generates a random number $h \in \{0,1\}^k$, and adds ($ID_i, ID_j, T_i, Z_1, Z_2, h$) into L_{H_2}; otherwise, $\mathcal{C}$ computes $\overline{Z}_1 = \dfrac{Z_1}{e(T_i, D_j)}$, $\overline{Z}_2 = Z_2 - x_j T_i$ and verifies whether both of the equations $\overline{Z}_2 = cdh(P_i, P_j)$ and $\overline{Z}_1 = bcdh(P_{pub}, Q_i, Q_j)$ hold. If both of them hold, $\mathcal{C}$ stores ($ID_i, ID_j, T_i, Z_1, Z_2, sk$) into L_{H_2}, where sk is the session key from L_S; else $\mathcal{C}$ selects a random number $h \in \{0,1\}^k$ and puts ($ID_i, ID_j, T_i, Z_1, Z_2, h$) into L_{H_2}.

(2) Otherwise ($ID_j \neq ID_J$), $\mathcal{C}$ refers to the table L_S for entry ($ID_i, ID_j, T_i, *$). If there is no such entry, $\mathcal{C}$ selects a random number $h \in \{0,1\}^k$ and adds the new entry ($ID_i, ID_j, T_i, Z_1, Z_2, h$) into L_{H_2}. Otherwise, $\mathcal{C}$ computes $\overline{Z}_1 = \dfrac{Z_1}{e(D_i, Q_j)}$, $\overline{Z}_2 = Z_2 - x_i P_j$ and verifies $\overline{Z}_1 = bcdh(P_{pub}, Q_j, T_i)$ (by handing ($P_{pub}, Q_j, T_i, \overline{Z}_1$) to the decision oracle $\mathcal{O}_{dbdhp}$) and $\overline{Z}_2 = cdh(T_i, P_j)$ (by checking $e(\overline{Z}_2, P) = e(T_i, P_j)$). If both of the equations are equal, $\mathcal{C}$ stores ($ID_i, ID_j, T_i, Z_1, Z_2, sk$) into L_{H_2}, where sk comes from L_S. Else $\mathcal{C}$ selects a random number $h \in \{0,1\}^k$ and stores ($ID_i, ID_j, T_i, Z_1, Z_2, h$) into L_{H_2}.

$RevealPartialPrivateKey(i)$: Upon receiving the query, $\mathcal{C}$ checks whether i and J are equal. If they are equal, $\mathcal{C}$ stops the simulation; otherwise, $\mathcal{C}$ loops up L_C for the tuple (ID_i, Q_i, D_i, x_i, P_i) and returns D_i to $\mathscr{A}1$.

$RevealSecretValue(i)$: Upon receiving the query, $\mathcal{C}$ loops up L_C for the tuple (ID_i, Q_i, D_i, x_i, P_i) and returns x_i to $\mathscr{A}1$.

$RevealSecretValue(i)$: Upon receiving the query, $\mathcal{C}$ loops up L_C for the tuple (ID_i, Q_i, D_i, x_i, P_i). Then, he replaces x_i and P_i with x_i' and P_i' separately, where $pk = (P_i')$ and $P_i' = x_i' P$.

$RevealEphemeralKey(\prod_{i,j}^s)$: Upon receiving the query, $\mathcal{C}$ checks whether $\prod_{i,j}^s$ and $\prod_{I,J}^S$ are equal. If they are equal, $\mathcal{C}$ terminates the simulation; else returns the ephemeral private key to $\mathscr{A}1$.

$RevealSessionKey(\prod_{i,j}^{s})$: Upon receiving the query, $\mathcal{C}$ checks whether either $\prod_{i,j}^{s} = \prod_{I,J}^{S}$ or $\prod_{i,j}^{s} = \prod_{J,I}^{T}$ is equal. If either of them is equal, $\mathcal{C}$ terminates the simulation; else returns the session key to $\mathcal{A}1$.

$Send(\prod_{i,j}^{s}, m)$: Upon receiving the query, $\mathcal{C}$ checks whether $\prod_{i,j}^{s}$ and $\prod_{I,J}^{S}$ are equal. If they are equal, $\mathcal{C}$ returns $T_i = U$ to $\mathcal{A}1$; otherwise, $\mathcal{C}$ checks whether ID_i and ID_J are equal. If they are not equal, $\mathcal{C}$ replies the query by the description of the scheme; otherwise; $\mathcal{C}$ checks the equation $m = \lambda$ holds and does as follows.

(1) If the equation holds, $\mathcal{C}$ randomly chooses a number $t_i \in Z_n$ and returns $T_i = t_i P$ to $\mathcal{A}1$. $\mathcal{C}$ refers to the list L_{H_2} for entry ($ID_i, ID_j, T_i, *, *, *$). If there is no such entry, $\mathcal{C}$ randomly chooses a number $sk \in \{0,1\}^k$ and adds the new entry (ID_i, ID_j, T_i, sk) into L_S; otherwise, $\mathcal{C}$ computes $\overline{Z}_1 = \dfrac{Z_1}{e(t_i P_{pub}, Q_j)}$, $\overline{Z}_2 = Z_2 - t_i P_j$ and verifies whether both of the equations $\overline{Z}_1 = bcdh(P_{pub}, Q_i, Q_j)$ and $\overline{Z}_2 = cdh(P_i, P_j)$ hold. If both of them are equal, $\mathcal{C}$ stores (ID_i, ID_j, T_i, h) into L_S. Otherwise, $\mathcal{C}$ chooses a random number sk and adds (ID_i, ID_j, T_i, sk) into L_S.

(2) Otherwise ($m \neq \lambda$), $\mathcal{C}$ refers to the list L_{H_2} for entry ($ID_j, ID_i, T_j, *, *, *$). If none of tuple is included in L_S, $\mathcal{C}$ generates a random number $sk \in \{0,1\}^k$ and stores (ID_j, ID_i, T_j, sk) in L_S; otherwise, $\mathcal{C}$ computes $\overline{Z}_1 = \dfrac{Z_1}{e(D_j, Q_i)}$, $\overline{Z}_2 = Z_2 - x_i T_j$ and verifies whether both of the equations $\overline{Z}_1 = bcdh(P_{pub}, Q_i, T_j)$ and $e(\overline{Z}_2, P) = e(P_i, P_j)$ hold. If both of them hold, $\mathcal{C}$ inserts the tuple (ID_i, ID_j, T_i, h) into L_S; else $\mathcal{C}$ generates a random number $sk \in \{0,1\}^k$ and inserts the tuple (ID_i, ID_j, T_i, sk) into L_S.

$Test(\prod_{i,j}^{s})$: Upon receiving the query, $\mathcal{C}$ checks whether $\prod_{i,j}^{s}$ and $\prod_{I,J}^{S}$ are equal. If there are not equal, $\mathcal{C}$ terminates the simulation; else $\mathcal{C}$ chooses a random number $\xi \in \{0,1\}^k$ and sends it to $\mathcal{A}1$.

When the adversary carry out the forging attack successfully, he/she must make the H_2 query with $Z_1 = e(t_I P_{pub} + D_I, Q_J)$ and $Z_2 = (t_I + x_I)P_J$, where $T_I = U$ is the outgoing message of the $Test$ session. Then, $\mathcal{C}$ chooses a tuple of L_{H_2} randomly, computes $\overline{Z}_1 = \dfrac{Z_1}{e(D_I, Q_J)} = e(t_I P_{pub}, Q_J) = e(P, P)^{uzw}$ and outputs $\overline{Z}_1$ as the solution to $bdh(U,Z,W)$. It is easy to say $Adv_{\mathcal{C}}^{GBCDH}(k) \geq \frac{1}{n_0 n_1^2 n_2} Adv_{\mathcal{A}1}(k)$, where $Adv_{\mathcal{C}}^{GBCDH}(k)$ denotes the advantage that $\mathcal{C}$ solves the GBCDH problem. Since $Adv_{\mathcal{A}1}(k)$ is not negligible, then $Adv_{\mathcal{C}}^{GBCDH}(k)$ is also non-negligible. The conclusion is against the GBCDH assumption.

CASE 1.2: The adversary $\mathcal{A}1$ gets the Test session's ephemeral private key and does not get I 's partial private key.

Given a GBCDH problem case ($U = uP$, $Z = zP$, $W = wP$, $\mathcal{O}_{dbdhp}$), we will use $\mathcal{A}1$ to build an algorithm to overcome the problem, where $\mathcal{O}_{dbdhp}$ denotes a decision oracle. For input (uP, zP, wP, X), the decision oracle outputs 1 if the equation $bcdh(uP, zP, wP) = X$ holds; otherwise, it outputs 0. $\mathcal{C}$ sets $Z = zP$ as the public key P_{pub}, sends the parameters $params = \{G_1, G_2, q, P, e, P_{pub} = Z, H_1, H_2\}$ to $\mathcal{A}1$. $\mathcal{C}$ simulates $\mathcal{A}1$'s $H_1(ID_i)$, $RevealMasterKey$, $RevealSecretValue(i)$, $ReplacePublicKey(i, pk)$, $RevealSessionKey(\prod_{i,j}^{s})$ and $Test(\prod_{i,j}^{s})$ queries as described in above case. He replies other queries like follows.

$Create(i)$: If $i = I$, $\mathcal{C}$ generates a random number $x_i \in Z_n^*$, computes $P_i = x_i P$, sets $Q_i = H_1(ID_i) \leftarrow U$, $D_i \leftarrow \perp$ and inserts ($ID_i, Q_i, \perp, x_i, P_i$) and (ID_i, Q_i) in L_C and L_{H_1} separately; otherwise, he answers the query like he does in the above case.

$H_2(ID_i, ID_j, T_i, Z_1, Z_2, h)$: Upon receiving the query, $\mathcal{C}$ checks whether the query is of the form ($ID_I, ID_J, T_I, Z_1, Z_2$) or ($ID_J, ID_I, T_J, Z_1, Z_2$). If it is not with the two forms, he/she answers the query like he/she does in the above case; otherwise, $\mathcal{C}$ checks whether either of ($ID_I, ID_J, T_I, Z_1, Z_2, h$) or ($ID_J, ID_I, T_J, Z_1, Z_2, h$) is included in L_{H_2}. If one or them is included, $\mathcal{C}$ returns h ; else $\mathcal{C}$ generates a random number $h \in \{0,1\}^k$, inserts ($ID_I, ID_J, T_I, Z_1, Z_2, h$) or ($ID_J, ID_I, T_J, Z_1, Z_2, h$) into L_{H_2} and return h to $\mathcal{C}$.

$RevealPartialPrivateKey(i)$: Upon receiving the query, $\mathcal{C}$ checks whether either of equations $i = I$ or $i = J$ is equal. If they are equal, $\mathcal{C}$ terminates the simulation; else $\mathcal{C}$ loops up L_C for the tuple (ID_i, Q_i, D_i, x_i, P_i) and returns D_i to $\mathcal{A}1$.

$RevealEphemeralKey(\prod_{i,j}^{s})$: Upon receiving the query, $\mathcal{C}$ returns the corresponding ephemeral private key to $\mathcal{A}1$.

$Send(\prod_{i,j}^{s}, m)$: Upon receiving the query, $\mathcal{C}$ checks whether $\prod_{i,j}^{s}$ and $\prod_{I,J}^{S}$ are equal. If they are equal, $\mathcal{C}$ returns $T_i = t_iP$ to $\mathcal{A}1$; otherwise, $\mathcal{C}$ checks whether either of the equations $i = I$ or $j = J$ holds. If neither of them holds, $\mathcal{C}$ answers the query like he does in the above case; otherwise, $\mathcal{C}$ checks the equation $m = \lambda$. If it holds, $\mathcal{C}$ chooses $t_i \in Z_n$ and returns $T_i = t_iP$ to $\mathcal{A}1$; else $\mathcal{C}$ randomly selects a number $sk \in \{0,1\}^k$ and adds (ID_I, ID_J, T_I, sk) or (ID_J, ID_I, T_J, sk) into the L_S .

When the adversary carries out the forging attack successfully, he/she must make the H_2 query with $Z_1 = e(t_IP_{pub} + D_I, Q_J)$ and $Z_2 = (t_I + x_I)P_J$, where $T_I = U$ is the outgoing message of the $Test$ session. Then, $\mathcal{C}$ chooses a tuple of L_{H_2} randomly, computes $\overline{Z}_1 = \dfrac{Z_1}{e(t_IP_{pub}, Q_J)} = e(D_I, Q_J) = e(P,P)^{uzw}$ and outputs $\overline{Z}_1$ as the solution to $bdh(U,Z,W)$. It is easy to say $Adv_{\mathcal{C}}^{GBCDH}(k) \geq \dfrac{1}{n_0n_1^2n_2}Adv_{\mathcal{A}1}(k)$,

where $Adv_{\mathcal{C}}^{GBCDH}(k)$ denotes the advantage that $\mathcal{C}$ solves the GBCDH problem. Since $Adv_{\mathcal{A}1}(k)$ is not negligible, then $Adv_{\mathcal{C}}^{GBCDH}(k)$ is also non-negligible. This conclusion is against the GBCDH assumption.

- *Responder session*

When the $Test$ is a responder session, $\mathcal{A}1$ cannot gets I and J's partial private key. Then $\mathcal{C}$ is able to reply $\mathcal{A}1$'s queries like he/she does in CASE 1.2. We could process this case like we do in CASE 1.2.

- *The analysis of CASE 2*

The $Test$ session $\prod_{I,J}^S$ has a matching session held by another honest party J. Like CASE 1, the $\prod_{I,J}^S$ may be one of two sessions which are initiator and responder sessions. By the definition of freshness, there are four ways for the adversary $\mathcal{A}1$ to conduct the attacks.

- *Initiator session*

CASE 2.1: $\mathcal{A}1$ gets I 's partial private key but does not get the Test session's ephemeral private key. We could process this case like we do in CASE 1.1.

CASE 2.2: $\mathcal{A}1$ gets the Test session's ephemeral private key but not does get I 's partial private key. We could process this case like we do in CASE 1.2.

Responder session

CASE 2.3: The adversary $\mathcal{A}1$ gets J 's partial private key and does not get the matching session's ephemeral private key. We could process this case like we do in CASE 1.1.

CASE 2.4: The adversary $\mathcal{A}1$ gets the matching session's ephemeral private key but does not get J's partial private key. We could process this case like we do in CASE 2.2.

From the above analysis, we could conclude that the advantage of a Type 1 adversary against our scheme is negligible if the GBCDH problem is intractable.

Lemma 3. Suppose that the GBCDH problem is intractable, our proposed scheme is secure against the Type II adversary.

The proof of Lemma 3 is very similar to Lemma 2, we will not give the detail here. Based on three lemmas above, we could get the theorem for our proposed scheme's security.

Theorem 1. Suppose the GBCDH problem is intractable, then our proposed one-pass CLAKA scheme is provably secure in the eCK model.

COMPARISON WITH PREVIOUS SCHEMES

As a matter of convenience, some notations are defined:

- T_{par} : The time of performing a pairings operation of point.
- T_{mul} : The time of performing a scalar multiplication operation of point.
- $T_{\exp}$: The time of performing an addition operation of point.

We will give efficiency comparison about our proposed scheme and Chen *et al.*'s scheme (Chen *et al.*, 2009). Table 1 shows two schemes' efficiency comparison. Scalar multiplication operations and exponentiation operations are much cheaper than pairing operation. Note that according to Barreto *et al.*'s work (Barreto *et al.*, 2002), the effort to evaluate one pairing operation is approximately three times than the effort of computing scalar multiplications. Therefore the proposed scheme performs better. Moreover, it is easy to verify that Chen *et al.*'s scheme is not secure in the eCK model. Consequently, the presented scheme is more suitable for practical applications.

Table 1: Comparisons between two schemes

	Computational cost		**Provablly secure**
	Sender	**Receiver**	
Chen *et al*'s scheme[14]	$2T_{par}+2T_{mul}+2T_{\exp}$	$T_{par}+3T_{mul}$	No
Our scheme	$T_{par}+3T_{mul}$	$T_{par}+T_{mul}$	Yes

CONCLUSION

We first introduce a security model for the one-pass certificateless one-pass authenticated key agreement scheme, and then propose the new one-pass CLAKA scheme, which is provable security in the eCK model. Comparisons with related schemes show that our scheme can provide better efficiency and security. Therefore, our proposed scheme is more practical than others.

REFERENCES

Al-Riyami, S. & Paterson, K.G. 2003. Certificateless public key cryptography, Proceedings of ASIACRYPT 2003, Taipei, Taiwan.

Barreto, P., Kim, H., Lynn, B. & Scott, M. 2002. Efficient algorithms for pairing-based cryptosystems. Advances in Cryptology – Crypto 2002. California, USA.

Bellare, M. & Rogaway, P. 1993. Entity authentication and key distribution. In: Proceedings of the CRYPTO 1993. California, USA.

Bellare, M. & Rogaway, P. 1995. Provably secure session key distribution: the three party case. Proceedings of the 27th ACM symposium on the theory of computing. Massachusetts, USA.

Boneh, D. & Franklin, M. 2003. Identity-based encryption from the Weil pairing. SIAM Journal on Computing **32**:586–615.

Canetti, R. & Krawczyk, H. 2001. nalysis of key-exchange protocols and their use for building secure channels. Proceedings of the EUROCRYPT 2001. Innsbruck, Austria.

Chen, W., Zhang, L. & Qin B. 2009. Certificateless One-Way Authenticated Two-Party Key Agreement Protocol , Fifth International Conference on Information Assurance and Security, Xian China.

Choo, K., Boyd, C. & Hitchcock Y. 2005. Examining indistinguishability-based proof models for key establishment protocols. Proceedings of the ASIACRYPT 2005. Chennai, India.

Galbraith, S., Harrison, K. & Soldera, D. 2002. Implementing the Tate pairing, Proceedings of the 5th International Symposium on Algorithmic Number Theory – ANTS, Sydney, Australia.

He, D., Chen, Y., Chen, J. & Zhang, R. 2011. A new two-round certificateless authenticated key agreement protocol without bilinear pairings. Mathematical and Computer Modelling **54**: 3143–3152.

He, D., Padhye, S. & Chen, J. 2012. An efficient certificateless two-party authenticated key agreement protocol. Computers & Mathematics with Applications **64**(6): 1914–1926.

Huang, H. & Cao, Z. 2011. IDOAKE: strongly secure ID-based one-pass authenticated key exchange protocol. Security and Communication Networks **4**(10): 1153–1161.

LaMacchia, B., Lauter, K. & Mityagin, A. 2007. Stronger security of authenticated key exchange. Proceedings of the ProvSection 2007. Alberta, CANADA.

Lippold, G., Boyd, C. & Nieto, J. 2009. Strongly secure certificateless key agreement. Pairing 2009, 206-230.

Shamir, A. 1984. Identity-based cryptosystems and signature protocols. Proc. CRYPTO1984. California, USA.

Submitted : 20/09/2013
Revised : 11/04/2014
Accepted : 09/06/2014

مخطط اتفاقية مفتاح مصادقة أحادي المرور ذو حماية قوية لا تعتمد على الشهادات

***بوجون هوانغ، **هانغ تو**

*قسم الرياضيات والإحصاء – جامعة ووهان – ووهان 430072 – هيوباي – الصين

*قسم الحاسوب – جامعة ووهان – ووهان 430072 – هيوباي – الصين

بريد إلكتروني: tuhang2013@193.com

خلاصة

بما أنه يمكن لمفتاح التشفير العام بدون شهادات (CLPKC) أن يحل مشكلة الضمان الرئيسي، تم في الآونة الأخيرة تناول دراسته أكثر وأكثر. اتفاقية مفتاح المصادقة بدون شهادات (CLAKA)، باعتبارها جزءاً مهماً من CLPKC، تجتذب أيضاً اهتماماً كبيراً. وحتى الآن، تم اقتراح العديد من أنواع CLAKA أحادية المرور والآمنة بصورة مبرهنة. ولكن في قليل منها فقط ت تقديم برهان الحماية بنموذج رسمي. في هذه الورقة، نقترح مخطط CLAKA أحادي المرور مبتدع ونبين أنه آمن بصورة مبرهنة تحت افتراض الفجوة شبه خطية الحسابية الخاص بديفي–هيلمان (GCDH) في نموذج تمديد كانيتي–كراوزيك (eCK). وبقدر ما نعلم، إن مخططنا هو أول مخطط CLAKA آمن بصورة مبرهنة في نموذج eCK.

كلمات مفتاحية: اتفاقية مفتاح المصادقة، التشفير بدون شهادات، نموذج eCK، أحادي المرور، حماية مبرهنة.