A Strongly Secure Certificateless Digital Signature Scheme in The Random Oracle Model

Mohammed Hassouna[1], Eihab Bashier[2,3], and Bazara Barry[3]
(Corresponding author: Eihab Bashier)
Computer Studies, National Ribat University[1]
P.O. Box 55, Khartoum, Sudan
Department of Mathematics, Physics and Statistics, Arts and Sciences, Qatar University[2]
P.O. Box 2713, Doha, Qatar
Department of Computer Science, Mathematical Sciences, University of Khartoum[3]
P.O. Box, 321, Khartoum, Sudan
(Email: ebashier@qu.edu.qa)
(Received Sep. 20, 2015; revised and accepted Nov. 16 & Dec. 7, 2015)

Abstract

The main purpose of this paper is to provide a security proof for the certificateless digital signature scheme found in [Hassouna, Bashier, and Barry, A short certificateless digital signature scheme, *International Conference of Digital Information Processing*, Data Mining and Wireless Communications, 2015, pp. 120–127] in the random oracle model. Two types of attacks are considered: The first type can be carried out by an outsider attacker and referred to as Type I, whereas the second one can be carried out by a malicious KGC and referred to as Type II. The possible oracles for each of the two types of attacks are discussed, and hence, the security of the proposed digital signature scheme was proved in the random oracle model.

Keywords: Certificateless cryptography, certificateless signature, pairings in elliptic curves, public-key replacement attack

1 Introduction

In 2003, Al-Riyami and Paterson [1] introduced the concept of Certificateless Public Key Cryptography (CL-PKC) to overcome the key escrow limitation of the identity-based public key cryptography (ID-PKC). In CL-PKC a trusted third party called Key Generation Center (KGC) supplies a user with a partial private key. Then, the user combines the partial private key with a secret value (that is unknown to the KGC) to obtain his/her full private key. In this way, the KGC does not know the user's private key. Then the user combines his/her secret value with the KGC's public parameters to compute his/her public key.

Al-Riyami and Paterson [1] proved that their certificateless encryption scheme is secure against fully-adaptive chosen ciphertext attack (IND-CCA). They also proposed a certificateless digital signature scheme along with certificateless key agreement protocol and hierarchal certificateless encryption scheme (HCL-PKE). Even after using the binding technique, the scheme does not have trust level 3 according to Girault's [11] definition.

Since Al-Riyami and Paterson original CL-PKC scheme was proposed [1], many certificateless cryptography schemes have appeared in literature. These schemes include the uses of certificateless encryption [7, 14], certificateless signatures [16, 19, 20] and certificateless signcryption [15, 17, 18].

Hassouna et al. [12] introduced an integrated certificateless public key infrastructure model. That model used a different key generation technique with a different binding method from Al-Riyami and Paterson [1] model. The integrated certificateless public key infrastructure model provided many practical features, like two-factor private key authentication, private key recovery, private key portability and private key archiving. These features were provided because Hassouna et al. [12] separated the process of generating private key from the process of generating the public key.

The binding technique that was proposed by Hassouna et al. [12] provided a more robust way to link the user's identity with his/her public/private keys. Furthermore, the binding technique raised very important and non-mentioned feature: it made the CL-PKC resistant to the public key replacement attack that can be done by the KGC or any adversary in case of sending the user's partial private key in an insecure channel. This was because the user's full private key is generated from a different secret value that used in the user's public key calculation.

In 2015, Hassouna et al. [13] extended their origi-

nal model that was proposed in [12], by proposing a new strong and efficient certificateless digital signature scheme. They verified its consistency and efficiency.

Furthermore, Hassouna et al. [13], proposed a new different security model that was suitable for their proposed signature scheme. In their proposed security model, the definitions of Type I and Type II adversaries had become different from the definitions introduced by Xiong et al. in [19]. However, Hassouna et al. [13] stated that their signature scheme was provably secure against their proposed security model in the Random Oracle Model (ROM), but no security proof was provided.

The main purpose of this paper is to prove the security of Hassouna et al. [13] certificateless digital signature scheme against their proposed security model. The security scheme that was introduced in [13] was based on two mathematical hard problems, namely the Computational Diffie-Hellman Problem (CDHP) and the Bilinear Diffie-Hellman Problem (BDHP) in addition to using a set of predefined hash functions. Therefore, we will prove its security in the Random Oracle Model (ROM).

The rest of this paper is organized as follows. Section 2 gives backgrounds about pairing in elliptic curves and its related cryptographic primitives, Hassouna et al. [13] digital signature scheme and their security model are in Section 3. In Section 4, we state the security proof of Hassouna et al.'s [13] signature scheme. Finally, Section 5 concludes the paper.

2 Backgrounds

In this section, we give backgrounds about pairing in eliptic curves and its related cryptography primitives that are used in this paper. Here, G_1 denotes an additive group of prime order q (particulary elliptic curve group) and G_2 a multiplicative group of the same order. We let P denote a generator of G_1.

Definition 1. Elliptic Curve Computational Diffie-Hellman Problem (ECDHP)*: Given (P, aP, bP) in G_1 where $a, b \in \mathbb{Z}_q^*$, compute abP.*

2.1 Pairing in Elliptic Curve

A pairing is a map $e : G_1 \times G_1 \longrightarrow G_2$ with the following properties:

1) The map e is bilinear: given $Q, W, Z \in G_1$, we have: $e(Q, W + Z) = e(Q, W) \cdot e(Q, Z)$ and $e(Q + W, Z) = e(Q, Z) \cdot e(W, Z)$.
   Consequently, for any $a, b \in Z_q$, we have $e(aQ, bW) = e(Q, W)^{ab} = e(abQ, W)$, etc.
2) The map e is non-degenerate: $e(P, P) \neq 1_{G_2}$.
3) The map e is efficiently computable.

Definition 2. BDH Parameter Generator: *As in [4], a randomized algorithm $\mathcal{G}$ is a BDH parameter generator if $\mathcal{G}$:*

1) *takes security parameter $k \geq 1$,*
2) *runs in polynomial time in k, and*
3) *outputs the description of groups G_1, G_2 of prime order q and a pairing $e : G_1 \times G_1 \longrightarrow G_2$.*

Formally, the output of the algorithm $\mathcal{G}(1^k)$ is (G_1, G_2, e). Typically, the map e will be derived from either the Weil or Tate pairing on an elliptic curve over a finite field.

We refer to [2, 3, 4, 5, 6, 8, 9, 10] for a more comprehensive description of how these groups, pairings and other parameters should be selected in practice for efficiency and security.

Definition 3. Bilinear Diffie-Hellman Problem (BDHP): *Let G_1, G_2, P and e be as above. The BDHP in G_1, G_2, e is as follows: Given P, aP, bP, cP with uniformly random choices of $a, b, c \in Z_q^*$, compute $e(P, P)^{abc} \in G_2$. An algorithm A has advantage ϵ in solving the BDHP in G_1, G_2, e if:* $\mathbf{Pr}[A(P, aP, bP, cP) = e(P, P)^{abc}] = \epsilon$.

Here, the probability is measured over the random choices of $a, b, c \in Z_q^*$ and the random bits of A.

3 Hassouna et al's Certificateless Digital Signature Scheme

In this section, we state the certificateless digital signature scheme that was proposed by Hassouna et al. [13].

- **Setup (running by the KGC):** The KGC chooses a secret parameter k to generate G_1, G_2, P, e where G_1 and G_2 are two groups of a prime order q, P is a generator of G_1 and $e : G_1 \times G_1 \longrightarrow G_2$ is a bilinear map. The KGC randomly generates the system's master key $s \in \mathbb{Z}_q^*$ and computes the system public key $P_{pub} = sP$. Then the KGC chooses cryptographic hash functions H_1 and H_2, where $H_1 : \{0,1\}^* \longrightarrow G_1$ (Map-to-Point hash function), and $H_2 : \{0,1\}^n \longrightarrow \mathbb{Z}_q^*$ (any cryptographic hash function like MD5 or SHA family). Finally, the KGC publishes the system parameters $params = < G_1, G_2, e, P, P_{pub}, H_1, H_2, n >$, while the secret master-key is saved and secured by the KGC.
- **Set-Secret-Value (running by the user):** The user m with the identity ID_m downloads the system parameters, picks two random secret values $x_m, x'_m \in \mathbb{Z}_q^*$. Then, user m computes $X_m = x'_m P$ and sends X_m to the KGC. The proposed scheme enforces the user to choose a strong password $pass$, the system at client hashes the password to be $z_m = H_2(pass)$, multiplies the base point P by the hashed password to be $z_m P$, uses the hashed value z_m as key encrypt the secret value x_m and generates the Password-based Encryption Code(PEC) as $PEC_{z_m}(x_m)$, sends copy of it to the KGC's public directory and stores copy of it along with the point $z_m P$ locally.

- **Partial-Private-Key-Extract (running by the KGC):** On receiving X_m computed by user m with identity ID_m, the KGC first computes $Q_m = H_1(ID_m)$, then it generates the partial private key of user m as $D_m = sQ_m$.

- **Set-Public-Key (running by the user):** The user m with identity ID_m computes $Q_m = H_1(ID_m)$, $Y_m = x'_m Q_m$ and sets $< X_m, Y_m >$ as his/her long-term public key P_m. Finally, user m sends Y_m to the KGC.

- **Set-Private-Key:** User m's private key is $S_m = (x_m + z_m)D_m = (x_m + z_m)sQ_m = (x_m + z_m)sH_1(ID_m)$. Also, the user generates the secret term $Z_m = x_m P$.

- **Sign:** The user generates the signature of the message M using his/her secret terms $\{x_m, Z_m\}$ as follows:

  1) The signer generates big random integer $a \in G_2^*$.
  2) The signer calculates $MP_m = H_1(m) \in G_1^*$.
  3) The signer calculates $MP_{1m} = ax_m MP_m \in G_1^*$.
  4) The signer calculates $s_m = e(MP_m, Z_m)^{ax'_m} = e(MP_m, P)^{ax_m x'_m}$.
  5) The signer sends $\sigma = (m, MP_{1m}, s_m)$ as the signature.

- **Verify:** After receiving the signature $\sigma = (m, MP_{1m}, s_m)$, the verifier uses user m's public key $< X_m, Y_m >$ to verify the signature as follows:

  1) The verifier checks whether $e(X_m, Q_m) = e(Y_m, P)$. If it holds then user m's public key is authentic, otherwise the signature is rejected.
  2) The verifier calculates $MP'_m = H_1(m) \in G_1^*$.
  3) If $MP_{1m} = MP'_m$ or $s_m = e(H_1(m), X_m)$ then the verifier rejects the signature.
  4) Otherwise, the verifier calculates $r_m = e(MP_{1m}, X_m)$.
  5) The verifier accepts the signature iff $r_m = s_m$, otherwise he/she rejects the signature.

3.1 Hassouna et al.'s Security Model

In Hassouna et al. [13] two types of adversaries were considered: Type I and Type II adversaries according to the term Z_m as follows:

1) **Type I Adversary**

   A_I which is allowed to replace the term Z_m by a valid value of his/her choice, but is not allowed to replace users' public keys and has not access to the master secret key s.

2) **Type II Adversary**

   A_{II} which has access to the master secret key s, is allowed to replace users public keys with valid values of his/her choice, but is not allowed to replace the term Z_m.

Type I adversary represents outsider attacker and Type II attacker is a malicious KGC. Two games are defined as follows.

- **Game I.** The first game is performed between a challenger C and a Type I adversary A_I as follows.

  1) Setup. The challenger C runs Setup algorithm and generates a master secret key msk and public system parameters params. C gives params to A_I, while keeping msk secret.
  2) Queries. A_I may adaptively issue the following queries to C.
     - Partial private key queries: Upon receiving a partial private key query for an identity ID, C returns the partial private key with respect to identity ID to A_I.
     - Public key queries: Given an identity ID, C returns the corresponding public key terms $< X_A, Y_A >$ to A_I.
     - Replace public key: Given an identity ID with a pair of values (x'^1_{ID}, pk^1_{ID}) which are chosen by A_I, C updates the user ID original secret/public key (x'_{ID}, pk_{ID}) to the new (x'^1_{ID}, pk^1_{ID}).
     - $Z-key$ Extraction queries: This is a new oracle in this security model, given an identity ID, C returns the corresponding $Z-key$ value Z_{ID}.
     - Replace $Z-key$: This is a new oracle in this security model which on input (ID, x^1_{ID}, Z^1_{ID}), C replaces the user ID original term (x_{ID}, Z_{ID}) by (x^1_{ID}, Z^1_{ID}).
     - Private key queries. Upon receiving a private key query for an identity ID, C returns the corresponding private key sk_{ID} to A_I.
     - Sign queries: Proceeding adaptively, A_I can request signatures on any messages m with respect to an identity ID. C computes signature, and returns to A_I.
  3) Forgery. Eventually, A_I outputs a certificateless signature σ^* on message m^* corresponding to public key pk_{ID^*} for an identity ID^*. A_I wins the game if Verify($params$, ID^*, pk_{ID^*}, m^*, σ^*) = 1 and the following conditions hold:
     - A_I has never been queried Partial private key oracle on ID^*.
     - A_I never replaced the user ID^*'s public key.
     - A_I has never been queried Private key oracle on ID^*.

– A_I has never been queried Sign oracle on (ID^*, m^*).

The success probability of A_I is defined as the probability that it wins in Game I.

- **Game II.** This game is performed between a challenger C and a Type II adversary A_{II} as follows.

  1) Setup. The challenger C runs A_{II} on k and a special Setup, and returns a master secret key msk and public system parameters params to A_{II}.
  2) Queries. In this phase, A_{II} can adaptively access the Private key oracle, Public key oracle, Replace public key oracle, $Z-key$ oracle, Replace $Z-key$ oracle and Sign oracle, which are the same as that in Game I.
  3) Forgery. A_{II} outputs a certificateless signature σ^* on message m^* corresponding to public key pk_{ID^*} for an identity ID^*. A_{II} wins the game if Verify$(params, ID^*, pk_{ID^*}, m^*, \sigma^*) = 1$ and the following conditions hold:
     – A_{II} has never been queried Private key oracle on ID^*.
     – A_{II} has never been queried Replace $Z-key$ oracle on ID^*.
     – A_{II} has never been queried Signature oracle on (ID^*, m^*).

  The success probability of A_{II} is defined as the probability that it wins in Game II.

Accordingly, the security definitions of any certificateless digital signature scheme in the Random Oracle Model (ROM) can be given as follows.

Definition 4. *A certificateless signature scheme is* $(t, q_H, q_e, q_z, q_{sk}, q_{pk}, q_s, \epsilon)$*-existentially unforgeable against Type I adversary under adaptively chosen message attacks if no t-time adversary* A_I*, making at most* q_H *to the random oracles,* q_e *partial private key queries,* q_z *to the* $Z-key$ *queries,* q_{sk} *private key queries,* q_{pk} *public key queries and* q_s *signature queries, have a success probability at least* ϵ *in Game I.*

Definition 5. *A certificateless signature scheme is* $(t, q_H, q_z, q_{sk}, q_{pk}, q_s, \epsilon)$*-existentially unforgeable against Type II adversary under adaptively chosen message attacks if no t-time adversary* A_{II}*, making at most* q_H *to the random oracles,* q_z *to the* $Z-key$ *queries,* q_{sk} *private key queries,* q_{pk} *public key queries and* q_s *signature queries, have a success probability at least* ϵ *in Game II.*

Definition 6. *A certificateless signature scheme is existentially unforgeable under adaptively chosen message attack (EUF-CMA), if the success probability of any polynomially bounded adversary in the above two games is negligible.*

4 Security Analysis

The main interesting security feature in the Hassouna et al.'s [13] signature scheme, is that its security does not depends on the security of the KGC, because the master secret of the KGC is not involved directly in the signature generation/verification. This way, the such certificateless signature schemes can enjoy the same security feature as the traditional signature scheme that are based on PKI.

This is because in the PKI context, the private key of the CA does not impact the security of the signatures that are generated by the users, and that is because the users' private keys are not connected directly with the public/private key of the CA, and the public/private key of the CA is just used to ensure the authenticity of the users by signing the users' certificates.

Furthermore, the security of Hassouna et al.'s [13] signature scheme depends on the the term Z_m which is considered as one of the private keys of the user m. The term Z_m links the user's public/private keys and any compromise in the user's public key leads to compromise in term Z_m and hence in the signature scheme.

Thinking this way, the certificateless schemes can have better chances in securing real applications, because this approach will reduce the risk of trusting the KGC without decreasing the features of the certificateless cryptography as concept, i.e eliminating the certificates and some of its management problems and also eliminating the risk of trust on the KGC.

Now we state the general definition of the security of Hassouna et al.'s [13] signature scheme in the random oracle model (ROM) given that the Adversary A has access to the oracles that have been described later.

Theorem 1. Hassouna et al.'s [13] short CLS scheme is secure against existential forgery under adaptively chosen message attacks in the random oracle model with the assumptions that the ECDHP (Elliptic Curve Computational Diffie-Hellman Problem) and BDHP (Bilinear Diffie-Hellman Problem) in G_1 are intractable.

The proof of Theorem 1 is based on the following two lemmas.

Lemma 1. Let A_I be a Type I Adversary in Game I that (t, ϵ)-breaks the proposed CLS scheme. Assume that A_I makes q_H queries to a random oracle H_1, q_e queries to the partial-private-key extraction oracle, q_z queries to the $Z-key$ extraction oracle, q_{sk} queries to the private-key extraction oracle, q_{pk} queries to the public-key request oracle and q_s queries to signing oracle and can replace $Z-key$ of any user. A_I cannot replace the public key of the challenged user and does not have the master secret. Then, there exists a (ϵ', t')-algorithm C that is able to solve the BDHP problem in group G_1, G_2 where $\epsilon' < \epsilon(\frac{q_H-1}{q_H})^{q_e+q_{sk}+q_s}$, $t' < t + (q_s+q_z)t_{sm} + q_s t_p$, t_{sm} denotes the cost of the scalar multiplication in G_1 and t_p the cost of calculating one bilinear pairing operation.

Lemma 2. Let A_{II} be a Type II Adversary in Game II that (t, ϵ)-breaks the proposed CLS scheme. Assume that A_{II} makes q_H queries to random oracles H_1, q_z queries to the $Z - key$ extraction oracle, q_{sk} queries to the private-key extraction oracle, q_{pk} queries to the public-key request oracle, q_s queries to signing oracle and can replace the public key of any user. A_{II} cannot replace $Z - key$ of the challenged user but have the master secret. Then, there exists a (ϵ', t')-algorithm C that is able to solve the ECDHP problem in group G_1 where $\epsilon' < \epsilon(\frac{q_H - 1}{q_H})^{q_{sk}+q_s}$ and $t' < t + (q_s + q_z)t_{sm} + q_s t_p$.

4.1 Proof of Lemma 1

Suppose that C is given a challenge: given $Z_m = x_m P$, abP and $X_m = r_m P$ compute $e(P, P)^{abx_m r_m}$ after interacting with A_I. Now C and A_I play the role of the challenger and the adversary respectively. C will interact with A_I as follows:

- **Setup:** C runs algorithm Setup, chooses a generator P and sets $P_{pub} = sP$, where s is the system master key, which is unknown to C. C picks an identity ID^* at random as the challenged ID in this game, and gives $params =< P, P_{pub}, H_1 >$ to A_I as the public parameters. For simplicity, we assume that for any ID_i, A_I queries H_1 before ID_i is used as an input of any query Public-key Extraction, Partial-private-key Extraction, Private-key Extraction and Signing oracles.

- **H_1-Queries:** C maintains a hash list H_1^{list} of tuple (ID_i, Q_i) as explained below. The list is initially empty. When A_I makes a hash oracle query on ID_i, if the query ID_i has already appeared on the H_1^{list}, then the previously defined value is returned. Otherwise, C chooses a random integer $a \in \mathbb{Z}_q^*$ and sets $Q_i = aP$, inserts the pair (ID_i, Q_i) in the list H_1^{list} and returns it to the adversary A_I.

- **Partial-private-key Extraction Queries:** C maintains a list E^{list} of tuple (ID_i, Q_i, D_i) which is initially empty. For any given identity ID_i, C recovers the corresponding tuple (ID_i, Q_i) from the list H_1^{list}, if $ID_i \neq ID^*$ then sets $D_i = sQ_i$ and returns it to the adversary A_I and adds (ID_i, Q_i, D_i) to the list E^{list}. Otherwise($ID_i = ID^*$), C aborts and outputs "failure"(denote this event by E_1).

- **Public-key Extraction Queries:** C maintains a list pk^{list} of tuple (ID_i, Q_i, r_i, pk_i) which is initially empty. When A_I queries on input ID_i, C checks whether pk^{list} contains a tuple for this input. If it does, the previously defined value is returned. Otherwise, C recovers the corresponding tuple (ID_i, Q_i) from the list H_1^{list} and picks a random value $r_i \in \mathbb{Z}_q^*$, computes $pk_i =< X_i, Y_i >=< r_i P, r_i Q_i >$ and returns pk_i. Then, adds (ID_i, Q_i, r_i, pk_i) to the list pk^{list}.

- **$Z - key$ Extraction Queries:** C maintains a list Z^{list} of tuple (ID_i, Z_i) which is initially empty. if Z^{list} already contains the pair (ID_i, Z_i), then it returns it to the adversary A_I, otherwise C calls the oracle Private Key Extraction on identity ID_i and gets the value Z_i, gives it to the adversary A_I and inserts it in the list Z^{list}.

- **Private-key Extraction Queries:** C maintains the list sk^{list} for query on input ID_i. If $ID_i = ID^*$, C stops and returns "failure" (denote the event by E_2). Otherwise, C picks a random number $x_i \in \mathbb{Z}_q^*$ and performs as follows:
  - If the E^{list} and the pk^{list} contain the corresponding tuple (ID_i, Q_i, D_i) and the tuple (ID_i, Q_i, r_i, pk_i) respectively, C sets $sk_i = x_i D_i$, $Z_i = x_i P$, returns (ID_i, x_i, sk_i, Z_i) to A_I and adds them to the list sk^{list}.
  - Otherwise, C makes a partial-private-key Extraction query and a Public-key Extraction query on ID_i, then simulates as the above process, sends (ID_i, x_i, sk_i, Z_i) to A_I and adds them to the list sk^{list}.

- **$Z - key$ Replacement (ID_i, x_i', Z_i'):** When A_I queries on input (ID_i, x_i', Z_i'), C checks whether the tuple (ID_i, Z_i) is contained in the Z^{list} . If it is, C sets $Z_i = Z_i'$ and adds (ID_i, Z_i') to the Z^{list}. Here, we assume that C can obtain a replacing secret value x_i' corresponding to the replaced $Z - key = Z_i'$ from A_I. Otherwise, C executes Private Key extraction to generate (ID_i, sk_i, Z_i), then sets $Z_i = x_i' P$ and inserts it in the list Z^{list}.

- **Signing Queries:** When a signing query (ID_i, m_j) is coming, C acts as follows:
  - If $ID_i = ID^*$, C stops and returns "failure status" (denote the event by E_3).
  - Otherwise, C recovers the tuple (ID_i, x_i, sk_i, Z_i) from the sk^{list} and the tuple (ID_i, Q_i, pk_i) from the pk^{list} and the tuple (m_j, MP) from H_1^{list}.
  - Picks a random integer $a \in \mathbb{Z}_q^*$.
  - Computes $MP_1 = ax_i MP$.
  - Computes $s_i = e(MP, Z_i)^{ar_i}$ and (MP_1, s_i) is the signature for the identity ID_i on the message m_j. C returns (MP_1, s_i) to A_I as response to the signing oracle.

Finally, A_I stops and outputs a signature $\sigma = (V^*, S^*)$ on the message m^* for the identity ID^*, which satisfies the equation $\text{Verify}(m^*, ID^*, pk^*, S^*) = 1$. C recovers the tuple (ID^*, Q^*, pk) from pk^{list}, the tuple (ID^*, x^*, Z^*), (m^*, MP^*) from Z^{list} and H_1^{list} picks a random integer $a^* \in \mathbb{Z}_q^*$. Then, we have $e(V^*, X_i) = e(a^* x^* b^* P, rP) = S^*$, that is: $e(P, P)^{a^* x^* r b^*} = S^*$.

Hence C can successfully compute and output $e(P,P)^{a^*r} = S^{*1/(x^*b^*)}$ as solution to the A_I's challenge. So, C breaks the BDHP problem in G_1, G_2. Now we analyze the advantage of C in this game.

Note that the responses to A_I's H_1 queries are indistinguishable from the real life. Since each response is uniformly random and independently distributed in G_1^*. The responses of queries H_1 provided for A_I are all valid. The responses of Partial-private-key extraction queries, Private-key extraction queries and signing queries are valid if the events E_1, E_2 and E_3 never happen. Furthermore, if A_I forges a valid signature and events E_1, E_2 and E_3 do not happen, then C can solve the BDHP problem. Therefore, if none of the events E_1, E_2 and E_3 happens, C can solve the BDHP problem successfully. Now, Let's bound the probability for these events. From the description above we have: $Pr(\neg E_1 \wedge \neg E_2 \wedge \neg E_3) = (\frac{q_H-1}{q_H})^{q_e+q_{sk}+q_s}$.

In conclusion, challenger's C advantage is $\epsilon' < \epsilon(\frac{q_H-1}{q_H})^{q_e+q_{sk}+q_s}$ with the running time cost as $t' < t + (q_s+q_z)t_{sm} + q_s t_p$, where t_{sm} denotes the cost of the scalar multiplication in G_1 and t_p the cost of calculating one bilinear pairing operation.

4.2 Proof of Lemma 2

Suppose that C is given a challenge: given $Z_m = x_m P$ and abP, compute $abx_m P$ after interacting with A_{II}. Now C and A_{II} play the role of the challenger and the adversary respectively. C will interact with A_{II} as follows:

- **Setup:** C runs algorithm Setup, chooses generator P and sets $P_{pub} = sP$, where s is the system master key. C picks an identity ID^* at random as the challenged ID in this game, and gives $params =< P, P_{pub}, H_1 >$ and the master secret s to A_{II} as the public parameters. For simplicity, we assume that for any ID_i, A_{II} queries H_1 before ID_i is used as an input of any query Public-key Extraction, Private-key Extraction and Signing oracles.

- **H_1-Queries:** C maintains a hash list H_1^{list} of tuple (ID_i, Q_i) as explained below. The list is initially empty. When A_{II} makes a hash oracle query on ID_i, if the query ID_i has already appeared on the H_1^{list}, then the previously defined value is returned. Otherwise, C chooses a random integer $a \in \mathbb{Z}_q^*$ and sets $Q_i = aP$. Then, he inserts the pair (ID_i, Q_i) in the list H_1^{list} and returns it to the adversary A_{II}.

- **Public-key Extraction Queries:** C maintains a list pk^{list} of tuple (ID_i, Q_i, r_i, pk_i), which is initially empty. When A_{II} queries on input ID_i, C checks whether pk^{list} contains a tuple for this input. If it does, the previously defined value is returned. Otherwise, C recovers the corresponding tuple (ID_i, Q_i) from the list H_1^{list} and picks a random value $r_i \in \mathbb{Z}_q^*$, computes $pk_i =< X_i, Y_i >=< r_i P, r_i Q_i >$ and returns pk_i. Then, C adds (ID_i, Q_i, r_i, pk_i) to the list pk^{list}.

- **Public-key Replacement** (ID_i, r_i', pk_i')**:** When A_{II} queries on input (ID_i, pk_i), C checks whether the tuple (ID_i, Q_i, r_i, pk_i) is contained in the pk^{list} . If it does, C sets $pk_i = pk_i'$ and adds (ID_i, Q_i, r_i', pk_i') to the pk^{list}. Here, we assume that C can obtain a replacing secret value r_i' corresponding to the replaced $pk_i' =< r_i' P, r_i' Q_i >$ from A_{II}. Otherwise, C executes Public Key extraction to generate (ID_i, Q_i, r_i, pk_i), then sets $pk_i = pk_i'$ and inserts it in the list pk^{list}.

- $Z-key$ **Extraction Queries:** C maintains a list Z^{list} of tuples (ID_i, Z_i), which is initially empty. If Z^{list} already contains the pair (ID_i, Z_i), then C returns it to the adversary A_{II}, otherwise C calls the oracle Private Key Extraction on identity ID_i and gets the value Z_i, forwards it to the adversary A_{II} and inserts it in the list Z^{list}.

- **Private-key Extraction Queries:** C maintains the list sk^{list}, for query on input ID_i, If $ID_i = ID^*$, C stops and outputs "failure" (denote the event by E_1). Otherwise, C picks a random number $x_i \in \mathbb{Z}_q^*$ and performs as follows:
  - If the E^{list} and the pk^{list} contain the corresponding tuple (ID_i, Q_i, D_i) and the tuple (ID_i, Q_i, r_i, pk_i) respectively, then C sets $sk_i = x_i D_i$, $Z_i = x_i P$, returns (ID_i, x_i, sk_i, Z_i) to A_{II} and adds them to the list sk^{list}.
  - Otherwise, C makes a Partial-private-key Extraction query and a Public-key Extraction query on ID_i, then simulates as the above process, sends (ID_i, x_i, sk_i, Z_i) to A_{II} and adds them to the list sk^{list}.

- **Signing Queries:** When C receives a signing query (ID_i, m_j), it acts as follows:
  - If $ID_i = ID^*$, C stops and returns "failure status" (denote the event by E_2).
  - Otherwise, C recovers the tuple (ID_i, x_i, sk_i, Z_i) from the sk^{list}, the tuple (ID_i, Q_i, pk_i) from the pk^{list} and the tuple (m_j, MP) from H_1^{list}.
  - Picks random integer $a \in \mathbb{Z}_q^*$.
  - Computes $MP_1 = ax_i MP$.
  - Computes $s_i = e(MP, Z_i)^{ar_i}$ and (MP_1, s_i) is the signature for the identity ID_i on the message m_j. C returns (MP_1, s_i) to A_{II} as response to the signing oracle.

Finally, A_{II} stops and outputs a signature $\sigma = (V^*, S^*)$ on the message m^* for the identity ID^*, which satisfies the equation $\text{Verify}(m^*, ID^*, pk^*, S^*) = 1$. C recovers the tuple (ID^*, Q^*, pk^*) from pk^{list}, the tuple (ID^*, Z), (m^*, MP^*) from Z^{list}, H_1^{list} and picks a random integer $a^* \in \mathbb{Z}_q^*$. Then, we have $e(V^*, X_i^*) = e(a^* x b^* P, r^* P) = S^*$, then $a^* b^* x P = V^*$.

Hence C can successfully compute and output $a^*b^*xP = V^*$ as solution to the A_{II}'s challenge. So, C breaks the ECDHP problem in G_1.

Also, C can solve the ECDHP problem successfully, if none of the events E_1 and E_2 happens. Now, we have:

$$Pr(\neg E_1 \wedge \neg E_2) = \left(\frac{q_H - 1}{q_H}\right)^{q_{sk}+q_s}.$$

Again, the challenger's C advantage is $\epsilon' < \epsilon(\frac{q_H-1}{q_H})^{q_{sk}+q_s}$ with a running time cost as $t' < t + (q_s + q_z)t_{sm} + q_s t_p$.

Therefore, if the attacker has no advantage in winning Game I and Game II which are defined as in Lemma 1 and Lemma 2, then the proposed certificateless digital signature scheme is existential unforgeable against adaptively chosen message attacks in the random oracle model with the assumptions that ECDHP and BDHP in G_1 are intractable.

5 Conclusions and Remarks

In this paper, the security proof of the digital signature scheme proposed by Hassouna et al. [13] was introduced in the random oracle model. The proposed signature scheme is strong, efficient, and resistant to the key-replacement attack.

Furthermore, since this proven signature scheme does not depend on the KGC master secret, then any cryptographic system utilizes this signature scheme can provide authentication and non-repudiation services even if the KGC is compromised as in the traditional PKI-based systems.

References

[1] S. S. Al-Riyami and K. G. Paterson, "Certificateless public key cryptography," in *Advances in Cryptology (Asiacrypt'03)*, pp. 452–473, Springer, 2003.

[2] P. S. L. M. Barreto, H. Y. Kim, B. Lynn, and M. Scott, "Efficient algorithms for pairing-based cryptosystems," in *In Advances in Cryptology (Crypto'02)*, LNCS 2442, pp. 354–368, Springer, 2002.

[3] P. S. L. M. Barreto, B. Lynn, and M. Scott, "Constructing elliptic curves with prescribed embedding degrees," in *Security in Communication Networks (SCN'2002)*, LNCS 2576, pp. 263–273, Springer, 2002.

[4] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," in *Advances in Cryptology (Crypto'01)*, LNCS 2139, pp. 213–229, Springer, 2001.

[5] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," *SIAM Journal on Computing*, vol. 32, no. 3, pp. 586–615, 2003.

[6] D. Boneh, H. Shacham, and B. Lynn, "Short signatures from the weil pairing," in *Advances in Cryptology (Asiacrypt'01)*, LNCS 2248, pp. 514–532, Springer, 2001.

[7] A. W. Dent, B. Libert, and K. G. Paterson, "Certificateless encryption schemes strongly secure in the standard model," in *Public Key Cryptography*, pp. 344–359, 2008.

[8] R. Dupont, A. Enge, and F. Morain, "Building curves with arbitrary small mov degree over finite prime fields," *Journal of Cryptology*, vol. 18, no. 2, pp. 78–89, 2002.

[9] S. D. Galbraith, "Supersingular curves in cryptography," in *Advances in Cryptology (Asiacrypt'01)*, LNCS 2248, pp. 495–513, Springer, 2001.

[10] S. D. Galbraith, K. Harrison, and D. Soldera, "Implementing the tate pairing," in *5th International Symposium on Algorithmic Number Theory*, LNCS 2369, pp. 324–337, Springer, 2002.

[11] Girault, "Self-certified public keys," in *Advances in Cryptology (Eurocrypt'91)*, LNCS 547, pp. 490–497, Springer, 1992.

[12] M. Hassouna, B. Barry, N. Mohamed, and E. Bashier, "An integrated public key infrastructure model based on certificateless cryptography," *International Journal of Computer Science and Information Security*, vol. 11, pp. 1–10, 2013.

[13] M. Hassouna, E. Bashier, and B. Barry, "A short certificateless digital signature scheme," in *International Conference of Digital Information Processing, Data Mining and Wireless Communications*, pp. 120–127, 2015.

[14] S. Sharmila Deva Selvi, S. Sree Vivek, and C. Pandu Rangan, "CCA2 secure certificateless encryption schemes based on RSA," *IACR Cryptology ePrint Archive*, vol. 2010, pp. 459, 2010.

[15] S. Sharmila Deva Selvi, S. Sree Vivek, and C. Pandu Rangan, "Certificateless kem and hybrid signcryption schemes revisited," in *International Conference of Information Security, Practice and Experience (ISPEC'10)*, pp. 294–307, 2010.

[16] C. Wang, D Long, and Y. Tang, "An efficient certificateless signature from pairing," *International Journal of Network Security*, vol. 8, no. 1, pp. 96–100, 2009.

[17] W. Xie and Z. Zhang, "Certificateless signcryption without pairing," *IACR Cryptology ePrint Archive*, vol. 2010, pp. 187, 2010.

[18] W. Xie and Z. Zhang, "Efficient and provably secure certificateless signcryption from bilinear maps," in *IEEE International Conference on Wireless Communications, Networking and Information Security (WCNIS'10)*, pp. 558–562, 2010.

[19] H. Xiong, Z. Qin, and F. Li, "An improved certificateless signature scheme secure in the standard model," *Fundamenta Informaticae*, vol. 88, pp. 193–206, 2008.

[20] L. Zhang and F. Zhang, "A new provably secure certificateless signature scheme," in *IEEE International Conference on Communications*, pp. 1685–1689, 2008.

Mohammed Alfateh Hassouna is Assistant Professor at Department of Computer Science - Faculty of Computer Studies - The National Ribat University - Sudan. He gained his PhD in cryptography from the University of Khartoum - Sudan. Currently he is working as ICT Manger at the National Ribat University. He has many published papers in the international journals and conferences related to the information security and cryptography.

Bazara Barry is an associate professor at the department of Computer Science - University of Khartoum and formerly the head of the same department. He was director of research at the Faculty of Mathematical Sciences. Bazara is a reviewer and TPC head/member of many international journals/conferences and a member of the IEEE. He has won several best paper and research awards at the international level.

Eihab Bashier obtained his PhD in 2009 from the University of the Western Cape in South Africa. He is an associate professor of applied mathematics at University of Khartoum, since 2013 and recently, he joined the department of Mathematics, Physics and Statistics of Qatar University. The research interests of Dr. Bashier are mainly in numerical methods for differential equations with applications to biology and in information and computer security. In 2011, Dr. Bashier won the African Union and the Third World Academy of Science (AU-TWAS) young scientists national award in basic sciences, technology and Innovation. Dr. Bashier is a reviewer for many international journals and an IEEE member.