Secure Certificateless Signature Scheme with Batch Verification from Bilinear Pairings

N.B. Gayathri(✉) and P. Vasudeva Reddy

Department of Engineering Mathematics,
Andhra University, Visakhapatnam, AP, India
gayatricrypto@gmail.com, vasucrypto@yahoo.com

Abstract. In view of simplifying certificate management complexities in the traditional Public Key Cryptography (PKC) and to abolish the key escrow problem in identity based PKC (ID-PKC), concept of Certificateless Public Key Cryptography (CL-PKC) was introduced. Batch Cryptography emphasizes new developments in information security and communication networks. It has been developed to enhance the efficiency of signatures verification, by verifying a batch of n message, signature pairs in a single instance. Batch Verification (BV) can be used in various areas where many clients interact with a single server. Mail servers, Sensor Networks, e-commerce are the best examples for BV. In this paper, we present a certificateless signature (CLS) scheme that supports BV using pairings. The proof of security is presented in Random Oracle Model (ROM) under the assumption of Computational Diffie-Hellman (CDH) Problem is intractable. More over the security proofs are made without using forking lemma [20] to achieve tight security. The efficiency analysis shows that our CLS scheme is more secure and efficient than the existing schemes.

Keywords: Public key cryptography · CLS scheme · Batch verification · Bilinear pairing · ROM · CDH problem

1 Introduction

Digital Signatures play a very important role in information security and communication networks by providing message authentication, data integrity and non-repudiation. The concept of CL-PKC was first proposed by Al-Riyami et al. [1], in 2003. Unlike the traditional PKC and ID-PKC, CL-Public key cryptographic schemes allow the verifier to verify the signatures without certificate and resolve the certificate management problem. Moreover, this cryptosystem completely abolishes the key escrow problem in ID-PKC by taking user's secret key as a combination of partial secret key generated by Key Generation Centre (KGC) and secret value chosen by user.

P. Mueller et al. (Eds.): SSCC 2016, CCIS 625, pp. 225–235, 2016.
DOI: 10.1007/978-981-10-2738-3_19

The basic idea of BV is to amortize the computational cost and time in verification process. In BV process, different signatures of different users on different messages can be batched to verify the signatures in a single instance instead of verifying them one after the other. This feature enables us to achieve high efficiency by reducing computational cost and time. BV can be used in many applications such as banking transactions, mail server, traffic control, military applications, wireless sensor networks etc. where many members interact with a single server. The concept of BV was first presented by Fiat [6] in 1990, based on RSA signatures. In 1994, Naccahe et al. [9] presented the first efficient batch verifying scheme for DSA signatures. In 1998, Bellare et al. [3] presented a pioneer work for BV and explained three standard methods for batching modular exponentiations. In 2005, Yoon et al. [13] proposed the first ID-based signature scheme with BV. Later many batch verifying schemes were proposed in traditional and ID-based setting. But there is no considerable work in certificateless setting.

Combination of BV technique with certificateless signatures integrates the advantages of both. The first batch verifying CLS scheme was presented by M. Geng et al. [7] in 2009. This scheme uses small exponent test to achieve efficient and secure BV. However, it achieves only Girault's Level 2 security [5]. Later, in 2014, C. I. Fan et al. [5] proposed a strongly secure CLS scheme supporting BV. It also uses small exponent test. This scheme achieves Girault's Level 3 security [5]. But BV cost is more due to more number of map to point hash functions. Moreover these two schemes use Forking Lemma [10] in their proof of security. Since the reductions using forking lemma are not tight, these schemes do not achieve tight security. To the best of our knowledge, these are the only schemes in certificateless setting which supports BV.

In this paper, to improve the efficiency of verification process and to achieve tight security we develop a CLS scheme which supports BV. This scheme is designed using bilinear pairings over elliptic curves. The security of this scheme is proved in ROM under the assumption that the CDH problem is hard. Moreover the security proofs are made without using forking lemma [10] to achieve tight security.

Structure of the Paper. Remainder of this paper is structured as follows. In Sect. 2 we presented some notations. In Sect. 3 we presented our CLS scheme with BV and its security proof. In Sect. 4 we presented the efficiency analysis of our scheme. Finally Sect. 5 deals with conclusion.

2 Preliminaries

We omit the Preliminaries, Definition of BV, Syntax and security model of CLS scheme in batch verification due to space constraint. Please refer to [5, 11] for details. Some notations are used throughout this paper for our convenience and are represented in Table 1.

Table 1. Notations and their meanings

Notation	Meaning
l, s	Security parameter & master secret key of the system generated by KGC
τ	System Parameter
z_q^*	The group with elements 1,2…q–1 under addition modulo q
G_{Adt}, G_{Mlt}	Additive & Multiplicative cyclic groups of same prime order q
H_1, H_2, H_3	Cryptographic one way hash functions
ID	User Identity
$UPSK_{ID}, USK_{ID}, UPK_{ID}$	User partial secret key, User secret key & User public key of the identity respectively
$\mathcal{ADV}_1, \mathcal{ADV}_2$	Type-I & Type-II adversaries respectively
ξ	An algorithm to solve CDH problem by using adversaries
$e : G_{Adt} \times G_{Adt} \rightarrow G_{Mlt}$	An admissible bilinear map
Ω	Signature on a message

3 New CLS Scheme with BV and Security Analysis

In this section we present an efficient CLS scheme with BV and its formal security analysis.

3.1 Proposed CLS Scheme with BV

Master Key Gen: KGC run this algorithm by taking security parameter $l \in Z^+$ as input and performs the following.

- Choose additive and multiplicative cyclic groups as G_{Adt} and G_{Mlt} of same prime order q with a bilinear pairing $e : G_{Adt} \times G_{Adt} \rightarrow G_{Mlt}$; and $P \in G_{Adt}$ as a generator of G_{Adt}.
- Select a random $s \in Z_q^*$ as the master secret key and sets master public key as $Q_{Pub} = sP$.
- Choose three cryptographic hash functions $H_1 : \{0,1\}^* \rightarrow G_{Adt}$, $H_2 : \{0,1\}^* \rightarrow Z_q^*$ and $H_3 : \{0,1\}^l \rightarrow G_{Adt}$. KGC publishes the system parameters as $\tau = \{q, G_{Adt}, G_{Mlt}, e, P, Q_{Pub}, H_1, H_2, H_3\}$ and keeps s secretly.

Partial Key Gen: KGC runs this algorithm by taking ID as input. KGC computes $K_{ID} = H_1(ID)$ and $UPSK_{ID} = sK_{ID}$ and sends $UPSK_{ID}$ to ID via secure channel.

User Key Gen: User runs this algorithm by choosing $x_{ID} \in Z_q^*$ randomly and sets $USK_{ID} = x_{ID}$ and $UPK_{ID} = x_{ID}P$.

Signature Generation: Signer runs this algorithm by taking $\tau, ID, UPK_{ID}, USK_{ID}$, $UPSK_{ID}$, message $m \in \{0,1\}^*$ as input and generates the signature Ω on a message $m \in \{0,1\}^*$ by performing the following.

- The signer chooses $r \in Z_q^*$ and computes $R = rP$, $h = H_2(m, ID, R, UPK_{ID})$ and $S = H_3(\Delta)$ where Δ is an arbitrary string of length t.
- The signer computes $T = hUPSK_{ID} + S(x_{ID} + r) + rQ_{Pub}$.

Now $\Omega = (R, T)$ is a signature on a message m.

Signature Verification: The verifier runs this algorithm by taking signature $\Omega = (R, T)$ on a message m with ID and corresponding public key UPK_{ID} as input and checks the validity of a signature as follows.

Compute $h = H_2(m, ID, R, UPK_{ID})$ and $S = H_3(\Delta)$, verifies the equation

$$e(T, P) = e(hK_{ID} + R, Q_{Pub})\, e(UPK_{ID} + R, S) \tag{1}$$

If the Eq. (1) holds, verifier accepts the signature $\Omega = (R, T)$; rejects otherwise.

Batch Verification: To verify n signatures $(\Omega_i)_{i=1\text{to}n}$ of n individual users $(\mathrm{U}_i)_{i=1\text{to}n}$ with identities $(ID_i)_{i=1\text{to}n}$ on messages $(m_i)_{i=1\text{to}n}$ respectively, a verifier performs the following.

- Choose $(\delta_i)_{i=1\text{to}n} \in Z_q^*$ randomly.
- Compute $K_{ID_i} = H_1(ID_i)$,$h_i = H_2(m_i, ID_i, R_i, UPK_{ID_i})$ and $S = H_3(\Delta)$, for $i = 1, 2, 3\ldots$
- The verifier accepts the validity of n signatures if the following equation holds

$$e(\sum_{i=1}^{n} \delta_i T_i, P) \;=\; e(\sum_{i=1}^{n} \delta_i (h_i K_{ID_i} + R_i), Q_{Pub})\; e(\sum_{i=1}^{n} \delta_i (UPK_{ID_i} + R_i), S) \tag{2}$$

3.2 Analysis of the Proposed Scheme

In this section we present the proof of correctness of the presented scheme and its security analysis.

3.2.1 Proof of Correctness of Single Signature

The correctness of the scheme can be verified by verifying Eq. (1) as follows.

$$\begin{aligned} e(T_i, P) &= e(h_i UPSK_{ID_i} + S(x_{ID_i} + r_i) + r_i Q_{Pub}, P) \\ &= e(h_i UPSK_{ID_i} + r_i Q_{Pub}, P) e(S(x_{ID_i} + r_i), P) \\ &= e(h_i K_{ID_i} + r_i P, sP) e(S, (x_{ID_i} + r_i)P) \\ &= e(h_i K_{ID_i} + R_i, Q_{Pub}) e(x_{ID_i} P + r_i P, S) \\ &= e(h_i K_{ID_i} + R_i, Q_{Pub}) e(UPK_{ID_i} + R_i, S). \end{aligned}$$

3.2.2 Proof of Correctness of Batch Verification

The correctness of BV can be verified by verifying Eq. (2) as follows.

$$
\begin{aligned}
e(\sum_{i=1}^{n}\delta_i T_i, P) &= e(\sum_{i=1}^{n}\delta_i(h_i UPSK_{ID_i} + S(x_{ID_i} + r_i) + r_i Q_{Pub}), P) \\
&= e(\sum_{i=1}^{n}\delta_i(h_i UPSK_{ID_i} + r_i Q_{Pub}), P) e(\sum_{i=1}^{n}\delta_i S(x_{ID_i} + r_i), P) \\
&= e(\sum_{i=1}^{n}\delta_i(h_i K_{ID_i} + r_i P), sP) e(S, \sum_{i=1}^{n}\delta_i(x_{ID_i} + r_i)P) \\
&= e(\sum_{i=1}^{n}\delta_i(h_i K_{ID_i} + R_i), Q_{Pub})\, e(\sum_{i=1}^{n}\delta_i(UPK_{ID_i} + R_i), S).
\end{aligned}
$$

3.2.3 Security Analysis

We prove the security of our CLS scheme against Type I and Type II adversary [11] using the following theorem.

Theorem 1. *The proposed CLS scheme with BV is existentially unforgeable against adaptive chosen message attacks in the ROM with the assumption that the CDH problem is hard.*

We prove this theorem with the help of the following two lemmas.

Lemma 1. *If there exists a probabilistic polynomial-time bounded Type-I batch forger* $\mathcal{ADV}_1$ *who can forge any signature of our batch of signatures under adaptive chosen message attack by asking at most* $q_{H_1}, q_{H_2}, q_{H_3}$ *questions to random oracles* H_1, H_2, H_3 *respectively,* q_{Cuser} *questions to the* ***Create User*** *request oracle,* q_{Rpsk} *questions to the* ***Reveal Partial Secret Key*** *extraction oracle,* q_{Rsk} *questions to the* ***Reveal Secret Key*** *extraction oracle and* q_{Sign} *questions to the* ***Sign*** *oracle in ROM then there exists an algorithm* ξ *that can be used by* $\mathcal{ADV}_1$ *to solve the CDH problem in elliptic curve group.*

Proof: Let $\mathcal{ADV}_1$ is a Type-I batch forger. Suppose that $\mathcal{ADV}_1'$*s* target identity is ID^*, and he can forge a valid signature on a message (m^*, ID^*). Now we prove that anyone can construct an algorithm ξ who can solve the CDH problem using $\mathcal{ADV}_1$. Challenger ξ is given $(A = uP, B = vP)$ as a random instance of the CDH problem in G_{Adt}.

Initialization Phase: Algorithm ξ sets $Q_{Pub} = A = uP$ and runs **Master Key Gen** to generate τ. ξ then gives τ and master public key to $\mathcal{ADV}_1$ and keeps s secretly.

Queries Phase: In this phase, $\mathcal{ADV}_1$ performs the oracle simulation and ξ responds to these oracles as follows.

Queries on oracle $H_1(H_1(ID_i))$**:** ξ maintains a list L_1, which is initially empty. It contains the tuples of the form (ID_i, l_{1i}, K_{ID_i}). After receiving a query $H_1(ID_i)$, if there is a tuple $H_1(ID_i, l_{1i}, K_{ID_i})$ on L_1, ξ returns K_{ID_i}. Otherwise, if $ID_i \neq ID^*$, ξ picks a random l_{1i}, sets $K_{ID_i} = l_{1i}P$ else (if $ID_i = ID^*$) it sets $K_{ID_i} = l_{1i}B = l_{1i}vP$. Finally, ξ returns K_{ID_i} and adds K_{ID_i} to L_1.

Queries on Oracle $H_2(H_2(m_i, ID_i, UPK_{ID_i}, R_i))$**:** ξ maintains a list L_2, which is initially empty. It contains the tuples of the form $(m_i, ID_i, UPK_{ID_i}, R_i, l_{2i})$. After receiving H_2 query on $(m_i, ID_i, UPK_{ID_i}, R_i)$, if a tuple $(m_i, ID_i, UPK_{ID_i}, R_i, l_{2i})$ exists on L_2,ξ returns l_{2i}. otherwise, ξ picks a random $l_{2i} \in Z_q^*$ and returns l_{2i}.ξ adds $(m_i, ID_i, UPK_{ID_i}, R_i, l_{2i})$ to L_2.

Queries on Oracle $H_3(H_3(\Delta_i))$**:** ξ maintains a list L_3, which is initially empty. It contains the tuples of the form (Δ_i, S_i, l_{3i}). After receiving a query on $H_3(\Delta_i)$, ξ gives the same answer from L_3, if the query has been made earlier. Otherwise, ξ picks a random $l_{3i} \in Z_q^*$, computes $S_i = l_{3i}P$ and returns S_i. ξ adds (Δ_i, S_i, l_{3i}) to L_3.

Reveal Partial Secret Key Oracle $(PSK(ID_i))$**:** ξ maintains a list L_{PSK}, which is initially empty. It contains the tuples of the form $(ID_i, UPSK_{ID_i})$. After receiving a query on $PSK(ID_i)$,ξ gives $UPSK_{ID_i}$ if the request has been made earlier. If $ID_i \neq ID^*$, ξ recovers the corresponding (ID_i, l_{1i}, K_{ID_i}) from the list L_1 and sets $UPSK_{ID_i} = l_{1i}Q_{Pub} = l_{1i}A$ and returns $UPSK_{ID_i}$ to $\mathcal{ADV}_1$ and adds $(ID_i, UPSK_{ID_i})$ to L_{PSK}. Otherwise, (if $ID_i = ID^*$), ξ aborts.

Create User Oracle $(Cuser(ID_i))$: ξ maintains a list L_{Cuser}, which is initially empty. It contains the tuples of the form $(ID_i, UPK_{ID_i}, USK_{ID_i})$. After receiving a query on $Cuser(ID_i)$, the current UPK_{ID_i} from the list L_{Cuser} will be given if the request has been made earlier. Otherwise, ξ will choose a random $w_i \in Z_q^*$ and sets $UPK_{ID_i} = w_iP$ and $USK_{ID_i} = w_i$. ξ gives UPK_{ID_i} and adds $(ID_i, UPK_{ID_i}, USK_{ID_i})$ to L_{Cuser}.

Reveal Secret Key Oracle $(RSK(ID_i))$: When $\mathcal{ADV}_1$ makes this query on $RSK(ID_i)$, if $(ID_i = ID^*)$,ξ aborts. Otherwise (if $ID_i \neq ID^*$), ξ finds the tuple $(ID_i, UPK_{ID_i}, USK_{ID_i})$ in a list L_{Cuser}, and returns USK_{ID_i} to $\mathcal{ADV}_1$. If there is no tuple in L_{Cuser},ξ makes a query on $Cuser(ID_i)$ to generate $(UPK_{ID_i} = w_iP, USK_{ID_i} = w_i)$.$\xi$ saves these values in L_{Cuser}, and returns $USK_{ID_i} = w_i$.

Replace Public Key Oracle $(RPK(ID_i))$: After receiving a query on $RPK(ID_i)$,ξ finds $(ID_i, UPK_{ID_i}, USK_{ID_i})$ in L_{Cuser}.ξ replaces $UPK_{ID_i} = UPK'_{ID_i}$ and $USK_{ID_i} = \perp$

Signing Oracle: When $\mathcal{ADV}_1$ makes this query on (ID_i, m_i), ξ chooses $r_i, h_i \in Z_q^*$ and computes $R_i = r_iP - h_iK_{ID_i}$. If the tuples containing h_i already exists in list L_2, then ξ chooses another $r_i, h_i \in Z_q^*$ and tries again. Set $h_i = H_2(m_i, ID_i, UPK_{ID_i}, R_i)$ and $S_i = H_3(\Delta_i)$. Then ξ computes $T_i = l_{3i}(R_i + UPK_{ID_i}) + r_iQ_{Pub}$. Finally ξ responds to $\mathcal{ADV}_1$ with $\Omega_i = (R_i, T_i)$. (R_i, T_i) is a valid signature on message m_i as it satisfies Eq. (1).

Forgery: Hence, $\mathcal{ADV}_1$ forges a valid signature $\Omega^* = (R^*, T^*)$ on messages m^* under ID^*. Suppose ξ can construct n valid signatures $\left(\Omega_i^*\right)_{i=1\text{to}n}$ on messages $\left(m_i^*\right)_{i=1\text{to}n}$ of the signers under $\left(ID_i^*\right)_{i=1\text{to}n}$ and the corresponding $\left(UPK_{ID_i}^*\right)_{i=1\text{to}n}$ of n users $(\mathrm{U}_i)_{i=1\text{to}n}$ with a state of information Δ^* by $\mathcal{ADV}_1$ such that

i. BV holds.
ii. There exists $I \in \{1, 2, 3, \ldots n\}$ such that $\mathcal{ADV}_1$ has not asked the Partial Secret Key queries for ID_I^* and $\mathcal{ADV}_1$ has not asked the Sign oracle query.

Without loss of generality, we let I = 1. In addition, the forged signature must satisfy Eq. (2) i.e.

$$e(\sum_{i=1}^{n}\delta_i T_i^*, P) = e(\sum_{i=1}^{n}\delta_i(h_i^* K_{ID_i}^* + R_i^*), Q_{Pub})e(\sum_{i=1}^{n}\delta_i(UPK_{ID_i}^* + R_i^*), S^*)$$

$$= e(\delta_1(h_1^* K_{ID_1}^* + R_1^*), Q_{Pub})e(\sum_{i=2}^{n}\delta_i(h_i^* K_{ID_i}^* + R_i^*), Q_{Pub})$$

$$\times e(\sum_{i=1}^{n}\delta_i(UPK_{ID_i}^* + R_i^*), S^*)$$

By our setting $K_{ID_1}^* = l_{1i}^* vP, S^* = l_{3i}^* P, R_1^* = r_1^* P, UPK_{ID_1}^* = w_1^* P$

$$e(\delta_1(h_1^* K_{ID_1}^* + R_1^*), Q_{Pub}) = e(\sum_{i=1}^{n}\delta_i T_i^*, P)\times$$

$$\left\{e(\sum_{i=2}^{n}\delta_i(h_i^* K_{ID_i}^* + R_i^*), Q_{Pub})e(\sum_{i=1}^{n}\delta_i(UPK_{ID_i}^* + R_i^*), S^*)\right\}^{-1}$$

$$\Rightarrow e(\delta_1(h_1^* l_{1i}^* vP + r_1^* P), uP) = e(\sum_{i=1}^{n}\delta_i T_i^*, P)\times$$

$$\left\{e(\sum_{i=2}^{n}\delta_i(h_i^* K_{ID_i}^* + R_i^*), Q_{Pub})e(\sum_{i=1}^{n}\delta_i(UPK_{ID_i}^* + R_i^*), S^*)\right\}^{-1}$$

$$\Rightarrow \delta_1(h_1^* l_{1i}^* uvP + r_1^* Q_{Pub}) = \sum_{i=1}^{n}\delta_i(T_i^* - (w_i^* + r_i^*)S^*) - \sum_{i=2}^{n}\delta_i(h_i^* l_{1i}^* + r_i^*)Q_{Pub}$$

$$\Rightarrow uvP = \left\{\sum_{i=1}^{n}\delta_i(T_i^* - (w_i^* + r_i^*)S^* - r_i^* Q_{Pub}) - \sum_{i=2}^{n}\delta_i(h_i^* l_{1i}^* Q_{Pub})\right\}(\delta_1 h_1^* l_{1i}^*)^{-1}.$$

Lemma 2. *If there exists a probabilistic polynomial-time bounded Type-II batch forger $\mathcal{ADV}_2$ who can forge any signature of our batch of signatures under adaptive chosen message attack by asking at most q_{H_2}, q_{H_3} questions to random oracles H_2, H_3 respectively, q_{Cuser} questions to the **Create User** request oracle, q_{Rsk} questions to the **Reveal Secret Key** extraction oracle and q_{Sign} questions to the **Sign** oracle in ROM then there exists an algorithm ξ that can be used by $\mathcal{ADV}_2$ to solve the CDH problem in elliptic curve group.*

Proof: Let $\mathcal{ADV}_2$ is a Type-II batch forger. Suppose that $\mathcal{ADV}_2's$ target identity is ID^*, and he can forge a valid signature on a message (m^*, ID^*). Now we prove that anyone can construct an algorithm ξ who can solve the CDH problem using $\mathcal{ADV}_2$. Challenger ξ is given $(A = uP, B = vP)$ as a random instance of the CDH problem in G_{Adt}.

Initialization Phase: $\mathcal{ADV}_2$ chooses a random value $s \in Z_q^*$ as master secret key and sets $Q_{Pub} = sP$.$\mathcal{ADV}_2$ runs **Master Key Gen** to generate τ and master public key and then gives s and master public key to the challenger ξ.

Queries Phase: In this phase, $\mathcal{ADV}_2$ performs the oracle simulation and ξ responds to these oracles as follows.

Create User Oracle $(Cuser(ID_i))$**:** ξ maintains a list L_{Cuser}, which is initially empty. It contains the tuples of the form $(ID_i, UPK_{ID_i}, USK_{ID_i})$. After receiving a query on $Cuser(ID_i)$, the current UPK_{ID_i} from the list L_{Cuser} will be given if the request has been made earlier. ξ will choose a random $l_{1i} \in Z_q^*$ and sets $UPK_{ID_i} = l_{1i}P$ if $ID_i \neq ID^*$; otherwise, it sets $UPK_{ID_i} = l_{1i}B = l_{1i}vP$. In both cases, ξ sets $USK_{ID_i} = l_{1i}$. ξ gives UPK_{ID_i} and adds $(ID_i, UPK_{ID_i}, USK_{ID_i})$ to L_{Cuser}.

Reveal Secret Key Oracle $(RSK(ID_i))$: When $\mathcal{ADV}_2$ makes this query on $RSK(ID_i)$, if $(ID_i = ID^*)$, ξ aborts. Otherwise (if $ID_i \neq ID^*$), ξ finds the tuple $(ID_i, UPK_{ID_i}, USK_{ID_i})$ in a list L_{Cuser}, and returns USK_{ID_i} to $\mathcal{ADV}_2$. If there is no tuple in L_{Cuser}, ξ makes a query on $Cuser(ID_i)$ to generate $(UPK_{ID_i} = l_{1i}P, USK_{ID_i} = l_{1i})$.$\xi$ saves these values in L_{Cuser}, and returns $USK_{ID_i} = l_{1i}$.

Queries on Oracle: $H_2(H_2(m_i, ID_i, UPK_{ID_i}, R_i))$: ξ maintains a list L_2, which is initially empty. It contains the tuples of the form $(m_i, ID_i, UPK_{ID_i}, R_i, l_{2i})$. After receiving H_2 query on $(m_i, ID_i, UPK_{ID_i}, R_i)$, if a tuple $(m_i, ID_i, UPK_{ID_i}, R_i, l_{2i})$ exists on L_2,ξ returns l_{2i}. otherwise, ξ picks a random $l_{2i} \in Z_q^*$ and returns l_{2i}.ξ adds $(m_i, ID_i, UPK_{ID_i}, R_i, l_{2i})$ to L_2.

Queries on Oracle: $H_3(H_3(\Delta_i))$: ξ maintains a list L_3, which is initially empty. It contains the tuples of the form (Δ_i, S_i, l_{3i}). After receiving a query on $H_3(\Delta_i)$, ξ gives the same answer from L_3, if the query has been made earlier. Otherwise,ξ picks a random $l_{3i} \in Z_q^*$ and computes $S_i = l_{3i}uP$, returns S_i.ξ adds (Δ_i, S_i, l_{3i}) to L_3.

Signing Oracle: When $\mathcal{ADV}_2$ makes this query on (ID_i, m_i),ξ chooses $r_i, h_i \in Z_q^*$ and computes $R_i = r_iP - h_iK_{ID_i}$. If the tuples containing h_i already exists in list L_2, then ξ chooses another $r_i, h_i \in Z_q^*$ and tries again. Set $h_i = H_2(m_i, ID_i, UPK_{ID_i}, R_i)$ and $S_i = H_3(\Delta_i)$. Then ξ computes $T_i = l_{3i}(R_i + UPK_{ID_i}) + r_iQ_{Pub}$. Finally ξ responds to $\mathcal{ADV}_2$ with $\Omega_i = (R_i, T_i)$. (R_i, T_i) is a valid signature on message m_i as it satisfies Eq. (1).

Forgery: Hence, $\mathcal{ADV}_2$ forges a valid signature $\Omega^* = (R^*, T^*)$ on messages m^* under ID^*. Suppose ξ can construct n valid signatures $\left(\Omega_i^*\right)_{i=1\text{to}n}$ on messages $(m_i^*)_{i=1\text{to}n}$ of the signers under $(ID_i^*)_{i=1\text{to}n}$ and the corresponding $(UPK_{ID_i}^*)_{i=1\text{to}n}$ of n users $(U_i)_{i=1\text{to}n}$ with a state of information Δ^* by $\mathcal{ADV}_2$ such that

i. BV holds.
ii. There exists $I \in \{1, 2, 3, \ldots n\}$ such that $\mathcal{ADV}_2$ has not asked the Partial Secret Key queries for ID_I^* and $\mathcal{ADV}_2$ has not asked the Sign oracle query.

Without loss of generality, we let I = 1. In addition, the forged signature must satisfy

$$e(\sum_{i=1}^{n}\delta_i T_i^*, P) = e(\sum_{i=1}^{n}\delta_i(h_i^* K_{ID_i}^* + R_i^*), Q_{Pub})e(\sum_{i=1}^{n}\delta_i(UPK_{ID_i}^* + R_i^*), S^*)$$
$$= e(\sum_{i=1}^{n}\delta_i(h_i^* K_{ID_i}^* + R_i^*), Q_{Pub})e(\delta_1(UPK_{ID_1}^* + R_1^*), S^*)$$
$$\times e(\sum_{i=2}^{n}\delta_i(UPK_{ID_i}^* + R_i^*), S^*)$$

By our setting $UPK_{ID_1}^* = l_{1i}^* vP, S^* = l_{3i}^* uP, R_1^* = r_1^* P, UPK_{ID_i}^* = l_{1i}^* P, R_i^* = r_i^* P$ and $Q_{Pub} = sP$.

$$e(\delta_1(UPK_{ID_1}^* + R_1^*), S^*) = e(\sum_{i=1}^{n}\delta_i T_i^*, P)\times$$
$$\left\{e(\sum_{i=1}^{n}\delta_i(h_i^* K_{ID_i}^* + R_i^*), Q_{Pub})e(\sum_{i=2}^{n}\delta_i(UPK_{ID_i}^* + R_i^*), S^*)\right\}^{-1}$$
$$\Rightarrow \delta_1(l_{1i}^* l_{3i}^* uvP + r_1^* l_{3i}^* uP) = \left\{\sum_{i=1}^{n}\delta_i(T_i^* - (h_i^* UPSK_{ID_i}^* + sR_i^*)) - \sum_{i=2}^{n}\delta_i(UPK_{ID_i}^* + R_i^*)l_{3i}^* u\right\}$$
$$\Rightarrow uvP = \left\{\sum_{i=1}^{n}\delta_i(T_i^* - (h_i^* UPSK_{ID_i}^* + sR_i^*)) - \sum_{i=2}^{n}(\delta_i(l_{1i}^* uP + r_i^* uP)l_{3i}^*) - \delta_1 r_1^* l_{3i}^* uP\right\}(\delta_1 l_{1i}^* l_{3i}^*)^{-1}.\square$$

4 Efficiency Analysis

In this part we compare our scheme with the relevant schemes [5, 7] in terms of security, signature length, verification cost and computation cost. We consider the experimental results presented in [2, 4, 8, 12] for various cryptographic operations and their conversions. Table 2 presents these conversions and detailed comparison with relevant schemes is presented in Table 3.

From the following Table 3, the communicational and computation cost of our scheme is more efficient than C. I. Fan et al. [5] and almost as efficient as Geng et al. [7] scheme. But our scheme is more secure than all other schemes since the security reduction do not use Forking lemma.

Table 2. Notations and descriptions of various cryptographic operations and their conversions

Notations	Descriptions
T_M	Time required to compute modular multiplication operation
T_E	Time required to compute the elliptic curve point multiplication (Scalar multiplication in G_{Adt}): $T_E = 29T_M$
T_P	Time required to compute the bilinear pairing in G_{Mlt}: $T_P = 87T_M$
T_H	Time required to compute a map to point hash function: $1T_H = 1T_E = 29T_M$
T_A	Time required to compute the elliptic curve point addition (point addition in G_{Adt}): $T_A = 0.12T_M$

Table 3. Comparison of the proposed scheme with the related schemes

Scheme	Signing cost	Batch verification cost	Sign. length	Without forking lemma
M. Geng et al. (2009)	$3T_E + 1T_A + 1T_H$	$3T_P + 4nT_E + (4n-3)T_A + 1T_H$	$2\|G_{Adt}\|$	No
C.I. Fan et al. (2014)	$5T_E + 3T_A + 1T_H$	$3T_P + 5nT_E + (5n-3)T_A + (n+1)T_H$	$4\|G_{Adt}\|$	No
Our scheme	$4T_E + 2T_A + 1T_H$	$3T_P + 4nT_E + (5n-3)T_A + 1T_H$	$2\|G_{Adt}\|$	Yes

5 Conclusions

In this paper, we have proposed a novel and secure batch verifiable CLS scheme using bilinear pairings over elliptic curves. This scheme verifies the batch of signatures using small exponent test. The presented scheme is unforgeable under CDH assumption. Moreover the scheme is proved without using Forking lemma which results our scheme is tightly secure. Thus we can apply our scheme in practical environments such as Internet of Things environments associated with intelligent transportation systems (ITS) to manage traffic caused by vehicles in a metropolitan area etc.

Acknowledgements. The authors are grateful and sincerely thank the reviewers for their valuable suggestions. This work is supported by WOS-A, DST, Govt. of India under the grant No.SR/WOS-A/PM-1033/2014 (G), WOS-A, DST.

References

1. Al-Riyami, Sattam S., Paterson, Kenneth G.: Certificateless public key cryptography. In: Laih, Chi-Sung (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 452–473. Springer, Heidelberg (2003)
2. Barreto, Paulo S.L.M., Kim, Hae Y., Lynn, Ben, Scott, Michael: Efficient algorithms for pairing-based cryptosystems. In: Yung, Moti (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 354–368. Springer, Heidelberg (2002)
3. Bellare, Mihir, Garay, Juan A., Rabin, Tal: Fast batch verification for modular exponentiation and digital signatures. In: Nyberg, Kaisa (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 236–250. Springer, Heidelberg (1998)
4. Cao, X., Kou, W., Du, X.: A Pairing –free Identity Based Authenticated Key Agreement Protocol with Minimal Message Exchanges. Inf. Sci. **180**(15), 2895–2903 (2010)
5. Fan, C.I., Ho, P.H., Tseng, Y. F.: Strongly secure certificateless signature scheme supporting batch verification In: Mathematical Problems in Engineering, vol. 2014, Article ID 854135, 11 pages. Hindawi Publishing Corporation. http://dx.doi.org/10.1155/2014/854135. (2014)
6. Fiat, A.: "Batch RSA," in Advances in cryptology-CRYPTO, pp. 175–185. (1990)

7. Geng, M., Zhang, F.: Batch verification for certificateless signature schemes. In: Proceedings of the International Conference on Computational Intelligence and Security (CIS 2009), pp. 288– 292, December. 2009
8. MIRACL Library. http://certivox.org/display/EXT/MIRACL
9. Naccache, David, Raihi, DavidM, Vaudenay, Serge, Raphaeli, Dan: Can D.S.A. be improved? In: De Santis, Alfredo (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 77–85. Springer, Heidelberg (1995)
10. Pointcheval, D., Stern, J.: Security arguments for digital signatures and blind signatures. J. Cryptology **13**(3), 361–369 (2000)
11. Shim, K.A.: Security models for certificateless signature schemes revisted. Inf. Sci. **296**, 315–321 (2015)
12. Tan, S-Y., Heng, S-H., Goi, B-M.: Java implementation for pairing-based cryptosystems. In: Taniar, D., Gervasi, O., Murgante, B., Pardede, E., Apduhan, B.O. (eds.) ICCSA 2010, Part IV. LNCS, vol. 6019, pp. 188–198. Springer, Heidelberg (2010)
13. Yoon, HyoJin, Cheon, Jung Hee, Kim, Yong-Dae: Batch verifications with id-based signatures. In: Park, Choon-sik, Chee, Seongtaek (eds.) ICISC 2004. LNCS, vol. 3506, pp. 233–248. Springer, Heidelberg (2005)