A Provable Secure Short Signature Scheme Based on Bilinear Pairing over Elliptic Curve

Subhas Chandra Sahana and Bubu Bhuyan
(Corresponding author: Subhas Chandra Sahana)

Department of Information Technology, North-Eastern Hill University
Shillong, Meghalaya, India
(Email: subhas.sahana@gmail.com)
(Received July 31, 2017; revised and accepted Oct. 22, 2017)

Abstract

Currently, short signature is receiving significant attention since it is particularly useful in communication with low-bandwidth as the size of the generated signature is shorter than other conventional signature schemes. In this paper, a new short signature scheme is proposed based on bilinear pairing over elliptic curve. The proposed scheme is efficient as it takes lesser number of cost effective pairing operations than the BLS signature scheme. Moreover, the proposed scheme does not require any special kind of hash function such as Map-To-Point hash function. The efficiency comparison of the proposed scheme with other similar established short signature schemes is also done. The security analysis of our scheme is done in the random oracle model under the hardness assumptions of a modified **k-CAA** problem, a variant of the original **k-CAA** problem. In this paper, we also provide an implementation result of the proposed scheme.

Keywords: BLS Signature Scheme; Bilinear Pairing; Elliptic Curve; Map-To-Point Hash Function; Short Signature

1 Introduction

Short signature is a variant of digital signature. As the size of the signature generated by a short signature scheme is shorter so, it is suitable in low-bandwidth communication environments. For instance, as said in Bellare and Neven [5] (2006), wireless devices have a short battery life. Communicating even one bit of information uses essentially more power than executing one 32-bit instruction (Barr and Asanovic, 2003). Consequently, diminishing the number of bits in communication saves power and increase the battery life. In numerous settings, communication channels are not reliable. So with the short signature, it reduces the number of bits to be sent over a communication channel. In addition to this, signature scheme with shorter signature length has higher priority in many applications. For example, considering those applications where signatures are going to be printed on papers or CDs, the signature size is the principal factor. Due to its numerous application, many short signature schemes have been proposed fitted in different cryptosystem. For example, the short signature schemes in [2,14,20] are Public Key Infrastructure (PKI) based and the short signature schemes in [10,12,17] are fitted in certificate-less cryptosystem.

In 2001, the first short signature scheme, called BLS [7] signature, was proposed by Boneh, Lynn and Shacham. Since then, short signature has been investigated intensively and many short signature schemes have been proposed [1,19]. The technique behind the achieved a shorter length signature is the use of bilinear pairing over the elliptic curve group. Actually, the elliptic curve group provides shorter key size with same security level of Diffie-Hellman (DH) group. The Table 1. shows the NIST's recommendation of key size to be used for achieving same security level of symmetric key cryptosystem. It can be observed from the table that Elliptic Curve Cryptography (ECC) has the shorter key size than the RSA with same level of security.

Table 2 shows the comparison on the number of bits present in the produced signature of different signature generation algorithms. From the table it is clear that to get a security level of λ bits, the BLS, Schnoor, ECDSA, RSA signature scheme produces a signature of size $2\lambda, 3\lambda, 4\lambda, O(\lambda^3)$ bits respectively.

Recently, bilinear paring mainly Weil pairing and Tate pairing are used as tools to construct variant signature schemes. There are some cryptographic schemes which can only be constructed by bilinear pairing, for example ID-based encryption, non-trivial aggregate signature, tripartite one round Diffie-Hellman key exchange, etc. Besides these, some primitives which can be constructed using other techniques, but for which pairings provides improved functionality and makes the cryptographic schemes simple and efficient such as tripartite one round Diffie-Hellman key exchange, etc. Short signature can provide a high security level with relatively shorter

Table 1: Recommend key sizes NIST [3]

Symmetric Key Size (bits)	**RSA and Diffie-Hellman Key Size (bits)**	**Elliptic Curve Key size (bits)**
80	*1024*	*160*
112	*2048*	*224*
128	*3072*	*256*
192	*7680*	*384*
256	*15360*	*512*

Table 2: Signature size at security level $\lambda = 128$bits

Algorithm	**Signature size (bits)**	$\lambda = 128$
RSA [8]	$\mathcal{O}(\lambda^3)$	2048
ECDSA [11]	4λ	512
Schnorr [16]	3λ	384
BLS [7]	2λ	256

signature length. The best known shortest signature is BLS [7] short signature which has half the size of a Digital Signature Algorithm (DSA) [9] signature but gives a same security level. The DSA [9] was the best known algorithm to generate a shorter length signature before the introduction of bilinear pairing. The length of the generated signature by the DSA [9] over the finite field $\mathbb{F}_q$ is about $2\log q$. On the other side, using bilinear pairing as a tool, the signature length is approximately $\alpha \log q$ where $\alpha = \log q / \log r$ and r is chosen in such a way that it is the largest prime divisor of the total number points on an elliptic curve. The logic behind of using elliptic curve is to get same level of security of RSA cryptosystem using lesser number of bits used in underline field on which the elliptic curve constructed. From the Table 1, it is clear that if we decide to use NISTs figure, then to achieve 256 bits of security level, we will need to select a elliptic curve group $E(\mathbb{F}_q)$ of size 512 bits. On the other hand, it is equivalent to a field $\mathbb{F}_q$ of size 15360 bits.

The rest of this paper is organized as follows: In Section 2, some basic preliminaries behind our work are discussed. In Section 3, a new short signature scheme inspired by Sedat *et al.* [1] is proposed from bilinear pairing, followed by, security analysis of the proposed scheme in the random oracle model is done in Section 4. In Section 5, an implementation results have been given. The efficiency analysis of our scheme with most similar established signature schemes has been provided in Section 6. Finally, we conclude our work in Section 7.

2 Preliminaries

In this Section, the basic mathematical background on which the proposed scheme stans has been discussed.

2.1 Bilinear Pairing

Let G_1 be an additive cyclic group generated by P whose order is a prime q and G_2 be a multiplicative cyclic group of the same order q. A **bilinear pairing** is a map e : $G_1 \times G_1 \to G_2$ with the following properties:

- **Bilinearity**: e(aP, bQ)= $e(P,Q)^{ab}$ for all $P, Q \in G_1$ and all a,b $\in Z_q^*$.
- **Non-Degenerate**: There exists $P, Q \in G_1$ such that e$(P,Q) \neq 1$.
- **Computable**: There is an efficient algorithm to compute e(P,Q) , for all $P, Q \in G_1$.

2.2 Diffie-Hellman Problem

Actually, the cryptographic schemes from bilinear pairing are based on the difficulty of solving certain Diffie-Hellman problem which is assumed to be a hard problem.

- **Decisional Diffie-Hellman Problem (DDHP)**: For $a, b, c \in_R Z_q^*$, If P, aP, bP, cP is given, to decide whether $c \equiv ab \bmod q$, is known as Decisional Diffie-Hellman Problem. The DDHP is not a hard problem as bilinear pairing can be used to solve this decision problem in polynomial time.

- **Computational Diffie-Hellman Problem(CDHP)**: For $a, b \in_R Z_q^*$, given P, aP, bP, to compute abP is known as Computational Diffie-Hellman Problem which is a hard problem.

- **Gap Diffie-Hellman (GDH) group**: A group G is called a Gap Diffie-Hellman (GDH) group if DDHP can be solved in polynomial time but no probabilistic algorithm can solve CDHP with non-negligible advantage within polynomial time in G.

- **k-CAA Problem:** For a integer k, given P, sP and k pairs $\left\{e_1, \dfrac{1}{s+e_1}.P\right\}, \left\{e_2, \dfrac{1}{s+e_2}.P\right\}, \left\{e_3, \dfrac{1}{s+e_3}.P\right\}$ ...... $\left\{e_k, \dfrac{1}{s+e_k}.P\right\}$; compute $\left\{e, \dfrac{1}{s+e}.P\right\}$ for some $e \notin \{e_1, e_2, e_3 ....e_k\}$. It is believed that the **k-CAA** problem is a hard problem. The problem firstly introduced by Mitsunari *et al.* [13]. However, the security of our proposed scheme is based on a modified version of the original **k-CAA** problem. We call it as the Modified **k-CAA** Problem which is the cubic version of the original **k-CAA** problem.

- **Modified k-CAA Problem:** For a integer k, given P, sP and k pairs $\left\{e_1, \left(\dfrac{1}{s+e_1}\right)^3 \cdot P\right\}, \left\{e_2, \left(\dfrac{1}{s+e_2}\right)^3 \cdot P\right\},$ $\left\{e_3, \left(\dfrac{1}{s+e_3}\right)^3 \cdot P\right\} \cdots \left\{e_k, \left(\dfrac{1}{s+e_k}\right)^3 \cdot P\right\};$

Compute $\{e, \left(\frac{1}{s+e}\right)^3 \cdot P\}$ for some $e \notin \{e_1, e_2, \cdots, e_k\}$. The modified **k-CAA** problem is not harder than original version of **k-CAA** problem [18].

3 The Proposed Short Signature Scheme

Our proposed scheme has been constructed from symmetric bilinear pairing, which means the two input groups in pairing operation are same. Let G_1 and G_2 be cyclic additive and multiplicative group respectively of prime order q each. Let P is the generator point of G_1 and the bilinear map is the $e : G_1 \times G_1 \rightarrow G_2$. Let H be general cryptographic hash function such as MD5, SHA-1. Suppose that Alice wants to send a signed message to Bob. Like other signature scheme, the proposed scheme consists of four steps.

1) System Initialization: In this step all the system parameters G_1, G_2, e, q, P, H are setup.

2) Key Generation: A random value $x \in Z_q^*$ chosen by Alice and computes $P_{pub1} = x^3P, P_{pub2} = 3x^2P, P_{pub3} = 3xP$. In this setup, $P_{pub1}P_{pub2}, P_{pub3}$ are the public keys, x is the secret key.

3) Signing: Given a secret key x and a message m, Alice computes the signature $\sigma = (H(m) + x)^{(-3)}P$.

4) Verification: Using public keys $P_{pub1}, P_{pub2}, P_{pub3}$, a message m and a signature σ, Bob verifies the signature σ by the following equation holds or not.

$$\begin{aligned} & e(H(m)^3P + P_{pub1} + P_{pub2}H(m) + P_{pub3}H(m)^2, \sigma) \\ & = e(P, P) \end{aligned}$$

If the above equation holds, Bob accepts the signature σ of the message m otherwise bob rejects it.

Correctness:

$$\begin{aligned} & e(H(m)^3P + P_{pub1} + P_{pub2}H(m) + P_{pub3}H(m)^2, \sigma) \\ & = e((H(m)^3 + x^3 + 3x^2H(m) + 3xH(m)^2)P, \sigma) \\ & = e(P, P)^{(H(m)+x)^{-3}(H(m)+x)^3} \\ & = e(P, P) \end{aligned}$$

4 Security Analysis

In this Section, we give the security proof for our proposed short signature scheme in the random oracle model. The above short signature is secure against existential forgery under adaptive chosen message attack in the random oracle model with the assumption that the modified **k-CAA** Problem in G_1 is hard.

Theorem 1. *Let us assume that there is an adaptively chosen message attacker $F(t, q_h, q_s, \epsilon)$-breaks the proposed scheme where it is assumed that F makes q_h queries to the hashed oracle and q_s queries to signature oracle and can break the proposed scheme with non-negligible probability ϵ and time t. Then there exists an algorithm $\mathcal{A}$ which, as a black box, can solve the modified **k-CAA** with non-negligible probability*

$$\epsilon' \geq \frac{1}{qs}.\left(1 - \frac{1}{qs+1}\right)^{qs+1}.\epsilon$$

and time $t' \leq t + t_{serach}.q_s + C.q_h + t_s$, where t_{serach} is the time to searching a list , C is the constant time for each hash request and t_s is the running time of the simulator.

We assume that F is well-behaved in the sense that it always requests the hash of a message m before it requests a signature for m, and that it always requests a hash of a message m for which it outputs as its forgery.It is trivial to achieve this property by modifying any forger algorithm F. In addition to this, it is needed that $\mathcal{A}$ would be engaged in a certain amount of book-keeping work. In particular, it must maintain a list of the messages m_i on which F requests hashed value h_i and signatures σ_i.

Proof. Suppose that $\mathcal{A}$ is a given a challenge:
For a integer k, given P, sP and k pairs $\left\{e_1, \left(\frac{1}{s+e_1}\right)^3.P\right\}, \left\{e_2, \left(\frac{1}{s+e_2}\right)^3.P\right\}, \left\{e_3, \left(\frac{1}{s+e_3}\right)^3 \cdot P\right\} \cdots \left\{e_k, \left(\frac{1}{s+e_k}\right)^3.P\right\}$;

Compute $\{e, \left(\frac{1}{s+e}\right)^3 \cdot P\}$ for some $e \notin \{e_1, e_2, \cdots, e_k\}$. Now, $\mathcal{A}$ and F play the role of challenger and adversary respectively. □

4.1 Construction of $\mathcal{A}$

For Simplicity, $\mathcal{A}$ is constructed in a series of games. Each game is a variant of the previous game. It is worth of mentioning that $\mathcal{A}_1$, $\mathcal{A}_2$, $\mathcal{A}_3$ and $\mathcal{A}_4$ denotes the adversary for the *Game 1, Game 2, Game 3 and Game 4* respectively. The $\mathcal{A}$ has the power to simulate the behavior of the attacker F. In each game, we will use a probability ξ which will be optimized later. The symbol β_ξ denotes the probability distribution over the set {0,1} where 1 is drawn from the set with probability ξ and 0 with $(1 - \xi)$.

Game 1. In *setup*, all the system parameters are generated. The public parameter are published in the public. The secret parameter s is kept secret from the $\mathcal{A}$ and from F.The public keys pk are constructed, as follows.

- $P_{pub1} = s^3P$;
- $P_{pub2} = 3s^2P$;

- $P_{pub3} = 3sP$.

All the above public keys are sent to attacker F. The values of $sP = 3^{-1}P_{pub3}$ is given to the algorithm $\mathcal{A}$. Then, for each message $m_i, 1 \le i \le q_h, \mathcal{A}_1$ picks a random bit $s_i \xleftarrow{R} \beta_\xi$ and set $H(m_i) = h_i$. The value of h_i is set to e_i where $1 \le i \le q_h$ and return the value e_i as a response of the has query. When the Adversary F makes a signature query on a message m_i, then the $\mathcal{A}$ searches the e_i value in the list and return $\left(\frac{1}{s+e_i}\right)^{-3}.P$ as a responded signature. Actually, the list consists of the tuple $\{m_i, e_i, \sigma_i\}$, where the message m_i is stored in a list with its hashed value e_i, and its signature $\sigma_i = \left(\frac{1}{s+e_i}\right)^{-3}.P$. Note that, $(m_i, \mathsf{pk}, h_i = e_i, \sigma_i = \left(\frac{1}{s+e_i}\right)^{-3} P)$ is valid Diffie-Hellman tuple as it passes the signature verification process.

$$\begin{aligned}
L.H.S. &= e(H(m_i)^3P + P_{pub1} + P_{pub2}H(m_i) \\
&\qquad + P_{pub3}H(m_i)^2, \sigma_i) \\
&= e(e_i^3P + P_{pub1} + P_{pub2}e_i \\
&\qquad + P_{pub3}e_i^2, \left(\frac{1}{s+e_i}\right)^{-3}.P) \\
&= e(e_i^3 + s^3 + 3s^2e_i + 3se_i^2)P, \{s+e_i\}^{-3}.P) \\
&= e(P,P)^{(e_i+s)^{-3}(e_i+s)^3} \\
&= e(P,P) \\
&= R.H.S
\end{aligned}$$

Finally, F halts, either conceding he failed or returning a forged signature $(m^*; \sigma^*)$, where $m^* = m_i{}^*$ for some i^* on which F he did not requested a signature. Suppose F succeeds in forging, $\mathcal{A}_1$ outputs *success*; otherwise, it outputs *"failure"*.
Thus

$$\begin{aligned}
Adv_{\mathcal{A}_1} &= Prob.\Big[\mathcal{A}_1^F(\textit{modified } \textbf{\textit{k-CAA}} \textit{ Problem}) \\
&\qquad = success\Big] \\
&= Prob.\Big[Verify(\mathsf{pk}, m^*, \sigma^*) = Valid\Big] \\
&= \epsilon
\end{aligned}$$

Game 2. $\mathcal{A}_2$ acts as does $\mathcal{A}_1$, with a little difference. If F fails, $\mathcal{A}_2$ outputs "*failure*"; if F succeeds, giving output a forgery (m^*, σ^*), where i^* is the index of m^*, then $\mathcal{A}_2$ outputs *success*, if $s_i{}^* = 1$, but *failure* if $s_i{}^* = 0$. Clearly, F can get no information about any $s_i{}^*$, so its behavior cannot depend on their values. As the value of $s_i{}^* = 1$ is chosen from the set $\{0, 1\}$ with probability ξ thus we have

$$Adv_{\mathcal{A}_2} = Adv_{\mathcal{A}_2}.Pr[s_i^* = 1] = \xi.\epsilon$$

Game 3. $\mathcal{A}_3$ acts as does $\mathcal{A}_2$, with a minor difference. If F unable to forge signature, $\mathcal{A}_3$ also fails. If F able to forge signature for the message m_i^* then $\mathcal{A}$ also claims the success to get a solution to the undertaken computational problem if $s_i^* = 1$ and F would submit signature query only for the message m_i for which $s_i = 0$.

As no information is supplied about the s_i to the F, each signature query can cause $\mathcal{A}$ to declare a failure with probability $(1 - \xi)$. Thus we have

$$\begin{aligned}
Adv_{\mathcal{A}_3} = Adv_{\mathcal{A}_2}.Pr[s_{ij} = 0, j = 1....k] &= \xi\epsilon.(1-\epsilon)^k \\
&\ge (1-\epsilon)^{qs}\epsilon\xi
\end{aligned}$$

Game 4. $\mathcal{A}_4$ acts like $\mathcal{A}_3$ does. However, if $\mathcal{A}_4$ succeeds, outputs $\sigma^* = (\frac{1}{s+e})^3P$ as forgery of the message m_{i^*}, where e is the hashed value of the massage m_{i^*}, *i.e.* $H(m_{i^*}) = e$ for which F output a forged signature σ^*. Clearly, $\mathcal{A}_4$ succeeds with precisely the same probability as $\mathcal{A}_3$, so

$$\begin{aligned}
Adv_{\mathcal{A}_4} &= Adv_{\mathcal{A}_3} \\
&= Adv_{\mathcal{A}_2}.Pr[s_{ij} = 0, j = 1, 2, \cdots, k] \\
&= \epsilon(1-\epsilon)^k\xi \\
&\ge (1-\epsilon)^{q_s}\epsilon\xi.
\end{aligned}$$

Moreover, $\mathcal{A}_4$ only succeeds if $s_i^* = 1$, which means that $h_i{}^* = e$ and σ^* is the signature of the message m^* indexed by i^*, then $(m_i{}^*; \mathsf{pk}; \sigma^*)$ must be a valid Diffie-Hellman tuple, so $\sigma^* = \left\{\frac{1}{s+e}\right\}^3 P$, which is indeed the solution of the modified **k-CAA** problem. As per the games, disscussed above the $\mathcal{A}$ can solve the modified **k-CAA** problem with probability $\epsilon' \ge (1-\epsilon)^{q_s}\epsilon\xi$.

4.2 Optimization and Conclusion

In this subsection, we want to optimize the parameter ξ to achieve a maximal probability of success. The function $(1-\xi)^{qs}\xi\epsilon$ is maximized at $\xi = \frac{1}{qs+1}$, where it has the value

$$\frac{1}{qs+1}.\left(1 - \frac{1}{qs+1}\right)^{qs}.\epsilon = \frac{1}{qs}.\left(1 - \frac{1}{qs+1}\right)^{qs+1}.\epsilon$$

So, the modified **k-CAA** problem can be solved by the $\mathcal{A}$ with probability $\epsilon' \ge \frac{1}{qs}.\left(1 - \frac{1}{qs+1}\right)^{qs+1}.\epsilon$

Next, we would estimate the time taken by $\mathcal{A}$ to solve the modified **k-CAA** problem. $\mathcal{A}$'s running time includes the running time of F. The additional overhead imposed by $\mathcal{A}$, is dominated by the need to search the list containing the tuples $\{m_i, e_i, \sigma_i\}$ for getting the corresponding signature, queried by F. Except the searching cost, no extra computation involved to generate the signature because the signatures are already given in the problem. We can assume constant amount time needed for each hash request from F as the hashed values are already given in the problem. Let us assume that the time needed for searching the list t_{search}. So, the total running time needed to answer as many as $(q_s + q_h)$ such requests, is

$$t^{'} \leq t + t_{search}.q_s + C.q_h + t_s,$$

where C, t_s are constant time to serve a hash query and running time of the simulator respectively.

5 Implementation Result

The proposed scheme has been implemented using Pairing based cryptography (PBC) library [15]. The explanation of the result of the proposed scheme is given below.
P = [192986486123713519393909328933523037284784
91004662265196503727693139637922709870433202758965
13457059148073430447824268885706106109906254206093
280693836, 501420448535073578989280727624093969333
85741208012663562069875078736229197761631701102288
66412462023685774069351431940207437204128961514477
050043811715742].

Let x = [1265908932634150451647717716712340277
08815422770] be the secret key.
P_{pub1} = [441689870023147090314403447339502824958611
68245371714845567562608664843497742265604064077942
57867578675786195471261396070944205290475705582841
7854661281237608698, 34360822137243754182196882839
97667670494305117850675397880306409364822200407239
96368559693036272553667711608621748740995578549186
17618318334875150086572529] be the first public key.

P_{pub2} = [19592480446279217949323193484764934053494
03661038358823669680250914482416056635807620995525
11498174693498774587900294239228651778055439102034
608839661404887, 494932513117050606495199878175923
63599436085377815538669761267040792935628973099520
76315362404051854588737494637770645919286622782851
288494074935292488790] be the second public key.

P_{pub3} = [177947893349060111593373570739170119977352
81536961967944958277308829161095598576985932011979
74932404668694060764315425265578143926947470957791
09360055160,73855654342439210919869820314375450311
00042376054573955814295886041128145544221416232106
67884397800716188170054984546984622552622105684703
117970371757588] be the third public key.

The message m is hashed using cryptographic hash function H and the message digest value $H(m)$ = 4418547219333555354423734373189812993762095987.
Signature of the message is: σ = [4346107938024
46962992888284367076304445172433417427304324796033
14873865704216658622752136882995707071770929985204
2269728915887678507962335880209573733453, 86895239
18563758982056666423922767096191922774023324509608
68772463768039101268313400354970675113261572051882
3046921034904851084360618485324305660338117047].

To verify, the message is hashed using cryptographic hash function H and generate the hashed message as: H(m) = 441854721793313555354423734373189812993762095987.

Compute the pairing:
$e(H(m)^3 P + P_{pub1} + P_{pub2} H(m) + P_{pub3} H(m)^2, \sigma)$ =
[1120539905030284386666594372752990165598444056724
45344618219640847132537270510436302325301680489741
914195924714355523808930098392282225166595052035468
24425, 8026017746651598938313304770558504235314467
20028089621862621602849870011382901646770550520983
49160063385074718510818542219066791183590504166149
112060847591].

Compute the pairing:
e(P,P) = [11205399050302843866665943727529901655984
44056724 45344618219640847132537270510436302325301
680489741 91419592471435552380893009839228222251665
9505203546 824425, 80260177499515989381330477055850
4235314467 2002808962186262160284987001138290164677
0550520983 4916006385074718510818542290667911835905
0416614911 2060847591].

From the above result, we can claim that the signature is valid.

5.1 Running Time Efficiency Comparison

We compare running time of our proposed scheme with other three established short signature schemes *i.e.* BLS [7], ZSS [19], Sedat [1]. All the schemes have been implemented using Pairing-Based Cryptography (PBC) library [15] in C on Linux systems with an Intel Core i3 CPU 2.13GHz and 6.00GB RAM. All schemes are different in the process of user-key-generation, signature-generation and the signature-verification. So, it is worth of giving a running time comparison of all the schemes, in the phases of key generation, signature generation and the signature verification. The running time and signature length of all the schemes can be seen in Table 3. The $|G_1|$ denotes the size of an element in the group G_1. For easy understanding, the results which is given in Table 3 have been represented by bar chart separately. Figure 1, Figure 2 and Figure 3 illustrate the the running time in the phases of user-key-generation, signature-generation and the signature-verification respectively.

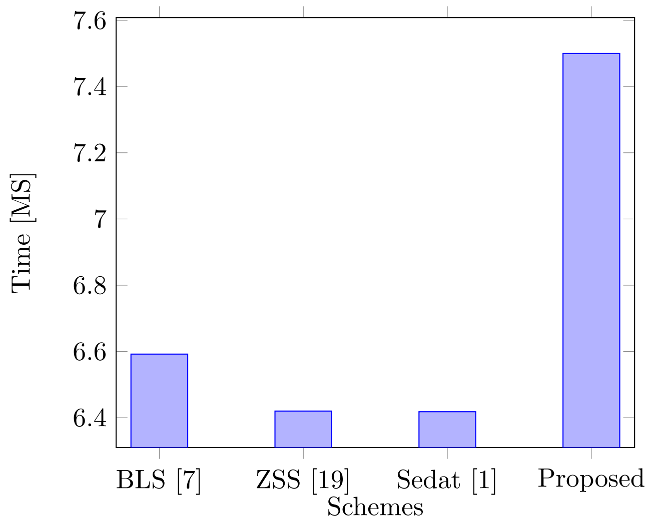

Figure 1: User Key Generation

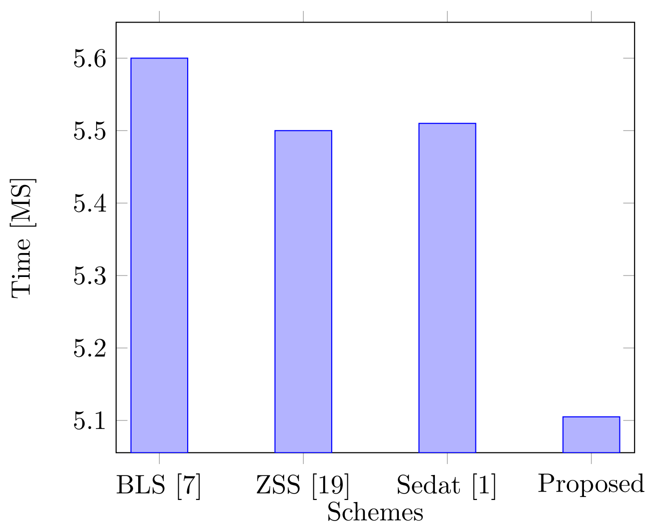

Figure 2: Signature Gneneration

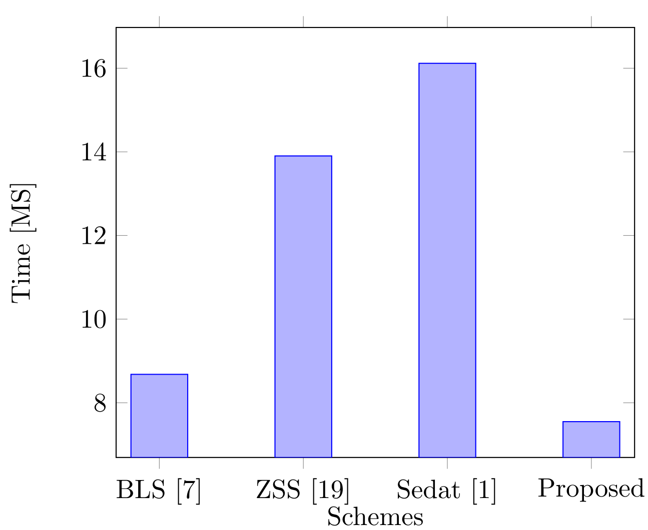

Figure 3: Signature Verification

Table 3: Comparison of the running time

Scheme	**Keygen (ms)**	**Sign (ms)**	**Verify (ms)**	**Signature Length**
BLS [7]	6.592	5.6	8.680	$\|G_1\|$
ZSS [19]	6.42	5.5	13.902	$\|G_1\|$
Sedat [1]	6.418	5.51	16.117	$\|G_1\|$
Proposed	7.5	5.105	7.549	$\|G_1\|$

Table 4: Operation notation and description

Notation	Description
τ_{po}	Execution of a bilinear pairing operation
τ_{inv}	Execution of an inversion in Z_q^*
τ_h	Execution of a hash function
τ_{p-add}	Execution of an point addition in G_1
τ_{squ}	Execution of a square operation in Z_q^*
τ_{cube}	Execution of a cube operation in Z_q^*
τ_{sm}	Execution of scalar multiplication in G_1
τ_{ec-add}	Execution of a elliptic curve point addition G_1
τ_{MTP}	Execution of Map to point hash function

6 Efficiency Analysis

Sometimes, relying on the running time is not up to the mark as it may be heavily affected by several factors such as the machine may be heavily loaded or lightly loaded at the execution time of the programs. So, it is worth of giving theoretical efficiency comparison of our proposed scheme. The various notations for time complexity of the operations involved in those schemes are given in the Table 4. The efficiency comparison of our proposed scheme with the scheme BLS [7], ZSS [19] and Sedat *et al.* [1] is shown in Table 5. In the proposed scheme, the value of $e(P, P)$ can be pre-computed. It can be claimed that, the signature verification process of the proposed scheme is constructed with only one bilinear pairing operations but the BLS [7] scheme has two bilinear pairing operations. In pairing based cryptographic scheme, it is well known that compare to other operations, pairing operation is the most time consuming operation. Instead of many attempts [4] to reduce the cost of pairing operation, still the pairing operation is very costly.

7 Conclusions

The scheme presented in this paper is based on bilinear pairing. The main advantage of our proposed scheme is that it does not require any special kind of hash function

Table 5: Efficiency Comparison

Schemes	Key-Generation	Signing	Verification
BLS [7]	$1\tau_{sm}$	$1\tau_{sm} + 1\tau_{MTP}$	$1\tau_{MTP} + 2\tau_{po}$
ZSS [19]	$1\tau_{sm}$	$1\tau_{sm} + 1\tau_h + \tau_{inv}$	$1\tau_{sm} + 1\tau_h + 1\tau_{po} + 1\tau_{p-add}$
Sedat Akley [1]	$2\tau_{sm} + 2\tau_{squ}$	$1\tau_h + 1\tau_{inv} + 1\tau_{squ} + 1\tau_{sm}$	$2\tau_{sm} + 1\tau_h + 1\tau_{squ} + 1\tau_{po} + 2\tau_{p-add}$
Proposed	$3\tau_{sm} + 1\tau_{squ} + 1\tau_{cube}$	$1\tau_{sm} + 1\tau_{cube} + 1\tau_h + 1\tau_{inv}$	$3\tau_{sm} + 1\tau_h + 1\tau_{cube} + 1\tau_{squ} + 1\tau_{po} + 3\tau_{p-add}$

such as map-to-point hash function. Any general cryptographic hash function such as MD5, SHA-I can be used for creating the hashed value from a massage. Moreover, our proposed scheme requires only one pairing operation where BLS [7] scheme requires two pairing operations in the process of signature verification.

References

[1] S. Akleylek, B. B. Kirlar, Ö. Sever, Z. Yüce, *Short Signature Scheme from Bilinear Pairings*, RTO-MP-IST-091, 2011.

[2] J. Alperin-Sheriff, "Short signatures with short public keys from homomorphic trapdoor functions," in *IACR International Workshop on Public Key Cryptography*, pp. 236–255, 2015.

[3] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid, "Recommendation for key management part 1: General (revision 3)," *NIST Special Publication*, vol. 800, no. 57, pp. 1–147, 2012.

[4] P. S. Barreto, B. Lynn, and M. Scott, "On the selection of pairing-friendly groups," in *International Workshop on Selected Areas in Cryptography (SAC'03)*, pp. 17–25, 2003.

[5] M. Bellare and G. Neven, "Multi-signatures in the plain public-key model and a general forking lemma," in *Proceedings of the 13th ACM Conference on Computer and Communications Security*, pp. 390–399, 2006.

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

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

[8] D. Boneh *et al.*, "Twenty years of attacks on the rsa cryptosystem," *Notices of the AMS*, vol. 46, no. 2, pp. 203–213, 1999.

[9] Federal Information Processing Standards Publication, *Digital Signature Algorithm (DSA)*, FIPS 186, May 19, 1994.

[10] Y. H. Hung, Y. M. Tseng, and S. S. Huang, "A revocable certificateless short signature scheme and its authentication application," *Informatica*, vol. 27, no. 3, pp. 549–572, 2016.

[11] D. Johnson, A. Menezes, and S. Vanstone, "The elliptic curve digital signature algorithm (ECDSA)," *International Journal of Information Security*, vol. 1, no. 1, pp. 36–63, 2001.

[12] A. Karati and G. P. Biswas, "Cryptanalysis and improvement of a certificateless short signature scheme using bilinear pairing," in *Proceedings of the International Conference on Advances in Information Communication Technology & Computing*, pp. 64–79, 2016.

[13] S. Mitsunari, R. Sakai, and M. Kasahara, "A new traitor tracing," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 85, no. 2, pp. 481–484, 2002.

[14] N. A. Moldovyan, "Short signatures from difficulty of factorization problem," *International Journal of Network Security*, vol. 8, no. 1, pp. 90–95, 2009.

[15] PBC Library, *The Pairing Based Cyptography*, Aug. 12, 2018. (`https://crypto.stanford.edu/pbc/`)

[16] C. P. Schnorr, "Efficient signature generation by smart cards," *Journal of Cryptology*, vol. 4, no. 3, pp. 161–174, 1991.

[17] J. L. Tsai, "A new efficient certificateless short signature scheme using bilinear pairings," *IEEE Systems Journal*, vol. 11, no. 4, pp. 2395–2402, 2017.

[18] R. Tso, X. Yi, and X. Huang, "Efficient and short certificateless signature," in *International Conference on Cryptology and Network Security (CANS'08)*, pp. 64–79, 2008.

[19] F. Zhang, R. Safavi-Naini, and W. Susilo, "An efficient signature scheme from bilinear pairings and its applications," *International Workshop on Public Key Cryptography (PKC'04)*, pp. 277–290, 2004.

[20] M. Zhang, B. Yang, Y. Zhong, P. Li, and T. Takagi, "Cryptanalysis and fixed of short signature scheme without random oracle from bilinear parings," *International Journal of Network Security*, vol. 12, no. 3, pp. 130–136, 2011.

Biography

Subhas Chandra Sahana was born at Bankura, India. He Received the B.Tech (bachelor degree) in Computer science and Engineering from Jalpaiguri Govt. Engineering College under West Bengal University of Technology. He got his M.Tech(IT) degree from Tezpur University , Assam and pursuing Ph.D. in North-Eastern Hill University. His research interest includes Cryptography and Network Security, Algorithm Analysis and Design, Information Theory and Coding etc.. Currently he is Assistant Professor in the department of Information Technology, North Eastern Hill University, Shillong, Meghalaya, India.

Dr. Bubu Bhuyan was born in India. He received his M.Tech (IT) and Ph.D. degree from Tezpur University and Jadavpur University respectively. His research interest includes Cryptography and Network Security, Algorithm Analysis and Design, Information Theory and Coding etc.. Currently he is Associate Professor in the department of Information Technology, North-Eastern Hill University, Shillong, Meghalaya, India.