Provably Secure Online/Off-line Identity-Based Signature Scheme for Wireless Sensor Network

Jayaprakash Kar

Department of Information Systems
Faculty of Computing & Information Technology
King Abdulaziz University, Kingdom of Saudi Arabia
{jayaprakashkar, jpkar.crypto }@yahoo.com

Abstract. This paper describes an efficient and secure online and off-line signature scheme for wireless sensor network (WSN). Security of the proposed scheme is based on difficulty of breaking Bilinear Diffie-Hellman problem (BDHP). WSN systems are usually deployed in hostile environments where they encounter a wide variety of malicious attacks. Information that is the cooked data collected within the sensor network, is valuable and should be kept confidential. In order to protect this transmitted information or messages between any two adjacent sensor nodes, a mutual authentication and key establishment protocol is required for wireless sensor networks. Because some inherent restrictions of sensor nodes which include low power, less storage space, low computation ability and short communication range most existing protocols attempt to establish a pairwise key between any two adjacent sensor nodes by adopting a key pre-distribution approach. In order to further reduce the computational cost of signature generation, online/off-line is suitable for WSN. In on-line/off-line signature scheme, the signing process can be broken into two phases. The first phase, performed off-line, is independent of the particular message to be signed; while the second phase is performed on-line, once the message is presented.

Keywords: Bilinear Pairing,BDHP, multi-signature, online/off-line.

1 Introduction

Wireless sensor networks consist of small nodes also called motes that monitor physical or environmental conditions around them such as temperature, sound, vibration etc. It process data, and communicate through wireless links [4]. A wireless sensor network (WSN) generally consists of a base station, which holds the ability to communicate with a number of wireless sensors present nearby by use of a radio link. Once the data is collected by some intermediate node, it is then compressed, and transmitted to the gateway directly or, if not directly connected then uses other wireless sensor nodes to forward data to the gateway. Once this data reaches at the base-station then it is presented to the system by the gateway connection [5]. Wireless Sensor Networks are widely used these days and are very popular in research for use of embedded systems in our daily life. WSNs are used in applications involving monitoring, tracking, or controlling such as habitat monitoring, robotic toys, battlefield monitoring, packet insertion [6], traffic monitoring, object tracking and nuclear reactor control.

2 Preliminaries

2.1 Notation

Definition 1. Bilinearity *Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic groups of same prime order q. $\mathbb{G}_1$ is an additive group and $\mathbb{G}_2$ is a multiplicative group. Let e be a computable bilinear map $e : \mathbb{G}_1 X \mathbb{G}_1 \to \mathbb{G}_2$, which satisfies the following properties:*

- **Bilinear**: $e(aP, bQ) = e(P,Q)^{ab}$, *where* $P, Q \in \mathbb{G}_1$ *and* $a, b \in \mathbb{Z}_q^*$ *and for* $P, Q, R \in \mathbb{G}_1, e(P+Q, R) = e(P,R)e(Q,R)$.
- **Non-degenerate**: *If* P *is a generator of* $\mathbb{G}_1$, *then* $e(P,P)$ *is generator of* $\mathbb{G}_2$. *There exists* $P, Q \in \mathbb{G}$ *such that* $e(P,Q) \neq 1_{\mathbb{G}_2}$
- **Computability**: *There exists an efficient algorithm to compute* $e(P,Q)$ *for all* $P, Q \in \mathbb{G}_1$.

We call such a bilinear map e is an admissible bilinear pairing.

2.2 Mathematical Assumption

Definition 2. Bilinear Parameter Generator : *A bilinear parameter generator* $\mathcal{G}$ *is a probabilistic polynomial time algorithm that takes a security parameter* k *as input and outputs a* 5*-tuple* $(q, \mathbb{G}_1, \mathbb{G}_2, e, P)$ *as the bilinear parameters, including a prime number* q *with* $|q| = k$, *two cyclic groups* $\mathbb{G}_1, \mathbb{G}_2$ *of the same order* q, *an admissible bilinear map* $e : \mathbb{G}_1 X \mathbb{G}_1 \rightarrow \mathbb{G}_2$ *and a generator* P *of* $\mathbb{G}_1$

Definition 3. Bilinear Diffie-Hellman Problem: *Let* $(q, \mathbb{G}_1, \mathbb{G}_2, e, P)$ *be a* 5*-tuple generated by* $\mathcal{G}(k)$, *and let* $a, b, c \in \mathbb{Z}_q^*$. *The BDHP in* $\mathbb{G}$ *is as follows: Given* (P, aP, bP, cP) *with* $a, b, c \in \mathbb{Z}_q^*$, *compute* $e(P,P)^{abc} \in \mathbb{G}_T$. *The* (t, ϵ) *-BDH assumption holds in* $\mathbb{G}$ *if there is no algorithm* $\mathcal{A}$ *running in time at most* t *such that*

$$\mathbf{Adv}_{\mathbb{G}}^{BDH}(\mathcal{A}) = Pr[\mathcal{A}(P, aP, bP, cP) = e(P,P)^{abc}] \geq \epsilon$$

where the probability is taken over all possible choices of (a, b, c). Here the probability is measured over random choices of $a, b, c \in \mathbb{Z}_q^*$ and the internal random operation of A. More formally, for any PPT algorithm $\mathcal{A}$ consider the following experiment:
Let $\mathcal{G}$ be an algorithm which on input 1^k outputs a (description of a) group G of prime order q (with $|q| = k$) along with a generator $P \in \mathbb{G}$. The computational Diffie-Hellman (CDH) problem is the following:

$\underline{\mathbf{Exp}_{\mathcal{G}(k)}^{CDH}}$

1. $(\mathbb{G}, q, P) \leftarrow \mathcal{G}(1^k)$
2. $a, b, c \leftarrow \mathbb{Z}_q^*$
3. $U_1 = aP, U_2 = bP, U_3 = cP$
4. if $W = e(P,P)^{abc}$ return 1 else return 0

We assume that BDHP is a hard computational problem: letting q have the magnitude $2k$ where k is a security parameter, there is no polynomial time (in k) algorithm which has a non-negligible advantage (again, in terms of k) in solving the BDHP for all sufficiently large k.

Definition 4. Decisional Diffie-Hellman Problem : *Let* $(q, \mathbb{G}, \mathbb{G}_T, e, P)$ *be a* 5*-tuple generated by* $\mathcal{G}(k)$, *and let* $a, b, c, r \in \mathbb{Z}_q^*$. *The DBDHP in* $\mathbb{G}$ *is as follows: Given Given* (P, aP, bP, cP, r) *with some* $a, b, c \in \mathbb{Z}_q^*$, *Output is* **yes** *if* $r = e(P,P)^{abc}$ *and* **no** *otherwise. The* (t, ϵ)*-HDDH assumption holds in* $\mathcal{G}$ *if there is no algorithm* $\mathcal{A}$ *running in time at most* t *such that*

$$\mathbf{Adv}_{\mathbb{G}}^{DBDH}(\mathcal{A}) = |Pr[\mathcal{A}(P, aP, bP, cP, e(P,P)^{abc})) = 1] - Pr[\mathcal{A}(P, aP, bP, cP, r) = 1]| \geq \epsilon$$

where the probability is taken over all possible choices of (a, b, c, h).

Definition 5. Hash Decisional Diffie-Hellman Problem :*Let* $(q, \mathbb{G}, \mathbb{G}_T, e, g)$ *be a* 5*-tuple generated by* $\mathcal{G}(k)$,$\mathcal{H} : \{0,1\}^* \rightarrow \{0,1\}^l$ *be a secure cryptographic hash function, whether* l *is a security parameter, and let* $x, y \in \mathbb{Z}_q^*, h \in \{0,1\}^l$, *the HDDH problem in* $\mathbb{G}$ *is as follows: Given* (P, aP, bP, cP, h), *decide whether it is a hash Diffie-Hellman tuple* $((P, aP, bP, cP\mathcal{H}(e(P,P)^{abc}))$. *If it is right, outputs* 1*; and* 0 *otherwise. The* (t, ϵ)*-HDDH assumption holds in* $\mathcal{G}$ *if there is no algorithm* $\mathcal{A}$ *running in time at most* t *such that*

$$\mathbf{Adv}_{\mathbb{G}}^{HDDH}(\mathcal{A}) = |Pr[\mathcal{A}(P, aP, bP, cP\mathcal{H}(e(P,P)^{abc})) = 1] - Pr[\mathcal{A}(P, aP, bP, cP, h) = 1]| \geq \epsilon$$

where the probability is taken over all possible choices of (a, b, h).

3 Online/Off-line ID based Signature Scheme

Online/Off-line signature schemes [1] divide the process of message signing into two phases, the Off-line phase and the Online phase. The Off-line phase, which consists of complex computations are performed before the message to be signed becomes available. Once the message is known, the Online phase starts. This phase retrieves the partial signature calculated during the Off-line phase and performs some minor quick computations to obtain the final signature. The Online phase is assumed to be very fast, consisting of small computations. The Off-line phase can be performed by a resourceful device. Online/Off-line allows a resource constrained sensor node to sign a message quickly, once it has some critical event to report.

4 Framework of ID-based online/Off-line Signature Scheme(IBS)

An ID-based online/off-line signature(IBS) scheme comprises the following five probabilistic polynomial time (PPT) algorithms:

- **Setup**: $(param, msk) \leftarrow \mathrm{Set}(1^k)$ takes a security parameter $k \in \mathbb{N}$ and generates *param*, the global public parameters and *msk*, the master secret key of the KGC.
- **Extract**: $D_{ID} \leftarrow Ext(1^k, param, msk, ID)$ takes a security parameter k, the global parameters *param*, a master secret key *msk* and an identity ID to generate a secret key D_{ID} corresponding to this identity.
- **Off-lineSign**: $\sigma_{off} \leftarrow Sgn_{off}(1^k, param)$ takes a security parameter k and the global parameters *param* to generate an off-line signature σ_{off}.
- **OnlineSign**: $\sigma_{on} \leftarrow Sgn_{on}(1^k, param, m, \sigma_{off}, ID)$ takes a security parameter k, the global parameters *param*, a message m, an off-line signature σ_{off}, an identity ID to generate a signature σ.
- **Verify**: $(\text{“}accept\text{”}, \text{“}Reject\text{”}) \leftarrow Ver(1^k, param, \sigma_{off}, D_{ID})$ takes a security parameter k, the global parameters *param*, a signature σ, a secret key of the receiver D_{ID} to generate the outputs “*accept*” if σ is valid and outputs “*reject*” otherwise.

5 Previous Work

In modern cryptography, the notion of digital signature is one of the most fundamental and useful goal. Since the public key cryptography was introduced, various signature schemes have been proposed to meet various requirements in practical circumstances. In order to reduce the computational cost of signature generation, the notion of on-line/off-line signatures was first introduced by Even, Goldreich and Micali [1] in 1989. It is used in a particular scenario where the signer must respond quickly once the message to be signed is presented. The idea is to perform the signature generating procedure in two phases. The first phase is performed off-line (without knowing the signed message) and the second phase is performed online (after knowing the signed message). The on-line phase should be very fast and require only very light computation, such as integer multiplication or hashing. Other heavier computation such as exponentiation should be avoided in the online phase. This is the basic characteristic of online/off-line signature schemes. In this way, online/off-line signature schemes are particularly useful for low-power devices such as smart card, wireless sensor network (WSN) application.

Shi et al. proposed an efficient identity based signature scheme [11] with batch verification. Though the scheme in [11] achieves efficiency in computation with just two pairing operations and linear exponentiation operations, it is required to pass all the signatures separately and hence increases the communication complexity. Also a universal forgery of the signature of any singer is possible in this scheme as shown in [3].

Wang et al designed an identity based aggregate signature [14] and it is claimed to be the most efficient scheme. It uses constant pairing operation for signature verification. But the aggregate signature in this scheme [14] is not secure since universal forgery of signature of any user is possible in this scheme. Also, the scheme achieves only partial aggregation. The attack in Wang et al. scheme [14] is shown in [3]. Xiangguo et al. gave a aggregate signature scheme [12] which uses the BLSR scheme [7] as the base signature scheme. In this scheme all the signers have to broadcast their own random values used for singing to all the co-signers so that everyone agrees upon a common randomness before the generation of aggregate signature. This results in quadratic communication complexity which is a big overhead. Mutual interaction between all the signers is not a desirable step in aggregate signatures. Hyo et al. gave a number of batch verification techniques [10]. During verification it requires linear number of pairings which also increases the computation complexity considerably. Yiling et al. proposed an efficient aggregate signature scheme with full aggregation and constant pairing operations in [8].

Javier Herranz came up with an identity based signature scheme [20] with partial aggregation. But his scheme produces deterministic signature where the signature component on a message will always be the same. This is a major draw back in real world scenarios. It also uses linear number of pairing operations leading to inefficiency in computation.

Xu et al. in [9] proposed an identity based aggregate signature scheme. This scheme uses Sakai et al. signature construct as the base signature scheme. This achieves only partial aggregation and also requires linear number of pairings during signature verification. Gentry and Ramzan proposed an efficient identity based aggregate signature scheme [22]. This scheme achieves both full aggregation and also constant number of pairing operations during signature verification. But the scheme in [22] has certain weaknesses which makes it unsuitable for real life scenarios. The weaknesses of the scheme are briefly reported in the appendix. Boldyreva et al. proposed an identity based sequential signature scheme [13]. Hwang et al. in [15] proposed an attack on [13] and claimed that the only existing efficient aggregate signature scheme is of Gentry and Ramzan [22] which involves interaction between all the signers whose signatures are to be aggregated. The design of an efficient identity based aggregate signature scheme without any interaction between the signers was left open by Hwang et al. [15].

6 Security Model

The attacks against signature schemes are of without message attack and chosen-message attack. The strong one is an adaptive chosen-message attack. In this scenarios the attacker can ask the signer to sign any message that he/she chooses. He also knows the public key of the signer. Then he can customize his queries according to the previous message .

Definition 6. *An identity-based online/off-line signature is defined by the four tuples $\mathcal{IBS} = (Setup, Extract, OfflineSign, OnlineSign)$ is said to be existentially unforgeable under chosen-message attacks if no probabilistic polynomial time adversary has a non-negligible advantage in the following game played between a challenger C and an adversary $\mathcal{A}$.*

- **Setup** The challenger C runs this algorithm to generate the system parameters and sends to the adversary $\mathcal{A}$.
- The adversary $\mathcal{A}$ performs the following queries adaptively:
  1. **Key Extraction Oracle**: when $\mathcal{A}$ requests the private key on an identity ID, C runs the Extract algorithm to obtain the D_{ID} and returns to the adversary $\mathcal{A}$.
  2. **Off-line Signing Oracle**: when $\mathcal{A}$ requests the off-line signature on an identity ID, C runs the Off-Sign algorithm to obtain the σ_{off} and returns to the adversary $\mathcal{A}$.
  3. **Online Signing Oracle**: when $\mathcal{A}$ requests the online signature on the message m for an identity ID, C runs the On-Sign algorithm to obtain the σ_{on} and returns to the adversary $\mathcal{A}$.
- After a polynomial number of queries, $\mathcal{A}$ outputs a signature $(ID^*, m^*, \sigma^*_{off}, \sigma^*_{on})$ such that
  1. ID^* has been requested as one of the key extraction queries.

2. (ID^*, m^*) has not been requested as one of the off-line signing queries and online signing queries.
3. $(ID^*, m^*, \sigma^*_{off}, \sigma^*_{on})$ is a valid ID-based online/off-line signature.

The success probability of an adversary $\mathcal{A}$ wins the above game is defined by

$$Suss_{\mathcal{A}}^{EF-\mathcal{IBS}-CMA}(k) \leq \frac{1}{2} + \epsilon.$$

ϵ is called advantage for the adversary in the above game.

Definition 7. *$\mathcal{A}$ win the game if σ^* is a valid signature of m^*. An adversary is said to be an $(\epsilon, t, q_e, q_s, q_h)$-forger if it has advantage at least ϵ in the above game, run in time at most t, and make at most q_e, q_s and q_h extract, signing and random oracle queries, respectively.*

An identity based online/off-line signature scheme $(\epsilon, t, q_e, q_s, q_h)$ is secure if no $(\epsilon, t, q_e, q_s, q_h)$-forger exists.

7 Liu et al.'s Online/Off-line ID based Signature for WSN

The scheme consists of the following 5 phases.

- **Setup**: Let $\mathbb{G}$ be a multiplicative group of prime order q. The $\mathcal{PKG}$ selects a random generator $g \in \mathbb{G}$ and randomly chooses $x \in \mathbb{Z}_q^*$ at random. It sets $X = g^x$. Let $H : \{0,1\} \rightarrow \mathbb{Z}_q^*$ be a cryptographic hash function. The public parameters *param* and master secret key *msk* are given by $param = (\mathbb{G}, q, g, X, H), msk = x$.
- **Extract**: To generate a secret key for identity ID, the $\mathcal{PKG}$ randomly selects $r \in \mathbb{Z}_q^*$ at random, computes $R \leftarrow g^r$ $s \leftarrow r + H(R, ID)x \bmod q$. The user secret key is (R, s). Note that a correctly generated secret key should fulfill the following equality:
$$g^s = RX^H(R, ID) \tag{1}$$
- **Off-line Sign**: At the off-line stage the signer computes: $\hat{Y}_i \leftarrow g^{2^i} \forall i = 0, 1 \ldots |q| - 1$.
- **Online Sign**: At the online stage, the signer randomly selects $y \in \mathbb{Z}_q^*$. Let $y[i]$ be the i^{th} bit of y. Define $\mathcal{Y} \subset \{1, 2 \ldots |q|\}$ to be the set of indices such that $y[i] = 1$. Computes $Y \leftarrow \prod_{i \in \mathcal{Y}} \hat{Y}_{i-1}, h \leftarrow H(Y, R, m), z \leftarrow y + hs \bmod q$. The signature is (Y, R, z).
- **Verification** The signature is valid only if the following equation holds
$$g^z = YR^h X^{(R,ID)} \tag{2}$$

7.1 Vulnerability in the Scheme

At the off-line stage the signer computes: $\hat{Y}_i \leftarrow g^{2^i} \forall i = 0, 1 \ldots |q| - 1$ and at the online stage, the signer randomly selects $y \in \mathbb{Z}_q^*$. Let $y[i]$ be the i^{th} bit of y. Define $\mathcal{Y} \subset \{1, 2 \ldots |q|\}$ to be the set of indices such that $y[i] = 1$. Computes $Y \leftarrow \prod_{i \in \mathcal{Y}} \hat{Y}_{i-1}, h \leftarrow H(Y, R, m), z \leftarrow y + hs \bmod q$. For computation of Y, let us consider the following three cases.

- **Claim-1**: The position of 1 in the string is in odd or even place *i.e* alternately 1s. Consider the random number $y \in \mathbb{Z}^*$ of length 6 in binary 101010. Here the set $\mathcal{Y} = \{1, 3, 5\}$ which is in a proper sequence, we can compute $Y = \hat{Y}_0\hat{Y}_2\hat{Y}_4$. So it is easy for an attacker to compute Y which is the partial signature (Y, R, z). Therefore it can be forged.
- **Claim-2**:The string contains all 1s. Consider y of length 6 in binary 111111. Set $\mathcal{Y} = \{0, 1, 2, 3, 4, 5\}$. Also the attacker can compute directly as $Y = \hat{Y}_0\hat{Y}_1\hat{Y}_2\hat{Y}_3\hat{Y}_4\hat{Y}_5$.
- **Claim-3**:If the string contains all 0s, then it is not possible to computes Y which is understood for an attacker that Y consists of all 0's is having 0s only.

The proposed scheme has not suggested for all these above cases.

8 Proposed Online/Off-line ID based Signature on Bilinear Pairings

We have considered all the above cases and suggested a provably secure scheme on random oracles. The scheme comprises the following five PPT algorithms.

- **Setup** Given security parameters k, the $\mathcal{PKG}$ chooses groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order q. A generator P of $\mathbb{G}_1$, a bilinear map $\hat{e} : \mathbb{G}_1 X \mathbb{G}_1 \to \mathbb{G}_2$ and collision resistant hash function $\mathcal{H}_0 : \{0,1\}^* \to \mathbb{G}_1$, $\mathcal{H}_1 : \{0,1\}^* \to \mathbb{Z}_q^*$. It chooses a master-key $s \in \mathbb{Z}_q^*$ and computes $P_{pub} = sP$. the system public parameters are given by

$$\mathcal{P} = (\mathbb{G}_1, \mathbb{G}_2, q, \hat{e}, P, P_{pub}, \mathcal{H}_0, \mathcal{H}_1)$$

- **Extract** This algorithms follows of the following steps
  - Given an identity $ID \in \{0,1\}^*$ the secret key is $d_{ID} = s \cdot Q_{ID}$, where $Q_{ID} = \mathcal{H}_0(ID)$.
- **Off-line Sign**: During this phase, the signer computes the followings
  - $\hat{\alpha} i = \hat{e}(P, P_{pub})^{2^i}, \forall i = 0, 1 \ldots |q| - 1$. During this off-line phase we neither require the knowledge of the message nor the secret key.
- **Online Sign**: During this phase the signer follows the following steps
  - Select randomly $\beta \in \mathbb{Z}_q^*$. Let $\beta[i]$ be the i^{th} bit of β.
  - Define $\mathcal{D} \subset \{1, 2 \ldots |q|\}$ be the set of indices such that $\beta[i] = 1$ and $C \subset \{1, 2 \ldots |q|\}$ be the set of indices such that $\beta[i] = 0$.
  - Computes $\psi_1 = \prod_{i \in \mathcal{D}} \hat{\alpha}_{i-1}$ and $\psi_2 = \prod_{i \in C} \hat{\alpha}_{i-1}$.
  - Computes $\alpha = \psi_1 \psi_2$
  - Selects $\gamma \in \mathbb{Z}_q^*$, computes $U = \gamma \cdot P$, $r = \mathcal{H}_1(ID, U \| m)$
  - $V = (\gamma + \beta) \cdot P_{pub} + r d_{ID}$.

  The signature is $\sigma = (\alpha, U, V)$
- **Verify** : To verify the signature (α, U, V) for the message m and the identity ID, the verifier follows the following steps.
  - Computes $r = \mathcal{H}_1(ID, U \| m)$
  - Checks whether the following equation holds

$$\hat{e}(V, P) \stackrel{?}{=} \alpha \hat{e}(Q_{ID}, P_{pub})^r \hat{e}(U, P_{pub}) \tag{3}$$

9 Proof of Correctness

First computes $r = \mathcal{H}_1(ID, U \| m)$ and verify the following equation

$$\hat{e}(V, P) \stackrel{?}{=} \alpha \hat{e}(Q_{ID}, P_{pub})^r \hat{e}(U, P_{pub})$$

Also check

$$\alpha = \hat{e}(P, P_{pub})^\beta$$

$$\alpha = \hat{\alpha}_0 \hat{\alpha}_1 \ldots$$

$= \hat{e}(P, P_{pub})^{2^0} \hat{e}(P, P_{pub})^{2^1} \ldots$

$= \hat{e}(P, (2^0 + 2^1 + \ldots) P_{pub})$

$= \hat{e}(P, \beta P_{pub}) = \hat{e}(P, P_{pub})^\beta$

$$\hat{e}(V,P) = \hat{e}((\gamma+\beta)P_{pub} + rd_{ID})$$

$$= \hat{e}((t+\beta)P_{pub}, P)\hat{e}(rd_{ID}, P)$$

$$= \hat{e}(P_{pub}, (t+\beta)P)\hat{e}(rd_{ID}, P)$$

$$= \hat{e}(P_{pub}, tP)\hat{e}(P_{pub}, \beta P)\hat{e}(rd_{ID}, P)$$

$$= \hat{e}(P_{pub}, U)\hat{e}(P_{pub}, P)^{\beta}\hat{e}(rd_{ID}, P)$$

$$= \alpha\hat{e}(P_{pub}, U)\hat{e}(rd_{ID}, P)$$

$$= \alpha\hat{e}(P_{pub}, U)\hat{e}(rsQ_{ID}, P)$$

$$= \alpha\hat{e}(P_{pub}, U)\hat{e}(rQ_{ID}, sP)$$

$$= \alpha\hat{e}(P_{pub}, U)\hat{e}(rQ_{ID}, P_{pub})$$

$$= \alpha\hat{e}(P_{pub}, U)\hat{e}(Q_{ID}, P_{pub})^{r}$$

10 Security and Performance Analysis

The computational cost if this proposed scheme is as follows:

- The sender needs to compute a point multiplication, a pairing evaluation, an encryption, as well as a hash evaluation. In addition, the most expensive computation is to be perform is the use of a public-key digital signature algorithm.
- Since the receiver and the sender stand in the symmetric position, so the receiver shares the same computation costs. The communication cost of the proposed protocol is that the sender and the receiver carry out two rounds for communications in order for the receiver to obtain a message from the sender.

For practical implementation, we can use some existing tools for these computations including point multiplication, bilinear pairing evaluation, and hash function evaluation over elliptic curves. The protocol is based on the elliptic curve cryptography (ECC) and thus it has high security complexity with short key size.

10.1 Security Analysis

Theorem 1 *In random oracle model, Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be (ϵ^*, t^*)-BDH two cyclic group of same prime order q. P be a generator of $\mathbb{G}_1$. $\mathbb{G}_1$ is an additive group and $\mathbb{G}_2$ is a multiplicative group. Let e be a computable bilinear map $e : \mathbb{G}_1 X \mathbb{G}_1 \rightarrow \mathbb{G}_2$. The proposed scheme is $(\epsilon, t, q_e, q_s, q_h)$-secure with respect to EF-$\mathcal{IBS}$-CMA, where for any t and ϵ satisfying*

$$\epsilon \geq e(q_e+1)\epsilon^*, t \leq t^* - t_m(2q_{\mathcal{H}_0} + q_e + 4q_s)$$

where e is the base of the natural logarithm, and T is the time of computing a scalar multiplication in $\mathbb{G}_1$ and an inversion in $\mathbb{Z}_q^*$. and $q_e, q_s, q_{\mathcal{H}_0}$ are the number of extraction, signing and hashing queries respectively the adversary is allowed to make and t_m is the time for computing scalar multiplication.

Proof: Suppose that $\mathcal{A}$ is a forger who breaks the scheme $\mathcal{IBS}$. A CDH instance (P, xP, yP) is given for $x, y \in \mathbb{Z}_q^*$. By using the forgery algorithm $\mathcal{A}$, we will construct an algorithm $\mathcal{B}$ which outputs the CDH solution xyP in $\mathbb{G}_1$. This performs the following simulation by interacting with the forger $\mathcal{A}$.

- **Setup**: Algorithm $\mathcal{B}$ sets $P_{pub} = xP$ and starts by giving $\mathcal{A}$ the system parameters *param* including (P, P_{pub}). At any time, $\mathcal{A}$ can make query on the random oracles $\mathcal{H}_0$ $\mathcal{H}_1$ and **Extract** and **Sign** queries. To answer these queries, $\mathcal{B}$ does the following:
- **Queries on Oracle** $\mathcal{H}_0$: To respond $\mathcal{H}_0$-queries, $\mathcal{B}$ maintains a list of tuples (ID, ω, b, c) denoted by L_0. When $\mathcal{A}$ queries the oracle $\mathcal{H}_0$ at a point $ID \in \{0, 1\}^*$, $\mathcal{B}$ responds as follows:
  1. If the query ID already appears on the L_0-list in a tuple (ID, ω, b, c) then $\mathcal{B}$ responds with $\mathcal{H}_0(ID) = \omega \in \mathbb{G}_1$.
  2. Otherwise, $\mathcal{B}$ picks a random coin $c \in \{0, 1\}$ with $Pr[c = 0] = \frac{1}{(q_e+1)}$.
     - If $c = 0$ then $\mathcal{B}$ computes $\omega = b(yP)$ for a random $b \in \mathbb{Z}_q^*$.
     - If $c = 1$ then $\mathcal{B}$ computes $\omega = bP$ for a random $b \in \mathbb{Z}_q$.

     $\mathcal{B}$ adds the tuple (ID, ω, b, c) to the L_0-list and responds to $\mathcal{A}$ with $\mathcal{H}_0(ID) = \omega$.
- **Queries on Oracle** $\mathcal{H}_1$ To respond to $\mathcal{H}_1$-queries, $\mathcal{B}$ maintains a list of tuples (ID, U, m, r) denoted by L_1. When $\mathcal{A}$ queries the oracle $\mathcal{H}_1$ at (ID, U, m), $\mathcal{B}$ responds as follows:
  1. If the query (U, m) already appears on the L_1-list in a tuple (ID, m, U, r) then $\mathcal{B}$ responds with $\mathcal{H}_1(ID, U\|m) = r \in \mathbb{Z}_q^*$.
  2. Otherwise, $\mathcal{B}$ picks a random $r \in \mathbb{Z}_q^*$ and adds the tuple (ID, m, U, r) to the L_1-list and responds to $\mathcal{A}$ with $\mathcal{H}_1(ID, U\|m) = r$.
- **Extract Queries**: When $\mathcal{A}$ queries the private key corresponding to ID, $\mathcal{B}$ first finds the corresponding tuple (ID, ω, b, c) from the L_0-list:
  1. If $c = 0$ then $\mathcal{B}$ fails and halts.
  2. Otherwise, $\mathcal{B}$ computes $d_{ID} = b \cdot P_{pub} = b(xP)$ by using the tuple (ID, ω, b, c) in the $\mathcal{H}_0$-list and responds to $\mathcal{A}$ with d_{ID}.
- **Off-line & Online Signing Query**: Suppose $\mathcal{A}$ makes queries q_s a off-line and online signing algorithms on message m with the signer's identity ID , $\mathcal{B}$ finds the corresponding tuple (ID, ω, b, c) from the $\mathcal{H}_0$-list and chooses a random $r, \gamma, \beta \in \mathbb{Z}_q^*$ and computes $U = \gamma rP - \beta P - r\omega$. If the tuple (ID, m, U, r) already appears on the L_1-list, $\mathcal{B}$ chooses another $r, \gamma, \beta \in \mathbb{Z}_q^*$ and tries again. Otherwise, $\mathcal{B}$ computes $V = r \cdot \gamma P_{pub}$ and stores (ID, m, U, r) in the $\mathcal{H}_1$-list. Then, $\mathcal{B}$ responds to $\mathcal{A}$ with $\sigma = (\alpha, U, V)$. All responses to Sign queries are valid; indeed, the output (α, U, V) of Sign query is a valid signature on m for ID, to see this,
  $\alpha\hat{e}(Q_{ID}, P_{pub})^r\hat{e}(U, P_{pub})$
  $= \hat{e}(P, P_{pub})^\beta\hat{e}(Q_{ID}, P_{pub})^r\hat{e}(U, P_{pub})$
  $= \hat{e}(\beta P + rQ_{ID} + U, P_{pub}$
  $= \hat{e}(\beta P + r\omega + U, P_{pub}) = \hat{e}(\gamma rP, P_{pub})$
  $= \hat{e}(\gamma rP_{pub}, P) = e(V, P)$.
- **Output** If $\mathcal{B}$ does not abort as a result of $\mathcal{A}$'s Extract query then $\mathcal{A}$'s view is identical to its view in the real attack. By Forking Lemma (Pointcheval and Stern, 2000), after replaying $\mathcal{A}$ with the same random tape, $\mathcal{B}$ obtains two valid signatures $\sigma = (ID^*, m, r, \alpha, U, V)$ and $\sigma' = (ID^*, m, r', \alpha, U, V')$ within a polynomial time, where $V = (\gamma + \beta) \cdot P_{pub} + rd_{ID^*}$ and $V' = (\gamma + \beta) \cdot P_{pub} + rd_{ID^*}$. Then $\mathcal{B}$ finds the corresponding tuple (ID, R) from the list L_0, if $c = 1$ then $\mathcal{B}$ fails and halts. Otherwise, $\mathcal{B}$ computes $V - V' = (r - r')d_{ID^*}$. Finally, $\mathcal{B}$ outputs xyP as a solution to the CDH instance by computing $(r - r')(r - r')^{-1}sQ_{ID^*} = xyP$.
- **Probability Analysis**: $\mathcal{B}$ solves the given instance of the CDH problem with probability at least ϵ^*. We analyze three independent events needed for $\mathcal{B}$ to succeed:
  - E_1 : $\mathcal{B}$ does not abort as a result of any $\mathcal{A}$'s Extract query.
  - E_2 : $\mathcal{A}$ generates a valid and non-trivial signature forgery $\sigma = (\alpha, U, V)$ on m for ID.
  - E_3 : Event E_2 occurs and $c = 0$ for tuples containing ID on the L_0-list.

$\mathcal{B}$ succeeds if all these events happen. The probability is $Pr[E_1 \wedge E_2 \wedge E_3]$

$$Pr[E_1 \wedge E_2 \wedge E_3] = Pr[E_1] \cdot Pr[E_2 \mid E_1] \cdot Pr[E_3 \mid [E_1 \wedge E_2]$$

- **Claim 1** The probability that $\mathcal{A}$ does not abort as a result of $\mathcal{A}$'s Extract query is at least $(1 - \frac{1}{q_e+1})^{q_e}$.
- **Claim 2** If $\mathcal{A}$ does not abort as a result of $\mathcal{A}$'s Extract query than $\mathcal{A}$'s view is identical to its view in the real attack. Hence, $Pr[E_2 \mid E_1] \geq \epsilon$.
- **Claim 3** The probability that $\mathcal{B}$ does not abort after $\mathcal{A}$ outputs a valid and nontrivial forgery is at least $(\frac{1}{(q_e+1)}$. Algorithm $\mathcal{B}$ will abort only if $\mathcal{A}$ generates a forgery such that $c = 1$. Hence $Pr[E_3 \mid E_1 \wedge E_2] \geq \frac{1}{(q_e+1)}$

$$\text{So } Pr[E_1] \cdot Pr[E_2 \mid E_1] \cdot Pr[E_3 \mid [E_1 \wedge E_2] \geq (1 - \frac{1}{q_e+1})^{q_e} \cdot \epsilon \cdot \frac{1}{(q_e+1)} \geq \frac{1}{e} \cdot \frac{\epsilon}{(q_e+1)} \geq \epsilon^*.$$

Algorithm $\mathcal{B}$'s running time is equal to the sum of the running time of s $\mathcal{A}$'s and the time it takes to respond to $q_{\mathcal{H}_0}$ hash oracle $\mathcal{H}_0$ queries, q_e key extract queries, and q_s online/off-line signature queries. Each $\mathcal{H}_0$ query requires one scalar multiplications in $\mathbb{G}_1$. Each key extract query needs one scalar multiplication in $\mathbb{G}_1$. Each online/off-line signature requires 4 scalar multiplications in $\mathbb{G}_1$. If we assume one scalar multiplications in $\mathbb{G}_1$ takes time t_m, the total running time is at most $t + (2q_{\mathcal{H}_0} + q_e + 4q_s)t_m$. Therefore we can write

$$t + (2q_{\mathcal{H}_0} + q_e + 4q_s)t_m \leq t^*$$

10.2 Performance Evaluation

We can estimate the computational cost and memory requirements *i.e* the bit size required in this scheme. Consider the following notation.

- T_M : The time for point scalar multiplication on EC.
- T_{PO} : Time for pairing execution.
- T_H : Time taken for execution of hash function.

Total time for Key extraction, signature generation and verification is given by

$$T = 4T_M + 3T_{PO} + 2T_H$$

11 Extension for Aggregation

D. Boneh et.al,[18] proposed the aggregate signature scheme.

Definition 8. *Let there are n distinct users $U = \{u_1, u_2 \ldots u_n\}$ having signing public key-private key $(pk_i, sk_i)_{1\leq i\leq n}$ pair. To aggregate signatures on subsets of users U, each user generates a signature σ_i, for all $i = 1, 2 \ldots n$ on any message m_i. These signatures are aggregated by an aggregating party in to a single signature σ_{agg}, which is the same length of the signature $(\sigma_i)_{1\leq i\leq n}$.*

The main goal in the design of such protocols is that the length of σ_{agg} be constant, independent of the number of messages and signers. To check correctness of an aggregate signature, the verifier will also need the messages m_i and the public keys pk_i, but this is not taken into account when considering the length of σ_{agg}. In the identity-based framework, the only proposal which achieves constant length aggregation is that of [22] however, this scheme only works in a more restrictive scenario where some interaction or sequentiality is needed among the signers of the messages, which later will be aggregated (in the same direction as[19] for the PKI-based scenario). With respect to non-interactive aggregate signatures in the identity-based setting, the most efficient proposal is from [20]that does not achieve constant-length aggregation: the length of the aggregate signature does not depend on the number of signed messages, but on the number of different signers. Using the approach of this work, we can achieve exactly the same level of partial aggregation for identity-based signatures. In effect,let us consider our generic construction, and let us assume that the employed PKI-based signature scheme S allows constant-length aggregation. The the input of the aggregation algorithm would be $\{(id_i, sig_{msk}(id_i \| pk_i), pk_i, m_i, sig_i\}_{1\le i\le n}$, where sig_i and $sig_{sk_i}(m_i)$ are signatures resulting from scheme S, and can therefore be aggregated into a PKI-based aggregate signature σ_{agg}, of constant length. Then the final identity-based aggregate signature would be $\sigma^{IB}_{agg} = (\sigma_{agg}, pk_i)_{1\le i\le n}$. This aggregate signature, along with the n messages and the n identities, is sufficient to verify the correctness of the n signatures. Therefore the length of the identity-based aggregate signature σ^{IB}_{agg} is linear with respect to the number of different signers.

It would be useful if a (single) sensor node can sign multiple messages, say n messages, but the size of resulting signature is significantly smaller than n times the size of a single signature. Such an aggregated (shortened) signature is of great importance in WSNs as reducing communication overheads in WSNs is crucial for resource-constrained sensor nodes. As an extension to our online/off-line IBS scheme, we propose the following aggregation technique when a single user (node) wants to sign multiple messages.

11.1 Framework of Aggregate Signatures

An ID-based online/off-line signature(IBS) scheme consists of the following five probabilistic polynomial time (PPT) algorithms:

- **Setup**: $(param, msk) \leftarrow$ Set (1^k) . The private key generator $\mathcal{PKG}$ provides the security parameter as the input to this algorithm, generates the system parameters $params$ and the master private key msk. $\mathcal{PKG}$ publishes $params$ and keeps msk secret.
- **Extract**: $D_{ID_i} \leftarrow Ext(1^k, param, msk, ID_i)$. The user U_i provides his identity ID_i to $\mathcal{PKG}$. The $\mathcal{PKG}$ runs this algorithm with identity ID_i, $params$ and msk as the input and obtains the private key D_{ID_i}. The private key D_{ID_i} is sent to user U_i through a secure channel.
- **Off-lineSign**: $\sigma_{off} \leftarrow Sgn_{off}(1^k, param)$ takes a security parameter k and the global parameters $param$ to generate an off-line signature σ_{off}.
- **OnlineSign**: $(\sigma_{on})_{i=\ 1 to n} \leftarrow Sgn_{on}(1^k, param, m, \sigma_{off}, ID_i)$. The algorithm takes a security parameter k, the global parameters $param$, a message m, an off-line signature σ_{off}, an identity ID_i to generate a online signature σ_{on_i}, for all $i = 1, 2 \ldots n$. so the Signature generated by all users U_i individually is the pair $\sigma_i = (\sigma_{off}, \sigma_{on_i})$, for all $i = 1, 2 \ldots n$.
- **Verify**: ("$accept$", "$Reject$") $\leftarrow Ver(1^k, param, \sigma, D_{ID})$. This algorithm takes a security parameter k, the global parameters $param$, a signature σ, a secret key D_{ID} to generate the outputs "$accept$" if σ is valid and outputs "$reject$" otherwise.
- **Aggregate** : $\sigma_{agg} \leftarrow Agg(\sigma_i)$. For aggregation, the algorithm receive the various signatures $(\sigma_i)_{1\le i\le n}$ from different users $(U_i)_{1\le i\le n}$, any third party or one of the signers can run this algorithm and generate the aggregate signature σ_{agg} for the pairs $(m_i, ID_i)_{1\le i\le n}$.
- **Aggregate Verify**: ("$Valid$", "$Invalid$") $\leftarrow AggVer(\sigma_i, m_i, ID_i, param)$ This algorithm takes on input of an aggregate signature σ_{agg} for pair $(m_i, ID_i)_{1\le i\le n}$ and the $param$ checks whether σ_{agg} is a valid aggregate signature on m_i by ID_i for all $i = 1, 2 \ldots n$. If true, it outputs "$Valid$", else outputs "$Invalid$".

11.2 Security Model

Unforgeability Gentry et al. in [22] proposed a formal model for aggregate signature scheme. Their scheme used a common randomness. We follow the security model proposed by Gentry et al. with slight variations since we do not have a common random value.

Definition 9. *An $\mathcal{IBS}$ scheme is secure against existential forgery under adaptive-chosen-identity and adaptive-chosen-message attack if no probabilistic polynomial time algorithm A has non-negligible advantage in the following game.*

- **Setup phase** : The challenger C runs the setup algorithm and generates the $params$ and msk. Challenger C gives params to adversary $\mathcal{A}$.
- **Training phase** : After the setup, $\mathcal{A}$ starts interacting with C by querying the various oracles provided by C in the following way:
- **KeyGen oracle** : When $\mathcal{A}$ makes a query with ID_i, C outputs D_i, the private key of ID_i to $\mathcal{A}$, provided C knows the private key for the queried identity. Else it aborts.
- **Signing oracle** : When $\mathcal{A}$ makes a signing query with ID_i, message m_i, C outputs a valid signature σi on m_i by ID_i.
- **Forgery phase** : The adversary $\mathcal{A}$ generates output an aggregate signature σ_{agg} for signatures $i = 1$ to n from the users $(ID_i)_{1\le i\le n}$ on messages $(m_i)_{1\le i\le n}$ where there exists at least one target identity $ID_{\mathcal{T}} \in \{ID_i\}_{1\le i\le n}$, for which private key has not been queried for. The adversary $\mathcal{A}$ wins the game if σ_{agg} is a valid aggregate signature and $\mathcal{A}$ has not queried for the signature from the signing oracle for $(ID_{\mathcal{T}}, m_{\mathcal{T}})$ pair on which it has generated the forgery.

$$Adv_{\mathcal{A}}^{UF-\mathcal{IBS}} = \{Pr[\mathcal{A}(Verify(\sigma_{agg}))] = valid\}$$

11.3 Aggregate Signature Scheme

- **Setup** Given security parameters k, the $\mathcal{PKG}$ chooses groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order q. A generator P of $\mathbb{G}_1$, a bilinear map $\hat{e} : \mathbb{G}_1 X \mathbb{G}_1 \to \mathbb{G}_2$ and collision resistant hash function $\mathcal{H}_0 : \{0,1\}^* \to \mathbb{G}_1$, $\mathcal{H}_1 : \{0,1\}^* \to \mathbb{Z}_q^*$. It chooses a master-key $s \in \mathbb{F}_q^*$ and computes $P_{pub} = sP$. the system public parameters are given by

$$\mathcal{P} = (\mathbb{G}_1, \mathbb{G}_2, q, \hat{e}, P, P_{pub}, \mathcal{H}_0, \mathcal{H}_1)$$

- **Extract** This algorithms follows of the following steps
  - Given an identity $ID \in \{0,1\}^*$ the secret key is $d_{ID} = s \cdot Q_{ID}$, where $Q_{ID} = \mathcal{H}_0(ID)$.
- **Off-line Sign**: During this phase, the signer computes the followings
  - $\hat{\alpha} i = \hat{e}(P, P_{pub})^{2^i}, \forall i = 0, 1 \ldots |q| - 1$.
- **Online Sign**: During this phase the signer follows the following steps
  - Select randomly $\beta_l \in \mathbb{Z}_q^*$. Let $\beta_l[i]$ be the i^{th} bit of β_l.
  - Define $\mathcal{D}_l \subset \{1, 2 \ldots |q|\}$ be the set of indices such that $\beta_l[i] = 1$ and $C \subset \{1, 2 \ldots |q|\}$ be the set of indices such that $\beta_l[i] = 0$.
  - Computes $\psi_1 = \prod_{i\in\mathcal{D}} \hat{\alpha}_{i-1}$ and $\psi_2 = \prod_{i\in C} \hat{\alpha}_{i-1}$.
  - Computes $\alpha = \psi_1\psi_2$.
  - Selects $\gamma_l \in \mathbb{Z}_q^*$, computes $U_l = \gamma_l \cdot P$, $r_l = \mathcal{H}_1(ID_l, U_l \| m_l)$, $\forall l = 1, 2 \ldots n$
  - Computes
  $$V = \textstyle\sum_{l=1}^{n} V_l,\ V_l = (\gamma_l + \beta_l) \cdot P_{pub} + r_l d_{ID}.\ \forall l = 1, 2 \ldots n$$

  The aggregate signature $\sigma = (\alpha_l, U_l, V)$ for $l = 1, 2 \ldots n$.
- **Verify** : To verify the signature (α_l, U_l, V) for the message m_l and the identity ID_l, $l = 1, 2 \ldots n$, the verifier follows the following steps.
  - Computes $r_l = \mathcal{H}_1(ID_l, U \| m_l)$, $\forall l = 1, 2 \ldots n$
  - Checks whether the following equation holds

$$\hat{e}(V, P) \stackrel{?}{=} \prod_{l=1}^{n} \alpha_l\, \hat{e}(U_l, P_{pub}) \prod_{l=1}^{n} \alpha_l \prod_{l=1}^{n} \hat{e}(Q_{ID}, P_{pub})^{r_l} \qquad (4)$$

11.4 Proof of Correctness

First computes $r_l = \mathcal{H}_1(ID_l, U_l||m_l)\forall l = 1, 2 \ldots n$ and verify the following equation

$$\hat{e}(V, P) \stackrel{?}{=} \prod_{l=1}^{n} \hat{e}(U_l, P_{pub}) \prod_{l=1}^{n} \alpha_l \prod_{l=1}^{n} \hat{e}(Q_{ID}, P_{pub})^{r_l} \tag{5}$$

Also check $\alpha = \hat{e}(P, P_{pub})^{\beta}$

$$\begin{aligned} \alpha &= \hat{\alpha_0}\hat{\alpha_1} \ldots \\ &= \hat{e}(P, P_{pub})^{2^0}\hat{e}(P, P_{pub})^{2^1} \ldots \\ &= \hat{e}(P, (2^0 + 2^1 + \ldots)P_{pub}) \\ &= \hat{e}(P, \beta P_{pub}) = \hat{e}(P, P_{pub})^{\beta} \end{aligned}$$

$V = \sum_{l=1}^{n} V_l, V_l = (\gamma_l + \beta_l) \cdot P_{pub} + r_l d_{ID}. \forall l = 1, 2 \ldots n$ $\hat{e}(V, P) = \hat{e}(\sum_{l=1}^{n} V_l, P)$

$= \hat{e}(\sum_{l=n}^{n}(\gamma_l + \beta_l) \cdot P_{pub} + r_l d_{ID}, P)$

$= \hat{e}(\sum_{l=n}^{n}(\gamma_l + \beta_l) \cdot P_{pub}, P) \prod_{l=1}^{n} \hat{e}(r_l d_{ID}, P)$

$= \hat{e}(P_{pub}, \sum_{l=n}^{n}(\gamma_l + \beta_l) \cdot P) \prod_{l=1}^{n} \hat{e}(r_l d_{ID}, P)$

$= \hat{e}(P_{pub}, \gamma_l P)\hat{e}(P_{pub}, \sum_{l=n}^{n}(\beta_l) \cdot P \prod_{l=1}^{n} \hat{e}(r_l d_{ID}, P)$

$= \hat{e}(P_{pub}, \gamma P)\hat{e}(P_{pub}, P)^{\sum_{l=n}^{n}(\beta_l)} \prod_{l=1}^{n} \hat{e}(r_l d_{ID}, P)$

$= \prod_{l=1}^{n} \hat{e}(P_{pub}, U_l) \prod_{l=1}^{n} \alpha_l \prod_{l=1}^{n} \hat{e}(r_l s \cdot Q_{ID}, P)$

$= \prod_{l=1}^{n} \hat{e}(P_{pub}, U_l) \prod_{l=1}^{n} \alpha_l \prod_{l=1}^{n} \hat{e}(r_l Q_{ID}, s \cdot P)$

$= \prod_{l=1}^{n} \hat{e}(P_{pub}, U_l) \prod_{l=1}^{n} \alpha_l \prod_{l=1}^{n} \hat{e}(r_l Q_{ID}, P_{pub})$

$= \prod_{l=1}^{n} \hat{e}(P_{pub}, U_l) \prod_{l=1}^{n} \alpha_l \prod_{l=1}^{n} \hat{e}(Q_{ID}, P_{pub})^{r_l}$

12 Security Analysis

Theorem 2 *In random oracle model, Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be (ϵ^*, t^*)-BDH two cyclic group of same prime order q. P be a generator of $\mathbb{G}_1$. $\mathbb{G}_1$ is an additive group and $\mathbb{G}_2$ is a multiplicative group. Let e be a computable bilinear map $e : \mathbb{G}_1 X \mathbb{G}_1 \to \mathbb{G}_2$. The proposed scheme is $(\epsilon, t, q_e, q_s, q_h, \xi)$-secure with respect to EF-$\mathcal{IBS}$-CMA, where for any t and ϵ satisfying*

$$\epsilon \geq e(q_e + \xi)\epsilon^*, t \leq t^* - t_m(2q_{\mathcal{H}_0} + q_e + 4q_s + 2\xi + 2)$$

where e is the base of the natural logarithm, and T is the time of computing a scalar multiplication in $\mathbb{G}_1$ and an inversion in $\mathbb{Z}_q^*$. and $q_e, q_s, q_{\mathcal{H}_0}$ are the number of extraction, signing and hashing queries respectively the adversary is allowed to make and t_m is the time for computing scalar multiplication.

Proof: Suppose that $\mathcal{A}$ is a forger who breaks the scheme $\mathcal{IBS}$. A CDH instance (P, xP, yP) is given for $x, y \in \mathbb{Z}_q^*$. By using the forgery algorithm $\mathcal{A}$, we will construct an algorithm $\mathcal{B}$ which outputs the CDH solution xyP in $\mathbb{G}_1$. Algorithm $\mathcal{B}$ performs the following simulation by interacting with the forger $\mathcal{A}$.

- **Setup**: Algorithm $\mathcal{B}$ sets $P_{pub} = xP$ and starts by giving $\mathcal{A}$ the system parameters *param* including (P, P_{pub}). At any time, $\mathcal{A}$ can query the random oracles $\mathcal{H}_0$ and $\mathcal{H}_1$ and **Extract** and **Sign** queries. To answer these queries, $\mathcal{B}$ does the following:
- **Queries on Oracle** $\mathcal{H}_0$: To respond $\mathcal{H}_0$-queries, $\mathcal{B}$ maintains a list of tuples (ID, ω, b, c) as explained below. We refer to this list as the L_0-list. When $\mathcal{A}$ queries the oracle $\mathcal{H}_0$ at a point $ID \in \{0,1\}^*$, $\mathcal{B}$ responds as follows:
  1. If the query ID already appears on the L_0-list in a tuple (ID, ω, b, c) then $\mathcal{B}$ responds with $\mathcal{H}_0(ID) = \omega \in \mathbb{G}_1$.
  2. Otherwise, $\mathcal{B}$ picks a random coin $c \in \{0,1\}$ with $Pr[c = 0] = \frac{1}{(q_e+1)}$.
     - If $c = 0$ then $\mathcal{B}$ computes $\omega = b(yP)$ for a random $b \in \mathbb{Z}_q^*$.
     - If $c = 1$ then $\mathcal{B}$ computes $\omega = bP$ for a random $b \in \mathbb{Z}_q$.

     $\mathcal{B}$ adds the tuple (ID, ω, b, c) to the L_0-list and responds to $\mathcal{A}$ with $\mathcal{H}_0(ID) = \omega$.
- **Queries on Oracle** $\mathcal{H}_1$ **and Sign Queries** When $\mathcal{A}$ makes $\mathcal{H}_1$-queries and sign queries. To respond to $\mathcal{H}_1$-queries, $\mathcal{B}$ maintains a list of tuples (ID_i, U_i, m_i, ri) for $i = 1, 2 \ldots n$ as explained below. We refer to this list as the L_1-list. When $\mathcal{A}$ queries the oracle $\mathcal{H}_1$ at (ID_i, U_i, m_i), $\mathcal{B}$ responds as follows:
  1. If the query (U_i, m_i) already appears on the L_1-list in a tuple (ID_i, m_i, U_i, r_i) then $\mathcal{B}$ responds with $\mathcal{H}_1(ID_i, U_i \| m_i) = r_i \in \mathbb{Z}_q^*$, for $i = 1, 2 \ldots n$.
  2. Otherwise, $\mathcal{B}$ picks a random $r_i \in \mathbb{Z}_q^*$ and adds the tuple (ID_i, m_i, U_i, r_i) to the L_1-list and responds to $\mathcal{A}$ with $\mathcal{H}_1(ID_i, U_i \| m_i) = r_i$.
- **Output**$\mathcal{A}$ returns an aggregate signature $\sigma = (\alpha, U_i, V)$ for $ID_1, ID_2 \ldots ID_n$, where $\gamma \in \mathbb{Z}_q^*$, $U_i = \gamma_i \cdot P$, $r_i = \mathcal{H}_1(ID_i, U_i \| m_i)$, $\forall i = 1, 2 \ldots n$

$$V = \textstyle\sum_{i=1}^n V_i,\ V_i = (\gamma_i + \beta_i) \cdot P_{pub} + r_i d_{ID}.\ \forall i = 1, 2 \ldots n$$

. $\mathcal{B}$ finds the n-tuples $(ID_i, \omega_i, a_i, b_i, c_i)$ for $= 1, 2 \ldots n$ from $\mathcal{H}_0$ list and proceeds only $c_k = 0$ and $c_j = 1, 2, \ldots n, j \neq k$. Here (ID_k, m_k) has never requested to the sign oracle. Otherwise $\mathcal{B}$ fails and halts. when $\mathcal{H}_k = \omega_k = b_k(y \cdot P)$ and $\mathcal{H}_0(ID_j) = \omega_j = b_j \cdot P$ for $j = 1, 2 \ldots n, j \neq k$. The aggregate signature σ satisfies the following aggregate verification.

$$\hat{e}(V, P) = \textstyle\prod_{i=1}^n \hat{e}(U_i, P_{pub}) \prod_{i=1}^n \alpha_i \prod_{i=1}^n \hat{e}(\omega_i, P_{pub})^{r_i}$$

$\mathcal{B}$ finds the corresponding tuples (ID_i, m_i, U, r_i) from L_1-list. Let $V_i = b_i \cdot P_{pub}$. Computes $\hat{e}(V_i, P) = \hat{e}(Q_{ID_i}, P_{pub})$ for $1 \leq i \leq n, i \neq k$. Finally $\mathcal{B}$ constructs V' as $V - \sum_{i=1, i\neq k}^n V_i$ and $V' = d_k + \sum_{i=1}^n r_i \cdot \gamma_i \cdot P_{pub}$, for $U' = (r_k^*)^{-1} \sum_{i=1}^n r_i \cdot U_i$. Then $\mathcal{B}$ execute the hash value $\mathcal{H}_1(ID_k, m_k \| U')$ and return r_k^*, *i.e* $\mathcal{H}_1(ID_k, m_k \| U') = r_k^*$. If the tuples exist in L_1-list then tries another r_k^*. Continue until such collision does not occur. Therefore $\sigma' = (U', V', \alpha)$ is a valid signature on m_k for ID_k. Its verification equation $\hat{e}(U', P_{pub})\alpha\hat{e}(\omega_i, P_{pub})^{r^*} = \hat{e}((r_k^*)^{-1} \sum_{i=1}^n r_i \cdot U_i, P_{pub})$
$= \hat{e}(d_k + \sum_{i=1}^n r_i \cdot \gamma_i \cdot P_{pub}, P) = \hat{e}(V', P)$ .Finally, $\mathcal{B}$ returns σ' as a forgery of the scheme.

12.1 Probability Analysis

$\mathcal{B}$ solves the given instance of the CDH problem with probability at least ϵ^*. We analyze three independent events needed for $\mathcal{B}$ to succeed:

- E_1 : $\mathcal{B}$ does not abort as a result of any $\mathcal{A}$'s Extract query.
- E_2 : $\mathcal{A}$ generates a valid and non-trivial signature forgery $\sigma = (\alpha, U_i, V)$ on m_i for ID_i, $i = 1, 2 \ldots n$.
- E_3 : Event E_2 occurs and $c_k = 0, c_i = 1$ for $1 \leq i \leq n, i \neq k$, where for each i, c_i is the c-component of the tuples containing ID_i on the L_0-list.

$\mathcal{B}$ wins if all these events happen. The probability is $Pr[E_1 \wedge E_2 \wedge E_3]$

$$Pr[E_1 \wedge E_2 \wedge E_3] = Pr[E_1] \cdot Pr[E_2 \mid E_1] \cdot Pr[E_3 \mid [E_1 \wedge E_2]$$

- **Claim 1** The probability that $\mathcal{A}$ does not abort as a result of $\mathcal{A}$'s Extract query is at least $(1 - \frac{1}{q_e+\xi})^{q_e}$.
- **Claim 2** If $\mathcal{A}$ does not abort as a result of $\mathcal{A}$'s Extract query than $\mathcal{A}$'s view is identical to its view in the real attack. Hence, $Pr[E_2 \mid E_1] \geq \epsilon$.
- **Claim 3** The probability that $\mathcal{B}$ does not abort after $\mathcal{A}$ outputs a valid and nontrivial forgery is at least $(\frac{1}{(q_e+\xi)^{\xi-1}}$. Algorithm $\mathcal{B}$ will abort only if $\mathcal{A}$ generates a forgery such that $c = 1$. Hence $Pr[E_3 \mid E_1 \wedge E_2] \geq \frac{1}{(q_e+1)}$

So $Pr[E_1] \cdot Pr[E_2 \mid E_1] \cdot Pr[E_3 \mid [E_1 \wedge E_2] \geq (1 - \frac{1}{q_e+1})^{q_e} \cdot \epsilon \cdot \frac{1}{(q_e+1)} \geq \frac{1}{e} \cdot \frac{\epsilon}{(q_e+1)} \geq \epsilon^*$.

Algorithm $\mathcal{B}$ will abort unless $\mathcal{A}$ generates a forgery such that $c_k = 0$ and $c_i = 1$ for $1 \leq i \leq n, i \neq k$. Therefore $Pr[c_k = 0] = \frac{1}{(q_e+\xi)}$ and the probability that $c_i = 1$, for $1 \leq i \leq n, i \neq k$, is given by

$$Pr[c_i = 1, \forall 1 \leq i \leq n, i \neq k] \geq (1 - \frac{1}{q_e+\xi})^{\xi-1}$$
$$\Rightarrow Pr[E_3 \mid [E_1 \wedge E_2] \geq (1 - \frac{1}{q_e+\xi})^{\xi-1} \cdot (\frac{1}{q_e+\xi}).$$

Thus $Pr[E_1] \cdot Pr[E_2 \mid E_1] \cdot Pr[E_3 \mid [E_1 \wedge E_2] \geq (1 - \frac{1}{q_e+\xi})^{q_{e+\xi-1}} \cdot \epsilon \cdot \frac{1}{(q_e+\xi)} \geq \frac{1}{e} \cdot \frac{\epsilon}{(q_e+\xi)} \geq \epsilon^*$.

Algorithm $\mathcal{B}$'s running time is equal to the sum of the running time of $\mathcal{A}$'s and the time it takes to respond to $q_{\mathcal{H}_0}$ hash oracle $\mathcal{H}_0$ queries, q_e key extract queries, and q_s online/off-line signature queries. Each $\mathcal{H}_0$ query requires one scalar multiplications in $\mathbb{G}_1$. Each key extract query needs one scalar multiplication in $\mathbb{G}_1$. Each online/off-line signature requires 4 scalar multiplications in $\mathbb{G}_1$. The outpu phases requires 2ξ and one inversion operation. If we assume one scalar multiplications in $\mathbb{G}_1$ takes time t_m, the total running time is at most $t + (2q_{\mathcal{H}_0} + q_e + 4q_s + 2\xi + 2)t_m$. Therefore we can write

$$t + (2q_{\mathcal{H}_0} + q_e + 4q_s + 2\xi + 2)t_m \leq t^*$$

13 Implementation on WSN

The signatures generated by the sensor nodes can be verified mutually by sensor nodes and by the base station. In WSN application off-line phase can be executed at the base station, while the online phase is to be executed in the WSN node. Like the case for general WSNs, we assume that the base station is powerful a sufficient amount to perform computationally intensive cryptographic operations, and the sensor nodes, on the other hand, have limited resources in terms of computation, memory and battery power. The sensor nodes may be one of the above described. To implement the proposed signature scheme on WSN, we can follow the similar method [2]. Let us consider the system parameters *param* is generated by the base station and is embedded in each sensor node when they are deployed. The Signatures generated by the sensor nodes can be verified either by the sensor nodes or by the base station.

Let us consider n no of sensor nodes as $SN_1, SN_2 \dots SN_n$ with identity $ID_1, ID_2 \dots ID_n$. The system parameters $(\mathbb{G}_1, \mathbb{G}_2, q, \hat{e}, P, P_{pub}, \mathcal{H}_0, \mathcal{H}_1)$ is generated by the base station and all parameters will be embedded on each sensor node. Then signature (α, U, V) will be generated by the nodes .

14 Conclusion

This paper proposes a secure and efficient online/off-line signature scheme for WSN. The scheme is secure against existential forgery on chosen message attack in random oracle model under the assumption of Computational Diffie-Hellman Problem (CDH) is hard. Here we have shown the vulnerability of Liu et al's scheme and proposed a provably secure scheme.

References

1. S. Even, O. Goldreich, and S. Micali *On-Line/Off-Line digital signatures*, in Proc. Advances in Cryptology CRYPTO 89, ser. LNCS, vol. 435. Springer Berlin, 1990, pp. 263275.
2. Joseph K. Liu, Joonsang Baek, Jianying Zhou, Yanjiang Yang and Jun Wen Wong *Efficient Online/Offline Identity-Based Signature for Wireless Sensor Network*, in IACR Arcieve ePrint-2010/03.
3. S.Sharmila Deva Selvi, S.Sree Vivek, J.Shriram, S.Kalaivani, and C.Pandu Rangan. Security analysis of aggregate signature and batch verification signature schemes. Cryptology ePrint Archive, Report 2009/290, 2009. http://eprint.iacr.org.
4. F. Amin, A.H Jahangir, and H. Rasi fard *Analysis of Public-Key Cryptography for Wireless Sensor Networks Security*. World Academy of Science, Engineering and Technology, 2008.
5. Chris Townsend, Steven Arms *Wireless Sensor Networks: Principles and Applications*: microstrain.com.
6. Jing Deng, Richard Han, Shivakant Mishra Enhancing Base Station Security in Wireless Sensor Networks: University of Colorado, Department of Computer Science. Technical Report CU-US-951-03.
7. Dan Boneh Bls short digital signatures. In Henk C. A. van Tilborg, editor, Encyclopedia of Cryptography and Security. Springer, 2005.
8. Yiling Wen and Jianfeng Ma An aggregate signature scheme with constant pairing operations. In CSSE (3), pages 830833. IEEE Computer Society, 2008.
9. Jing Xu, Zhenfeng Zhang, and Dengguo Feng Id-based aggregate signatures from bilinear pairings. In Yvo Desmedt, Huaxiong Wang, Yi Mu, and Yongqing Li, editors, CANS, volume 3810 of Lecture Notes in Computer Science, pages 110119. Springer, 2005.
10. HyoJin Yoon, Jung Hee Cheon, and Yongdae Kim Batch verifications with id-based signatures. In Choonsik Park and Seongtaek Chee, editors, ICISC, volume 3506 of Lecture Notes in Computer Science, pages 233248. Springer, 2004.
11. Shi Cui, Pu Duan, and Choong Wah Chan An efficient identity-based signature scheme with batch verifications. In Xiaohua Jia, editor, Infoscale, volume 152 of ACM International Conference Proceeding Series, page 22. ACM, 2006
12. Xiangguo Cheng, Jingmei Liu, and Xinmei Wang Identity-based aggregate and verifiable encrypted signatures from bilinear pairing. In Osvaldo Gervasi, Marina L. Gavrilova, Vipin Kumar, Antonio Lagan‘a, Heow Pueh Lee, Youngsong Mun, David Taniar, and Chih Jeng Kenneth Tan, editors, ICCSA (4), volume 3483 of Lecture Notes in Computer Science, pages 10461054. Springer, 2005.
13. Alexandra Boldyreva, Craig Gentry, Adam ONeill, and Dae Hyun Yum Ordered multisignatures and identity-based sequential aggregate signatures, with applications to secure routing. In Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson, editors, ACM Conference on Computer and Communications Security, pages 276285. ACM, 2007.
14. Zhu Wang, Huiyan Chen, Ding feng Ye, and Qian Wu. Practical identity-based aggregate signature scheme from bilinear maps. volume 13(6), pages 684687. Shangai Jiao Tong University Press, 2008.
15. Jung Yeon Hwang, Dong Hoon Lee, and Moti Yung Universal forgery of the identity-based sequential aggregate signature scheme. In Wanqing Li, Willy Susilo, Udaya Kiran Tupakula, Reihaneh Safavi-Naini, and Vijay Varadharajan, editors, ASIACCS, pages 157160. ACM, 2009.
16. Craig Gentry and Zulfikar Ramzan Identity-based aggregate signatures. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, Public Key Cryptography, volume 3958 of Lecture Notes in Computer Science, pages 257273. Springer, 2006.
17. Javier Herranz Deterministic identity-based signatures for partial aggregation. Comput. J., 49(3):322330, 2006
18. D. Boneh, C. Gentry, B. Lynn and H. Shacham Aggregate and verifiably encrypted signatures from bilinear maps. EUROCRYPT 2003, volume 2656 of LNCS, pages 416432, 2003.
19. S. Lu, R. Ostrovsky, A. Sahai, H. Shacham and B. Waters Sequential aggregate signatures and multi-signatures without random oracles. EUROCRYPT06, 2006.
20. J. Herranz Deterministic identity-based signatures for partial aggregation. The Computer Journal, 49 (3):322330, 2006.
21. M.Bellare and P.Rogaway "The exact security of digital signatures-How to sign with RSA and Rabin" Proceedings of Eurocrypt 96, LNCS Vol 1070, pp-399-416, Springer-Verlag, 1996
22. Craig Gentry and Zulfikar Ramzan Identity-based aggregate signatures. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, Public Key Cryptography, volume 3958 of Lecture Notes in Computer Science, pages 257273. Springer, 2006.