Securing Data Transfer in Cluster-based Wireless Sensor Networks with Less Overhead

Sithara Chandran
Student, Department of IT
Nehru College of Engineering and Research Centre
Pampady, Thrissur, India

Meji Jose
Assistant Professor, Department of IT
Nehru College of Engineering and Research Centre
Pampady, Thrissur, India

Abstract **— A wireless sensor network consists of a large number of battery powered sensor nodes that are capable of sensing and processing the data, and can transmit the data wirelessly to the base station. The effective and practical way for enhancing the performance of WSNs is clustering. As the clusters are formed dynamically and periodically in a cluster based WSN, securing the data transmission in these networks is a critical issue. A secure and efficient data transmission (SET) protocol, called SET-IBS is been recently proposed whose security relies on the hardness of the Diffie-Hellman problem in the pairing domain. Though results show that this protocol has better performance than the existing secure protocols for CWSNs in terms of security overhead and energy consumption, the computational overhead of this protocol is comparatively high. The existing SET-IBS protocol is based on bilinear pairings, where the computation of pairing is time-consuming. This paper aims to provide a reliable and efficient data transmission for CWSNs with low computational overhead using identity based digital signature from elliptic curve digital signature algorithm (IB-ECDSA), which is an identity based signature protocol without pairings, to enhance the existing SET-IBS protocol. As a result, the computation cost can be saved when compared to the existing SET-IBS scheme that uses pairing.**

Keywords— SET-IBS; IB-ECDSA; Bilinear Pairing

I. INTRODUCTION

A wireless sensor network consists of several number of battery powered sensor nodes that are able to sense and process the data, and can transmit it wirelessly to the base station. The nodes are capable of sensing the environment conditions such as temperature, pressure, sound, motion, etc. Hence, a sensor network interacts with the environment rather than interacting with humans. Clustering is an effective way to enhance the performance of WSNs [5]. As the cost of data transmission is much more expensive than the cost of data processing, it is better that an intermediate node (CH) aggregates data from all other nodes and sends it to the base station than each sensor node sending data directly to the base station. Since the wireless sensor networks are deployed in harsh, neglected and adversarial physical environments in applications such as that in military domains, a secure and efficient method for data transmission is very necessary in such practical applications.

The low- energy adaptive clustering hierarchy (LEACH) protocol presented by Heinzelman et al. [6] is a widely known protocol for CWSNs. This protocol randomly rotates the CHs among all the sensor nodes in the network in every round, in order to prevent the quick energy consumption in the CHs. Several other protocols have been presented using the similar concepts of LEACH afterwards. But adding security to LEACH-like protocols is very challenging as the CHs are changing randomly and dynamically. The problem with such LEACH-like protocols is that they all uses symmetric key management for security and hence, they suffer from orphan node problem, which occurs when a node does not share a pair wise symmetric key with the other nodes in the network, its preloaded key ring. Thus asymmetric key management can be used instead of the symmetric key management. Digital signature can be used for asymmetric key management, where the binding between the public key and the signer's identification is obtained via a digital certificate.

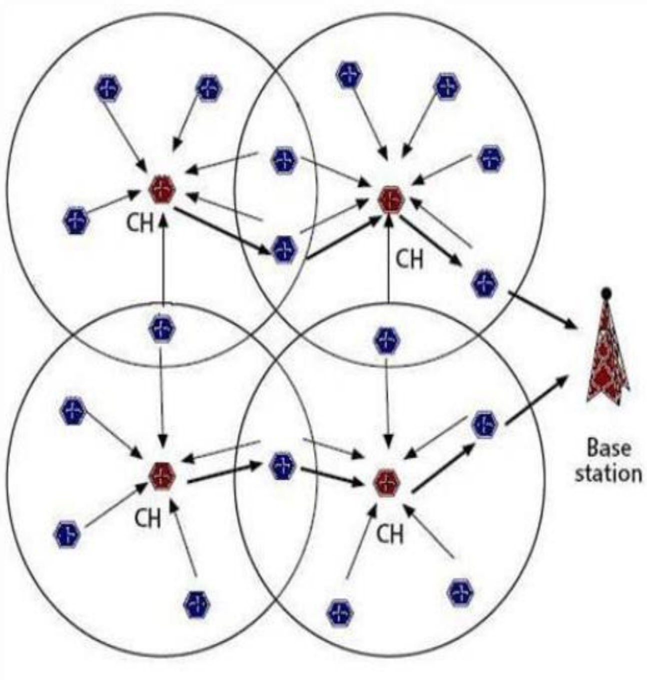


Fig 1.1: Data transfer in Cluster-based WSNs

SET-IBS protocol [1] has been proposed recently from the already existing IBS scheme, for securing the data transmission in cluster based WSNs. Though it has better performance than the existing secure protocols for CWSNs, the computational overhead of this protocol is comparatively high. The key management for this protocol uses bilinear pairings on elliptic curves. Therefore, in this paper we are using an identity based digital signature scheme based on elliptic curve digital signature algorithm (IB-ECDSA) [2] without pairings with the existing SET-IBS [1] in order to reduce the computational overhead further. The network architecture in the existing protocol consists of a fixed base station (BS) and a large number of homogeneous sensor nodes with similar functionalities. The BS is assumed to be a trusted authority. Here, the sensor nodes are been grouped into clusters and each cluster will have a cluster head (CH) sensor node that is elected autonomously based on the LEACH protocol [6]. There is a TDMA control used for data transmission and for each round, a sensor node decides whether to become a cluster head or not based on a threshold value *T(n)* which is computed in node *n* based on the equation in LEACH protocol[6].

$$T(n) = \frac{\rho}{1-\rho \times \left(r \bmod \left\lfloor \frac{1}{\rho} \right\rfloor\right)} \cdot \frac{E_{cur}(n)}{E_{init}(n)} \qquad \forall n \in G_n,$$

$$T(n) = 0 \qquad \forall n \notin G_n.$$

In the above equation, $E_{cur}(n)$ is the current energy of the node and $E_{init}(n)$ is the initial energy of the node. The ratio of the residual energy is multiplied to increase the energy efficiency in dynamic clustering. Here, ρ is a priori determined value that stands for the desired percentage of cluster heads during a round, r is the current round number and G_n is the set of sensor nodes that had not been a CH in the last ⌊1/ ρ⌋ rounds.

The remaining of this paper is organized as follows: Section II is about bilinear pairings, bilinear map and bilinear pairing in SET-IBS protocol. Section III describes the existing SET-IBS protocol. Section IV is the problem statement that describes about the problem with the existing SET-IBS protocol and the solution for the problem. Section V is about elliptic curve group, ECDSA and IB-ECDSA. Section VI describes about the proposed SET-IBS using IB-ECDSA without bilinear pairings. Section VII which is the last section concludes this paper.

II. Bilinear Pairing

Bilinear maps [3] are the tool of pairing-based cryptography which has become a hot topic started with an identity based encryption scheme by Boneh and Franklin in 2001.They establishes relationship between cryptographic groups.

A. *Definition of A Bilinear Map*

Let G_1, G_2, and G_t be cyclic groups of the same order. A bilinear map from $G_1 \times G_2$ to G_t is a function $e : G_1 \times G_2 \rightarrow G_t$ such that: $\forall$ u $\in G_1$, v $\in G_2$, a, b $\in$ Z, e(ua, vb) = e(u, v)ab .

Bilinear maps are called pairings because they associate pairs of elements from G_1 and G_2 with elements in G_t. Let e : $G_1 \times G_2 \rightarrow G_t$ be a bilinear map. Let g_1 and g_2 be generators of G_1 and G_2, respectively. The map e is an admissible bilinear map if e (g_1, g_2) generates G_t and e is efficiently computable. These are the only bilinear maps we care about. Sometimes such a map is denoted ˆe; we continue to use e. Also, from now on we implicitly mean admissible bilinear map when we say bilinear map.

B. *Relation between G_1 ,G_2 , and G_t*

As in [3], G_1, G_2, and G_t are all isomorphic to one another since they have the same order and are cyclic. They are different groups in the sense that we represent the elements and compute the operations differently. Normally, however, $G_1 = G_2$ (in addition to being isomorphic). We assume this unless otherwise noted. Denote both by G = G_1 = G_2. G and G_t may have either composite or prime order.

C. *Bilinear Pairing in SET-IBS Protocol*

The first functional and efficient ID-based encryption scheme based on bilinear pairings on elliptic curves has been introduced by Boneh and Franklin [4]. Here, two large primes p and q are selected randomly and let E/F_p be the elliptic curve over a finite field F_p. The q-order subgroup of the additive group of points in E/F_p is denoted as G_1 and a q-order subgroup of the multiplicative group in the finite field F_p^*.

From [1], the pairing is a mapping denoted by e such that e: $G_1 \times G_1 \rightarrow G_2$, is a bilinear map that satisfies the following properties:

1. *Bilinear*: : $\forall$ P,Q,R,S $\in G_1$, e(P+Q+R+S) = e(P,R) e(P,S) e(Q,R) e(Q,S). Similarly, $\forall$ c, d $\in Z_q^*$, e(cP, dQ) = e(P, d Q)^c = e(c P, Q)^d = e(P,Q)^{cd} , etc.
2. *Non-degeneracy*: If generator of G_1 is P, then e(P.P) will be the generator of G_2.
3. *Computability*: There exists an efficient algorithm to compute the mapping e (P, Q) in G_2, $\forall$ P, Q $\in G_1$.

Weil pairing and Tate pairing are known examples of bilinear mappings. They uses very complicated math and are non-trivial to compute. Weil and Tate pairings computed using Miller's algorithm and are computationally expensive. Tate pairing is normally somewhat faster than Weil pairing. Making these pairings faster still is current research.

III. Existing SET-IBS Protocol

A. *Protocol Initialization for SET-IBS protocol*

The SET-IBS protocol [1] executes in rounds, which consists of two phases – setup phase and steady state phase, in each round. The time is divided into successive time slots by the TDMA control as that in LEACH protocol .Mainly two timestamps are there for communication- T_s and t_j, T_s for BS to node communication and t_j for leaf node to cluster head communication. The user's public key is ID||t and the corresponding private pairing parameters are preloaded in the nodes during the protocol initialization. Additively homomorphic encryption scheme, in which an operation performed on the plain text is equivalent to that performed on the cipher text, is been adopted in SET-IBS protocol for encrypting the sensed data, in order to allow efficient aggregation of data both at the CHs and at the BS.

In the protocol initialization phase, the BS performs the key predistribution operations to all the sensor nodes. The operation is as follows:

- Generate k, which is the homomorphic encryption key to encrypt the sensed plaintext, where $k \in [m-1]$, m is a large integer.
- Generate the pairing parameters $(p, q, E/F_p, G_1, G_2, e)$. Select a generator P of G_1.
- Choose two cryptographic hash functions: H and h, for mapping strings to elements in G and for mapping arbitrary inputs to fixed length outputs, respectively.
- Pick a random integer $\tau \in Z_q^*$ as the master key msk and set $P_{pub} = \tau P$ as the master public key.
- Preload the parameters $(k, m, p, q, E/F_p, G_1, G_2, e, H, h, P, \tau)$ in each sensor node.

B. Key management in SET-IBS protocol

If a leaf node j wants to transmit a message M to the cluster head CH_i, then the node j will encrypt the message using the homomorphic encryption key k to get the ciphertext C. The SET-IBS scheme does three operations- extraction, signing and verification, after the protocol initialization step.

Extraction: The sensor node j will extract its private key sek_j from its ID, msk and timestamp t_j. t_j is node j's timestamp in the current round generated by its CH I from the TDMA control.

$$sek_j = \tau\, H(ID_j||t_j)$$

Signing: For signing the message bilinear mapping is been used. The node j picks a random number $\alpha_j \in Z_q^*$ and computes $\theta_j = e\,(P,\, P)^{\alpha j}$. Then the node computes c_j and σ_j as,

$c_j = h(C_j||t_j||\theta_j)$ and

$$\sigma_j = c_j\, sek_j + \alpha_j P$$

This $<\sigma_{j,} c_j>$ is the digital signature of node j for the ciphertext C_j.

Verification: On receiving the message, each node verifies the authenticity of the message by first checking the timestamp t_j for the current round in order to verify whether the message is fresh or not. If the node verifies that the message is fresh, then it computes θ'_j as,

$$\theta'_j = e(\sigma_j, P)e(H(ID_j||t_j), -P_{pub)})^{cj}$$

If the received message is authentic, then $\theta'_j = \theta_j$, using the formula,

$$\theta'_j = e(\sigma_j, P)e(H(ID_j||t_j), -P_{pub)})^{cj}$$

$$= e(\sigma_j, P)e(H(ID_j||t_j), -\tau P)^{cj}$$

$$= e(c_j\, sek_j + \alpha_j P,\, P)e(\tau H(ID_j||t_j),\, \tau P)^{-cj}$$

$$= e(c_j\, sek_j + \alpha_j P,\, P)e(\tau H(ID_j||t_j),\, P)^{-cj}$$

$$= (e(sek_j, P)^{cj}\, e(P,P)^{\alpha j})\, e(\tau H(ID_j||t_j),\, P)^{-cj}$$

$$= e(sek_j, P)^{cj}\, e(P,P)^{\alpha j}\, e(sek_j, P)^{-cj}$$

$$= e(P,P)^{\alpha j} = \theta_j\, .$$

If $h(C_j||t_j||\theta'_j) = h(C_j||t_j||\theta_j) = c_j$, in the received message, then the node considers the message as authentic and will transmit the message to the next hop. Otherwise, the node considers the message as a bogus or mistaken one and will ignore it.

C. SET-IBS protocol Execution

The protocol operates in rounds during communication, where each round is having two phases- *setup phase* for constructing clusters from cluster heads based on the received signal strength from the cluster head's *adv* message, and *steady-state phase* for transmitting data from the sensor nodes to the BS.

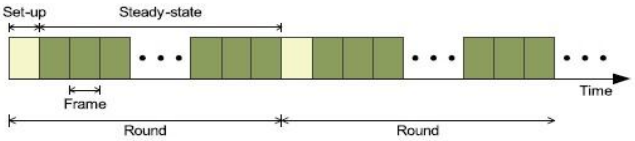
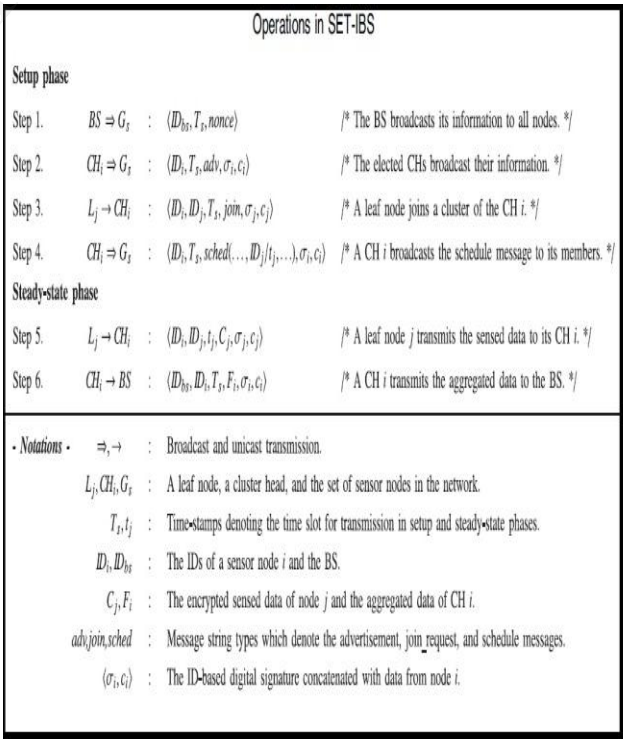


Fig 3.1: Operation in the existing SET-IBS protocol

The operations of the setup phase and steady-state phase in SET-IBS is as shown in the figure below:



Operations in SET-IBS

Setup phase

Step 1.	$BS \Rightarrow G_s$	:	$\langle ID_{bs}, T_s, nonce\rangle$	/* The BS broadcasts its information to all nodes. */
Step 2.	$CH_i \Rightarrow G_s$	:	$\langle ID_i, T_s, adv, \sigma_i, c_i\rangle$	/* The elected CHs broadcast their information. */
Step 3.	$L_j \rightarrow CH_i$	:	$\langle ID_i, ID_j, T_s, join, \sigma_j, c_j\rangle$	/* A leaf node joins a cluster of the CH *i*. */
Step 4.	$CH_i \Rightarrow G_s$	:	$\langle ID_i, T_s, sched(\ldots, ID_j/t_j, \ldots), \sigma_i, c_i\rangle$	/* A CH *i* broadcasts the schedule message to its members. */

Steady-state phase

Step 5.	$L_j \rightarrow CH_i$	:	$\langle ID_i, ID_j, t_j, C_j, \sigma_j, c_j\rangle$	/* A leaf node *j* transmits the sensed data to its CH *i*. */
Step 6.	$CH_i \rightarrow BS$	:	$\langle ID_{bs}, ID_i, T_s, F_i, \sigma_i, c_i\rangle$	/* A CH *i* transmits the aggregated data to the BS. */

· *Notations* ·	$\Rightarrow, \rightarrow$	:	Broadcast and unicast transmission.
	L_j, CH_i, G_s	:	A leaf node, a cluster head, and the set of sensor nodes in the network.
	T_s, t_j	:	Time-stamps denoting the time slot for transmission in setup and steady-state phases.
	ID_i, ID_{bs}	:	The IDs of a sensor node *i* and the BS.
	C_j, F_i	:	The encrypted sensed data of node *j* and the aggregated data of CH *i*.
	adv,join,sched	:	Message string types which denote the advertisement, join_request, and schedule messages.
	$\langle\sigma_i, c_i\rangle$	:	The ID-based digital signature concatenated with data from node *i*.

Fig 3.2: Operation in SET-IBS

The TDMA control divides the timeline into consecutive time slots in each round. The time stamp Ts is used for the signature generation in the setup phase for securing the CHs to BS data transmission whereas, the time stamp t_j is used for signature generation in the steady-state phase for securing inner cluster communications.

In the step 1 of the setup phase, at the beginning of each new round, the BS broadcasts its ID, a nonce, and the time stamp T_s to all the sensor nodes for signing and verification in the setup phase.

In step 2, a sensor node determines whether to become a cluster head based on the equation for cluster head election as in the LEACH protocol described in section I. The sensor node that decides to become a CH, then broadcasts its ID, T_s and the signature $<\sigma_{j,}c_j>$ concatenated with the *adv* message, to all the neighboring nodes in the network.

In step 3, the sensor nodes that have decided to be the leaf nodes will join with a CH based on the largest signal strength received from the *adv* message. For this the nodes will communicate with the CH_i, by sending a *join* message, that is concatenated with its own ID ID_j, the destination CH's ID ID_i, the time stamp T_s, and the signature $<\sigma_{j,}c_j>$.

In step 4, an allocation message *alloc*, that contains a time schedule from the TDMA control, for allocating a time stamp for a leaf node j is been broadcasted by a CH i to all the members in its cluster. This schedule $ID_j||t_j$ is used by each node j for the communication during the steady-state phase.

In step 5, the network turns into steady-state phase after the setup phase has been completed. Each leaf node j transmits the cipher text C_j along with the corresponding digital signature to its CH as per the time schedule allocated to it in step 4. Thus, the CH collects the data from all the leaf nodes and aggregates *the* data.

In step 6, the CHs send the fused data F to the BS along with the signature. The steady-state phase is longer when compared to the setup phase.

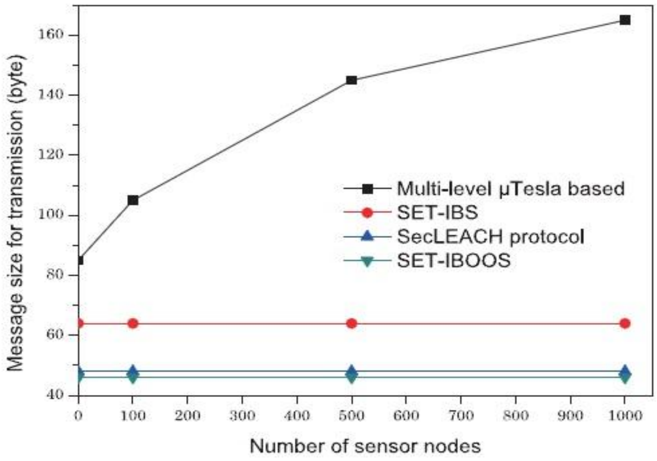


Fig 3.3: Message size for transmission Vs No. of sensor nodes

IV. PROBLEM STATEMENT

From Table I, we can see that the computational overhead of the existing SET-IBS protocol is comparatively higher than the prior secure data transmission protocols. Thus, it is clear from the table that the problem with the existing SET-IBS protocol is the high computational overhead.

One of the reasons for the computational overhead is the use of bilinear pairing for the key management in the SET-IBS protocol. The bilinear pairing operation is very time-consuming task and thus will affect the performance of the protocol. Weil and Tate pairing are the commonly used pairing algorithms that uses very complicated math and are non-trivial to compute. They are computationally expensive and time- consuming. Making them faster is still under current research.

The solution for this problem is to use a method to overcome the computation cost of bilinear pairings. Thus, by using identity based digital signature based on Elliptic Curve Digital Signature algorithm (ECDSA) [2], the computation of bilinear pairings can be avoided, since this algorithm does not use pairings. That is, by using this algorithm, the number of bilinear pairing operations will become zero.

TABLE I. COMPARISON OF SET-IBS WITH OTHER SECURE DATA TRANSMISSION PROTOCOLS

Features	***SET-IBS***	***Prior Protocols***
Key Management	Asymmetric	Symmetric
Neighborhood Authentication	Yes	Limited
Storage Cost	Comparatively Low	Comparatively High
Network Scalability	Comparatively High	Comparatively Low
Communication Overhead	Deterministic	Probabilistic
Computational Overhead	Comparatively High	Low – High

V. IDENTITY BASED DIGITAL SIGNATURE USING ECDSA

Several identity-based digital signature schemes based on bilinear pairings are there, but the computation of pairing is time consuming. While using identity based digital signature based on Elliptic Curve Digital Signature algorithm (ECDSA) [2], the computation of pairings can be avoided and therefore, about 95% of the computation cost can be saved.

A. Elliptic Curve Group and ECDSA

An elliptic curve E (F_p) is a curve defined on a prime field F_p and is defined by Weierstrass equation:

$y^2 = x^3+ax+b \quad a, b \in F_p$, with discriminant, $\Delta= 4a^3+27b^2 \neq 0$.

The points on the curve E (F_p) together with an extra point called the point at infinity denoted as O, forms a group G of order *n*.

The ECDSA is composed of three steps–key pair generation, sign and verify.

Key pair generation: For an elliptic curve E defined over a field F_q of characteristic p, and base point P, an entity A will,

1. Select a random integer d in the interval [1, n-1].
2. Compute Q= d P.
3. Q is the public key of A and d is the private key of A.

Sign: To sign a message m, D=(q, a, b ,P ,n) will be the domain parameters for A and (d, Q) will be its associated key pair. A does the following:

1. Select a random integer k, which is less than n-1.
2. Compute k P = (x_1,y_1) and convert x_1 to an integer.
3. Compute r = $x_1$modn and if r = 0, then go to step 1.
4. Compute k^{-1}modn.
5. Compute SHA-1(m) and convert this bit string to an integer e, where SHA-1() is a hash function.
6. Compute s= k^{-1}(e +d r) mod n, if s=0, then go to step 1.
7. The signature of A for message m is (r, s).

Verify: B obtains a copy of the domain parameters D of A along with A's public key Q and verifies A's signature on the message m as follows:

1. Verify whether r and s are integers between 1 and n-1.
2. Compute SHA-1(m) and convert this bit string to an integer e.
3. Compute w = s^{-1} mod n.
4. Compute u_1 = e w mod n and u_2 = r w mod n.
5. Compute X = $u_1P + u_2Q$.
6. If X = 0, then reject the signature, else, convert the x-coordinate x_1 of X to an integer x_1, and compute v = x_1 mod n and accept the signature if and only if v = r.

B. *Identity Based Elliptic Curve Digital Signaure Algorithm*

This algorithm has four steps – setup, extract, sign and verify.

Setup: This step takes a security parameter k, and returns the system parameters and the master key. For this, the PKG works as follows:

1. Choose a k-bit prime number p and determine the parameters D = (q, a, b, P, n).
2. Choose the msk $x \in_R Z_n^*$ and compute the master public key P_{pub} = x P.
3. Choose H_1: $\{0,1\}^* \rightarrow Z_n^*$, H_2: $\{0,1\}^* \times G \rightarrow Z_p^*$, where H_1 and H_2 are two cryptographic hash functions and the order of group G is n.
4. Publish the system parameters {D, n, P_{pub},H_1, H_2} and keep x secretly.

Extract: The private key generator PKG does the following for each entity A with ID ID_A:

1. Choose a random $r \in_R Z_n^*$ and compute R_A = r P and h= $H_1(ID_A||R_A)$.
2. Compute s_A = (r + h x) mod n.
3. Now A's private key (R_A, s_A) is transmitted to A via a secure channel and A can validate this private key by checking whether the equation: $s_A P = R_A + H_1(ID_A||R_A) P_{pub}$ holds.

Sign: To sign a message m, the entity A does the following:

1. Generate signature (r, s) of m using ECDSA signature generation algorithm in section II A.
2. The signature of entity A for m is generated as (R_A, r, s).

Verify: To verify the signature of A, an entity B does the following after obtaining A's domain parameters and the public key P_{pub}.

1. Compute h= $H_1(ID_A||R_A)$.
2. Compute Q = $R_A + h P_{pub}$.
3. Verify the signature (r, s) of message m using ECDSA signature verification algorithm in section II A, where Q is the public key.
4. Output the result of signature verification.

TABLE II. NO. OF OPERATIONS IN IB-ECDSA

Item	No. of operations in IB-ECDSA	
	Sign	*Verify*
No. of scalar multiplications	1	2
No. of hash operations	1	1
No. of bilinear pairing operations	0	0

Thus, using IB-ECDSA, the computation overhead for pairing operations can be saved. The number of scalar multiplications, hash operations and bilinear pairing operations during the signing and verification phases of IB-ECDSA is shown in TABLE I. We can see that the number of bilinear pairing operations required during both signing and verification phase is zero.

VI. Proposed SET-IBS Using IB-ECDSA

The enhanced SET-IBS protocol using IB-ECDSA is based on the key management without bilinear pairings. In the existing SET-IBS scheme, bilinear mapping e is also been published along with the domain parameters. Since, the computation of bilinear mapping is time consuming; we are using IB-ECDSA [2] in the existing SET-IBS protocol [1] in order to avoid bilinear pairing operations. The enhanced SET-IBS works as follows.

A. Protocol Initialization for the SET-IBS using IB-ECDSA

The execution of the proposed SET-IBS using IB-ECDSA is similar to that of the existing SET-IBS protocol and the difference between the two is that; here we are avoiding the bilinear mapping operation during the protocol initialization phase. The SET-IBS protocol [1] executes in rounds, which consists of two phases – setup phase and steady state phase, in each round. The time is divided into successive time slots by the TDMA control as that in LEACH protocol .Mainly two timestamps are there for communication- T_s and t_j, T_s for BS to node communication and t_j for leaf node to cluster head communication. The user's public key is ID||t and the corresponding private pairing parameters are preloaded in the nodes during the protocol initialization. Additively homomorphic encryption scheme, in which an operation performed on the plain text is equivalent to that performed on the cipher text, is been adopted in SET-IBS protocol for encrypting the sensed data, in order to allow efficient aggregation of data both at the CHs and at the BS.

Setup: In the protocol initialization phase, the BS performs the key pre distribution operations to all the sensor nodes. This phase can be viewed as the setup phase for the BS. The operation is as follows:

- Generate k, which is the homomorphic encryption key to encrypt the sensed plaintext, where $k \in [m-1]$, m is a large integer.
- Generate the pairing parameters $(p, q, E/F_p, G, n)$. Select a generator P of G stochastically.
- Choose two cryptographic hash functions: H and h, $H: \{0,1\}^* \rightarrow Z_n^*$, $h: \{0,1\}^* \times G \rightarrow Z_p^*$.
- Pick a random integer $\tau \in Z_q^*$ as the master key *msk* and set $P_{pub} = \tau P$ as the master public key.
- Preload the parameters $(k, m, p, q, E/F_p, G_, n, H, h, P, \tau)$ in each sensor node.

B. Key management in SET-IBS using IB-ECDSA

If a leaf node j wants to transmit a message M to the cluster head CH_i, then the node j will encrypt the message using the homomorphic encryption key k to get the ciphertext C. The SET-IBS scheme does three operations- extraction, signing and verification, after the protocol initialization step.

Extraction: The sensor node j will extract its private key sek_j from its ID, msk τ and timestamp t_j, which is node j's timestamp in the current round generated by its CH i from the TDMA control. Node j does the following for extracting its private key sek .

1. Choose a random number $r \in Z_n^*$and compute $R_j = r P$, where P is the generator point.
2. Node j will then compute $sek_j = r + \tau H (ID_j||R_j||t_j)$.

Signing: For signing the ciphertext C_j generated using the homomorphic encryption scheme using the key k; the node j does the following:

1. Choose a random integer f such that, $1 \leq f \leq n-1$.
2. Compute $f P = (x_1 , y_1)$ and convert x_1 to an integer $x_{1.}$
3. Compute $\sigma_j = x_1 \bmod n$. If $\sigma_j = 0$, then go to step 1.
4. Compute $f^{-1} \bmod n$.
5. Compute SHA-1(C_j) and convert this bit string to an integer e.
6. Compute $c_j = f^{-1}(e + sek_j \sigma_j) \bmod n$.
7. Node j's signature for the ciphertext C_j is (R_j, σ_j, c_j).

Verification: For verifying the signature each node obtains the authentic copy of the node j's domain parameters along with the public key P_{pub}. On receiving the message, each node verifies the authenticity of the message by first checking the timestamp t_j for the current round in order to verify whether the message is fresh or not. If the node verifies that the message is fresh, then it does the following to verify the signature (R_j, σ_j, c_j) on ciphertext C_j.

1. Compute $Q = R_j + H (ID_j||R_j||t_j) P_{pub}$, where Q is the public key.
2. Verify that (σ_j, c_j) are integers in the interval [1, n-1].
3. SHA-1(C_j) and convert this bit string to an integer e.
4. Compute $w = c_j^{-1} \bmod n$.
5. Compute $u_1 = e\, w \bmod n$ and $u_2 = \sigma_j\, w \bmod n$.
6. Compute $X= u_1 P + u_2 Q$.
7. If X=O, then reject the signature. Otherwise, convert the x-coordinate x_1 of X to an integer x_1, and compute $v = x_1 \bmod n$.
8. Accept the signature if and only if $v = \sigma_j$.

Thus, the above described enhanced SET-IBS algorithm using IB-ECDSA without bilinear pairings helps in reducing the computation overhead of the existing SET-IBS scheme and thus up to 95% of the computation cost can be saved when compared to other identity based signature protocols using pairing.

VII. Conclusions

The enhanced SET-IBS protocol using IB-ECDSA is based on the key management without bilinear pairings. In the existing SET-IBS scheme, bilinear mapping e is also been published along with the domain parameters. Since, the computation of bilinear mapping is time consuming; we are using IB-ECDSA [2] in the existing SET-IBS protocol [1] in order to avoid bilinear pairing operations. Replacing DSA with ECDSA, the key size will also be low. For example, to provide a security level of 80 bits, the size of a DSA public key is at least 1024 bits, whereas the size of an ECDSA public key would be 160 bits. On the other hand, the signature size is the same for both DSA and ECDSA. For a security level of ***t*** bits, the public key will be ***2t*** bits and the signature size will be ***4t*** bits using ECDSA.

References.

[1] G. Huang Lu, Jie Li and Mohsen Guizani, "Secure and Efficient Data Transmission for Cluster-Based Wireless Sensor Networks," IEEE Transactions on Parallel and Distributed Systems, vol. 25, no. 3,pp.750-761, March 2014.

[2] Hu Jin, He Debiao, Chen Jianhua,School of Mathematics and Statistics,Wuhan University,Wuhan, China," An Identity Based Digital Signature from ECDSA", 2010 Second International Workshop on Education Technology and Computer Science

[3] John Bethencourt, Computer Sciences Department, Carnegie Mellon University," Intro to Bilinear Maps".

[4] D. Boneh and M. Franklin, "Identity-Based Encryption from the Weil Pairing," Proc. 21st Ann. Int'l Cryptology Conf. Advances in Cryptology (CRYPTO '01), pp. 213-229, 2001.

[5] A.A. Abbasi and M. Younis, "A Survey on Clustering Algorithms for Wireless Sensor Networks," Computer Comm., vol. 30, nos. 14/15, pp. 2826-2841, 2007. J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68-73.

[6] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "An Application-Specific Protocol Architecture for Wireless Microsensor Networks," IEEE Trans. Wireless Comm., vol. 1, no. 4, pp. 660-670, Oct. 2002.