A Bilinear Pairing Based Secure Data Aggregation Scheme for WSNs

Vimal Kumar
Department of Computer Science
University of Waikato
Hamilton, New Zealand
vimal.kumar@waikato.ac.nz

***Abstract*—End to end secure data aggregation scheme for wireless sensor networks that are based on public key cryptography generally use elliptic curves. However elliptic curve based protocols require messages to be mapped to elliptic curves before performing any operations and finally reverse mapped to retrieve the message back. No mapping function, however, which is both homomorphic and has an efficient reverse mapping function is currently known. The mapping functions used in many previous protocols require brute forcing to reverse map the message from a point on the elliptic curve. This solution may be feasible on a base station with unlimited energy and processing power but it means that decrypting becomes very inefficient on ordinary sensors. We propose a secure data aggregation algorithm based on bilinear pairing that avoids this problem and makes decrypting data feasible on ordinary sensors.**

***Index Terms*—data aggregation, wireless sensor network, homomorphic encryption, bilinear pairing, elliptic curves**

I. Introduction

A Wireless Sensor Network (WSN) is a network of small battery powered motes also known as wireless sensors, which have sensing, processing and communication capabilities. WSNs have applications in many fields, particularly in places where deploying a wired network is infeasible, dangerous or too expensive. Wireless sensors have the advantage of being small in size and relatively inexpensive and therefore can be deployed in large numbers. These sensors once deployed can be made to organize themselves in a network and start sensing and gathering data. They however work on batteries and a small size means that the battery capacity is also relatively small. Since, the sensors are generally deployed in a rugged environment, replacing the batteries is usually not an option. In WSNs, it is therefore imperative that any algorithm running on the sensors be energy efficient.

In-network data aggregation has been previously proposed as a method of conserving energy in WSNs. It reduces the total number of radio transmissions required to transport sensor data to the base station. We know that radio transmission constitutes a substantial part of a wireless node's total energy expenditure [13]. Reducing the total number of transmissions therefore saves a large amount of energy, resulting in greater network life-time. Data aggregation however introduces new security challenges. For an intermediate node in the aggregation tree to aggregate data received from nodes in its subtree, the data needs to be either in plaintext, or the intermediate node needs to have the shared keys to decrypt everything it receives. While the former is inherently in-secure, the latter is also not very secure and is expensive in terms of energy. An intermediate node which has shared keys of all the nodes in its subtree makes it a very attractive target for attackers. Moreover, the additional energy-intensive task of decrypting all the received data and encrypting the aggregate means intermediate nodes will run out of energy and die much earlier than the leaf nodes. This will render the network useless, even though there may be many live nodes in the network. In a complete tree of N nodes and degree δ, the number of intermediate nodes are $\simeq N/\delta$. Thus, if our network is a binary tree, and the intermediate nodes run out of battery, as many as half the nodes may be alive while the network is unusable. A binary tree is the best case scenario, in trees of higher degrees, this problem will be even worse with only a small number of dead intermediate nodes crippling the entire network.

To avoid this problem, an *end-to-end* data aggregation scheme is required. In such schemes once the data is encrypted at a node, it is only decrypted at the base station while the intermediate nodes aggregate the encrypted data. Such schemes are commonly known as concealed data aggregation schemes and can be realized using homomorphic encryption. Homomorphic encryption allows us to operate on encrypted data and eliminates the need of intermediate decryption if certain aggregation functions need to be applied on it. In general, an encryption scheme is said to be homomorphic if it allows for the following property to hold,

$$enc(a) \otimes enc(b) = enc(a \otimes b)$$

where $\otimes$ denotes a mathematical operation. Generally the mathematical operations supported by encryption schemes are addition and multiplication. Schemes that support either one of addition or multiplication are called partially homomorphic schemes while the ones that support both are known as fully homomorphic schemes. While fully homomorphic schemes [2] [5] [17] are still too inefficient to be useful in WSNs, there have been many *end-to-end* schemes proposed [3] [4] [8] [12] [15], etc. which make use of partially homomorphic encryption. Many of these schemes make use of Elliptic

Curve Cryptography (ECC) which allows them to be more efficient than traditional public key schemes. Elliptic curve cryptography works on points defined on a predefined elliptic curve. Therefore, before an integer $k \in \mathbb{Z}$ can be encrypted using ECC based schemes, it needs to be mapped to a point on the curve. For an ECC based scheme to be *end-to-end*, the encryption scheme obviously needs to be homomorphic but there are two additional requirements that need to be satisfied. First, the mapping function $map(\mathbb{Z}) \rightarrow G_1$ that maps an integer to a point on the curve needs to be homomorphic and second, it should have an efficient reverse mapping function $rmap(G_1) \rightarrow \mathbb{Z}$ that can map a point on the curve to an integer. In the current state of the art however, there doesn't exist a mapping function which is both homomorphic and has an efficient reverse mapping function. Reverse mapping an integer from a point requires brute forcing which consumes a large amount of energy. Previous schemes got around this restriction by assuming that the decryption operation is only performed at the base station and base station has unlimited energy [8] [12] [15]. They also assume the use of heuristics such as Pollard's-rho method [14], to reduce the amount of computation involved. Such solutions however are not applicable, if the decryption needs to be performed on a sensor node, such as in the system model defined in [16] and [20], where data is collected on an intermediate repository node in the network and then fine grained access control is applied on the data.

In this paper we present a homomorphic encryption scheme based on bilinear pairing. In our construction the mapping is done from integers to an extension field, for which an easy and efficient reverse mapping function exists. It does not require brute forcing which makes it possible to perform decryption on wireless sensors nodes. Since homomorphic encryption schemes are by definition malleable, we also provide a construction for a pairing based signature scheme which can be made additive by simple modifications.

Formally, our contributions in this paper are as follows.

- A construction for a pairing based homomorphic encryption scheme that does not require brute forcing for decryption
- A construction for a pairing based additive digital signature scheme
- An *end-to-end* secure data aggregation scheme using the above two constructions to provide confidentiality and integrity of data

II. Related Work

Previous work on end to end secure data aggregation schemes can be divided into those that use symmetric key cryptography and those that use ECC based public key cryptography. Symmetric key based secure data aggregation has been discussed in [6] [10] [18] etc. but our focus is on public key based secure data aggregation.

In [11], the authors compared various elliptic curve based encryption algorithms in terms of energy and bandwidth requirement. Their comparison showed that EC-Elgamal is more efficient than others in terms of energy required for encryption and bandwidth consumption. The decryption however depends on the efficiency of the reverse map function. This led to various schemes being proposed that used EC-Elgamal for end-to-end encryption for data aggregation.

Chen et. al. in [4] proposed a scheme that used EC-Elgamal for end-end encryption and Boneh et. al's aggregate signature scheme for providing end-to-end data integrity. The aggregate signature scheme however requires all individual data items to be present at the verifier before the aggregate signature can be verified. To accomplish this, the scheme encodes data in a fashion such that aggregating data leads to it being concatenated. The aggregated (concatenated) data can then be decoded at the base station into individual data items. This solution however, is not scalable as the size of the ciphertext increases with the number of nodes in the network. Parmar et. al. in [12] use the same setup for encryption but use homomorphic MACs for data integrity. Homomorphic MACs however suffer from the same problem as aggregate signatures in [4]. To remedy this problem, Parmar et. al's scheme relies on pre-distributed symmetric key secrets among the nodes. [8] also uses EC-Elgamal for end-to-end encryption but uses a modified version of elliptic curve digital signature scheme (ECDSA). Their modification involves using the message directly for signatures instead of hashing it first. This makes the signature insecure particularly in a wireless sensor network with multiple sensors sensing the same environment. In such a situation multiple sensors will sense the same data value, which will give the adversary additional information with many plaintext and signature pairs.

There have been numerous other works such as [1] and [9] that all use EC-Elgamal for encryption and some other integrity preserving technique to provide data integrity. However, as mentioned previously, all these schemes rely on the assumption that the decrypting and the verifying entity has enough resources to brute force the decryption. Such solutions are not acceptable in WSNs which need public key systems and where the decrypting and verifying entity is a resource constrained wireless sensor. In this paper, we provide a homomorphic encryption scheme that can be used in such WSNs. We also provide an additive digital signature algorithm that can be used for data integrity. Further we outline a secure data aggregation algorithm that uses our new cryptographic constructs.

III. Models

A. System Model

We consider a wireless sensor network with N nodes where each node has a unique identity $i \in \mathbb{Z}_N$. One of the nodes is designated the sink node and is denoted by β. In our setup the sink node does not need to be a powerful and energy rich base station. We assume the nodes are arranged in a hierarchical manner, such as a tree rooted at β. This can be achieved by using an algorithm such as the one described in [8]. The data is sensed by the sensor nodes and then passed upstream to their parent nodes. The parent nodes aggregate the data received

from their children and pass it to their parents. This process is followed until the sink node β receives the aggregate of the data sensed by all the nodes in the network.

B. Adversary Model

From an adversary's point of view, in such a system, the aggregator nodes are more attractive than the leaf nodes. By capturing nodes at the aggregator level an adversary would be able to create a larger impact on the system. We, therefore, focus on the security of the aggregator nodes. The goals of the adversary are

- To compromise the confidentiality of an aggregator's data.
- To inject false data into the network.

IV. Preliminaries

Our scheme is based on bilinear pairing, therefore in this section we go over some background on bilinear pairing and our security assumptions based around them.

A. Bilinear pairing

Let G_1 and G_T be cyclic groups of order q for some large prime q and let g be a generator of G_1. A bilinear pairing then is an injunctive function $e : G_1 \times G_1 \rightarrow G_T$, which has the following properties.

- Bilinearity: $\forall g, h \in G_1, \forall a, b \in \mathbb{Z}_q^*$, we have $e(g^a, h^b) = e(g, h)^{ab}$
- Non-degeneracy: $e(g, h) \neq 1$
- Computability: There exists an efficient algorithm for computing $e(g, h), \forall g, h \in G_1$

B. Security Assumptions

1) Bilinear Diffie-Hellman (BDH) Assumption : The security of our encryption scheme is based on the BDH assumption. The BDH assumption states that given two groups G_1 and G_T of prime order q where g is a generator of G_1 and given a bilinear function $e : G_1 \times G_1 \rightarrow G_T$ and given the information $< g, g^a, g^b, g^c >$ it is hard to compute $e(g, g)^{abc}$ in polynomial time. Assuming a, b, c are chosen randomly over $\mathbb{Z}_q^*$. In other words, given $< g, g^a, g^b, g^c >$, the probability of computing $e(g, g)^{abc}$ in polynomial time is ϵ, where ϵ is a negligibly small number.

2) gap Bilinear Diffie-Hellman (gap-BDH) Assumption: The security of our signature scheme is based on the gap-BDH assumption. The gap-BDH assumption works on groups where BDH problem is hard but decisional BDH is solvable in polynomial time. The assumption states that given two groups G_1 and G_T of prime order q where g is a generator of G_1 and given a bilinear function $e : G_1 \times G_1 \rightarrow G_T$ and given the information $< g, g^a, g^b, g^c >$ and $p \in G_T$, it is possible to check if $p = e(g, g)^{abc}$ but hard to compute $e(g, g)^{abc}$ in polynomial time.

V. Pairing Based Homomorphic Encryption Algorithm

In this section we describe the construction of our pairing based encryption algorithm. Our encryption algorithm is homomorphic and consists of four sub-algorithms,

- *Key-Gen*: Takes a security parameter λ and generates two keys, an encryption key, EK and a decryption key, DK. EK is public while DK is private
- *Encrypt*: Takes a message M and the encryption key EK and generates ciphertext C.
- *Decrypt*: Takes the ciphertext C and the decryption key DK and retrieves the message M.
- *Aggregate*: Takes two ciphertexts C_1 and C_2, created using the same encryption key EK and generates ΣC, such that ΣC generates $M_1 + M_2$, when Decrypt is applied on it.

The proposed encryption scheme works as follows:

Key-Gen: Choose a number x for which it is easy to calculate the discrete log. Based on the security parameter λ, choose a random number $s \in \mathbb{Z}_q^*$ and compute $e(g, h)^s$. Output the decryption key $DK = \{s\}$ and the encryption key $EK = \{x, g, h, e(g, h)^s\}$. EK is public and given to the encrypting party(s) while DK is private and is given to the decrypting party.

Encrypt: The ciphertext C consists of two parts C' and C''. To encrypt a message M, using the encryption(public) key EK, the encrypting node i chooses a random number $r \in \mathbb{Z}_q^*$ and computes $(e(g, h)^s)^r = e(g, h)^{sr}$. It then outputs,

$$C_i = \begin{cases} C_i' = x^M e(g, h)^{sr}, \\ C_i'' = h^r \end{cases}$$

Decrypt: To decrypt C_i and retrieve the message, the decrypting party computes

$$\begin{aligned} M &= dlog_x(C_i'/e(g^s, C_i'')) \\ &= dlog_x(x^M e(g, h)^{sr}/e(g, h)^{sr}) \end{aligned}$$

Aggregate: To aggregate ciphertexts C_i and C_j from two nodes i, j, compute

$$C_{i+j} = \begin{cases} C' = C_i' * C_j', \\ C'' = C_i'' * C_j'' \end{cases}$$

Note that,

$$\begin{aligned} C_i' * C_j' &= x^{M_i} e(g, h)^{sr_i} * x^{M_j} e(g, h)^{sr_j} \\ &= x^{M_i + M_j} e(g, h)^{s(r_i + r_j)} \end{aligned}$$

and,

$$\begin{aligned} C_i'' * C_j'' &= h^{r_i} * h^{r_j} \\ &= h^{r_i + r_j} \end{aligned}$$

Thus, decrypting C_{i+j} will retrieve M_{i+j}.

VI. Pairing Based Signature Algorithm

The pairing based signature algorithm consists of three sub-algorithms.

- *Key-Gen*: Takes a security parameter λ and generates two keys, a signing key, SK and a verification key, VK. SK is private while VK is public.
- *Sign*: Takes a message M and the signing key SK and generates a signature σ
- *Verify*: Takes the signature σ, the verification key VK and a message $\hat{M}$ and outputs $true$ if σ is a valid signature on $M = \hat{M}$

The proposed scheme works as follows.

Key-Gen: Choose a random number $d \in \mathbb{Z}_q^*$ and calculate g^d. Output the signing key (secret) $SK = \{d\}$ and the verification key (public) $VK = \{g^d, h\}$.

Sign: For signing a message M, the signing party computes

$$\sigma = e(g,h)^{Md}$$

Verify: For verification of the signature σ over a received message $\hat{M}$, the verifier computes

$$\begin{aligned}\hat{\sigma} &= e(g^d, h^{\hat{M}}) \\ &= e(g,h)^{d\hat{M}}\end{aligned}$$

If $\sigma == \hat{\sigma}$, the verification is successful.

A. Additive Digital Signature Scheme

Additive digital signatures are those that allow the summation of two signatures σ_i and σ_j on two messages M_i and M_j respectively, such that the $\sigma_i + \sigma_j$ verifies $M_i + M_j$. In this subsection we describe the construction of an additive digital signature scheme based on the pairing based signature scheme described above.

Key-Gen: Choose a random number $d \in \mathbb{Z}_q^*$ and compute g^d. For each signing party i generate a random nonce η_i. Output the signing key $SK_i = \{d, \eta_i, f(.)\}$, where $f(.)$ is the description of a pseudo random function (PRF). The verification key then will be $VK_i = \{g^d, \eta_i, f(.)\}$.

Sign: To sign a message M_i, the signing party i computes

$$\sigma_i = e(g,h)^{(M_i + f(\eta_i))d}$$

Sign-Aggregate: Two signatures σ_i and σ_j can be aggregated as

$$\begin{aligned}\sigma_{i+j} &= \sigma_i * \sigma_j \\ \sigma_{i+j} &= e(g,h)^{(M_i+f(\eta_i))d} * e(g,h)^{(M_j+f(\eta_j))d} \\ \sigma_{i+j} &= e(g,h)^{(M_i+f(\eta_i))d+(M_j+f(\eta_j))d} \\ \sigma_{i+j} &= e(g,h)^{((M_i+M_j)+f(\eta_i)+f(\eta_j))d}\end{aligned}$$

in general, for n signatures,

$$\sigma_{\Sigma_i} = e(g,h)^{((\Sigma M_i)+\Sigma(\eta_i))d}$$

Verify: To verify the aggregate signature σ_{Σ_i} , compute the aggregate of the received messages $\Sigma\hat{M}$ and then $\Sigma f(\eta)$. Then calculate

$$\begin{aligned}\hat{\sigma}_{\Sigma_i} &= e(g^d, h^{\Sigma\hat{M}+\Sigma f(\eta))} \\ &= e(g,h)^{d(\Sigma\hat{M}+\Sigma f(\eta))}\end{aligned}$$

If $\sigma_{\Sigma_i} == \hat{\sigma}_{\Sigma_i}$, the verification is successful.

Note that, even though we use the message M directly in our additive signature scheme like [8], our scheme is not susceptible to existential forgery attacks, because we introduce randomness in the signatures through $f(\eta)$. In our signature scheme, when additive signatures are not required, M can be replaced by $h(M)$ in the signing and verification sub-algorithms without any loss of functionality to avoid exitential forgery attacks. Here $h()$ refers to a cryptographic hash function.

VII. The Secure Data Aggregation Scheme

Based on the cryptographic constructs described above, in this section we present our secure data aggregation algorithm. The secure data aggregation algorithm makes use of the pairing based homomorphic encryption algorithm for providing *end-to-end* data confidentiality. It further makes use of the pairing based additive digital signature scheme to provide *end-to-end* data integrity.

A. Setup

Our system model has been described in section III-A. We assume that the sensor nodes in the network are organized in a tree as shown in Fig. 1. During the network setup the network administrator generates a public key pair (EK, DK). EK is loaded on the sensor nodes for encryption while DK is loaded on the sink node for decryption. For each sensor node i the network administrator also generates the pubic key pair (SK_i, VK_i). Each node i is loaded with the signing key SK_i while the sink node is given the verification key VK_i for all the nodes. The administrator also generates a PRF $f(.)$ and loads it on every sensor as well as the sink. Thus, each sensor node is loaded with the following parameters,

$$\{EK \cup SK\} = \{x, g, h, e(g,h)^s, d, f(.), \eta_i\}$$

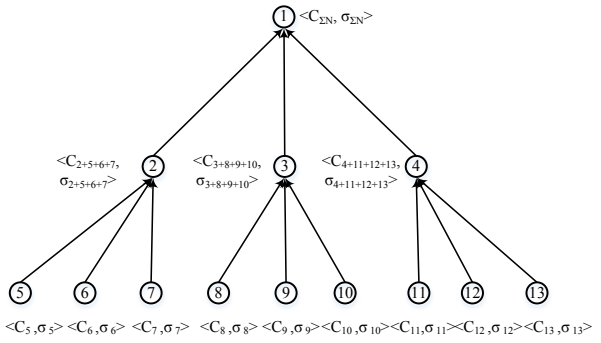

Fig. 1. Example Network

The sink node is loaded with the decryption key DK and the verification key VK_i for all nodes. The sink node has the following parameters

$$\{DK \cup VK_i \cup ... \cup VK_N\} = \{x, g, h, e(g,h), s, g^d, f(.), \eta_i \forall i \in \mathbb{Z}_N\}$$

B. At the sensor nodes

Each node i has a sensor reading M_i that needs to be encrypted. Sensor nodes start by encrypting M_i using EK and generate C_i. Each node then further generates a signature σ_i on the reading M_i. Leaf nodes then send the tuple $< C_i, \sigma_i >$ forward to their parent nodes. Intermediate nodes gather the tuples $< C_i, \sigma_i >$ sent by their children nodes and aggregate the ciphertext and digital signatures, $< C_{sum}, \sigma_{sum} >$. Intermediate nodes then send this tuple up the hierarchy in the aggregation tree until the aggregated ciphertexts and digital signatures reach the sink. Fig. 1 shows the the data transmission and aggregation in the network tree.

C. At the sink

The sink node after receiving the aggregate tuple $< C_{sum}, \sigma_{sum} >$, decrypts C_{sum} using the decryption key DK to recover $\hat{M}_{sum}$. For verifying the decrypted message $\hat{M}_{sum}$, the sink node computes $\Sigma f(\eta)$ and then verifies $\hat{M}_{sum}$ using VK.

VIII. Evaluation

In this section we first analyse the security of our scheme and then provide a discussion on its performance.

A. Security Analysis

The ciphertext generated by a sensor node in our scheme C_i consists of two parts $C_i' = x^M e(g,h)^{sr}$ and $C_i'' = h^r$, which a leaf node sends to an intermediate node. As defined in our adversary model, we assume that the intermediate node is under the control of an adversary A. The first goal of A is to find the value of the message M. The adversary knows the public parameters $\{x, g, h, e(g,h)^s, d, f(.)\}$. The adversary A, would be able to get the value of the message M, if it can calculate $z = e(g,h)^{sr}$. This will enable it to get $x^M = C_i'/z$, and by definition it is easy to calculate the discrete log of x and find M.

Let's denote our encryption scheme by $\mathbb{E}$. The advantage of the adversary A over $\mathbb{E}$ can be written as,

$$Adv_A(\mathbb{E}) = Adv_A(e(g,h)^{sr})$$

given $\{g, h, h^r, e(g,h)^s\}$. However, we also know that according to the BDH assumption as described in section IV-B1, $Adv_A(BDHP) = \epsilon$, where ϵ is a very small negligible value and $BDHP$ stands for the Bilinear Diffie-Hellman Problem. Upon close inspection one can see that the problem of computing $e(g,h)^{sr}$ from the given information is in fact the BDH problem. Therefore,

$$Adv_A(\mathbb{E}) = \epsilon$$

In other words, deciphering the value of M from the ciphertext in our scheme is at least as hard as solving the BDH problem.

The second goal of adversary A, is to be able to inject false data in the network, that will be accepted by the sink node. The adversary has the public parameters as described above and the signature $\sigma_i = e(g,h)^{(M_i + f(\eta_i))d}$ produced by a leaf node i on a message M_i. The adversary will succeed in its goal, if it is able to come up with M_A, such that $e(g,h)^{M_A} = e(g,h)^{(M_i + f(\eta_i))d}$. Denoting our additive signature scheme by $\mathbb{S}$, the advantage of A over $\mathbb{S}$ can be written as,

$$Adv_A(\mathbb{S}) = Pr[e(g,h)^{M_A} = e(g,h)^{(M_i + f(\eta_i))d}]$$

With the gap-BDH assumption, A can make guesses about M_A and perform the above comparison. Thus, the above expression can be simplified to,

$$Adv_A(\mathbb{S}) = Pr[M_A = (M_i + f(\eta_i))d]$$

The adversary can launch an exhaustive attack, comparing every possible value for M_A. The probability, then will depend upon the size of the search space for M_A, which is the large prime q as defined in section IV. Thus,

$$Adv_A(\mathbb{S}) = 1/|q|$$

Therefore, if q is chosen to be large enough to make $1/|q|$ negligible, the adversary will not be able to inject false data in the network.

B. Complexity Analysis

Table I presents the breakdown of the various sub-algorithms of our scheme in terms of cryptographic operations. In the table Key-Gen(E) refers to the key generation part of the encryption algorithm and Key-Gen (S) refers to the key generation part of the signature algorithm. We also assume that x^M in the ciphertext can be written as an element of G_T and therefore the computation of C' in the encryption sub-algorithm involves one G_T multiplication.

TABLE I
COMPUTATION COMPLEXITY OF VARIOUS SUB-ALGORITHMS

Operation	G_1 **Exponentiation** (G_1X)	G_T **Multiplication** (G_TM)	G_T **Exponentiation** (G_TX)	**Pairing** (P)
Key-Gen(E)	0	0	0	1
Encrypt	1	1	1	0
Decrypt	0	1	0	1
Aggregate	0	1	0	0
Key-Gen(S)	1	0	0	0
Sign	0	0	1	0
Verify	1	0	0	1
Sign-Agg	0	1	0	0

TABLE II
COMPUTATION COMPLEXITY ACCORDING TO NODE TYPE

Node Type	**Computation Complexity**
Leaf Nodes	$1G_1X + 1G_TM + 2G_TX$
Intermediate Nodes	$2G_TM$
Sink Node	$1G_1X + 1G_TM + 2P$

Table II shows the computation complexity at each type of node in the network. For this table we assume that the leaf nodes perform Encrypt and Sign, while the intermediate node only perform the Aggregate and the Sign-Agg functions and the sink node performs the Decrypt and the Verify functions. As can be seen from the table, the Aggregate and the Sign-Agg functions performed at the intermediate nodes only require two extension field (G_T) multiplication operations which typically are lightweight operations [7]. The intermediate nodes therefore lose very little energy as a result of these operations. However, what is interesting to note is that the energy consumption of sink and the leaf nodes looks very similar, with the leaf nodes requiring two extension field (G_T) exponentiation operations compared to two pairing operations required by the sink nodes in addition to one element exponentiation and one extension field multiplication. The pairing operation is typically only slightly more energy intensive than extension field G_T exponentiation operation [19], thus the sink node expends only slightly more energy than the leaf nodes, which is not possible when using EC-Elgamal based schemes.

IX. CONCLUSION AND FUTURE WORK

In this work, we have proposed pairing based homomorphic encryption and digital signature algorithms and have provided the construction of a secure data aggregation scheme based on these algorithms. This secure data aggregation scheme can be used in scenarios where the sink node is an ordinary node and not a powerful base station. We provided a theoretical complexity analysis of our scheme to support this claim. We have also provided a security analysis to prove that our scheme is secure against the adversary model we stated. Our next step would be to implement the algorithms on various off the shelf mote platforms to quantify the increased network life time our scheme provides compared to schemes using EC-Elgamal.

REFERENCES

[1] T. Ahmad, J. Hu, and S. Han. An efficient mobile voting system security scheme based on elliptic curve cryptography. In *Network and System Security, 2009. NSS'09. Third International Conference on*, pages 474–479. IEEE, 2009.

[2] Z. Brakerski and V. Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. *SIAM Journal on Computing*, 43(2):831–871, 2014.

[3] C. Castelluccia, A. C.-F. Chan, E. Mykletun, and G. Tsudik. Efficient and provably secure aggregation of encrypted data in wireless sensor networks. *ACM Trans. Sen. Netw.*, 5(3):20:1–20:36, 2009.

[4] C. M. Chen, Y. H. Lin, Y. C. Lin, and H. M. Sun. Rcda: Recoverable concealed data aggregation for data integrity in wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 23(4):727–734, 2012.

[5] C. Gentry and S. Halevi. Implementing gentry's fully-homomorphic encryption scheme. In *EUROCRYPT*, volume 6632, pages 129–148. Springer, 2011.

[6] V. Kumar and S. Madria. Pip: Privacy and integrity preserving data aggregation in wireless sensor networks. In *2013 IEEE 32nd International Symposium on Reliable Distributed Systems*, pages 10–19, 2013.

[7] V. Kumar and S. Madria. Distributed attribute based access control of aggregated data in sensor clouds. In *Reliable Distributed Systems (SRDS), 2015 IEEE 34th Symposium on*, pages 218–227. IEEE, 2015.

[8] V. Kumar and S. K. Madria. Secure hierarchical data aggregation in wireless sensor networks: Performance evaluation and analysis. *Mobile Data Management, IEEE International Conference on*, 0:196–201, 2012.

[9] L. Li, A. A. A. El-Latif, and X. Niu. Elliptic curve elgamal based homomorphic image encryption scheme for sharing secret images. *Signal Processing*, 92(4):1069–1078, 2012.

[10] Q. Li, G. Cao, and T. F. La Porta. Efficient and privacy-aware data aggregation in mobile sensing. *IEEE Transactions on Dependable and Secure Computing*, 11(2):115–129, 2014.

[11] E. Mykletun, J. Girao, and D. Westhoff. Public key based cryptoschemes for data concealment in wireless sensor networks. In *Communications, 2006. ICC'06. IEEE International Conference on*, volume 5, pages 2288–2295. IEEE, 2006.

[12] K. Parmar and D. C. Jinwala. Malleability resilient concealed data aggregation in wireless sensor networks. *Wireless Personal Communications*, 87(3):971–993, 2016.

[13] S. Peter, K. Piotrowski, and P. Langendoerfer. On concealed data aggregation for wireless sensor networks. In *Proceedings of the 4th IEEE Consumer Communications and Networking Conference, CCNC '07*, 2007.

[14] J. M. Pollard. Monte carlo methods for index computation (mod-p). *Mathematics of computation*, 32(143):918–924, 1978.

[15] M. B. O. Rafik and F. Mohammed. Fast and secure implementation of ecc-based concealed data aggregation in wsn. In *Global Information Infrastructure Symposium - GIIS 2013*, pages 1–7, 2013.

[16] S. Ruj, A. Nayak, and I. Stojmenovic. Distributed fine-grained access control in wireless sensor networks. In *Parallel Distributed Processing Symposium (IPDPS), 2011 IEEE International*, pages 352–362, 2011.

[17] M. Van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan. Fully homomorphic encryption over the integers. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 24–43. Springer, 2010.

[18] D. Westhoff and O. Ugus. Malleability resilient (premium) concealed data aggregation. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2013 IEEE 14th International Symposium and Workshops on a*, pages 1–6. IEEE, 2013.

[19] X. Xiong, D. S. Wong, and X. Deng. Tinypairing: A fast and lightweight pairing-based cryptographic library for wireless sensor networks. In *Wireless Communications and Networking Conference (WCNC), 2010 IEEE*, pages 1–6. IEEE, 2010.

[20] S. Yu, K. Ren, and W. Lou. Fdac: Toward fine-grained distributed data access control in wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 22(4):673–686, 2011.