International Journal of Advance Research in Science and Engineering
Vol. No.4, Special Issue (01), September 2015
www.ijarse.com
IJARSE
ISSN 2319 - 8354

A LINEAR SCHEME FOR SECURE DATA AGGREGATION

Ankita Jain[1], Manoj Singh[2]

[1]Student, [2]Asso. Professor, Gurukul Institute of Engineering & Technology/CSE, Kota, (India)

ABSTRACT

Data aggregation is important in wireless sensor ne tworks to save energy. In an untrusted environment, it is required that this aggregation be performed in a secure manner. Cryptographic techniques to achieve this are time-consuming, hence violate the basic principles of energy saving. This paper proposes a mechanism based on homomorphisms to perform several aggregation functions like min, max, sum, average in a manner which is secure yet fast.

Keyword: Homomorphic Encryption, Secure Data Aggregation, Wireless Sensor Networks

I. INTRODUCTION

A wireless sensor network (WSN) consists of hundreds of sensor nodes deployed over a geographical area. Advances in wireless technology have resulted in increased use of WSNs in applications like as medical, military, monitoring disaster areas etc and relatively newer applications like habitat monitoring and target tracking [1]. The sensor nodes are usually devices with low computation capability and limited energy and memory resources. These limitations have a direct implication that communication between the nodes should be kept as low as possible. Given the enormous amount of data that a sensor node records and communicates, data aggregation techniques have to be used essentially. Data aggregation is defined as the process of aggregating the data from multiple sensors and calculates a smaller message that summarizes the important information from a group of sensors.

Though data aggregation is necessary to reduce communication overhead significantly, but at the same time it may lead to certain problems when the sensors are being deployed in an untrusted environment. Security requirements make encryption of data necessary, but this interferes with data aggregation. Secure data aggregation problem is defined in [2] as efficient delivery of the summary of sensor readings that are reported to an off-site user in such a way that ensures these reported readings have not been altered. They consider an aggregation application where the querier is located outside the WSN and the base station acts as an aggregator. One possible solution is that data remains encrypted only during transit, while each node may use plaintext data to perform aggregation operations. This leads to a related security issue - each intermediate node can modify, forge or discard messages, or simply transmit false aggregation values. The root of the problem is that keys need to be shared with the intermediate nodes and even a single compromised node is able to significantly alter the final aggregation value. Thus, conventional cryptographic approach fails to achieve secure data aggregation. Also, public key cryptography is very expensive for sensor nodes.

Recently, use of homomorphic cryptographic primitives to achieve security in data aggregation has gained mo-

mentum. Proposals like [3, 4, 5, 6] suggest that homomorphic encryption holds great promise in this area since it allows manipulation of encrypted data. This paper also discusses a symmetric key based homomorphism that can be used to perform aggregation of encrypted data.

II. HOMOMORPHIC ENCRYPTION IN SENSOR NETWORKS

Homomorphism has been much celebrated in literature for its use in secure aggregation of sensor data. Aggregation that employs homomorphism is proved to be more efficient in comparison to hop-by-hop encryption as demonstrated in [3] and [4]. Concealed Data Aggregation (CDA) [3] protocol uses an additive and multiplicative homomorphic encryption scheme suggested by Domingo-Ferrer [7] that allows the aggregator to aggregate encrypted data. Since security of the protocol depends on security of the underlying homomorphic scheme, CDA ensures only data confidentiality. The encryption in CDA is very expensive and adds between 0%-22% additional data overhead which increases the power consumption of the sending node.

Castellucia et al [4] proposed EDA, another scheme based on homomorphic encryption. This allows an aggregator to execute the aggregation function and aggregate the encrypted data that are received from its children with no need for decryption and to recover the original messages. It uses a modular addition instead of the xor (Exclusive-OR) operation that is found in the stream ciphers. Thus, even if an aggregator is being compromised, original messages cannot be revealed by an attacker.

A secure end to end encrypted data aggregation scheme [5] is based on elliptic curve cryptography that exploits a smaller key size. As well as it also allows the use of higher number of operation on cipher text and prevents the distinction between two identical texts from their cryptography. An approach [6] uses homomorphic encryption Elliptic Curve ElGamal algorithm to achieve data confidentiality while allowing in-network aggregation.

A similar approach is presented in [8] for heterogeneous sensor networks by employing simple modular arithmetic operations for homomorphism. However, this approach suffers from a drawback – the communication overhead grows with the increase in the size of aggregation tree. This problem was resolved through a hierarchical data aggregation model in [9]. A survey of secure data aggregation methods is presented in [10], which recommends privacy homomorphisms based on symmetric key cryptography for secure data aggregation in wireless sensor networks.

Since the breakthrough work of Gentry [11] in 2009, many fully homomorphic schemes have been proposed as an improvement on basic Gentry's blueprint. But all suffered from a major drawback – infeasibility due to very large keys. More recent and practical variants [12, 13] are based on sound security assumptions of learning with errors and belong to public key paradigm. There are very few homomorphic cryptographic primitives which are based on symmetric keys. Two recent symmetric homomorphic encryption methods are described in [14, 15]. Ensuring data privacy in clouds through a linear transformation fully homomorphic functions over square matrices of size 4 is basic idea in [14]. While [15] uses a variant of [16] in symmetric setting, which eliminates the requirement of bootstrappability in FHE (a common feature of FHE schemes derived on Gentry's blueprint).

III. PROPOSED SCHEME

3.1 Public and Private Key Generation

In this section we show how we can generate the public and private keys for encryption. Private key generation inputs a security parameter λ, another parameter *l*. N is a large prime number of bit length $\leq \sqrt{\lambda}$. Let S_1 and S_2

be two secret keys generated as follows. S_1 is a $l \times l$ size matrix where elements of matrix can be any random number of size √λ bits. S_2 is a $l \times 1$ size vector where elements of vector are randomly chosen numbers in Z_N. P is a public key calculated as $P = inv\ (S_1^2\text{-}1)$ where inv stands for multiplicative inverse of any number modulo N.

3.2 Encryption Function

The encryption function uses two secret keys $\mathbf{S_1}$and $\mathbf{S_2}$, the prime number *N* and takes a takes input the plaintext **m** which is a $l \times 1$ matrix. Encryption is performed as follows:

$$x_1 = (S_1 \times m - S_2) mod\ N$$

$$x_2 = (S_1 \times x_1 - m) mod\ N$$

$$x_3 = (S_1 \times x_2 - x_1) mod\ N$$

$$c = (x_2, x_3)$$

is ciphertext

Encryption algorithm is as follows:

> Encryption(S_1, S_2, m)
>
> Set $x_{-1} = S_2$ and x0 = m where m is the plaintext vector.
> For $0 \leq i <$ k, compute $x_{i+1} = S_1 x_i - x_{i-1}$
> Output Ciphertext $c = (x_{k-1}, x_k)$

3.3 Decryption Function

Once the message C arrived at the destination we uses the private key S_1 to decrypt it. Decryption is performed as follows:

Parse cipher text **c** into two cipher texts c1 and c2 which are matrices of size $\times 1$.

$$z_1 = (S_1 \times c_1 - c_2) Mod\ N$$

$$z_2 = (S_1 \times z_1 - c_1)\ Mod\ N$$

$$m = (z_2)$$

Decryption algorithm is as follows:

> **Decryption(S_1, c)**
>
> 1. For $k - 1 \geq i > 0$, compute
> $x_{i-1} = S1\, x_i - x_{i+1}$
> 2. Plaintext **p** = ($\mathbf{x}_{k-2}$)

As we mentioned our scheme supports homomorphic properties, which gives us the ability to execute operations on values even though they have been encrypted. It also allows additions, multiplication and comparison directly on ci-pher text, which prevents the decryption phase at the aggregator's level and saves nodes energy, which is crucial in sensor networks.

Some fundamental operations of Homomorphic are as follow:

3.3.1 Summation

Summation over cipher-texts are done as follows: let $E(m_1)$ and $E(m_2)$ are two cipher text of their plain text m_1 and m_2 respectively Then the sum of $E(m_1)$ and $E(m_2)$, let call it *c*, is represented by $c = E(m_1) + E(m_2)$ This sum operation guarantees that the decryption value of *C* is the sum $m_1 + m_2$.

This function takes encryption of two or more plaintext and outputs a cipher text pertaining to summation of plaintext values. Homomorphic summation function is as follows:

$$E(m_1 + m_2 + \dots + m_n) = E(m_1) + E(m_2) + \dots + E(m_n)$$

Example 1

Let's take an example with following values for matrix size 1×1.

$S_1 = 953, S_2 = 317, N = 1021$

Suppose there are two plaintext m1 and m2:

$m_1 = 312, \quad m_2 = 521$

Encryption of m_1 is as follows:

$x_1 = (S_1 \times m_1 - S_2) mod\ N$

$x_1 = (953 \times 312 - 317) \text{mod } 1021$

$x_1 = 929$

$x_2 = (S_1 \times x_1 - m_1) mod\ N$

$x_2 = (953 \times 929 - 312) \text{mod } 1021$

$x_2 = 839$

$x_3 = (S_1 \times x_2 - x_1)\ mod\ N$

$x_3 = (953 \times 839 - 929) \text{mod } 1021$

$x_3 = 216$

$E(m_1) = (839,216)$

Encryption of m_2 is as follows:

$y_1 = (S_1 \times m_2 - S_2) mod\ N$

$y_1 = (953 \times 521 - 317) \text{mod } 1021$

$y_1 = 1011$

$y_2 = (S_1 \times y_1 - m_2) \text{ mod N}$

$y_2 = (953 \times 1011 - 521) \text{mod } 1021$

$y_2 = 159$

$y_3 = (S_1 \times y_2 - y_1) \text{mod N}$

$y_3 = (953 \times 159 - 1011) \text{mod } 1021$

$y_3 = 429$

$E(m_2) = (159,429)$

Performing summation operation over encrypted values

$E(m1 + m2) = (998,645)$

$c = (998,645)$

Decryption is as follows:

$z_1 = (S_1 \times c_1 - c_2) mod\ N$

$z_1 = (953 \times 998 - 645) mod\ 1021$

$z_1 = 919$

$z_2 = (S_1 \times z_1 - c_1) mod N$

$z_2 = 953 \times 919 - 998$

$z_2 = 833$

Hence we get back $m_1 + m_2$.

3.3.2 Weighted Multiplication

Weighted Multiplication can be computed by taking one scalar *Sc* and encryption of a plaintext m which is E (m). Scalar Multiplication function is performed as follows:

$Scalar\ Multiplication = (Sc \times E(m))\ Mod\ N$

Example 2

Compute $E(2m_1)$

Here Sc is 2 and from the above example value of encrypted m_1 is

$E(m_1) = (839,216)$

$Scalar\ Multiplication = (1678,432)\ Mod\ 1021$

$Scalar\ Multiplication = (657,432)$

Decryption is as follows:

$z_1 = (S_1 \times c_1 - c_2)\ mod\ N$

$z_1 = (953 \times 657 - 432) \bmod 1021$

$z_1 = 837$

$z_2 = (S_1 \times z_1 - c_1) \bmod N$

$z_2 = (953 \times 837 - 657)\ \text{Mod}\ 1021$

$z_2 = 624$

Hence we get $2 \times m_2$

Comparison of two numbers over encrypted data can be done as follows: let q_1 is subtraction of two encrypted numbers and P is public key. To calculate minimum value between two number $Min = q_1 \times P \geq {}^{N}/_{2}$

Example 3:

From the above example

$E(m_1) = (839,216)$ and $E(m_2) = (159,429)$

Subtraction is: $E(m1 - m2) = (680,808)$

$(q_1, q_2) = (680,808)$

$Min = q_1 \times P \geq {}^{N}/_{2}$

Here q_1 = 680, ${}^{N}/_{2}$ =511,

p = inv (S_1^2*-1)*

p = 824

$Min = (680 \times 824)\ mod\ 1021 \geq 511$

Min =812>511

Hence we get back 312 is smaller than 521

We can extend the size of l by which we can perform these operations on multiple data. Below an example is example is presented for l=2 .

Example 4:

Let the size of l is 2

S1 is *l*l* size matrix and S2 is *l* size vector these are as follows:

$S_1 = \begin{vmatrix} 23 & 2 \\ 10 & 15 \end{vmatrix}$ $S_2 = \begin{vmatrix} 2 \\ 5 \end{vmatrix}$ $N = 11$

m_1 and m_2 are two plaintext vector of size *l*

$m_1 = \begin{vmatrix} 3 \\ 4 \end{vmatrix}$ $m_2 = \begin{vmatrix} 2 \\ 4 \end{vmatrix}$

Encryption of m_1 is as follows:

$X_1 = (S_1 \times m_1 - S_2) \, mod \, N$

$X_1 = \begin{vmatrix} 23 & 2 \\ 10 & 15 \end{vmatrix} \times \begin{vmatrix} 3 \\ 4 \end{vmatrix} - \begin{vmatrix} 2 \\ 5 \end{vmatrix} \bmod 11$

$X_1 = \begin{vmatrix} 1 \\ 0 \end{vmatrix} - \begin{vmatrix} 2 \\ 5 \end{vmatrix} \bmod 11$

$X_1 = \begin{vmatrix} 10 \\ 6 \end{vmatrix} \bmod 11$

$X_2 = (S_1 \times X_1 - m_1) \bmod N$

$X_2 = \begin{vmatrix} 4 \\ 10 \end{vmatrix}$

$X_3 = (S_1 \times X_2 - X_1) \, mod \, N$

$X_3 = \begin{vmatrix} 0 \\ 0 \end{vmatrix}$

$E(m_1) = \begin{vmatrix} 4 \\ 10 \end{vmatrix}, \begin{vmatrix} 0 \\ 0 \end{vmatrix}$

Encryption of m_2 is as follows:

$Y_1 = (S_1 \times m_2 - S_2) \bmod N$

$Y_1 = \begin{vmatrix} 0 \\ 0 \end{vmatrix} - \begin{vmatrix} 2 \\ 5 \end{vmatrix}$

$Y_1 = \begin{vmatrix} 0 \\ 0 \end{vmatrix}$

$Y_2 = (S_1 \times Y_1 - m_2) \, mod \, N$

$Y_2 = \begin{vmatrix} 10 \\ 2 \end{vmatrix}$

$Y_3 = S_1 \times Y_2 - Y_1$

$Y_3 = \begin{vmatrix} 0 \\ 5 \end{vmatrix}$

$E(m_2) = \begin{vmatrix} 10 \\ 2 \end{vmatrix}, \begin{vmatrix} 0 \\ 5 \end{vmatrix}$

Addition of $E(m_1)$ and $E(m_2)$ is as follow:

$E(m_1+m_2) = E(m_1) + E(m_2)$

$= \begin{vmatrix} 3 \\ 3 \end{vmatrix} \begin{vmatrix} 0 \\ 3 \end{vmatrix}$

Decryption of $E(m_1+m_2)$ is as follows

$z_1 = (S_1 \times c_1 - c_2) mod \, N$

$z_1 = \begin{vmatrix} 3 \\ 3 \end{vmatrix}$

$z_2 = (S_1 \times z_1 - c_1) mod \, 11$

$z_2 = \begin{vmatrix} 7 \\ 5 \end{vmatrix}$

Hence we get back m_1+m_2

3.3.3 Aggregation Functions

This additively Homomorphic scheme can be used to aggregate the encrypted data and can be used to compute average, min, max. When the user queries the network, instead of sending each sensor node's data to the base station, aggregators collect the information from its neighboring nodes, aggregates them (e.g., computes the average), and send the aggregated data to the base station.

Basic Functions of Aggregation:

Average: Average is a basic and very useful function for data aggregation it can be performed over encrypted data by using summation operation as explained above.

Minimum Function: Finding minimum values over encrypted data can be possible with this function. It can be calculated by using comparison function.

Maximum Function: Finding maximum values over encrypted data can be possible with this function.

IV. EXPERIMENTAL RESULTS

To show the effectiveness of our approach we have taken the results for the different values of λ and μ. Here λ and μ are the size (in bits) of Secret keys S_1 and S_2 respectively. We implement our algorithm as a Java program and evaluate its execution time in Millisecond. The computations were performed on JDK1.7.0 software, Windows 7 platform is used and processor is Intel Core 2 Duo 21 GHZ.

Tables I shows the time taken to generate Secret keys, Encryption and Decryption. We varied the keys sizes and obviously the security levels.

Tables II shows the time taken to Addition and Scalar Multiplication. Table III shows the time taken to perform Comparison function for the different value of Key size.

Runtime for number of plain text 5,10,50 are taken which is shown in the graph.

Table 1.Runtime of Keygen, Encryption and Decryption of The Propose Scheme, in Milliseconds, for Different Values of Security Parameter

S_no	μ	λ	Keygen	Encryption	Decryption
1	4	16	0.427	0.084	0.049
2	4	20	0.437	0.089	0.064
3	6	36	0.453	0.146	0.112
4	6	40	0.457	0.190	0.140
5	8	64	0.476	0.262	0.358
6	8	70	0.483	0.364	0.397
7	10	100	0.502	1.283	0.576
8	10	120	0.516	1.739	0.948
9	16	256	0.523	19.26	2.614
10	16	280	0.529	25.51	3.281
11	20	400	0.536	43.69	28.34
12	20	440	0.552	73.22	64.72

Table 2 Runtime of Addition and Scalar Multiplication of the Propose Scheme, in Milliseconds, for Different Values of Security Parameter

S_no	μ	λ	Addition	Scalar Multiplication
1	4	16	0.014	0.017
2	4	20	0.015	0.018
3	6	36	0.016	0.019
4	6	40	0.019	0.020
5	8	64	0.021	0.021
6	8	70	0.021	0.0219
7	10	100	0.022	0.022
8	10	120	0.023	0.025
9	16	256	0.024	0.026
10	16	280	0.025	0.027
11	20	400	0.025	0.028
12	20	440	0.026	0.029

Table 3 Runtime of Comparison of The Propose Scheme, in Milliseconds, for Different Values of Security Parameter

S_no	μ	λ	Comparison
1	4	16	0.048
2	4	20	0.050
3	6	36	0.060
4	6	40	0.063
5	8	64	0.066
6	8	70	0.082
7	10	100	0.087
8	10	120	0.099

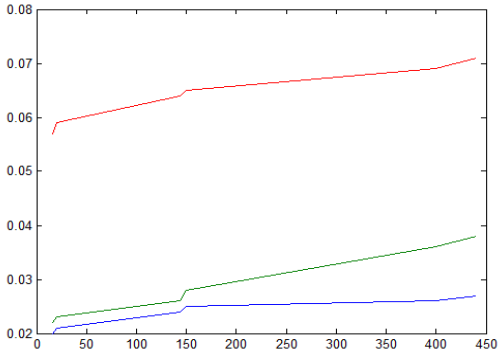

Fig 1 Runtime of Summation of the Propose Scheme for Number of Plain text 5, 10, 50 in Milli-seconds

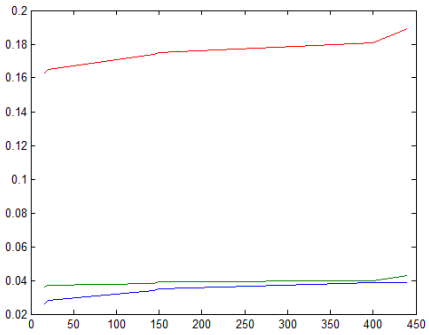

Fig 2 Runtime of Scalar Multiplication of the Propose Scheme for Number of Plain Text 5, 10, 50 in Milliseconds, for Different Values of Security Parameter

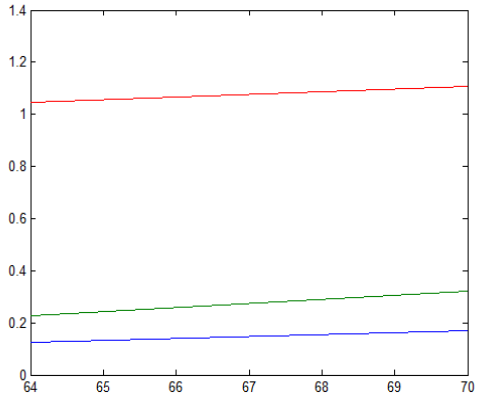

Fig 3 Runtime of Comparison of the Propose Scheme for Number of Plain text 5, 10, 50 in Milli-seconds, for Different Values of Security Parameter

V. CONCLUSION

Applications of Wireless sensor networks are increasing, and with the same pace is the need of securing them. Data aggregation is essential in WSNs and untrusted environments are unavoidable. So there is need of protocols that achieve secure data aggregation. Homomorphic schemes with symmetric keys are most promising solution to this problem. We proposed a homomorphic scheme that can perform many aggregation functions over

encrypted data. Runtime of the scheme is also practical for limited resources of sensor nodes.

REFERENCES

[1]. Alzaid, E Foo, J M G Neito and D G Park, "A taxonomy of secure data aggregation in wireless sensor networks", International Journal of Communication Networks and Distributed Systems Volume 8 Issue 1/2, December 2012 Pages 101-148

[2]. Przydatek, B., Song, D. X. & Perrig, A. (2003), SIA: Secure Information Aggregation in Sensor Networks in I. F. Akyildiz, D. Estrin, D. E. Culler & M. B. Srivastava, eds, 'SenSys', ACM, pp. 255–265.

[3]. Girao, J., Westhoff, D., and Schneider, M. (2005, May). CDA: Concealed data aggregation for reverse multicast traffic in wireless sensor networks. In Communications, 2005. ICC 2005. 2005 IEEE International Conference on (Vol. 5, pp. 3044-3049). IEEE.

[4]. Castelluccia, C., Mykletun, E., and Tsudik, G. (2005)Efficient Aggregation of encrypted data in Wireless Sensor Networks. Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous'05)

[5]. Jacques M. Bahi, Christophe Guyeux and Abdallah Makhoul "Efficient and Robust Secure Aggregation of Encrypted Data in Sensor Networks"

[6]. Jacques M. Bahi, Christophe Guyeux and Abdallah Makhoul "Secure Data Aggregation in Wireless Sensor Networks"

[7]. J.I. Domingo-Ferrer, "A new privacy homomorphism and applications," Information Processing Letters 60.5 (1996): pp 277-282.

[8]. Ozdemir, S. (2007) Concealed data aggregation in heterogeneous sensor networks using privacy homomorphism, in: Proceedings of the ICPS'07: IEEE International Conference on Pervasive Services, Istanbul, Turkey, 2007, pp. 165–168.

[9]. Ozdemir, S. (2008) Secure data aggregation in wireless sensor networks via homomorphic encryption. Journal of The Faculty of Engineering and Architecture of Gazi University 23 (2), pp 365-373.

[10]. Ozdemir, S. and Xiao, Y. (2009) Secure data aggregation in wireless sensor networks: A comprehensive overview. Computer Networks 53 (2009) 2022–2037 Elsevier. doi:10.1016/j.comnet.2009.02.023

[11]. C. Gentry, "Fully homomorphic encryption scheme," Diss. Stanford University, 2009.

[12]. Z. Brakerski, C. Gentry, and V. Vaikuntanathan. "Fully homomorphic encryption without bootstrapping", Cryptology ePrint Archive, Report 2011/277.

[13]. C Gentry, A Sahaiy and B Waters. "Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based". 2013/340

[14]. C.P. Gupta and I. Sharma, "A fully homomorphic encryption scheme with symmetric keys with application to private data processing in clouds," Network of the Future (NOF), 2013 Fourth International Conference on the , vol., no., pp.1,4, 23-25 Oct. 2013 doi: 10.1109/NOF.2013.6724526

[15]. N Aggarwal, C Gupta and I Sharma, "Fully Homomorphic Symmetric Scheme without Bootstrapping", International Conference on Cloud Computing and Internet of Things (CCIOT), 2014, Chengdu, China.

[16]. Marten van Dijk and Craig Gentry and Shai Halevi and Vinod Vaikuntanathan. Fully Homomorphic Encryption over the Integers.https://eprint.iacr.org/2009/616