Proceedings of the 4th Annual ISC Research Symposium
ISCRS 2010
April 21, 2010, Rolla, Missouri

Performance Analysis of Secure Hierarchical Data Aggregation in Wireless Sensor Networks

Vimal Kumar
Dept of Computer Science
Missouri University of Science & Technology

Sanjay Madria
Dept of Computer Science
Missouri University of Science & Technology

ABSTRACT

In this paper we study the performance of an end to end secure data aggregation scheme on wireless sensors. We measure the execution time and energy consumption of various cryptographic functions on the motes and analyze how an end to end scheme increases the network life time in a WSN compared to a hop by hop scheme. The scheme is implemented on Mica2 motes and makes use of elliptic curves for the implementation of public key cryptography on the motes.

1. INTRODUCTION

Wireless sensor networks have a number of applications in various fields like military, medicine, habitat monitoring, target tracking etc. The small size of wireless sensors allows us to easily deploy them in hostile environment in large numbers without them being noticed but, it also presents us with some constraints. The small size of a wireless sensor limits the amount of memory, the available processing power and the size of battery on the sensors. Limited memory means that the amount of data and the code that can be stored on a sensor is limited. Sensors are usually deployed in hostile environments, where it is not feasible to change the batteries once they expire. This calls for judicious use of the batteries so that the sensor lifetime and hence the network life time can be maximized. One way of increasing network lifetime of a WSN is data aggregation.

The challenge posed by limited available power becomes harder when applications require security due to the high cost of security techniques. Two different types of secure data aggregation schemes have been proposed by researchers, *hop by hop* secure data aggregation schemes and *end to end* secure data aggregation schemes.The paradigm shift now is towards *end to end* schemes which offer better security than the *hop by hop* ones. *End to end* schemes also require lesser amount of computation which helps in reducing energy consumption. Secure hierarchical data aggregation in wireless sensor networks [1] is one such *end to end* scheme. In this scheme the sensor nodes first organize themselves into a tree hierarchy and then use a homomorphic encryption algorithm (ECEG) and an aggregate digital signature algorithm (ECDSA) to achieve *end to end* cryptography. We have designed and implemented the algorithm proposed in [1] with some modifications on a Mica2 mote. We discuss some of the results we obtained and evaluate the algorithm based on these results. The *end to end* scheme of [1] provides better security by virtue of public key cryptography. We analyze our results to show how it saves energy on the aggregators and increases the network lifetime by 57%. Although our focus is on a particular scheme, our results hold for any *end to end* secure data aggregation scheme in general.

In the following section we survey the literature on secure data aggregation schemes and compare this work with others. We elaborate on data aggregation and security in wireless sensor networks in section 3. In section 4 we revise the secure hierarchical data aggregation algorithm as introduced in [1] while section 5 discusses the sub algorithms within the algorithm. Section 6 details the implementation of the scheme on Mica2 motes. In section 7 we provide some initial results and the analysis from our implementation, in section 8 we provide our conclusion and discuss our future work and expected results in section 9.

2. RELATED WORK

Data aggregation in wireless sensor networks has been of interest to researchers because of its ability to save energy on the sensors. Early secure data aggregation schemes were *hop by hop* schemes, these schemes like the one by Hu and Evans [8] mostly dealt with the issue of data confidentiality in the face of a single compromised node. Schemes tackling the issue of multiple compromised nodes were introduced later, for example the scheme by Chan et al [9]. This algorithm supported any arbitrary tree and was resilient to any number of malicious nodes. Schemes like SecureDAV [10] and SDAP [11] also provided for data integrity by making use of threshold cryptography and Merkle hash trees respectively. Next were the *end to end* schemes some of which are discussed in [1], [12] and [13]. These algorithms use the concept of homomorphic encryption. While [13] does not provide data integrity, both [1] and [12] use the aggregate signature protocols for it. The former uses a form of ECDSA and the latter uses Boneh and Gentry's aggregate signature scheme. We focus on the secure hierarchical data aggregation paper of [1]. This is an end to end scheme providing both data confidentiality and data integrity. Our objective is to compare the performance of this scheme with others. The work in [5] and [6] deals with evaluating the performance of public key encryption and homomorphic encryption on wireless sensors. The authors in [14] calculate the cost of cryptography on wireless sensors. They take into consideration an energy model and based on that model calculate the cost of key distribution, encryption and communicating secure data. In our work, we evaluate the performance of the homomorphic encryption as well as analyze the performance of a secure data aggregation scheme and calculate the energy savings due to it. Although we take into consideration a particular homomorphic encryption scheme and

a particular signature scheme, our results are valid for any *end to end* secure data aggregation scheme in general.

3. BACKGROUND

3.1. Data Aggregation

As concurred in the previous section, the biggest challenge while working with wireless sensors is the limited available battery power. As pointed out in [6] and [7] radio communication consumes a large amount of energy on a sensor. Owing to this, one of the goals in sensor network research is to minimize the number of radio transmissions within the network. Data aggregation is one way of doing this. An aggregate function like SUM, AVERAGE etc. takes as input a number of values and outputs a single aggregate value. In applications requiring data aggregation, the aggregator receives input values from various sensors, performs the required operation, and sends forward only the output, thus saving transmissions. Consider the example sensor network in Fig. 1. In case of no data aggregation each of the leaf node sensors generates a reading and sends it to its parent. At this level 9 messages are generated. At the next level, the sensors forward each message they receive from their children up the hierarchy as well as their own readings. At this level in the hierarchy, a total of 9 + 3 = 12 messages are transmitted. Similarly at the next level, 12+1 = 13 messages are transmitted. In the network, a total of 9+12+13= 34 messages are sent. If data aggregation is employed then each of the 9 sensors sends their readings to their parents as above but the readings and the parent's own reading are aggregated and merged into a single entity so only 3 messages are communicated by the 3 parents. At the next level, these three are again aggregated into 1 combined reading which is sent to the base station. Thus a total of only 13 messages are communicated in the network compared to the 34 earlier when data aggregation was not used. The saving in transmissions is substantial when we consider a large sensor network with thousands of nodes.

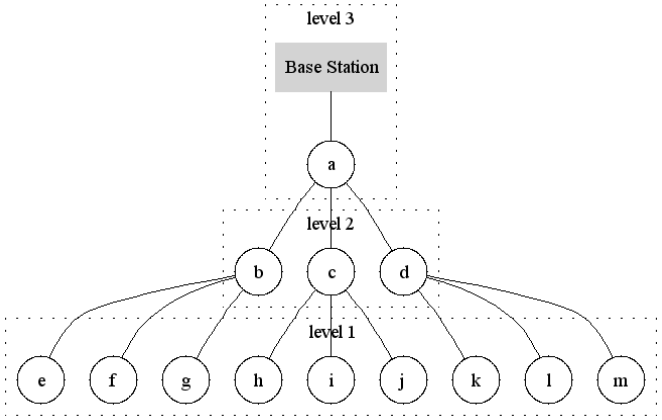

Figure 1. Data Aggregation in a Wireless Sensor Network.

3.2. Security in Wireless Sensor Networks

Two important security primitives in wireless sensor networks are data confidentiality and data integrity. Confidentiality means preventing any unauthorized entity from listening to the network traffic while data integrity means making sure that the data received by the receiver has not been tampered on its way. These two security primitives are the ones which are addressed in this paper. Security schemes in wireless sensor networks can be of two types. *Hop by hop* schemes and *end to end* schemes. In a *hop by hop* scheme communication between each hop is made secure. A sensor senses its environment, encrypts the data with a key shared between the two sensors and sends the encrypted data to the receiver. The receiver decrypts the received data and encrypts it again with a key shared between it and its next neighbor and sends it to the neighbor. This process continues till the data reaches the base station. In the hop by hop scheme, the data is decrypted and encrypted again at each intermediate node in the network. This introduces a potential security risk as the data is exposed after decrypting. If the node is compromised the attacker can easily get hold of the data. Also decryption and encryption at each node requires power. In *end to end* schemes on the other hand, encryption and decryption only takes place once in the system. The data is encrypted at the node where it originated and decrypted at the base station, thus removing the possibility of attack at the intermediate nodes. End to end scheme also helps save energy by not making every intermediate node perform the decryption and encryption operations for each datum they receive. *End to end* schemes are more secure than the hop *by hop* ones however, it limits our ability to perform aggregation in the network. For an intermediate node to be able to perform aggregation, it needs to have unencrypted data. *End to end* scheme prevents the intermediates nodes from decrypting the data. This calls for methods which enable us to work on encrypted data.

Homomorphic encryption

One of the ways of working on encrypted data is through the use of homomorphic encryption. An encryption algorithm is said to be homomorphic, if it allows for the following property to hold.

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

The two data items a and b are encrypted and the operation $\otimes$ is applied on the encrypted data. If the encryption scheme is homomorphic than its result would be the same when the operation $\otimes$ is performed on a and b first and the result is encrypted. Homomorphisms can be of two types, additive homomorphism and multiplicative homomorphism.

Aggregate digital signature

Homomorphic encryption provides for data confidentiality for integrity though we need digital signatures. An aggregate digital signature algorithm provides the functionality to aggregate n signatures on n distinct messages by n distinct users, into a single signature. This single signature will convince the verifier that the n users signed the n original messages. Once the signature is verified the verifier can be sure that the integrity of the data is intact. The assumption being that the sender's signing key has not been compromised.

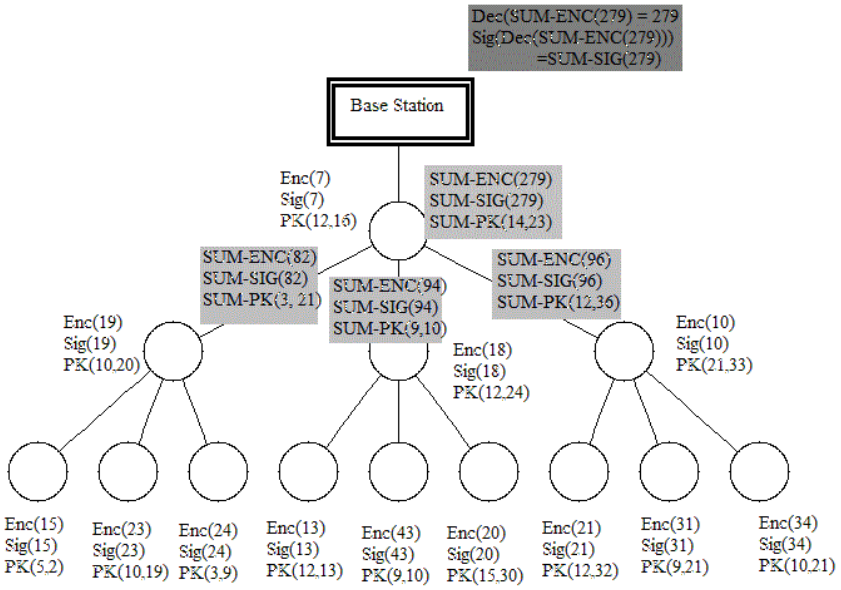

Figure 2.Sensor network using homomorphic encryption and agg. digital signatures according to the secure hierarchical data aggregation algorithm.

Elliptic curve cryptography

Public key cryptography is widely used in traditional systems. Unfortunately traditional security approaches for confidentiality and data integrity are not feasible on wireless sensors. Public key cryptography although very secure can exhaust a sensor's power very quickly. Elliptic curve cryptography (ECC) has come up as an attractive and viable alternative to public key cryptography in resource constrained environments. In ECC for a given level of security a smaller key can be used and a smaller key translates to lesser computations and hence less power consumption. Elliptic curve cryptography makes use of the points on an elliptic curve defined by $y^2 = x^3 + ax + b$ over a finite field $\mathcal{F}$. Elliptic curve cryptography makes use of the elliptic curve analog of the discrete log problem known as the elliptic curve discrete log problem (ECDLP). The ECDLP is computationally harder than the original discrete log problem and elliptic curve cryptosystem is built around it.

4. SECURE HIERARCHICAL DATA AGGREGATION ALGORITHM

The secure hierarchical data aggregation algorithm is an *end to end* scheme which employs homomorphic encryption for confidentiality and digital signatures for signing the data to ensure data integrity. The original algorithm [1] specifies the use of elliptic curve integrate encryption scheme (ECIES) for encryption and a modified version of elliptic curve digital signature algorithm (ECDSA) for signing. Also the algorithm does not specify any particular tree construction algorithm. We found out that ECIES is not suitable for homomorphic encryption and hence we replaced it with elliptic curve elgamal (ECEG) in the implementation. We also implemented a tree construction algorithm and provide the algorithm in the algorithms section. The secure data aggregation algorithm assumes that the sensors are organized in a tree hierarchy, with the base station at the root. When the network boots up each sensor node generates a reading *x*. The reading is signed using the aggregate signature algorithm and *Sig(x)* is generated. The reading is then encrypted using the homomorphic encryption algorithm and *Enc(x)*. The leaf nodes then send the encrypted data, signature and the public key corresponding to the private key used for generating signature to their parent. After a node has received data from all its children, it sums up all the encrypted readings, which is possible because homomorphic encryption was used. It sums up the signature using the aggregate signature algorithm and all the public keys. The SUM-ENC, SUM-SIG and SUM-PK are then sent to the node's parent. This process is repeated at every node until the data reaches the base station. An example network is shown in Fig. 2, where the data flows upwards towards the base station from the leaf nodes.

5. ALGORITHMS

In this section we provide three algorithms; *tree construction algorithm, the HAgg algorithm running on the sensors* and *the HAgg algorithm running on the base station.*

The tree construction algorithm

We assume the TinyOS lossy radio model, considering all the sensors are alike and spread in a grid topology with one sensor placed in each grid. The channel is not ideal and has a definite error rate. Moreover in a bidirectional channel the error rate in both the directions is different. The lossy radio model assumes each mote has a transmission range of 50 feet and the probability of error in transmission increasing with the distance from the sensor.

Requires: Parameters MAX_ALLOWED_REQUESTS, MAX_CHILDREN to be set before deployment.

1: The base station starts by broadcasting a HELLO message.
2: If a sensor which has not yet elected its parent receives a HELLO message, it sends a PARENT REQUEST to the originator of the HELLO message.
3: When a node receives a PARENT REQUEST, it makes the following two checks
    i: The number of children is less than the MAX_CHILDREN limit.
    ii: The number of requests from a particular node is less than the MAX_ALLOWED_REQUESTS limit.
4: When the above two checks are satisfied the node sends an ACCEPTED message to the sender of the PARENT REQUEST and adds the node to its children list.
5: Upon receiving an ACCEPTED message a node elects the sender of the ACCEPTED message as its parent and broadcasts a HELLO message.
6: If a sensor has not been able to elect a parent after a certain period of time it broadcasts a HELP message.
7: Any sensor which receives a HELP message makes the two checks defined in step 3. When the checks are satisfied, it sends a HELP RESPONSE message.
8: The originator of the HELP message accepts the sender of the first HELP RESPONSE as its parent and sends a HELP ACK.
9: On receiving the HELP ACK from a node, the receiver accepts the sender of HELP ACK as its child.

Algorithm 1. The tree construction algorithm.

HAgg algorithm running on the sensor nodes.

The HAgg algorithm assumes the sensors are preloaded with the appropriate elliptic curve parameters, the base station's public key and a network wide random integer. The random integer is used to compute a new *k* for each round. At the start of each round each sensor chooses its private key and an appropriate public key. An elliptic curve private key is just a point on the elliptic curve and the public key is another point obtained by multiplying the base point by the private key. Each sensor computes *R*, and the multiplicative inverse of *k mod p*. The sensor then generates the signature s_i . This is followed by homomorphically encrypting the message m_i. The message is first mapped to a point on the elliptic curve and then encrypted by the ECEG algorithm using the base station's public key, although the original HAgg algorithm specifies the use of ECIES for encryption.

HAgg algorithm running at the base station.

The base station receives the sum of the signatures, the sum of the corresponding public keys and the sum of the encrypted messages. The base station decrypts the aggregate result using its private key, and then reverse maps the elliptic curve point to plaintext information. For verification of the signature the base station calculates a point on the elliptic curve using the received signature, decrypted message and *k*. If the x coordinate of the calculate point is same as *r(x)* the signature is verifies. For security analysis and proof of the algorithm refer to [1].

The additive digital signature algorithm is a modification of the ECDSA algorithm, based on the observation that in a wireless sensor network environment all our messages are going to be of the same size. The original ECDSA algorithm calculates the hash of a message before encrypting it with the private key. Since all the messages in a sensor environment are of the same size, there is no need of a hashing algorithm. In the modified ECDSA, the authors in [1] do away with the hashing and the resulting signature algorithm is additive in nature.

In the ECDSA algorithm a signature is a tuple (*r, s*) such that $r = (r(x) \bmod p)$, where $(r(x), r(y)) = k\mathrm{T}$, *k* is a randomly chosen number and T is the base point. S is found out using the formula $s = k^{-1} (h(m) + z * r(x)) \bmod p$. Here *h* is a secure hash function and *z* is the private key of the node. When two signatures $d_1=(r_1,s_1)$ and $d_2=(r_2,s_2)$ on two messages m_1 and m_2 are added r_1 and r_2 remain the same while s_1 and s_2 can

Requires: Elliptic curve parameters D=(q,FR,a,b,T,p,h), sensor reading m_i , private key z_i , base station's public key Q, a network wide random integer *k*.

1: Each sensor computes $z_i * T = (x; y)$, its public key.
2: Each sensor computes $R = (r(x), r(y)) = k * T$.
3: Each sensor computes $k^{-1} \bmod p$.
4: Each sensor computes $s_i = k^{-1} (m_i + z_i * r(x)) \bmod p$.
5: Each sensors signature for the message m_i is s_i.
6: Each sensor maps its reading m_i onto the elliptic curve D.
7: Each sensor generates ciphertext $\boldsymbol{m_i} = \mathrm{enc}(m_i)$
8: **if** Sensor is a parent **then**
9: The sensor combines the signatures into $s = \sum s_i$
10: The sensor combines the all ciphertexts into one ciphertext $\sum \boldsymbol{m_i}$
11: **end if**

Algorithm 2.The HAgg algorithm at the sensor node [1].

Requires: Elliptic Curve Parameters D = (q, FR, a, b, T, p, h), sum of encrypted sensor readings $m = \sum \boldsymbol{m_i}$, sum of the signatures $s = \sum s_i$, base station private key *qi*, sum of public keys *Z*, a network wide random integer k

1: Decrypt ciphertext $\sum \boldsymbol{m_i} = \sum mi$
2: Map reading *m* from the elliptic curve D into plaintext.
3: Compute $R = (r(x), r(y)) = k * T$.
4: Compute $w = s^{-1} \bmod p$.
5: Compute $u1 = mw \bmod p$.
6: Compute $u2 = r(x)w \bmod p$.
7: Compute $X = u1T + u2Z$.
8: Compute $v = X(x) \bmod p$.
9: **if** $v == r$ **then**
10: The signature verified
11: **end if**

Algorithm 3.The HAgg algorithm at the base station [1].

be written as $s_1 = k^{-1} (h(m_1) + z * r(x))$ and $s_2 = k^{-1} (h(m_2) + z * r(x))$. ECDSA is not an aggregate signature scheme because when these two signatures are added $h(m_1)$and $h(m_2)$ need to be added. Hashing is not homomorphic so $h(m_1) + h(m_2) \neq h(m_1 + m_2)$ hence an aggregate signature is not the same as the signature on the sum of messages. On the other hand if we replace the hash of the message by the message itself in the formula the signature becomes additive because in that case we are just summing up integers. The signature (r, s) in the modified signature scheme is $r = (r(x) \bmod p)$ and $s = k^{-1} (m + z * r(x)) \bmod p$.

The EC Elgamal encryption.

The secure hierarchical data aggregation paper discusses the digital signature algorithm but not the encryption scheme. In this subsection we discuss the encryption scheme. The encryption scheme we use is the elliptic curve elgamal encryption which is an additive homomorphic encryption scheme. Before we can encrypt a message we first need to map the plaintext data to a point on the elliptic curve. The mapping should be such that it supports homomorphic property. The encrypted data is another point on the elliptic curve. To get the plaintext back, this point is first decrypted and a reverse mapping function is used to convert the elliptic curve point to the plaintext. We use a simple mapping technique in which we multiply the plaintext message m by the base point T, to get the elliptic curve point mT. This mapping satisfies the homomorphic criteria as shown below.

$$\text{map}(m_1) + \text{map}(m_2) + \ldots\ldots + \text{map}(m_n) = m_1\text{T} + m_2\text{T} + \ldots\ldots m_n\text{T}$$

This translates the algorithm into a homomorphic encryption algorithm as follows. Each message m_i maps to a point M_i on the elliptic curve. The points on the elliptic curve M_i s are added, and the addition of the elliptic curve points is equivalent to the addition of the plaintext data. The plaintext can be found out by reverse mapping the final result.

$$\begin{aligned} \text{M}_1 + \text{M}_2 + \ldots\ldots + \text{M}_n &= \text{map}(m_1) + \text{map}(m_2) + \ldots\ldots + \text{map}(m_n) \\ &= m_1\text{T} + m_2\text{T} + \ldots\ldots m_n\text{T} \\ &= (m_1 + m_2 + \ldots\ldots + m_n)\text{T} \\ &= (\textstyle\sum m_i)\text{T} \end{aligned}$$

Requires: Elliptic curve parameters D=(q,FR,a,b,T,p,h), sensor reading m_i and the private key z_i.
Encryption
1: Map the message m to an elliptic curve point M using a mapping technique.
2: Generate a random integer k.
3: Calculate $C_1 = kT$ and $C_2 = M + kQ$.
4: C (C_1, C_2)= (kT, M+kQ) is the ciphertext.
Decryption
1: Calculate ($- z_i C_1$) and add it to C_2.
2: The decrypted message M is the addition ($- z_i C_1$)+C_2

Algorithm 4. EC elgamal encryption and decryption algorithm [5].

6. IMPLEMENTATION

We chose the mica2 sensor mote for our implementation. The coding is done on the TinyOS/TOSSIM platform for the mica2 mote. TinyOS is an open source OS for wireless networked sensors [1], requiring minimal hardware. The programming language for TinyOS is networked embedded systems C (nesC), which is a derivative of C. We make use of the TinyECC library [3]. TinyECC is implemented over the prime field $\mathcal{F}_p$ where p is a large prime number. The library consists of routines for large natural number operation, ECC operations and a key distribution algorithm ECDH, an encryption algorithm ECIES and a digital signature algorithm ECDSA. More information on TinyECC library can be found in [3]. In our implementation the *secp160r1* 160 bit elliptic curve was used which provides security equivalent to 1024 bit RSA key. We first simulated the code on TOSSIM, using AVRORA [4] as the simulation environment. AVRORA is a set of simulation and analysis tools for programs written for the AVR microcontroller which is used by the mica2 mote. The implementation was done on the mica2 mote. We ported the code on the motes and measured the execution times of the various procedures which are tabulated in Table 1.

Operation	**Time Taken**	**Energy Consumed**
Encryption	117905 ms	2829.7 mJ
Decryption	79099.5 ms	1898.4 mJ
Sign	38884 ms	933.21 mJ
Verify	75075.9 ms	1801.8 mJ
Addition of ciphertext	317.3 ms	7.61 mJ
Addition of signatures	.183 ms	4.392 µJ
Addition of Public keys	160.5 ms	3.85 mJ

Table 1. Execution time of various functions on motes.

7. ANALYSIS

Consider the example network of Fig. 2. Each sensor in the network signs and encrypts its data before sending it to the aggregator. In case of a hop by hop algorithm the aggregator will first decrypt and verify all the messages it receives, after that it will add them together and finally encrypt and sign the aggregate before sending it further. In contrast to this, the aggregator in our algorithm only needs to add the ciphertext, the digital signatures and the public keys. So we are replacing the decryption and verify operations with three additions. If we look up Table 1 we find that the addition operations are significantly faster and consume less power than decryption and verify. As we said earlier we perform the decryption and verification only at the root node so we do not consider them here. The decryption and encryption consume 3700 mJ of energy while the three additions cost a mere 11.464 mJ. Replacing the signing and encryption with a few additions saves us 3688.5 mJ at each intermediate node (aggregator). Considering that two AA batteries have 18720000 mJ of energy, the saving of 3688.5 mJ in each round of communication is

significant. Our implementation of the operations can be further optimized which is one of our future goals but the important thing to note here is the saving in energy due to the scheme. The saving in energy can be maximized when we have optimized implementation of the above operations. In a data aggregation scheme aggregator holds an important position and is required to perform more tasks than an ordinary sensor. This leads to the aggregator getting exhausted quicker than the rest of the sensors. Our scheme reduces the burden on the aggregator and increases its life.

Energy spent by the aggregator in a hop by hop scheme (decryption + verification + signing + encryption) = 7463.11 mJ
Number of rounds before exhaustion = 18720000/7463.11 = 2508
Energy spent by the aggregator in the scheme under investigation (signing + encryption) = 4739.5mJ
Number of rounds before exhaustion = 18720000/4739.5 =3949

This means that in the secure hierarchical data aggregation scheme an aggregator lasts 57% longer compared to a hop by hop scheme. In a tree based schemes aggregators are entrusted with more tasks than any other nodes which means they are the first ones to go down. Thus, increasing the aggregator life time by an amount will mean the network life time is also increased by the same amount.

8. CONCLUSIONS

As seen in the previous section, the secure hierarchical data aggregation scheme offers a definite advantage over hop by hop data aggregation schemes. It increases the aggregator life time, the network life time as well as provides greater security by not letting the aggregator decrypt the data. We are in the process of performing a detailed analysis of the scheme which will reinforce our preliminary analysis which clearly shows the advantages of this scheme.

9. FUTURE WORK

We are working on optimizing our functions which will further reduce the execution time and energy consumed. We are also working on enabling the optimizing switches of TinyECC [3] in our code, which will greatly reduce the statistics further. As a further extension we are also looking to minimize the redundancy of the messages in the network by implementing some form of data compression which will help it further reducing the number of transmissions in the network and thus will increase the network lifetime. As concluded above the algorithm saves energy compared to a hop by hop encryption scheme, we are performing more detailed analysis measuring the performance of the algorithm further on the parameters such as the overhead compared to a no security scheme, the energy saved and the throughput.

10. ACKNOWLEDGMENTS

This research is partly supported by Intelligent Systems Center.

11. REFERENCES

[1]Julia Albath and Sanjay Madria, “Secure Hierarchical Aggregation in Sensor Networks,”. In *Proceedings of IEEE Wireless Communications and Networking Conference*, 2009.

[2] J. P. Walters, Z. Liang, W. Shi, and V. Chaudhary, “Wireless Sensor Network Security: A Survey,”. *Auerbach Publications*, CRC Press,2006.

[3] An Liu and Peng Ning, “ Tinyecc: A configurable library for elliptic curve cryptography in wireless sensor networks”. In *7th International Conference on Information Processing in Sensor Networks (IPSN 2008)*, April 2008.

[4] Ben Titzer, Daniel K. Lee and Jens Palsberg, “Avrora: scalable sensor network simulation with precise timing”. In *Proceedings of the Fourth International Symposium on Information Processing in Sensor Networks, IPSN* 2005.

[5]O. Ugus, A. Hessler, and D. Westhof, “Performance of additive homomorphic EC-Elgamal encryption for TinyPEDS”. Technical report,6. *Fachgesprach "Drahtlose Sensornetze*", July 2007.

[6] K. Piotrowski, P. Langendoerfer, and S. Peter. “How public key cryptography influences wireless sensor node lifetime”. In *Proceedings of the 4rd ACM Workshop on Security of ad hoc and Sensor Networks*, SASN 2006.

[7] S. Peter, K. Piotrowski, and P. Langendoerfer. “On concealed data aggregation for wireless sensor networks”. *In Proceedings of the IEEE Consumer Communications and Networking Conference* 2007.

[8] Lingxuan Hu and David Evans, “Secure Aggregation for Wireless Networks”. In *Workshop on Security and Assurance in Ad hoc Networks*, 2003.

[9] Haowen Chan, Adrian Perrig and Dawn Song “A Secure Hierarchical In-network Aggregation in Sensor Networks”. In *CCS 2006.*

[10] Ajay Mahimkar and Theodore S Rappaport, “SecureDAV: A secure data aggregation and verification protocol for sensor networks” In *Proceedings of the IEEE Global Telecommunications Conference*, 2004.

[11] Yi Yang, Xinran Wang, Sencun Zhu and Guohong Cao, “SDAP: a secure hop-by-hop data aggregation protocol for sensor networks”. In *MobiHoc ’06: Proceedings of the seventh ACM international symposium on Mobile ad hoc networking and computing*

[12] Hung-Min Sun, Ying-Chu Hsiao, Yue-Hsun Lin, Chien-Ming Chen, “An Efficient and Verifiable concealed Data Aggregation Scheme in Wireless Sensor Networks”. In *Proceedings of the 2008 International Conference on Embedded Software and Systems* ,pp. 19-26

[13] C. Castelluccia, E. Mykletun and G. Tsudik. “Efficient Aggregation of Encrypted Data in Wireless Sensor Networks”. In *MobiQuitous* 2005.

[14] G. de Meulenae, F. Gosset, F. Standaert and O. Pereira. “On the Energy Cost of Communication and Cryptography in Wireless Sensor Networks”. *In IEEE International Conference on Wireless and Mobile Computing 2008.*