

SEA: Secure Encrypted-Data Aggregation in Mobile Wireless Sensor Networks

Shih-I Huang^{1,2} Shiuhpyng Shieh²

Industrial Technology Research Institute¹

Dept. of Comp. Sci. and Info. Eng., National Chiao Tung University²

E-mail:sihuang@itri.org.tw, {sihuang,ssp}@csie.nctu.edu.tw

1 Abstract

This paper proposes a Secure Encrypted-data Aggregation (SEA) scheme in mobile wireless sensor networks (MWSN) environment. Our design for data aggregation eliminates redundant sensor readings without using encryption and maintains data secrecy and privacy during transmission. In contrast to conventional schemes, our proposed scheme provides security and privacy, and duplicate instances of original readings will be aggregated into a single packet; therefore, more energy can be saved.

2 Introduction

Wireless Sensor Networks (WSN) have emerged as an important new area in wireless technology. A wireless sensor network [2] is a distributed system interacting with physical environment. It consists of *motes* equipped with task-specific sensors to measure the surrounding environment, e.g. temperature, movement, etc. It provides solutions to many challenging problems such as wildlife, battlefield, wildfire, or building safety monitoring. A key component in a WSN is the *mote*, which contains a) a simple microprocessor, b) application-specific sensors, and c) a wireless transceiver. Each sensor mote is typically powered by batteries, making energy consumption an issue.

A major application for a wireless mote is to measure environmental values using embedded sensors, and transmit sensed data to a remote repository or a remote server. Because of limited transmission capabilities, this often requires multi-hop forwarding of messages, and is power consuming.

One specific power-saving mechanism used in wireless sensor networks is *data aggregation* [1][3][7][10]. Our paper proposes a novel method for eliminating duplicate encrypted data during aggregation without decryption. Data aggregation has been put forward as an essential paradigm in sensor networks. The aggregator uses specific functions, such as addition, subtraction or exclusive-or, to aggregate incoming readings, and only aggregated result are forwarded. Therefore, communication overhead can be reduced by decreasing the number of transmitted packets. Without encryption, adversaries can monitor and inject false data into the network. Encryption can solve this problem, but how can we aggregate over encrypted data [6]?

Previous work in data aggregation assumes that every mote is honest and only transmits their correct readings. Intanagonwiwat, Govindan, Estrin, and Heidemann proposed a data-centric diffusion method to aggregate data [9]. Their method enables diffusion to achieve energy savings by selecting empirically good paths and by caching and processing data in-network. Though their method can achieve significantly energy saving, security is not put into consideration in their design. Hu and Evans further examined the problem that a single compromised sensor mote can render the networks useless, or worse, mislead the operator into trusting a false reading [8]. They proposed an aggregation protocol that is resilient to both intruder devices and single device key compromises, but their scheme suffers a problem that the aggregated data will be expanded every time when it was aggregated and forwarded by any intermediate sensor mote.

This paper proposes a new method for determining and eliminating duplicate data while protecting privacy (using encryption) without excessive key-management or power management issues. The rest of the paper is organized as follows: Section II provides background on related work. In Section III, we describe our system architecture and proposed aggregation protocol. Section IV offers conclusions.

* This work was supported in part by National Science Council and Taiwan Information Security Center (TWISC) under the grants NSC 95-2218-E-001-001, and NSC95-2218-E-011-015.; by iCAST under NSC96-3114-P-001-002-Y; by Industrial Technology Research Institute(ITRI), Chung Shan Institute of Science and Technology, and Telecom. Technology Center.

3 SEA: Secure Encrypted-DATA

Aggregation

Our proposed scheme comprises five phases: *group forming*, *aggregator election*, *initialization*, *data encryption*, and *data aggregation*. In group forming phase, a MWSN is automatically divided into non-overlapped tree-based *aggregation groups*, and each node joins only one group. In aggregation election phase, a sensor will be elected as the *group aggregator*. Each mote in the same group transmits its readings to the aggregator. In initialization phase, each node secure exchange its pre-stored keys with the aggregator and the aggregator uses these keys to perform secure data aggregation in last phase to eliminate redundant readings without decrypting received packets.

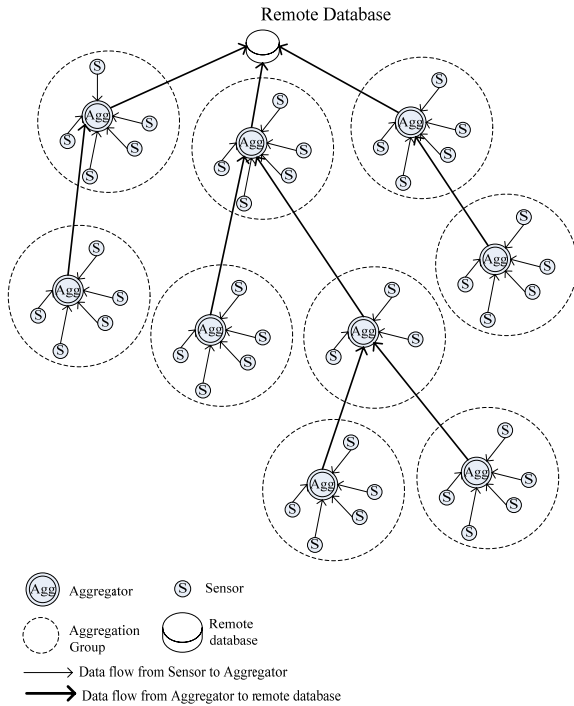


Figure 1. A clustered sensor network topology

Here we list notations we use in this paper:

Notations:

S_i : Sensor mote i

SID_i : Sensor identity of sensor mote i

g : An one-way function having the following property:

$$g(x \oplus y) = g(x) \oplus g(y)$$

K_i^{RK} : An *encryption key* randomly generated by sensor mote i

K_i^{VK} : A *verification key* used to verify data from sensor mote i

3.1 Group Forming

After arbitrarily deploy motes in a geographical area, the motes automatically form several non-overlapped groups. Each group is called an "*aggregation group*". When forming groups, the remote server broadcasts a beacon, containing an index of hop-count, *layer*, indicating the distance from received node to remote database, to all motes. Each mote then can be categorized according to different layer. Next each mote broadcasts a packet contains its *ID* and *layer* to other motes. If other motes can listen to this packet and has the same layer value, these nodes are then grouped into the same aggregation group. Figure 2 depicts overall group forming phase in detail.

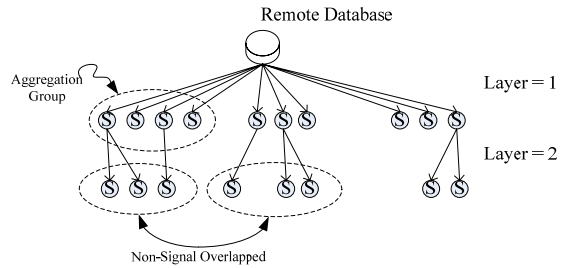


Figure 2. Group forming phase

3.2 Aggregator Election

After non-overlapped groups are formed, motes in the same group select a mote as the *group aggregator*. The group aggregator is responsible for forwarding readings to remote database after data are aggregated. Several Criteria for electing group includes: number of descendent, number of ascendant, remaining power, maximum RSSI, and so on. Different electing criteria cause different routing topology. How to find a best group aggregator is not our intention.

When groups are formed and group aggregators are elected, motes then forward their readings to the group aggregator in the same group. The aggregator when forward the aggregated value to its parent aggregator. Eventually, these readings from all motes will be forwarded to the remote database via the group aggregator whose layer value is one.

3.3 Initialization

Before deploying a wireless sensor network, we have to set up three roles: the sensor mote, the aggregator, and the remote database.

The sensor mote: Each sensor mote i is assigned a sensor ID SID_i , an one-way hash function f , an one-way function g , and a verification key K_i^{VK} .

The remote database: The remote database needs to decrypt aggregated data, and thus we need to store the one-way has function f , the one-way function g , and all verification key K_i^{VK} for all i .

The aggregator: The aggregator will be given the one-way has function f , the one-way function g , and all $K_i^{VK} \oplus K_j^{VK}$, $\forall i \neq j$. Hereafter, these keys are referred as *aggregation verification keys*. This step must keep aggregator from knowing K_i^{VK} or K_j^{VK} . Otherwise, the aggregator can decrypt the ciphertext passing through it.

3.4 Data Encryption Phase

Our encryption design aims to provide lightweight encryption overhead and secrecy while providing data aggregation property.

When a sensor mote i has a reading m_i and wishes to transmit this reading to the aggregator, it first uses its pre-installed verification key K_i^{VK} and the one-way function g to generate $g(K_i^{VK})$. Then, this sensor mote randomly generates a new key K_i^{RK} , which will be used as the next-round encryption key. Next, the sensor mote generates corresponding ciphertext $E_i(m_i)$:

$$E_i(m_i) = m_i \oplus g(K_i^{EK}) \oplus K_i^{EK} \parallel K_i^{EK} \oplus K_i^{VK}, \quad \text{Eq(1)}$$

where \parallel indicates data concatenation.

Our proposed scheme is very close to one-time pad method as each mote changes different key for encrypting data but provides more capabilities. Next, we will introduce how to find out redundant readings among these ciphertexts without decrypting them in our data aggregation phase.

3.5 Data Aggregation Phase

Our data aggregation method provides a pair-wise method to identify if two readings are identical. Although the goal of our data aggregation scheme is to find redundant readings among n incoming encrypted packets in the aggregator, our aggregation scheme can be further extended by pairing off these n incoming encrypted packets. By iteratively performing pair-wise comparisons we can eliminate all redundant readings

among them. If n same readings are encrypted and transmitted to the aggregator, the aggregator needs to check $n-1$ times to verify these inputs and save $n-1$ packet transmission. It needs computation overhead for data aggregation but saves more energy from fewer data transmissions.

In the following section, first we will introduce our approach to find redundant reading in two packets; then, we will introduce how to extend our approach to find redundant readings among n packets.

Assume sensor mote i and j sends two encrypted readings to the aggregator, and these encrypted readings can be expressed by the following equations:

$$E_i(m_i) = m_i \oplus g(K_i^{EK}) \oplus K_i^{EK} \parallel K_i^{EK} \oplus K_i^{VK}, \quad \text{Eq(2)}$$

$$E_j(m_j) = m_j \oplus g(K_j^{EK}) \oplus K_j^{EK} \parallel K_j^{EK} \oplus K_j^{VK}. \quad \text{Eq(3)}$$

First, the aggregator XOR the first parts of these two ciphertexts, and it can be expressed by the following equation:

$$m_i \oplus g(K_i^{EK}) \oplus K_i^{EK} \oplus m_j \oplus g(K_j^{EK}) \oplus K_j^{EK} \quad \text{Eq(4)}$$

Next, since the aggregator is pre-installed with $K_i^{VK} \oplus K_j^{VK}$, the aggregator can XOR the last two part of Eq(2) and Eq(3) to obtain:

$$\begin{aligned} & K_i^{EK} \oplus K_i^{VK} \oplus K_j^{EK} \oplus K_j^{VK} \oplus K_i^{VK} \oplus K_j^{VK} \\ & = K_i^{EK} \oplus K_j^{EK} \end{aligned} \quad \text{Eq(5)}$$

It can be found that the aggregator can use $E_i(m_i)$ and $E_j(m_j)$ to retrieve $K_i^{EK} \oplus K_j^{EK}$, but cannot retrieve K_i^{EK} or K_j^{EK} separately; therefore, the aggregator cannot decrypt $E_i(m_i)$ and $E_j(m_j)$.

Next, we define a check value V , and V is calculated by XOR Eq(4), Eq(5) and $g(K_i^{EK} \oplus K_j^{EK})$. The check value is used to distinguish if two encrypted readings are the redundant in their plaintext format. As a result, the check value V can be expressed by the following equation:

$$\begin{aligned} V_{(i,j)} &= m_i \oplus g(K_i^{EK}) \oplus K_i^{EK} \oplus m_j \oplus g(K_j^{EK}) \\ &\oplus K_j^{EK} \oplus K_i^{EK} \oplus K_j^{EK} \oplus g(K_i^{EK} \oplus K_j^{EK}) \end{aligned} \quad \dots \text{Eq(6)}$$

By using the properties of function g , Eq (6) can be further reduced to:

$$\begin{aligned} V_{(i,j)} &= m_i \oplus g(K_i^{EK}) \oplus K_i^{EK} \oplus m_j \oplus g(K_j^{EK}) \\ &\oplus K_j^{EK} \oplus K_i^{EK} \oplus K_j^{EK} \oplus g(K_i^{EK} \oplus K_j^{EK}) \end{aligned} \quad \dots \text{Eq(7)}$$

$$= m_i \oplus m_j$$

It is easier observed that if m_i equals to m_j , $V_{(i,j)} = m_i \oplus m_j = 0$, and vice versa. We can formally describe $V_{(i,j)}$ by the following equations:

$$\text{if } \begin{cases} V_{(i,j)} = 0, \text{ then } m_i = m_j \\ V_{(i,j)} \neq 0, \text{ otherwise} \end{cases} \dots \text{Eq(8)}$$

If these two readings are the same, the aggregator just needs to send either $E_i(m_i)$ or $E_j(m_j)$ to the remote server. If these two readings are different, the aggregator then sends $E_i(m_i) \parallel E_j(m_j)$ to the remote server. Since remote server is pre-installed with the verification key K_i^{VK} , the remote server therefore can use K_i^{VK} to obtain K_i^{EK} by

$$K_i^{EK} = (K_i^{EK} \oplus K_i^{VK}) \oplus K_i^{VK}.$$

Then, the original data m_i can be recovered by:

$$m_i = (m_i \oplus g(K_i^{EK}) \oplus K_i^{EK}) \oplus g(K_i^{EK}) \oplus K_i^{EK}.$$

4 Security Analysis

In this section, we use random oracle model to justify our protocol is secure in terms of provable security.

A random oracle [4][5] is a theoretical block box that replies to queries with random response chosen uniformly in its output domain. A methodology for designing a cryptographic protocol can be divided into two steps. In first step, one designs an ideal system in which all participants as well as adversaries have oracle access to a truly random function, and proves the security of the ideal system. In second step, we replace the random oracle by a ‘‘good cryptographic hashing function’’. We can therefore obtain an implementation of the ideal system in a real-world where random oracles do not exist. This methodology is referred to as the random oracle methodology.

Before we build our ideal system, we first describe the notion.

$\{0,1\}^*$: the space of finite binary strings

$\{0,1\}^\infty$: the space of infinite binary strings

$G: \{0,1\}^* \rightarrow \{0,1\}^\infty$: a random generator

f : a trapdoor permutation with inverse f^{-1}

k : the security parameter

$H: \{0,1\}^* \rightarrow \{0,1\}^k$: a random hash function

$G(r) \oplus x$: the bitwise XOR of x with the first $|x|$ bits of the output of $G(r)$

A function $\varepsilon(k)$ is negligible if for every c there exists a k_c satisfying $\varepsilon(k) \leq k^{-c}$ for every $k \geq k_c$. If A_p is a probabilistic algorithm, then for any inputs m_1, m_2, \dots $A_p(m_1, m_2, \dots)$ is the probability space

which to the string σ assigns the probability that A_p outputs σ . For probabilistic spaces S, T, \dots ,

$$P_r[x \leftarrow S; y \leftarrow T; \dots : p(x, y, \dots)]$$

denotes the probability that the predicate $p(x, y, \dots)$ is true after the execution of the algorithms $x \leftarrow S$, $y \leftarrow T$, etc.

For convenience, a random oracle R is a map from $\{0,1\}^*$ to $\{0,1\}^\infty$ chosen by selecting each bit of $R(x)$ uniformly for every x .

Our proposed scheme can be formulated as the following oracle:

$$E_{r_1, r_2}^G(m) = m \oplus H(G(r_1)) \oplus G(r_1) \parallel G(r_2) \oplus G(r_2) \dots \text{Eq(9)}$$

Known-Plaintext Security

For known-plaintext attacks, the adversary knows some m , and

$P_r[\text{The attacker successfully guesses } G(r_1)]$ can be described as:

$$P_r[r_1 \leftarrow G(r)] = \frac{1}{2^{|r_1|}} \approx 0, \text{ when } r_1 \text{ is large enough.}$$

We suggest that $|r_1| \geq 88$ is adequate.

Chosen-Plaintext Security

We adapt the notion of CP-adversary (chosen-plaintext adversary) in [3] to the random oracle model. A CP-adversary A is a pair of non-uniform polynomial algorithms (F, A_1) , each with access to an oracle. For an encryption algorithm \mathcal{G} to be secure, it requires that

$$P[\text{Chosen - Plaintext Fails}] =$$

$$P_r[R \leftarrow 2^\infty; (E, D) \leftarrow \mathcal{G}(1^k); (m_0, m_1)$$

$$\leftarrow F^R(E); b \leftarrow \{0,1\}; \sigma \leftarrow E^R(m_b) \dots \text{Eq(10)}$$

$$: A_1^R(E, m_0, m_1, \alpha) = b]$$

$$\leq 0.5 + k^{-w(1)}$$

In our proposed scheme,

$$P[\text{Chosen - Plaintext Fails}] = C_m^r \times \frac{1}{2^{|r|}},$$

is negligible to the random oracle.

Chosen-Ciphertext Security

The chosen-ciphertext attack is defined as: the adversary can adaptively choose ciphertexts and access to the decryption algorithm to get the corresponding plaintexts. Though it is usually occurred in asymmetric cryptographic systems, it can also be happened in our scheme as the adversary can know both ciphertexts and plaintexts (by using same sensors) in the same time. We adapt the definition of [3] and [11] to the random oracle [5] setting. An RS-adversary (‘‘Rackoff-Simon adversary’’) A is a pair of non-uniform algorithm $A = (F, A_1)$, each with access to an oracle R and a

black box implementation of D^R . The algorithm F is used to generate two messages m_0 and m_1 such that if A_1 is given the encryption α , A_1 won't be able to guess well whether α comes m_0 or m_1 . Formally, an encryption scheme \mathcal{G} is secure against RS-attack if the following equation is satisfied:

$$\begin{aligned} &P[\text{Chosen - Ciphertext Fails}] = \\ &P_r[R \leftarrow 2^\infty; (E, D) \leftarrow \mathcal{G}(1^k); (m_0, m_1) \\ &\leftarrow F^{R, D^R}(E); b \leftarrow \{0,1\}; \alpha \leftarrow E^R(m_b): \\ &A_1^{R, D^R}(E, m_0, m_1, \alpha) = b] \leq 0.5 + k^{-w(1)} \end{aligned} \quad \text{Eq(11)}$$

To see our scheme is secure against chosen ciphertext attacks, we prove the above equation is satisfied. Let A_k denotes the event that $a \parallel b \leftarrow F(E)$, for some a and b . Let $A = (F, A_1)$ be an RS-adversary that succeeds with probability $\frac{1}{2} + \lambda(k)$ for some non-negligible function $\lambda(k)$. The adversary A can make some oracle call of $G(r_1)$ or $H(G(r_1))$. Let L_k denotes the event that A_1 asked $D^{G, H}$ some queries where $a = m \oplus f^{-1}(r_1) \oplus H(f^{-1}(r_1))$, but A_1 never asked its H -oracle for $H(f^{-1}(r_1))$. Let $n(k)$ denotes the total number of oracle queries made. It is easy to see that $Pr[L_k] \leq n(k)2^{-k}$ and $Pr[A \text{ succeeds } \bar{L}_k \cap \bar{A}_k] = 0.5$ according to [3].

Thus

$$\begin{aligned} P_r[A \text{ succeeds}] &= P_r[\text{Chosen - Cipher Attack succeeds}] \\ &= \frac{1}{2} + \lambda(k) \end{aligned}$$

is bounded above by

$$\begin{aligned} &P_r[A \text{ succeeds } L_k] P_r[L_k] \\ &+ P_r[A \text{ succeeds } \bar{L}_k \cap A_k] P_r[\bar{L}_k \cap A_k] \\ &+ P_r[A \text{ succeeds } \bar{L}_k \cap \bar{A}_k] P_r[\bar{L}_k \cap \bar{A}_k] \\ &\leq n(k)2^{-k} + P_r[A_k] + \frac{1}{2} \end{aligned}$$

Therefore, our proposed scheme satisfies eq. (11), and is chosen-ciphertext-attack resistant.

5 Conclusion

In this paper, we proposed a secure encrypted-data aggregation scheme in mobile wireless sensor networks. Our scheme has the following enhancements: 1) the aggregator does not need to decrypt its received encrypted-data to verify if these data are the same; no extra power are wasted in data decryption, 2) the

aggregator does not have decryption keys and therefore cannot know anything about the data, and 3) our proposed scheme uses random keys to encrypt data; this property makes our scheme resilient to known-plaintext attacks, chosen-plaintext attacks, ciphertext-only attacks, and man-in-the-middle attacks. Aiming at secrecy and privacy, our proposed scheme is resilient to several attacks in sensor networks, and makes data aggregation more practical in these environments.

6 References

- [1] M. Acharya and J. Girao, "Secure comparison of encrypted data in wireless sensor networks", In 3rd Intl. Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks, p.47-53, 2005.
- [2] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks", IEEE Communication Magazine, p.102-114, 2002.
- [3] J. Al-Karaki, R. Ul-Mustafa, and A. Kamal, "Data aggregation in wireless sensor networks – exact and approximate algorithms", In Proceedings of the Workshop on High Performance Switching and Routing, p.241-245, 2004.
- [4] M. Bellare and P. Rogaway, "Random oracles are practical: a paradigm for designing efficient protocols," In Proceedings of 1st Conference on Computer and Communications Security, p. 62-73, 1993
- [5] R. Canetti, O. Goldreich, and S. Halevi, "The random oracle methodology, revisited", In Proceedings of the 30th Annual ACM Symposium on the Theory of Computing, p.209-218, 1998.
- [6] R. Chandramouli, S. Bapatla, and K. P. Subbalakshmi, "Battery power-aware encryption," In Proceedings of ACM Transactions on Information and System Security, p.162-180, 2006.
- [7] J. Girao, D. Westhoff, and M. Schneider, "CDA: Concealed data aggregation for reverse multicast traffic in wireless sensor networks", In Proceedings of 40th International Conf. on Communications, p.3044-3049, 2005.
- [8] L. Hu and D. Evans, "Secure aggregation for wireless networks", In Proceedings of Applications and Internet Workshops, p.27-31, 2003.
- [9] C. Intanagonwiwat, R. Govindan, D. Estrin, and J. Heidemann, "Directed diffusion for wireless sensor networking", In IEEE/ACM Transactions on Networking, p.2-16, 2003.
- [10] T. Li, Y. Wu, and H. Zhu, "An Efficient Scheme for Encrypted Data Aggregation on Sensor Networks," In Proceedings of Vehicular Technology Conference, p. 831-835, 2006.
- [11] C. Rackoff and D. Simon, "Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack," In Proceedings of Advances in Cryptology, 1991