Design and Implementation of Wireless Sensor Network Re-encryption Protocol Based on Key Distribution

Huaming Song\textsuperscript{1,2}, Jian Zhou\textsuperscript{1,*}

\textsuperscript{1} School of Information Engineering, Wuhan University of Technology, Wuhan, China
\textsuperscript{2} Department of Science and Technology, Xianning City Public Security Bureau, Xianning

*Corresponding Author: jianzhou@whut.edu.cn

Abstract. Since its introduction, wireless sensor networks (WSNs) have been the focus of researchers and have a wide range of applications, some of which involve sensitive and confidential information, which has caused security problems. Due to limitations in memory, energy, and computing power of wireless sensor networks, adding appropriate security to sensor nodes poses a barrier. This paper introduces a new security encryption protocol framework for wireless sensor network. The implementation of this framework will provide effective security for sensor nodes and we prove the effectiveness and superiority of the proposed framework compare with existing research.

1. Introduction

In this era of globalization, information is easily accessible, so its security is not guaranteed. When a sensor network processes data and transmits signals between sensor nodes and receivers (such as smartphones), unauthorized users can intercept the information, which is a security issue in wireless sensor networks. A large number of studies have been conducted by researchers in the field of wireless sensor network security. Pu and Chung\cite{1} proposed a set of key update methods for using RC4 in wireless sensor networks and implemented them on the Telos platform. Aziz et al.\cite{2} introduced a wireless system using Xbee to monitor and process temperature and humidity data. Kukkurainen et al.\cite{3} described message encryption and message authentication codes based on RC5 block ciphers in the ILAVI sensor network platform. ILAVI is an open platform for deploying low data rate and low power applications\cite{4}. Dickerson et al.\cite{5} proposed an RC4-based cryptographic sensor network and tested it with the Atmega128 microcontroller in the Mica2 network sensor. They also compared the performance between RC4 and RC5.

Existing research has focused on selecting the right sensor devices and implementing security in the network. By analyzing the background research, it can be concluded that symmetric key encryption is more suitable for wireless sensor networks than public key encryption because the rules for changing keys make it more robust. In earlier studies\cite{6}, \cite{7}, \cite{8}, the authors described methods for re-encryption in WSN. In the design of this paper, the shortcomings of the above research are overcome. According to the study by Nilsson et al.\cite{7}, five encryption keys are required to complete their design, and in the design of this paper, only two encryption keys are needed. Hu et al.\cite{8} designed and implemented the framework in practice, but did not discuss the method of generating new keys. In the architecture designed in this paper, a hash function is used to generate a new key of 40 bytes long. Herrera et al.\cite{6} used six steps to create a new key that was sent by the sensor node and receiver over the network. The
design of this article requires only two steps to build a new key and does not require a key to be sent over the network. All of the above three aspects involve only the key allocation mechanism, and there is no description of the data transmission procedure. The framework designed in this paper includes key distribution and data transmission part to ensure the integrity of data authentication and its actual implementation.

The main contribution of this paper is to design and develop a framework for wireless sensor network security measures. Through authentication and verification, a new re-encryption method based on symmetric key and data transmission security is discussed, from hardware design to program software implementation. The solution has been implemented in a real-time DigiMesh network based on ZigBee's wireless sensor network and tested using real-time sensors. Finally, the test results and performance of the framework are compared with the previous research results, which verifies the superiority of the proposed framework.

The rest of this article is organized in the following order. The introduction section introduces the application of wireless sensor networks and attacks in the network, summarizing the security requirements of wireless sensor networks. The second part describes the design and implementation process of the security framework designed in this paper. The third part analyzes and tests the performance of the system and compares it with the existing research results. Finally, the paper summarizes the paper and proposes future improvements.

2. Security framework design
This section highlights the details of the security framework design presented in this article, including how to implement security features and prevent attacks. This section proposes a new wireless sensor network security protocol based on re-encryption, which ensures the confidentiality and authentication of sensor data.

The symbols used in the security framework designed in this paper are shown in Table 1.

| Symbol | Description |
|--------|-------------|
| $N_1,...,N_N$ | Sensor Nodes |
| $R$ | Receiver/Base Station |
| $TO_p$ | One time password based on system time |
| $M_k$ | Initial Symmetric key between receiver and sensor nodes |
| $NM_k$ | Unique new symmetric key for every sensor shared with receiver |
| $E_k$ | Encryption operation executed by the sensor nodes |
| $D_k$ | Decryption operation executed by the receiver |
| $\text{hash}$ | Operation to create new key |
| $H()$ | Generate HMAC for data verification |

The framework for re-encryption is shown in Figure 1.
At initialization, the sensor node loads the initial primary key, and the receiver also has the same key, which can be re-queried periodically in the node. Each node contains a software-based real-time clock (RTC) to generate a password. To prevent a reply attack, this one-time password is included with each re-encryption start up. This password is a randomly generated number, depending on the real-time clock for that particular system. Here is a function that takes system time $F(t)$ as input and produces a one-time password as output.

$$\text{Onetimepassword}(T_{op}) = F(\text{system-time})$$  \hspace{1cm} (1)

The system time is based on the current date and time, so the password changes after it is used once. As mentioned earlier, this one-time password prevents reply attacks and prevents attackers from replying to old messages. This is a basic requirement for re-encryption operations in the proposed framework. This unique password is encrypted using the initial master key and sent to the recipient along with the signature, the signature is a Hash Message Authentication Code (HMAC), the recipient decrypts the password, retrieves the private password and verifies the signature.

To perform the encryption operation, you can use the stream cipher based encryption algorithm RC4. The new master key is formed in the sensor node and receiver using a one-way hash function of the initial primary key and the one-time password, and then the receiver confirms to the sensor node that a new master key has been created. In order to meet the requirement that the key is not reused multiple times, the key generated each time is different. The new primary key is based on the SHA-1 hash function. The new primary key is 40 bytes long and $h(\cdot)$ is generated using the following formula:

$$\text{Newmasterkey}_{NMK} = h(T_{op}||MK)$$  \hspace{1cm} (2)

When a new master key is generated, the sensor node uses this key to encrypt the sensor data and uses a cryptographic hash to calculate the signature, and then transmits the encrypted data and signature to the recipient. Once the recipient receives the encrypted data and signature, the new master key is used to decrypt the sensor data, then a signature is formed, the new signature is compared to the received signature and verified, and finally, if all the content is Authentication, the data will be displayed on the monitor. The signature is generated using the following function, where HMAC is a message digest based on the key and input hash function:

$$\text{Signature} = \text{HMAC}_{NMK}(\text{“SensorData”})$$  \hspace{1cm} (3)

3. Experiment and evaluation

This section demonstrates the performance of the security protocol framework proposed in this paper from the aspects of execution time, power, energy, delay and throughput, including testing the performance of the encryption and encryption hash algorithm on the designed architecture, and presenting the test results.
This paper compares the proposed security framework with existing research in terms of execution time, power and energy consumption. Herrera et al.[6] tested the performance of their proposed protocol on the CSIRO protein model[9]. This protein model consists of an integrated TPM, which is built into RSA. The author encrypts the data using RSA. For signature generation and verification, they use cryptographic hash SHA-1. Hu et al.[8] evaluated their agreement on trusted fleck, which is based on the ATMEG AT97SC3203S TPM chip. They tested their performance on the basis of RSA, TinyECC and XTEA, but they concentrated the results on the RSA as if it was as established in the TPM. During the work, the researchers measured the performance of individual sensor nodes based on key criteria. They uploaded the program and monitored the performance of individual sensor nodes based on execution time, power, and energy. In the experiments in this paper, the same test was performed and then compare the test results with their work. In the comparison process, memory constraints are also considered to implement the encryption algorithm.

3.1. Execution time
Figure 2 shows the comparison between the proposed framework and the previous two studies, the execution time of the new key. For key generation, the framework of this paper is close to the previous research, and in other operations, the framework proposed in this paper performs better. The encryption and decryption operations of this paper require 8.5 milliseconds and 8.7 milliseconds respectively, while the encryption of Herrera et al.[6] requires 257 milliseconds, the decryption requires 972 milliseconds, and the encryption and decryption of Hu et al.[8] requires 55ms and 750ms respectively. Better performance. The framework proposed in this paper consumes 4.5ms for signature and 26ms for initialization. In contrast, Herrera and Ayre[6] studied 305ms for signature and 836ms for initialization, respectively. Hu et al.[8] did not mention initialization in their study and their protocol consumed 787ms for signatures, which exceeded the time spent on the architecture proposed in this paper.

![Figure 2: Comparison of execution time of Herrera et al. and Hu et al.](image)

3.2. Current consumption
Figure 3 shows a comparison of the current consumption of the study in this paper with the research done by Herrera. Figure 3 shows that in almost all operations, the design consumes more current than the other two designs. Herrera et al.[6] required a 70 mA generated key, 80 mA encryption, 80 mA decryption, 80 mA signature and 60 mA initialization. The framework proposed in this paper requires 116mA for key generation, 11ma for encryption, 114ma for decryption, 115ma for signature, and 117Ma for key initialization, reflecting that the ArduinoUno and XBee radios consume more current. Hu et al.[8] did not discuss key generation or initialization, but their design requires 50.4 mA for encryption, 60.8 mA for decryption, and 60.8 mA for signature generation, which consumes less current than Herrera et al.
3.3. Energy consumption

Figure 4 shows the energy consumption comparison between the protocol designed in this paper and the architecture designed by Herrera. Figure 4 reveals that in the various operations of the framework proposed in this paper, if it matches the other two synergies, its energy consumption will continue to decrease. Among all the functions, the energy required for initialization is the most, the energy consumed by the framework of this paper is 15.21mJ, and other energy-consuming functions include 2.32mJ for key generation, 4.80mJ for encryption, 4.80mJ for decryption, and 2.32mJ for signature. According to Herrera's agreement, 5.7mJ is used for key generation, 84.3mJ for encryption, 326.59mJ for decryption, 326.59mJ for signature, and 273.87mJ for initialization, exceeding the capabilities of the protocol proposed in this paper. Consumption. Hu et al. [8] limited their protocols to encryption, decryption, and signature, but consumed less energy than Herrera. They consumed 8.31 mJ for encryption, 136 mJ for decryption, and 143 mJ for signature.

3.4. Delay

The delay is an indicator of the time required to transmit data between the sender and the receiver. The delay can be determined as a single trip delay or round trip delay. In one transmission, the one-way transmission time is measurable, and the round-trip is the two-way transmission time. In the experiment, this paper considers the delay of various data sizes. In this paper, the effects of different data sizes are tested in several operations on the sensor nodes. In order to calculate the delay time, the sensor nodes closest to and farthest from the receiver are considered. The delays of different sensor nodes are measured while encrypting various data size values and generating their signatures using a hash function. During the experiment, the receiver was equipped with an Xbee connected to the laptop, on the other side, there were four sensor nodes, two in the range of the receiver, two outside the range of the receiver, and the nearest node could act as far from the receiver An intermediate router that transmits data to end nodes.
The framework proposed in this paper first considers the nearest node for data encryption and signature generation. In the encryption process, about 100 sample data is used and averaged, and the same process is repeated when the signature is generated. A similar process is repeated when encryption and signature generation are performed on all other nodes. To complete these measurements, a program is written at the receiving end that calculates the number of bytes and extracts the time difference between successive data. In the experiments in this paper, the data length was limited to 127 bytes, because XBee based on ZigBee network can transmit up to 127 bytes of data.

The data is encrypted at the beginning of the experiment, and Figure 5 compares the time it takes for the data to be transmitted with or without an encryption key.

Figure 5 shows the time required for data transfers of 8, 16, 32, 64, and 127 bytes. When data is not encrypted, time is consumed less, while encrypted data takes more time. It is worth noting that the increase is linear.

In the second step of the delay experiment, the signature generation time for various data sizes was calculated. Regardless of the size of the data, the signature generation time is similar, staying within the range of 64-65 milliseconds as shown in Figure 6. Contrary to previous experiments, the signature generation time presents a different trend. In the first experiment, the delay was increased, and the delay of the hash operation was negligible, and the time required for any size of data was almost the same.

The reason for this is that the hash operation creates the same output size for any given input data size, which is why the use of hash generation signatures can ignore the delay.

Throughput

Throughput refers to the number of bits transmitted over the network over a specific time period. The throughput depends on the delay discussed in the previous section, and the formula for calculating throughput is\[10\].
Figure 7 Throughput for sensor data with and without encryption

The figure above shows the throughput of encrypted and unencrypted sensor node data. As the size of the data increases, throughput increases linearly. Furthermore, if encryption is not done, throughput is greater than with encryption. At the same time, the hash function is applied to the sensor node. After the hash function is introduced, the performance of the sensor node in terms of throughput is improved, as shown in Figure 8. As can be seen from the figure, the throughput of the hash function also increases linearly with the increase of the data size. As with the previous section, as the data size increases, the execution of the hash operation is better than the output data size. Encryption operation.

Figure 8 Throughput for sensor data with hash generating signature

4. Conclusions and future work

After successfully implementing the security framework, this paper studies the performance of the design, which is measured in terms of execution time, power consumption and power. The experimental results were compared with the existing studies and showed good performance. Through the research in this paper, we can conclude that secure key distribution and data reading based on re-encryption can provide better security in the network.

In summary, this article achieves the following goals:

(1) Propose and implement symmetric encryption based on key distribution protocol, and periodically generate a new 40-byte key;

(2) The framework implements data integrity, confidentiality, authenticity, and forward and reverse secrecy. This design prevents common attacks such as man-in-the-middle attacks, back-trace attacks, tampering, witch attacks, and flood attacks;

(3) The performance of the proposed security framework is measured by execution time, energy efficiency, current consumption, delay, and throughput. The test results of the protocol are compared
with existing research work, considering execution time and energy. When consumed, good results have been achieved.

Future work includes the following aspects:

1. This protocol has been tested on a small network with four nodes. In the future, by introducing more nodes into the network, the robustness of the protocol can be verified.
2. Since each smartphone is equipped with a Bluetooth device, the protocol can be further extended, the addition of Bluetooth will ensure that the framework is extended to a larger user community;
3. After processing the data in the GUI, the data can be further transmitted through the Internet, which will help to expand the application scope of the protocol framework;
4. The response speed of the protocol can be tested by different real-time attacks on the sensor nodes. The system can be upgraded with a key longer than 40 bytes.

Reference

[1] Chuan-Chin Pu and Wan-Young Chung. Group key update method for improving RC4 stream cipher in wireless sensor networks. In International Conference on Convergence Information Technology, pages 1366–1371. IEEE, 2007.
[2] Noor Hafizah Abdul Aziz, Suzi Seroja Sarmin, Kama Azura Othman, Norfishah Wahab, Norzatina Misman, and Ahmad Tarmizi Hashim. Supervisory data acquisition of temperature and humidity in oil palm tissue culture laboratory. In Second International Conference on Computational Intelligence, Modelling and Simulation (CIMSiM), pages 470–475. IEEE, 2010.
[3] Juha Kukkurainen, Mikael Soini, and Lauri Sydanheimo. RC5-based security in wireless sensor networks: utilization and performance. WSEAS Transactions on Computers, pages 1191–1200, 2010.
[4] Arduino UNO description. http://arduino.cc/en/Main/arduinoBoardUno. Accessed: 2019-02-20.
[5] Damian Dickerson, Sung H Yoon, Jung H Kim, and Gi T Hur. Efficient RC4 based encryption system for sensor network. Proc. SPIE, 5778: 819–823, 2005.
[6] Adrian Herrera and Wen Hu. A key distribution protocol for wireless sensor networks. In Proceedings of the 2012 IEEE 37th Conference on Local Computer Networks (LCN 2012), pages 140–143. IEEE Computer Society, 2012.
[7] Dennis K Nilsson, Tanya Roosta, Ulf Lindqvist, and Alfonso Valdes. Key management and secure software updates in wireless process control environments. In Proceedings of the First ACM Conference on Wireless Network Security, pages 100–108. ACM, 2008.
[8] Wen Hu, Hailun Tan, Peter Corke, Wen Chan Shih, and Sanjay Jha. Toward trusted wireless sensor networks. ACM Transactions on Sensor Networks (TOSN), 7(1):1–5, August 25, 2010.
[9] Raja Jurdak, Kevin Klues, Brano Kusy, Christian Richter, Koen Langendoen, and Michael Brunig. Opal: A multiradio platform for high throughput wireless sensor networks. Embedded Systems Letters, IEEE, 3(4):121–124, 2011.
[10] Rajeev Piyare and Seong-ro Lee. Performance analysis of XBee ZB module based wireless sensor networks. International Journal of Scientific & Engineering Research, 4(4):1615–1621, 2013.