An Efficient Key Distribution Scheme in Wireless Sensor Architecture with Arduino and XBee

A. F. M. Sultanul Kabir
Department of Mathematics and Computer Science
University of Lethbridge
Lethbridge, Canada

ABSTRACT
Since the time of their introduction, Wireless Sensor Networks (WSN) have been catching the interest of researchers. WSN have a wide range of applications, some even involving sensitive and secret information, thereby raising security concerns. Nevertheless, WSN have some constraints like limited memory, energy and computational capability, which pose an obstacle for the addition of proper security in sensor nodes. This paper introduces a new rekeying design for WSN security framework whose implementation would dispense effective security in the sensor nodes. This proposed security framework is endowed with the capacity to address security issues, such as message integrity, confidentiality, authenticity and freshness based on symmetric key cryptography. In addition, this design does not allow the storage of any key except the initial master key in the sensor nodes prior to network deployment. This paper also investigates reconfigurable sensor nodes in terms of execution time, memory, power consumption, and cost while running the security framework.

Keywords
WSN, rekeying, security, symmetric cryptography

1. INTRODUCTION
In WSN, data pass through the wireless medium and, hence, are sensitive to external attacks. In addition, WSN has been deployed in different process control environments like water, gas or oil usages, in monitoring and billing. In these infrastructures, secret information is being passed over the network. Scientists are proposing different cryptographic solutions for efficient network security. Key management is one of the pivotal issues regarding the security of WSN. This motivates us to introduce a new symmetric rekeying mechanism.

Previous research shows that few numbers of asymmetric and symmetric key based solution exist [1] [2] where a cryptographic key is preloaded to the sensor nodes prior to network deployment. This approach has certain advantages like less memory and message overhead. However, a single compromised node leads to the destruction of the whole network [3]. This shortcoming of the key management has lead us to design a Wireless Sensor Network based on rekeying. As the sensor nodes operate for a longer period of time, rekeying is very important after a certain period. Rekeying is proposed by [4] [5] [6] which are based on combination of public key and symmetric key, among which two of them are practically implemented. The other details of the background research is discussed in the next section. Moreover, this design also considers the cost, limited computational and energy resource facility in the node level along with providing efficient security in the network.

2. RELATATE WORK
RSA and elliptic curve cryptography are widely used in public key cryptography. According to some researchers, due to the limitations of the sensor nodes these two techniques may place a heavy burden over the network. In spite of this, Gura et al. [7] and Watro et al. [8] tested public key cryptography in resource constrained WSN. The authentication and key agreement between two sensor networks is permitted in this protocol without letting any rekeying among the nodes.

Minisec [9] was implemented on Telos mote platform. It provides high level security while consuming less energy. An encryption key is stored in the sensor node in Minisec but the actual technique for this is not yet revealed. There is still an opportunity to upgrade Minisec by improving the key management strategy. Karlof et al. [10] implemented an efficient link layer encryption protocol for WSN called TinySec which is less energy and memory consuming. In this design a network key was loaded before the network formation. One of the major drawbacks is, it does not prevent reply attacks. Liu et al. [11] presented faster and more energy efficient public key cryptography algorithm for WSN named as TinyECC but some factors concerned with the patent have limited its use.

Nilsson et al. [5] presented asymmetric key management for the wireless process control environment. In their design, they proposed a key changing methodology of sensor nodes after a certain period of time. Their main intention was to design and verify the framework. Hu et al.[6] designed and implemented a sensor node which incorporates a RSA public key and an XTEA symmetric key. They analyzed the performance in terms of computation time, memory size, energy and cost. Herrera et al. [4] published a paper which focused on key distribution in WSN. They designed and implemented symmetric key distribution among sensor nodes using public key cryptography. They claimed their design as energy efficient as well as scalable.

In the design, the proposed framework shows key distribution as well as data sending which ensures data authentication and integrity with its real life implementation.

3. PROPOSED KEY DISTRIBUTION SCHEME
At the time of initiation, the sensor nodes are loaded with the initial master key and the receiver also possesses the same key. Rekeying can be instituted in the nodes periodically. Each node containes a software based real time clock (RTC) to generate a one time password. To protect against reply attacks, each and every rekeying initiation is accompanied with this one-time password. This password is a randomly generated number depending upon the real time clock of that particular system. Here F () is a random number generator function which takes the system time as input and produces a one time password as output.

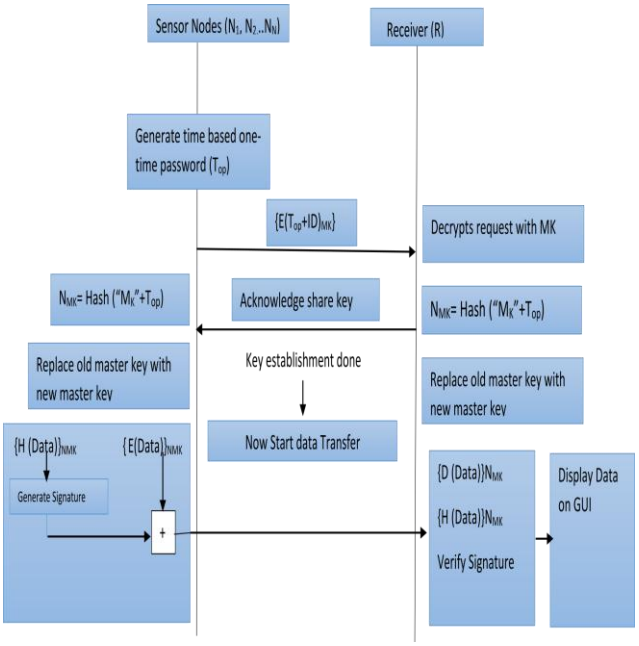

Figure 1: Proposed key Distribution and Data Transfer between Sensor Node and Receiver

At the time of initiation, the sensor nodes are loaded with the initial master key and the receiver also possesses the same key. Rekeying can be instituted in the nodes periodically. Each node containes a software based real time clock (RTC) to generate a one time password. To protect against reply attacks, each and every rekeying initiation is accompanied with this one-time password. This password is a randomly generated number depending upon the real time clock of that particular system. Here F () is a random number generator function which takes the system time as input and produces a one time password as output.

One time password (T_{op}) = F (system_time)

The system time is based on the current date and time, so the password changes after using it once. As previously mentioned, this one time password protects against reply attacks as well as defends an attacker to reply to old messages which is a fundamental requirement for the rekeying operation in proposed framework.

This unique password is encrypted with an initial master key and is sent to the receiver along with a signature. This signature is a hash message authentication code (HMAC). The receiver decrypts the password, retrieves that exclusive password and verifies the signature. To perform the encryption operation stream cipher based encryption RC4 is used. The new master key is formed in both the sensor nodes and in the receiver using a one way hash function from the initial master key and the one-time password.

Table 1 Notation used in the proposed scheme

Symbol	Description
N_1,N_2…..N_n	Sensor Nodes
R	Receiver/ Base Station
T_{op}	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
Hash ( )	Operation to create new key
H ( )	Generate HMAC for data verification

Following this, the receiver acknowledges to the sensor nodes that a new master key has been created. To address the requirement of not reusing a key more than once, each produced key is distinct. The new master key is the product of a hash function Hash ( ) based on SHA-1. It is produced using the following formula :

New master key $NM_K = Hash\ (M_k+T_{op})$

When the new master key is generated, the sensor nodes encrypt sensor data using this key and also calculate a signature using a cryptographic hash. The encrypted data and signature are then transmitted to the receiver side.

As soon as the receiver receives the encrypted data and signature, it uses the new master key to decrypt the sensor data and this is followed by the formation of a signature. The new signature is compared with the received signature and verified. Finally, if everything is authenticated, the data is displayed on the monitor. The signature is generated using the following function on both sides. Here HMAC is the message digest of hash function based on the secret key and input.

Signature= $H\{ NM_K(\text{“Sensor Data”})\}$

4. SECURITY ISSUES ATTAINED BY THE DESIGN

The proposed framework has achieved the following security issues

4.1 Confidentiality

In the design, symmetric key cryptography RC4 is used to encrypt a new key generation session as well as during data transmission. As data is encrypted, it is protected from the intruders and confidentiality is ensured.

4.2 Data integrity and Authenticity

The data should not be altered or modified during transmission known as data integrity, and to confirm that the data is from a genuine sender is termed data authenticity. The incorporation of HMAC safeguards both data authenticity and integrity. Cryptographic hash SHA-1 is used to produce HMAC to attain both data integrity and authenticity.

4.3 Forward and Backward Secrecy

When the attackers compromise a current key, they should not gain access to either the preceding messages encrypted with the former keys or the upcoming messages encrypted with the onward keys. These are referred to respectively as backward and forward secrecy. To attain forward and backward secrecy, the use of the same key for a long period of time is avoided. Instead, there is an option for changing that key according to the requirements of the application.

5. IMPLEMENTATION

The implementation phase is divided into two parts. One part deals with sensor nodes and another part is all about receiver.

5.1 Sensor Node

The sensor node was composed of Arduino Uno, XBee shield, XBee, Breadboard and Temperature sensors. Arduino Uno is a microprocessor based controller which executes program in the sensor nodes. To program in Arduino Uno we used sketch. By using sketch we written a program to read sensor data, generated key, encrypted key, signature generation and verification. For encryption, RC4 based stream cipher is used whereas for signature generation and verification we implemented SHA-1. The working sensor node is depicted in the following figure.

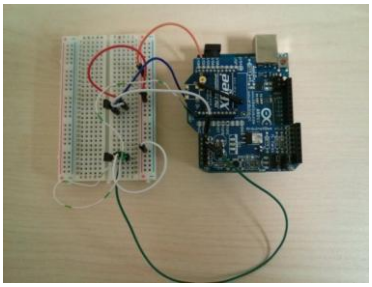

Figure.2 Working Sensor node

5.2 Receiver Side

The receiver side consisted of a computer with XBee. The XBee read data and sent it to laptop through COM pot. In the receiver side, Java is used to process the data.The same cryptographic operation is implemented like sensor node in the receiver side which was RC4 for encryption and SHA-1 for signature verification.

6. PERFORMANCE

6.1 Execution Time

In the testing phase the execution time was divided into two parts. One portion dealt with the time taken by the sensor nodes and the second portion was the time consumed by the receiver. Code was written, both on the sender and the receiver side, to measure the execution time. In the sensor node, the Arduino UNO was loaded with a program which measured the time consumption for each operation. The clock was started before the operation, which recorded the starting time, while the clock was stopped after operation, which recorded the ending time. After that, the measurement was taken by subtracting the starting time from the ending time. The same procedure was followed on the receiver side. For better performance, the time was calculated in nanoseconds and it was later converted to milliseconds.

The sensor node spent 26 ms to initialize the key, 8.5 ms for encryption, 4.5 ms for signature and 0.5 ms in sensor reading. Therefore, 39.5 ms time in total was needed from sensor reading to cryptographic operation. In contrast, the receiver side needed 2.65 ms time in total from generation of key to cryptographic operation, which included 0.35 ms for key generation, 1.4 ms for decryption and 0.9 ms for signature. When considering temperature reading, the proposed architecture took about 42.15 ms for the sender and receiver portions together. The transmission time for ZigBee DigiMesh varied based on the distance of the node. In the testing phase, the nearest node took around 85-100 millisecond for 8 byes of data, whereas the farthest node took around 120-130ms, which made the total time consumed approximately 120 millisecond for the closest node and 165 millisecond for the farthest node. The computation time in the sensor node is depicted in table

Table 2 Execution time (Sensor node)

Operation	Time (ms)
Key initialization	26
New rekey generation	4
Encryption	8.5
Decryption	8.7
Signature	4.5
Sensor read	0.5

6.2 Current Consumption

In the proposed design, the current consumption was measured by the multimeter. For example, a code for signature generation was uploaded in Arduino Uno and then the current consumption of this mote was measured using the multimeter. This procedure was repeated for different operations. A library of the Arduino was used which helped Arduino to consume less current.

For sensor node operations the current consumption for diverse functions ranges from 50-57 mA. However, by adding radio, the current consumption increases which is clearly depicted in table 3.

Table 3 Current Consumption (Sensor node)

Operation	Current consumption (mA)
Key initialization	117 mA
New rekey generation	116 mA
Encryption	113 mA
Decryption	114 mA
Signature	115 mA
Sensor read	110 mA

7. CONCLUSION

This scheme was based on the symmetric encryption technique. Following the implementation, the security issues attained by our offered design are mentioned in the design and analysis section. At the time of key formation, a time based password was used with hash function which also provided uniqueness to the key.

In this paper, after designing, the protocol was implemented in real life with the help of an Arduino Uno microcontroller and XBee, along with sensors and a laptop as the receiver. After successful implementation of the framework, the empirical performance of the design was studied. The performance of the proposed scheme was measured in respect to execution time and current consumption.

8. REFERENCES

[1] B. Dutertre, S. Cheung, and J. Levy, "Lightweight key management in wireless sensor networks by leveraging initial trust," tech. rep., Technical Report SRI-SDL-04-02, SRI International, 2004..

[2] L. Eschenauer and V. D. Gligor, "A key-management scheme for distributed sensor networks," in Proceedings of the 9th ACM conference on Computer and communications security, pp. 41–47, ACM, 2002.

[3] A.-N. Shen, S. Guo, and H.-Y. Chien, "An efficient and scalable key distribution mechanism for hierarchical wireless sensor networks," in Sarnoff Symposium, 2009.

SARNOFF'09. IEEE, pp. 1–5, IEEE, 2009

[4] A. Herrera and W. Hu, "A key distribution protocol for wireless sensor networks," in Proceedings of the 2012 IEEE 37th Conference on Local Computer Networks (LCN 2012), pp. 140–143, IEEE Computer Society, 2012.

[5] D. K. Nilsson, T. Roosta, U. Lindqvist, and A. Valdes, "Key management and secure software updates in wireless process control environments," in Proceedings of the first ACM conference on Wireless network security, pp. 100–108, ACM, 2008..

[6] W. Hu, H. Tan, P. Corke, W. C. Shih, and S. Jha, "Toward trusted wireless sensor networks," ACM Transactions on Sensor Networks (TOSN), vol. 7, no. 1, p. 5, 2010.

[7] N. Gura, A. Patel, A. Wander, H. Eberle, and S. C. Shantz, "Comparing elliptic curve cryptography and rsa on 8-bit cpus," in Cryptographic Hardware and Embedded Systems-CHES 2004, pp. 119–132, Springer, 2004.

[8] R. Watro, D. Kong, S.-f. Cuti, C. Gardiner, C. Lynn, and P. Kruus, "Tinypk: securing sensor networks with public key technology," in Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks, pp. 59–64, ACM, 2004.

[9] M. Luk, G. Mezzour, A. Perrig, and V. Gligor, "Minisec: a secure sensor network communication architecture," in Proceedings of the 6th international conference on Information processing in sensor networks, pp. 479–488, ACM, 2007.

[10] C. Karlof, N. Sastry, and D. Wagner, "Tinysec: a link layer security architecture for wireless sensor networks," in Proceedings of the 2nd international conference on Embedded networked sensor systems, pp. 162–175, ACM, 2004

[11] A. Liu and P. Ning, "Tinyecc: A configurable library for elliptic curve cryptography in wireless sensor networks," in Information Processing in Sensor Networks, 2008. IPSN'08. International Conference on, pp. 245–256, IEEE, 2008.