SCUR: Secure Communications in Wireless Sensor Networks using Rabbit

Ruhma Tahir, Muhammad Younas Javed, Attiq Ahmad and Raja Iqbal

***Abstract*—In this paper we propose a Lightweight Encryption Mechanism based on the Rabbit stream cipher for providing confidentiality in Wireless Sensor Networks (WSNs) that fulfils both requirements of security as well as energy efficiency. Our proposed security protocol is an idea for resource constrained WSNs, and can be widely used in the applications of secure communication where the communication nodes have limited processing and storage capabilities while requiring sufficient levels of security. The features of SCUR lead to the conclusion that this particular scheme might be more efficient in terms of security and resource consumption than the existing schemes for providing data confidentiality for the domain of WSNs.**

***Index Terms*—Confidentiality, SCUR, Wireless Sensor Networks, Rabbit.**

I. Introduction

Wireless networks have been increasing in size and complexity for quite some time now. WSNs have numerous applications including ocean and wildlife monitoring, manufacturing machinery performance monitoring, building safety and earthquake monitoring, and many military applications

Security is still an emerging field, in some general-purpose computing applications it is more developed, while in others such as WSNs it is still underway. Security allows WSNs to be used with confidence. Without security, the use of WSNs in any application domain would result in undesirable consequences [6]. Sensor networks pose unique challenges, as a result of which traditional security techniques cannot be applied. The security of WSNs poses these challenges because of the criticality of the data sensed while having severe constraints on these sensors nodes namely their minimal energy, computational and communicational capabilities. WSNs are exposed to various security threats such as disruption which can be defeated by encrypting the transmitted data. We have taken up this challenge and introduce SCUR, a lightweight security scheme for WSNs applications. We incorporate the Light Weight Rabbit based Encryption/Decryption Module in WSNs referred to as SCUR; a security architecture for providing confidentiality of information in WSNs.

SCUR will cover the data confidentiality needs of all security critical applications in WSNs. For a security solution to achieve high deployment rates in WSNs, it must use minimal resources while providing maximum level of security. Failure to meet either requirement creates a justifiable reason for unsuitability. Our design choices for SCUR are driven by the capabilities and realities of WSNs that make it a perfect choice for WSN security.

The rest of this paper is organized as follows. Section 2 lays down the issue of security in WSNs which paves the way for the proposed SCUR model. Section 3 throws light on the design goals of the proposed scheme. Section 4 explains the security primitives that generally characterize our SCUR. Section 5 talks about the possible key mechanisms that can be employed with SCUR. Section 6 explains the Rabbit Stream Cipher and the appropriateness of utilizing Rabbit for WSN security. Our SCUR is presented in Section 7, with details of its working and analysis of its cost-effectiveness for security in WSNs. Section 8 concludes our research, with our future work presented in Section 9.

II. Security in Wireless Sensor Networks

WSNs consist of small resource-constrained devices called sensor nodes, which consist of an 8-bit processor with memory, sensors, radio unit and power supply that collect data from their surroundings and transmit that data on to a base station as shown in Fig. 1. Base stations have available more computing resources and a larger energy source. The base stations aggregate information from the sensor nodes and then pass them to the external world. The applications for WSNs, range from medical scenarios to agricultural, military and environmental monitoring [1]. A lot of the data in WSNs may be very critical so, security mechanisms are required to ensure secrecy of the sensed data [4].

WSNs require wireless communication links. Wireless media have broadcast nature because of which interception of transmitted data can be very easy for the adversary. So, WSNs must keep the information they are collecting secret from the adversary, so that no private information is accessible to adversaries [5].

Manuscript received February 20, 2007.

R.T. Ruhma Tahir is with Department of Information Security, College of Signals, National University of Sciences and Technology (NUST), Lalkurti, Rawalpindi Cantt, Pakistan. (phone: 0092-051-2113672; fax: 0092-051-9257201; e-mail: ruhma@mcs.edu.pk).

M.Y. Muhammad Younas Javed is the Head of Computer Engineering Department, College of Electrical and Mechanical Engineering, National University of Sciences and Technology (NUST), Peshawar Road, Rawalpindi, Pakistan. (e-mail: myjaved@ceme.edu.pk).

A.A. Attiq Ahmad is the Head of Department of Information Security, College of Signals, National University of Sciences and Technology (NUST), Lalkurti, Rawalpindi Cantt, Pakistan. (e-mail: attiq-mcs@nust.edu.pk).

R. I. Raja Iqbal is with the Engineering Division, College of Signals, National University of Sciences and Technology (NUST), Lalkurti, Rawalpindi Cantt, Pakistan. (e-mail: rajaiqbal@mcs.edu.pk).

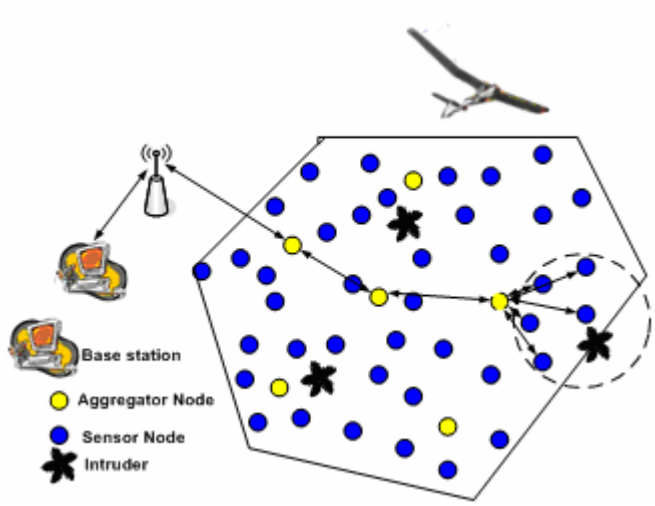

Figure 1. WSNs used in Military Applications

Security mechanisms in WSNs are devised based on a set of principles which are consequences of the hardware restrictions of sensor nodes. The principles that pave the way for our proposed SCUR architecture are described below:

1) The hardware restrictions of sensor nodes suggest that, as communication is three orders of magnitude more expensive than computation, so security protocol which favors computation over communication will be privileged. But the superlative approach would be a security protocol requiring minimum communication as well minimal amounts of computation.

2) Public-key algorithms remain prohibitively expensive on sensor nodes both in terms of storage and energy. Security schemes cannot completely rely on public-key cryptography. So a hybrid approach should be employed; public key cryptography for key establishment and symmetric key cryptography for future communications.

III. Design Goals

In this section, we formalize the three design goals of our proposed SCUR which are described below:

A. Security Goals

A security protocol should satisfy the basic security property of message confidentiality. Confidentiality means keeping information secret from unauthorized parties. It is typically achieved with encryption in order to prevent message recovery. We propose achieving message confidentiality using the Rabbit Stream Cipher which is discussed in more detail in Section VI.

WSNs communicate highly sensitive data so a sensor node should not leak sensor readings to neighboring networks. The standard approach for keeping sensitive data secret is to encrypt the data with a secret key that only intended receivers possess, hence achieving confidentiality.

B. Performance

Due to the extreme resource limitations in WSNs, it is important to carefully adjust the security mechanisms employed in a way that provides reasonable protection while limiting the overhead. The primary reason being that a overly conservative choice of security parameters will consume resources too quickly. A security protocol for WSNs will generally incur increased overhead in the length of messages sent as well as in extra demands on the processor and RAM. So a performance tradeoff has to be set which achieves the required level of security with minimal resource usage.

C. Ease of Use

An important design goal of SCUR is that it should be easy for application programmers to adjust the security performance tradeoffs if their WSNs applications have greater security.

IV. Security Primitives

Considering the threats WSNs are exposed to, we aim to provide privacy of data with our proposed SCUR, in WSNs. This section gives an overview of the security primitives that have been employed in our proposed SCUR:

A. Encryption Schemes

Encryption schemes are classified into symmetric-key encryption schemes and asymmetric encryption schemes. In the symmetric-key encryption scheme, the data is encrypted using the same encryption and decryption key. A secure system, therefore, is required between the sender and recipient of encrypted data for the shared key, as shown in figure 2. The asymmetric key encryption scheme overcomes the issues related to secure key-sharing, as the encryption key and decryption key are different. The recipient provides his/her encryption key to the sender, the sender encrypts the data with this encryption key and sends the data to the recipient, who then decrypts the data with his/her decryption key. However, in public-key encryption schemes, since the encryption key differ from the corresponding decryption key, its encryption and decryption processes are complicated, and is less efficient compared with symmetric-key encryption scheme.

Therefore, it is common to use a symmetric-key encryption scheme to encrypt bulk data and to use an asymmetric key encryption scheme to encrypt the encryption key which was used for bulk data encryption [11]. Therefore we propose the use a of symmetric key encryption scheme for encrypting information between sensor nodes, once the keys have been exchanged using a public key encryption scheme.

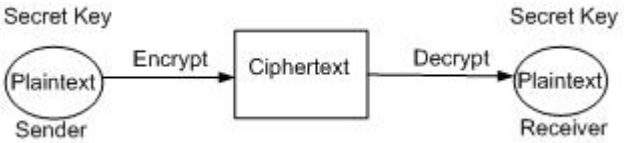

Figure 2. Symmetric Key Encryption

B. Stream Ciphers

Stream ciphers refer to the symmetric-key encryption schemes to encrypt bulk data using a pseudorandom number generator which generates a random data stream. Since stream ciphers have the advantage of being able to encrypt the data bit by the bit it is very suitable for WSNs. Stream ciphers are also superior to block ciphers in terms of efficiency of encryption/decryption processing, and are therefore very suitable for deployment in WSNs.

C. Synchronous Ciphers

These are stream ciphers, in which the state update mechanism is updated independently of the plaintext as shown in figure 3. If a digit is modified due to a transmission error, only this digit will be decrypted erroneously. The transmission error will not affect the decryption of other digits. This is a useful property for WSNs, where bits are often tripped due to channel noise [7].

The fact that the plaintext does not affect the key stream generator makes the security analysis of synchronous stream ciphers radically different from the analysis of non synchronous stream ciphers and block ciphers. For the two latter, the attacker can mount chosen plaintext/ciphertext attacks whereby he can influence the internal state or intermediate variables. For synchronous stream ciphers, this is of no use. The only attack scenario on the cipher during key stream generation that is feasible is a known plaintext attack, which is equivalent to a known key stream attack and a chosen plaintext attack.

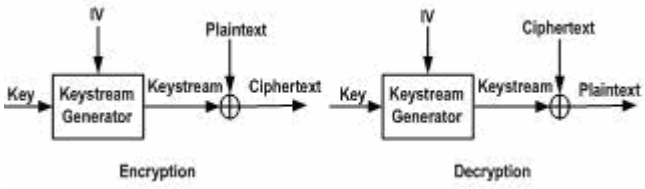

Figure 3. Synchronous Stream Cipher

D. Initialization Vector (IV)

When there is little variation in the set of messages, or when there are chances of identical messages being transmitted, as in the case of WSNs where identical messages may be transmitted repeatedly between two neighboring sensor nodes, initialization vectors are used as side input to the encryption algorithm so that encrypting the same plaintext two times should give two different ciphertexts [3].

V. Keying Mechanisms

Keying mechanisms determine how cryptographic keys are distributed and shared throughout the network. The SCUR protocol is not limited to any particular keying mechanism; any keying mechanism can be used in conjunction with SCUR. Table 1 mentions the positives and negatives associated with different possible keying mechanisms in WSNs. The appropriate keying mechanism for a particular network depends on several factors such as the ease of deployment and the security requirements of applications. A keying mechanism can be selected on the basis of the required level of security and the amount of resources which can be expended for the application. So a tradeoff has to be set between the resources and required security on the basis of the requirements of the application.

The simplest keying mechanism is to use a single network wide symmetric key among the authorized nodes. A network-wide shared key provides a baseline level of security with minimal configuration complexity. Any authorized node can exchange messages with any other authorized node, and all communication is encrypted. However, in network-wide keying if an adversary compromises a single node, the entire network is overtaken by the adversary.

To address the node capture threat, a keying mechanism with finer granularity such as a group based keying mechanism can be employed. Sensor nodes share keys on a per group basis, hence making it a little complex in terms of key distribution between each group but making node capture less effective as compared to network wide key.

A more robust option is for each node to share a key with every other node only if they need to communicate with each other. This provides between better resilience against node capture attacks. But this approach has drawbacks in that key distribution becomes challenging.

Table 1. Comparison of Keying Mechanisms

Keying Mechanism	Positives	Negatives
Network Wide key	Low deployment complexity	Network compromise
Per Group key	Medium complexity in deployment	Group compromise
Per link key	High deployment complexity	Only one link compromise

VI. Rabbit Stream Cipher

Rabbit [10] is a symmetric synchronous stream cipher submitted to the European ECRYPT Stream Cipher Project. It is selected as a Focus Phase 3 candidate as of April 2007. Rabbit is designed with both security and efficiency in mind to satisfy the need for lightweight algorithms, dedicated to hardware environments where the available resources are heavily restricted. It takes a 128-bit secret key and a 64-bit IV (if desired) as input and generates for each iteration an output block of 128 pseudo-random bits from a combination of the internal state bits. Encryption/decryption is done by XOR'ing the pseudo-random data with the plaintext/ciphertext as shown in figure 4. The size of the internal state is 513 bits divided between eight 32-bit state variables, eight 32-bit counters and one counter carry bit. The eight state variables are updated by eight coupled nonlinear functions.

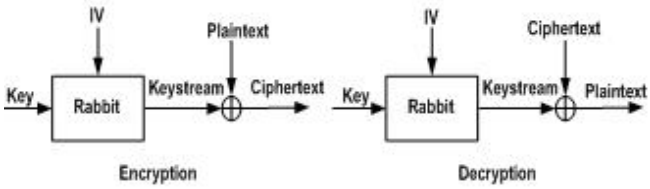

Figure 4. Rabbit Cipher

Rabbit was designed to be faster than commonly used ciphers and to justify a key size of 128 bits for encrypting up to 2^{64} bytes of plaintext. This means that for an attacker who does not know the key, it should not be possible to distinguish up to 2^{64} bytes of cipher output from the output of a truly random generator, using less steps than would be required for an exhaustive key search over 2^{128} keys [10].

A. Key Setup Scheme

The algorithm is initialized by expanding the 128-bit key into both the eight state variables and the eight counters such that there is a one-to-one correspondence between the key and the initial state variables and the initial counters. The key

is divided into eight subkeys and the state and counter variables are initialized from the subkeys. The system is iterated four times, according to the next-state function defined in section 6.3, to diminish correlations between bits in the key and bits in the internal state variables. Finally, the counter variables are modified to prevent recovery of the key by inversion of the counter system.

B. Initialization Vector Scheme

The IV setup scheme works by modifying the counter state as function of the IV. The system is iterated four times according to the next-state function defined in the sub-section C, to make all state bits non-linearly dependent on all IV bits. The modification of the counter by the IV guarantees that all 2^{64} different IVs will lead to unique keystreams.

C. Next-state function

The core of the Rabbit is the Next-state function which can be referred to in [10].Next state function is involved in both key setup and keystream generation. It takes eight counter variables as input and produces a 128 bit keystream block after going through system iteration, counter modification and iteration of the g-function. The good diffusion and non-linearity properties of next-state function prevent against all known attacks.

VII. Design of SCUR

In our design, we aim to improve the security and performance properties without increasing too much energy consumption [8]. We construct an encryption scheme called SCUR which performs confidentiality of each message. Due to the restrictions imposed on WSNs, our major objective in designing a new security model is to minimize cost-effect of the following while maintaining required levels of security [9]:

1) Communication overhead, in case of communicating the encrypted packet.
2) Computation overhead, in securing the network, in order to save sensor's lifetime.
3) Utilized key space.

A. Encryption

Rabbit was designed with both security and efficiency in mind to satisfy the need for lightweight algorithms, dedicated to hardware environments where the available resources are heavily restricted. The simplicity and small size of Rabbit makes it suitable for WSN platform. The exact specification can be referred to [10]. The algorithm is initialized by expanding the 128-bit key into both the eight state variables and the eight counters such that there is a one-to-one correspondence between the key and the initial state variables, and the initial counters. Key and IV is fed into the internal states and going through four iterations of the next update function, to minimize the correlations between the bits in the key, IV and the bits in the internal state variables. Keystream generation uses the same components for further modification of the state and counters variables. In each iteration eight state variables, eight counter variables and a counter carry stored between each iteration supplies input to the Next-state function. A smaller number of rounds translate directly into high rekeying performance.

To send an encrypted packet, a sender encrypts the packet with the agreed symmetric key (EKey) and IV as input to the Rabbit algorithm as shown in Figure 5. When a receiving node gets a decrypted packet, it can perform decryption with the symmetric key Ekey and IV from the packet header, to get the recovered plaintext.

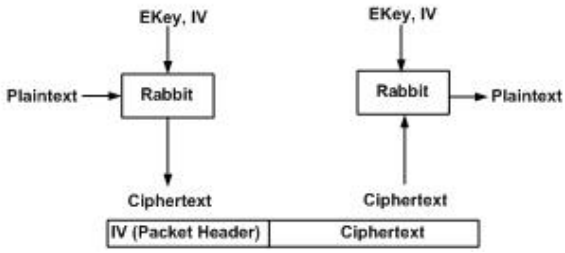

Figure 5. SCUR Design

B. Packet Format

We based SCUR's packet format on the current packet format in TinyOS as in figure 6. The fields common to TinyOS are destination address, active message (AM) type, group and length. We propose appending the source address (Src) and counter (Ctr) field in addition to the standard TinyOS header fields for the SCUR packet format.

Dest	AM	Len	Src	Ctr	Data	CRC

Figure 6. SCUR Packet Format

These fields are unencrypted because the benefits of sending them in the clear generally outweigh the advantages of keeping them secret. To detect transmission errors, TinyOS senders compute a 16-bit cycle redundancy check (CRC) over the packet. The receiver recomputes the CRC during reception and verifies it with the received CRC field. The TinyOS packet format contains a group field which has been replaced by a source address field, so at the cost of just one extra byte, the packet format of SCUR has brought a variation on per link basis. The IV will be computed on the basis of source and destination address, so now there are even less chances of IV repetition. Table 2 below specifies the size of each field which is part of the SCUR packet format in bytes.

Table 2. Packet Field Size

Field Name	Number of Bytes
Destination Address(Dest)	2
Active Message(AM)	1
Length(Len)	1
Source Address(Src)	2
Counter (Ctr)	2
Data	0…29
Cyclic Redundancy Code(CRC)	2

C. IV Format

To further cut down the cost-effect of communication while maintaining security our scheme uses frame header as the IV

for each packet. The IV is taken from the packet header of the radio packet format and sent in clear to the decrypting party. This saves the overhead of communicating a secured IV to the receiving sensor node.

Dest	AM	Len	Src	Ctr

Figure 7. SCUR IV Format

D. Security Analysis

The security of SCUR reduces to the length of the IV. With an 8 byte IV, avoiding repetition is relatively easy. Although SCUR uses an 8 byte IV, we limited ourselves to 4 additional bytes of overhead in the packet to represent the IV. The other 4 bytes of the IV borrow from the existing header fields: the destination address, the AM type, and the length. SCUR partitions the last 4 bytes of the IV into src||ctr, where src is the source address of the sender and ctr is a 16 bit counter starting at 0 [3]. Our format for the last 4 bytes strives to maximize the number of packets each node can send before there is a global repetition of an IV value. The src||ctr format of the last 4 bytes guarantees each node can send at least 2^{16} packets before IV reuse occurs.

We specifically selected Rabbit for SCUR because of its simplified design, hence providing required level of security with minimal resource consumption. The first 6 bytes of the IV, dest||AM||len||src, help prevent information leakage during the unfortunate event of a counter on a node repeating. If a counter value for a particular source address is reused, there is only potential information leakage when the dst||AM||len||src values are exactly the same for both messages. The means both messages were sent to the same group, destination and AM type, and both messages have the same length.

VIII. Conclusion

In this paper, we have proposed a lightweight encryption scheme SCUR, which we expect will outperform existing security protocols for WSNs in terms of security as well as resource consumption. Rabbit is very suitable to safeguard WSNs because of its security properties and simplicity. The main contribution of the proposed SCUR scheme is to ensure confidentiality using Rabbit Encryption Scheme for the domain of WSNs.

IX. Future Work

As of the future work, we will incorporate our SCUR into TinyOS, the light-weight operating system designed specially for small embedded systems with limited resources. This will help us prove through experimental results whether the proposed scheme works up to mark. In future we plan to implement our proposed scheme in real sensor test bed, to determine the flexibility that our SCUR is able to cope with.

References

[1] Mayank Saraogi, “Security in Wireless Sensor Networks” Department of Computer Science, University of Tennessee, Knoxville.
[2] Adrian Perrig, Robert Szewczyk, Victor Wen, David Culler, and J.D. Tygar. “SPINS: Security protocols for Sensor Networks.” In The Seventh Annual International Conference on Mobile Computing and Networking (MobiCom 2001), 2001.
[3] Chris Karlof, Naveen Sastry, David Wagner, “TinySec: A Link Layer Security Architecture for Wireless Sensor Networks”, Proceedings of the 2nd ACM Conference on Embedded Networked Sensor Systems (SenSys 2004), Baltimore, MD, November 2004.
[4] A. Perrig, J. Stankovic, and D. Wagner, “Security in Wireless Sensor Networks,” Commun. ACM, vol. 47, no. 6, pp. 53–57, 2004.
[5] Hill, Jason, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, and Kristofer Pister. “System Architecture directions for Networked Sensors.” In Proceedings of the Ninth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS IX) (November 2000).
[6] Chris Karlof and David Wagner, “Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures”, IEEE International Workshop on Sensor Network Protocols and Applications, 2003.
[7] B. Schneier, "Applied Cryptography”.
[8] Elaine Shi and Adrian Perrig, “Designing Secure Sensor Networks”, IEEE Wireless Communications, pp 38-43 December 2004.
[9] Prasanth Ganesan, Ramnath Venugopalan, Pushkin Peddabachagari, Alexander Dean, Frank Mueller and Mihail Sichitiu, “Analyzing and Modeling Encryption Overhead for Sensor Network Nodes" Workshop on Wireless Sensor Networks and Applications (WSNA '03) with MobiCom'03, Sep 2003.
[10] Martin Boesgaard, Mette Vesterager, Thomas Christensen, Erik Zenner “The Stream Cipher Rabbit” ECRYPT Stream Cipher Project Report 2005/006.
[11] http://www.hitachiappliances.com