

State of Art: Survey of Cryptographic Algorithm in Wireless Sensor Networks

Mayur K. Joshi¹ Asst.Prof. Haresh Rathod²

¹P.G. Student ²Assistant Professor

^{1,2}Department of Computer Engineering

^{1,2}School of Engineering, RK University, Rajkot, Gujarat, India

Abstract— Wireless Sensor Network (WSN) is an emerging technology that shows great promise for various futuristic applications both for mass public and military. The sensing technology combined with processing power and wireless communication makes it lucrative for being exploited in abundance in future. The inclusion of wireless communication technology also incurs various types of security threats. The intent of this paper is to investigate the security related issues and challenges in wireless sensor networks. We identify the security threats, review proposed security mechanisms for wireless sensor networks. We also discuss the holistic view of security for ensuring layered and robust security in wireless sensor networks. Cryptography is the technique to provide the security. Various security algorithms are used to provide the security to the wireless sensor networks. These mechanisms are briefly described in this paper.

Keywords: wireless sensor networks, security, issues, challenges, cryptography, algorithms

I. INTRODUCTION

Advances in wireless communication and electronics have enabled the development of low-cost, low power, multifunctional sensor nodes. These tiny sensor nodes, consisting of sensing, data processing, and communication components, make it possible to deploy Wireless Sensor Networks (WSNs), which represent a significant improvement over traditional wired sensor networks. WSNs can greatly simplify system design and operation, as the environment being monitored does not require the communication or energy infrastructure associated with wired networks [1].

WSNs are expected to be solutions to many applications, such as detecting and tracking the passage of troops and tanks on a battlefield, monitoring environmental pollutants, measuring traffic flows on roads, and tracking the location of personnel in a building. Many sensor networks have mission-critical tasks and thus require that security be considered [2, 3].

Improper use of information or using forged information may cause unwanted information leakage and provide inaccurate results. While some aspects of WSNs are similar to traditional wireless ad hoc networks, important distinctions exist which greatly affect how security is achieved. The differences between sensor networks and ad hoc networks are [4]:

- The number of sensor nodes in a sensor network can be several orders of magnitude higher than the nodes in an ad hoc network.
- Sensor nodes are densely deployed.
- Sensor nodes are prone to failures due to harsh environments and energy constraints.

- The topology of a sensor network changes very frequently due to failures or mobility.
- Sensor nodes are limited in computation, memory, and power resources.
- Sensor nodes may not have global identification.

These differences greatly affect how secure data-transfer schemes are implemented in WSNs. For example, the use of radio transmission, along with the constraints of small size, low cost, and limited energy, make WSNs more susceptible to denial-of-service attacks [5]. Advanced anti-jamming techniques such as frequency-hopping spread spectrum and physical tamper-proofing of nodes are generally impossible in a sensor network due to the requirements of greater design complexity and higher energy consumption [5]. Furthermore, the limited energy and processing power of nodes makes the use of public key cryptography nearly impossible. While the results from recent studies show that public key cryptography might be feasible in sensor networks [6, 7], it remains for the most part infeasible in WSNs. Instead, most security schemes make use of symmetric key cryptography. One thing required in either case is the use of keys for secure communication. Managing key distribution is not unique to WSNs, but again constraints such as small memory capacity make centralized keying techniques impossible. Straight pair wise key sharing between every two nodes in a network does not scale to large networks with tens of thousands of nodes, as the storage requirements is too high. A security scheme in WSNs must provide efficient key distribution while maintaining the ability for communication between all relevant nodes. In addition to key distribution, secure routing protocols must be considered. These protocols are concerned with how a node sends messages to other nodes or a base station. A key challenge is that of authenticated broadcast. Existing authenticated broadcast methods often rely on public key cryptography and include high computational overhead making them infeasible in WSNs.

II. CONSTRAINTS AND SECURITY REQUIREMENTS

A. Constraints:

Individual sensor nodes in a WSN are inherently resource constrained. They have limited processing capability, storage capacity, and communication bandwidth. Each of these limitations is due in part to the two greatest constraints limited energy and physical size. Table 1 shows several currently available sensor node platforms. [5] The design of security services in WSNs must consider the hardware constraints of the sensor nodes:

1) Energy:

Energy consumption in sensor nodes can be categorized into three parts:

- Energy for the sensor transducer

- Energy for communication among sensor nodes
- Energy for microprocessor computation

The study found that each bit transmitted in WSNs consumes about as much power as executing 800–1000 instructions. Thus, communication is more costly than computation in WSNs. Any message expansion caused by security mechanisms comes at a significant cost. Further, higher security levels in WSNs usually correspond to more energy consumption for cryptographic functions. Thus, WSNs can be divided into different security levels, depending on energy cost.

2) *Computation:*

The embedded processors in sensor nodes are generally not as powerful as those in nodes of a wired or ad hoc network. As such, complex cryptographic algorithms cannot be used in WSNs.

3) *Memory:*

Memory in a sensor node usually includes flash memory and RAM. Flash memory is used for storing downloaded application code and RAM is used for storing application programs, sensor data, and intermediate computations. There is usually not enough space to run complicated algorithms after loading OS and application code. In the SmartDust project, for example, TinyOS consumes about 3500 bytes of instruction memory, leaving only 4500 bytes for security and applications. This makes it impractical to use the majority of current security algorithms [8]. With an Intel Mote, the situation is slightly improved, but still far from meeting the requirements of many algorithms.

4) *Transmission Range:*

The communication range of sensor nodes is limited both technically and by the need to conserve energy. The actual range achieved from a given transmission signal strength is dependent on various environmental factors such as weather and terrain.

B. *Security Requirements*

The goal of security services in WSNs is to protect the information and resources from attacks and misbehaviour. The security requirements in WSNs include [6]:

- Availability, which ensures that the desired network services are available even in the presence of denial-of-service attacks
- Authorization, which ensures that only authorized sensors can be involved in providing information to network services
- Authentication, which ensures that the communication from one node to another node is genuine, that is, a malicious node cannot masquerade as a trusted network node
- Confidentiality, which ensures that a given message cannot be understood by anyone other than the desired recipients
- Integrity, which ensures that a message sent from one node to another is not modified by malicious intermediate nodes
- Nonrepudiation, which denotes that a node cannot deny sending a message it has previously sent
- Freshness, which implies that the data is recent and ensures that no adversary can replay old messages

- Moreover, as new sensors are deployed and old sensors fail, we suggest that forward and backward secrecy should also be considered:

- Forward secrecy: a sensor should not be able to read any future messages after it leaves the network.

- Backward secrecy: a joining sensor should not be able to read any previously transmitted message.

The security services in WSNs are usually centered on cryptography. However, due to the constraints in WSNs, many already existing secure algorithms are not practical for use. We discuss this problem in the section “Cryptography in WSNs” below.

III. CRYPTOGRAPHY IN WSNs

Before Selecting the most appropriate cryptographic method is vital in WSNs because all security services are ensured by cryptography. Cryptographic methods used in WSNs should meet the constraints of sensor nodes and be evaluated by code size, data size, processing time, and power consumption. In this section, we focus on the selection of cryptography in WSNs. Cryptography Ciphers are described below.

Cryptographic ciphers often provide the most basic security requirements such as confidentiality, authenticity and integrity checking in any system. However, not all cryptographic ciphers that are suitable for conventional networks will also be suitable for WSNs. This chapter discusses security primitives through the use of cryptographic ciphers and their applicability to the ultra-low power WSN environment. The background of block ciphers as well as modes of operation are investigated and discussed here. The only stream cipher implemented in this paper, RC4, is also discussed here.

A. *TEA:*

TEA (Tiny Encryption Algorithm) [3] and its related variants (XTEA, Block TEA, XXTEA) are symmetric key block ciphers designed for modern 32-bit word architecture. The emphasis of TEA is on small code size and easy implementation with typically few lines of codes. It uses a large number of iterations rather than a complicated algorithm. All TEA and its variants are based on the Feistel structure, every TEA cycle consists of two Feistel rounds (Figure 4.1). TEA and XTEA operate on two 32-bit words as a 64-bit data blocks with a 128-bit key, therefore all operations are done in 32-bit words. Block TEA and XXTEA operate on variable-length blocks of arbitrary multiples of 32 bits size. The advantage of Block and XXTEA is that it eliminates the need for using a mode of operation (CBC, OFB, CFB, OCB etc.) on messages larger than one block. i.e. they can be applied directly to a complete message.

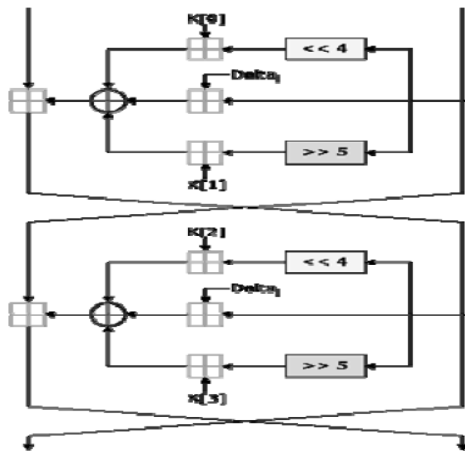


Fig. 1: One TEA cycle (two Feistel rounds) [3]

B. Cryptanalysis of TEA:

TEA suffers from two types of cryptanalysis, the related-key [9] and equivalent-key [8] attacks. The equivalent-key attack is targeted at TEA’s extremely simple key-schedule. This results in the problem that when flipping the most significant bits of the first two 32-bit words of the key, the encryption will not be affected. This attack has allowed hackers to successfully run Linux operating system on the Microsoft’s Xbox gaming console. The best related-key attack on TEA requires 223 chosen plaintexts and 232 computation time to recover the key. XTEA is proposed by TEA designers to prevent weaknesses found in TEA. The best attack so far on XTEA is a related-key differential attack on 27 rounds [10]. This attack requires 220.5 chosen plaintexts and has a time complexity of 27-round XTEA encryptions. For minimum of drag. Here inlet velocity is taken as 40 m/s and k-e turbulence model is selected for capturing turbulent motion of vortices [5].

C. SAFER K-64:

SAFER K-64 [4] (stands for Secure And Fast Encryption Routine with a Key of length 64 bits) is a non-proprietary secret (symmetric) key block cipher. The block length is 64 bits (8 bytes) and only byte operations are used for key scheduling, encryption and decryption. The encryption structure of SAFER K-64 is shown in the following figure. The encryption/decryption algorithm consists of r rounds, typically 6 rounds are recommended. Each round (shown in Figure 4.4) requires two 64-bit (8 bytes) subkeys and the output transformation needs one 64-bit subkey. In total $2r + 1$ subkeys are needed, which is derived from the user-selected secret key “K1”. The output transformation involves byte XOR and byte addition (modulo 256) of the last subkey ($K_{2r + 1}$) with output from the r -th round. The decryption structure is similar to the encryption structure except that the output transformation now becomes the input transformation and is executed first. The subkeys in the decryption structure are also used in a reversed order.

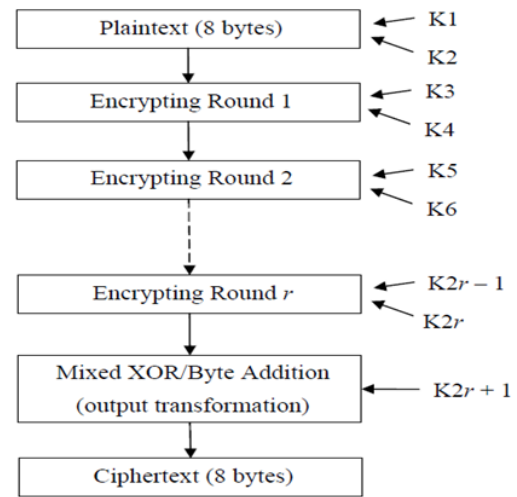


Fig. 2: Encryption structure of SAFER K-64

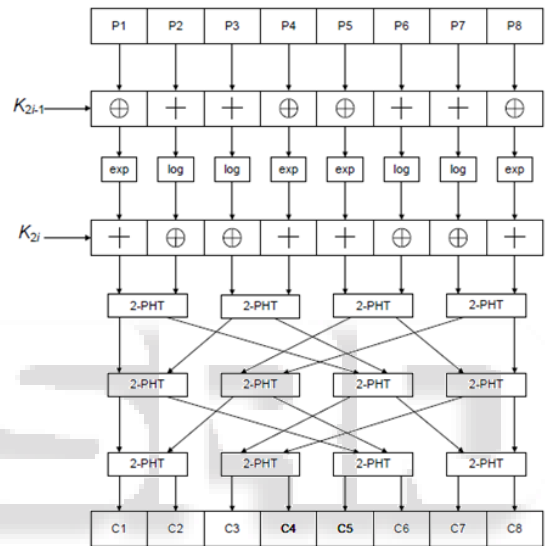


Fig. 3: One encryption round structure of SAFER K-64

D. DES (Data Encryption Standard):

DES is block cipher. A block cipher is a function which maps-it plaintext blocks to cipher-text blocks; is called the block length. It may be viewed as a simple substitution cipher with large character size. The function is parameterized by a bit key, taking values from a subset (the key space) of the set of all -bit vectors. It is generally assumed that the key is chosen at random. Use of plaintext and cipher text blocks of equal size avoids data expansion. To allow unique decryption, the encryption function must be one-to-one (i.e., invert- idle).

E. 3DES:

- The triple-DES ciphers use three iterations of DES
- The three-key variant is defined by
- $DES_3(K_1 \ k \ K_2 \ k \ K_3; \ M) = DES(K_3; DES_1(K_2; DES(K_1; M)))$

F. TREYFER:

TREYFER is a 64-bit block cipher with 64-bit symmetric key and is proposed by Yuval [4]. It is aimed at applications with extremely limited resources, e.g. smart card and is designed to be very compact (less than 50 bytes of code on an 8051 microcontroller with assembler language). It can be

executed on a very constrained architecture, for example an 8051 microcontroller with typically 1 KB flash EPROM, 64 bytes RAM, 128 bytes EPROM and a peak instruction rate of 1 MHz. TREYFER is designed to use only byte operations and requires fixed bit rotations and modulo 256 additions. The algorithm is as

```
for (r = 0; r < NumRounds; r++){
text[8] = text[0];
for(i = 0; i < 8; i++){
text[i+1] = (text[i+1] + Sbox[(key[i]+text[i])%256]) <<< 1;
//rotate 1 left
Text[0] = text[8];
}
```

In the above pseudo code, “text” represents the 8-byte plaintext, “Sbox” is the 256×8-bit (256 bytes) S-box chosen at random, and “NumRounds” is the number of rounds executed in TREYFER, which is typically 32. One of the motivations of the TREYFER design is the use of a large number of rounds (32) to thwart any possible practical attacks in spite of the simple round function design. The S-box was suggested by the author to be taken from another place in the memory running non-cryptographic codes. In this way there is no need to explicitly define a 256-byte S-box and thus code space is saved.

G. AES:

AES (Advanced Encryption Standard) was published by NIST (National Institute of Standards and Technology) to replace DES (Data Encryption Standard). Out of the many candidates for AES, the Rijndael cipher was eventually selected to become the new AES [8]. AES is a symmetric key block cipher with a block size of 128 bits and three key size alternatives of 128, 192, or 256 bits. Unlike many conventional symmetric key block ciphers, AES does not use the Feistel structure, where typically half of the data block is used to modify the other half of the data block before the two halves are swapped in the next round. AES processes the entire data block (128 bits) in parallel during each round. AES typically has 10 rounds; each round has four different stages, one of permutation and three of substitution. The encryption and decryption functions in AES differ. The encryption and decryption speed does not vary significantly, however, the key setup performance is slower for decryption and requires more memory than for encryption. All AES operations can be byte operations allowing it to be efficiently implemented on 8-bit processors. Its operations can also be defined in 32-bit words for efficient implementation on 32-bit processors [2]. Although AES has been well studied over the years and proven to be secure, it does not seem to be suitable for the platform which this paper is based on, or in many other WSN environments. One of the main reasons is that although AES has been designed for lowend 8-bit microcontroller, its baseline version still uses over 800 bytes of look-up tables. A speed optimized AES version, which runs about 100 times faster, uses over 10 KB of lookup tables. This memory requirement is not acceptable to many sensor node platforms. For example, the microcontroller MSP430F1232 used in the sensor nodes (TinyMote) of this paper has only 8KB of flash code memory in total. Apart from the large code size, AES also requires large RAM space to store expanded subkeys, typically larger than 156

bytes. Furthermore, because of the small packet size of WSN, a cipher with 128-bit (16 bytes) block size may not be very efficient. For example the last cipher call may only need to encrypt the last two bytes of the data packet, since the cipher uses 16-byte block.

H. RC5:

RC5 is a symmetric encryption algorithm with a block size of 32, 64, or 128 bits [2]. The key length ranges from 0 to 2040 bits. RC5 encrypts two-word blocks, for example a 32-bit block has a word size of 16-bit. The maximum number of RC5 rounds is 255, but typically 12 rounds encryption/decryption algorithm is suggested. RC5 has a simple structure similar to a Feistel structure. Instead of half of a block being updated as in the classic Feistel structure, both halves are updated in each RC5 round [6]. RC5 uses only three primitive operations: modulo 2 addition/subtraction (n is the word size), XOR, and circular rotation. The encryption/decryption algorithm is very simple and can be implemented in few lines of codes. These characteristics make RC5 suitable for both hardware and software implementations. RC5 requires complex key expansion operations on user-selected secret keys. The number of sub keys that are needed is $2r + 2$, where r is the total number of rounds. RC5 has also been around for some years and appears to be secure. Although it was designed to be of small size for efficient software and hardware implementation, its smallest word size is still 16-bit. The key setup operations have been shown to be very time consuming [5] and also require a relatively large amount of RAM space to store the expanded subkeys [11]. Furthermore, RC5 rotation operations are data-dependent, meaning that it has to rotate variable number of bits and often requires a large number of bit rotations. This large number of bit rotations is especially time consuming for processors with a word size smaller than that of the RC5 word size (e.g. 16-bit RC5 word on an 8-bit processor). Law et. al. [5] have compared RC5, AES and several other block ciphers on the same family of microcontrollers (TI MSP430) as the one used in this paper. These comparisons have shown that RC5 is not the most efficient cipher nor does it have the smallest code size..

I. Quantitative Analysis of All Algorithm

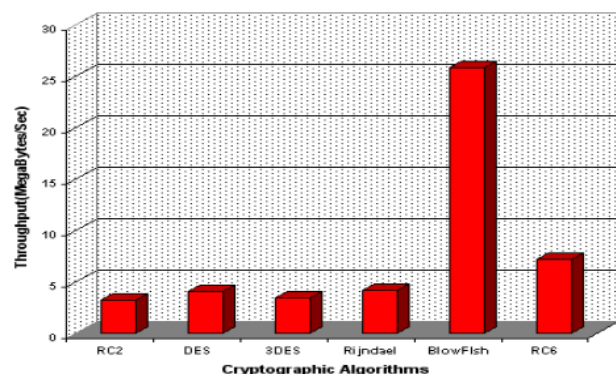


Fig. 4: Throughput of each encryption algorithm

Experimental results for this comparison point are shown Figure 4 at encryption stage. The results show the superiority of Blowfish algorithm over other algorithms in terms of the processing time. Another point can be noticed here; that RC6 requires less time than all algorithms except

Blowfish. A third point can be noticed here; that AES has an advantage over other 3DES, DES and RC2 in terms of time consumption and throughput. A fourth point can be noticed here; that 3DES has low performance in terms of power consumption and throughput when compared with DES. It always requires more time than DES because of its triple phase encryption characteristics. Finally, it is found that RC2 has low performance and low throughput when compared with other algorithms in spite of the small key size used.

IV. CONCLUSION

The well-known AES and the WSN-popular RC5 block ciphers have been shown to be not very suitable for WSN. The block cipher SAFER K-64 has been investigated for the first time for its applicability in WSN. Compared to other block ciphers investigated for WSN environment, SAFER K-64 achieves the best performance in CPU usage known to the author. It, however, requires slightly more RAM. XTEA requires a fairly small amount of flash/ROM memory and no RAM is needed for the subkeys setup. Even though XTEA is designed for a 32-bit architecture, it performed well on the 16-bit MSP430 platform and outperformed both AES and RC5 on the same MSP430 platform. Although TREYFER requires the least flash memory and also does not need RAM for the subkey setup, it requires a considerable number of CPU cycles.

REFERENCES

- [1] A. Perrig, R. Szewczyk, V. Wen, D. Culler and J.D.Tygar, "SPINS: Security Protocols for Sensor Networks", Proceedings of 7th Annual International Conference on Mobile Computing and Networks (Mobicom), pp. 189-199, Rome, Italy, 2001.
- [2] Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone, Handbook of Applied Cryptography, 5th ed., CRC Press, 1996.
- [3] C. Karlof and D. Wagner, "Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures", Proceedings of the 1st IEEE International Workshop on Sensor Network Protocols and Applications (SPNA), pp. 113-127, Anchorage, USA, May 2003.
- [4] P. Ganesan, R. Venugopalan, P. Peddabachagari et. al., "Analyzing and Modeling Encryption Overhead for Sensor Network Nodes", Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications, San Diego, USA, 2003.
- [5] Y. Law, S. Dulman, S. Etalle et. al., "Assessing Security-Critical Energy-Efficient Sensor Networks", Department of Computer Science, University of Twente, Tech.Rep. TR-CTIT-02-18, 2002.
- [6] C. Karlof, N. Sastry and D. Wagner, "TinySec: A Link Layer Security Architecture for Wireless Sensor Networks", Proceedings of the 2nd ACM Conference on Embedded Networked Sensor System (SenSys), vol. 47 issue 6, Baltimore, USA, November 2004.
- [7] S. Mahlknecht, "Energy-Self-Sufficient Wireless Sensor Networks for the Home and Building Environment", Doctor's thesis, Technical University of Vienna, 2004.
- [8] H. Y. Yang, H. Luo, F. Ye et. al., "Security in Mobile Ad Hoc Networks: Challenges and Solutions", IEEE Wireless Communications Magazine, pp. 38-47, February 2004.
- [9] E. Shi and A. Perrig, "Designing Secure Sensor Networks", IEEE Wireless Communications Magazine, pp. 38-43, December 2004.
- [10] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cayirci, "A Survey on Sensor Networks", IEEE Communications Magazine, pp. 102-114, August 2002.
- [11] Y. Chun Hu, A. Perrig, and D. Johnson, "Ariadne: A Secure On-demand Routing Protocol for Ad Hoc Networks", Proceedings of 8th Annual International Conference on Mobile Computing and Networks (Mobicom), Atlanta, USA, September 2002.