ENERGY EFFICIENCY IN SECURE WIRELESS SENSOR NETWORKS

XUEYING ZHANG

Energy Efficiency in Secure Wireless Sensor Networks

By

© Xueying Zhang
Master of Engineering

A thesis submitted to the
School of Graduate Studies
in partial fulfillment of the
requirements for the degree of
Master of Engineering

Faculty of Engineering and Applied Science
Memorial University of Newfoundland

July 2010

St. John's Newfoundland Canada

Abstract

Cryptography plays an important role in protecting the data security of wireless sensor networks (WSNs). However, it is greatly constrained by the limited energy supply of a sensor node. In this thesis, we focus on the energy efficiency of secure communication in wireless sensor networks (WSNs). Our research considers both the cryptographic algorithms and the cryptographic schemes when link layer security is applied to a WSN. By implementing symmetric key ciphers in both software and hardware, we evaluate the computational energy cost considering both the algorithm characteristics and channel quality. We further explore different factors that affect the communication energy cost of link layer cryptographic schemes, such as the payload size, the mode of operation, the initialization vector distribution, and the communication channel quality. We evaluate the energy efficiency of different cryptographic schemes for data transmission by developing an analysis model that is constructed considering various factors affecting both the computational cost and communication cost. Its appropriateness is further verified and supported by simulation results. We also investigate the energy cost of session key establishment in a WSN. In conclusion, we recommend using a lightweight block cipher, byte-wise substitution permutation network (BSPN), along with the ciphertext feedback scheme for wireless sensor networks to achieve security and better energy efficiency. To enhance the data security further in a WSN, we also suggest using a key distribution protocol based on symmetric key cryptography, which can lead to a robust system with acceptable additional energy consumption.

Acknowledgements

Foremost, I owe my deepest gratitude to my supervisors Dr. Howard M. Heys and Dr. Cheng Li, for their guidance and encouragement throughout my graduate study. I could not finish this thesis without their patience, motivation, enthusiasm, and immense knowledge.

I am also heartily thankful to Dr. Dobre Octovia, Dr. Yuanzhu Peter Chen, Dr. Ramachandran Venkatesan, and Dr. Mohamed Hossam Ahmed, for their teaching the graduate courses and broadening my scope of knowledge. I also want to thank Dr. Lihong Zhang, for his understanding, support and encouraging me to pursue my interests in research.

I wish to thank all my fellow labmates of CERL in Memorial University for their generous support. Special thanks to Cheng Wang, Xi Chen, Abdulah Abdulah Zadeh, Rui He, Yuting Liu, Aslinda Hasan, Hala Mostafa and Muthu Gandi, for their precious friendship and unreserved sharing of knowledge.

I am indebted to my parents and my husband, Tao Bian. Thank you for your unconditional love and support.

Last but not the least, I offer my regards and blessings to all of those who supported me in any respect during the completion of the thesis.

Contents

List of Figures

List of Tables

List of Abbreviations

ACK	Acknowledgement
AES	Advanced Encryption Standard
AH	Authentication Header
AP	Access Point
BSC	Binary Symmetric Channel
BER	Bit Error Rate
BSPN	Byte-oriented Substitution-Permutation Network
CBC	Cipher Block Chaining
CCM	Counter with CCB mode
CFB	Cipher Feedback
CHAP	Challenge-handshake Authentication Protocol
Codebook	OCB
CCM	Counter with CBC-MAC
CDA	Concealed Data Aggregation
DES	Data Encryption Standard
DoS	Denial of Service
EAP	Extensible Authentication Protocol
ECC	Elliptic Curve Cryptography
ESP	Encapsulating Security Payload
FPGA	Field-Programmable Gate Array
GCM	Galois/counter mode

GDI	Great Duck Island
GF	Galois Field
IV	Initialization Vector
IPsec	Internet Protocol Security
ISO	International Organization for Standardization
KSA	Key-Scheduling Algorithm
LEAP	Localized Encryption and Authentication Protocol
LUT	Lookup Table
MAC	Message Authentication Code
MIC	Message Integrity Code
μTESLA	Micro Timed Efficient Streaming Loss-tolerant Authentication
MANET	Mobile Ad-hoc Network
MPDU	MAC Protocol Data Unit
MPR	Multipoint Relay
OLSR	Optimized Link State Routing
PDA	Personal Digital Assistant
PPP	Point-to-Point Protocol
PAP	Password Authentication Protocol
PH	Private Homomorphism
PGP	Pretty Good Privacy
PRGA	Pseudorandom Generation Algorithm
PSK	Pre-shared Key
RSA	Rivest-Shamir-Adleman
RTL	Register Transfer Level
SNEP	Secure Network Encryption Protocol
SDA	Secure Data Aggregation
SDAP	Secure hop-by-hop Data Aggregation Protocol

SIA	Secure Information Aggregation
SET	Secure Electronic Transaction
SPN	Substitution Permutation Network
SHDA	Secure Hierarchical Data Aggregation
SRDA	Secure Reference-Based Data Aggregation
SSH	Secure Shell
SSL	Secure Socket Layer
SSP	Secure Simple Pairing
TKIP	Temporal Key Integrity Protocol
TSL	Transport Layer Security
UDP	User Datagram Protocol
VCD	Value Change Dump
WEP	Wireless Equivalent Privacy
WLAN	Wireless Local Area Network
WPAN	Wireless Personal Area Network
WPA	Wi-Fi Protected Access
WSN	Wireless Sensor Network
WMSN	Wireless Multimedia Sensor Network
WBSN	Wireless Body Sensor Network
ZC	Zigbee Coordinator
ZR	Zigbee Router
ZED	Zigbee End Device

List of Symbols

η	The energy efficiency, defined by the average number of data bits successfully transmitted per Joule by the sensor node.
γ	The number of previous packets whose ciphertext is used for the IV
AM	Active message handler type
$\lceil . \rceil$	denotes the ceiling operator
b	For a block cipher, *b* is the block size. While for a stream cipher, *b* is the keystream block size, which is the amount of keystream produced at one time.
b_{mac}	The block size of a block cipher which is used in CBC mode to produce MAC
C_{MCU}	The number of operation cycles being used for processing
C_{crc}	The number of clock cycles required for CRC
C_{enc}	The number of clock cycles required to encrypt one block
DEST	Destination address of the receiver
E_{enc}	The energy cost of processing encryption of one packet
E_{mac}	The energy cost of processing MAC of one packet
E_{rx}	The energy cost of receiving one packet
E_{tx}	The energy cost of transmitting one packet
f_{cpu}	the CPU's frequency
I_{rx}	The current in receiving mode
I_{tx}	The current in transmitting mode
ID_A	The ID of an aggregator
ID_N	The ID of a sensor node
K_{sess}	The session key, determined by the block cipher

K	The period of the IV packet, i.e., the number of data packets sent for each IV packet
LEN	Size of the packet
n	the number of the transmitted data packets
N_1, N_2, N_3	The nonce
N_{pkt}	The size of the packet
N_{pld}	The payload size in bits
N_{ptext}	The number of plaintext bits
N_{ss}	Start symbol used for medium access.
N_{hd}	The size of packet header in bits, including DEST, AM, LEN
N_{mac}	Message authentication code
N_{ivpkt}	The size of IV packet in bits
N_{ackpkt}	The size of ACK packet in bits
p_e	the probability of error for each bit
P_o	The probability that a received packet has no errors
P_{cpu}	the CPU's power
P_{data}	The probability that a data packet is correctly received
P_{valid}	The probability that the data packet is correctly decrypted
P_{iv}	The probability that the IV packet is correctly received
P_{ack}	The probability that the ACK packet is correctly received
R	The bit rate of transmission
T_{cpu}	The CPU's operation time
T	The period of session key updated.
U	The voltage of the MCU chip

Chapter 1

INTRODUCTION

Nowadays, wireless sensor networks (WSNs) provide a link between the real world application and the widely used Internet, by se nsing, monitoring, and managing our physical world. Although this type of network was initially designed for military usage, it has been become an important part of our daily life. For example, in the field of environmental monitoring and protection, sensor networks can be used to sense potential disasters such as earthquake and forest fire, and monitor the habit of wildlife for ecosystem research and protection. In the field of human healthcare, it can be used to detect abnormal health status of patients. In the field of industry automation, it can be used to monitor areas that are not accessible human beings such as some industrial processes or disaster zones. It is worth noticing that no applications can achieve the best effect unless the security of transferred information can be well protected. However, the limited energy supply of a sensor node makes it impractical to use traditional security protocols in a WSN. Hence, security in WSNs becomes a challenging issue and its protocols need to be specifically designed [1].

In this chapter, we first review the state-of-the-art work in WSNs and introduce the most important component in achieving security: cryptography. After that, we show the motivation of our research and present the organization of this thesis.

1.1 Wireless Sensor Networks

A sensor node is a small sized device, capable of sensing parameters of its environments, such as sound, light, temperature, and so on. A large amount of these sensors can work cooperatively to form a network, referred to as a wireless sensor network, by transmitting data through wireless channels. Recent developments in sensing, wireless communications, and networking technologies make it possible to sense a broad range of information, from small amounts of information such as moisture, light and smoke to larger sized information such as voice and image. In this section, we review different WSNs from the application context perspective.

In WSNs, a transceiver is an important component, in charge of sending and receiving data. The transceiver of most sensor nodes operates on narrowband radios (at frequencies such as 433MHz and 868MHz) so that the sensor node can have a longer life span by satisfying the balance between low data rate and small energy consumption [2]. Some high-end sensor nodes may use wideband radios (at 2.4GHz), such as Bluetooth or IEEE 802.15.4 (MAC and physical layer of Zigbee), to achieve high throughput for certain application environments. With the development of microelectronic technologies, new generation of wireless sensors tend to use Wi-Fi technology due to the availability of low power integrated circuits. Nevertheless, the main trend of wireless sensor network development still follows the original path and typical sensor products still operate on narrowband radios.

1.1.1 Classical WSN with Low Throughput

We use the term *classical WSN* to refer to a WSN using typical sensor products, which can sense and transmit small amounts of data at each time. For example, in environmental monitoring, certain types of physical and chemical parameters for the environment are periodically sensed and transmitted out to the base station. Habitat monitoring on Great Duck Island (GDI) is a typical project for this type of sensor network application [3]. The goal is to study the relationship between the environment and duck's habitat based on the sensing data. Hence, their normal lives will not be disturbed. In this project, a popular type of sensor device called Mica is utilized and the architecture of the monitoring system is shown in Fig. 1.1. In this figure, a sensor node functions as a transmit-only device and transmits small-sized packets (with payload sizes less than 30 bytes). Each node has a limited energy capability since they are only equipped with two AA batteries. In each region of the monitored area, there is a node called a gateway, which has more energy supply and stronger processing capability. The gateway node receives messages from sensor nodes within its control area, performs aggregation algorithms on the collected data and sends aggregation results to the base station. Finally, through the Internet, the researchers can retrieve the data for further analysis.

Similar architectures can also be used in the monitoring of seismic parameters, forest fire indicators, contaminant transport, etc.. Usually, the sensor node is low cost and has low throughput for this type of WSN. The small-sized packet occupies the bandwidth discontinuously and does not need to be transmitted with a high speed. However, energy limitation is a critical problem in the sensor node due to the non-rechargeable battery inside. Thus lightweight protocols are needed to extend the sensor node's life span as long as possible.

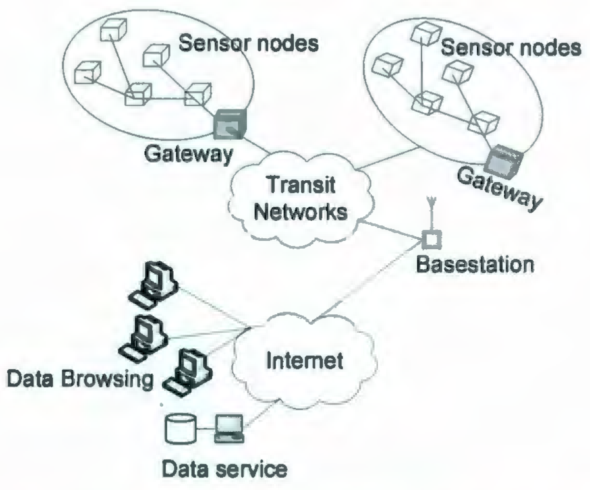


Fig. 1.1. System architecture for habitat monitoring on GDI.

1.1.2 WMSN with High Throughput

The development of the electronic technology has greatly decreased the cost of CMOS cameras and microphones and has led to the emergence of wireless multimedia sensor networks (WMSN) [4]. In this kind of network, sensor nodes are equipped with certain modules to capture video or audio information to be transmitted through a wireless link. The abundant information contained in video and audio streams makes the WMSN a suitable candidate for such application. Therefore, WMSN has become very attractive since it can satisfy many application requirements and improve the performance of current services such as surveillance and traffic control. The architecture of typical architectures of WMSN is illustrated in Fig. 1.2. There are ma ny different types of architecture in WMSN [5], such as (a) single-tier flat architecture with homogeneous sensors, distributed processing, and centralized storage; (b) single-tier clustered architecture with heterogeneous sensors, centralized processing, and centralized storage; and (c) multi-tier architecture with heterogeneous sensors, distributed processing, and distributed storage.

Compared to classical WSNs, a WMSN includes sensor devices with relatively higher cost and possesses higher throughput since a large amount of data needs to be transmitted in real time. Although sensor nodes may be rechargeable, energy is still a critical issue since a large volume of data needs to be transmitted frequently. Efficient data compression algorithms and effective communication protocols are important factors affecting the performance of this kind of network.

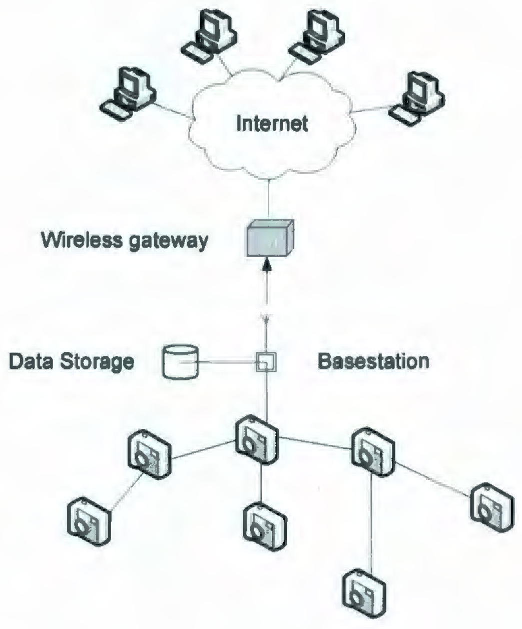


Fig. 1.2 Architecture of WMSN.

1.1.3 WBSN with Hybrid Needs

Wireless body sensor networks (WBSNs) (also called medical sensor networks) are attracting much attention nowadays due to their significant contribution to healthcare and to people's daily lives. WBSNs enable the continuous monitoring of people's health status via various types of sensors for medical sensing. Among those sensors, a special one called cluster head is used to gather data from all the other medical sensors on the body and send them out through wireless networks periodically. If there are abnormal symptoms, an emergency alert will be sent to a health-monitoring centre, or to the doctor's clinic. There are two typical scenarios in WBSN: (1) lon g term healthcare monitoring during a person's daily life (2) real-time monitoring in hospitals. The first scenario is appropriate for people who are susceptible to illness due to aging or chronic diseases, while the second scenario is applicable to patients who are already ill and are hospitalized for a period.

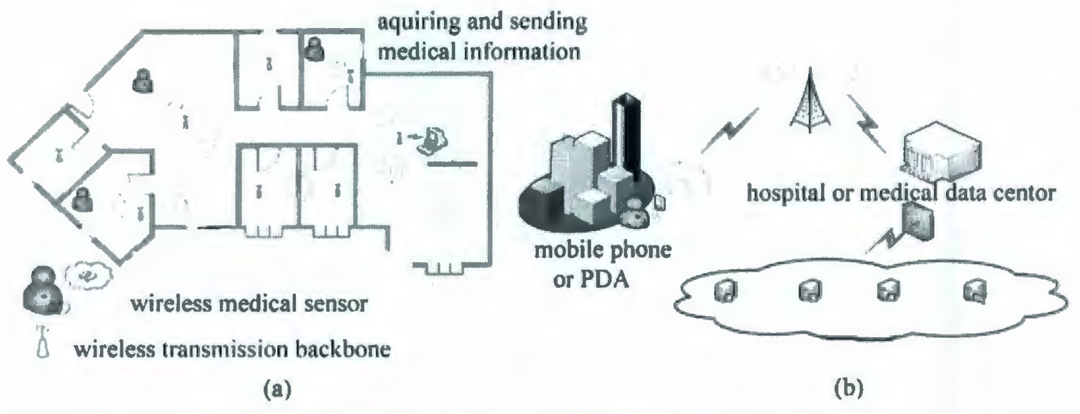


Fig. 1.3. Scenario of long-term healthcare monitoring [6].

The long-term monitoring scenario is shown in Fig. 1.3, including (a) home mode and (b) nomadic mode [6]. The home mode is active when the person to be monitored is at home, where the computer functions as a base station, i.e., gathering various types of data from the person and the surrounding environment through wireless communications and finally sent out through the Internet. The nomadic mode is activated when the person is outside the home. Since no computer functions as a base station, a mobile phone or personal digital assistant (PDA) will be used to collect the data and send it out through wireless networks.

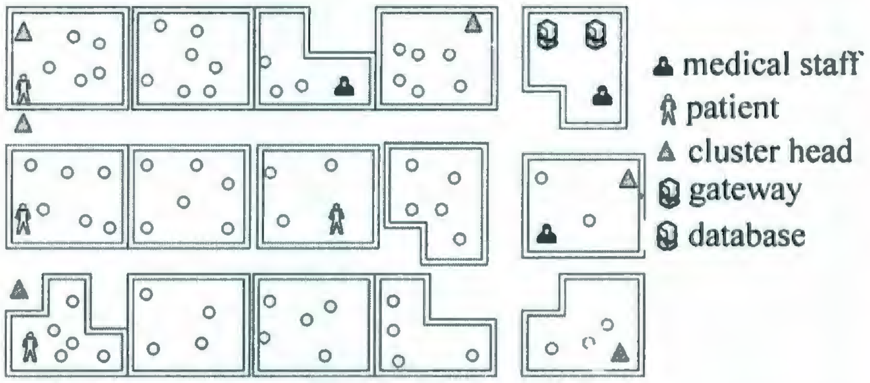


Fig. 1.4. Scenario of monitoring in hospitals [7].

The scenario of monitoring in hospitals is illustrated in Fig. 1.4, where wireless nodes installed in hospitals will gather medical information from patients and send to the cluster head [7]. The utilization of medical sensors frees patients in the hospital from being constrained in bed all day and allows them moving as they want. Large amounts of sensor nodes with special functions are deployed densely all over the hospital, which collect and analyze data from the patients in their sensing range. If the data is beyond the normal range, the data will be immediately sent to the monitoring center along with the accurate location of the patient, and in certain urgent cases, emergency alarming will be triggered as well. Otherwise, the data will be stored as a common record and will be sent to the monitoring centre when the network is available.

Compared with other applications in wireless sensor networks, a medical application is especially challenging due to its application environment, which combines characteristics of classical WSN and WMSN. It is essential to give more consideration on the data diversity and information security. Various data with different packet lengths and formats are involved and there are multiple types of sensors used in the network. These sensors monitor parameters of various objects, including patient parameters such as heart rate, oximetry, blood pressure, environmental parameters such as temperature and moisture, or even a real-time image of a patient using a camera.

1.2 Cryptography

Cryptography is the science of information security, which studies the technique of protecting information from being known and/or altered by unauthorized people [8]. The data is transformed with a key (additional secret information shared between two communicating parties) so that it can only be accessed by a party with authorization. The process of converting the original data to a secret message is called encryption; correspondingly, the inverse process is called decryption. The security of a cryptography algorithm is assessed based on the assumption that the structure of the algorithm is known while the key is not known. Cryptography can be mainly categorized into two types: symmetric key cryptography and asymmetric key cryptography (also called public key cryptography).

In this section, we introduce the basic knowledge of cryptography and the objectives to be achieved by using cryptography.

1.2.1 Different Types of Cryptography

1) Symmetric Key Cryptography

Symmetric key cryptography is developed from classical ciphers in ancient times, which typically process data through a relatively simple way, such as substituting an alphabet character with another one or transposing a character's position. Symmetric key cryptography focuses on the structure of iteratively applied simple cryptographic operations using the same key for encryption and decryption. A simplified model for symmetric key cryptography is illustrated in Fig. 1.5, where the input data to be encrypted is called plaintext, and the corresponding encryption result is called ciphertext. Only the authorized receiver can understand the message by decryption using the key shared with the sender so that the message is protected from being known by others.

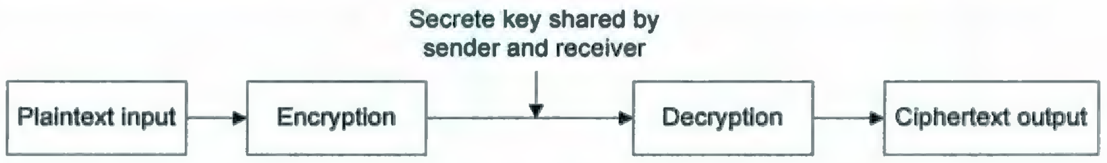


Fig. 1.5. Simplified model for symmetric key cryptography.

Symmetric key cryptography includes two types of ciphers: stream ciphers and block ciphers, which are illustrated in Fig. 1.6 and explained as follows:

- **Block cipher**

As illustrated in Fig. 1.6 (a), a block cipher operates on a block of data (typically 64 bits or 128 bits) by iterating rounds of simple cryptographic operations. In this figure, *b* and *k* represents the key size and block size, respectively. An S-box is typically an important component in a block cipher, and functions as a nonlinear transformation for the data block thereby introducing a complex mathematical relationship between plaintext and ciphertext. Another important component in a block cipher is linear transformation, such as transposition of data bits. The linear transformation aims to bring about diffusion, which means if there is one bit changed in the plaintext, there will be a random change for the ciphertext. Each round of the cipher operation involves a different sub-key (or round key), which is generated according to the algorithm called key scheduling.

The block cipher can be further divided into two types from the structural perspective: Feistel network and substitution permutation network (SPN) [8]. The Feistel structure only operates on a half block of data within one round. The Data Encryption Standard (DES) [9] is an example block cipher using this type of structure. An SPN processes the whole block of data in each round. An example SPN cipher is the Advanced Encryption Standard (AES) [10].

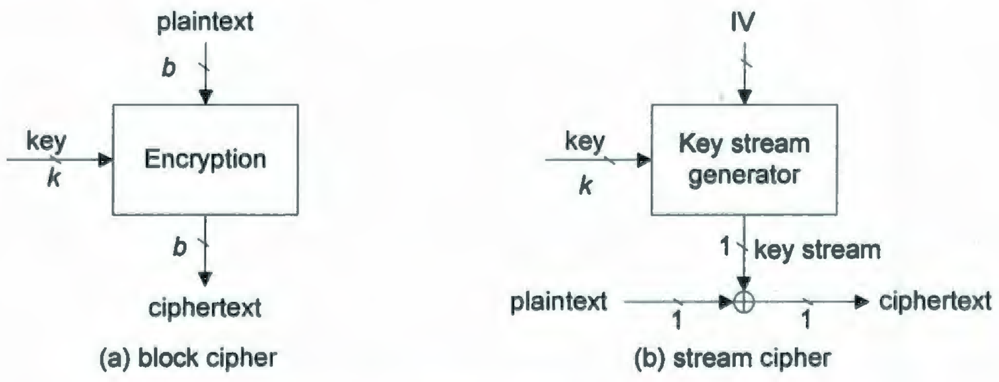


Fig. 1.6. Symmetric key ciphers: block cipher vs. stream cipher.

- **Stream cipher**

As illustrated in Fig. 1.6 (b), a stream cipher typically operates on one bit of data at a time by XORing the generated keystream with plaintext. The keystream is a stream of unpredictable pseudorandom bits. The critical problem in stream ciphers is losing synchronization between the transmitter and receiver, which is caused by lost data. As a result, the following data will be corrupted. To solve this problem, extra information needs to be transmitted from time to time to ensure the keystream is synchronized so that the data at the receiver side can be properly restored. The stream cipher can be divided into two types according to its keystream characteristic: synchronous and self-synchronizing. Synchronous stream ciphers generate keystream independent from the plaintext and ciphertext, and rely on synchronization of keystream at the transmitter and receiver. In contrast, self-synchronizing stream ciphers (such as MOSQUITO [11]) use previous ciphertext bits to generate the keystream, and the receiver can be resynchronized automatically. Although currently most self-synchronizing stream ciphers are found to have security flaws, it is possible to use the mode of operation, a scheme operating on a block cipher, to configure block ciphers as a stream cipher using cipher feedback to ensure self-synchronization characteristics, such as SCFB mode [12].

Project eSTREAM, organized by EU ECRYPT, evaluated new stream ciphers [13]. The final selected eSTREAM portfolio algorithms include: (1) HC-128, Rabbit, Salsa20/12, and SOSEMANUK, which are designed for software purpose (Profile 1), (2) F-FCSR-H v2, Grain v1, Mickey v2 and Trivium, which are designed for hardware purpose (Profile 2).

2) Asymmetric Key Cryptography

Asymmetric key cryptography depends on the difficulty of a mathematical problem, and two different keys are involved for encryption and decryption, respectively. This pair of keys includes one public key (known publicly) and one private key (kept secretly), which are different and mathematically related. Asymmetric key cryptography can provide both the data confidentiality and the identity authentication as illustrated in Fig. 1.7.

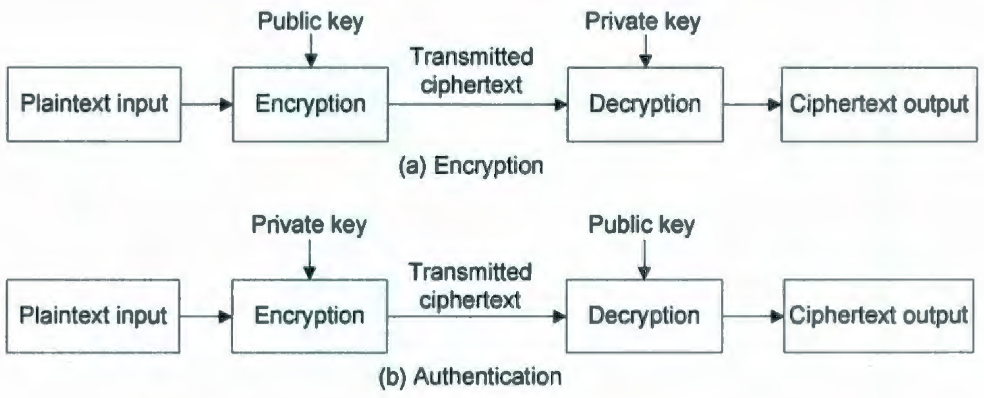


Fig. 1.7. Simplified model for asymmetric key cryptography.

There are two well-applied a symmetric key ciphers: RSA [14] and elliptic curve cryptography (ECC) [15]. RSA is an early public key cipher, which bases its security on the difficulty of factoring a large composite number into prime numbers. RSA is widely used in security protocols for authentication. However, it introduces large complexity for implementation. ECC is based on the algebraic structure of elliptic curves over finite fields where it is assumed that it is difficult to find the discrete logarithm of a random elliptic curve element according to a publicly known base point. Compared to RSA, the implementation complexity of ECC is much lower and, hence, it is recommended for use in lightweight security protocols.

1.2.2 Cryptography Objectives

By applying cryptography, many security features can be achieved, as summarized below:

- Access control: Access control is a security requirement for the physical layer. Only the authorized users can access to the network or host system, and they must obey their organization's policy.
- Confidentiality: Confidentiality guarantees that information can only be read by authorized users, that is, the message cannot be revealed by simple eavesdropping by unauthorized users.
- Authentication: Authentication is a mechanism used to ensure the identity of both communicating parties, so that the message is communicated between parties with valid identity. Authentication is used to prevent attackers from forging data and masquerading as legitimate users.
- Integrity: Integrity means that data should be transmitted without being changed. The information should be protected in the whole network, including both users' private data and control information.
- Freshness: Freshness ensures prevention of the message replay attack. That is, a data packet transmitted earlier cannot be transmitted again, since denial of service (DoS) may happen if the attacker stores old packets and keeps resending them. Usually, nonces and timestamps are used in the security scheme.

- Non-repudiation: Non-repudiation means that the user cannot deny the fact that it had used the network. That is to say, the truth of an already happened activity in the network should be able to be proved.

1.3 Security in WSNs

In this section, we will focus on the security mechanisms in wireless systems, in particular, the importance and challenges of achieving security in WSNs.

1.3.1 Security in General Wireless Systems

We focus on introducing several aspects of security in general wireless networks: (1) attacks from different perspectives and their consequences; (2) current security approaches functioning in different ways to protect the whole network's security; (3) cryptography and its two important components, cryptographic algorithm and cryptographic scheme.

1) Attacks: Data vs. Channel

The objective of applying security mechanisms is to prevent the network from being successfully attacked. Attacks can be classified from different perspectives: from the data transmission aspect, it can be classified as passive attack and active attack; while from the communication channel aspect, it can be divided into overusing and jamming attacks. Although the detailed ways of attacks may vary greatly, attack consequences can be classified into a few categories. The detailed consequences of attacks are discussed as following:

- **Eavesdropping**

Eavesdropping is a common way to attack and is a passive attack. Attackers overhear the information being transmitted. It is especially easy for such attacks in wireless networks due to the use of open medium. As a result, the privacy may not be protected, although attackers may not change the content.

- **Data Alteration**

Altering the communication data is generally regarded as an active attack. Attackers not only steal the message from the sender, but also change the content before sending it out. The altered message will bring in great danger because the receiver and transmitter will make wrong decisions based on the message they receive.

- **Radio Channel Overuse**

Some users may overuse the radio spectrum, which is supposed to be shared by all users. This will not happen in cellular phone systems because protocols constrain the subscriber's use. However, for other networks where spectrum is shared, like Wi-Fi, such issues may exist.

- **Channel Jamming**

Jamming can happen at multiple protocol layers, including the physical layer, link layer, and higher layers. At the physical layer, noise is the general form, which has serious impact on the normal communication activity. At the link layer, jamming occurs as the attacker keeps sending messages when other users are going to transmit and/or receive information. The jammed messages ensure other users cannot access the medium and cannot communicate with each other. Such an attack is also known as the denial of service (DoS) attack. In WSNs, the jammed packet can even deplete the energy supply of a sensor node and eventually cause it not to work anymore.

2) Approach: Protection vs. Detection

Generally, security mechanisms in wireless networks can be divided into two categories: protection based and detection based. Protection based technologies can be viewed as the first defense line of the security system, in which cryptography is used to prevent malicious outside attackers from using both passive attacks and active attacks. At the same time, detection based technologies form the second line of defense of security system. They are used to detect the possible intrusion inside the network in order to prevent the situation that the cryptographic system has been compromised. For example, in a WSN, a sensor node may be obtained and modified by attacker after deployment. It is not easy to notice the attack by using cryptography; however, it may be easy to detect by observing the abnormal behavior of the sensor node.

- **Protection Based**

In order to achieve proper data safety, a cryptographic algorithm is used to protect data from being revealed. Typically, a cipher is used to execute operations of encryption and decryption, which enables the secure communication over an insecure channel. The data transmitted securely in a network is ensured by cryptographic schemes, which can be viewed as a combined consideration of selecting ciphers and how to use them to achieve security goals. The security of cryptography relies on the condition that the secret key is kept safe by the users on each side of the communication and not known by others. A typical method of breaking such security is to collect messages between two communicating parties and analyze them to find the weakness.

- **Detection Based**

There is no cryptography involved in the detection-based approaches. Detection based approaches mainly focus on the signs of being attacked by monitoring the network traffic. For example, misuse detection depends on the experience accumulated. The known attacker activities are collected to build the database for checking for future misuse behavior. Anomaly detection focuses on detecting abnormality of the network activity, which depends on the definition of what kind of activity should be regarded as normal or abnormal.

3) Cryptography: Algorithm vs. Scheme

Cryptography includes two important components: the cryptographic algorithm and the cryptographic scheme, whose research focus is illustrated in Fig. 1.8. The cryptographic algorithm directly operates on data, which needs to be protected. In contrast, the cryptographic scheme concentrates on how to implement the cryptographic algorithm to ensure the security in the whole system. The cryptographic algorithm can be further divided into two categories: symmetric key algorithms and asymmetric key algorithms. As discussed previously, symmetric key algorithms can be further categorized into two types: block ciphers and stream ciphers.

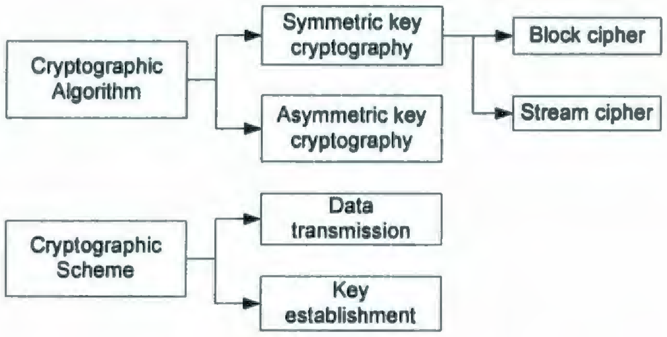


Fig. 1.8. Cryptography category.

Usually, cryptographers design a cipher and conduct cryptanalysis to ensure that the cipher will not be broken using a practical amount of resources in a practical amount of time. Each type of the cipher has different characteristics, and the application environment determines which one is chosen for use. In Table 1.1, we list the ciphers selected for different application environments due to their corresponding network characteristics.

Table 1.1. Cipher used in different application environment.

Network	Cipher used	Cipher type
ZigBee (IEEE 802.15.4)	AES	Block cipher
Bluetooth (IEEE 802.15.1)	E0	Stream cipher
WLAN (IEEE 802.11)	RC4 / AES	Stream / block cipher
GSM	A5/1 and A5/2	Stream cipher
3G networks	KASUMI	Block cipher
UWB (IEEE 802.15.3a)	Compact ciphers	Stream cipher and block cipher

Cryptographic schemes may cover multiple aspects, such as secure data transmission and key distribution through an insecure communication channel. For secure data transmission, a scheme needs to consider which cipher to use and how to use it in an efficient way. For the key establishment, a scheme needs to consider how to make two communicating parties agree on a key securely without revealing the key to others. Identity authentication is an important aspect and usually depends on a trusted third party. During communication, a message authentication code (MAC) value [8] and digital signatures [8] are often used to ensure the authentication and integrity, meanwhile, nonces and timestamps are used to prevent message replaying attacks. A good cryptographic scheme needs to consider many factors as shown in Table 1.2.

Table 1.2. Common considerations for cryptographic scheme.

From the cryptographic algorithm aspect	From the network requirement aspect
• The complexity of the cipher • The energy cost of the cipher • The restrictions of the cipher • The implementation environment (software or hardware)	• The security requirements • The life time span of the network • The network throughput requirement • The size of the data being operated upon • The feature of the data (real time processing or post time processing)

It is worth noting that cryptography is not omnipotent and it has its own limitations. For example, a MAC can determine whether the received data is different from the transmitted data, however, it cannot discern the reason of the difference (a malicious attack or just a noisy channel). Cryptographic algorithms cannot make everything magically secure. For instance, they cannot prevent traffic analysis or refuse to receive the replayed packets. In addition, cryptography cannot solve the problem of packets being jammed and cannot get rid of a malicious insider such as a captured node. Nevertheless, cryptography is still an efficient and straightforward approach to protect the network communication from being attacked.

1.3.2 Challenge of Security in WSNs

A WSN is a specific type of wireless network and its security system largely inherits the same characteristics as other wireless networks. Here we just reinforce the importance of securing a WSN and the challenges of a secure WSN.

Information security in WSNs is important:

- In classical WSNs, information security directly relates to the validity of the monitored data and the result of analysis. In cases of security and disaster monitoring applications, maliciously changed data may hide the possibility of a disaster or generate false alarms, which may mislead people into panic.
- In WMSNs, the data may include private information of people, which should be kept confidential from any unauthorized personnel.
- Information security is particularly important in WBSNs. First, malicious modification of the medical data will seriously affect medical analysis. As a more serious result, it may threaten the safety of a patient's life. Second, since the medical data may directly involve the privacy of the person monitored, the data should be guaranteed secure in transmission and storage.

Information security in WSNs is challenging:

- The data is transmitted in an open wireless media, which makes the security system more vulnerable. This is a typical nature of the wireless network, and the transmitted information can be easily eavesdropped.
- Sensor nodes are often deployed in an area lacking supervision by people, which greatly increases the possibility of being attacked.

- Limited energy supply in sensor devices is a huge problem, which limits the security level that can be achieved in a WSN. The reason is that a sensor node cannot afford to use complicated cryptographic schemes due to their large energy cost, although those schemes may provide strong protection and be difficult to break.

1.4 Motivation for Research

As discussed in the previous section, there are many challenges to achieve security goals in WSNs. Some physical constraints may not be avoidable due to the nature of WSN itself, such as openness of the wireless media, limited power supply, and the non-protected sensor device during com munication. Nevertheless, the problem of limited energy can be mitigated by using appropriate methods.

We maintain that both cryptographic algorithms and cryptographic schemes affect the energy efficiency in a secure WSN. Although there are energy-oriented schemes proposed for WSNs, no effort has been made to investigate the energy efficiency of a sensor node in a noisy environment, and jointly considering the cryptographic algorithm and the cryptographic scheme as a whole. Motivated by this observation, we explore the energy efficiency when cryptography is applied to WSNs. The objective of our research is to investigate the energy efficiency in secure WSNs by jointly considering the cryptographic algorithms and the cryptographic schemes.

In considering cryptographic algorithms for WSNs, we investigate the implementation of the algorithms in both software and hardware environments. For software implementation, we evaluate the computational energy cost of different symmetric key ciphers considering both the algorithm characteristics and channel quality. For hardware implementation, we focus on the energy efficient design, specifically, investigating the performance of hardware and software cooperation.

In the study of cryptographic schemes in WSNs, we explore the effects of various factors on energy consumption for different schemes, including packet size, process used in the generation of the initialization vector, and channel quality. We present the integrated performance evaluation for different ciphers and schemes by developing an analysis model for a sensor node. The appropriateness of the model is supported by simulation results. We also investigate the energy consumption for the session key establishment in WSNs, considering different effects, such as cryptographic algorithms and key update frequencies.

1.5 Thesis Organization

The organization of thesis is as follows and illustrated in Fig. 1.9:

In Chapter 2, we provide the background and literature review pertaining to our research work.

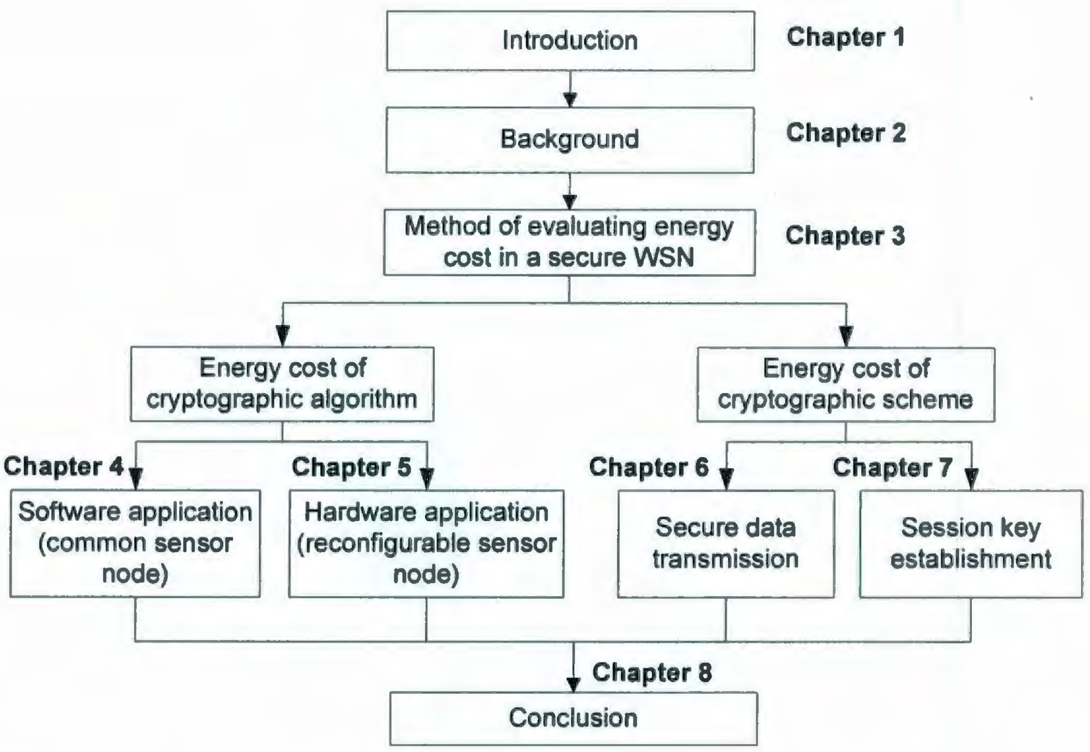

Introduction
Chapter 1
Background
Chapter 2
Method of evaluating energy cost in a secure WSN
Chapter 3
Energy cost of cryptographic algorithm
Energy cost of cryptographic scheme
Chapter 4
Chapter 5
Chapter 6
Chapter 7
Software application (common sensor node)
Hardware application (reconfigurable sensor node)
Secure data transmission
Session key establishment
Chapter 8
Conclusion

Fig. 1.9. Thesis organization.

In Chapter 3, we present the method to evaluate the energy cost in a sensor node. We propose to use *energy efficiency*, i.e., the total amount of valid information transmitted per Joule, as the performance metric and present a method of energy calculation including both the computational cost and communication cost. After that, we focus on investigating a case including no security features, which provides a basis for further investigating the energy performance of a secure WSN. In this case, we specifically evaluate the energy cost of a sensor node and investigate different factors affecting the energy performance such as the payload size and the channel quality.

In Chapter 4, we evaluate the energy cost for cryptographic algorithms considering their software application environment, which is provided for commonly used sensor nodes. We implement the ciphers (including both stream ciphers and block ciphers) in assembly language and derive the computational energy cost model of the ciphers in terms of the number of CPU cycles required to perform encryption. Furthermore, we evaluate the energy performance of cryptographic algorithms when they are applied to a WSN in noisy channel conditions.

In Chapter 5, we conduct the energy cost evaluation for cryptographic algorithms considering their hardware application environment, which is provided by reconfigurable sensor nodes. We propose using involutional block ciphers since the characteristic of involution operation enables performing encryption and decryption using the same piece of logic circuit. We choose two involutional block ciphers KHAZAD and BSPN for implementation. Our particular focus is on the analysis of the energy cost of different implementation methods. In addition, we investigate the energy cost considering software and hardware cooperation in reconfigurable sensor nodes.

In Chapter 6, we focus on the energy cost analysis for different cryptographic schemes in a WSN, which directly affect the communication energy cost for a wireless sensor node. Specifically, we investigate different parameters, which affect the energy cost of link layer cryptographic schemes, such as the payload size, the distribution of initialization vector, and the channel quality. We propose an analysis model, which takes into account these parameters, to evaluate the performance of cryptographic communication schemes. We further conduct simulation experiments to verify and support the appropriateness of the proposed model.

In Chapter 7, we investigate the energy cost for session key establishment in a WSN. We examine the energy cost of session key establishment using the model developed in the previous chapter. We explore the effects of a key establishment to the overall energy supply in a sensor node, especially considering the key establishment frequency and the channel quality.

In Chapter 8, we conclude the thesis and provide suggested future research directions.

Chapter 2

BACKGROUND

This chapter presents the background for our research. We first review security protocols in various types of networks (from wired to wireless) to show a general concept of the security protocols: cryptographic schemes are designed to satisfy th e specific requirements of application environments. Then we focus on presenting the state-of-the-art development in secure WSNs for different aspects, such as cryptographic algorithms and cryptographic schemes.

2.1 Cryptography Applied to Various Networks

In this section, we review how cryptography is applied to different types of networks, such as the Internet wired network, and wireless local area networks (WLANs), wireless personal area networks (WPANs), and mobile ad hoc networks (MANETs) as examples of wireless networks.

2.1.1 Wired network: the Internet

Network communication is a complicated process since it is impacted by many factors, such as speed, scalability, bandwidth, reliability and security. To satisfy these requirements, a layered protocol is proposed to deal with the information independently based on functionality [16]. The Internet can be divided into five major layers, as shown in Fig. 2.1, which are physical layer, link layer, network layer, transport layer and application layer. When sending out a message, the transmitter attaches different headers to the original message from top to bottom according to different layers. While receiving a packet, these headers are stripped out at different layers from bottom to top, finally obtaining the original message. Protocols are applied to different layers to ensure the order and validity of the data among the communicating parties by well-defined packet formats and rules.

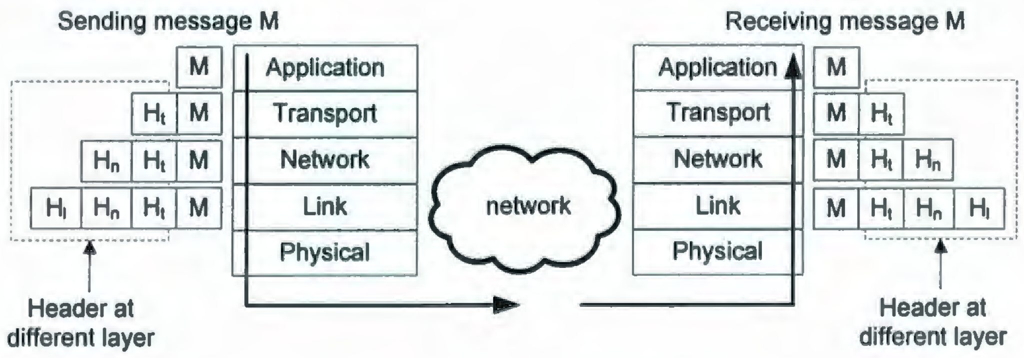


Fig. 2.1. Protocol layers in the Internet [16].

In the Internet, various security protocols are applied to different protocol layers of the network:

- **Security at the Link layer**

Point-to-point protocol (PPP) (RFC 1661) [17] is a typical link layer protocol, establishing a direct connection between two communicating nodes. PPP can operate on multiple types of physical media, such as a telephone line, a SONET/SDH link, an X.25 connection, or an ISDN circuit. There are many authentication protocols running over

PPP, such as password authentication protocol (PAP) (RFC1334) [18], challenge-handshake authentication protocol (CHAP) (RFC1994) [19], and extensible authentication protocol (EAP) (RFC2284) [20]. PAP is the simplest authentication protocol, which transmits the user name and password directly without encryption. It is vulnerable to attacks since anyone can get the password directly by monitoring a nd analyzing the packet. CHAP provides higher security level, using MD5 to generate a hash value based on the private information of the remote user and the challenge information from the authenticator. EAP provides a framework of authentication (for both PPP and IEEE 802) and it only defines the packet format to indicate how the packet should be encapsulated.

- **Security at the network layer**

The Internet protocol security (IPsec) is a typical security protocol operating on IP packets at the network layer [16]. IPsec provides authentication, confidentiality, and key management, and it defines two types of protocols: authentication header (AH) protocol and encapsulating security payload (ESP) protocol, providing authentication and encryption-authentication, respectively. There are two working modes for both AH protocol and ESP protocol: transport mode and tunnel mode. Transport mode is used in end-to-end communication providing protections for upper-layer protocols, which means it only operates on the payload of the IP packet. In contrast, ESP protocol focuses on protecting the entire IP packet between two security gateways by encapsulating the whole IP packet as a payload with a new packet header. IPsec provides an overall security service for IP packets and the details are determined by the utilization of AH protocol and/or ESP protocol. The security services provided by IPsec include access control, connectionless integrity, data origin authentication, rejection of replayed packets (a form of partial sequence integrity), confidentiality (encryption), and limited traffic flow confidentiality.

- **Security at the transport layer**

In the Internet, transport layer security (TLS) (or secure socket layer (SSL)) is a security protocol operating at the transport layer [16]. It is a two-layer security protocol and provides protection for the TCP segment for end-to-end communication.

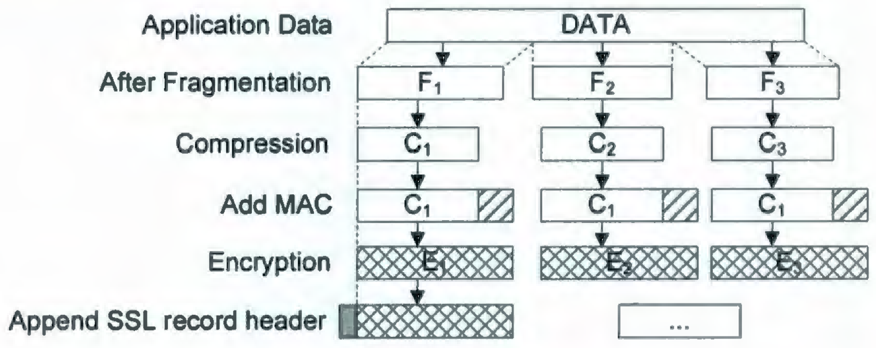


Fig. 2.2. Operation of SSL record protocol [16].

The basic layer in TLS/SSL is the record protocol, which provides basic security services such as data confidentiality and integrity to its upper layer. TLS/SSL provides confidentiality by using symmetric key cipher and provides message integrity by using message authentication code (MAC) [8]. The specific operation process of the record protocol is shown in Fig. 2.2.

Within the layer of the record protocol, there are three specific protocols: the handshake protocol, the change cipher spec protocol, and the alert protocol. The handshake protocol is used to decide the algorithm and encryption key for both communicating parties and it should be used before the start of data transmission. The change cipher spec protocol is a single byte message, which is used to update the cipher being used in the connection. The alert protocol functions as transmitting alerts such as indicating the status of the connection (warning or fatal) and the detailed alert (such as unexpected message received, incorrect MAC received, and so on).

- **Security at the application layer**

At the application layer, there are many security protocols providing data confidentiality, integrity and identity authentication, such as secure shell (SSH) [21], pretty good privacy (PGP) [22], secure electronic transaction (SET) [23] and Kerberos protocols [24]. These protocols focus on specific applications: PGP focuses on protecting the privacy of Emails, SET focuses on securing credit card transactions, and Kerberos focuses on key distribution and identity authentication. Security protocols at the application layer can be very flexible and designed according to the specific needs.

2.1.2 Cryptography Applied to Wireless Network

1) Cryptography Applied to WLAN

A wireless local area network (WLAN) is a popular wireless network being used in small range area such as schools and office buildings. Laptops and PDAs are typical devices in WLANs, using an access point (AP) to connect to a wide area network (usually the Internet). WLANs may also be called "802.11", or "Wi-Fi". There are two working modes in a WLAN: infrastructure mode and ad hoc mode, which are illustrated in Fig. 2.3, with the difference of whether an AP exists in the network [16].

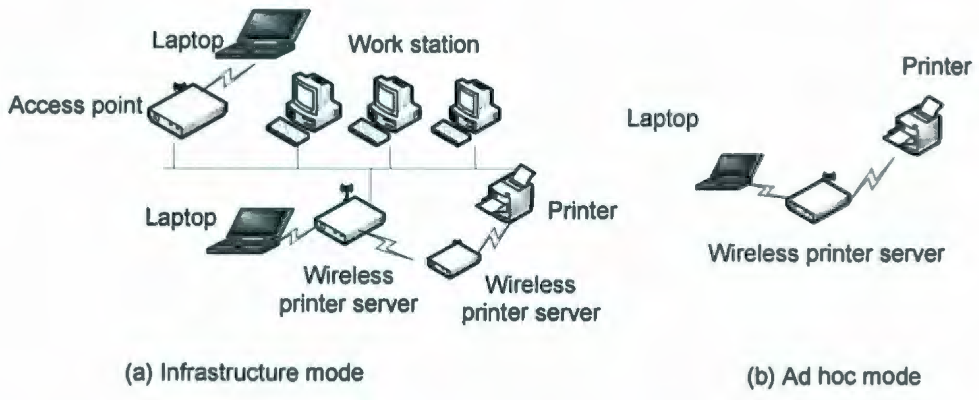


Fig. 2.3 IEEE 802.11 LAN architecture

In WLANs, the development of security protocols has involved fixing security flaws for higher security requirements. These involved Wi-Fi security standards include wired equivalent privacy (WEP) [25] and Wi-Fi protected access (WPA and WPA2) [26].

WPA is built upon WEP, adding more security mechanisms (such as key management, encryption and authentication) to make it more secure. WPA2 is designed to replace WPA by introducing more security mechanisms. For data confidentiality, both WEP and WPA use the stream cipher RC4, while WPA2 select AES for better security; for the data integrity, WEP uses CRC, while WPA and WPA2 use Michael and counter with CBC-MAC (CCM) mode on AES, respectively, to generate a message integrity code (MIC). WEP uses the same key for all communication and the key never expires; WPA fixes this weakness by providing a rekeying mechanism called temporal key integrity protocol (TKIP) and using multiple keys; WPA2 also uses multiple keys for encryption and it requires extra certification from the Wi-Fi Alliance. There are two modes of key management for WPA and WPA2: pre-shared key (PSK) mode and 802.1x which are designed for personal usage and company usage, respectively.

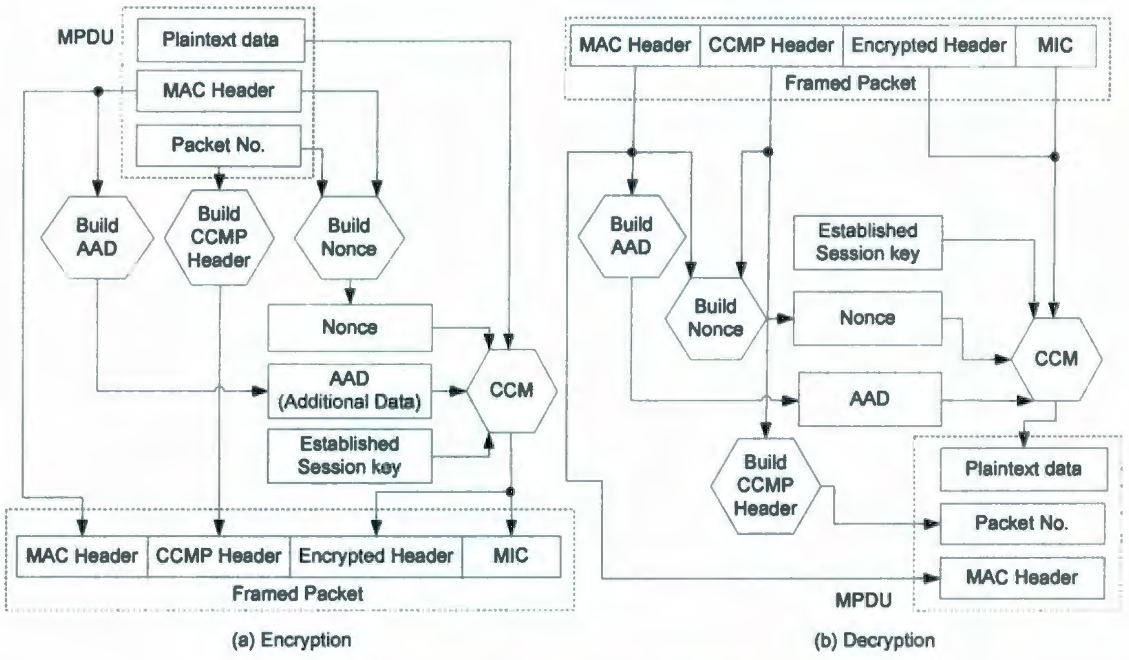


Fig. 2.4. CCMP Encryption in IEEE 802.11i [27].

Counter with CBC-MAC mode protocol (CCMP) is currently the strongest security protocol in WPA2, which uses CCM mode of operation based on AES. CCMP defines multiple rules to encrypt and decrypt the data frame in Wi-Fi and offers better integrity and confidentiality for the whole data packet by incorporating the packet header into the operation process. The whole encryption and decryption system provides greater security for MPDU (MAC Protocol Data Unit), which are illustrated in Fig. 2.4.

2) Cryptography Applied to WPANs

A wireless personal area network (WPAN) is a type of network, which serves to interconnect computing devices, typically centering at a person's workspace. These devices may be a computer, PDA, cell phone, printer, mouse, etc.. Bluetooth and Zigbee are two typical technologies used in WPANs, catering to the need of low cost and low power. Both Bluetooth and Zigbee operate on a wideband radio with a high frequency of 2.4GHz, and a single chip usually integrates the RF transceiver with the security processing (such as data encryption and MAC generation) for the purpose of low cost, such as CC2420 and CC2540 [28], which integrate security processing with Zigbee and Bluetooth RF transceiver, respectively.

Here, we briefly review the two technologies and show how security is applied:

- **Bluetooth**

Bluetooth holds a master-slave structure as shown in Fig. 2.5. In a Bluetooth network, one master device and up to seven slave devices constitute a piconet. Several piconets together can form a larger network called a scatternet [16].

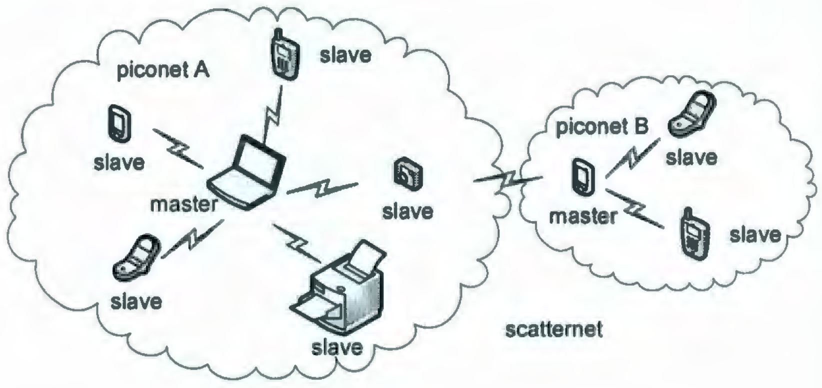


Fig. 2.5. Architecture of Bluetooth network.

NIST has published a guide to Bluetooth security [31] to provide suggestions on specific security methods for Bluetooth technologies. In this document, it defines four security modes, which can be divided into three types of services:

- Non-secure service: mode 1 belongs to this type, which bypasses all the security considerations.

- Service level enforced security: mode 2 and mode 4 belong to this type of service, which initializes after the link setup.
- Link level enforced security: mode 3 belongs to this type, which initializes before the link setup.

Pairing is an important process for both service level and link level enforced security, which helps two devices establish a relationship by creating a link key. Before Bluetooth v2.1, a PIN code is used for pairing, called legacy pairing, which allows the communication between two devices if they enter the same PIN code and provides encryption and authentication using a generated link key. However, for newer versions, the secure simple pairing (SSP) is used for simplifying the pairing process, which utilizes ECDH public key cryptography proposed by NIST.

- **Zigbee**

Zigbee is a specification for WPANs, including IEEE 802.15.4 of version 2003 [29] for the physical layer and link layer. It is designed to satisfy the requirement of low data rate and power context, such as home automation, entertainment and toys. Zigbee technology is designed to be easier and less expensive than Bluetooth and typically used in a device supplied by small battery. It supports three types of network topology as shown in Fig. 2.6. In this figure, there are three types of Zigbee devices:

- Zigbee coordinator (ZC) , which manages the whole network and there is only one ZC device inside the network
- Zigbee router (ZR), which functions as an intermediate router and pass data for other devices

- Zigbee end device (ZED), which can only communicate with ZC and ZR and it cannot relay data for other devices

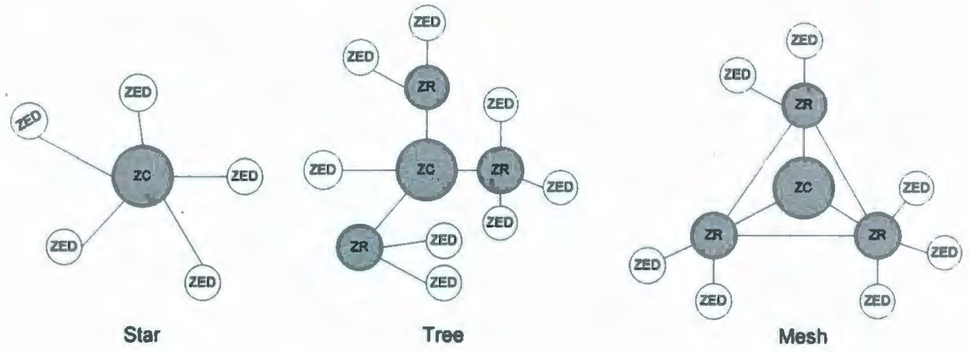


Fig. 2.6 Network topology of Zigbee.

Zigbee provides information security by symmetric key cryptography. The ZC device is used as a trust centre, which in charge of cryptographic key establishment and transportation, data frame protection, and device management. Each Zigbee device is preloaded with the address of the trust centre and a unique key called master key, which is used for an end-to-end secure communication between the device and trust centre. Link key and network key are two types of keys, which are established, distributed by the trust center, and used for peer-to-peer communication and broadcast communication, respectively. All the three type of keys (master key, link key and network key) are 128 bits long and used by cipher AES with CCM* mode, which is a minor variation of CCM providing additional CCM features such as encryption-only and integrity-only.

3) Cryptography Applied to MANETs

Mobile ad-hoc networks (MANETs) are popular due to their flexible style of communication. MANETs are infrastructureless: each node within the network functions as a packet forwarder, as well as a host. Routing in MANETs becomes critical since the node's mobility makes the topology of network change dynamically. Cryptographic schemes in MANETs especially focus on ensuring the safety of routing, since the dynamically changed routing information directly determines whether a data packet can be successfully delivered to the destination node. However, the limited resources, such as bandwidth and processing capability, make it impossible for a MANET to implement complicated protocols as in the Internet.

In MANETs, confidentiality is optional and even may not be necessary since the routing information itself is by broadcast throughout the network. More importantly, as MANET is a multi-hop network, the encryption and decryption for messages at each hop will introduce timing delay, which is not desired when the network topology is frequently changed. A signature is typically used to provide the information integrity and identity authentication. A m essage will be accepted by the receiver only when it passes the verification, otherwise, it will be dropped. Both symmetric key and asymmetric key cryptography can be used to generate the signature according to the security level required. The asymmetric key cryptography can provide higher level of security; however, shared secret key among a group of trusted nodes is also acceptable. Although it is generally believed that a group key cannot be used for authenticating the identity since each node that knows the key can generate the message, a tradeoff is made between the cost and the security level.

Currently proposed security protocols for MANETs are mostly developed from extending existing routing protocols. For example, researchers develop security schemes based on optimized link state routing (OLSR) [32], which is a proactive link state protocol designed for MANET and has been accepted in RFC3626. OLSR follows the layering design like in the Internet and the routing message uses user datagram protocol (UDP) with the port number 698.

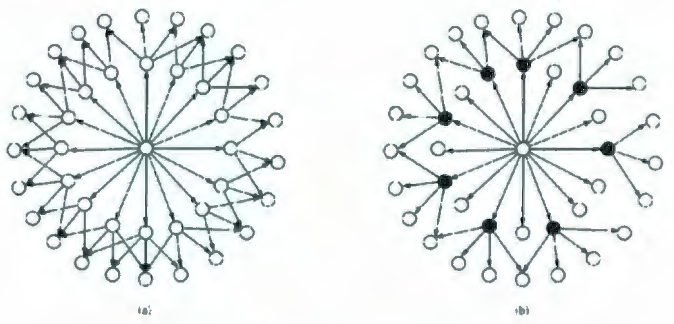


Fig. 2.7. OLSR protocol structure [32].

The general concept of the OLSR protocol is illustrated in Fig. 2.7, where one node is chosen as an example to describe the basic idea. As shown in figure (a), a central node can communicate with other nodes in the network through the wireless media. Nodes in the first circle around the central node are called one-hop neighbors. Neighbors of these one-hop neighbors (i.e., the second circle from the central node) are called two-hop neighbors of the central node. The central node selects some nodes within its one-hop neighbors as illustrated in figure (b). These selected nodes are called multipoint relay (MPR) nodes and via which the central node can reach all its two-hop neighbors. OLSR protocol includes two major processes: (1) Hello message flooding, where each node periodically broadcast its status and decides the MPR set and (2) Topology control message broadcast, where each node periodically broadcast who is selected as its MPR and the information to be used to generate the routing table.

Security extensions proposed for OLSR can be divided into two categories: end-to-end schemes and hop-by-hop schemes [33]. An end-to-end scheme treats every node along the routing path as a forward node except the source and destination nodes. The signature is generated by the source node and verified by the destination node, that is to say, other nodes along the path will not verify the signature. Unlike the end-to-end scheme, in a hop-by-hop scheme, routing information is aggregated and every node along the routing path will verify the signature. If the message passes the verification, the old signature will be removed. The node will add new information into the packet and append its own signature. Whether MPR nodes have retransmitted the message out is a critical factor to detect attacks in MANET when OLSR is applied. To solve this problem, a scheme, called SA-OLSR, is proposed where a node can detect malicious MPR nodes by receiving acknowledgement messages from its two-hop neighbors [34]. Although it will greatly increase the overhead of communication to include the ACK messages, the attack problem can be effectively solved in this scheme.

2.2 Cryptography Applied to WSNs

There is no unified security solution for wireless sensor networks, and neither for the cryptography applied to it. In this section, we focus on reviewing the state-of-the-art work for cryptography applied to WSNs. We focus on the study of currently available cryptographic algorithms and schemes, and particularly study the notion of secure data aggregation and typical architectures for secure WSNs.

2.2.1 Cryptographic Algorithms

1) Symmetric Ciphers vs. Asymmetric Ciphers

Many researchers focus on recommending appropriate ciphers for WSNs considering the constrained resources in a sensor device. Results show that symmetric key ciphers are more practical than asymmetric key ciphers to be used in a sensor node since the computational complexity of asymmetric key cryptographic algorithms introduce much more energy cost. The large difference of energy cost can be seen from Table 2.1, which is an analysis result of energy cost between asymmetric key cipher and symmetric key cipher according to [35].

Table 2.1. Energy cost comparison: asymmetric vs. symmetric [35].

Asymmetric key cryptographic algorithm energy consumption					
Algorithm	**Data size (bits)**	**Signature (*mJ*)**		**Key exchange (*mJ*)**	
		Sign	**Verify**	**Client**	**Server**
RSA-1024	1024	304	11.9	15.4	304
ESCSA-160	160	22.82	45.09	22.3	22.3
RSA-2048	2048	2302.7	53.7	57.2	2302.7
ECDSA-224	224	61.54	121.98	60.4	60.4
Symmetric key cryptographic algorithm energy consumption (*mJ*/Byte)					
SHA-1	0.0059				
AES-128 Encryption	0.00162				

Although asymmetric key ciphers consume much more energy than symmetric key ciphers, there are still some researchers showing interest in this field because asymmetric ciphers are more convenient for the key establishment and identity authentication [36] [37]. However, the huge overhead of energy cost is still the largest problem.

2) Block Ciphers vs. Stream Ciphers

Researchers focus on evaluating the efficiency of symmetric key ciphers in WSNs from two aspects: memory cost and CPU cycles, such as in [38] and [39].

- **Block cipher**

The performance of block ciphers varies from their different characteristics, such as the key size, the block size and the number of rounds. There are three main components for a block cipher, which are typically involved in an evaluation: encryption algorithm, decryption algorithm and key scheduling algorithm. Many popular block ciphers have been investigated, such as AES [10], Skipjack [40], RC5 [41], RC6 [42], Twofish [43], MISTY1 [44], KASUMI [45], Camellia [46], etc., and we briefly list their parameters in Table 2.2.

Table 2.2. Characteristics of popular block ciphers.

Cipher	Block size (Bytes)	Key size (Bytes)	Number of rounds
AES	16	16	10
Skipjack	8	10	32
RC5	8	16	12
RC6	16	16	20
Twofish	16	16	16
MISTY1	8	16	8
KASUMI	8	16	8
Camellia	16	16	18

In recent years, many lightweight block ciphers have been proposed to meet the low area and low power consumption requirements, such as ICEBERG [47], PRESENT [48], HIGHT [49], SEA [50], and PUFFIN [51]. These compact ciphers typically have smaller block sizes and S-boxes. Characteristics of these ciphers are shown in Table 2.3. The reason we list these ciphers here is that they might become good candidates for WSNs to provide energy efficiency and acceptable security level at the same time.

Table 2.3. Characteristics of lightweight block ciphers.

Cipher	S-box	Block size (Bytes)	Key size (Bytes)	# of rounds
ICEBERG	4×4	8	16	16
PRESENT	4×4	8	10 or 16	31
HIGHT	4×4	8	16	32
PUFFIN	4×4	8	16	32
SEA	3×3	variable	variable	variable

- **Stream Ciphers**

Stream ciphers are not as much understood as block ciphers (which provide flexible functionalities like block encryption, stream encryption and MAC generation). However, since a stream cipher does not propagate errors introduced by the communication channel, i.e., an error bit in the ciphertext only produces one error bit in the recovered plaintext, there is also on-going research focusing on energy evaluation of stream ciphers in WSNs. The recently investigated stream ciphers include: HC-128 [52], HC-256 [53], SOSEANUK [54], SNOW [55], Phelix [56], Dragon [57], LEX [58], RC4 [59], etc.. In [60], these modern stream ciphers are evaluated on an 8-bit AVR MCU to show the appropriateness of implementing stream ciphers in small microprocessors, which happens to be a typical implementation environment used by a sensor node. In addition, in [39], the stream ciphers are especially analyzed for WSN purposes according to the number of CPU cycles and memory requirements.

3) Modes of Operation

A mode of operation is a scheme to make use of a symmetric key block cipher when operating on a large bulk of data [61]. There are five basic modes as shown in Fig. 2.8, which provide considerable advantages to a block cipher, especially making its application more efficient to satisfy different requirements in a practical environment. Electronic codebook (ECB) mode and cipher block chaining (CBC) mode operate on the plaintext block by block, while cipher feedback (CFB) mode, output feedback (OFB) mode, and counter (CTR) mode make the block cipher function like a stream cipher. In WSNs, many encryption schemes have been proposed to utilize the basic modes of operation, such as SPINS using CTR mode [62] and TinySec using CBC mode [63].

The ciphertext size will be different according to the operation mode being used, which is summarized as follows:

- Ciphertext Size Equals Multiple of Block Size

ECB mode and CBC mode belongs to this case, which is illustrated as Fig. 2.8 (a) and (b), respectively. The plaintext will be padded with bits, for example, some zeroes, to make the size of plaintext be a multiple of the block size.

ECB mode is the most straightforward mode for block ciphers, which encrypts blocks of plaintext directly into blocks of ciphertext. It is considered ideal for encrypting small size plaintext. However, it is vulnerable to attacks when encrypting large amounts of data, since the same plaintext always results in the same ciphertext and a codebook for plaintext and ciphertext pair can be created. CBC mode is generally utilized to encrypt large bulks of data and overcome the ECB security flaw by using an initialization vector (IV). CBC mode functions like a chain, each block of ciphertext is obtained by encrypting the result of the previous ciphertext block XORed with the current plaintext block, except that the first block is generated by XORing plaintext with the IV.

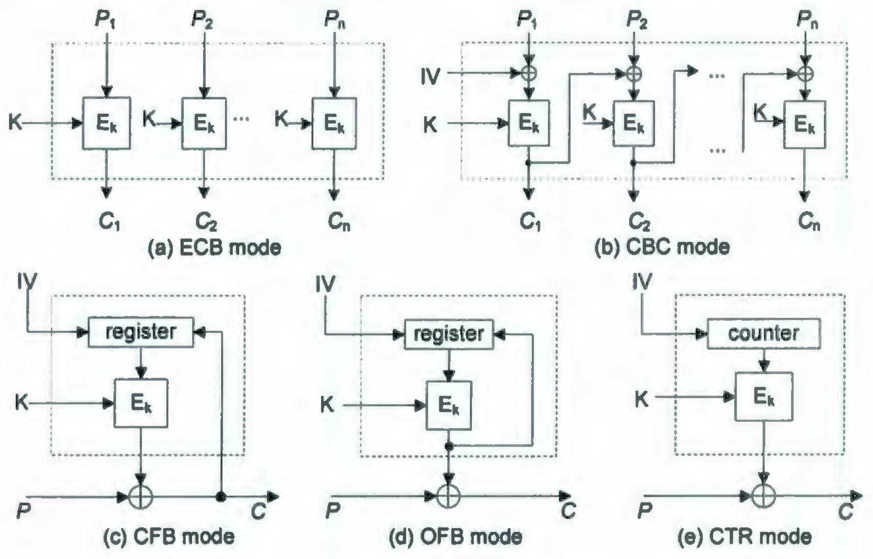


Fig. 2.8. Basic mode of operation on block ciphers.

- Ciphertext Size Equals Plaintext Size

It is the characteristic of a stream cipher that the ciphertext size equals the plaintext size and need not be a multiple of a block size. For a block cipher, three operation modes can achieve this characteristic: CFB mode, OFB mode and CTR mode, which are illustrated in Fig. 2.8 (c), (d) and (e), respectively.

CFB mode and OFB mode function similarly by using the feedback of previous results stored in a register. The value of the register is initialized by IV and encrypted by the cipher key to generate keystream. The size of feedback ranges from one to the block size. CTR mode functions like a stream cipher, generating keystream by encrypting the value of a counter, which is initialized by IV and incremented by one for each block. All these three modes involve only the block encryption function, which means the same function is performed at both the sides of sender and receiver.

Besides the five basic modes of operation, there are also many advanced modes of operation proposed in recent years. Some modes of operation integrate the cipher operation and MAC generation together, providing encryption, integrity and authentication in one pass of a block cipher operation. Such modes of operation include Galois/counter (GCM) mode [64], offset codebook (OCB) mode [65], and counter with CBC-MAC (CCM) mode [27]. In [66], the advanced modes of operation are evaluated through the implementation in a sensor node according to the code size and the number of CPU cycles. Among these modes, OCB mode is the most popular one, which is proposed to be used by many cryptographic schemes for WSNs [67][68][69].

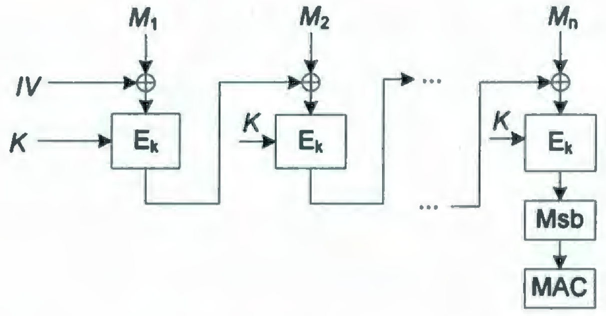


Fig. 2.9. CBC-MAC generation.

In a cryptographic scheme, a MAC can provide both data integrity and data authentication simultaneously. A typical CBC-MAC is illustrated in Fig. 2.9. Using a

MAC instead of a checksum is necessary in a WSN since both the channel quality and a malicious attack can influence the correctness of a received message. At the receiver side, to eliminate the potential possibility of malicious attack, a packet will be dropped if the recalculated MAC value does not equal the one transmitted.

2.2.2 Cryptographic Schemes

From the cryptographic scheme point of view, there are two separate stages: key establishment and normal communication with the established key. We will discuss these two stages respectively.

1) Secure Data Transmission

The normal data transmission constitutes the major secure communication process in WSNs. Compared to other networks, there are no obvious protocol stack layers in WSNs and the data transmission of a sensor is typically protected at the link layer. Most schemes proposed for secure data transmission utilize symmetric key cryptographic algorithms due to the limited resources provided by a sensor device. As security requirements for a sensor node, the data transmitted needs to be protected from being eavesdropped and altered. Although there exist energy efficient lightweight communication protocols like Zigbee, the optimization of security schemes for WSNs is especially needed from the energy consumption perspective.

- **SPINS**

SPINS is a suite of security protocols proposed for WSNs with consideration of optimizing energy cost for a sensor device [62]. SPINS includes two parts: secure network encryption protocol (SNEP) and micro timed efficient streaming loss-tolerant authentication (μTESLA). It is worth noting that in [62] there is no specific discussion of the implementation details.

SNEP is used to provide data confidentiality, authentication, integrity and freshness by using cipher RC5 with a counter mode of operation. For data encryption, the encryption result C is obtained by $C = E_K(D)$, which means using encryption key K to encrypt the plaintext data D. For MAC generation, the MAC value M is calculated by $M = E_{K_{MAC}}(C|E)$, which means using the MAC key K_{MAC} applying CBC-MAC on the concatenated message $C|E$. In this scheme, the counter value is incremented after each message, such that different ciphertexts will be transmitted even for the same message. At the receiver, if the recalculated MAC value passes the verification, the identity of the sender is verified. The counter in the MAC can prevent the attacker from replaying the message and it is maintained by periodic synchronization between the transmitter and receiver.

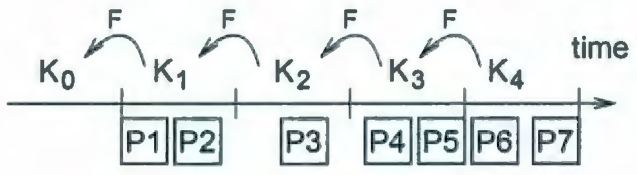


Fig. 2.10. Authentication flow for μTESLA.

μTESLA is an authentication scheme for broadcasting, which introduces the idea of delayed disclosure of symmetric keys. The authentication flow is shown in Fig. 2.10, which is based on the loose time synchronization between the base station and nodes. The key for generating the MAC is produced by a public one-way function $K_i = F(K_{i+1})$ (such as by MD5). A message encrypted by key K_i cannot be authenticated as soon as it is received. The node needs to buffer the message until the key K_{i+1} is broadcasted after a period of time, then using K_{i+1} to calculate K_i so that the message can be authenticated.

- **TinySec and Its Derivatives**

TinySec [63] is another popular encryption scheme proposed for link layer security in WSNs. TinySec presents all the specifics for implementation and shows the implementation results tested on the sensor node Mica2. In this scheme, block cipher Skipjack is used with CBC mode of operation and the initialization vector (IV) used for encryption is transmitted within each packet. To reduce the packet size transmitted by a sensor node, TinySec decreases the valid size of IV to 16 bits and utilizes the ciphertext stealing technique of CBC mode of operation.

There are many encryption schemes proposed after TinySec for better performance, such as MiniSec [67], FlexiSec [68], ContikiSec [69], and SenSec [70]. The improvements mainly focus on different aspects of cryptography, such as the cryptographic algorithm, the mode of operation, and the IV distribution. SenSec focuses on using Skipjack-X instead of Skipjack for better security and using XCBC mode for one-pass encryption and authentication [70]. MiniSec combines the characteristics of SPINS and TinySec, and utilizes several bits that are borrowed from the field of packet header to transmit fewer bits of the counter value. ContikiSec focuses on designing an operating system, which uses a valid 16-bit IV (the same as TinySec) and OCB mode of operation (the same as MiniSec). FlexiSec focuses on the configurability and provides nine security modes for different application contexts, such as only authentication without encryption, encryption with 64-bit MAC or 32-bit MAC and so on.

In Table 2.4, we briefly summarize the link layer security schemes proposed for WSNs. From this table, we can see that a cryptographic scheme is a combination of different aspects of cryptography, which need to be carefully designed to achieve the requirement of energy efficiency in a WSN.

Table 2.4. Characteristics of secure data transmission schemes.

Scheme	Year	Cipher	Mode	MAC	Sensor nodes
SPINS	2001	RC5	CTR	CBC	Mica2
TinySec	2004	Skipjack	CBC	CBC	Mica2
SenSec	2005	Skipjack-X	CBC-X	CBC-X	Mica2
MiniSec	2007	Skipjack	OCB	OCB	Telos motes
ContikiSec	2009	AES	OCB	OCB	MSB-430
FlexiSec	2009	XXTEA or AES	OCB	OCB or CBC	Mica2

2) Secure Key Distribution

Keys establishment in WSNs is important since the security in distributing the encryption key provides the basic insurance for secure data transmission. Currently, there is much research focusing on this area and proposing schemes for key establishment and management. These schemes can be divided into centralized schemes and distributed schemes based on whether there is a key distribution centre in the network or not. In addition, key establishment schemes can be divided into deterministic schemes and probabilistic schemes, according to whether there is a key ring preloaded in the sensor node before its deployment or not [1].

Most research about key establishment in WSNs focuses on symmetric key cryptography due to its lower computational cost. However, there are inevitable disadvantages introduced by symmetric key cryptography for key establishment, such as more complicated distribution process and management. Localized encryption and authentication protocol (LEAP) is a typical key establishment scheme in WSNs, which is a distributed and deterministic scheme using symmetric key cryptography [71]. LEAP includes four types of encryption keys, which are explained in Table 2.5 and these keys are used for different purposes. One of the major advantages for LEAP is that it can extremely constrain the attacking range. A compromised node can only construct relationships with its previous neighbors. As a result, a sensor node will not be accepted as a neighbor by other nodes if it was deployed in other places. Therefore, the range under attack will be effectively constrained to its previous neighbors.

Table 2.5. Four types of keys in LEAP.

Key type	Description
Individual key	Shared between the node and the base station, which is unique and preloaded in the sensor node before deployment
Pair-wise key	Shared between one-hop neighbor nodes
Cluster key	Shared among multiple neighbor nodes for secure locally broadcast
Group key	Used for base station broadcasts messages and shared among all the nodes.

Asymmetric key cryptography (e.g. RSA and ECC) has obvious advantages for key establishment, such as easier key management and more reliable identity authentication. However, they are not recommended for WSNs applications due to the large computational energy cost. Some research studies the possibility of using public key cryptography for key establishment in a WSN under certain assumptions. For example, in [72], the authors perform an analysis of the energy cost for cryptographic key establishment and results show that the total energy cost using ECC can even achieve better result than using symmetric key cipher when the key needs to be established in an environment of multi-hop communication. However, there is unfairness in the comparison in that the total energy cost used in the comparison includes the energy cost of all the nodes participating, even the base station which does not generally suffer from the same harsh energy constraints as the sensor nodes. In addition, this analysis does not consider the effects of a noisy channel.

2.2.3 Secure Data Aggregation

Since the energy shortage is a major problem in WSNs, data aggregation is widely utilized, offering an efficient way to enhance the transmission efficiency by reducing redundant message information. Although the process of aggregation introduces extra computational energy cost, energy is saved by reducing the number of data bits transmitted since communication consumes much more energy than computation. Another reason for implementing aggregation is that, from the information entropy point of view, transmitting all the raw data to the base station through the network may be not efficient, because sensor nodes are typically densely deployed, data gathered from nearby nodes may have great correlations or even overlap each other. Moreover, besides energy cost savings, reducing the redundancy can also reduce the network traffic and avoid data congestion. The functions for data aggregation in WSNs includes sum, average, median, minimum, maximum and count, which are described in Table 2.6.

Table 2.6. Function of data aggregation.

Function	Description
Sum	$f(s_1, s_2, \ldots, s_n) = \sum_{i=1}^{n} s_i$
Average	$f(s_1, s_2, \ldots, s_n) = (\sum_{i=1}^{n} s_i)/n$
Median	$f(s_1, s_2, \ldots, s_n) = s_{(r)}$; $r = \frac{n+1}{2}$; $s_{(1)}, s_{(2)}, \ldots, s_{(n)}$ is a sorted order of $s_1, s_2, \ldots, s_n$
Minimum	$f(s_1, s_2, \ldots, s_n) = Min\,\{s_i \mid i = 1,2,\ldots,n\}$
Maximum	$f(s_1, s_2, \ldots, s_n) = Max\,\{s_i \mid i = 1,2,\ldots,n\}$
Count	$f(s_1, s_2, \ldots, s_n) = \mid \{s_i \mid i = 1,2,\ldots,n\} \mid$

Security plays an important role for data aggregation in WSNs since the whole data sent from sensor nodes are ruined if an aggregator is compromised. Recent research focuses on two types of aggregation flow: hop-by-hop and end-to-end [73]. The difference between the two flows is focused on the data aggregation process as shown in Fig. 2.11. In hop-by-hop flow, the aggregation is processed after decrypting the data received from each node and then the aggregation data is encrypted again and transmitted to the base station. In contrast, end-to-end flow does not include decryption. The aggregation is processed on the collected ciphertext directly, which is done by using a cryptographic algorithm with characteristics of private homomorphism (PH). PH includes two types: additive PH and multiplicative PH, which are given by $a + b = D_k(E_k(a) +$

$E_k(b)$) and $a \times b = D_k(E_k(a) \times E_k(b))$, respectively. Some cryptographic algorithms have the PH feature, such as RSA and ECC-ElGamal for asymmetric key cipher, and Domingo-Ferrer [74] and HSC [75] for symmetric key ciphers.

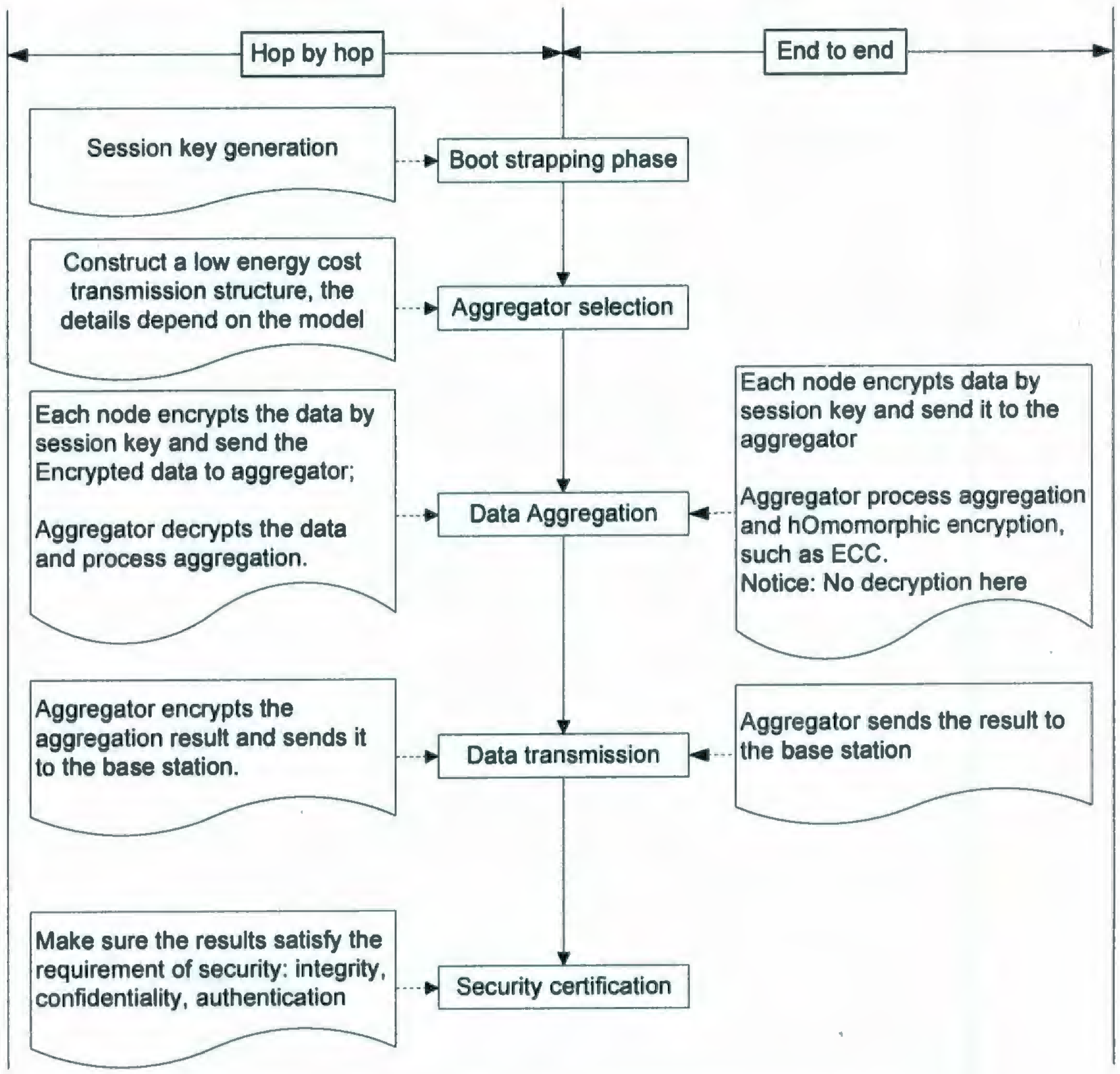

Hop by hop
End to end
Session key generation
Boot strapping phase
Construct a low energy cost transmission structure, the details depend on the model
Aggregator selection
Each node encrypts the data by session key and send the Encrypted data to aggregator;
Aggregator decrypts the data and process aggregation.
Data Aggregation
Each node encrypts data by session key and send it to the aggregator
Aggregator process aggregation and hOmomorphic encryption, such as ECC.
Notice: No decryption here
Aggregator encrypts the aggregation result and sends it to the base station.
Data transmission
Aggregator sends the result to the base station
Make sure the results satisfy the requirement of security: integrity, confidentiality, authentication
Security certification

Fig. 2.11. Data aggregation flow: hop-by-hop vs. end-to end.

In recent years, many schemes have been proposed for secure data aggregation. We list some typical ones in Table 2.7.

Table 2.7. Schemes of data aggregation in WSNs.

Protocol name	Feature
Secure Data Aggregation (SDA) [76]	SDA does not provide confidentiality and use the second hop for authentication.
Secure Information Aggregation (SIA) [77]	SIA includes only one aggregator and implement Merkle hash tree to committed value.
Secure Reference-Based Data Aggregation (SRDA) [78]	SRDA only support confidentiality and just send differential value.
Secure Hierarchical Data Aggregation (SHDA) [79]	SHDA is similar to SIA, except including multiple aggregators. Merkle hash tree is also used.
Secure hop-by-hop Data Aggregation Protocol (SDAP) [80]	SDAP increases the number of aggregators and using logical sub-tree for committed values.
Concealed Data Aggregation (CDA) [81]	CDA only provide confidentiality, using end-to-end flow by PH method.

2.3 Summary

In this chapter, we have reviewed cryptographic protocols for popular networks, such as the Internet for wired network, and WLANs, WPANs, and MANETs for wireless networks. We especially focused on reviewing the state-of-the-art work for security schemes applied to WSNs. We discussed the different aspects of cryptography applied to WSNs, such as the cryptographic algorithm and cryptographic schemes. We also introduced secure data aggregation, a popular architecture in secure WSNs to achieve better energy efficiency. We can see that, for different network requirements, the corresponding security schemes are different. The Internet focuses on security schemes at different protocol layers. WLANs focus on the data safety by continuously enhancing the security level of the protocols (from WEP, WPA, to WPA2). WPANs focus on low complexity and low power by using a chip to integrate both the security scheme with low power transceiver (such as Bluetooth and Zigbee). MANETs focus on the security of routing information due to its dynamically changing topology. Unlike these types of networks, WSNs focus on using energy efficient cryptographic schemes with suitable security since the constrained energy is a large problem for a sensor device.

Chapter 3

METHOD TO EVALUATE ENERGY COST IN SECURE WSNS

This chapter presents the method of evaluating energy cost in a sensor node when cryptography is applied to WSNs. In applications requiring security, a cryptographic algorithm is used to encrypt the data in order to avoid malicious attacks such as message eavesdropping and altering. However, at the same time, cryptography introduces more energy cost, not only for the computational cost brought in by data processing, but also for the communication cost brought in by the cryptographic schemes. The challenge is for the sensor node to transmit successfully as much information as possible in the sensor node's limited life span. Equivalently, it is desirable to minimize the energy consumed for each successfully transmitted bit of information to maximize the life span of the sensor node. It is especially important to consider this objective in the context of the communication environment, which is noisy due to the wireless medium, resulting in data packets inevitably being transmitted with errors.

In this chapter, we aim to present the method to evaluate the energy cost of a secure sensor node, including both the computational cost and communication cost. We also investigate a case including no security features, which provides a baseline for further energy performance investigation. We will focus on (1) evaluating the energy cost of a sensor node and (2) investigating different factors that affect the energy efficiency of a sensor node, such as the payload size and the channel quality.

3.1 Cryptography Implemented in WSNs

In many WSNs, the confidentiality of data is often critical since the information transmitted may relate to a disaster alarm (such as earthquake and forest fire) or private information (such as the health condition of a human being). For this purpose, we focus on exploring the cryptographic algorithm and scheme for the basic communication behaviour of a sensor node: encrypting the sensor data and transmitting the ciphertext out. As we know, symmetric key cryptographic algorithms include two types of ciphers: stream ciphers and block ciphers. A stream cipher typically operates on one bit of data each time by XORing the generated keystream with plaintext; while, a block cipher operates on a block size of data (typically 64 bits or 128 bits) by iterating rounds of simple cryptographic operations, such as nonlinear substitution and linear transformation. The same cipher key is used for both the encryption and decryption, which is assumed already known by the sensor node. This cipher key can be embedded inside the sensor node device before its deployment or be established by a specialized key agreement at the beginning of the communication. In WSNs, encryption is used to achieve the confidentiality between two communicating sides, preventing the potentially sensitive information from being obtained by an inappropriate party. At the same time, a message authentication code (MAC) [61] is transmitted in a packet, so that the receiver side can use it to judge the validity of a packet.

3.2 Performance Comparison Metric: Energy Efficiency

In a WSN, energy and security are two key considerations. Although security is a design goal, it is not practical to evaluate a cryptographic algorithm or scheme by taking the security level as a metric. Although security schemes can be identified to have weaknesses, such flaws are not always evident or easily quantifiable. Hence, we shall assume that schemes using accepted cryptographic methods with reasonable block and key sizes are secure, and the metric we shall use to evaluate the cryptographic performance in a WSN is based on the energy consumption. We choose as a metric the amount of valid information successfully transmitted from the sensor node per Joule of energy consumed in the transmission process and refer to this metric as *energy efficiency*. Because our focus is security, we consider only the energy costs of the security scheme in a sensor node and ignore the energy requirement of other types of processing.

The valid information successfully transmitted refers to the data in packets that is received and decrypted correctly. That is to say, if a packet is transmitted with a corruption of its start symbol or packet header, the packet is not taken as valid since the receiver will not receive it. As well, if other fields of the packet are corrupted, the MAC verification will fail and the data will be discarded and considered not valid. Many factors will influence the energy cost of valid data bits transmitted by a sensor node. In the following sections, we will analyze it from both the perspective of the cryptographic algorithm and the applied cryptographic scheme.

3.3 Energy Cost Calculation

In WSNs, the energy cost of a sensor device mainly consists of two parts: the communication cost and the computational cost. For communication energy cost, we consider the transmitting and receiving energy cost; while for computational cost, we consider the encryption cost and MAC calculation cost. For the purposes of our analysis, some types of energy are ignored, such as the energy cost when sensor is in the sleep mode, the computational costs of data processing other than encryption and MAC generation, etc..

3.3.1 Sensor Node Primitive Description

In our research, the energy cost calculation references to the sensor node structure of the Mica series [82], which are representative products that have been studied and used widely in recent years. However, it is worth noting that analysis results will be similar when applied to other sensor products.

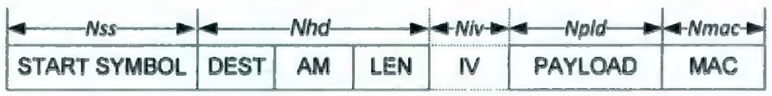


Fig. 3.1. Typical packet format for a sensor node.

The major work of a sensor node includes sensing data from the environment, processing the collected data by a microprocessor (for example, the 8-bit Atmel Atmega128 CPU), and transmitting messages out with small size packets (typically not larger than 30 bytes). Fig. 3.1 shows a typical packet format of a sensor node derived from TinySec [63], and its notation is explained in Table 3.1. In this figure, the IV can be transmitted with the payload or be transmitted independently, determined by the specific cryptographic scheme (which will be explicitly explained in the following chapters).

Table 3.1. Notation used in the packet format.

Symbol	Size (bits)	Description
START SYMBOL	N_{ss}	Start symbol used for medium access.
DEST	N_{hd} (sum)	Destination address of the receiver
AM		Active message handler type
LEN		Size of the packet
IV	N_{iv}	Initialization vector
PAYLOAD	N_{pld}	Payload, usually variable
MAC	N_{mac}	Message authentication code

3.3.2 Communication Energy Cost

The energy cost of transmitting one packet (E_{tx}) depends on the current in transmitting mode (I_{tx}), the voltage (U), the size of the packet (N_{pkt}), and the bit rate of transmission (R), and is given by

$$E_{tx} = (I_{tx} \times U \times N_{pkt})/R \; . \tag{3.1}$$

Similarly, the energy cost of receiving one packet (E_{rx}) can be expressed as

$$E_{rx} = (I_{rx} \times U \times N_{pkt})/R \; , \tag{3.2}$$

where I_{rx} is the current in receiving mode.

3.3.3 Computational Energy Cost

As we discussed before, a sensor node is manufactured to be low power and low cost. Hence, most devices use an 8-bit microprocessor for data processing, such as one from the Atmel AVR ATmega series [83]. The computational energy cost by the microprocessor to process a particular operation (such as a block encryption) can be calculated as

$$E_{op} = P_{cpu} \times T_{op} = P_{cpu} \times C_{op}/f_{cpu}, \tag{3.3}$$

where P_{cpu} represents the CPU's power and T_{op} represents the time to execute the operation which can be further calculated by the CPU's frequency (f_{cpu}) and the number of operation cycles being used (C_{op}).

When cryptography is applied to a WSN, we mainly consider two parts of computational energy cost: encryption and MAC generation. We calculate the energy by determining the number of CPU cycles used to finish the cryptographic processing. Two factors affect the computational energy cost: the cipher algorithm efficiency and the payload size. When considering the use of a symmetric key cipher, the block size is an important parameter in the energy cost calculation, as this determines how many encryption operations are carried out. From the perspective of the sensor node, whose main function is to transmit collected sensor information, the encryption and MAC processing energy costs per packet (E_{enc} and E_{mac}, respectively) are given by:

$$E_{enc} = (P_{cpu} \times C_{enc}/f_{cpu}) \times \lceil N_{pld}/b \rceil, \tag{3.4}$$

$$E_{mac} = (P_{cpu} \times C_{mac})/f_{cpu} \times \lceil (N_{pkt} - N_{ss} - N_{mac})/b_{mac} \rceil, \tag{3.5}$$

where C_{enc} and C_{mac} represent the number of clock cycles required to encrypt one block and generate MAC value, respectively and N_{pld} represents the payload size in bits. For a block cipher, b is the block size. While for a stream cipher, b represents the keystream block size, which is the amount of keystream produced at one time. The symbol $\lceil . \rceil$ denotes the ceiling operator. Note that we are assuming that the MAC is produced using a block cipher of block size b_{mac} in CBC mode [61] and is applied across all fields of the packets except the start symbol.

3.4 A Case Study: An Unsecured WSN

In this section, we focus on evaluating the energy performance of a case that a wireless sensor node transmits data without encryption and leave the an alysis of secure data transmission for the following chapters. In this case, we especially consider different factors affecting the energy cost of a sensor node such as the payload size and the channel quality.

3.4.1 Energy Cost Calculation

The packet format is shown in Fig. 3.2, including a start symbol, packet header, payload and a CRC checksum. The energy cost of a sensor node is calculated as the sum of the communication and computational energy costs.

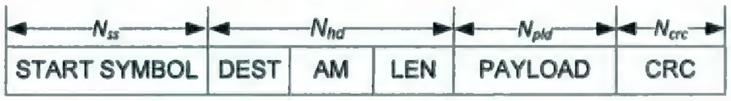


Fig. 3.2. Packet format for a sensor node without encryption.

The energy cost of transmitting one packet can be calculated using (3.1), which can be expressed as

$$E_{tx} = \left(I_{tx} \times U \times (N_{ss} + N_{hd} + N_{pld} + N_{crc})\right)/R. \tag{3.6}$$

Since the CRC is transmitted within the packet, the computational cost of calculating the CRC value by the microprocessor should be included. Using (3.3), we can obtain the computational energy cost for one data packet:

$$E_{crc} = (P_{cpu} \times C_{crc}/f_{cpu}) \times \lceil (N_{hd} + N_{pld})/b_{crc} \rceil, \tag{3.7}$$

where C_{crc} represents the number of cycles to compute the CRC operation per block of size b_{crc}. For the assembly implementation result provided by Atmel AVR [84], C_{crc} = 175 cycles and b_{crc} = 16 bits.

3.4.2 Channel Quality Consideration

For the unsecured network, although we consider the case that no malicious attacks are applied in the network, there is still a chance that bits of the packet are corrupted since it is transmitted through the noisy wireless media. As a result, information transmitted within the packet is sometimes not valid (as indicated by a failed CRC check, here we assume CRC detects all the errors).

The probability that a packet has one or more bit errors is decided by two factors: the probability of error for each bit (p_e) and the size of the packet (N_{pkt}). The probability that a received packet has no errors (P_o) is expressed as

$$P_o = (1 - p_e)^{N_{pkt}}, \tag{3.6}$$

where it is assumed that bit errors occur randomly and independently. By using this, we can evaluate the amount of valid information successfully transmitted by a sensor node.

3.4.3 Calculation of Valid Data Transferred

The total number of data bits transmitted successfully for a given energy cost (e.g. 1 Joule) is given by

$$\eta = n \times N_{pld} \times P_o \, , \tag{3.7}$$

which depends on three factors: the number of the transmitted data packets, n (which can be calculated by a given energy cost 1 Joule), the payload size, N_{pld}, and the probability that the packet is transmitted correctly, P_o.

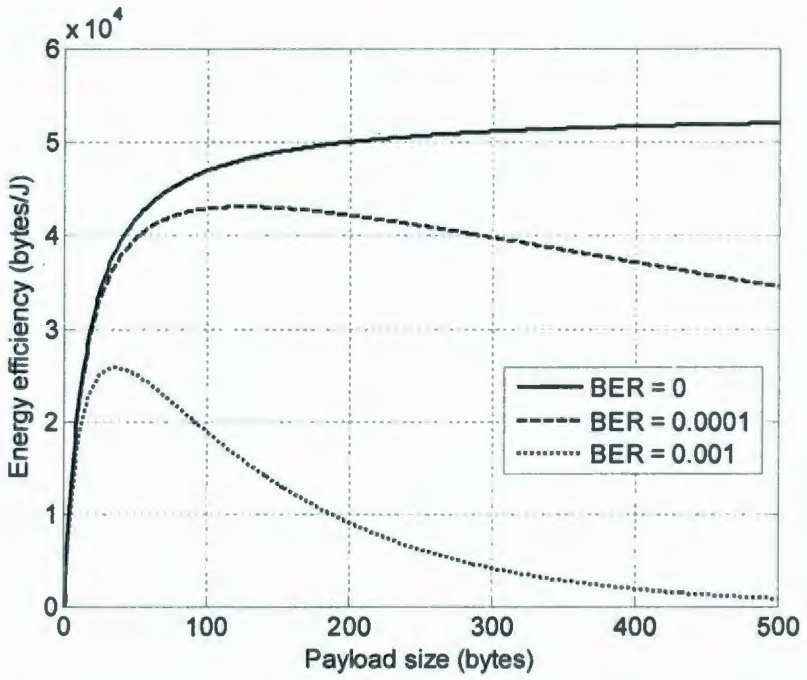


Fig. 3.3. Analysis of the valid data in wireless channel without encryption.

3.4.4 Analysis Result

We plot the energy efficiency for a fixed size packet for the scenarios of an error-free channel and a noisy channel. The analysis result is shown in Fig. 3.3, where the parameters are based on the Mica sensor node and are listed in Table 3.2. Notice that we analyze the energy performance for a wide range of payload sizes. However, the sensor node typically transmits small size packets of less than 30 bytes. We extend the payload size to give a general perspective on the energy efficiency, which is affected by the channel quality and the payload size.

Table 3.2. Parameters used in this analysis.

Object	Parameter	Value	Unit
CRC	C_{crc}	175	cycles
	b_{crc}	16	bits
Sensor board	P_{cpu}	13.8	mW
	f_{cpu}	8	MHz
	I_{xmt}	27	mA
	U	3.3	V
	R	38400	bps
Packet	N_{ss}	8	bytes
	N_{hd}	4	bytes
	N_{pld}	from 1 to 500	bytes
Channel	BER	10^{-4}, 0	-

As shown in the analysis result, for an error-free channel the energy efficiency first increases rapidly with the increase of the payload size, and then the rate of increase decreases and approximates to a fixed value. This trend can be explained by the overhead of the data packet. Since the channel is error-free, the larger payload size means relatively less energy consumed by transmitting the overhead of the packet.

Consider now the cases for a noisy channel. Unlike the error-free channel, there is a maximum value for the energy efficiency, which is obtained at a certain payload size. When the payload size increases to very large values the energy efficiency decreases. This is the result of a balance between the packet size and the quality of the channel. When the channel is noisy, larger packet size will lead to larger probability of the data packet transmitted with error, which means more data bits transmitted are not valid and are discarded due to the failed CRC check of the reviewer. However, the invalid data packet also consumes energy thereby decreasing the sensor node's energy efficiency.

3.5 Summary

In this chapter, we first introduce the application of cryptography to a wireless sensor network, and then propose using energy efficiency, the amount of valid data transferred per Joule from a sensor node, as the metric to evaluate the performance when cryptography is applied to the sensor node communication. We give an explicit explanation for the energy cost of a sensor node including computational cost and communication cost, which is the major factor determining the life span of a sensor node. Finally, we study a case by analyzing the energy cost of a sensor node under the condition of a WSN without considering security. We focus on calculating the energy efficiency for a sensor node, and analyzing the factors, which affect the number of valid data bits transmitted, such as the payload size and the channel quality. Results show that in a noisy channel, there exists a balance between the payload size and the quality of the channel and the energy efficiency is affected by both these factors. We shall use this basic case as a point of comparison when considering the energy efficiency of cryptographic schemes applied to WSNs in the subsequent chapters.

Chapter 4

Software-oriented Energy Cost of Cryptographic Algorithms

This chapter presents the energy cost evaluation for cryptographic algorithms implemented in a software-oriented sensor node. Selection of a suitable security scheme is critical in wireless sensor networks (WSNs) because of the open media broadcast communication and the limited energy supply of the sensor device [1]. To achieve the security requirements, several researchers have focused on evaluating cryptographic algorithms in WSNs [35][85] and proposing energy efficient ciphers [86][87]. Although the transmission of data is the most energy consuming activity in a wireless sensor node, it is also important to select an energy efficient cipher that will minimize the energy consumption of the energy constrained sensor node.

In this chapter, we examine the energy efficiency of symmetric key cryptographic algorithms applied in wireless sensor networks (WSNs). In our study, we consider both stream ciphers and block ciphers. We implement the ciphers in software using assembly language and derive the computational energy cost of the ciphers under consideration by comparing the number of CPU cycles required to perform encryption. After evaluating a number of symmetric key ciphers, we compare the energy performance of stream ciphers and block ciphers applied to a noisy channel in a WSN. From the analysis, we recommend using a lightweight block cipher referred to as byte-oriented substitution-permutation network (BSPN), to achieve energy efficiency with a level of security suitable for wireless sensor networks.

4.1 Software Implementation of Symmetric Key Ciphers

In this section, we study different structures of symmetric key ciphers and evaluate their energy efficiency. We choose some typical stream ciphers and block ciphers to implement in assembly language based on an Atmel AVR microprocessor ATmega128. From the software implementation, we can obtain the number of CPU cycles and calculate the computational energy cost for a sensor node, which i s making use of cryptography.

4.1.1 Platform of Software Implementation

- **ATmega128 and Simulation Tool**

Atmel ATmega128 [83] is an 8-bit microprocessor with low power and low cost, and it has been utilized by many sensor node devices, such as Mica series. It works at a clock frequency of up to 8MHz and has 4 KB of RAM and 128 KB of flash memory. There are 32 x 8 general-purpose working registers and 133 instructions, most of which are executed in one cycle. AVR Studio is the software simulation tool for Atmel AVR microprocessors and supports both C language and assembly language. AVR Studio provides the information of the resource utilization and the number of CPU cycles required to execute code.

- **C language vs. Assembly Language**

Although C language is easier to maintain and read, we choose assembly language to implement the cryptographic algorithms. Since the sensor node is energy-limited device, fewer the CPU cycles it operates, less the computational energy is consumed. Assembly language can especially satisfy this purpose because it offers a direct control of the CPU. Although C language can be compiled with optimization, it still results in a large number of redundant cycles. By using assembly language, we can precisely control the flow and fully utilize features provided by the hardware.

4.1.2 Implementation Considerations

Unlike hardware implementation, software implementation executes instructions in a relatively sequential manner and the specific implementation using assembly language directly f ollows the algorithm itself. T he efficiency of a software implementation is mainly based on the resources of microprocessor and its instruction set.

Implementation depends on the specifics of the cryptographic algorithm. Different tradeoffs are possible to make the operation more efficient. For example, the utilization of a lookup table approach can often save a number of CPU cycles since the value can be obtained directly by accessing memory instead of slow and long CPU calculation. Sometimes even multiple lookup tables are generated to cater to this purpose. However, in a sensor node, the resources of the microprocessor are limited. Considering this situation, we only use the lookup table approach for the S-box so that the sensor node can use memory for other purposes.

4.1.3 Implementation of Block Ciphers

We choose four block ciphers as candidates and consider the energy performance when applied to WSNs. These ciphers are chosen due to their different characteristics: AES [10] represents the most popular block cipher being used in recent years; Skipjack [40] is chosen for use in a commercial sensor product; as two lightweight block ciphers, Puffin [51] is designed for the hardware implementation purposes and BSPN [89] is efficient for 8-bit CPU operation.

1) AES

AES [10] is the most popularly deployed symmetric key cipher. Although, as we shall see, the energy cost per byte of AES is high, it is generally regarded as a secure choice when selecting ciphers for security schemes.

AES is a block cipher with 128-bit block size and 128-bit (or larger) key size. The structure of AES is illustrated in Fig. 4.1, which includes ten rounds of operation and each round with four stages. The block of plaintext is divided into four rows and four columns with each byte as an element. The four stages of one round operation include substitute bytes, shift rows, mix columns and add round keys. In the first stage, the block is substituted using an 8×8 S-box byte by byte. In the second stage, a permutation is operated on each row of the block. Then in the third stage, a linear transformation is operated column by column by multiplying a matrix. Finally, the round key is XORed with the result of the mix column stage.

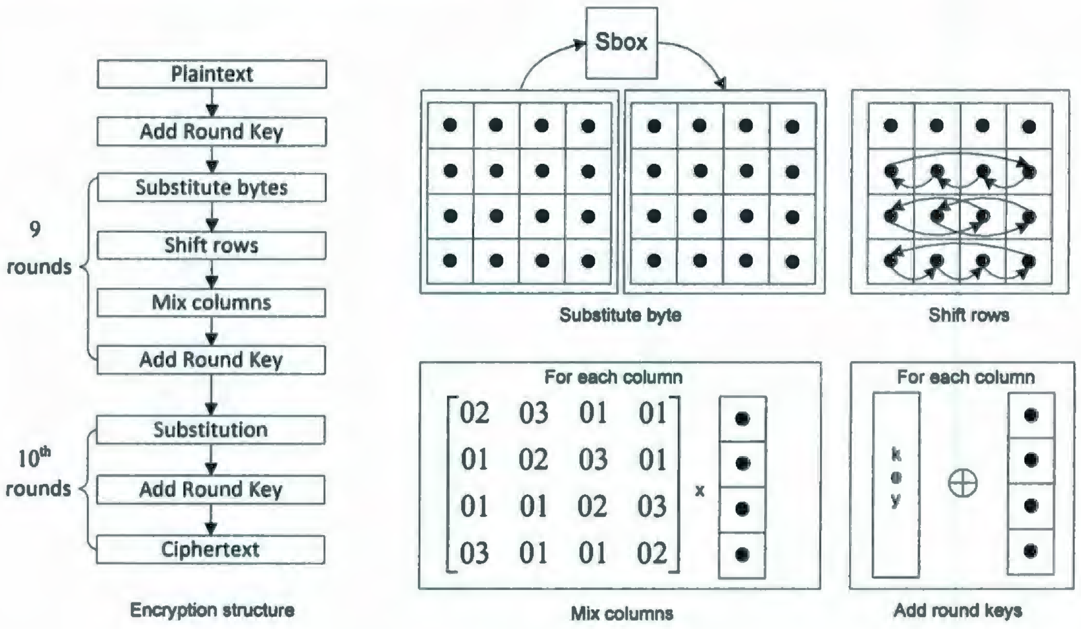


Fig. 4.1. AES encryption structure.

- **Implementation Result**

The implementation of AES is an efficient 8-bit implementation based on using the "xtime" operation for mix columns [93]. The software implementation result for AES is shown in Table 4.1. In this implementation, the round keys are generated first and stored in the memory.

Table 4.1. Implementation result of AES.

Resource Utilization (bytes)	Code	Data	Total Used	Size	Use%
	1042	272	1314	131072	1.0%
Encryption Timing	3266 cycles per block				

2) Skipjack

Skipjack [40] is designed to take the place of the Data Encryption Standards (DES) [9] and was first designed in 1987 and then declassified in 1998. It is utilized for WSNs in the TinySec scheme [63] due to its energy efficiency. However, some research has shown that Skipjack has security weakness under certain cryptanalyses [91][92]. Skipjack is a block cipher with 64-bit block size and 80-bit key size. It uses a Feistel structure and includes 32 rounds of operation. Within the 32 rounds, a data block is encrypted iteratively by rule A and rule B, which are slightly different in structure as shown in Fig. 4.2. In this figure, B_i and K_i represent the i^{th} plaintext byte and cipher key byte, respectively, and B_H and B_L represent the higher byte and lower byte, respectively. G is a permutation and F is a substitution using an 8×8 F-table (the same as an 8×8 S-box).

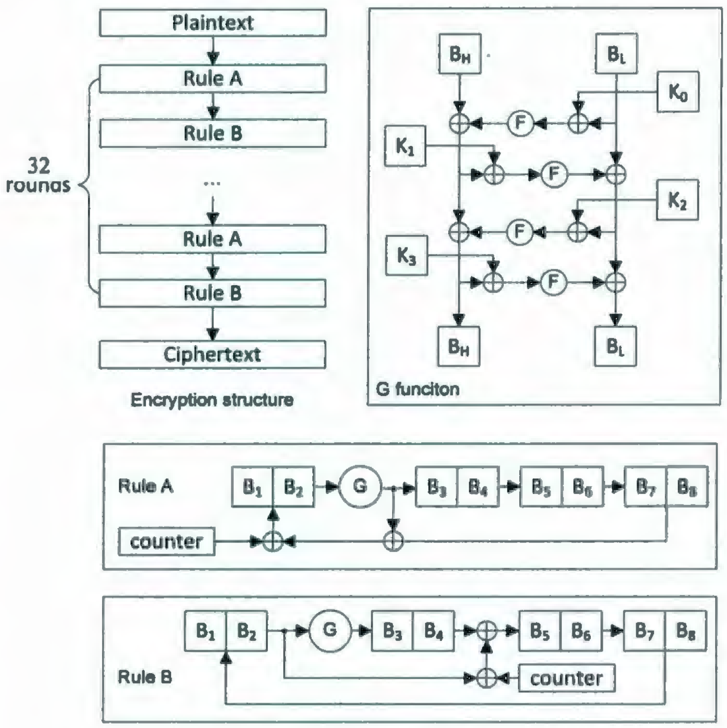


Fig. 4.2. Skipjack encryption structure.

- **Implementation Result**

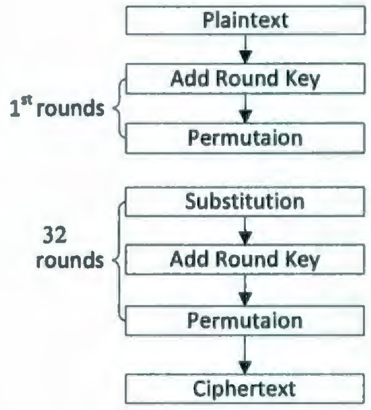
The software implementation result for Skipjack is shown in Table 4.2. In this implementation, the round keys are generated first and stored in the memory.

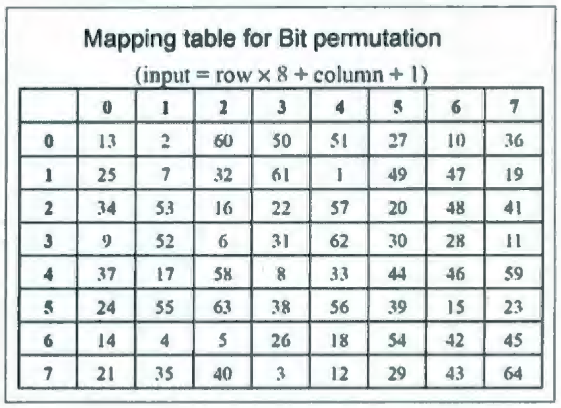
Table 4.2. Implementation result of Skipjack.

Resource Utilization (bytes)	Code	Data	Used	Size	Use%
	2426	282	2708	131072	2.1%
Encryption Timing	1482 cycles per block				

3) Puffin

Puffin [51] is a recently proposed compact block cipher designed for hardware implementations. Puffin can resist differential and linear cryptanalysis and it is resistant to related-key attacks and weak keys, which are two main insecurities of the key schedule. Puffin is a block cipher of 64-bit block size and 128-bit key size. There are 33 rounds of operation and each round includes three stages: substitution, add round keys and permutation. The substitution uses a 4×4 S-box to make the circuit more compact and the permutation is operated bit by bit. The encryption structure of cipher Puffin is shown in Fig. 4.3.




Mapping table for Bit permutation

(input = row × 8 + column + 1)

	0	1	2	3	4	5	6	7
0	13	2	60	50	51	27	10	36
1	25	7	32	61	1	49	47	19
2	34	53	16	22	57	20	48	41
3	9	52	6	31	62	30	28	11
4	37	17	58	8	33	44	46	59
5	24	55	63	38	56	39	15	23
6	14	4	5	26	18	54	42	45
7	21	35	40	3	12	29	43	64

Fig. 4.3. Puffin encryption structure.

- **Implementation Result**

The software implementation result for Puffin is shown in Table 4.3. In this implementation, only the cipher's encryption is included and the round keys are initialized in the memory.

Table 4.3. Implementation result of Puffin.

Resource Utilization (bytes)	Code	Data	Used	Size	Use%
	802	520	1322	131072	1.0%
Encryption Timing	43418 cycles per block				

4) BSPN

Byte-wise SPN (BSPN) is a compact block cipher we recommend for use in WSNs [89] to provide moderate security to the energy-limited environment. It has no apparent weaknesses and is resistant to both the differential and linear cryptanalysis attacks [89]. The block size of BSPN is 64 bits and the key size is originally designed as 64 bits, but can be easily expanded to 128 bits.

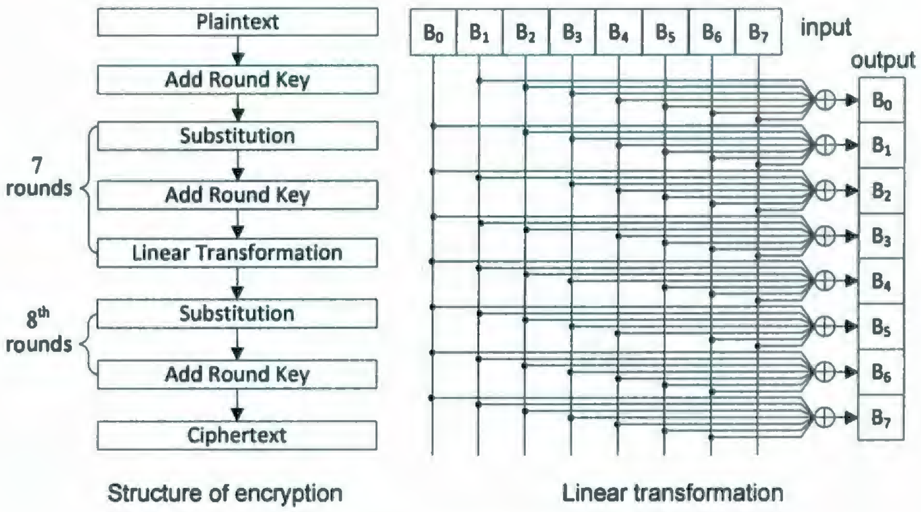


Fig. 4.4. BSPN Encryption structure.

The structure of BSPN encryption is illustrated in Fig. 4.4, which is similar to Puffin except that (1) there are 8 rounds of operation, (2) an 8×8 S-box is used, and (3) the linear transformation is based on 8-bit. In this figure, B_i represents the i^{th} byte of the data block. The structure of BSPN is especially efficient for 8-bit CPU operation.

- **Implementation Result**

The software implementation result for BSPN is shown in Table 4.4. In this implementation, the round keys are generated first and stored in the memory.

Table 4.4. Implementation result of BSPN.

Resource Utilization (bytes)	**Code**	**Data**	**Used**	**Size**	**Use%**
	514	336	850	131072	0.6%
Encryption Timing	796 cycles per block				

Table 4.5. RC4 algorithm description [59].

KSA	PRGA

for i **from** 0 **to** 255
  S[i] := i
endfor
j := 0
for i **from** 0 **to** 255
  j := (j + S[i] + key[i mod *keylength*]) mod 256
  swap(&S[i],&S[j])
endfor
	i := 0
	j := 0
	while GeneratingOutput:
	  i := (i + 1) mod 256
	  j := (j + S[i]) mod 256
	  swap(&S[i],&S[j])
	  output S[(S[i] + S[j]) mod 256]
	endwhile

4.1.4 Implementation of Stream Ciphers

1) RC4

RC4 is a high throughput stream cipher, which is widely used in security protocols such as WEP (wired equivalent privacy) and SSL (secure sockets layer) [61]. The cipher RC4 includes two components: the key-scheduling algorithm (KSA) and the pseudo-random generation algorithm (PRGA) [59]. The pseudo code is shown in Table 4.5.

- **Implementation Result**

The software implementation result for RC4 is shown in the table below:

Table 4.6. Implementation result of RC4.

Resource Utilization (Bytes)	Code	Data	Used	Size	Use%
	292	288	580	131072	0.4%
Encryption Timing	15890 cycles for keystream setup 31 cycles per byte of keystream generated				

2) Other Stream Ciphers

We also choose two other stream ciphers as candidates to compare, which are Sosemanuk [94] and Salsa [95]. We take the implementation results from [60], which are obtained by using the same platform.

4.2 Factors Affecting the Computational Cost

As we know, there are two fundamental categories of ciphers, symmetric key ciphers and asymmetric (or public) key ciphers, which use different ways to achieve security:

asymmetric cryptography depends on the difficulty of a mathematical problem and symmetric key cryptography focuses on the structure of iteratively applied simple cryptographic operations [61]. In WSNs, energy limitations make the security schemes focus on ciphers with efficient computational energy consumption. Hence, the symmetric key cipher is typically utilized to encrypt data during the transmission of sensor nodes. Before evaluating the energy consumption of different cryptographic algorithms in a WSN, we first discuss different factors that directly affect the computational energy cost. These factors can be divided into two types: (1) intrinsic nature of the algorithm structure and (2) extrinsic factor of channel quality, which influences the frequency of resynchronization. Both the intrinsic and extrinsic factors interactively affect the computational energy cost of cryptographic algorithms in a WSN.

4.2.1 Intrinsic Factors

1) Structure of the Cryptographic Algorithm

Cryptographic algorithms or ciphers vary significantly in structure as we have seen in the previous section. Many block ciphers, such as AES [10], make use of a number of rounds of operations such as substitutions (S-boxes) and linear transformations. To satisfy certain special needs, such as low circuit area and limited resources, there are also some lightweight ciphers designed with some degrees of simplicity [88]. Lightweight ciphers are designed to provide just sufficient security, however, it may not offer as much as provided by AES [10]. For example, lightweight block ciphers may use smaller block size and key size than AES. For many applications in WSNs, a sensor node is an energy-limited device transmitting low entropy information with a limited life span. Hence, lightweight ciphers with energy efficiency can also be good candidates for such application environments.

2) Size of Encryption Operands

The size of encryption operands is different for stream ciphers and block ciphers. Stream ciphers typically operate on one bit of plaintext data to produce one ciphertext bit. This is typically achieved by XORing plaintext bits with a pseudorandom sequence of bits called the keystream to produce the ciphertext bits. For practicality, blocks of plaintext bits can often be XORed with keystream bits in parallel. In contrast, block ciphers process an entire block of plaintext bits (typically, 64 bits or 128 bits) at one time to produce a block of ciphertext bits. When encrypting a large sequence of plaintext bits, stream ciphers can straightforwardly operate on variable lengths. However, block ciphers may need to pad plaintext out to have a length that is a multiple of the block size. In WSNs, the extra ciphertext bits will result in increased transmission energy cost of the sensor node.

3) Key Setup

Usually there is a key setup period included in the operation of symmetric key ciphers. For example, AES has a key expansion phase to generate round keys from the cipher key. Although the detailed operation of key setup varies for different ciphers, it can be complicated and take a relatively long time to finish. In many applications in WSNs, key setup is very infrequent since it will only follow the establishment of a new cipher key. Hence, this contribution of energy consumption is very small.

4) Keystream / IV Setup

Keystream setup or setup of the initialization vector (IV) takes place in symmetric key ciphers whenever a new IV is established. For example, stream ciphers must periodically re-initialize the keystream based on an updated IV to ensure that the transmitter and receiver are synchronized in their encryption and decryption processes, respectively. For block ciphers, all modes of operation [61], except ECB mode, use IVs, which are periodically updated to establish synchronization between encryption and decryption. When a block cipher is used in a stream cipher mode such as counter mode, the setup of the IV is equivalent to keystream setup. The energy cost of keystream setup for stream ciphers is typically much higher than that of block ciphers, and as we shall see, in a WSN, the energy cost of keystream setup is especially critical in the determination of whether to use a stream cipher or a block cipher.

4.2.2 Extrinsic Factors

In a noisy communication channel, bit errors may result in packets being lost or corrupted and the resulting decryption may lose synchronization with encryption. This can be resolved by periodic resynchronization involving the transfer of an initialization vector (IV). However, the energy cost of resynchronization will directly impact the lifetime of the sensor device and is influenced by the channel quality since poor quality channels result in the need to resynchronize frequently. In this section, we focus on resynchronization computational energy cost for different ciphers, with the communication cost of resynchronization being explicitly investigated in Section 4. The computational cost of resynchronization will be different for block ciphers and stream ciphers due to their characteristics.

1) Block Cipher Resynchronization

When using block ciphers, the computational cost of encryption can be calculated directly by the number of data blocks times the energy cost per block. That is, fixed computational energy is consumed per data block encrypted. The computational energy cost of IV synchronization for a block cipher only includes a few CPU cycles to complete operations, such as loading and storing the new IV. The computational cost of key setup for block ciphers can be more substantial. However, we assume that key setup is very infrequent and base our analysis on the assumption that the round keys used by a block cipher are already generated and stored.

2) Stream Cipher Resynchronization

For stream ciphers, two phases are included in the operation of the cipher: (1) keystream setup based on an updated IV and (2) the keystream generation and encryption of plaintext. For most stream ciphers, the computational energy cost of IV synchronization (causing keystream setup) may take a considerable proportion of the total computational energy cost. This is particularly notable for RC4, which will be presented by our analysis later.

4.3 Evaluation Considering Intrinsic Factors

Our evaluation of the cryptographic algorithms includes two parts: (1) consideration of intrinsic factors only, i.e., evaluating the symmetric key ciphers' energy performance directly; (2) consideration of extrinsic factors, which means applying the cipher to a protocol used in a noisy wireless channel. As a low cost device, a sensor node usually employs an 8-bit microcontroller to implement different kinds of operations. Hence, in our evaluation, we use the number of CPU cycles per byte to evaluate the computational energy efficiency of symmetric key ciphers. The number of cycles is obtained from an implementation of the cipher in assembly language on the ATmega128 CPU, a popular 8-bit microcontroller, used in wireless sensor devices such as Mica2 [82].

4.3.1 Block Cipher Comparison

We choose four block ciphers as candidates (which are presented in Section 4.1.3) to compare the different energy performance when applied in WSNs. In Table 4.7, we summarize the characteristics of these ciphers and their corresponding implementation results.

Table 4.7. Characteristics of block ciphers.

Block cipher	Block size	Key size	# Rounds	Cycles per block	Cycles per byte
AES	128 bits	128 bits	10	3266	204
Skipjack	64 bits	80 bits	32	1482	186
Puffin	64 bits	128 bits	32	43418	5427
BSPN	64 bits	$\geq$ 64 bits	8	796	99

From these results, we can see that BSPN requires the fewest number of CPU cycles per byte among the four block ciphers and thus has the lowest computational energy cost. The relationship between the CPU cycles and the size of plaintext is illustrated in Fig. 4.5. The total number of cycles to encrypt is based on operating the cipher in ECB mode. However, results for other modes, such as CBC mode or counter mode would be very similar. In this figure, BSPN achieves the best energy efficiency for all plaintext sizes. The number of CPU cycles of Puffin is noticeably higher than that of others because it is designed for hardware purposes. AES achieves a better result than Puffin, while slightly worse than the other two ciphers. Note that, although the performance of Skipjack is slightly better than AES, it is vulnerable to cryptanalysis. Notice that the curves show a staircase shape with fixed period, which is caused by the block size of the cipher.

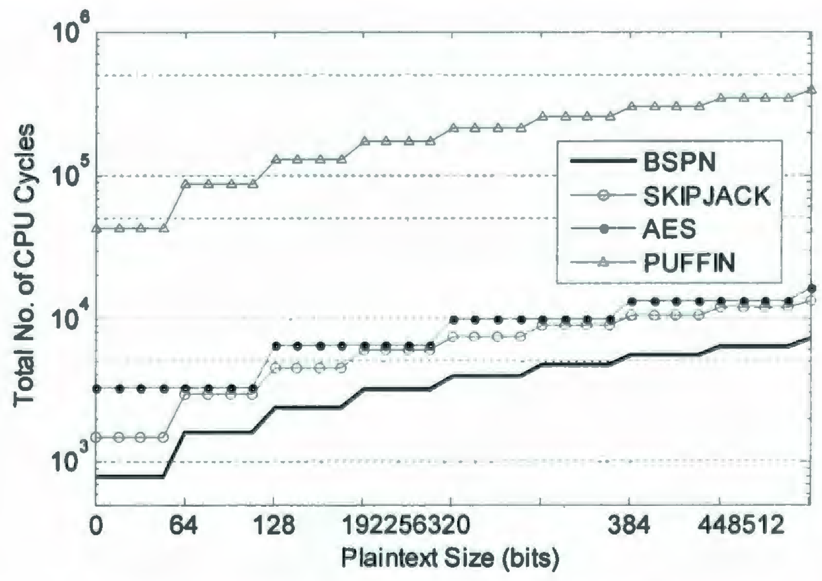


Fig. 4.5. Comparison of computational costs for block ciphers.

Since the cipher is implemented in software (i.e., in assembly language), the number of CPU cycles is directly related to the architecture of the block cipher and instruction set of the CPU. The characteristics of the block cipher's substitution and linear transformation are two critical factors affecting the number of CPU cycles. These characteristics are summarized in Table 4.8. The "Linear Trans. Unit" refers to the basic unit manipulated by the linear transformation. BSPN achieves the best energy performance among the four ciphers because of its efficiency of substitution and linear transformation for an 8 bit CPU.

Puffin, as block cipher aimed for compactness, was designed for hardware purposes making it not efficient for software implementation. BSPN's use of the 8×8 S-box makes the substitution value look-up convenient in an 8 bit CPU. In contrast, a 4×4 S-box used in Puffin requires two memory accesses to update one byte of data during the substitution. Although in Puffin, we can reconstruct the S-box to make it appropriate for the byte operation, extra memory needs to be allocated. BSPN performs a linear transformation on the data by XORing the output bytes of other S-boxes directly, which greatly reduces the complexity on an 8 bit CPU compared to the cipher Puffin, whose linear transformation is structured on a bitwise basis. Although part of the linear transformation can be optimized, it cannot change the result of taking a large number of cycles to execute.

Table 4.8. Characteristics of substitution and permutation.

Block cipher	Structure	S-box	Linear Trans. Unit
BSPN	SPN	8×8	8 bits
Skipjack	Feistel	8×8	8 bits
AES	SPN	8×8	8 bits
Puffin	SPN	4×4	1 bit

As shown in the table, both AES and BSPN have the 8×8 S-box and byte oriented linear transformation. The reason that AES costs 105 cycles more than BSPN per byte is the algorithm complexity of AES. It should be noted, however, that AES is designed for a 128-bit level of security for both key and block size, while for BSPN, it is designed for a 64-bit level of security both in key and block size.

4.3.2 Stream Cipher Comparison

We have selected three stream ciphers to compare, which are RC4 [61] (which is implemented in Section 4.1.4), Sosemanuk [94] and Salsa [95]. RC4 is a popular stream cipher generating a small size (8 bit) keystream block to XOR with 8 bits of plaintext, and Sosemanuk and Salsa are from the eSTREAM project (Profile I), which are considered secure and designed for software purposes. Although there are also two other stream ciphers in the Profile I, Rabbit and HC-128, we do not consider them for the same reasons explained in [60]: Rabbit is patented and HC-128 is too complicated to be implemented efficiently on an 8-bit CPU. Table 4.9 shows the characteristics and implementation results of the three stream ciphers, where the implementation results of Sosemanuk and Salsa are taken from [60], which uses the same platform as our implementation.

Table 4.9. Implementation results of different stream ciphers.

Stream cipher	Keystream block size	CPU cycles		Cycles per byte (encrypt)
		Setup	Encrypt	
RC4	8 bits	18787	31	31
Sosemanuk	640 bits	8739	8559	107
Salsa	512 bits	60	17812	279

From the implementation result, we can see that if we do not consider the keystream setup period, RC4 uses the fewest number of cycles to generate the keystream bytes for encryption. However, the energy efficiency of the stream ciphers is significantly influenced by the keystream setup period. The relation between the number of CPU cycles and the size of the plaintext is illustrated in Fig. 4.6, which is obtained on the assumption that the keystream setup period only happens once at the beginning of communication and then the keystream is generated continuously. In this figure, it can be seen that when the size of plaintext is smaller than 81 bytes, RC4 has the worst performance because of the larger number of setup cycles. However, after that point, RC4 achieves the best energy performance since dramatically fewer cycles are needed to generate the keystream bytes for encryption.

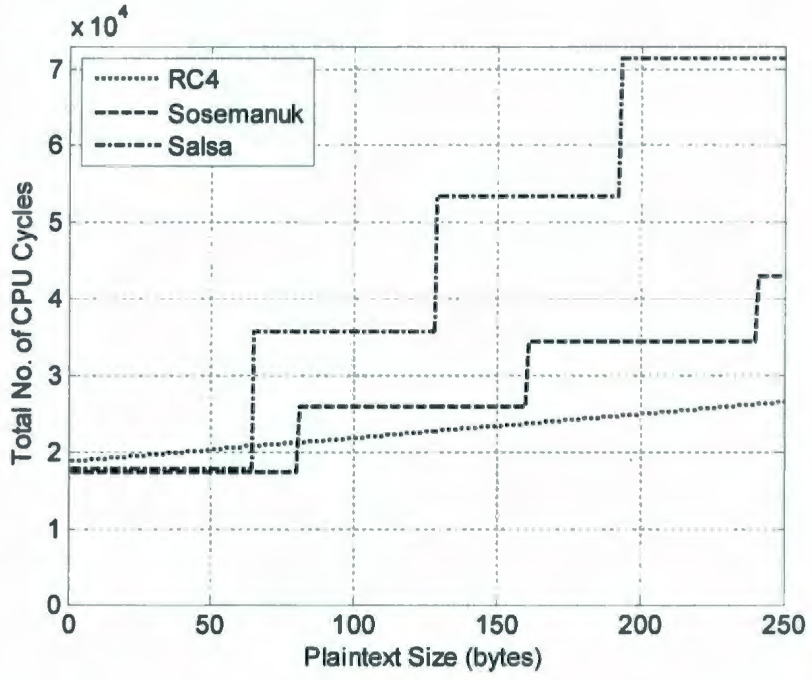


Fig. 4.6. Comparison of computational costs for stream ciphers.

4.4 Evaluation Considering Extrinsic Factors

Now we evaluate the energy performance of the block ciphers and stream ciphers in a noisy environment. We apply the counter mode of operation to the block cipher to ensure that it functions similarly to a stream cipher. This results in a fair comparison between stream ciphers and block ciphers and it is a suitable mode for WSN applications [62]. By using stream ciphers and block ciphers, we analyze the amount of valid data transmitted per Joule, i.e., the energy efficiency. The analysis is based on the Mica sensor node, which is briefly introduced in Chapter 3. The *Periodic IV without ACK* scheme (whose details will be discussed in Chapter 6) is used for the IV distribution, which functions by transmitting IV within an independent packet periodically to resynchronize the cipher. In the analysis, bit errors are generated randomly and independently. Fig. 4.7 shows the performance comparison result of different ciphers, which is obtained under the bit error rate (BER) of 10^{-4} and with the number of data packets between each IV packet being $K = 5$. Other details of the analysis can be found in the following sections, such as the parameters listed in Table 6.3 and packet format illustrated in Fig. 6.3. For generating the MAC used in each data packet, we assume that for each block cipher case, the block cipher is used in CBC mode [61] to generate the MAC [61]. For the stream cipher cases, the MAC is generated by AES used in CBC mode.

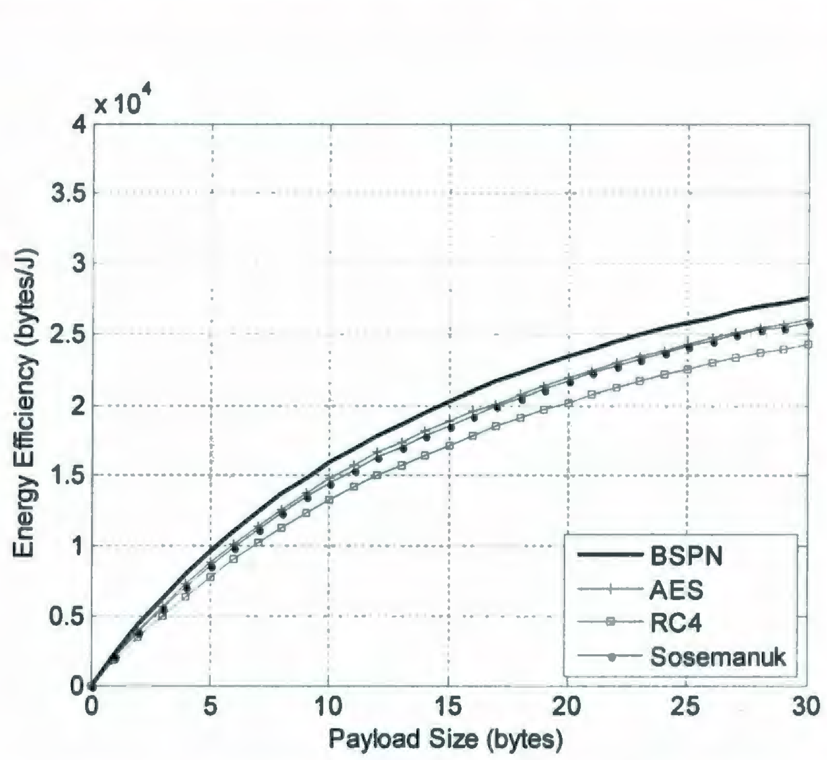


Fig. 4.7. Energy performance of different ciphers (*K*=5, BER = 10^{-4}).

Analysis results show that the BSPN cipher achieves the best energy performance, and in general, a sensor node can transmit more valid data bits using block ciphers. It is worth noting that, noisy channels result in the need for frequent resynchronization. Although for RC4, the number of cycles per byte for encryption is the fewest, it shows the worst energy performance under a noisy channel due to its large number of keystream setup cycles.

4.5 Summary

In this chapter, we have investigated the energy performance of symmetric key cryptographic algorithms when security is applied to the link layer of wireless sensor networks. We evaluate the energy efficiency by comparing the number of CPU cycles per byte for different symmetric key ciphers, including both stream ciphers and block ciphers. We further analyze the ciphers according to their characteristics and the effect of the channel quality when applied in WSNs. Finally, we conclude from the analysis results that the lightweight block cipher, BSPN, achieves good performance, providing energy efficiency as well as suitable security for sensor nodes in a WSN.

Chapter 5

HARDWARE-ORIENTED ENERGY COST OF CRYPTOGRAPHIC ALGORITHMS

This chapter presents the investigation of the energy cost for cryptographic algorithms with hardware implementation in WSNs. There is a recent trend to provide a reconfigurable hardware environment compared to the traditional architecture of wireless sensor nodes. This kind of sensor network is designed to cater to special requirements, such as speed acceleration, on-line debugging and remote reconfiguration. A reconfigurable sensor node typically includes reconfigurable hardware, such as a field-programmable gate array (FPGA), to provide more flexible functionality and much higher performance. The cryptographic algorithm, as a time consuming operation in WSNs, can be processed by the configurable hardware.

In this chapter, we especially investigate the hardware implementation of involutional block ciphers since the characteristics of involution enables performing encryption and decryption using the same circuit. This characteristic is particularly appropriate for a wireless sensor node, which requires the function of both encryption and decryption. We choose two involutional block ciphers: KHAZAD [98] and BSPN [89]; and focus on analyzing their energy efficiency by implementing using different methods. Since both these two ciphers are designed with an SPN structure, we further explore the ciphers' energy cost of different components, such as key scheduling, data path operation and linear transformation. We also implement these two ciphers by assembly language to obtain the energy cost of using software and we evaluate the energy performance for software and hardware cooperation using software for round key scheduling and using hardware for data encryption and decryption.

5.1 Reconfigurable WSNs

A reconfigurable WSN is a kind of sensor network, which incorporates a reconfigurable hardware into a sensor node. The structure of traditional sensor node and reconfigurable sensor node is shown in Fig. 5.1. Here, a traditional sensor node is referred to as the type of sensor node with a low cost general-purpose microcontroller (such as 8-bit Atmel AVR series) to control everything in the sensor node. As we know, a microcontroller functions by executing instructions serially and the efficiency of operation will be much affected when there is a lot of work to deal with at the same time, such as controlling the communication traffic, managing different sensors, processing different kinds of data, maintaining the system performance, etc.. Compared to a traditional sensor node, a reconfigurable sensor node works more efficiently with the help of a reconfigurable hardware. The detailed division of tasks for software and hardware depends on the specific requirements of the application environment. Usually, the hardware is in charge of the complicated and time-consuming data processing.

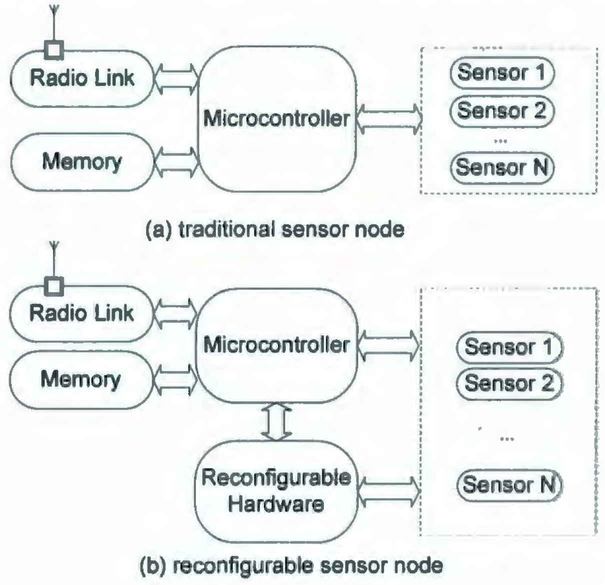


Fig. 5.1. Traditional sensor node vs. reconfigurable sensor node.

In recent years, several researchers have focused on implementing and analyzing a reconfigurable WSN. Some research uses a commercial FPGA functioning as the reconfigurable hardware, such as Altera series used in [99] [100], Xilinx series used in [101] [102], and Actel series used in [103]. Some research uses specific reconfigurable integrated circuits to achieve the hardware acceleration [104][105]. A typical commercial sensor node with reconfigurability is Cookie, which is designed by a modular architecture and use a FPGA of Xilinx Spartan 3 [106].

However, it is an inevitable issue that a FPGA in WSNs might introduce more energy cost while providing the higher performance and more flexibility. In addition, as we know, an FPGA is not a device especially designed for low power consumption. Fortunately, the development of microelectronic technology is trying to mitigate the disadvantage by designing ultra low power FPGA to fill the shortage, such as FPGA from Actel IGLOO series [107], which consumes power as low as 2μw in the best case.

5.2 Utilizing Involutional Block Cipher in WSNs

In this section, we focus on investigating two involutional block ciphers: KHAZAD and BSPN. We will discuss their commonality and detailed difference from the point of view of structure.

We call a function *F*(*x*) involutional when it is its own inverse, which means the following expression is satisfied:

$$x = F(F(x)). \tag{5.1}$$

An involution block cipher means a block cipher with an involutional characteristic as shown in Fig. 5.2. A block cipher encrypts the plaintext *P* to generate the ciphertext *C*, and the exactly same algorithm can be used to generate the plaintext back. The characteristics of involution may be specifically appropriate for a specific type of sensor nodes, which requires the ability of both encryption and decryption. For hardware implementations, the same circuit can provide both encryption and decryption. As the sensor node is a resource-limited device, involutional block ciphers can nicely satisfy this purpose.

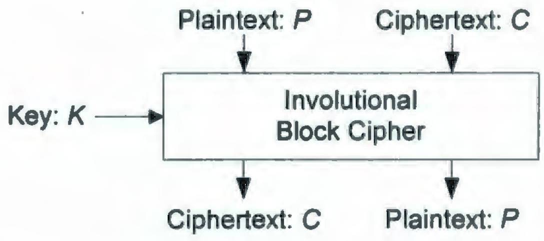


Fig. 5.2. Involutional block cipher.

5.2.1 Common Features of an Involutional SPN cipher

1) SPN Cipher Structure

Both KHAZAD and BSPN adopt an SPN structure which is illustrated in Fig. 5.3, including three basic components: substitution, permutation (linear transformation), and add round keys. They both have eight rounds of operation, an 8×8 S-box and a 64-bit block size. Notice that, the decryption step is in the reverse order of the encryption step. In the encryption process, the "Add Round Key" component is before the component "Linear Transformation"; while in the decryption process, the "Linear Transformation" component goes first. In addition, the round keys used in encryption process should be applied in a reverse order.

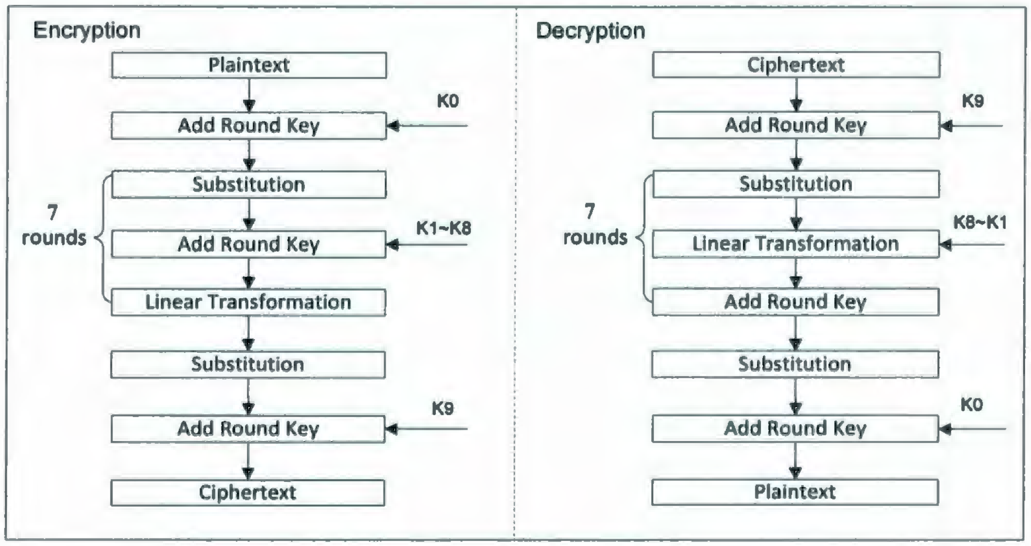


Fig. 5.3. Structure of SPN.

2) Involutional SPN Cipher Structure

Fig. 5.4 shows how involutional SPN cipher works:

- All these three components need to be involutional, which means an involutional S-box and an involutional linear transformation. (The "Add Round Key" operation is originally involutional due to the XOR operation.)
- The operation sequence of the component should be the same for both encryption and decryption. For this purpose, the round keys used in the encryption process need to be first operated through the transformation component, then applied in a reverse order.

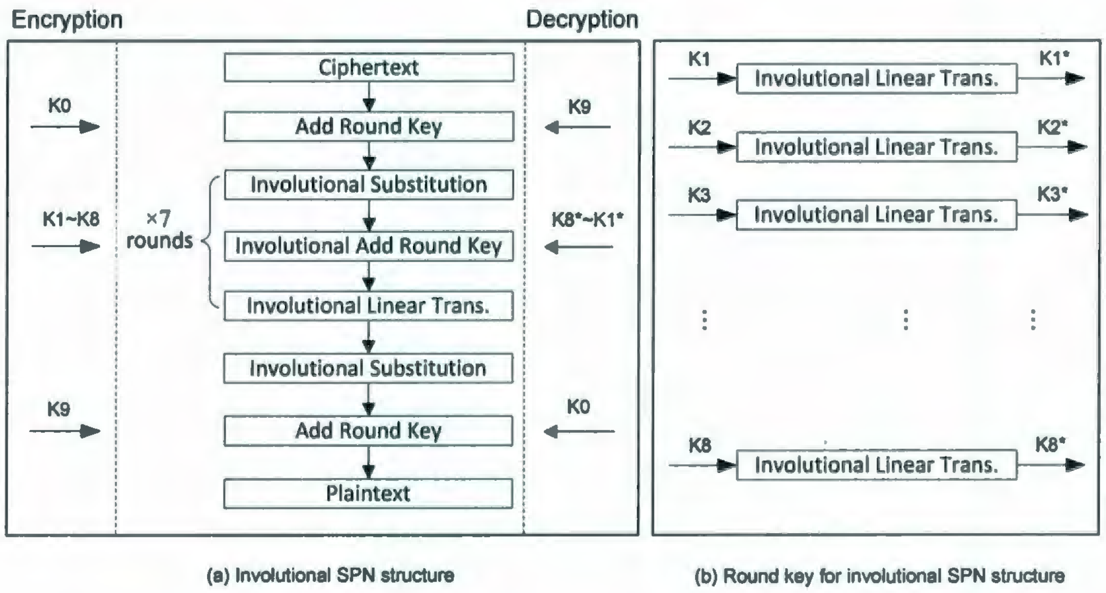

Encryption
Decryption
Ciphertext
Add Round Key
Involutional Substitution
Involutional Add Round Key
Involutional Linear Trans.
Involutional Substitution
Add Round Key
Plaintext
K0
K1~K8
×7 rounds
K9
K9
K8*~K1*
K0
K1
K1*
K2
K2*
K3
K3*
K8
K8*
Involutional Linear Trans.
(a) Involutional SPN structure
(b) Round key for involutional SPN structure

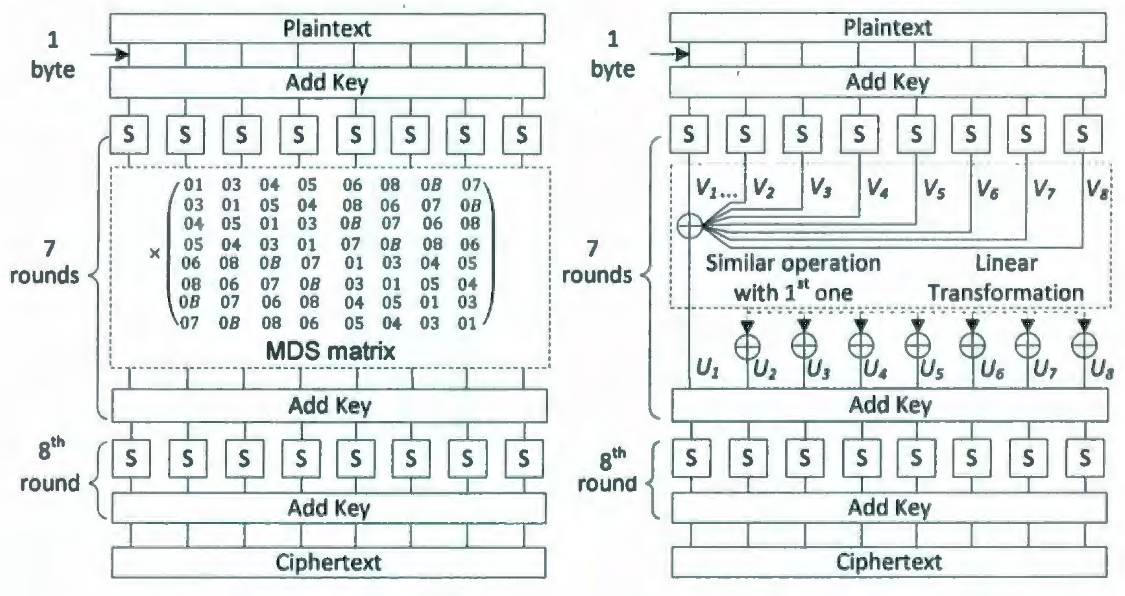
Fig. 5.4. Structure of involutional SPN.

3) Round Key Expansion

Both KHAZAD and BSPN have eight rounds of operation, with no linear transformation involved in the last round. The nine round keys involved in the "Add Round Key" operation are generated based on a cipher key and a process called round key expansion or key scheduling. Both the round key expansion process of cipher KHAZAD and BSPN are based on the concept of SPN.

5.2.2 Ciphers: KHAZAD and BSPN

In this section, the details of cipher KHAZAD and BSPN will be discussed since the structural complexity of the cipher will affect the energy cost in a sensor node. The detailed structures of the two ciphers are shown in Fig. 5.5, including the substitution, linear transformation and add round keys.



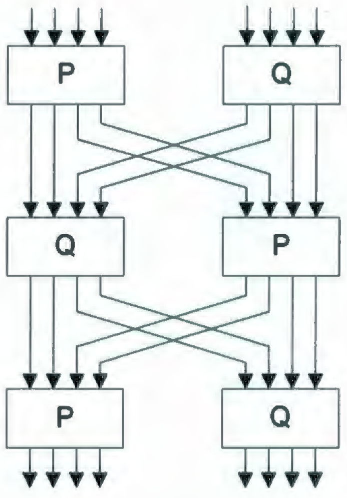
(a) Structure of KHAZAD (b) Structure of BSPN

Fig. 5.5. Structures of KHAZAD and BSPN.

1) Substitution: KHAZAD vs. BSPN

- KHAZAD

The involutional S-box of KHAZAD was initially designed by pseudo-random S-box generation. An appropriate S-box is chosen among these random S-boxes, which satisfies involutional functionality and other security requirements. However, it is pointed out that the pseudo-randomly generated S-box may be not efficient for hardware implementation under some conditions. For this purpose, the designers further proposed an alternative S-box, which is based on the structure of involutional permutation. The structure of the KHAZAD S-box is shown in Fig. 5.6, where *P* and *Q* are pseudo-randomly generated involutional mini-boxes, and the mini-box relation between input and output is shown. The final involutional S-box is shown Appendix A.1.



Mini-box

u	P[u]	Q[u]
0	3	9
1	F	E
2	E	5
3	0	6
4	5	A
5	4	2
6	B	3
7	C	C
8	D	F
9	A	0
A	9	4
B	6	D
C	7	7
D	8	B
E	2	1
F	1	8

Fig. 5.6. Structure of KHAZAD S-box.

- BSPN

There is no specific S-box provided in the cipher BSPN specification [89]. Hence, we have designed an involutional S-box based on the idea of AES's encryption S-box. The structure of the BSPN S-box is shown in Fig. 5.7. We use AES's multiplicative inverse part, which is in Galois field $GF(2^8)$ based on the irreducible polynomial

$$\mathrm{m}(x) = x^8 + x^4 + x^3 + x + 1 \tag{5.2}$$

due to its good cryptographic properties [10]. To make the S-box involutional, we use the affine transformation twice, which XORs the other 7 bits of the input but not the operated bit itself. We implement the S-box structure in C language, and obtain the substitution lookup table as shown in Appendix A.2. Although there are two fixed points (which is 0x00 and 0xFE), there is no proof that the fixed points can lead to an attack on the cipher.

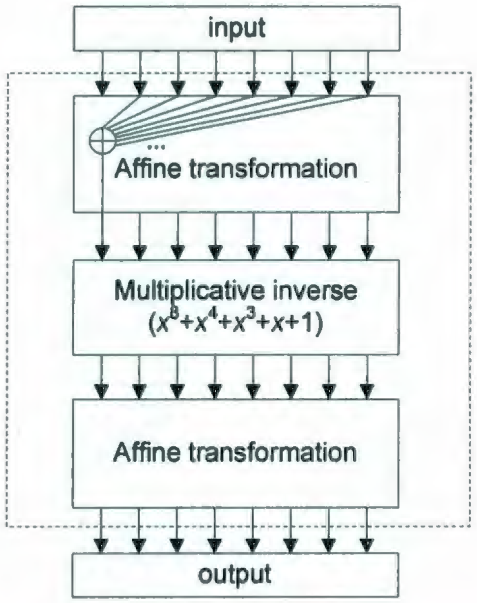


Fig. 5.7. The BSPN involutional S-box generation.

2) Linear Transformation: KHAZAD vs. BSPN

- Cipher KHAZAD

The linear transformation layer is based on an involutional MDS code, which is selected by exhaustive search. To achieve efficient implementation for both software and hardware, the final decision is made by two factors: the lowest possible hamming weight and the lowest possible integer values. The linear transformation is expressed as

$$(b_0 \quad b_1 \quad b_2 \quad b_3 \quad b_4 \quad b_5 \quad b_6 \quad b_7) =$$
$$(a_0 \quad a_1 \quad a_2 \quad a_3 \quad a_4 \quad a_5 \quad a_6 \quad a_7) \times \begin{pmatrix} 01 & 03 & 04 & 05 & 06 & 08 & 0B & 07 \\ 03 & 01 & 05 & 04 & 08 & 06 & 07 & 0B \\ 04 & 05 & 01 & 03 & 0B & 07 & 06 & 08 \\ 05 & 04 & 03 & 01 & 07 & 0B & 08 & 06 \\ 06 & 08 & 0B & 07 & 01 & 03 & 04 & 05 \\ 08 & 06 & 07 & 0B & 03 & 01 & 05 & 04 \\ 0B & 07 & 06 & 08 & 04 & 05 & 01 & 03 \\ 07 & 0B & 08 & 06 & 05 & 04 & 03 & 01 \end{pmatrix}, \tag{5.3}$$

where $a_{0...7}$ and $b_{0...7}$ represent input and output byte respectively, and the reduction polynomial in Galois field GF(2^8) is

$$\mathrm{m}(x) = x^8 + x^4 + x^3 + x^2 + 1. \tag{5.4}$$

- Cipher BSPN

In cipher BSPN, the result of the linear transformation, U, is achieved by bitwise XORing the output bytes, V_i, of the other seven S-boxes after adding the round key, i.e., for byte i,

$$U_i = \oplus_{j=1, j\neq i}^{8} V_j, \ 1 \leq i \leq \ 8. \tag{5.5}$$

Compared to cipher KHAZAD, cipher BSPN has a much simpler linear transformation function. However, it achieves better energy efficiency and is more suitable for the application of energy-limited environment, which will be shown in the next section.

3) Round Key Generation: KHAZAD vs. BSPN

- **Cipher KHAZAD**

The round key expansion of cipher KHAZAD is shown in Fig. 5.8. During the key expansion period, the 128-bit cipher key is expanded into nine 64-bit round keys, K^r, $0 \leq r \leq 8$ where r represents the round number. For the round 0, K^{-1} and K^{-2} is initialized by the cipher key, i.e., K^{-2} and K^{-1} take the first 8 bytes and the last 8 bytes of the cipher key, respectively.

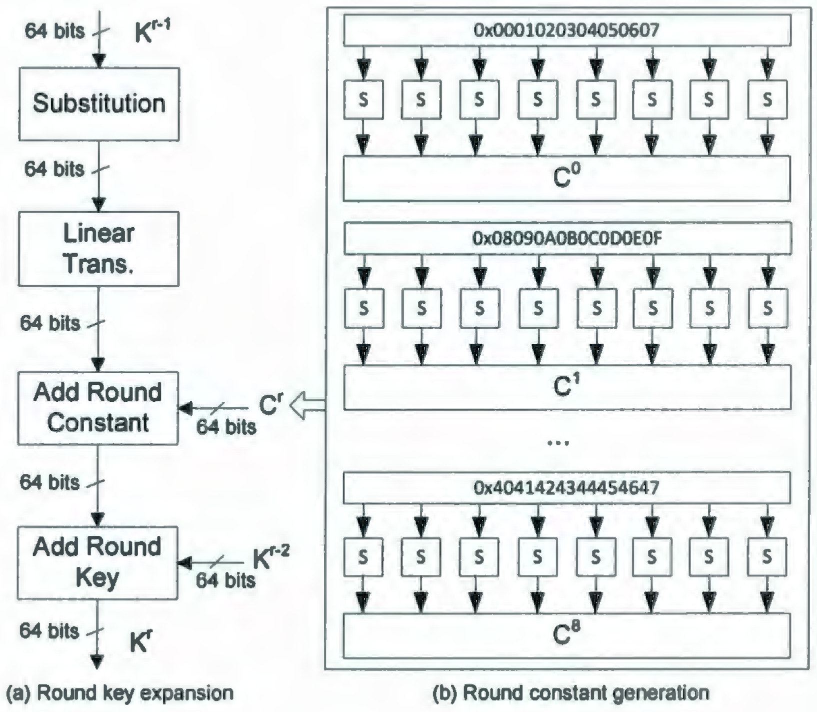


Fig. 5.8. KHAZAD key expansion.

- **Cipher BSPN:**

The round keys of BSPN are generated directly by using BSPN cipher itself to encrypt any arbitrary value except 0. That is to say, each round key is generated by eight rounds of operation. The round keys used for key expansion depend on the key size of the cipher

BSPN. As shown in Fig. 5.9, if the cipher key size is 64 bits, the cipher key will used directly in each round key; while if the cipher key size is 128 bits, the first 8 bytes and the last 8 bytes of the cipher key will be used alternatively. Compared to the key expansion of cipher KHAZAD, BSPN has a more complicated key expansion algorithm. However, in a WSN, the cipher key will not be changed very frequently, which means the energy cost of the key expansion process will not significantly affect the total energy performance.

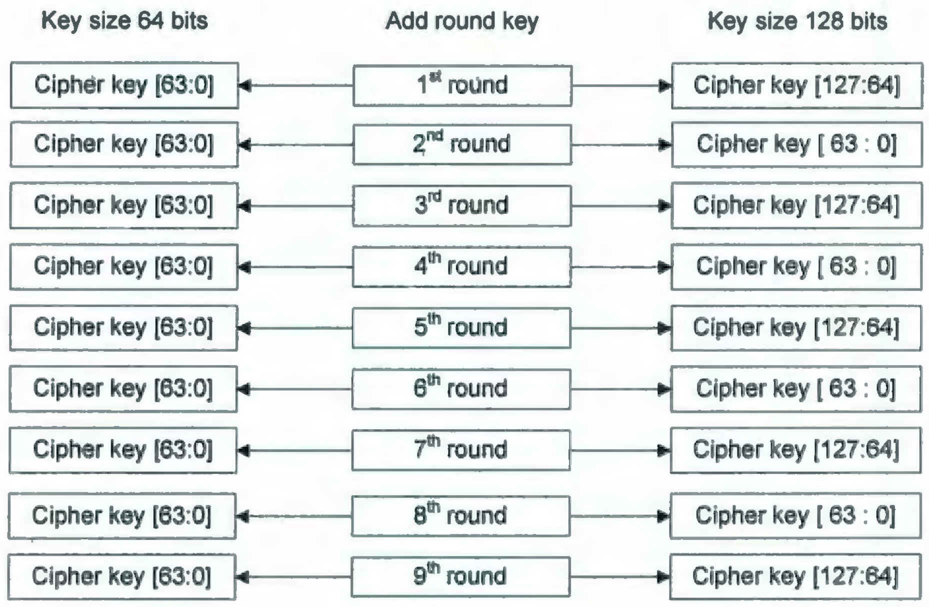


Fig. 5.9. BSPN round key used for key size of 128 bits and 64 bits.

5.3 Hardware Implementation

5.3.1 Design Objective

In this section, we focus on the hardware implementation of cipher KHAZAD and BSPN considering the application environment of WSNs. The hardware is implemented by Verilog HDL code with FPGA as the technology target. Since in WSNs, a sensor node is energy limited device, low power is the major consideration of the design. In addition, as a life limited device, key establishment will not be done very frequently. Unlike many cipher hardware implementations, which generate the round keys on the fly to save the area necessary to store round keys, we divide the design into two relatively independent parts: (1) the process of key scheduling and (2) the process of data en/decryption. In our design, the key expansion is only processed during the initial setup period or when a new cipher key is updated. The generated round keys will be stored for future use in the resources available on the FPGA. As we mentioned, in a sensor node, most activity of the circuit is encrypting or decrypting data, which will be much more power efficient since no key scheduling activity is involved in the circuit.

5.3.2 Interfaces and Requirements

In our design, both cipher KHAZAD and BSPN use the same interfaces and timing requirements such that it is convenient to understand and compare these two designs. The block diagram is shown in Fig. 5.10 with the description of input and output interfaces illustrated in Table 5.1. The timing requirements of the interfaces are shown in Fig. 5.11.

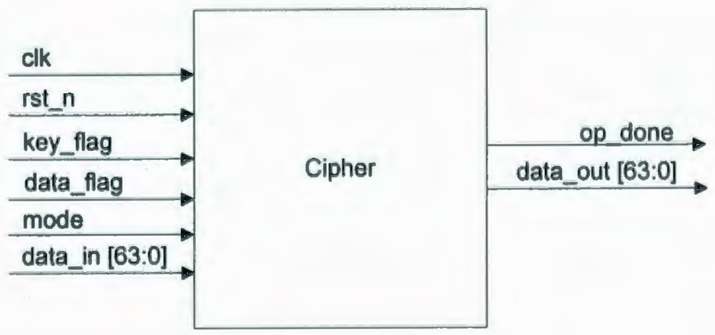


Fig. 5.10. Interface of the cipher.

Table 5.1. Interface description.

Port	Name	Bits	Description
input	clk	1	Clock of the design, rising edge sensitive
	rst_n	1	Reset signal of the design, which is asynchronous low level sensitive.
	key_flag	1	Rising edge sensitive signal indicating the key expansion process begins and there comes a new cipher key.
	data_flag		Rising edge sensitive signal indicating data operation process begins and there comes a 64-bit size data to be encrypted or decrypted.
	mode		When mode = 0, it indicates an encryption process; When mode = 1, it indicates a decryption process. It should be ready before the data_flag's rising edge.
	data_in	64	(1) Used as data: When mode=0, data_in is the plaintext; when mode=1, data_in is the ciphertext. It should be ready before the data_flag's rising edge. (2) Used as key: If the key size is 64 bits, it should be ready before key_flag's rising edge. If the key size is 128 bits, the first 8 bytes of the key should be ready before the rising edge of key_flag, and the last eight bytes of the key should be ready within 2 cycles after the rising edge of key_flag.
output	op_done	1	Level sensitive signal indicating the operation is finished. It means that the key expansion finished corresponding to the rising edge of key_flag; and means en/decryption finished corresponding to the rising edge of data_flag.
	data_out	64	When mode=0, data_out is the plaintext; when mode=1, data_out is the ciphertext. The data is valid after the rising edge of op_done.

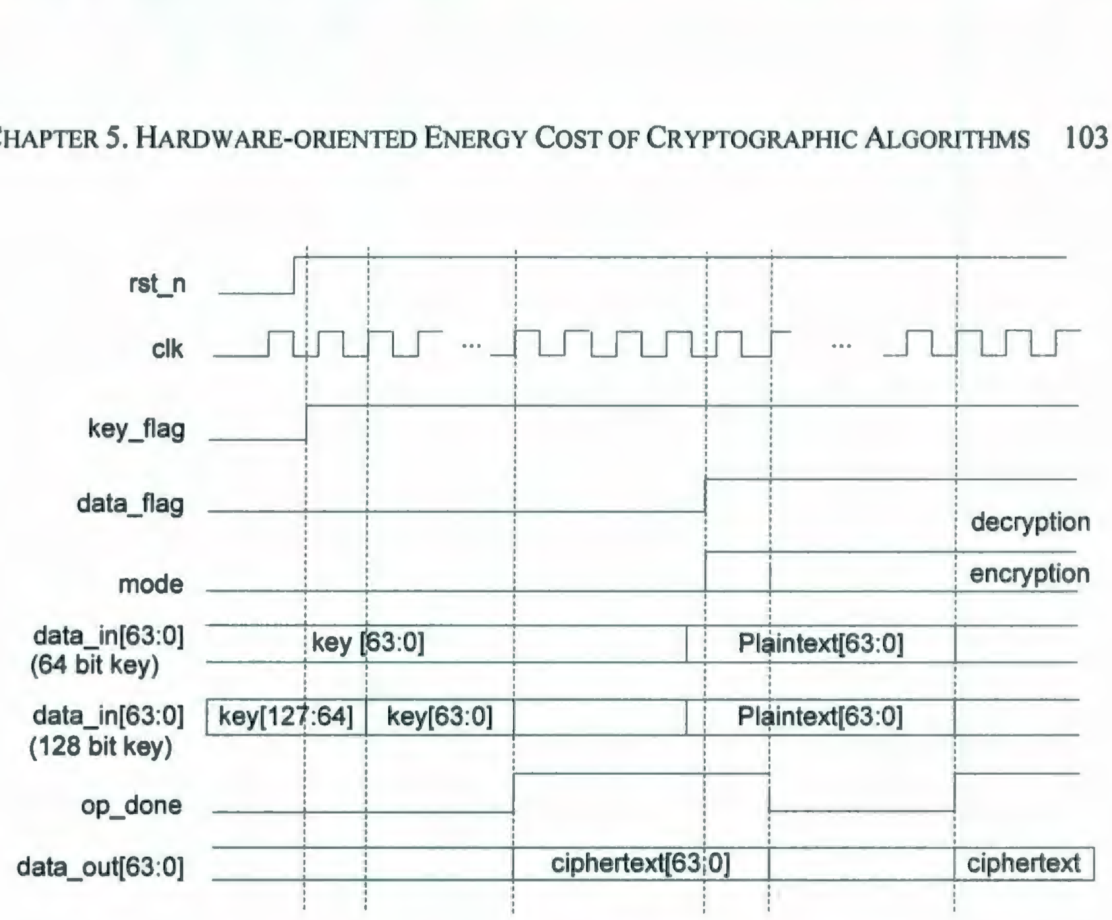


Fig. 5.11. Timing requirements of the interface signals.

5.3.3 Implementation Details

1) Substitution

The implementation of an 8×8 S-box for an FPGA target can be achieved in two ways: (1) using a lookup table (LUT) to implement the Boolean function directly which is constructed by combinational logic and (2) using the configurable block RAM core which is embedded in FPGA device. The first method can achieve low time delay, while the second method can fully utilize the resources provided by the device. Furthermore, since both KHAZAD and BSPN have a 64-bit block size, the whole substitution layer can also be designed for different purposes: (1) using only one S-box circuit for minimizing area or (2) using eight S-box circuits in parallel for increasing speed. In our design, to find a good solution for energy efficiency, we have tried different combinations.

- **LUT vs. BRAM**

Using LUTs (combinational logic) to implement the S-box is a quite straightforward way. The FPGA synthesis tool will generate the 8-bit output logic circuit according to the 8-bit input. However, the detailed result will be different depending on the mapping of the S-box.

BRAM needs to be configured and generated before use, and the memory size should not override the maximum size provided by the FPGA device. According to the needs of the design, we generate a single port synchronous read only memory with size 256 and depth 8.

The interface of the S-box circuit is shown in Fig. 5.12. It is worth noting that for the LUT design, the output will be valid with a short delay after the input is given, while for the BRAM design, the output will not be valid until the next clock rising edge because of the synchronous structure. Furthermore, the timing delay of the LUT design will be less than that of the BRAM design, since many factors affect the timing delay of BRAM such as the floor plan of the chip and specific technology being used.

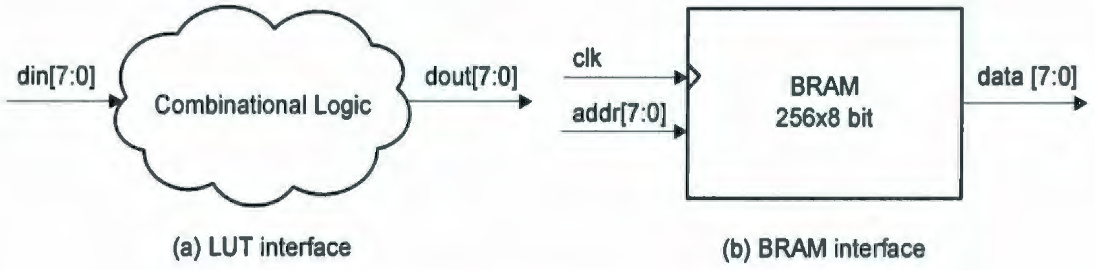


Fig. 5.12. Interfaces of S-box.

- **Small Area vs. High Speed**

For the purpose of small area, only one S-box circuit is used in the design, which serially generates the output value byte by byte. The design needs eight clock cycles all together to finish one round of 64-bit data substitution. We use a 3-bit counter and a demux to in which register to store the S-box output value. The circuit structure is shown in Fig. 5.13.

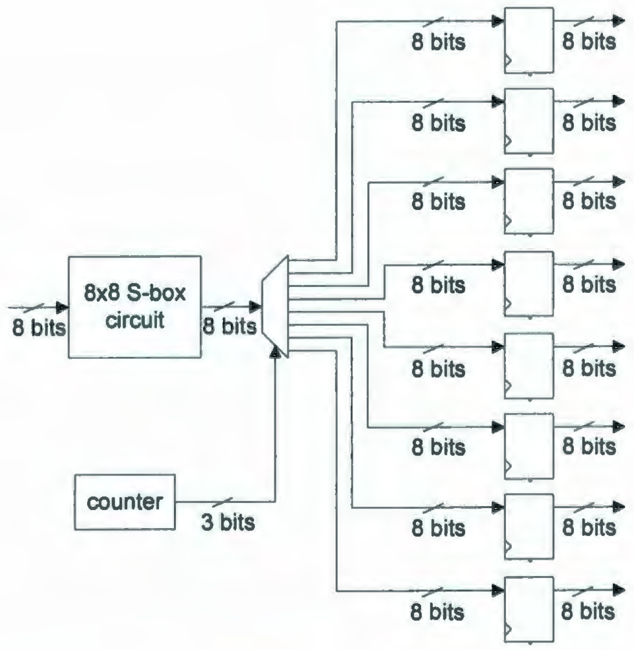


Fig. 5.13. Substitution implementation using one S-box unit.

For the purpose of high speed, eight S-box circuits are used in the design, which simultaneously generates the full 64-bit output in one clock cycle. The circuit structure is shown in Fig. 5.14.

Notice that both the speed and area will affect the energy efficiency in a sensor node. Processing the data for a long time or using a large number of transistors will both lead to large energy cost. The low power design is a balance between these two factors.

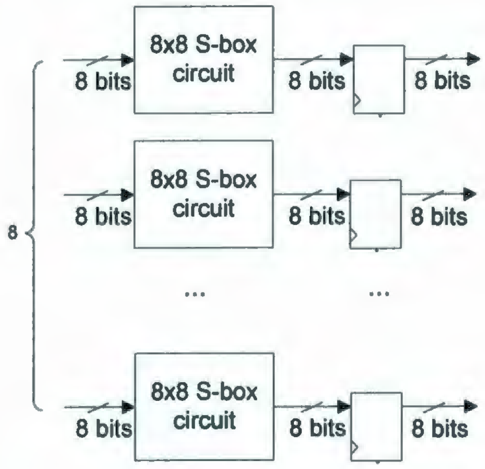


Fig. 5.14 Substitution implementation using eight S-box units

2) Linear Transformation

- **KHAZAD**

We implement the KHAZAD linear transformation by using combinational logic. To make the circuit work more efficiently, we need to do some transformation first to avoid using a complicated multiplier circuit for the matrix. The steps of the transformation are described as following:

<u>**Step 1**</u>: Expand the result of linear transformation by definition:

$$(b_0 \quad b_1 \quad b_2 \quad b_3 \quad b_4 \quad b_5 \quad b_6 \quad b_7) =$$

$$(a_0 \quad a_1 \quad a_2 \quad a_3 \quad a_4 \quad a_5 \quad a_6 \quad a_7) \times \begin{pmatrix} 01 & 03 & 04 & 05 & 06 & 08 & 0B & 07 \\ 03 & 01 & 05 & 04 & 08 & 06 & 07 & 0B \\ 04 & 05 & 01 & 03 & 0B & 07 & 06 & 08 \\ 05 & 04 & 03 & 01 & 07 & 0B & 08 & 06 \\ 06 & 08 & 0B & 07 & 01 & 03 & 04 & 05 \\ 08 & 06 & 07 & 0B & 03 & 01 & 05 & 04 \\ 0B & 07 & 06 & 08 & 04 & 05 & 01 & 03 \\ 07 & 0B & 08 & 06 & 05 & 04 & 03 & 01 \end{pmatrix},$$

Then we get the output results to be expressed as

$b_0 = a_0 \oplus 3a_1 \oplus 4a_2 \oplus 5a_3 \oplus 6a_4 \oplus 8a_5 \oplus Ba_6 \oplus 7a_7$

$b_1 = 3a_0 \oplus a_1 \oplus 5a_2 \oplus 4a_3 \oplus 8a_4 \oplus 6a_5 \oplus 7a_6 \oplus Ba_7$

$b_2 = 4a_0 \oplus 5a_1 \oplus a_2 \oplus 3a_3 \oplus Ba_4 \oplus 7a_5 \oplus 6a_6 \oplus 8a_7$

$b_3 = 5a_0 \oplus 4a_1 \oplus 3a_2 \oplus a_3 \oplus 7a_4 \oplus Ba_5 \oplus 8a_6 \oplus 6a_7$

$b_4 = 6a_0 \oplus 8a_1 \oplus Ba_2 \oplus 7a_3 \oplus a_4 \oplus 3a_5 \oplus 4a_6 \oplus 5a_7$

$b_5 = 8a_0 \oplus 6a_1 \oplus 7a_2 \oplus Ba_3 \oplus 3a_4 \oplus a_5 \oplus 5a_6 \oplus 4a_7$

$b_6 = Ba_0 \oplus 7a_1 \oplus 6a_2 \oplus 8a_3 \oplus 4a_4 \oplus 5a_5 \oplus a_6 \oplus 3a_7$

$b_7 = 7a_0 \oplus Ba_1 \oplus 8a_2 \oplus 6a_3 \oplus 5a_4 \oplus 4a_5 \oplus 3a_6 \oplus a_7$

Step 2: Using the following polynomial into the expressions:

$1 = (x^0)$, $3 = (x+1)$, $4 = (x^2+1)$, $5 = (x^2+1)$,
$6 = (x^2+x)$, $7 = (x^2+x+1)$, $8 = (x^3)$, $B = (x^3+x+1)$,

reorganize the expression and we get the final result

$b_0 = (a_0 \oplus a_1 \oplus a_3 \oplus a_6 \oplus a_7) \oplus (a_1 \oplus a_4 \oplus a_6 \oplus a_7)x \oplus (a_2 \oplus a_3 \oplus a_4 \oplus a_6)x^2 \oplus (a_5 \oplus a_6)x^3$

$b_1 = (a_0 \oplus a_1 \oplus a_2 \oplus a_6 \oplus a_7) \oplus (a_0 \oplus a_5 \oplus a_6 \oplus a_7)x \oplus (a_2 \oplus a_3 \oplus a_5 \oplus a_6)x^2 \oplus (a_4 \oplus a_7)x^3$

$b_2 = (a_1 \oplus a_2 \oplus a_3 \oplus a_4 \oplus a_5) \oplus (a_3 \oplus a_4 \oplus a_5 \oplus a_6)x \oplus (a_0 \oplus a_1 \oplus a_5 \oplus a_6)x^2 \oplus (a_4 \oplus a_7)x^3$

$b_3 = (a_0 \oplus a_2 \oplus a_3 \oplus a_4 \oplus a_5) \oplus (a_2 \oplus a_4 \oplus a_5 \oplus a_7)x \oplus (a_0 \oplus a_1 \oplus a_4 \oplus a_7)x^2 \oplus (a_5 \oplus a_6)x^3$

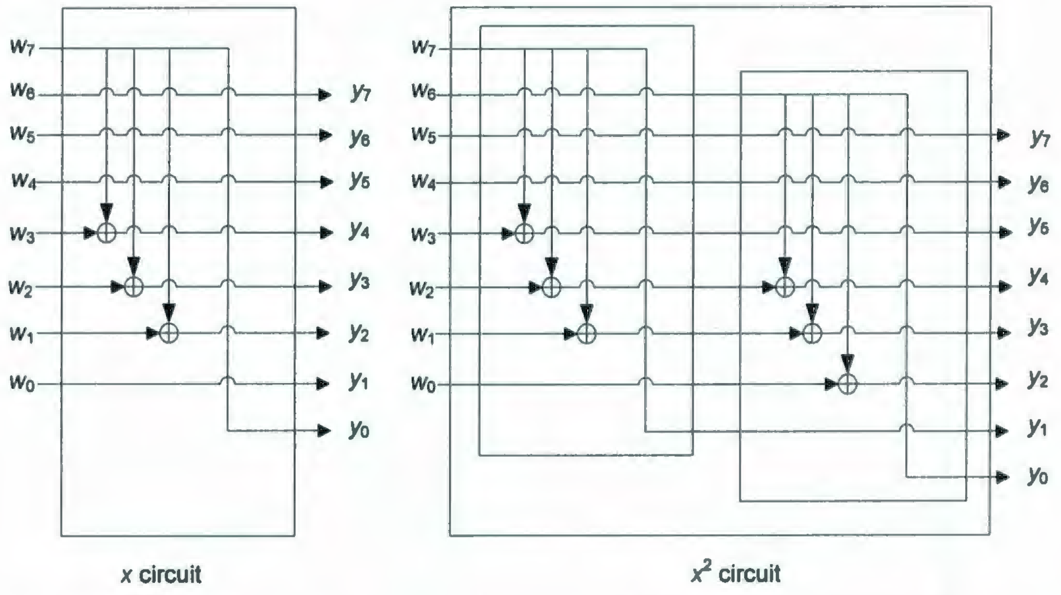
$b_4 = (a_2 \oplus a_3 \oplus a_4 \oplus a_5 \oplus a_7) \oplus (a_0 \oplus a_2 \oplus a_3 \oplus a_5)x \oplus (a_0 \oplus a_3 \oplus a_6 \oplus a_7)x^2 \oplus (a_1 \oplus a_2)x^3$

$b_5 = (a_2 \oplus a_3 \oplus a_4 \oplus a_5 \oplus a_6) \oplus (a_1 \oplus a_2 \oplus a_3 \oplus a_4)x \oplus (a_1 \oplus a_2 \oplus a_6 \oplus a_7)x^2 \oplus (a_0 \oplus a_3)x^3$

$b_6 = (a_0 \oplus a_1 \oplus a_5 \oplus a_6 \oplus a_7) \oplus (a_0 \oplus a_1 \oplus a_2 \oplus a_7)x \oplus (a_1 \oplus a_2 \oplus a_4 \oplus a_5)x^2 \oplus (a_0 \oplus a_3)x^3$

$b_7 = (a_0 \oplus a_1 \oplus a_4 \oplus a_6 \oplus a_7) \oplus (a_0 \oplus a_1 \oplus a_3 \oplus a_6)x \oplus (a_0 \oplus a_3 \oplus a_4 \oplus a_5)x^2 \oplus (a_1 \oplus a_2)x^3$

Step 3: We can see that four major components are involved in the design: x^0 (can be viewed as 1 and is not shown in the expression of b_i), x, x^2, and x^3. The irreducible polynomial is $m(x) = x^8+x^4+x^3+x^2+1$, hence, the x, x^2, x^3 circuits can be constructed by the combinational logic which are shown in Fig. 5.15. In this figure, each variable w_i and y_i is a bit. The input for w_i is the left expression of x^i (i=1,2,3) in step 2, such as for expression $(a_0 \oplus a_1 \oplus a_3 \oplus a_6)x$, the XOR result of $(a_0 \oplus a_1 \oplus a_3 \oplus a_6)$ is connected to w_i.



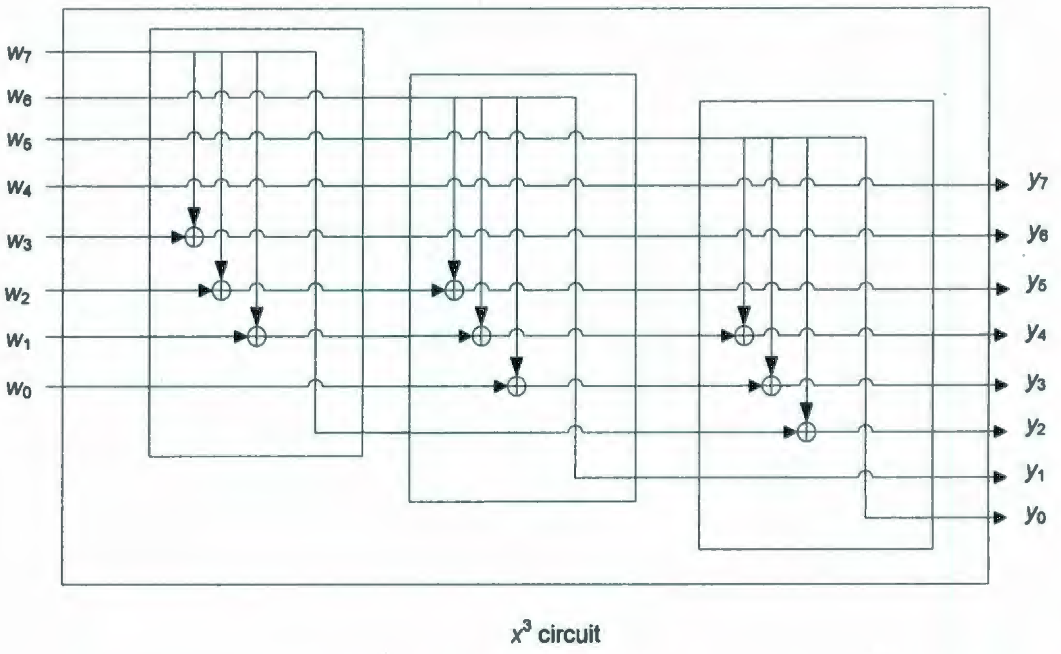


Fig. 5.15 x, x^2 and x^3 circuit of KHAZAD

Step 4: XOR all the results, y_i, of circuit 1, x, x^2, and x^3 together.

- **BSPN**

Compared to cipher KHAZAD, the implementation of BSPN is much simpler and straightforward. It can be implemented by the combinational logic following its linear transformation definition.

3) SPN Core Circuit Sharing

Three basic components "substitution", "linear transformation" and "add round key" constitute the core circuit of the SPN. The implementation details of "substitution" and "linear transformation" are discussed in the previous two sections, and "add round key" can be directly implemented by XOR logic. Now we consider using the SPN core to construct the cipher.

As we discussed before, to achieve better power efficiency, the key expansion process and data en/decryption processes are regarded as two individual processes. To save the circuit area further, we make use of the fact that these two processes will not happen at the same time. Hence, the SPN core is designed for use by both the key expansion and the data en/decryption. The block diagram is illustrated in Fig. 5.16. In this figure, we can see that the design is based on the reuse of the SPN core. A process indicator is generated from the input key_flag and data_flag to indicate which process is currently being operated. A counter is used to calculate the operation rounds, and its maximum value is decided by the specific cipher and process. Multiplexers are used to apply data to the corresponding process.

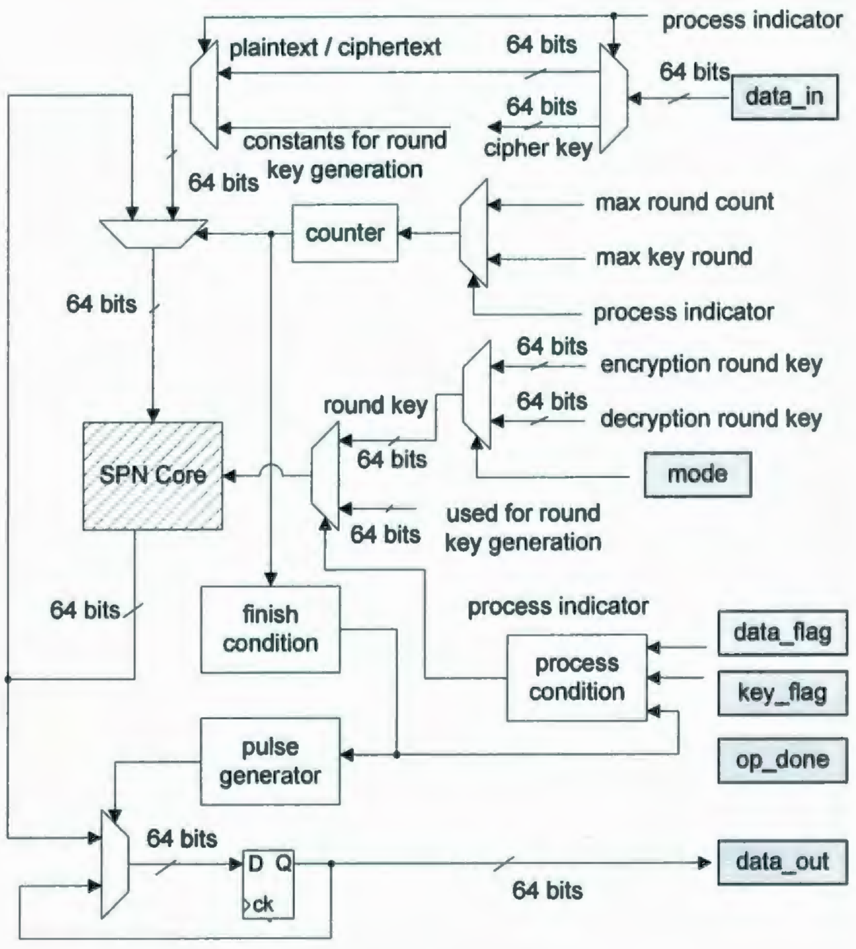


Fig. 5.16. SPN core.

5.4 Implementation Results

We choose Xilinx Spartan 3 series, xc3s200 with package 208, as the target device of implementation, which is a low end, low power FPGA. The working voltage of the device xc3s200 is 1.2v, and its available resources are listed Table 5.2. We use EDA tool ModelSim for simulation and Xilinx ISE for synthesis.

Table 5.2. Available resource of xc3s200.

Type	Available Resources
Number of slice flip flops	3840
Number of 4 input LUT	3840
Number of bounded IOBs	141
Number of block RAMs	12

Both cipher KHAZAD and BSPN are implemented in four methods, which represent different considerations as we discussed in previous sections. The four methods are:

- Using 1 S-box LUT circuit
- Using 8 S-box LUT circuit
- Using 1 S-box BRAM circuit
- Using 8 S-box BRAM circuit

By implementing the block ciphers with different methods, we focus on exploring the energy efficiency of hardware implementation, which is affected by both the circuit area and the total operation time. We also compare the energy performance between cipher KHAZAD and cipher BSPN at different levels: (1) full hardware including key scheduling and data operation, (2) only data operation, and (3) different structures of linear transformation.

5.4.1 LUT Method vs. BRAM Method

We first investigate the effects of different methods applied to a design, including using LUT and BRAM, as well as the number of S-boxes being involved. Here, we use labels BSPN-64 and BSPN-128 to represent the cipher BSPN with key size 64 bits and

128 bits, respectively. We implement BSPN-64 by four methods, and the implementation results are shown in Table 5.3. From these results, we can see that when using the same number of S-box circuits, the S-box constructed by LUT or BRAM is a dominant factor for the total equivalent gate count for design. This is because the equivalent gate count of BRAM is greatly larger than the same function using LUT. However, it makes use of the FPGA resources more efficiently. Compared to using LUT, the number of occupied slices is much smaller with the use of BRAM, which allows the use of FPGA slices for implementing other functions. Furthermore, designs using eight S-box circuits introduce more FPGA resources than using only one S-box. However, using eight S-boxes to finish processing substitution in one clock cycle will greatly decrease the total time needed for data encryption and decryption.

Table 5.3. Implementation result of 64-bit key BSPN.

Method	Resources	Used resources		Utilization of Resources	
		1 S-box	8 S-box	1 S-box	8 S-box
LUT	# slice flip flop	721	653	18%	17%
	# 4 input LUT	2,012	2,757	52%	71%
	# occupied slice	1,350	1,678	70%	87%
	#BRAM	0	0	0	0
	Total equivalent gate count for design	19,782	26,231	-	-
BRAM	# slice flip flop	722	589	18%	15%
	# 4 input LUT	1,884	1,675	49%	43%
	# occupied slice	1,291	1,141	67%	59%
	#BRAM	1	8	8%	66%
	Total equivalent gate count for design	84,198	540,635	-	-

5.4.2 KHAZAD vs. BSPN

In this part, we focus on comparing the hardware implementation of cipher KHAZAD and BSPN-128 with both of them using a 128-bit key.

1) Full Cipher Implementation

We implement cipher KHAZAD and BSPN-128 with full cipher functionality, including key scheduling, data encryption and data decryption. In the implementation, BRAM is used to construct the S-box, such that the FPGA resources can be efficiently utilized. The implementation results are shown in Table 5.4. In this table, we can see that cipher KHZAD used more resources than BSPN_128 for both the "1 S-box" method and the "8 S-box" method.

Table 5.4. Implementation results full ciphers using BRAM: KHAZAD vs. BSPN.

Method	Resources	Used resources		Utilization of Resources	
		KHAZAD	BSPN	KHAZAD	BSPN
1 BRAM S-box	# slice flip flop	779	780	20%	20%
	# 4 input LUT	2602	2183	67%	56%
	# occupied slice	1686	1443	87%	75%
	#BRAM	1	1	8%	8%
	Total equivalent gate count for design	89,690	87,214	-	-
8 BRAM S-box	# slice flip flop	655	654	17%	17%
	# 4 input LUT	2694	1907	70%	49%
	# occupied slice	1728	1258	90%	65%
	#BRAM	8	8	66%	66%
	Total equivalent gate count for design	547,207	543,056	-	-

We have also implemented these two ciphers using the "8 S-box LUT" method, and the results show the same trend. As shown in the shaded cell of Table 5.5, the resource utilization for cipher KHAZAD achieves 110%, which means our target device xc3s200 cannot even offer enough space.

Table 5.5. Implementation results full ciphers using LUT: KHAZAD vs. BSPN-128.

Method	Resources	Used resources		Utilization of Resources	
		KHAZAD	BSPN	KHAZAD	BSPN
8 LUT S-box	# slice flip flop	721	720	18%	18%
	# 4 input LUT	3764	2952	98%	76%
	# occupied slice	2117	1814	110%	94%
	#BRAM	0	0	0	0
	Total equivalent gate count for design	34,361	28,452	-	-

It is worth noting that these results are generated from the combination of both key scheduling and data operation. Although key scheduling is not working as frequently as the data operation does, the hardware will always requires FPGA resources dedicated to key scheduling regardless of its level of use.

2) Data Operation Implementation

As we know, the key scheduling operation is a relatively independent process in the cipher and is not processed frequently. Hence, we consider another scenario for the cipher that only data encryption and decryption is included in the design. Since our design purpose is to apply the circuit to a reconfigurable WSN, the cooperation of hardware and software can be used: first, the MCU can generate the round keys by software and store them in its memory for future use; then the MCU indicates to the hardware to directly load the round keys and release the memory for other use.

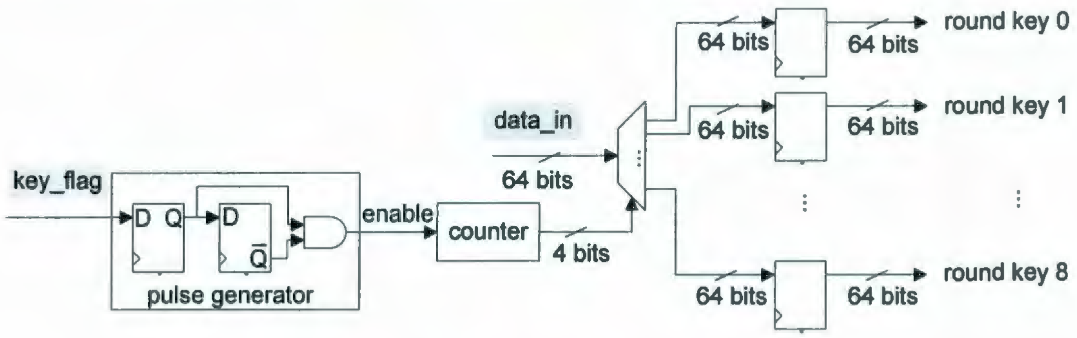


Fig. 5.17. Structure of loading round keys.

The interface of the design is the same as that of the full cipher except that the round keys are loaded in the following nine clock cycles after the rising edge of input key_flag. The circuit structure for the loading round keys is illustrated in Fig. 5.17. Nine 64-bit registers are used to store the encryption round keys for future use.

Table 5.6 Implementation results for only en/decryption: KHAZAD vs. BSPN-128.

Method	Resources	Used resources		Utilization of Resources	
		KHAZAD	BSPN	KHAZAD	BSPN
1 BRAM S-box	# slice flip flop	644	645	16%	16%
	# 4 input LUT	1784	1349	46%	35%
	# occupied slice	1234	1020	64%	53%
	#BRAM	1	1	8%	8%
	Total equivalent gate count for design	83,366	80,890	-	-
8 BRAM S-box	# slice flip flop	520	521	13%	13%
	# 4 input LUT	2,028	1,083	52%	28%
	# occupied slice	1,381	857	71%	44%
	#BRAM	8	8	66%	66%
	Total equivalent gate count for design	541,993	537,051	-	-

We implement these two ciphers using BRAM structures for S-box with two cases: the "1 S-box" method and the "8 S-box" method. The implementation result is shown in Table 5.6, which shows a similar trend as that of the full cipher implementation. The total equivalent gate count for cipher KHAZAD is much larger than that of BSPN. Moreover, the equivalent gate count for designs using eight S-box circuits is much larger than using only the one S-box circuit.

3) Only Linear Transformation

We now further investigate the resource consumption of the component of the ciphers KHAZAD and BSPN. Although the details of the two ciphers are different, they have similar SPN structure with eight rounds of operation. The two ciphers can be implemented with the same interfaces, timing requirements and method. The linear transformation part of these two ciphers is the major difference because the other basic components of an SPN are almost the same: (1) BRAM makes the S-box using the same resources of FPGA and (2) the operation of add round keys uses the same amount of XOR logic. Hence, we focus on analyzing the circuit of linear transformation explicitly.

Table 5.7. Synthesis result of linear transformation.

Macro Statistics	BSPN	KHAZAD
8-bit xor2	8	28
8-bit xor8	1	-
1-bit xor2	-	144
8-bit xor6	-	8

The synthesis result is generated by Xilinx ISE as shown in Table 5.7, where the macro statistics is provided by Xilinx library. From the result, we can see the amount of

XOR logic used by cipher KHAZAD is greatly larger than that of BSPN, which leads to the major difference of resource consumption between these two ciphers.

5.5 Energy Consumption Analysis

In this section, we will introduce the method of evaluating the design's power and present the power estimation results.

5.5.1 Power Estimation Using Power Analyzer

FPGA power includes two parts: static power (or quiescent power) and dynamic power [108]. Static power is the intrinsic power of the device and cannot be changed. It exists once the chip is powered on, even if there is no activity in the device. It includes transistor leakage, power consumed internally, and power dissipated in external termination resistors. Dy namic power is caused by the switching activity of CMOS transistors. Dynamic pow er is only consumed when the state of transistors changes, which depends on the specific implementation of the design. A design can consume less power if it is implemented in an appropriate way. That is to say, we can develop the design's energy efficiency for the part of dynamic power.

Xilinx Xpower is a tool providing the power estimation for a FPGA design. The power of a design can only be analyzed after the process of "place and route". Xpower calculates the power based on the switching activity of the transistors. Any component in FPGA with switching activity has a corresponding capacitance model used for the power calculation. There are two methods for power estimation using Xpower: (1) a rough estimation by setting an expected toggle rate; (2) a more accurate estimation by providing detailed transistor switching activity. Here, we choose the second method to get a relatively accurate result for our designs.

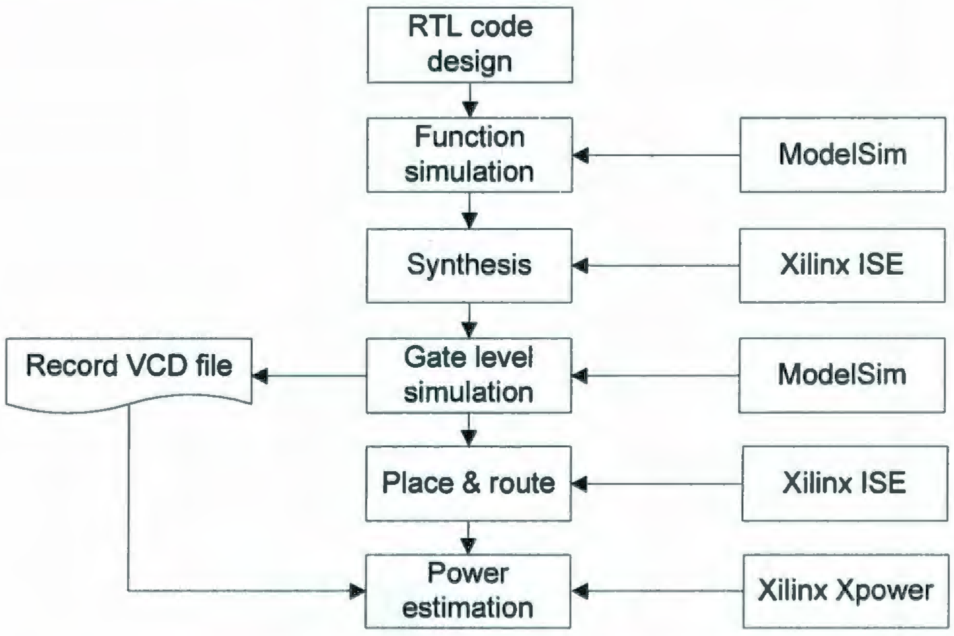


Fig. 5.18. Power estimation flow by Xpower.

The flow for the power estimation is shown in Fig. 5.18, where value change dump file (VCD) is a file to record the signal state at different timeslots and can be obtained by simulation. The switching activity of the circuit can be recorded in VCD file when we simulate the gate-level netlist of the design. Another purpose of the simulation at gate level is to examine whether the register transfer level (RTL) code is successfully synthesized into gate level netlist. Finally, Xpower reads the VCD file and generates the power estimation report. It is worth noting that the accuracy of the power estimation depends on the accuracy of the switching data. The simulation cases should be as extensive as possible so that the recorded file can include relatively complete switching information.

5.5.2 Power Estimation Result

We simulate the gate-level netlist of designs and calculate the energy consumed by processing encryption or decryption of a block size (64 bits) of data. The energy calculation does not include the key scheduling process since it is not frequently operated. We investigate the energy efficiency of cipher KHAZAD and BSPN, as well as the effects of different structures and target devices.

1) LUT vs. BRAM

We do the power estimation on cipher BPSN-64 by different implementation methods. The estimation result is shown in Table 5.8, which is obtained by simulating the design with 50 MHz clock. From the table, we can see that:

- Although the power of design using eight S-boxes is larger than that of using one S-box, the relation of energy cost is the reverse. This results because the operation time for one block size of data using the eight S-boxes method is much faster than using the "1 S-box" method. Therefore, the energy cost for the "8 S-box" method is less although the power is higher.
- Under the condition of using the same number of S-boxes, the design using BRAM consumes less energy than the design using LUT. This can be explained from the fact that the BRAM is designed and optimized directly for memory function, while LUT is designed for general-purpose usage.

In this table, we also present the results using two different FPGA devices: xc3s200 and xc2vp100. It is worth noting that, for the same design, energy consumption may vary for different devices. Compared to xc3s200, xc2vp100 is a relatively high-end FPGA product, which offers larger space for more complicated designs. From the results, we can see that the energy cost of xc2vp100 is much higher than that of xc3s200. This result is caused by many factors, such as the different working voltage and parameters of manufacturing technology. Here, we just use this example to show that choosing an appropriate device is an important factor when considering low energy cost hardware implementation.

Table 5.8. Power estimation for BSPN-64.

Device	S-box	Cycles	Power (mW)	Energy (nJ) (per block)
xc3s200	8 LUT	10	66.35	$E = \frac{66.35\text{mw} \times 10\text{cycle}}{50\text{M}} = 13.27\text{ nJ}$
	1 LUT	73	12.33	$E = \frac{12.33\text{mw} \times 73\text{cycle}}{50\text{M}} = 18.00\text{ nJ}$
	8 BRAM	10	39.85	$E = \frac{39.85\text{mw} \times 10\text{cycle}}{50\text{M}} = 7.97\text{ nJ}$
	1 BRAM	73	9.20	$E = \frac{9.20\text{mw} \times 73\text{cycle}}{50\text{M}} = 13.43\text{ nJ}$
xc2vp100	8 LUT	10	401.16	$E = \frac{401.16\text{mw} \times 10\text{cycle}}{50\text{M}} = 80.23\text{ nJ}$
	1 LUT	73	68.58	$E = \frac{68.58\text{mw} \times 73\text{cycle}}{50\text{M}} = 95.75\text{ nJ}$
	8 BRAM	10	210.52	$E = \frac{210.52\text{mw} \times 10\text{cycle}}{50\text{M}} = 42.10\text{ nJ}$
	1 BRAM	73	46.08	$E = \frac{46.08\text{mw} \times 73\text{cycle}}{50\text{M}} = 67.28\text{ nJ}$

2) KHAZAD vs. BSPN

We simulate cipher KHAZAD and BSPN-128 with 40MHz clock and calculate their energy cost. The power estimation is based on the structure of BRAM since it is the most energy efficient method as we discussed before. In this part, we investigate two scenarios: (1) the energy cost of full cipher hardware implementation and (2) the energy cost of hardware and software cooperation.

Table 5.9. Power estimation of full hardware (data operation).

Cipher	S-box	Cycles	Power (mW)	Energy (nJ) (per block)
BSPN-128	8 BRAM	10	38.72	$E = \frac{38.72\text{mw} \times 10\text{cycle}}{40\text{M}} = 9.68$
KHAZAD		10	48.67	$E = \frac{48.67\text{mw} \times 10\text{cycle}}{40\text{M}} = 12.17$
BSPN-128	1 BRAM	73	12.50	$E = \frac{12.61\text{mw} \times 73\text{cycle}}{40\text{M}} = 22.81$
KHAZAD		73	17.01	$E = \frac{17.01\text{mw} \times 73\text{cycle}}{40\text{M}} = 31.04$

Table 5.10. Power estimation of full hardware implementation (key scheduling).

Cipher	S-box	Cycles	Power (mW)	Energy (nJ) (per block)
BSPN-128	8 BRAM	83	36.16	$E = \frac{36.16\text{mw} \times 83\text{cycle}}{40\text{M}} = 75.03$
KHAZAD		15	39.97	$E = \frac{39.97\text{mw} \times 15\text{cycle}}{40\text{M}} = 14.99$
BSPN-128	1 BRAM	732	12.21	$E = \frac{12.21\text{mw} \times 732\text{cycle}}{40\text{M}} = 223.44$
KHAZAD		83	17.81	$E = \frac{17.81\text{mw} \times 83\text{cycle}}{40\text{M}} = 36.96$

- **Results of Full Cipher Hardware Implementation**

To make a fair comparison, the energy cost of key scheduling and data en/decryption are calculated separately. The energy estimation results are shown in Table 5.9 and Table 5.10, respectively.

From the tables we can see that, for en/decrypting one block of data, cipher KHAZAD consumes much more energy cost than BSPN. Also, for the same cipher, the 8 S-box BRAM structure is more energy efficient. BSPN with eight BRAM S-boxes can achieve the least energy consumption among these four implementations. However, for the round key scheduling, cipher BSPN costs much more energy than KHAZAD, which is because the key scheduling process in BSPN is more complicated than KHAZAD. Note that the key scheduling will only happen at the initial period of key setup and when there comes a new updated key. As the key is updated infrequently in WSNs, the data en/decryption process is the major factor for energy consumption.

- **Results of Cooperation of Hardware and Software Implementation**

The energy cost of hardware and software cooperation includes two major parts: (1) the energy cost of key scheduling by software and (2) the energy cost of data en/decryption by hardware. We implement cipher KHAZAD and BSPN-128 in assembly language on an 8-bit microprocessor Atmega128 as other software implementations in Chapter 4. The energy results are shown in Table 5.11, which includes detailed energy cost of the ciphers: key scheduling, data en/decryption and linear transformation. The energy cost results of the hardware implementation with only the data en/decryption are shown in Table 5.12. Compared to a full cipher implementation, the cipher with only data path can save at least 30% of the energy cost.

Table 5.11. Software implementation comparison of 128-bit key cipher.

Cipher	Process	CPU (cycle)	Energy Cost (nJ) (per block)
BSPN-128	Key schedule	9644	$E = \frac{13.8\text{mw} \times 9644\text{cycle}}{8\text{M}} = 16635.9$
	En/decryption	796	$E = \frac{13.8\text{mw} \times 796\text{cycle}}{8\text{M}} = 1373.1$
	Linear Trans.	14	$E = \frac{13.8\text{mw} \times 14\text{cycle}}{8\text{M}} = 24.15$
KHAZAD	Key schedule	7057	$E = \frac{13.8\text{mw} \times 7057\text{cycle}}{8\text{M}} = 12173.3$
	En /decryption	2491	$E = \frac{13.8\text{mw} \times 2491\text{cycle}}{8\text{M}} = 4296.975$
	Linear Trans.	264	$E = \frac{13.8\text{mw} \times 264\text{cycle}}{8\text{M}} = 455.4$

Table 5.12. Power estimation of only data path hardware implementation.

Cipher	S-box	Cycle	Power (mW)	Energy (nJ) (per block)
BSPN-128	8 BRAM	10	20.46	$E = \frac{20.46\text{mw} \times 10\text{cycle}}{40\text{M}} = 5.12$
KHAZAD		10	38.39	$E = \frac{38.39\text{mw} \times 10\text{cycle}}{40\text{M}} = 9.60$
BSPN-128	1 BRAM	73	8.06	$E = \frac{8.06\text{mw} \times 73\text{cycle}}{40\text{M}} = 14.71$
KHAZAD		73	11.86	$E = \frac{11.86\text{mw} \times 73\text{cycle}}{40\text{M}} = 21.64$

It is worth noting that, the energy cost above is the dynamic energy consumption and our analysis is based on a reconfigurable sensor node. Although the energy consumption of software implementation is much more than the hardware implementation, it does not mean an FPGA should be incorporated into a sensor node, which is not originally designed as a reconfigurable sensor node. The reason is that an FPGA is not a power saving device, as it consumes much more static energy than an especially designed CPU ASIC.

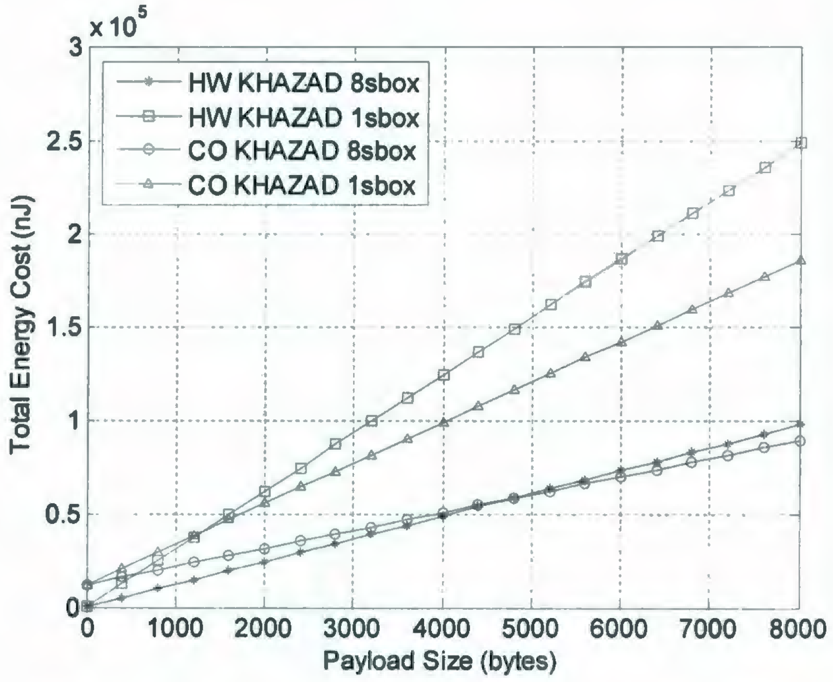


Fig. 5.19. HW implementation vs. HW & SW cooperation (KHAZAD).

The energy cost of cooperation is calculated by combining the software energy cost and hardware cost together, as shown in Fig. 5.19 and Fig. 5.20 for KHAZAD and BSPN-128, which are plotted based on only one key scheduling process at the beginning of key setup. We can see that cipher BSPN achieves better energy performance than KHAZAD under the same conditions. For both cipher KHAZAD and BSPN, the implementation using eight S-boxes achieves better energy performance than using one S-box. In addition, as the number of bytes of data increases, the cooperative method achieves better energy cost under the condition using the same number of S-boxes. The implication is that for sufficiently large amounts of data encrypted between re-keying events, a cooperative approach is preferred over a pure hardware implementation and greatly preferred over a pure software implementation (given the data from Table 5.11).

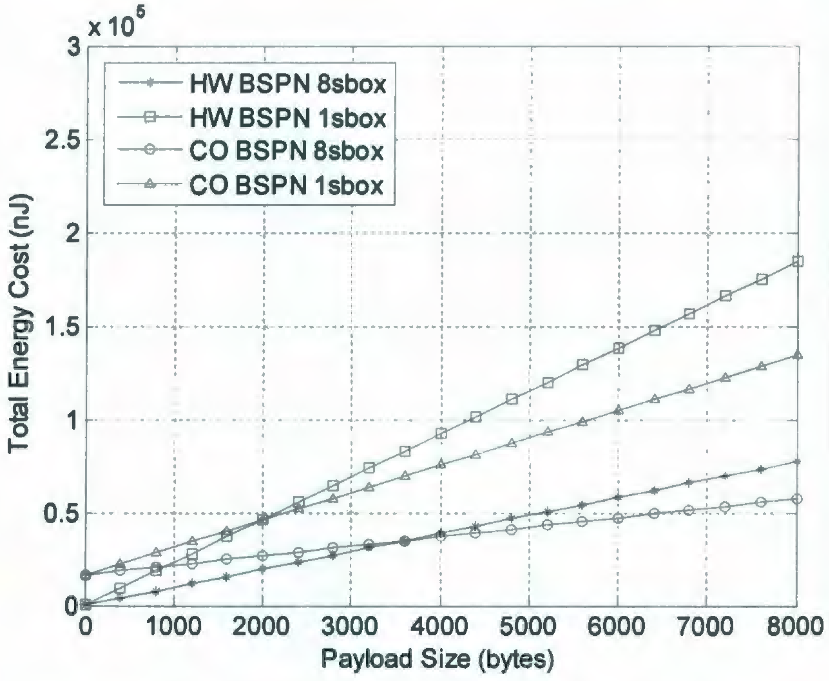


Fig. 5.20. HW implementation vs. HW & SW cooperation (BSPN-128).

5.6 Summary

In this chapter, we have focused on hardware implementation of involutional block ciphers in reconfigurable WSNs. We first introduce the background of reconfigurable wireless sensor networks and its state of art. In a reconfigurable WSN, the cryptographic algorithm implemented by hardware can achieve higher flexibility and speed. We recommend using involutional block ciphers because both the encryption and decryption can share the same circuit. Two involutional block ciphers are chosen as candidates:

KHAZAD and BSPN (with 64-bit key size and 128-bit key size). We focus on design of energy efficient ciphers for the target device of FPGA, considering different design factors: the structure of the design, the resource utilization of FPGA and the implementation architecture suc h as the number of S-box circuits. By implementing cipher KHAZAD and BSPN with different methods, we further analyze the dynamic power of circuit and calculate the energy cost of the design. The power estimation is obtained by using Xpower, a power analysis tool provided by Xilinx. We compare the implementation of two scenarios: full cipher implemented by hardware and cipher implemented by cooperation of hardware and software. Results show that (1) cipher BSPN achieves better energy performance than KHAZAD with the major reason of the simplicity of linear transformation; (2) ciphers using BRAM to implement 8 parallel S-boxes can achieve better energy performance, especially when software and hardware are working together.

Chapter 6

ENERGY COST IN CRYPTOGRAPHIC SCHEMES

This chapter presents the energy cost analysis for different cryptographic schemes in a WSN. After investigating the energy cost of cryptographic algorithms in the previous chapters (software-oriented in Chapter 4 and hardware-oriented in Chapter 5), we investigate cryptographic schemes in this chapter, which directly affect the communication energy cost for a wireless sensor node. The energy limitation of a sensor device is a challenging constraint in WSNs. Researchers have focused on studying cryptographic algorithms that directly impact on the computational energy cost in a WSN, by evaluating existing cryptographic algorithms [35][85] or proposing new energy efficient ciphers [86][87]. However, these studies are mainly based on the algorithm itself, independent of the cryptographic scheme. There are proposals that focus on the cryptographic scheme, such as TinySec [63], SPINS [62] and other encryption schemes developed from TinySec [70][67][68][69]. Although these proposed schemes do recommend a mode of operation for block ciphers applied to WSNs, no effort has been made to investigate the energy efficiency of a sensor node in a noisy environment, considering both the cryptographic algorithm and the cryptographic scheme as a whole.

In this chapter, we explore the effect of packet size, the distribution of the initialization vector and the channel quality on the energy consumption of cryptographic communication schemes in WSNs. We present the integrated performance evaluation for different ciphers and schemes by developing an analysis model for a sensor node, which considers various factors, and the appropriateness of the model is supported by simulation results. The cryptographic schemes to be considered will all be based on block ciphers and software-oriented environment. In conclusion, we recommend using the lightweight block cipher BSPN with the cipher feedback scheme to encrypt the data thereby achieving energy efficiency without compromising the security.

As we discussed before, in cryptographic communication schemes, encrypted data (or ciphertext) takes the place of the original payload (or plaintext) to achieve the data confidentiality. A message authentication code (MAC) functions as the cryptographic checksum, providing both data integrity and data authentication. We take a packet as correctly transmitted only when the recalculated MAC value equals the transmitted MAC value, and assume the difference is caused either by a noisy channel or by a malicious attacker.

6.1 Factors Affecting the Communication Cost

Before evaluating the energy cost of different cryptographic schemes, we first discuss several factors, which will affect the communication energy cost when cryptography is applied to a WSN.

6.1.1 Mode of Operation

A mode of operation is a method to make use of a symmetric key block cipher when operating on a large bulk of data [61]. The mode determines how to use the block cipher to derive the ciphertext and has an impact the communication energy cost in WSNs. CBC mode is a common selection for encrypting large amounts of data and is proposed for use in the TinySec scheme on a per-packet basis [63]. In [63], to reduce the size of ciphertext to the same number of bits as plaintext, the ciphertext stealing technique [97] is used. However, the ciphertext size cannot be decreased below the block size when the amount of plaintext is less than the block size of the cipher. Counter mode and CFB mode make the encryption process similar to a stream cipher, which generates the same number of ciphertext bits as plaintext bits, and counter mode is proposed for use in WSNs by the SPINS scheme [62]. Selection of an appropriate mode of operation for the block cipher is critical: issues such as mode related security weaknesses, error propagation, and loss of data synchronization must be considered in the context of practical WSN applications.

There are also many advanced modes of operation proposed in recent years, which integrate the MAC generation with encryption using a block cipher. This includes offset codebook (OCB) mode [65], counter with CBC-MAC (CCM) mode [27], and Galois/counter (GCM) mode [64]. Some researchers propose to use an advanced mode in WSNs since it can accelerate the speed of operation and save some of the computational energy cost of MAC generation. However, it is worth noting that in a WSN, the overall energy cost will not be significantly affected, as our analysis results will show later, since communication energy cost is the major factor affecting the energy consumption of encryption schemes applied to a sensor node.

6.1.2 Initialization Vector

The initialization vector (IV) plays an important role in cryptographic mechanisms, and although the content of the IV itself is not kept confidential, the use of a particular IV value should not be repeated. In CBC mode, the IV is used to produce the first block of ciphertext, so that the data can be randomized thereby effectively eliminating the repetition of data input to the cipher - an important security consideration [61]. In counter mode, the IV is used to initialize the counter value and, since this must be done periodically to ensure synchronization between the ends of the communication, it is also important in this case, that IV is not repeated. In CFB mode, the IV can be used to reset the feedback at the block cipher input. For security purposes, although the size of IV should ideally be similar to the block size of the cipher, to save the communication energy costs, some schemes reduce the size of IV. For instance, TinySec [63] reduces the effective IV size to 16 bits, which allows for the possibility of a repeated IV. Hence, there may be a potential weakness from a cryptographic point of view.

In WSNs, IV greatly affects the energy performance of a scheme because of two factors: (1) when transmitted between nodes it consumes communication energy and (2) the ciphertext will not be decrypted correctly if the IV is not reliably known by both communicating parties.

6.1.3 Payload Size and Channel Quality

To prevent malicious attacks, a packet will be discarded when the MAC verification fails. However, poor channel quality may also lead to an unsuccessful MAC verification. In addition, a packet with longer payload size will more likely lead to the packet being corrupted in a noisy channel. Hence, both the payload size of a packet and the quality of the communication channel will influence the amount of data that will be lost and consequently will affect the energy efficiency of a sensor node.

6.2 Description of Cryptographic Schemes

When a link layer cryptographic scheme is applied in a WSN, the IV is generated by the transmitter and must be known to both the communication sides. Hence, the origination and distribution of IV content are critical. In this section, we explore several typical methods that can be used to encrypt in WSNs and the associated IV distribution processes. Typically, IV distribution methods in WSNs have been classified into two types: (1) including IV in each packet being transmitted [63] and (2) transmitting IV in an independent packet periodically [62]. Here we also present another IV distribution method, which is based on previous ciphertext collection so that it does not require the designated IV information to be transmitted. We have selected several cryptographic schemes encompassing different modes of operation and IV distribution approaches, and analyze them in terms of their energy efficiency when applied to WSNs. First, we describe the schemes.

6.2.1 CBC Mode + Implicit IV Scheme

In this scheme, the IV is included in each packet and CBC mode is applied to a block cipher for encryption. We use *implicit IV* as the name to represent a general approach that transmits the IV in each packet so that the receiver can recover the IV directly from the received packet to use in the decryption. In this scheme, the IV size is the same as the block size of the cipher. The disadvantage of this scheme is that extra bits are transmitted, which increases the communication energy cost of each packet. The packet format is shown in Fig. 6.1. In the *implicit IV* scheme, the packet will be decrypted correctly when it is transmitted and received without error (i.e., the MAC verification passes).

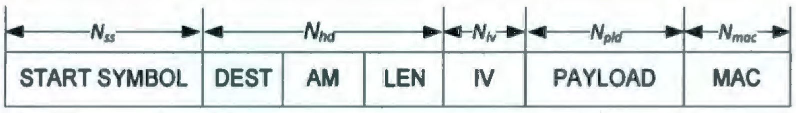


Fig. 6.1. Packet format of implicit IV scheme.

6.2.2 TinySec Scheme

TinySec [63] uses information in each packet to determine the IV and applies CBC mode to a block cipher for encryption. Although similar to the implicit IV approach, it minimizes transmitted bits by using some information for IV that is already available in the packet. The packet format is shown in Fig. 6.2, where SRC is the source address of the sender and CTR is a 16-bit counter. This scheme reduces the energy cost by minimizing the number of bits being transmitted. The IV is taken from the fields from DEST to CTR. Since the effective size of the IV is much less than the block size, it is conceivable that, in some contexts, IV will be repeated over a long duration of time. Nevertheless, it is argued that the semantic security of this scheme is sufficient [63]. The energy cost is also minimized by applying CBC mode with ciphertext stealing technique [97] to encrypt the data.

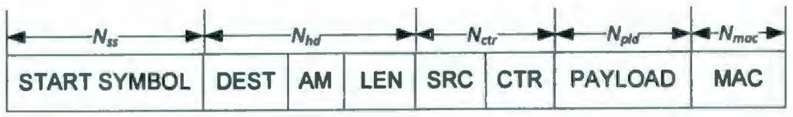


Fig. 6.2. Packet format of TinySec scheme.

6.2.3 Counter Mode + Periodic IV without ACK Scheme

The scheme uses counter mode of a block cipher for encryption, which requires a periodic transfer of IV to ensure that the encryption and decryption process remain synchronized. The SPINS scheme [62] proposes a similar approach, providing semantic security without transmission overhead. Without the periodic transfer of IV, a lost data packet will result in a permanent loss of cipher synchronization. For this purpose, IV can be communicated periodically within a special IV packet, separate from data packets. A newly received IV initializes the counter and subsequently the count increases by one for each block of data encrypted.

In our study, it is assumed that a corrupted IV packet is simply discarded, resulting in corruption of the subsequent data packets until a new IV is successfully exchanged. The packet format is shown in Fig. 6.3, which includes both the IV packet and the data packet. This scheme reduces the energy cost of a packet over the schemes that include IV in each packet, but results in a high probability that the packet is not properly decrypted when the channel quality is poor. It can be explained that, first, subsequent packets cannot be decrypted correctly until the next IV packet is transmitted if the IV packet has been received with errors. In addition, even if the IV packet is received correctly, one data packet being discarded due to errors will affect the decryption of the following packets until the next IV packet is transmitted.

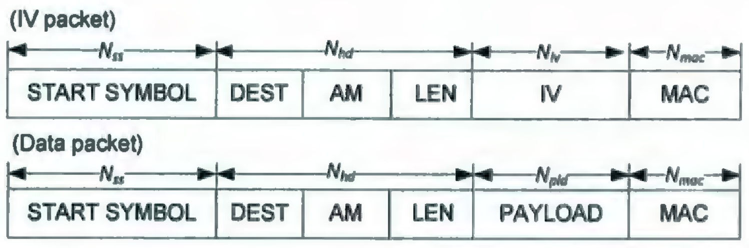


Fig. 6.3. Packet format of Periodic IV without ACK scheme.

6.2.4 Counter Mode + Periodic IV with ACK Scheme

This scheme is similar to the previous scheme except that a stop-and-wait ARQ protocol is used to guarantee the IV packet is being correctly received. Counter mode is also used in this scheme. Packet formats are shown in Fig. 6.4 for the IV packet, the acknowledgement (ACK) packet and the data packet. In the ACK packet, the "ACK" field represents the acknowledgement information. The energy cost will be higher than the *periodic IV without ACK* scheme, because two parts of energy are introduced by the ARQ scheme: first, reception of the ACK packet will consume communication energy; second, the IV packet may be retransmitted several times if the channel quality is particularly poor. It is worth noting that, although the IV packet is ensured to be correctly received, corrupted data packets will still be discarded resulting in a loss of synchronization in the decryption counter and, hence, the following data packets will not be decrypted correctly until the next IV distribution process.

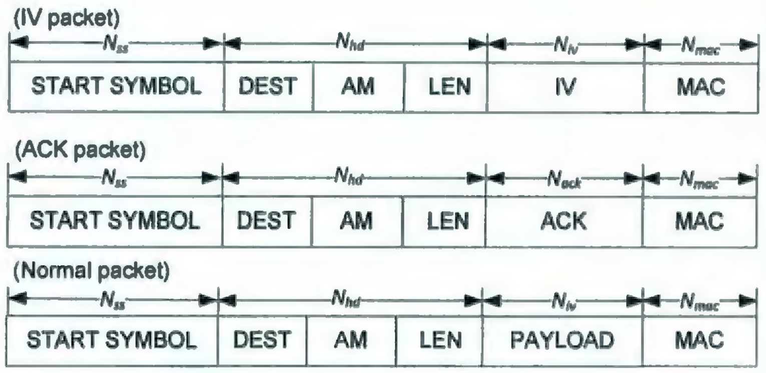


Fig. 6.4. Packet format of periodic IV with ACK scheme.

6.2.5 CFB Mode + Ciphertext Based IV Scheme

We propose this cryptographic scheme based on using previously transmitted ciphertext as the IV for the next encryption. A block cipher can be configured for CFB mode, where the data in a packet is encrypted by XORing the plaintext block with the output of the block cipher, which has used the previous ciphertext block as input. We shall consider an approach that resets the feedback at the start of each packet by using the preceding ciphertext from the payloads of previous packets as an IV block to be fed to the block cipher input. Unlike other schemes, this scheme does not consume extra energy for either including IV bits in each packet or transmitting separate IV packets. The packet format is shown as Fig. 6.5. In this scheme, the current packet being decrypted depends on both the packet itself and the previous packets. Note the initial IV used to produce the first bits of ciphertext can be exchanged during the key establishment phase.

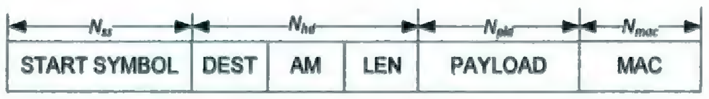


Fig. 6.5. Packet format of CFB scheme.

6.2.6 Other Schemes

Other schemes combining cipher modes (e.g. CBC, counter, and CFB) with different methods of IV agreement are possible but are not investigated in our research. For example, it is possible to have a scheme, which uses counter mode with an IV sent in every packet to re-initialize the counter value for decryption. Another example would be the application of CBC with IV periodically resent with IV packets. Still another example is CFB mode, which does not synchronize to a block at the start of each packet. In this thesis, we have chosen several typical cryptographic schemes to investigate.

6.3 Analysis of Cryptographic Schemes

In this section, we first focus on constructing an analysis model based on the assumption of fixed size packets to evaluate the energy performance of the link layer cryptographic schemes for software-oriented application environments. Subsequently, the study will be extended to variable size packets under different probability distributions. In our analysis, we take binary symmetric channel (BSC) as the channel model to study the energy performance of different encryption schemes in the presence of noise.

6.3.1 General Analysis for Fixed Size Packets

1) Probability of the Error-free Packets

The probability that a packet has one or more bit errors is decided by two factors: the probability of error for each bit (p_e) and the size of the packet (N_{pkt}). The probability that a received packet has no errors (P_o) is expressed as

$$P_o = (1 - p_e)^{N_{pkt}}, \tag{6.1}$$

where it is assumed that bit errors occur randomly and independently. The determination of N_{pkt} is listed in Table 6.1 considering different cryptographic schemes.

Table 6.1. Packet size for different schemes.

Scheme	Type	Packet size
CBC + Implicit IV	Data	$N_{pkt} = N_{ss} + N_{hd} + N_{iv} + N_{pld} + N_{mac}$
TinySec	Data	$N_{pkt} = N_{ss} + N_{hd} + N_{ctr} + N_{pld} + N_{mac}$
Counter + Periodic IV without ACK	Data	$N_{pkt} = N_{ss} + N_{hd} + N_{pld} + N_{mac}$
	IV	$N_{ivpkt} = N_{ss} + N_{hd} + N_{iv} + N_{mac}$
Counter + Periodic IV with ACK	Data	$N_{pkt} = N_{ss} + N_{hd} + N_{pld} + N_{mac}$
	IV	$N_{ivpkt} = N_{ss} + N_{hd} + N_{iv} + N_{mac}$
	ACK	$N_{ackpkt} = N_{ss} + N_{hd} + N_{ack} + N_{mac}$
CFB + Ciphertext based IV	Data	$N_{pkt} = N_{ss} + N_{hd} + N_{pld} + N_{mac}$

2) Energy Calculation

The energy cost calculation has already been discussed in Chapter 3. The total energy cost of one packet depends on the type of the packet. For example, in the schemes with periodic IV, the IV packet and ACK packet energy cost does not include the encryption energy since the IV and ACK information is not encrypted. This is summarized in Table V. For the IV packet and ACK packet in the calculation of E_{tx} and E_{rx}, respectively, N_{pkt} is replaced by N_{ivpkt} and N_{ackpkt}, respectively, as shown in Table 6.2.

Table 6.2. Energy cost of different packet types.

Packet Type	Energy per packet
Data packet	$E_{data} = E_{enc} + E_{tx} + E_{mac}$
IV packet	$E_{iv} = E_{tx} + E_{mac}$
ACK packet	$E_{ack} = E_{rx} + E_{mac}$

3) Expected Number of Valid Data Bits

As discussed in Chapter 3, we choose the energy efficiency, η, as defined by the average number of data bits successfully transmitted per Joule by the sensor node as the metric to evaluate the performance of different schemes. This can be expressed as

$$\eta = n \times N_{pld} \times P_{va} \quad , \tag{6.2}$$

which depends on three factors: the number of the transmitted data packets, n, the payload size, N_{pld}, and the probability that the packet is correctly decrypted, P_{valid}. Given parameters of the packet and sensor node energy of 1 Joule, the first two parameters can be calculated directly. In the following section, we will focus on the factor P_{valid}, which varies for different schemes.

6.3.2 Probability of Valid Packets for Different Schemes

The probability that a packet is successfully received and decrypted (P_{valid}) strongly depends on the specific scheme applied. In determining an expression for P_{valid}, we assume that the bit errors occur independently with a probability given by the bit error rate p_e. We consider now the probability for different schemes.

1) CBC Mode + Implicit IV Scheme

In this scheme, since an IV is included in each data packet, the packet can be decrypted correctly when every bit of the packet is received correctly. Since CBC mode is used, P_{valid} is given by P_o in (6.1) with

$$N_{pld} = \left\lceil \frac{N_{ptext}}{b} \right\rceil \times b, \tag{6.3}$$

where N_{pld} and N_{ptext} represent the number of payload bits and plaintext bits, respectively. This implies that the payload size is increased due to the application of CBC mode.

2) TinySec Scheme

Since the TinySec scheme uses CBC mode of operation with the ciphertext stealing technique, P_{valid} is given by P_o in (5) with

$$N_{pld} = \begin{cases} b \quad , & \text{if } N_{ptext} < b \\ N_{ptext}, & \text{if } N_{ptext} \geq b. \end{cases} \tag{6.4}$$

In this case, only when the amount of plaintext is less than one block, is the payload size increased due to application of the encryption.

3) Counter Mode + Periodic IV without ACK Scheme

In this scheme, the probability that a data packet of size N_{pkt} is correctly received is

$$P_{data} = (1 - p_e)^{N_{pkt}}, \tag{6.5}$$

where N_{pkt} is the number of bits in the data packet and, for an IV packet of size N_{ivpkt}, the probability that the packet is correctly received is

$$P_{iv} = (1 - p_e)^{N_{ivpkt}}. \tag{6.6}$$

The probability that the data packet can be decrypted correctly is based on the probability that the previous IV packet and all previous data packets following the IV packet are successfully received and is given by

$$P_{valid} = \frac{P_{iv} \times (P_{data} - P_{data}{}^{K+1})}{K \times (1 - P_{data})}, \tag{6.7}$$

where K represents the number of data packets sent for each IV packet and is assumed to be constant. We can see that P_{valid} is determined by both K and the channel quality. If the channel is very noisy, there will be high energy cost because bit errors result in many data bits being discarded when MAC verification fails.

4) Counter Mode + Periodic IV with ACK Scheme

Since the acknowledgement ensures the receiver side knows the initial value of the IV, the probability of the data packet being correctly decrypted (P_{valid}) depends on the probability of data packet being transmitted correctly (P_{data}) and the period of the IV packet (K). Similar to the *periodic IV without ACK* scheme, if one data packet is lost or transmitted with error, the counter value will lose the synchronization and the following packets will not be decrypted correctly. The factor K affects how many of the following packets will lose synchronization. Based on receiving the IV packet correctly, the probability that the data packet can be decrypted correctly for this scheme is expressed by

$$P_{valid} = \frac{P_{data} - {P_{data}}^{K+1}}{K \times (1 - P_{data})} \quad . \tag{6.8}$$

Note that two parameters, P_{iv} (probability of the IV packet transmitted correctly) and P_{ack} (probability of the ACK packet transmitted correctly), directly impact the number of successfully transmitted data packets (n in (6.2)), although they do not occur in the expression for P_{valid}. For each unit of energy, the less energy is consumed by the IV distribution process, the more energy can be provided to the data transmission. That is to say, with large value of P_{iv} and P_{ack}, more energy can be used to transmit data packets, which directly results in large value of n and η in (6.2). On the contrary, if the channel is very noisy, the IV packet and ACK packet are very likely to be corrupted and there will be much energy consumed during the process of the IV distribution.

5) CFB Mode + Ciphertext Based IV Scheme

For the CFB scheme, the probability of a data packet being decrypted correctly is given by

$$P_{valid} = (P_{data})^{\gamma+1}, \tag{6.9}$$

where the parameter γ is determined by

$$\gamma = \begin{cases} \lceil b/N_{pld} \rceil & \text{if } N_{pld} < b \\ 1 & \text{if } N_{pld} \geq b \end{cases} \tag{6.10}$$

and represents the number of previous packets whose ciphertext is used for the IV and therefore affects the decryption of the current packet. The value of γ will be large for a small payload size, which will decrease the probability of the current packet being decrypted correctly. However, the small payload size will make the packet itself more likely to be transferred correctly, which counteracts the effect of the large γ value.

6.4 Analysis Results

In this section, we present the results of the analysis model applied to the identified cryptographic schemes.

6.4.1 Parameters

The values of the parameters we have used in the analysis are listed in Table 6.3. We use the cipher AES, as it is the most applied block cipher today. However, similar results would result for other symmetric key ciphers. Physical parameters for the sensor device are derived from the specifications of the commercial product Mica2 [82]. In addition, we use several bit error rate values to represent the different channel qualities.

Table 6.3. Parameters used in the analysis.

Object	Parameter	Value	Unit
Block cipher (AES)	C_{enc}	3266	cycles
	b	128	bits
Sensor board	P_{cpu}	13.8	mW
	f_{cpu}	8	MHz
	I_{tx}	27	mA
	I_{rx}	10	mA
	U	3.3	V
	R	38400	bps
Packet	N_{ss}	8	bytes
	N_{hd}	4	bytes
	N_{iv}	4	bytes
	N_{ack}	1	bytes
	N_{pld}	from 1 to 30	bytes
	N_{mac}	4	bytes
	K	10	packets
Channel	BER	$5\times10^{-4},10^{-4},10^{-5},0$	-

6.4.2 Analysis Results for Different Schemes

The analysis results of the five schemes present the energy efficiency for different payload sizes. In the analysis, we also present the results under different channel bit error rates. As expected, as BER increases, the energy efficiency decreases due to the necessity of discarding many corrupted packets. All cryptographic schemes have the same trend for the relationship between BER and energy efficiency.

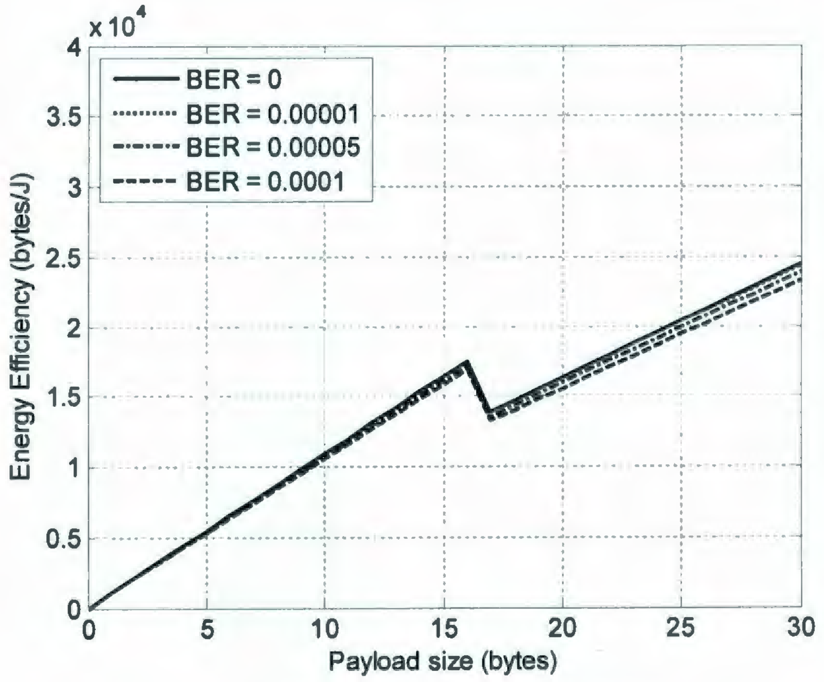


Fig. 6.6. Analysis of Implicit IV scheme.

1) CBC + Implicit IV Scheme

In Fig. 6.6, we can see that the curve for the *implicit IV* scheme shows a sawtooth shape, because CBC mode is used in this scheme and the size of encryption result is always an integer multiple of the block size. Here the size of IV is 128 bits, which is the same with the block size of AES. The extra bits being transmitted due to padding the plaintext data out to a multiple of a block size consume energy while providing no informational content to the communication.

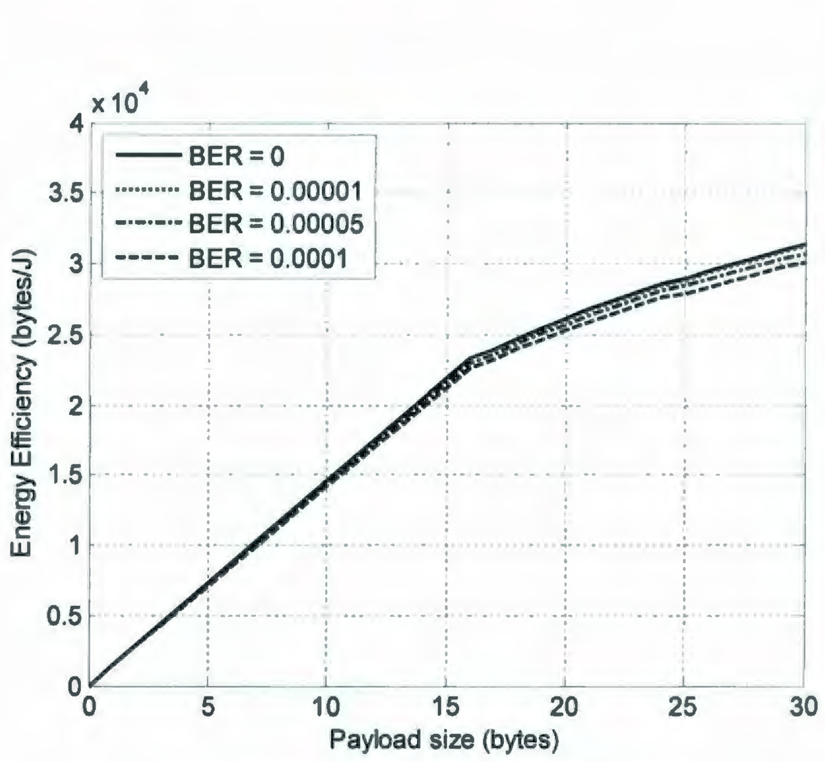


Fig. 6.7. Analysis of TinySec scheme.

2) TinySec Scheme

Fig. 6.7 shows the results for the specific format of TinySec, which uses CBC (with ciphertext stealing) and includes the IV in each packet. As seen in the figure, the slope decreases as the payload size increases. This occurs because the ratio of the IV size to the packet size decreases. Since AES uses a 128-bit block size, for payload sizes of 1 to 16 bytes, the plaintext is padded out to 128 bits and a full 128 bits of ciphertext will be produced resulting in a straight line for the corresponding portion of the graph. When the payload size is larger than the block size, the transmitted ciphertext size is the same as the plaintext size, since ciphertext stealing can be used.

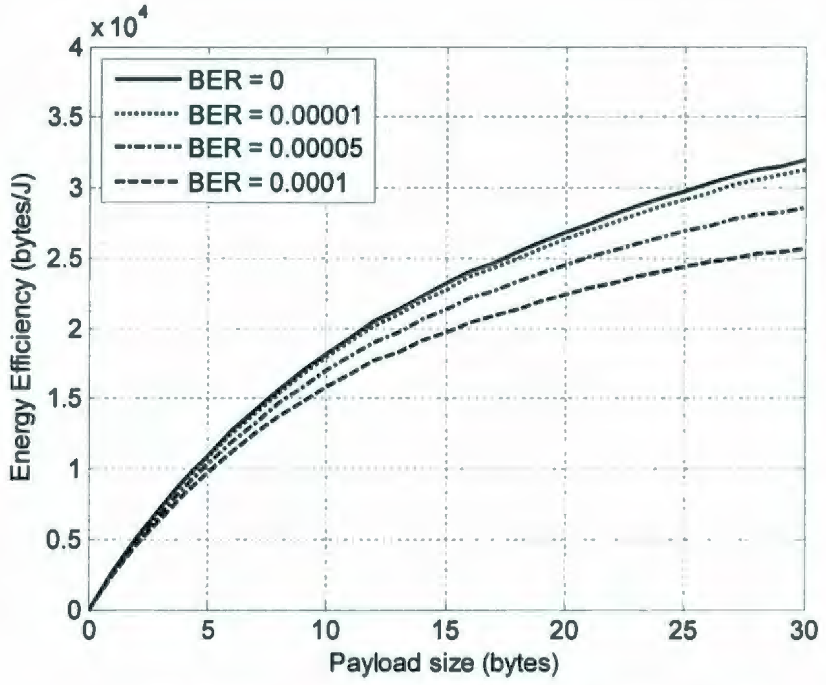


Fig. 6.8. Analysis of Periodic IV without ACK scheme.

3) Counter Mode + Periodic IV without ACK scheme

In Fig. 6.8, it can be seen that, for the *periodic IV without ACK* scheme applying counter mode, at high bit error rate, the energy efficiency decreases dramatically compared to lower bit error rate. This indicates that the periodic IV approach has a relatively poor performance in a poor channel condition. When the channel quality is not good, both the data packet and IV packet have a large probability of being discarded due to error, which will affect the decryption of the following data packets, which relies on synchronization of the counter value between transmitter and receiver.

4) Counter Mode + Periodic IV with ACK scheme

As shown in Fig. 6.9, the trend of this scheme is similar to the *periodic IV without ACK* scheme. The acknowledgement does not improve the performance of the scheme very much. This is because when the channel quality is poor, the data packet also has a large probability of being corrupted, which will affect the following data packets until a new IV is established.

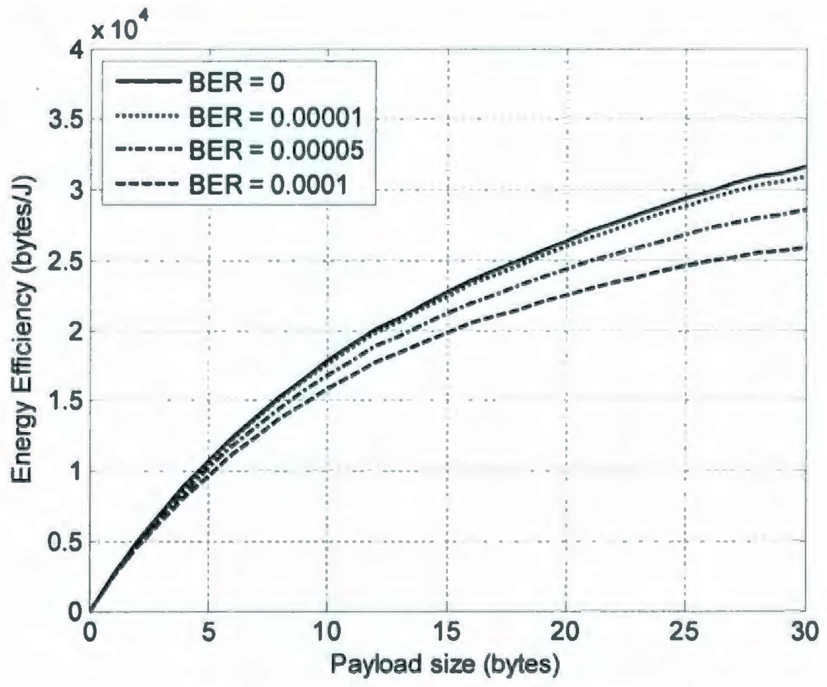


Fig. 6.9. Analysis of Periodic IV with ACK scheme.

5) CFB Mode + Ciphertext based IV scheme

As shown in Fig. 6.10, that the CFB scheme achieves better energy efficiency for all the payload sizes compared to other schemes. We can also see that, compared to the period IV schemes, there is little spread in the curves when bit error rate increases.

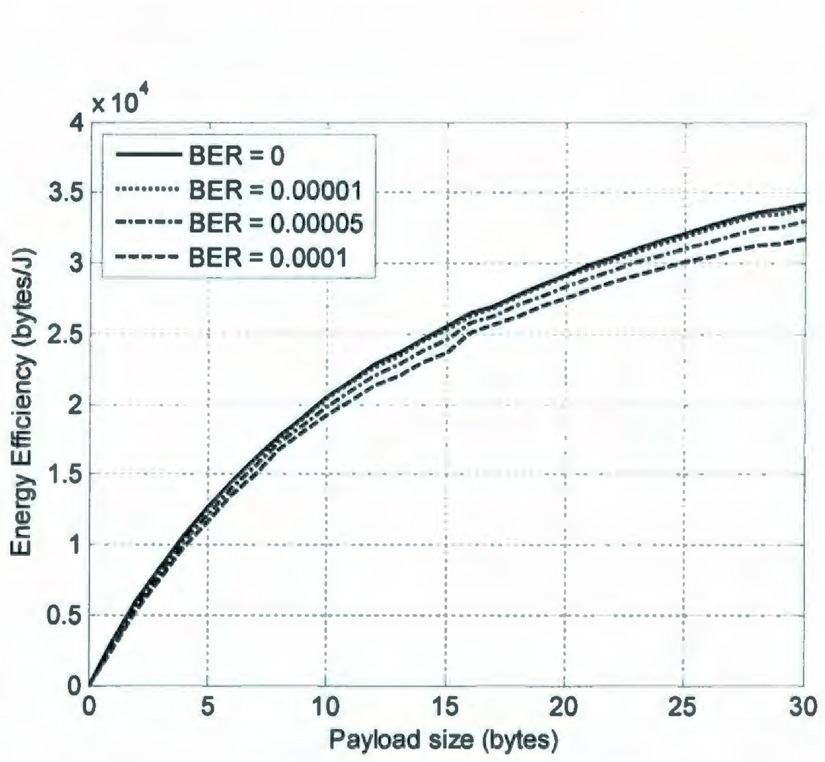


Fig. 6.10. Analysis of CFB scheme.

6.4.3 Other Considerations

1) Effect of Block Sizes

Consider the *implicit IV* scheme, which includes the IV in each packet. When CBC mode (without ciphertext stealing) is used on a block cipher, the block size of the cipher influences the energy efficiency. Since the block cipher processes data block by block, zeroes will be added to the end of the data if the payload size is not an integer multiple of the block size. For CBC mode, a cipher of larger block size tends to cause more energy inefficiency since extra communication energy cost is consumed to transmit the added dummy bits. Take a 7-byte payload for example: 72 bits of zeroes need to be added for a 128-bit block cipher while only 8 bits are needed for a 64-bit block cipher. When the payload size equals a multiple of the block size there is no extra communication energy cost introduced. When used in CBC mode, compared to cipher AES with a 128-bit block size, a 64-bit cipher such as BSPN causes fewer extra bits to be transmitted resulting in higher energy efficiency across the different payload sizes. For modes, such as counter mode and CFB mode, which do not result in extra bits of ciphertext to be transmitted to fill out blocks, the effect of different block sizes is very small, only being dependent on the efficiency of the block cipher.

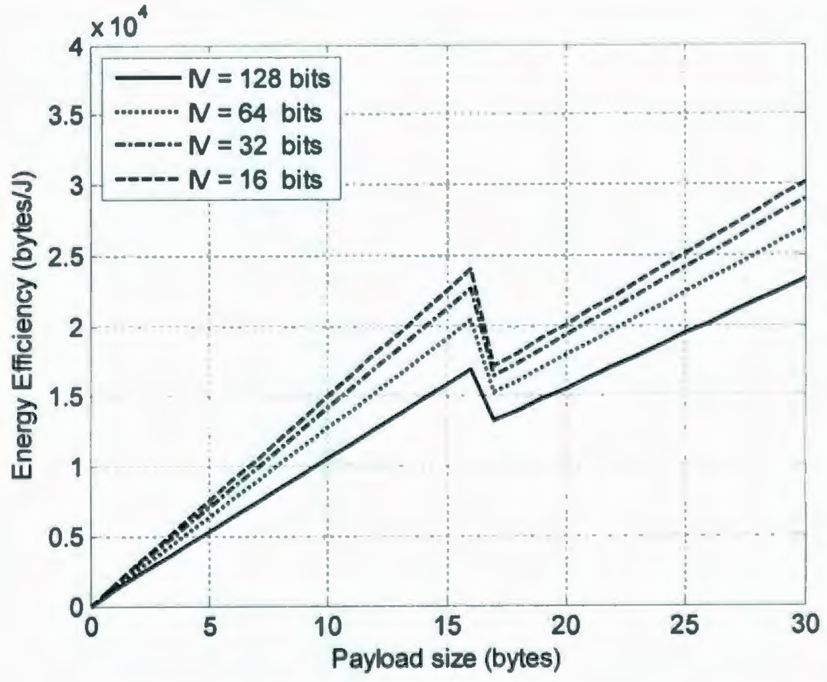


Fig. 6.11. Analysis for different IV size conditions (BER=10^{-4}).

2) Effect of IV Sizes

The size of IV directly impacts the energy cost when IV is included in the transmitted packet. As is exploited by the TinySec scheme, reducing the IV size can lead to a better energy performance. Fig. 6.11 shows the effects of different IV sizes for the condition of

BER equal 10^{-4} using the *implicit IV* scheme with CBC mode for AES. It can be seen that the energy efficiency increases when the IV size decreases. However, for security purposes, the IV size should be large enough to ensure that the IV is not repeated.

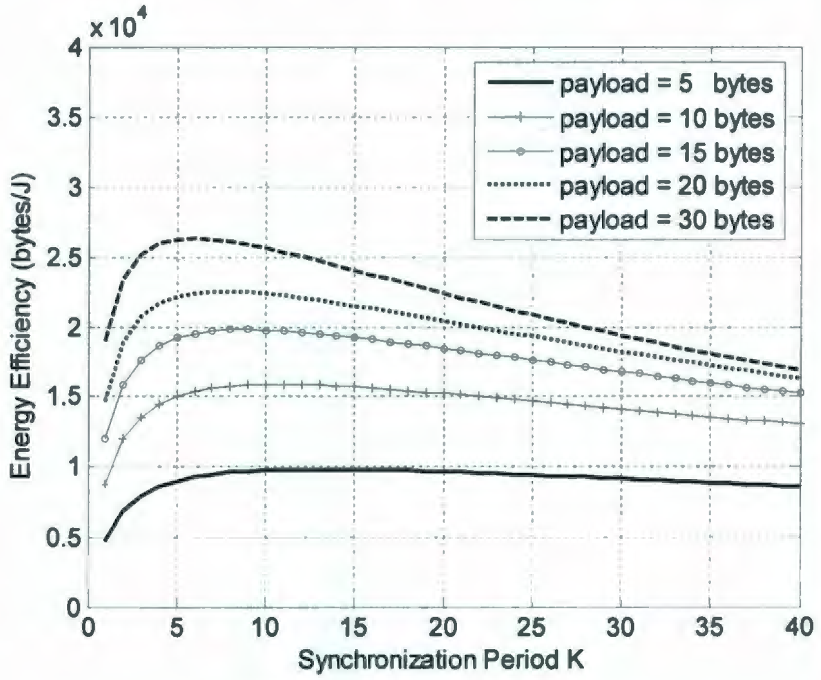


Fig. 6.12. Effects of period and payload size (AES, BER = 10^{-4}).

3) Effect of IV Period

In the schemes that periodically transmit an IV packet to establish synchronization, the value of *K*, together with the size of the payload, impacts energy efficiency. One packet transmitted with an error will lead to the packet being discarded and, hence, the following packets cannot be decrypted correctly until the next IV packet comes. This encourages the value of *K* to be small. On the other hand, smaller *K* causes significant communication energy cost to transfer the IV packet, which will also decrease the energy performance. It is desirable to find out that optimal value of K to balance these two factors.

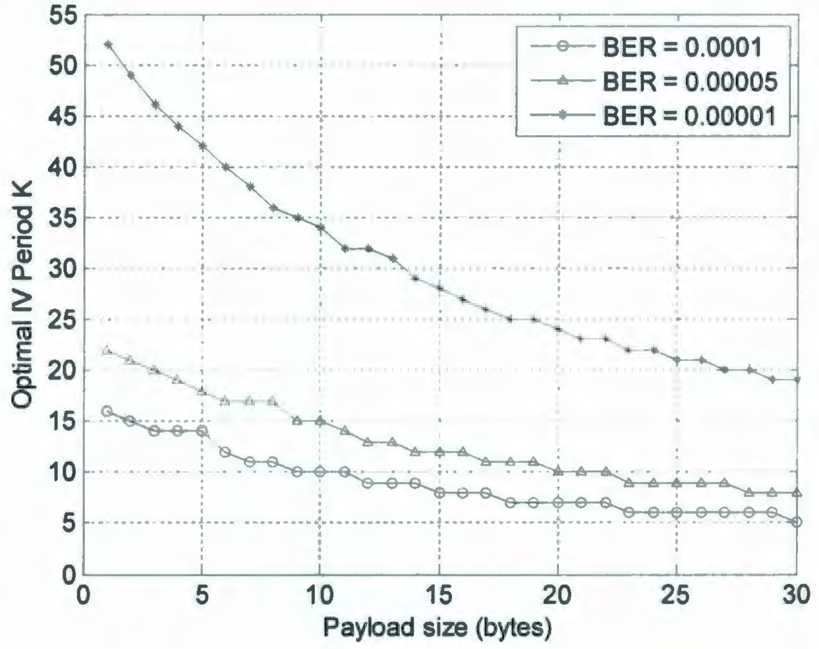


Fig. 6.13. Optimal K for different channel qualities.

The effect of period K and the payload size on the energy efficiency is shown in Fig. 6.12, which is plotted for the *periodic IV without ACK* scheme. In this figure, the cipher AES is applied and BER is 10^{-4}. The size of IV is 128 bits, which is the same with the block size of AES. It can be seen that, as the payload size increases, the energy efficiency can obtain a maximum value with an optimal selection of period K. Consider the *periodic IV without ACK* scheme as an example to analyze the optimal K values. The energy efficiency can be expressed as

$$\eta = \left(\frac{K}{K\times E_{data}+E_{iv}} \times \frac{P_{iv}\times(P_{data}-P_{data}{}^{K+1})}{K\times(1-P_{data})}\right) \times N_{pld}, \tag{6.11}$$

where E_{data} and E_{iv} represents the energy cost for transmitting one data packet and IV packet, respectively. The optimal value of K for maximizing the value of η can be derived by solving $\mathrm{d}\eta/\mathrm{d}K = 0$ for K. Fig. 6.13 illustrates the optimal period K according to payload sizes under different BER conditions. From this figure, we can see that the optimal period becomes shorter when the channel quality is poor.

6.4.4 Simulation Results for Variable Size Packets

To verify the suitability of the analysis models, we have performed extensive simulations for both fixed size packets and variable size packets. In our work, each scheme has been simulated using several types of probability distributions with different variances. The distributions considered are the binomial distribution, Poisson distribution, geometric distribution, and discrete uniform distribution. For example, in Fig. 6.14, Fig. 6.15, Fig. 6.16, Fig. 6.17 and Fig. 6.18, each scheme is simulated using a binomial distribution with different variances (1, 3, and 10). We can see that they are very similar to the results of the analysis model, which is based on fixed size payloads. Simulation results for other probability distributions also show that the resulting curves are very similar to the curves determined by the analysis model. This is illustrated in Fig. 6.19, where variable size packets with different distributions are used to simulate the CFB scheme and the results are presented along with fixed size analysis results. We conclude that the analysis model approximates well the energy cost behavior for sensor nodes for a large variety of packet size distributions.

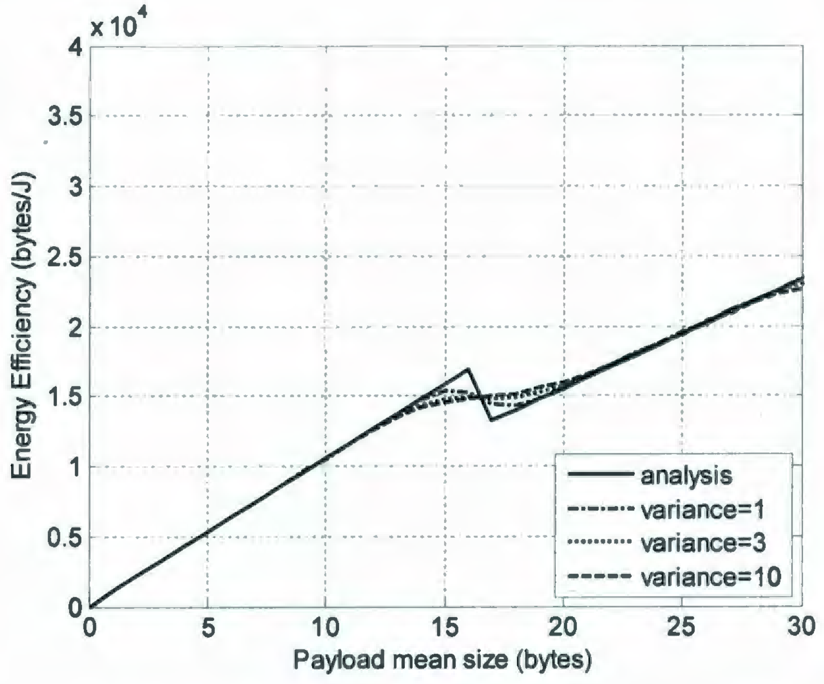


Fig. 6.14. Analysis for implicit IV scheme with binomial distribution.

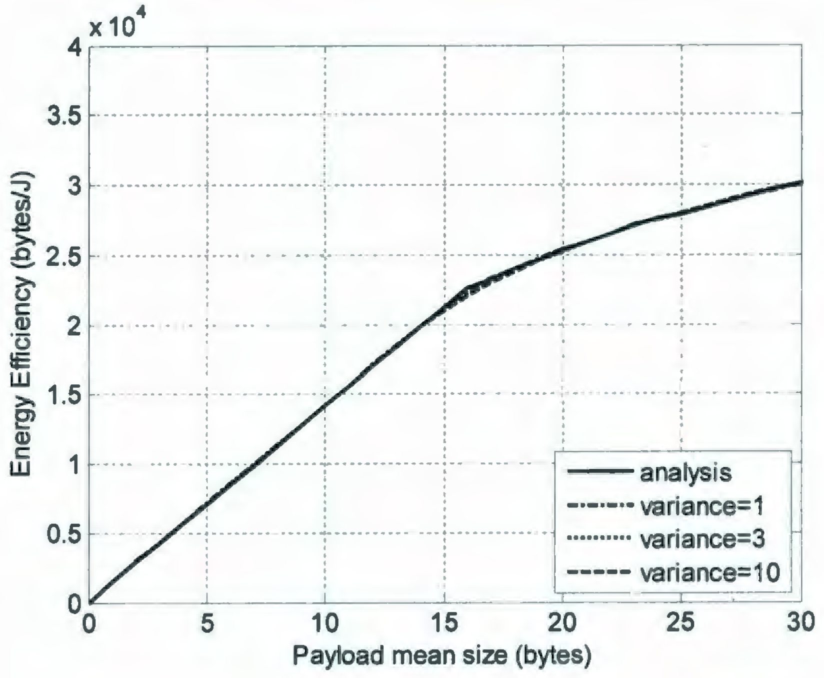


Fig. 6.15. Analysis for TinySec scheme with binomial distribution.

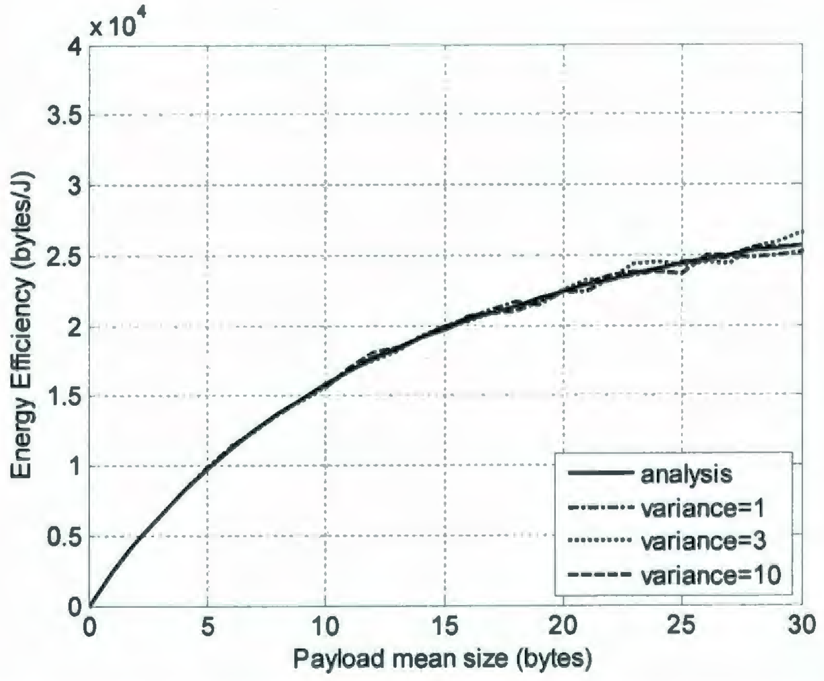


Fig. 6.16. Analysis for Periodic IV without ACK scheme with binomial distribution.

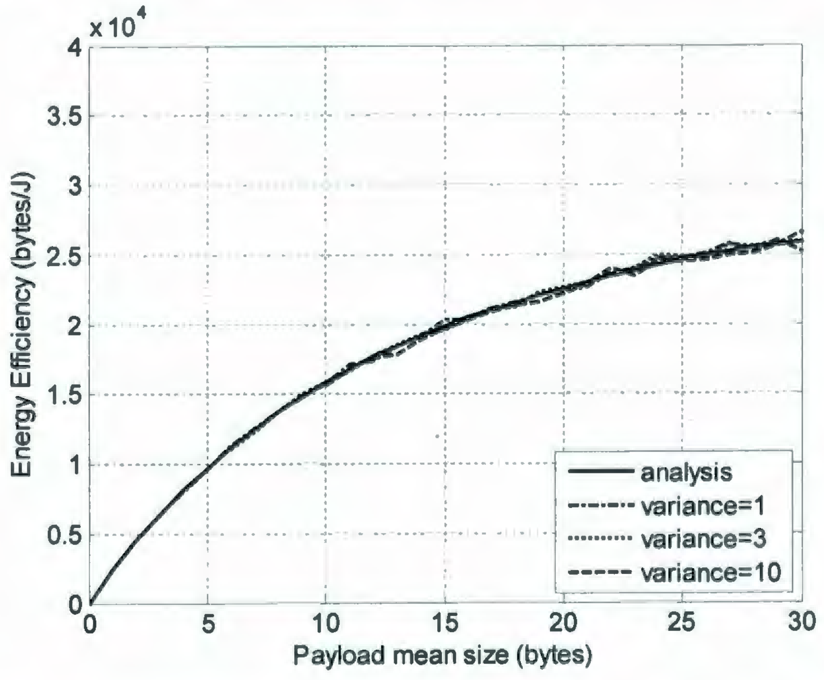


Fig. 6.17. Analysis for Periodic IV with ACK scheme with binomial distribution.

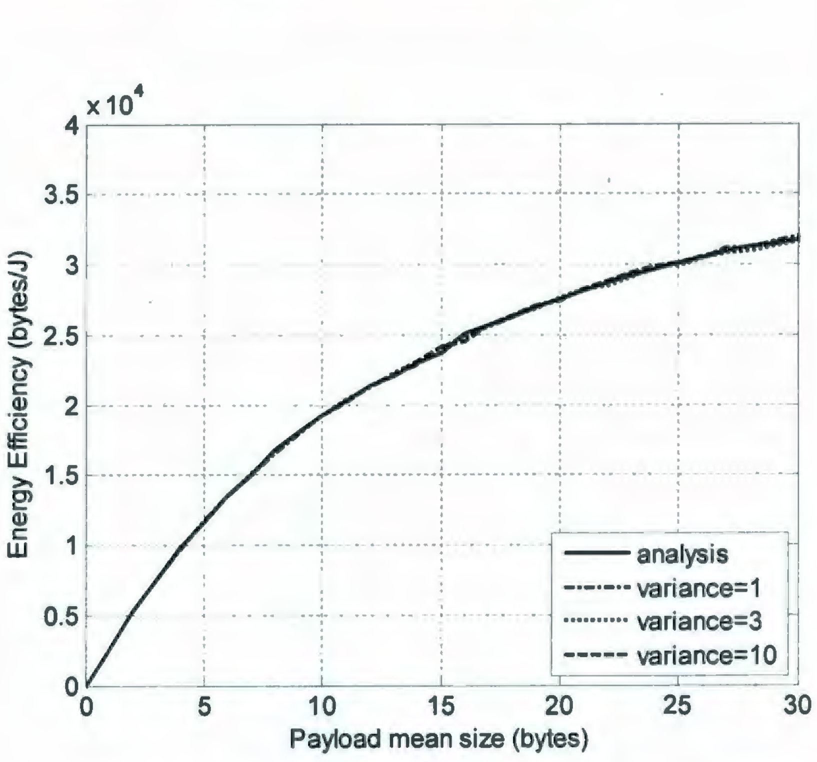


Fig. 6.18. Analysis for CFB scheme with binomial distribution.

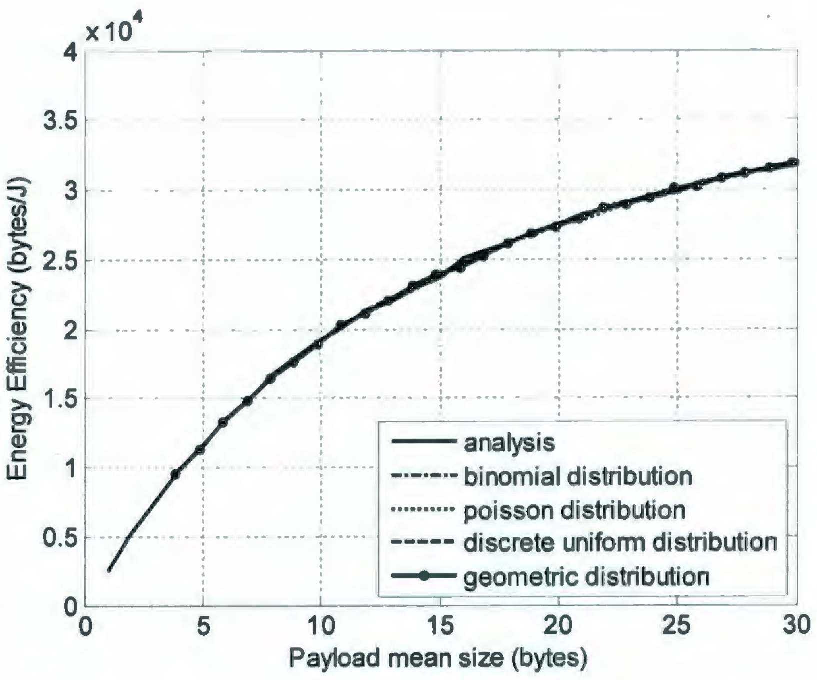


Fig. 6.19. Analysis for CFB scheme with different distributions.

6.5 Comparison of Cryptographic Schemes

In this section, we compare the performance of the five different schemes. We use the analytical result to evaluate each scheme, because we have found through simulations that the analysis is representative of results for many packet distributions. We also calculate the improvement of the CFB scheme comparing to other schemes, which is obtained by

$$\text{improvement} = \frac{\eta_A - \eta_B}{\eta_B} \tag{6.12}$$

where η_A is the energy efficiency of CFB scheme and η_B is the energy efficiency of the scheme we choose for comparison.

6.5.1 Error-free Channels

Fig. 6.20 compares the cryptographic schemes utilizing the cipher AES under the condition of an error-free channel. We can see that schemes of IV transmitted periodically (where we have used $K = 10$) achieve better results than the schemes of IV included in each packet. This is because periodic IV schemes can do the IV distribution without losing counter synchronization when the channel is error-free. In contrast, schemes transmitting the IV with every packet cost more energy than the periodic IV schemes. The CFB scheme also does not transmit IV within the packet and achieves a slightly better result than the periodic IV schemes.

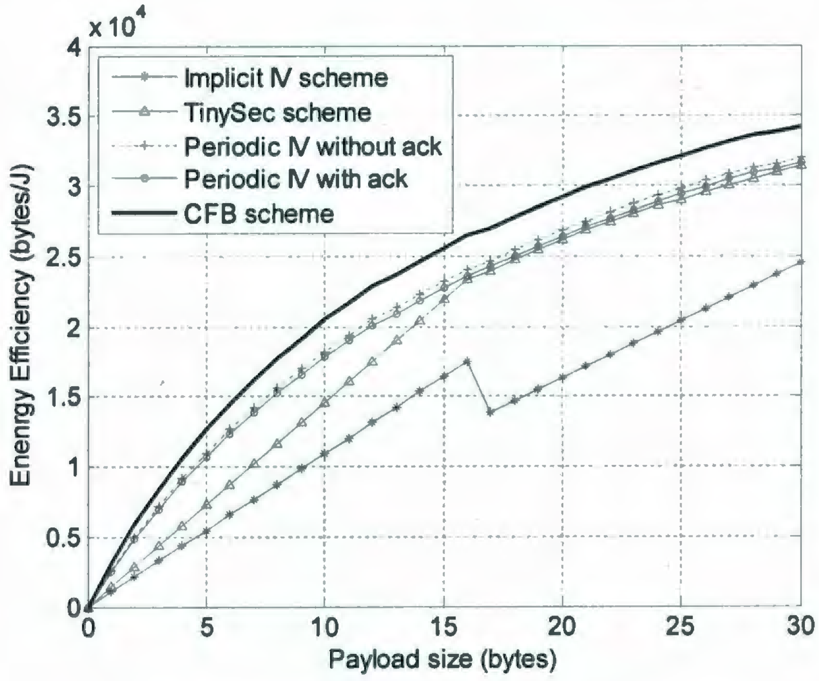


Fig. 6.20. Comparison of schemes with BER = 0 (AES).

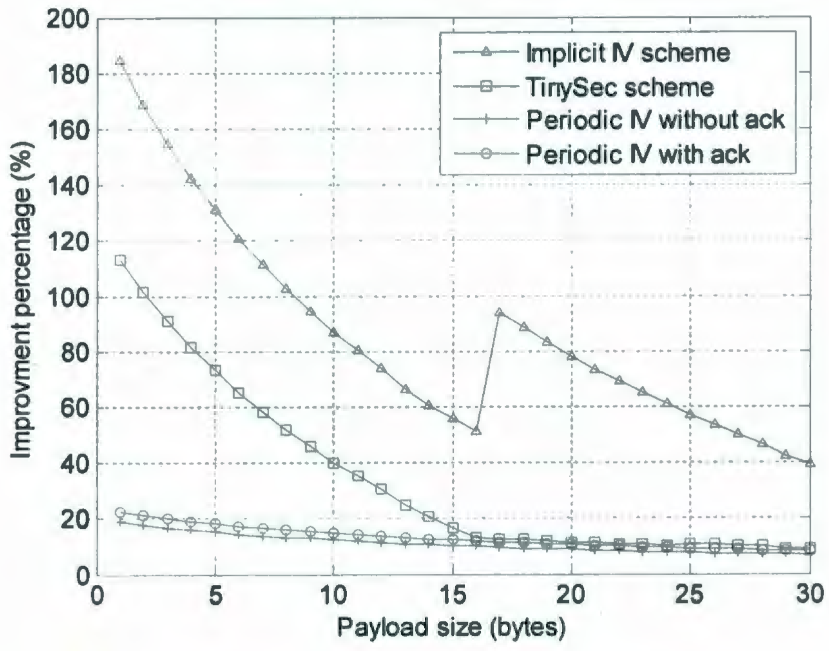


Fig. 6.21. Improvement of CFB scheme (BER = 0).

The performance improvement of the CFB scheme compared to other schemes under the error-free channel is shown as Fig. 6.21. The CFB scheme achieves significant improvement over the other schemes. For example, for small payloads of less than 10 bytes, the CFB scheme has an improvement of 40% or more over the TinySec scheme. In general, the exact value of improvement depends on the packet size with improvement being most notable for smaller payloads.

6.5.2 Noisy Channels

The results change significantly in a channel with noise, as shown in Fig. 6.22, which is based on BER of 10^{-4}. The energy efficiency of periodical IV schemes (where we have let K=10) dramatically decreases, because the successful decryption of the data packet depends on both the IV packet and previous data packets to be correctly received. In contrast, the TinySec scheme, which includes the IV information in every packet, achieves much better results. This can be explained by noting that, in the noisy channel, including an IV in each packet results in a larger probability to decrypt the packet correctly, because it only depends on the packet received to have no errors. The *implicit IV* scheme achieves the worst performance because the full size IV in each packet introduces too much communication energy cost. The CFB scheme still achieves the best performance result among all these schemes, as it reduces the extra communication energy cost of an IV in each packet with the small expense of introducing a modest decrease in the probability of a successfully decrypted packet.

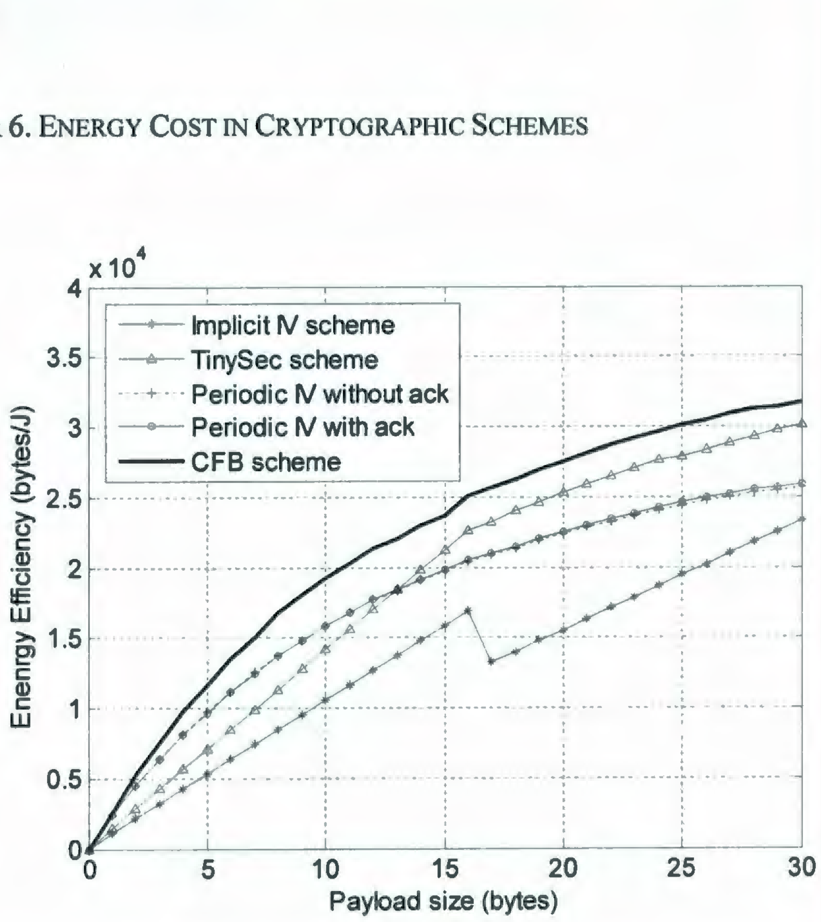


Fig. 6.22. Comparison of schemes with BER = 10^{-4} (AES).

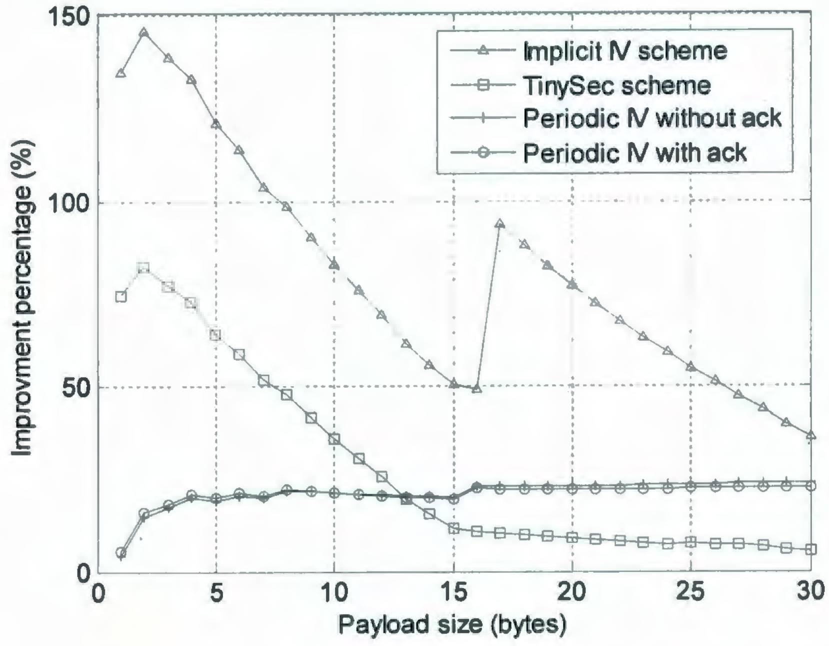


Fig. 6.23. Improvement of CFB scheme (BER = 10^{-4}).

Fig. 6.23 shows the performance improvement under the noisy channel with BER equal to 10^{-4}. The comparison result illustrates that the CFB scheme achieves significantly better performance than all other schemes even when the channel is noisy. Typically, improvements are most significant for smaller size payloads.

6.5.3 Considering Cryptographic Algorithm

To consider the effect of the cryptographic algorithm on the overall energy efficiency, we have applied different ciphers to the CFB scheme and *periodic IV without ACK* scheme with K=10. We also plot a curve for the case of no computational energy cost of MAC included in the scheme, which represents a lower bound for the energy cost of advanced mode of operation. The results are shown in Fig. 6.24. We can see that both the cryptographic algorithm and cryptographic scheme affect the final result. When the energy cost of the algorithm is relatively small, it will not affect the performance as much as the scheme. For example, there is a small difference between AES and BSPN when using the same cryptographic scheme. In the figure, we can also see that the effect of computational energy cost for MAC is also small, with slight difference shown for the CFB scheme using AES whether the computational energy cost of MAC is included or not.

As a broadly accepted secure cipher, AES is a good choice for WSNs. However, our analysis is based on a model with one-way nature of the communication: the sensor node exclusively encrypts and transmits data packets. It is worth noting that the decryption of AES is much less efficient than its encryption. Our implementation result shows that, compared to the 3266 cycles needed to encrypt a plaintext block, 49864 cycles are needed to decrypt a ciphertext block. In some scenarios, a sensor node may need both the functionality of encryption and decryption and the large number of cycles for decryption may dramatically decrease the energy performance of the sensor node. For example, if data is transmitted to a sensor node encrypted using AES in CBC mode, then the sensor node must use the AES decryption operation to decrypt the data. If communication in a WSN is bidirectional resulting in a significant amount of decryption, AES should either not be used or used in a mode such as counter or CFB that only requires the encryption functionality.

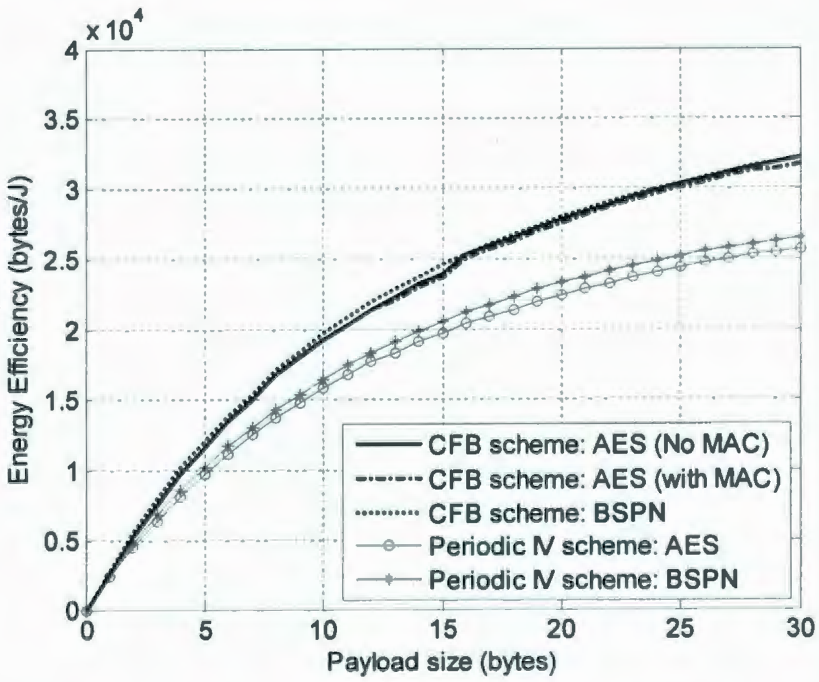


Fig. 6.24. CFB scheme with different ciphers (BER = 10^{-4}).

6.6 Summary

In this chapter, we have investigated the performance of secure data transmission at the link layer in wireless sensor networks, considering specifically the cryptographic schemes. The energy efficiency will be directly affected by many factors, such as the algorithm utilized, the mode of operation, the IV distribution, the packet size and the bit error rate of the channel. We have proposed using the amount of valid data transferred per Joule from the sensor node as the metric to evaluate the energy efficiency when cryptographic schemes are applied to communication between sensor nodes. We have compared the energy efficiency among several typical cryptographic schemes by developing an analysis model. Our results suggest that a cryptographic scheme using a symmetric key block cipher, such as AES, in cipher feedback mode achieves better performance for a wide range of channel qualities and provides significant improvement in energy efficiency compared to other schemes for small payload sizes.

Chapter 7

ENERGY COST IN KEY ESTABLISHMENT

This chapter presents the energy cost analysis for session key establishment in a WSN. The session key establishment is a relatively independent process compared to the secure data transmission, which is investigated in previous chapters. A session key is used to encrypt the data between two nodes, and it should be established before the starting of their secure data communication. In a WSN, the session key establishment is an important process in constructing a secure data communication. A successful session key establishment is a prerequisite for the subsequent secure data communication; otherwise, the encrypted data can be easily eavesdropped and altered if the key is leaked out.

In this chapter, we examine the energy cost of session key establishment in a WSN using the model developed from the previous chapter. We explore the effects of a key establishment to the overall energy cost in a sensor node, especially considering the frequency of key establishment and the quality of communication channel.

7.1 Characteristics of Session Key

The cryptographic algorithm determines the nature of a key. For a cryptosystem using a symmetric key cipher, both communicating parties need the same cipher key for encrypting and decrypting data; while for a cryptosystem using an asymmetric key cipher, such as RSA, a pair of keys, the public key and the private key, are used. Since, in WSNs, an asymmetric key cipher consumes too much energy, symmetric key ciphers are typically being used for link layer encryption. The objective of session key establishment is to make both communicating sides agree with a cipher key, and the key should be protected from being accessed by others.

In a WSN, a unique cipher key, called the initial key, is typically embedded in the sensor node device before deployment. This cipher key is unique and supposed to be only known by the base station. After nodes are deployed, a shared key is established through a secure method when two nodes want to communicate each other securely. The secure method is typically performed between the base station and the pair of the nodes utilizing the unique initial key.

It is obvious that the data communication becomes more secure if a sensor node frequently updates its session key shared with other nodes. The more frequently a sensor node changes its key, the less ciphertext can be collected by an attacker for the cryptanalysis. However, the sensor node is an energy-limited dev ice and the energy consumed in a session key establishment cannot be ignored.

7.2 Session Key Establishment

In this chapter, we consider a typical scenario of the session key establishment in a WSN. A cluster based topology is often applied in WSNs to achieve the energy efficiency, with certain nodes (usually called cluster head or aggregator) managing the sensor nodes within one hop. By doing this, the sensor nodes only deal with relatively simple task such that the energy cost is saved. It is illustrated in Fig. 7.1 that a typical sensor network includes three types of nodes: common sensor nodes, aggregators and a base station. In this figure, the common sensor nodes sense data (such as temperature, smoke, humidity, etc.) and send it to an aggregator. The aggregator is regarded as the cluster head and is assumed to have much more energy supply. After aggregating, data is sent to the base station, which has a continuous energy supply. The base station is a trust centre and in charge of generating and assigning all the session keys. Each aggregator and sensor node has a unique ID and unique initial key, which is only known by the base station.

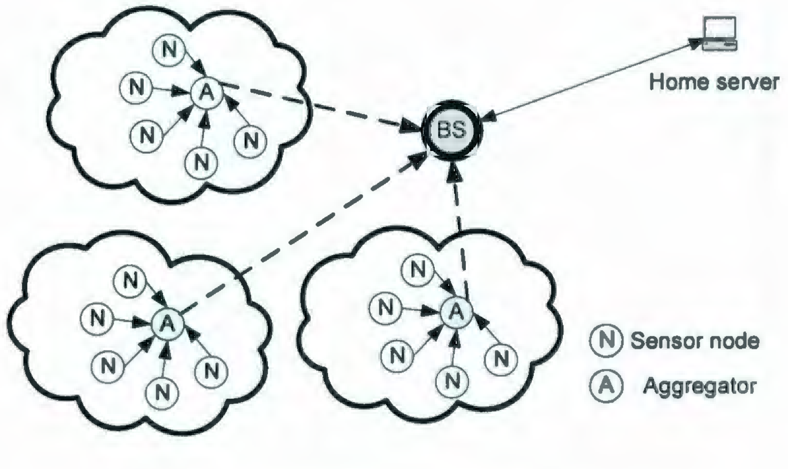


Fig. 7.1. A typical scenario for WSNs.

Table 7.1. Description of the symbol in the packet format.

Symbol	Size (bits)	Description
ID_A	8	The ID of the aggregator
ID_N	8	The ID of the node
K_{sess}	k_size	The session key, determined by the block cipher
N_1, N_2, N_3	b (block size)	The nonce

We consider Needham-Schroeder [8], a typical session key establishment protocol, between a sensor node (N) and an aggregator (A). The Needham-Schroeder protocol is based exclusively on the use of symmetric key ciphers. As shown in Fig. 7.2, there are six steps included in the key establishment. In this figure, we use $E_K(X)$ to represent encryption of the content X by key K, and the meaning of other symbols is listed in Table 7.1. The specifics of session key establishment are explained as following:

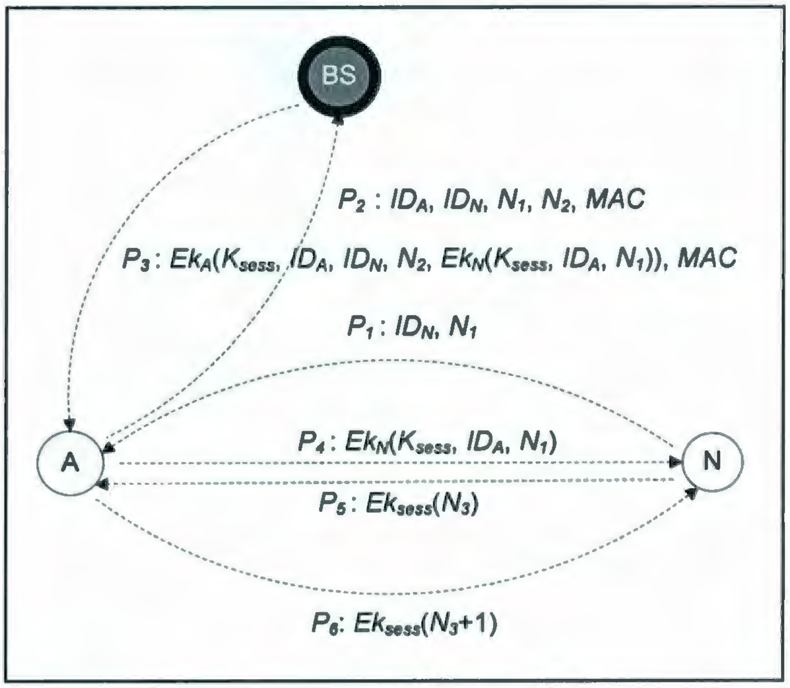

BS
P_2 : ID_A, ID_N, N_1, N_2, MAC
P_3 : $Ek_A(K_{sess}$, ID_A, ID_N, N_2, $Ek_N(K_{sess}$, ID_A, N_1)), MAC
P_1 : ID_N, N_1
P_4 : $Ek_N(K_{sess}$, ID_A, N_1)
A
N
P_5 : $Ek_{sess}(N_3)$
P_6: $Ek_{sess}(N_3+1)$

Fig. 7.2. Session key establishment process.

Step 1: The sensor node sends out packet P_1, including the node's ID_N and a random nonce N_1, to the aggregator asking for establishment of a session key.

Step 2: When the aggregator receives packet P_1, it sends a packet P_2 to the base station asking for a session key, which is to be only known by the aggregator and the sensor node. This packet includes (1) IDs of the aggregator and the sensor node (ID_A and ID_N), (2) nonces which are received from the sensor node (N_1) and generated by the aggregator (N_2), and (3) a MAC value generated by CBC mode using the initial key K_A of the aggregator. Since K_A is only known by the base station and the aggregator, the MAC value is used to ensure the data integrity and authentication.

Step 3: When the base station receives P_2 and verifies the MAC, it assigns a session key K_{sess} for the aggregator with ID_A and the sensor node with ID_N. The session key is transmitted within packet P_3, with other information of both the applicants. It is worth noting that the session key appears twice in this packet, one is for the aggregator, and the other is for the sensor node, which is encrypted by K_N, the initial key of sensor node. The whole packet is encrypted using the initial key K_A, which is also used for generating the MAC. Only the aggregator with ID_A can decrypt this packet and read the session key K_{sess}. However, the aggregator cannot decrypt the information for the sensor node, since K_N is only known by the base station and the sensor node.

Step 4: After decrypting the packet P_3, the aggregator obtains the details of this packet: (1) the ID indicates which sensor node shares the key with the aggregator, (2) the nonce ensures the information is not replayed, and (3) the information of the session key for the sensor node is sent to the sensor node within P_4. The sensor node decrypts the packet P_4

using key K_N. The nonce N_1 is very important information in this packet, since it prevents the replaying attack and ensures the integrity of this packet at the same time. The sensor node will compare the decrypted value for N_1 and the value it generated originally. If these two values are the same, K_{sess} will be accepted as the session key shared with the aggregator.

Step 5: The sensor node generates another nonce N_3, and encrypts it using the accepted session key K_{sess}. This information is transmitted to the aggregator within the packet P_5.

Step 6: The aggregator receives P_5, and obtains the nonce N_3 by decrypting it using their session key K_{sess}. Then the aggregator encrypts the data with value N_3+1 using the session key K_{sess} and sends it to the sensor node within the packet P_6.

The whole process protects the session key from being revealed by others. The first three steps focus on generating the session key and the following steps focus on exchanging and authenticating the key. During these steps, a session key can be safely established between the aggregator and the sensor node assuming the base station can be trusted. Meanwhile, the key cannot be known or even altered by others since the session key is well protected by the initial keys, which are only known by the base station. It can also avoid the replaying attack due to the utilization of the randomly generated nonce.

7.3 Simulation of the Session Key Establishment

We perform simulation to evaluate the energy cost of session key establishment in a WSN using the model developed in Chapter 6.

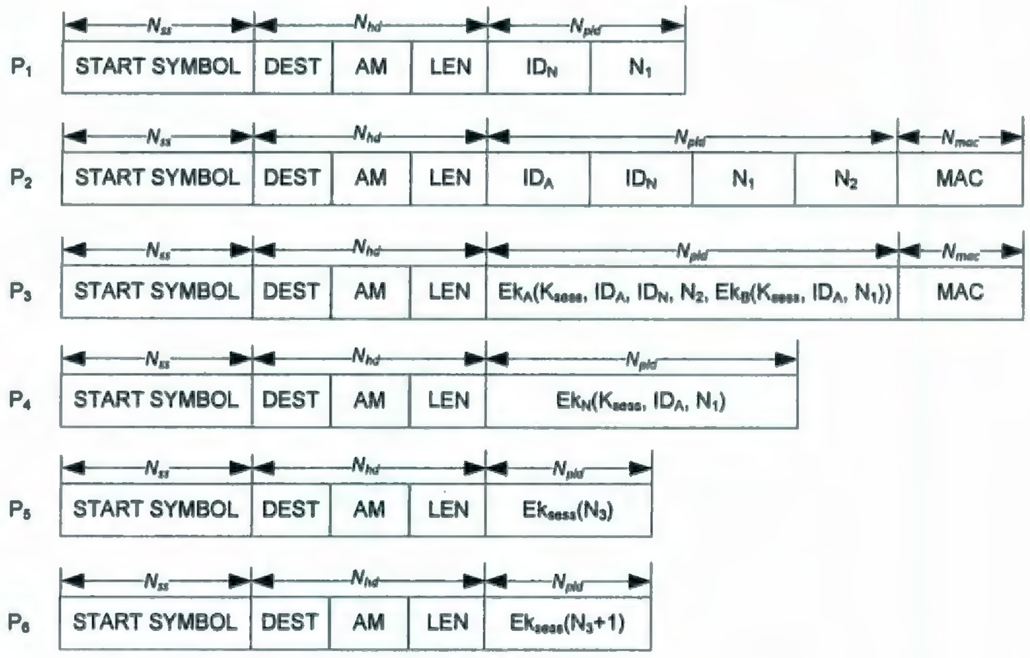

N_{ss}
N_{hd}
N_{pld}
N_{mac}
P1 START SYMBOL DEST AM LEN ID_N N_1
P2 START SYMBOL DEST AM LEN ID_A ID_N N_1 N_2 MAC
P3 START SYMBOL DEST AM LEN $Ek_A(K_{sess}, ID_A, ID_N, N_2, Ek_B(K_{sess}, ID_A, N_1))$ MAC
P4 START SYMBOL DEST AM LEN $Ek_N(K_{sess}, ID_A, N_1)$
P5 START SYMBOL DEST AM LEN $Ek_{sess}(N_3)$
P6 START SYMBOL DEST AM LEN $Ek_{sess}(N_3+1)$

Fig. 7.3. Packet formats for the session key establishment.

Table 7.2. Payload size of different packet.

Packet ID	Payload size
P_1	$N_{pld} = 8 + b$
P_2	$N_{pld} = 8 + 8 + b + b = 16 + 2b$
P_3	$N_{pld} = \lceil (b + 8 + 8 + k\_size + \lceil (b + 8 + k\_size)/b \rceil \times b)/b \rceil \times b$
P_4	$N_{pld} = \lceil (b + 8 + k\_size)/b \rceil \times b$
P_5	$N_{pld} = b$
P_6	$N_{pld} = b$

7.3.1 Packet Format

The packet formats used in the simulation are shown in Fig. 7.3. There are seven types of packets in the session key establishment as explained in the previous section. The payload size of each packet is calculated and the result is shown in Table 7.2. However, our research mainly focuses on evaluating the energy cost of a sensor node in the session key establishment. Hence, only packets transmitted or received by a sensor node are considered, which include P_1, P_4, P_5 and P_6.

7.3.2 Simulation Description

During the session key establishment period, both encryption and decryption are involved. Unlike the secure data transmission, key scheduling is processed during the session key establishment. Our research focuses on the traditional sensor node and the cryptographic algorithms are implemented for software. Two block ciphers are chosen as candidates: (1) the widely used block cipher AES and (2) the involutional cipher BSPN (with 64-bit and 128-bit key size). The number of CPU operation cycles for these block ciphers is listed in Table 7.3, including the key scheduling, encryption and decryption. Notice that for cipher BSPN, the number of operation cycles for the encryption and decryption is the same due to their involutional characteristics. The simulation is performed by Matlab. In this simulation, the random packets are generated with independent probability of bit errors and the packet will be discarded when errors being detected. As a result, the key establishment is failed and the session key needs to be reestablished until both communicating sides know the key. The energy cost is counted including both the communication cost and computational cost. Parameters used in the simulation are the same as in previous chapters.

Table 7.3. Operation cycles for block ciphers.

Cipher	Key scheduling	Encryption	Decryption
AES	841	3266	49864
BSPN_64	6256	796	796
BSPN_128	9644	796	796

7.4 Simulation Results

Now we focus on studying two effects of the session key establishment: different cryptographic algorithms used in a sensor node and various frequency of the session key updates. Both these effects are studied with the consideration of the channel quality.

Table 7.4 Key distribution energy cost (BER = 0).

# Key distribution events	AES	BSPN-64	BSPN-128
01	0.0222	0.0129	0.0137
02	0.0423	0.0247	0.0261
03	0.0625	0.0364	0.0385
04	0.0826	0.0482	0.0509
05	0.1028	0.0599	0.0633
06	0.1230	0.0716	0.0757
07	0.1431	0.0834	0.0881
08	0.1633	0.0951	0.1005
09	0.1834	0.1069	0.1129
10	0.2036	0.1186	0.1254

7.4.1 Effects of Cryptographic Algorithm

We compare the energy cost of session key establishment for different block ciphers assuming error free channel. Table 7.4 shows the energy cost in an error-free channel. In this table, we can see that for the same number of key establishment times, the energy cost using AES is much higher than the other two ciphers. The reason is that the decryption of cipher AES needs a large number of CPU cycles, and hence, consumes much more energy than other ciphers. BSPN-128 consumes more energy cost than BSPN-64 mainly due to the 128-bit key size, requiring a larger communication energy cost (i.e., *k_size* is larger in the packets of Table 7.2).

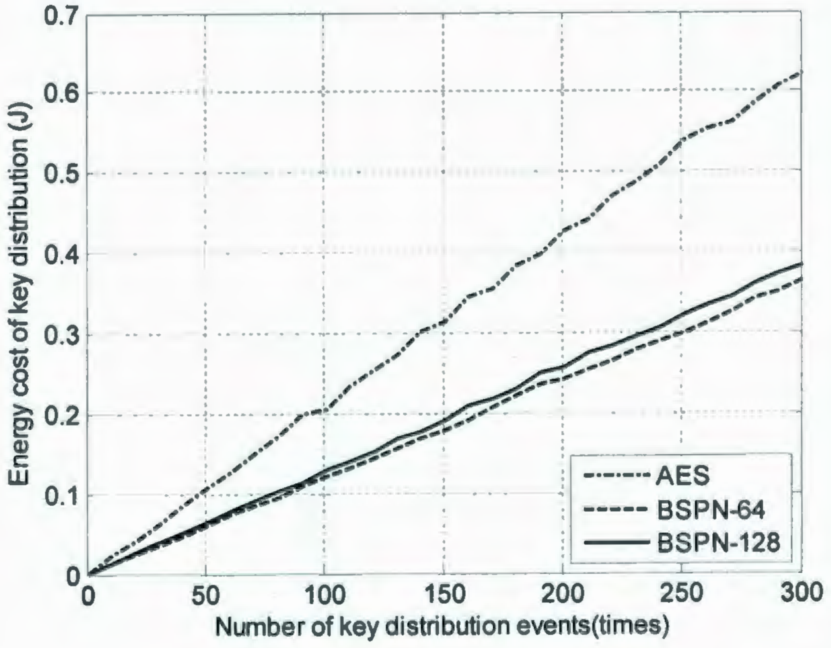


Fig. 7.4. Key distribution energy cost (BER = 10^{-4}).

Simulation results under the condition of a noisy channel with BER of 10^{-4} are shown in Fig. 7.4. Compared with the error-free channel, more energy is consumed since the transmission errors may lead to retransmission of the packet during the session key establishment. In this figure, we can see that the trend of energy cost by different ciphers is the same as that of error-free channel. Still, AES and BSPN-64 are the least and most energy efficient ciphers for the session key establishment, respectively.

7.4.2 Effects of the Key Update Frequency

There is a tradeoff between the session key establishment and the secure data communication. On one hand, updating the key frequently can make the data communication more secure since the attacker is constrained by the amount of ciphertext collected under one key; on the other hand, session key establishment consumes energy, which is a serious concern for a WSN. We now investigate the effect of the key update frequency to the energy efficiency of a sensor node's secure communication.

Simulation results are shown in Fig. 7.5 and Fig. 7.6, with different key update frequencies in an error-free channel and noisy channel, respectively. We let T represent the lifetime of a session key (i.e., the number of data packets sent between key establishment processes). For example, "T = 1" represents that a session key establishment happens for every data packet transmitted and "T = unlimited" means there is only one session key establishment at the start of a node's life span. In this simulation, we use the block cipher AES and the CFB scheme, which is discussed in previous chapters. From the figure, we can see that (1) both the error-free channel and noisy channel have similar results with energy efficiency decreasing when the channel quality becomes worse and (2) the energy efficiency increases when the payload size increases. We can also see that the energy efficiency is greatly affected when the session key is changed frequently. However, with the lifetime of a session key increased, the effect becomes less. Notice that the case "T=1" is an extreme case for session key establishment, since every packet is encrypted with a different key. However, it is not practical since it sacrifices too much energy, which can be otherwise used to transmit data.

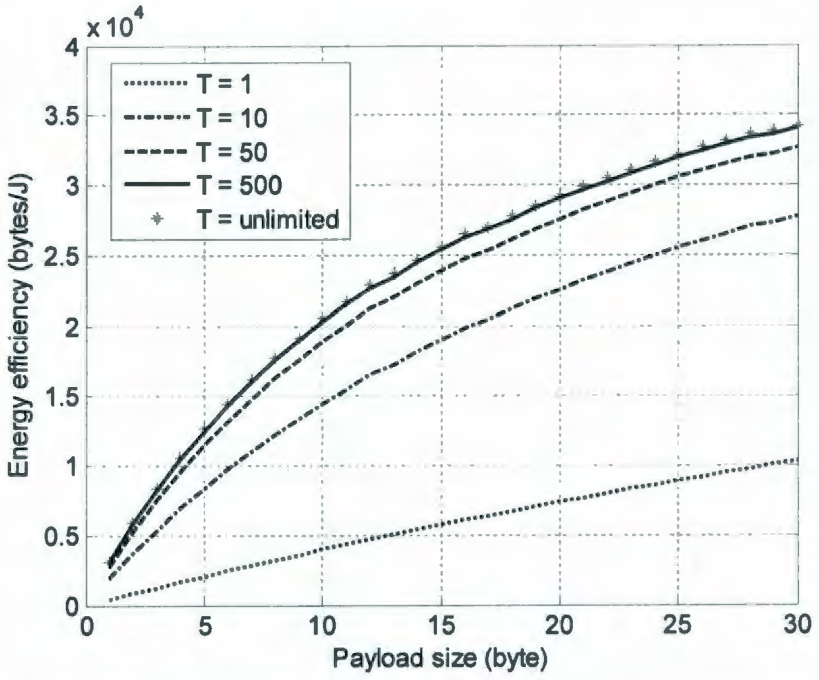


Fig. 7.5. Effect of key agreement frequency in CFB scheme (BER = 0).

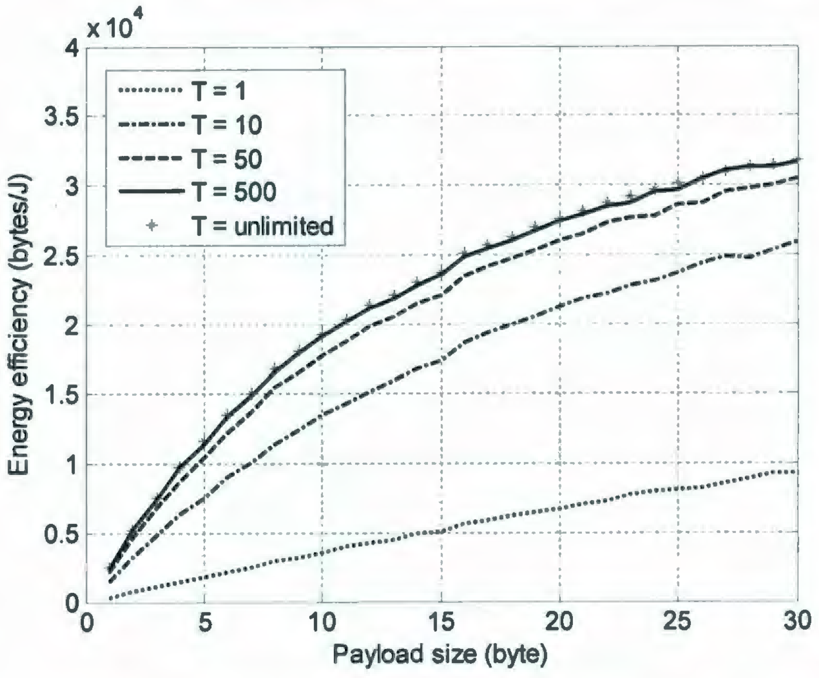


Fig. 7.6. Effect of key agreement frequency in CFB scheme (BER = 10^{-4}).

Fig. 7.7 shows the detailed frequency effects. We can see that, when the frequency is low enough, about more than 100 packets per session key, the energy efficiency is relatively constant. For T<20, energy efficiency is greatly reduced. Fig. 7.8 shows the data communication with session key establishment using different block ciphers. This simulation is under the condition of BER=10^{-4} and assumes the life of the session key is 10 packets. From the figure, we can see that for frequent session key distributions, the choice of block ciphers will affect the energy efficiency. In this figure, the energy efficiency for using AES is much lower than using BSPN, which is because AES needs much more CPU cycles to perform the decryption process. The BSPN-64 achieves better energy efficiency than BSPN-128 mainly due to the size the key, i.e., much communication energy is consumed to transmit the session key.

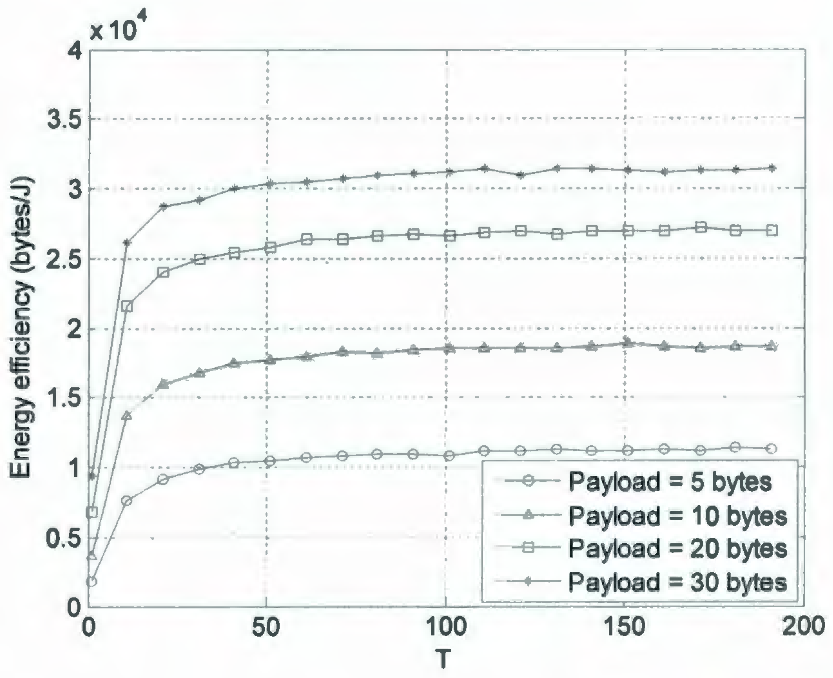


Fig. 7.7. Frequency effects (BER = 10^{-4}, AES).

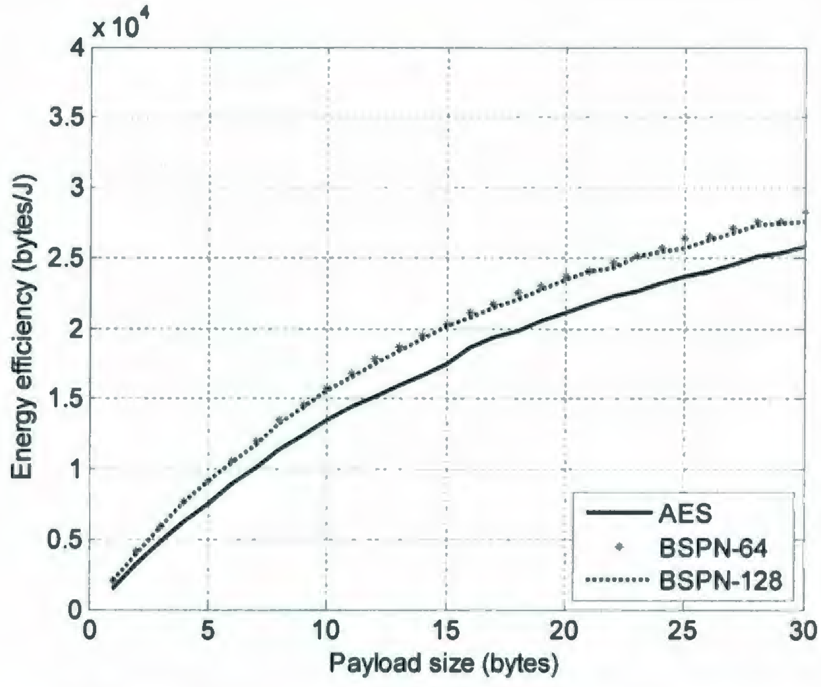


Fig. 7.8. Data communication with key establishment for different ciphers (T = 10).

7.5 Summary

In this chapter, we have focused on investigating the session key establishment in wireless sensor networks. Session key establishment plays an important role in the secure data communication, since both communicating parties need to agree on a key before encrypting a message. In a WSN, the session key has to be established between two nodes through an open broadcast media. We have investigated a typical session key establishment scenario in a cluster based WSN. We focus on the energy cost of the key establishment including both the computational cost and communication cost. In particular, we explore the effects brought by the channel quality and the key establishment frequency. Simulations are performed using the model we developed in the previous chapter and simulation results show that the energy efficiency of a sensor node will not be greatly affected unless the session key is updated very frequently. As a tradeoff between the security and performance, we recommend extending the life time of a session key long enough so that it will only slightly impact the total number of valid data bits transmitted for a sensor node.

Chapter 8

CONCLUSIONS AND FUTURE WORK

8.1 Conclusions

This thesis investigates the energy performance of link layer secure communication in wireless sensor networks (WSNs), considering both the cryptographic algorithms and cryptographic schemes. Energy limitation is one of the largest problems in WSNs. Therefore, it is crucial to choose appropriate cryptographic algorithms and schemes for a sensor node, so that the suitable security can be achieved and energy consumption is kept to a minimum. In a sensor node, the energy cost introduced by cryptography includes computational cost and communication cost. The former is mainly affected by cryptographic algorithms, while the latter is affected by cryptographic schemes being used, that is, the method of using an algorithm to achieve the security goal. To extend the life span of a sensor node as long as possible, both types of energy cost need to be minimized.

There are many types of sensor nodes produced for different purposes. However, from the implementation point of view, they can be categorized into software-oriented and hardware-oriented. A software-oriented sensor node is the classical type, which uses a microprocessor to execute all kinds of tasks. In contrast, a hardware-oriented sensor node is a new type, which combines reconfigurable hardware (typically FPGA) and a microprocessor together to provide higher performance. To evaluate the energy cost of cryptographic algorithms in sensor nodes, we have implemented several cryptographic algorithms for both these two implementation environments.

For software implemented cryptographic algorithms, we have chosen different types of symmetric ciphers to evaluate, including both block ciphers and stream ciphers. We estimated the energy consumption of the microprocessor by calculating the number of CPU cycles being used. We especially investigated different factors affecting the energy cost of cryptographic algorithms, including the intrinsic factor of algorithm characteristics and extrinsic factors of channel quality.

For hardware implemented cryptographic algorithms, we especially choose involutional block ciphers as the characteristic of involution allows using the same circuit for both encryption and decryption, which minimize resources required for a sensor node. We have focused on studying the method of energy-efficient design based on an FPGA and further investigated the energy c ost advantage of software-hardware cooperation based on a software implemented key scheduling process and hardware implemented encryption.

To evaluate the energy efficiency when cryptographic schemes are applied to the sensor node communication, we have proposed using energy efficiency defined as the amount of valid data transferred per Joule from the sensor node as the metric since the limited energy supply is an important constraint in WSNs. The energy efficiency for a sensor node can be directly affected by many factors, such as the algorithm utilized, the mode of operation, the packet size, the IV distribution, and the bit error rate of the channel. We explored the energy cost of secure data transmission in WSNs by developing an analysis model to compare the energy efficiency among several typical cryptographic schemes, and showing the model's appropriateness by simulations. In addition, we have investigated the energy cost of session key establishment in a WSN, which is an important part of secure data transmission. We have also studied different effects for key establishment in WSNs, such as the cryptographic algorithm used and the key update frequency.

We have investigated the performance of secure data transmission at the link layer in wireless sensor networks, considering both the cryptographic algorithms and cryptographic schemes. Results suggest that a cryptographic scheme using a lightweight symmetric key block cipher, BSPN, with a cipher feedback scheme achieves better performance for a range of channel qualities and for small payload sizes provides a large relative improvement in the energy efficiency compared to other schemes. In addition, as simulation results show that key establishment only takes a small proportion of the total energy cost in a sensor node, the security of data communication can be further strengthened by occasionally updating the encryption key based on a symmetric key distribution process.

8.2 Future Work

The future work may include several directions:

- In our research, we have implemented the involutional ciphers BSPN and KHAZAD targeting for FPGA. However, the FPGA is not especially designed as a low power device. To save the energy cost, ASIC can be a better choice when large amount of sensor devices are needed.
- Pipeline is an efficient approach for hardware implementation, which can greatly improve the efficiency and throughput when there is large amount of data needs to be operated. In our research, we did not use pipeline method since we focus on sensor device limited by energy cost. However, it will be a good direction in some special application environments, such as wireless multimedia sensor networks, which require transmitting large amount of multimedia data.
- We have studied the energy efficiency of cryptographic schemes based on the software-oriented sensor node. Further investigation may include the energy efficiency of cryptographic algorithm for hardware-oriented sensor devices.
- Compared to symmetric key cryptography, asymmetric key cryptography has an advantageous characteristic for broadcast authentication in a WSN. Hence, the future work may include analyzing the energy cost asymmetric key cryptography applied to WSNs and designing an appropriate key distribution scheme for WSNs.
- Our research is focused on building the analysis model and performing theoretical evaluation. It is worth building a prototype for secure wireless sensor networks and obtaining the practical value of energy cost by emulation.

References

[1] Y. Wang, G. Attebury, and B. Ramamurthy, "A Survey of Security Issues in Wireless Sensor Networks," *IEEE Communications Surveys & Tutorials*, vol.8, no.2, pp. 2-23, 2006.

[2] R. Roman, C. Alcaraz, and J. Lopez, "A Survey of Cryptographic Primitives and Implementations for Hardware-Constrained Sensor Network Nodes", *Mobile Networks and Applications*, Springer Netherlands, vol.12, pp.231-244, Oct. 2007.

[3] A. Mainwaring, J. Polastre, R. Szewczyk, and D. Culler, "Wireless sensor networks for habitat monitoring," In Proc. of *1st ACM Int'l Workshop Wireless Sensor Networks and Applications (WSNA 2002)*, pp. 88–97. ACM Press, New York, Sept. 2002.

[4] I. F. Akyildiz, T. Melodia, and K. R. Chowdhury, "A survey on wireless multimedia sensor networks," *Computer Networks (Elsevier)*, vol. 51, no. 4, pp. 921–960, Mar. 2007.

[5] I.F. Akyildiz, T. Melodia, and T. Chowdhury, "Wireless Multimedia Sensor Networks: Applications and Testbeds," Proceedings of the IEEE, vol. 96, no. 10, pp. 1588 - 1605, Oct. 2008.

[6] Z. Bin, H. Chao, W. Haibin, G. Ruiwe, and M.Q.-H. Meng, "A Wireless Sensor Network for Pervasive Medical Supervision," In *Proc. of Integration Technology 2007 (ICIT '07)*, pp.740-744, March 2007.

[7] K. Eleni, K. Elisavet, K. Georgios, and G. Stefanos, "Clustering Oriented Architectures in Medical Sensor Environments," In Proc. of *3rd Availability, Reliability and Security (ARES'08)*, pp.929-934, March 2008.

[8] Cryptography and Network Security Principles and Practices, Fourth Edition. *William Stallings*, 2005.

[9] National Bureau of Standards, "Data Encryption Standard", *Federal Information Processing Standard 46*, 1977.

[10] National Institute of Standards and Technology (NIST), "Advanced Encryption Standard (AES)," *Federal Information Processing Standard (FIPS) 197*, Nov. 2001.

[11] J. Daemen, and P. Kitsos, "The self-synchronizing stream cipher Mosquito: eSTREAM documentation," eSTREAM, ECRYPT Stream Cipher Project, 2005, [Online]. *Available: http://www.ecrypt.eu.org/stream/p3ciphers/mosquito/mosquito.pdf.*

[12] H. M. Heys, "Analysis of the Statistical Cipher Feedback Mode of Block Ciphers", *IEEE Transactions on Computers*, vol. 52, pp. 77-92, Jan. 2003.

[13] "Call for Stream Cipher Primitives, Version 1.3, 12th April 2005," eSTREAM, *ECRYPT Stream Cipher Project*, 2005. [Online]. *Available:* http://www.ecrypt.eu.org/stream/call/.

[14] R. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", *Communications of the ACM*, vol. 21, pp. 120-126, 1978.

[15] N. Koblitz, "Elliptic Curve Cryptosystems", *Mathematics of Computation*, vol. 48, pp. 203–209, 1987.

[16] J. F. Kurose and K. W. Ross, *Computer Networking: A Top-Down Approach, Pearson Addison-Wesley*, 4th edition, 2008.

[17] W. Simpson, "RFC 1661: The Point-to-Point Protocol (PPP)," Network Working Group of the IETF, July 1994, [online]. *Available: http://tools.ietf.org/html/rfc1661*.

[18] B. Lloyd, and W. Simpson, "RFC 1334: PPP Authentication Protocols," Network Working Group of the IETF, Oct. 1992, [online]. *Available: http://www.ietf.org/rfc/rfc1334.txt*.

[19] W. Simpson, "RFC 1994: PPP Challenge Handshake Authentication Protocol (CHAP)," Network Working Group of the IETF, Aug. 1996, [online]. *Available: http://www.ietf.org/rfc/rfc1994.txt*.

[20] L. Blunk, and J. Vollbrecht, "PPP Extensible Authentication Protocol (EAP)," Network Working Group of the IETF, Mar. 1998, [online]. *Available: http://www.ietf.org/rfc/rfc2284.txt*.

[21] T. Ylonen, and C. Lonvick, "RFC 4252: The Secure Shell (SSH) Authentication Protocol," Network Working Group of the IETF, Jan. 2006, [online]. *Available: http://www.ietf.org/rfc/rfc4252.txt*.

[22] J. Callas, L. Donnerhacke, H. Finney, D. Shaw, and R. Thayer, "RFC 4880: OpenPGP Message Format," Network Working Group of the IETF, Nov. 2007, [online]. *Available: http://tools.ietf.org/html/rfc4880*.

[23] Y. Kawatsura, "RFC 3538: Secure Electronic Transaction (SET) Supplement for the v1.0 Internet Open Trading Protocol (IOTP)," Network Working Group of the IETF, June 2003, [online]. *Available: http://www.rfc-editor.org/rfc/rfc3538.txt*.

[24] C. Neuman, T. Yu, S. Hartman, and K. Raeburn, "RFC 4120: The Kerberos Network Authentication Service (V5)," Network Working Group of the IETF, July 2005, [online]. *Available: http://www.ietf.org/rfc/rfc4120.txt*.

[25] L. M. S. C. of the IEEE Computer Society, Wireless LAN medium access control (MAC) and physical layer (PHY) specifications, *IEEE Standard 802.11*, ANSI/IEEE Std 802.11, 1999.

[26] J. Edney and W. A. Arbaugh, Real 802.11i Security Wi-Fi Protected Access and 802.11i, *Addison Wesley*, Aug. 2003.

[27] National Institute of Standards and Technology (NIST), "Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality," June 2007, [Online]. *Available: http: //csrc.nist.gov/ publications/nistpubs/800-38C/SP800-38C_updated-July20_2007.pdf.*

[28] *Datasheet Available online: http://focus.ti.com/general/docs/prod.tsp?DCMP= TIHeaderTracking&HQS=Other+OT+hdr_p_products*

[29] IEEE Standard 802, Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (WPANs), 2003.

[30] Y. Kawatsura, "RFC 3538: Secure Electronic Transaction (SET) Supplement for the v1.0 Internet Open Trading Protocol (IOTP)," Network Working Group of the IETF, June 2003, [online]. Available: http://www.rfc-editor.org/rfc/rfc3538.txt.

[31] National Institute of Standards and Technology (NIST), "Guide to Bluetooth Security Recommendations", *[online]. Available:* *http://csrc.nist.gov/publications/nistpubs/800-121/SP800-121.pdf.*

[32] T. Clausen, and P. Jacquet, "RFC 3626: Optimized Link State Routing Protocol (OLSR). Mobile Ad Hoc Networking Working Group of the IETF", Oct. 2003, [online]. Available: *http://www.faqs.org/rfcs/rfc3626.html.*

[33] E. Winjum, A.M. Hegland, P. Spilling, and O. Kure, "A performance evaluation of security schemes proposed for the OLSR protocol," in Proc. of *Military Communications Conference 2005 (MILCOM'05)*, pp. 2307-2313 Vol. 4, Oct. 2005.

[34] B. Kannhavong, H. Nakayama, and A. Jamalipour, "SA-OLSR: Security Aware Optimized Link State Routing for Mobile Ad Hoc Networks." in *Proc. of IEEE International Conference on Communications 2008 (ICC '08)*, Beijing, China, May 2008.

[35] A. S. Wander, N. Gura, H. Eberle, V. Gupta, and Sh. Ch. Shantz, "Energy analysis of public-key cryptography for wireless sensor networks", In Proc. of *2005 Pervasive Computing and Communications (PERCOM'05)*, pp. 324–328, Washington, DC, USA, 2005.

[36] R. Watro, D. Kong, S. Cuti, C. Gardiner, C. Lynn, and P. Kruus, "TinyPK: Securing Sensor Networks with Public Key Technology," In Proc. of *2nd ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN '04)*, pp. 59–64, New York 2004.

[37] A. Liu and P. Ning, "TinyECC: A Configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks (Version 1.0)," Sept. 2005, [online]. *Available:* http://discovery.csc.ncsu.edu/software/TinyECC/.

[38] Y.W. Law, J. Doumen, and P. Hartel, "Survey and benchmark of block ciphers for wireless sensor networks", *ACM Transactions on Sensor Networks (TOSN)*, ISSN 1550-4859, pp. 65–93, 2006

[39] N. Fournel, M. Minier, and S. Ubéda, "Survey and benchmark of stream ciphers for wireless sensor networks", Lecture Notes in *Computer Science 4462: Information Security Theory and Practices. Smart Cards, Mobile and Ubiquitous Computing Systems*, Springer, pp.202-214, July 2007.

[40] "SkipJack and KEA Algorithm Specifications," *National Institute of Standards and Technology*, May 1998.

[41] R. Rivest, "The RC5 Encryption Algorithm," In *Proc. of Leuven Workshop on Fast Software Encryption*, pp. 86–96, Springer-Verlag, 1995.

[42] R. Rivest, M. Robshaw, R. Sidney, and Y. Yin, "The RC6^{TM} Block Cipher Specification version 1.1," Aug. 1998, [online]. *Available:* http://www.rsa.com/rsalabs/node.asp?id=2512.

[43] B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, and N. Ferguson, "Twofish: A 128-Bit Block Cipher," June 1998, [online]. *Available:* *http://www.schneier.com/twofish.html.*

[44] M. Matsui, "New Block Encryption Algorithm MISTY," *Fast Software Encryption, 4th International Workshop (FSE '97), LNCS 1267*, pp.54–68, Springer-Verlag, 1997.

[45] Specification of the 3GPP Confidentiality and Integrity Algorithms Document 2: KASUMI Specification, ETSI/SAGE Specification Version: 1.0, Dec 1999, [online]. *Available: http://www.3gpp.org/ftp/Specs/html-info/35202.htm*.

[46] K. Aoki, T. Ichikawa, M. Kansa, M. Matsui, S. Moriai, J. Nakajima, and T. Tokita, "Camellia: A 128-Bit Block Ciphers Suitable for Multiple Platforms – Design and Analysis", *Lecture Notes in Computer Science*, vol. 2012, pp. 39-56, 2001.

[47] F. Standaert, G. Piret, G. Rouvroy, J. Quisquater, and J. Legat, "ICEBERG: an Involutional Cipher Efficient for Block Encryption in Reconfigurable Hardware", *Fast Software Encryption (FSE 2004), Lecture Notes in Computer Science*, Vol. 3017, pp. 279-299, 2004.

[48] A. Bogdanov, L.R. Knudsen, G. Leander, C. Paar, A. Poschmann, M.J.B. Robshaw, Y. Seurin, and C. Vikkelsoe, "PRESENT: An Ultra-Lightweight Block Cipher," In *Proc. of Cryptographic Hardware and Embedded Systems (CHES 2007), Springer-Verlag, LNCS 4727*, pp. 450-466, 2007.

[49] D. Hong, *et. al.*, “HIGHT: A New Block Cipher Suitable for Low Resource Device,” *Cryptographic Hardware and Embedded Systems (CHES'06), Spring-Verlag, LNCS 4249*, pp. 46-59, 2006.

[50] F.-X. Standaert, G. Piret, N. Gershenfeld, and J.-J. Quisquater, “SEA: A Scalable Encryption Algorithm for Small Embedded Applications,” *Smart Card Research and Applications (CARDIS'06), Springer-Verlag, LNCS 3928*, pp. 222–236, 2006.

[51] H. Cheng, H.M. Heys, and C. Wang, "PUFFIN: A Novel Compact Block Cipher Targeted to Embedded Digital Systems", in Proc. of *Euromicro Conference on Digital System Design (DSD'08)*, Parma, Italy, Sep. 2008.

[52] W. Hongjun, "Stream cipher hc-128," eSTREAM, ECRYPT Stream Cipher Project, [online]. *Available: http://www.ecrypt.eu.org/stream/p3ciphers/hc/hc128_p3.pdf.*

[53] W. Hongjun, "Stream cipher hc-256," eSTREAM, ECRYPT Stream Cipher Project, 2005, [online]. *Available: http://www.ecrypt.eu.org/stream/hc256p2.html.*

[54] C. Berbain, O. Billet, A. Canteaut, N. Courtois, H. Gilbert, L. Goubin, A. Gouget, L. Granboulan, C. Lauradoux, M. Minier, T. Pornin and H. Sibert, "Sosemanuk, a fast software-oriented stream cipher," *eSTREAM, ECRYPT Stream Cipher Project*, [online]. *Available: http://www.ecrypt.eu.org/stream/sosemanuk.html.*

[55] P. Ekdahl and T. Johansson, "SNOW: a new stream cipher," In *Proc. of 1st NESSIE Workshop*, Heverlee, Belgique, 2000.

[56] D. Whiting, B. Schneier, and S. Lucks, "Phelix - Fast Encryption and Authentication in a Single Cryptographic Primitive", *Symmetric Key Encryption*

Workshop, Aarhus, Denmark, May, 2005. *Available:* *www.ecrypt.eu.org/stream/phelixp2.html*.

[57] D. Ed, C. Kevin, H. Matt, M. William, S. Leonie,L. Hoon-Jae, and M. SangJae, "Dragon: A fast word based stream cipher, " ECRYPT - Network of Excellence in Cryptology, Call for stream Cipher Primitives - Phase2, 2005. *Available:* *http://www.ecrypt.eu.org/stream/dragonp3.html*.

[58] B. Alex, "A New 128-bit Key Stream Cipher LEX," eSTREAM, ECRYPT Stream Cipher Project, Phase 3 [online]. *Available:* *http://www.ecrypt.eu.org/stream/ciphers/lex/lex.pdf*.

[59] B. Schneier, Applied *Cryptography, Second Edition: Protocols, Algorithms, and Source Code in C*. John Wiley & Sons, Inc, 1996.

[60] G. Meiser, T. Eisenbarth, K. Lemke-Rust, and C. Paar, "Efficient Implementation of eSTREAM Ciphers on 8-bit AVR Microcontrollers," in Proc. of *2008 Industrial Embedded Systems (SIES'08)*, pp. 58-66, La Grande Motte, France, June 2008.

[61] A.J. Menezes, P. van Oorschot, and S.A. Vanstone, *Handbook of Applied Cryptography*. CRC Press, 1997.

[62] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar, "SPINS: Security Protocols for Sensor Networks," *ACM Wireless Networks*, vol. 8, no. 5, pp. 521-534, Sept. 2002.

[63] C. Karlof, N. Sastry, and D. Wagner, "TinySec: A Link Layer Security Architecture for Wireless Sensor Networks," in Proc. of *the 2nd international*

Conference on Embedded Networked Sensor Systems (SenSys '04), pp. 162-175, New York, November 2004.

[64] National Institute of Standards and Technology (NIST), "Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC", Nov. 2007, *[Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf.*

[65] P. Rogaway, M, Bellare, J, Black and T. Krovetz, "OCB: A Block-Cipher Mode of Operation for Efficient Authenticated Encryption", Aug. 2001, [Online]. *Available:http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/ocb/ocb-spec.pdf.*

[66] P. Zalachowski, B. Ksiezopolski, and Z. Kotulski, "CMAC, CCM and GCM/GMAC: Advanced modes of operation of symmetric block ciphers in wireless sensor networks", *Information Processing Letters*. vol. 110, pp. 247-251, Mar. 2010.

[67] M. Luk, G. Mezzour, A. Perrig, and V. Gligor, "MiniSec: a Secure Sensor Network Communication Architecture," in Proc. of *the 6th International Conference on Information Processing in Sensor Networks (IPSN 2007)*, pp. 479-488, *ACM*, New York, 2007.

[68] D. Jinwala, D. Patel and K. Dasgupta, "FlexiSec: A Configurable Link Layer Security Architecture for Wireless Sensor Networks", *Journal of Information Assurance & Security: Special Issue on Information Assurance and Data Security*, vol. 4, Issue 6, pp. 582-603, USA, June 2009.

[69] L. Casado and P. Tsigas, "ContikiSec: A Secure Network Layer for Wireless Sensor Networks under the Contiki Operating System," Lecture Notes in *Computer Science 5838: Identity and Privacy in the Internet Age*, Springer, pp.133-147, Sept. 2009.

[70] T. Li, H. Wu, X. Wang and F. Bao, "SenSec Design. Technical Report-TR v1.1," *InfoComm Security Department, Institute for Infocomm Research*, 2005.

[71] J. Hongbo and J. Shudong, "LEAP: Localized Energy-Aware Prediction for data collection in wireless sensor networks," in *Proc. of 5th Mobile Ad Hoc and Sensor Systems (MASS 2008)*, pp. 491 - 496, Atlanta, Georgia, Oct. 2008.

[72] J. Grobschadl, A. Szekely and S. Tillich, "The Energy Cost of Cryptographic Key Establishment in Wireless Sensor Networks," in Proc. of *the 2nd ACM Symposium on information, Computer and Communications Security*, pp. 380-382, Singapore, March, 2007.

[73] H. Alzaid, E. Foo, and J. G. Nieto, "Secure data aggregation in wireless sensor network: a survey," In Proc. of *the 6th Australasian Conference on information Security*, vol. 81, NSW, Australia, Jan., 2008.

[74] J. Domingo-Ferrer, "A Provably Secure Additive and Multiplicative Privacy Homomorphism," *Information Security, LNCS 2433*, pp.471–483, Springer-Verlag, 2007.

[75] Z. Yong, N. Xia-mu, L. Jun-cao, and L. Chun-ming, "Research on a Novel Hashing Stream Cipher," *Computational Intelligence and Security, LNCS 4456*, pp.481–490, Springer-Verlag, 2007.

[76] L. Hu, and D. Evans, “Secure aggregation for wireless network” in *Annual International Symposium on Applications and the Internet Workshops (SAINT)*, pp. 384–394, Florida, USA, Jan. 2003.

[77] B. Przydatek, D. X. Song, and A. Perrig, “SIA: Secure Information Aggregation in Sensor Networks”, in Proc. of *1st ACM Conference on Embedded Networked Sensor Systems (Sensys’03)*, pp. 255–265, Los Angeles, USA, Nov. 2003.

[78] H. O. Sanli, S. Ozdemir, and H. Cam, “SRDA: Secure reference-based data aggregation protocol for wireless sensor networks”, In Proc. of *60th Vehicular Technology Conference (VTC'04)*, pp. 4650– 4654, Milan, Italy, Apr. 2004.

[79] H. Chan,A. Perrig, and D. Song, "Secure hierarchical in-network aggregation in sensor networks", In Proc. of *the 13th ACM Conference on Computer and Communications Security (CCS '06)*, pp. 278-287, Virginia, USA, Oct. 2006.

[80] Y. Yang, X. Wang, S. Zhu, and G. Cao, “SDAP: a secure hop-by-hop data aggregation protocol for sensor networks”, in Proc. of *the 7th ACM Interational Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc’06)*, pp. 356–367, Florence, Italy, May 22-25.

[81] D. Westhoff, J. Girao, and M. Acharya, "Concealed Data Aggregation for Reverse Multicast Traffic in Sensor Networks: Encryption, Key Distribution, and Routing Adaptation", *IEEE Transactions on Mobile Computing*, vol. 5, pp. 1417-1431, Oct. 2006.

[82] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System Architecture Directions for Networked Sensors," in *Proc. of Special Interest Group on Programming Languages (SIGPLAN), ACM*, pp. 93-104, Nov. 2000.

[83] "8-bit Microcontroller with 128K Bytes In-System Programmable Flash", [online]. *Available : http://www.atmel.com/dyn/resources/prod_documents/doc2467.pdf.*

[84] "AVR236: CRC Check of Program Memory", [online]. *Available: http://www.atmel.com/dyn/resources/prod_documents/doc1143.pdf.*

[85] W. K. Koo, H. Lee, Y. H. Kim and D. H. Lee, "Implementation and Analysis of New Lightweight Cryptographic Algorithm Suitable for Wireless Sensor Networks," in Proc. of *2008 Information Security and Assurance (ISA 2008)*, pp.73-76, Korea, April 2008.

[86] M. Henricksen, "Tiny Dragon - An Encryption Algorithm for Wireless Sensor Networks," in *Proc. of 10th High Performance Computing and Communications, (HPCC '08)*, pp. 795-800, 25-27, DaLian, China, Sept. 2008.

[87] R. Tahir, M. Y. Javed, M. Tahir and F. Imam, "LRSA: Lightweight Rabbit Based Security Architecture for Wireless Sensor Networks," in Proc. of *2008 Intelligent Information Technology Application (IITA '08)*, vol.3, pp. 679-683, China, Dec. 2008.

[88] T. Eisenbarth, S. Kumar, C. Paar, A. Poschmann, and L. Uhsadel, "A Survey of Lightweight Cryptography Implementations", *IEEE Design & Test of Computers -- Special Issue on Secure ICs for Secure Embedded Computing*, vol 24, no 6, pp. 522-533, November 2007.

[89] A. Youssef, S.E. Tavares, and H.M. Heys, "A New Class of Substitution-Permutation Networks", in Proc. of *Workshop on Selected Areas in Cryptography(SAC '96)*, Queen's University, Kingston, Ontario, Aug. 1996.

[90] X. Zhang, H.M. Heys, and C. Li, " Energy Efficiency of Symmetric Key Cryptographic Algorithms in Wireless Sensor Networks," in Proc. of *25th Biennial Symposium on Communications (QBSC 2010)*, Kingston, Ontario, May 2010.

[91] L. Granboulan, "Flaws in Differential Cryptanalysis of Skipjack," Lecture Notes in *Computer Science 2355: Fast Software Encryption*, Springer-Verlag, pp. 81-98, 2002.

[92] L. R. Knudsen, M. J. B. Robshaw and D. Wagner, "Truncated Differentials and Skipjack," Lecture Notes in *Computer Science 1666: Advances in Cryptology – CRYPTO'99*, Springer-Verlag, pp. 790, 1999.

[93] J. Daemen, V. Rijmen, The Design of Rijndael: *AES - the Advanced Encryption Standard*, Springer-Verlag New York, 2002.

[94] C. Berbain, O. Billet, A. Canteaut, N. Courtois, H. Gilbert, L. Goubin, A. Gouget, L. Granboulan, C. Lauradoux, M. Minier, T. Pornin, and H. Sibert, "SOSEMANUK, a fast software-oriented stream cipher," *[Online]. Available: http://www.ecrypt.eu.org/stream/p3ciphers/sosemanuk/sosemanuk p3.pdf*

[95] D. J. Bernstein, "Salsa20," *[Online]. Available: http://www.ecrypt.eu.org/stream/p3ciphers/salsa20/salsa20 p3.zip*

[96] National Institute of Standards and Technology (NIST), "Recommendation for Block Cipher Modes of Operation Methods and Techniques," *[online]. Available: http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf*

[97] National Institute of Standards and Technology (NIST), "Proposal To Extend CBC Mode By Ciphertext Stealing," *Special Publication (SP) 800-38A*, May. 2007.

[98] P. S.L.M. Barreto and V. Rijmen, "The KHAZAD Legacy-Level Block Cipher," [online]. *Available: http://www.larc.usp.br/~pbarreto/KhazadPage.html*

[99] J. Majeed, "FPGA based communication security for wireless sensor networks," In Proc. of *the 4th international Conference on Electronic, Signal Processing and Control (WSEAS'05)*, Rio de Janeiro, Brazil, Apr, 2005.

[100] P. Muralidhar, and C.B.Rama Rao, "Reconfigurable Wireless Sensor Network Node based on NIOS core," In Proc. of *4th Wireless Communication and Sensor Networks (WCSN 2008)*, pp. 67-72, Dec. 2008.

[101] S. Peter, O. Stecklina, J. Portilla, E. de la Torre, P. Langendoerfer, and T. Riesgo, "Reconfiguring Crypto Hardware Accelerators on Wireless Sensor Nodes," In *Proc. of 6th Sensor, Mesh and Ad Hoc Communications and Networks Workshops*, (SECON Workshops '09), Rome, Italy, June 2009.

[102] Y. E. Krasteva, J. Portilla, J. M. Carnicer, E. de la Torre, and T. Riesgo, "Remote HW-SW Reconfigurable Wireless Sensor Nodes", In Proc. of *34th Annual Industrial Electronics (IECON 2008)*, pp. 2483 – 2488, Orlando, FL, Nov. 2008.

[103] J. Mihel, and R. Magjarevic, "FPGA based two-channel ECG sensor node for wearable applications," *IFMBE Proceedings: 4th European Conference of the International Federation for Medical and Biological Engineering*, Springer Berlin Heidelberg, vol. 22, pp. 1208-1211, Feb. 2009.

[104] H. Hinkelmann, P. Zipf, M. Glesner, "Design Concepts for a Dinamically Reconfigurable Wireless Sensor Node", in Proc. of *the 1st NASA/ESA Conference on Adaptive Hardware and Systems (AHS'06)*, pp. 436-441, Jun 2006.

[105] E. Susu, M. Magno, A. Acquaviva, D. Atienza, "Reconfiguration Strategies for Environmentally Powered Devices: Theoretical Analysis and Experimental Validation", *Transactions on High-Performance Embedded Architectures and Compilers I (HiPEAC I)*, pp. 341-360, 2007.

[106] J. Portilla, A. de Castro, E. de la Torre, T. Riesgo, "A Modular Architecture for Nodes in Wireless Sensor Networks", *Journal of Universal Computer Science (JUCS)*, vol. 12, no 3, Mar. 2006,pp. 328-339.

[107] H. Belhadj, V. Aggrawal, A. Pradhan, and A. Zerrouki, "Power-Aware FPGA Design,"[online]*Available:http://www.actel.com/documents/Power_Aware_WP.pdf.*

[108] "XPower Tutorial FPGA Design", [online]. *Available: ftp://ftp.xilinx.com/pub/documentation/...tutorials/xpowerfpgatutorial.pdf.*

Appendix A: S-box

Table A.1 the KHAZAD Involutional S-box

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	BA5	4	2F	74	53	D3	D2	4D5	0	AC	8D	BF	70	52	9A	4C
1	EA	D5	97	D1	33	51	5B	A6	DE	48	A8	99	DB	32	B7	FC
2	E3	9E	91	9B	E2	BB	41	6E	A5	CB	6B	95	A1	F3	B1	02
3	CC	C4	1D	14	C3	63	DA	5D	5F	DC	7DC	D	7F	5A	6C	5C
4	F7	26	FF	ED	E8	9D	6F	8E	19	A0	F0	89	0F	07	AF	FB
5	08	15	0D	04	01	64	DF	76	79	DD	3D	16	3F	37	6D	38
6	B9	73	E9	35	55	71	7B	8C	72	88	F6	2A	3E	5E	27	46
7	0C	65	68	61	03	C1	57	D6	D9	58	D8	66	D73	A	C8	3C
8	FA	96	A7	98	EC	B8	C7	AE	69	4B	AB	A9	67	0A	47	F2
9	B5	22	E5	EE	BE	2B	81	12	83	1B	0E	23	F5	45	21	CE
A	49	2C	F9	E6	B6	28	17	82	1A	8B	FE	8A	09	C9	87	4E
B	E1	2E	E4	E0	EB	90	A4	1E	85	60	00	25	F4	F1	94	0B
C	E7	75	EF	34	31	D4	D0	86	7E	AD	FD	29	30	3B	9F	F8
D	C6	13	06	05	C5	11	77	7C	7A	78	36	1C	39	59	18	56
E	B3	B02	4	20	B2	92	A3	C0	44	62	10	B4	84	43	93	C2
F	4A	BD	8F	2D	BC	9C	6A	40	CF	A2	80	4F	1F	CA	AA	42

Table A.2 the BSPN Involutional S-box

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	00	41	E5	F6	A0	AD	7B	DD	73	B0D	6	97	4F	C7	DC	38
1	4C	B4	AA	A2	99	D7	4E	5F	A7	CF	F1	CC	EB	BF	EE	B1
2	94	91	5A	62	55	3B	8B	C9	3E	E2	DB	EA	63	44	5D	85
3	D3	AC	FC	6C	FB	39	66	A9	0F	35	DF	25	77	D0	28	E6
4	A8	01	C8	BD	2D	89	8E	69	58	78	8A	EC	10	DA	16	0C
5	ED	D2	E7	CA	B5	24	87	D9	48	C1	22	70	AE	2E	83	17
6	E9	76	23	2C	88	A6	36	D5	72	47	6E	D8	33	CD	6A	C4
7	5B	B7	68	08	EF	95	61	3C	49	BE	8C	06	7F	FA	81	7C
8	82	7E	80	5E	96	2F	8F	56	64	45	4A	26	7A	FD	46	86
9	DE	21	93	92	20	75	84	0BB	C	14	9F	B6	F9	AF	A1	9A
A	04	9E	13	C3	F0	B8	65	18	40	37	12	B3	31	05	5C	9D
B	09	1F	E0	AB	11	54	9B	71	A5	F5	CE	C2	98	43	79	1D
C	F4	59	BB	A3	6F	D4	E4	0D	42	27	53	F3	1B6	D	BA	19
D	3D	F8	51	30	C5	67	0A	15	6B	57	4D	2A	0E	07	90	3A
E	B2	F2	29	FF	C6	02	3F	52	F7	60	2B	1C	4B	50	1E	74
F	A4	1A	E1	CB	C0	B9	03	E8	D1	9C	7D	34	32	8D	FE	E3

Appendix B: Simulation Flowchart

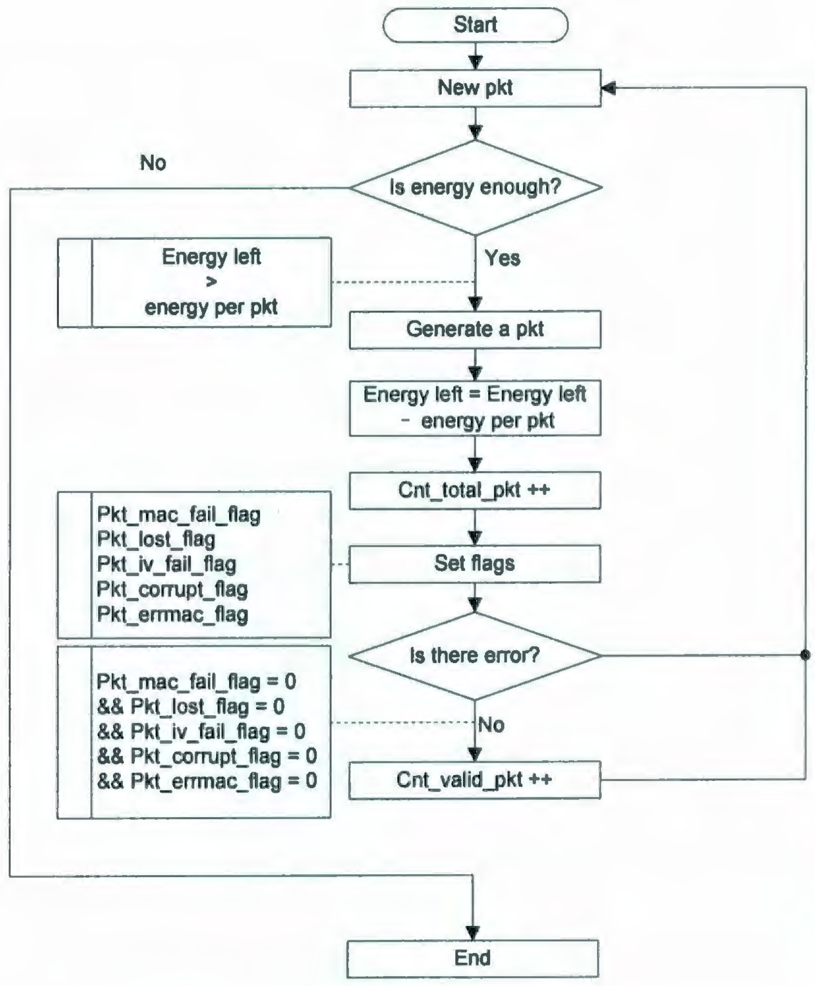


Fig.B.1. Flow chart of Implicit IV scheme and TinySec scheme.

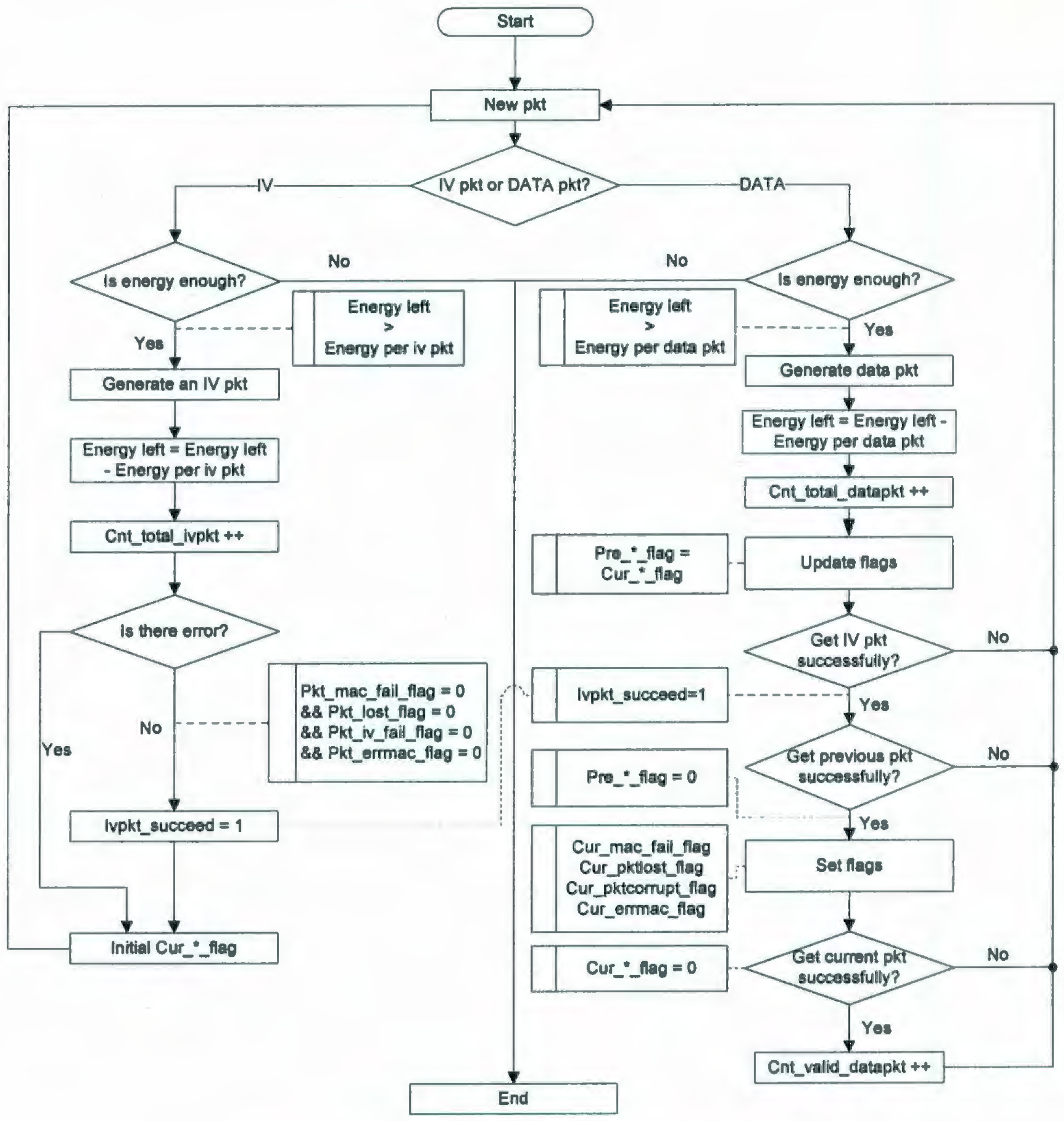

Start
New pkt
IV pkt or DATA pkt?
IV
DATA
Is energy enough?
No
Yes
Energy left > Energy per iv pkt
Generate an IV pkt
Energy left = Energy left - Energy per iv pkt
Cnt_total_ivpkt ++
Is there error?
Yes
No
Pkt_mac_fail_flag = 0 && Pkt_lost_flag = 0 && Pkt_iv_fail_flag = 0 && Pkt_errmac_flag = 0
Ivpkt_succeed = 1
Initial Cur_*_flag
Is energy enough?
No
Yes
Energy left > Energy per data pkt
Generate data pkt
Energy left = Energy left - Energy per data pkt
Cnt_total_datapkt ++
Pre_*_flag = Cur_*_flag
Update flags
Get IV pkt successfully?
No
Ivpkt_succeed=1
Yes
Get previous pkt successfully?
No
Pre_*_flag = 0
Yes
Cur_mac_fail_flag Cur_pktlost_flag Cur_pktcorrupt_flag Cur_errmac_flag
Set flags
Get current pkt successfully?
No
Cur_*_flag = 0
Yes
Cnt_valid_datapkt ++
End

Fig.B.2 Flow chart of periodic IV without ACK scheme.

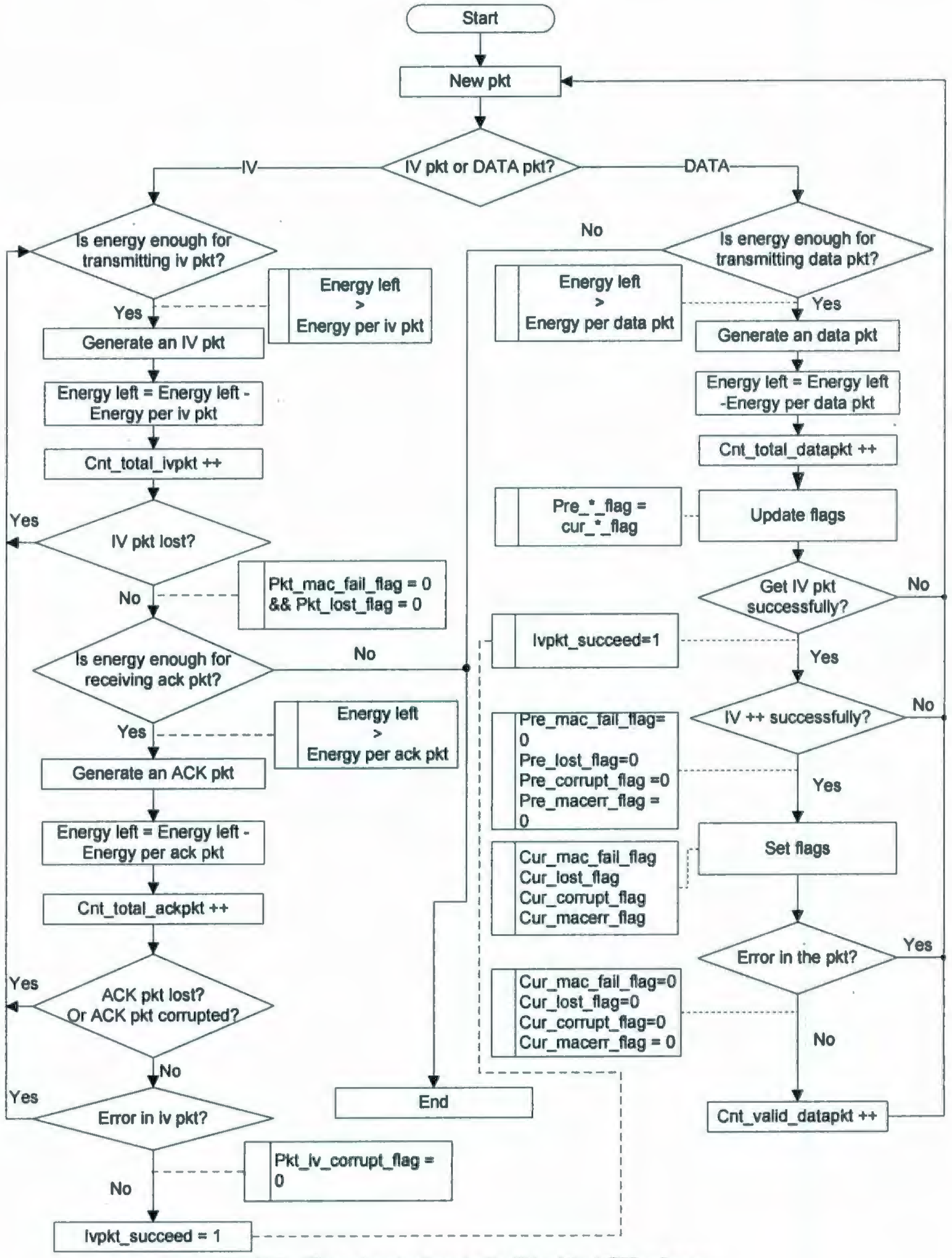


Fig.B.3. Flow chart of periodic IV with ACK scheme.

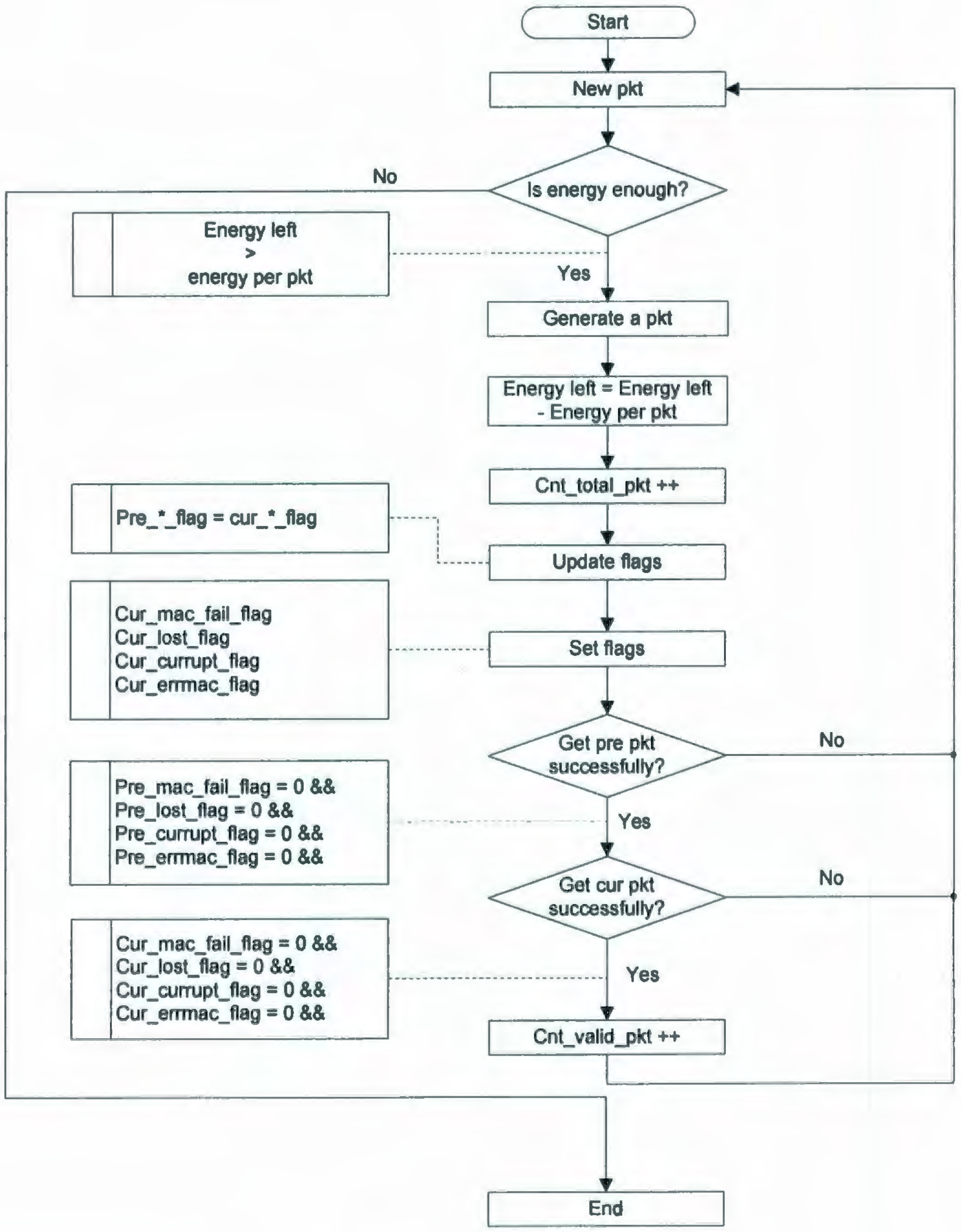


Fig.B.4. Flowchart of CFB scheme.

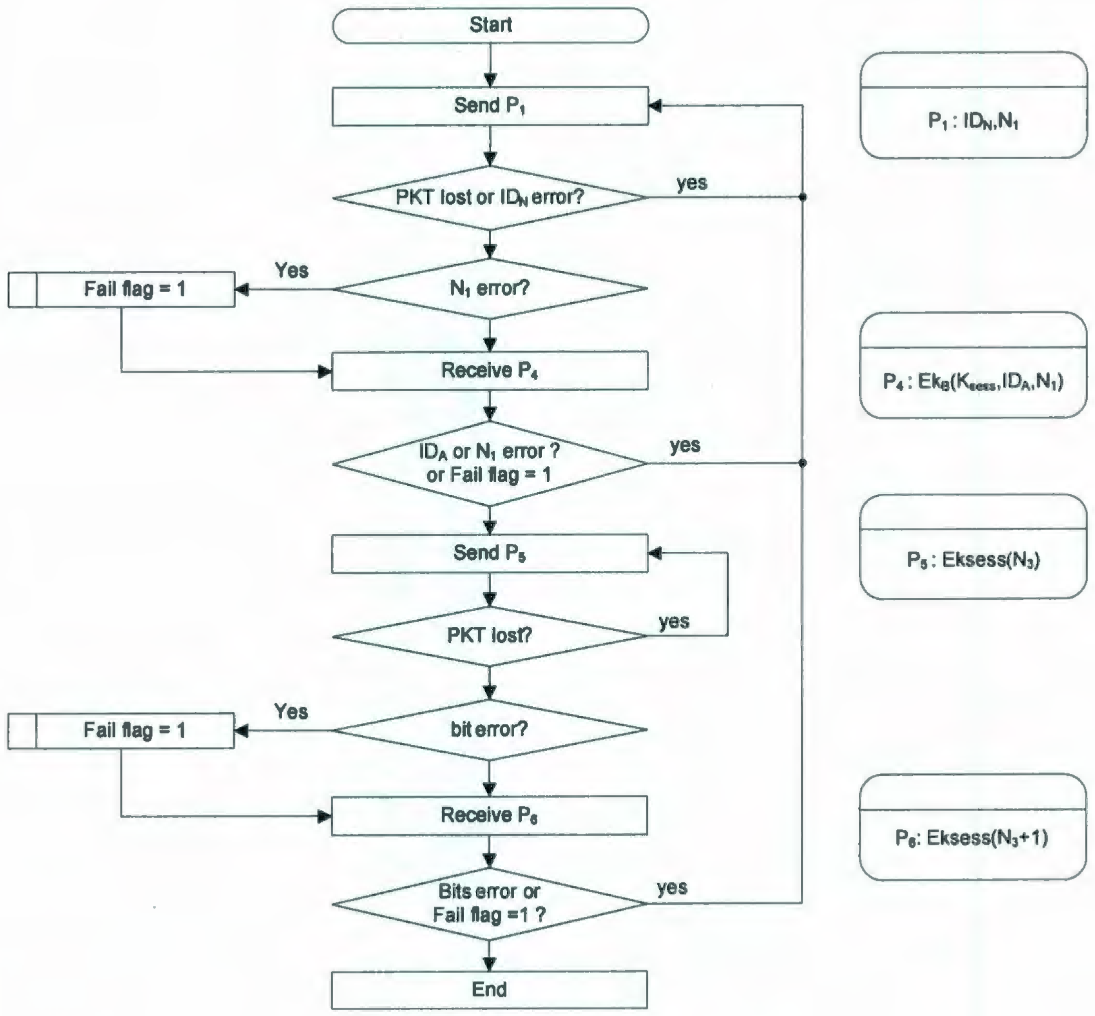


Fig.B.5. Flowchart of key distribution.

Appendix C: Simulation Results

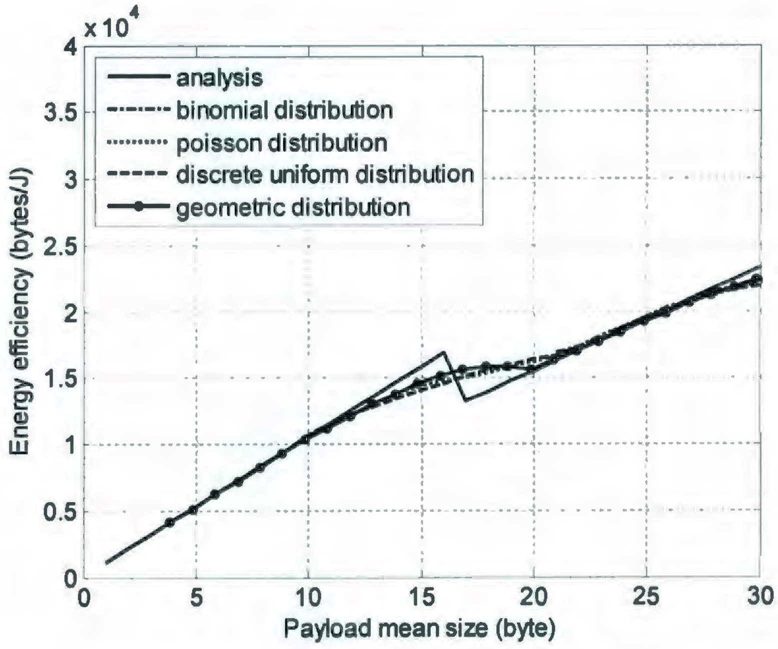


Fig. C.1. Implicit IV scheme with different distributions.

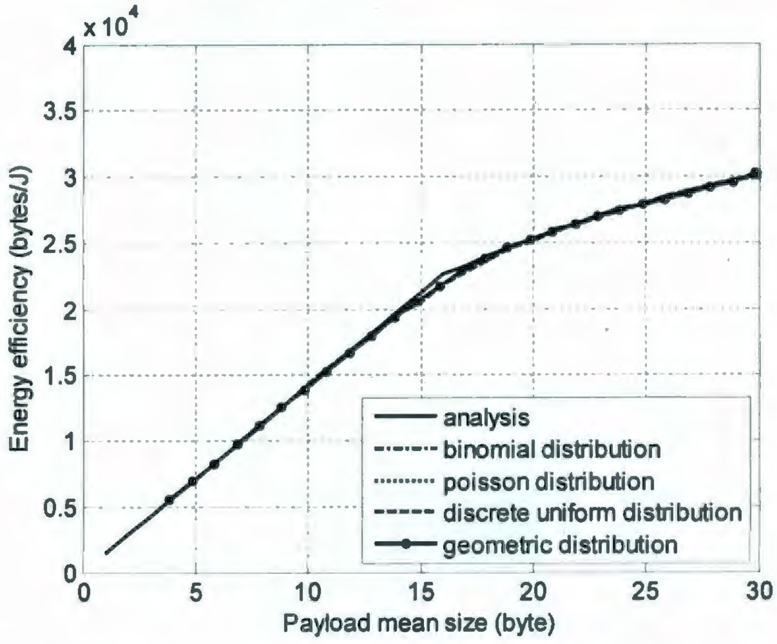


Fig. C.2. TinySec scheme with different distributions.

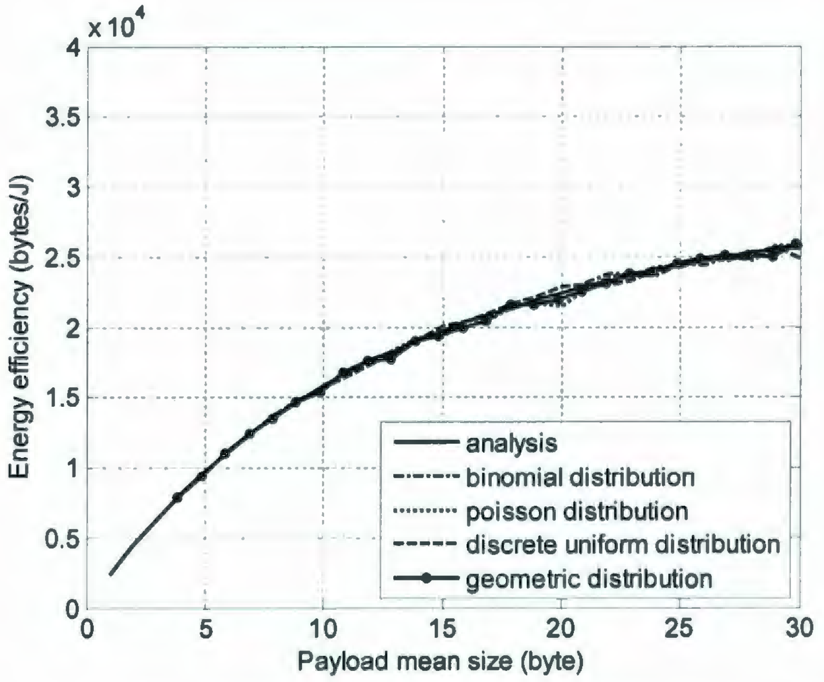


Fig. C.3. Periodic IV without ACK scheme with different distributions.

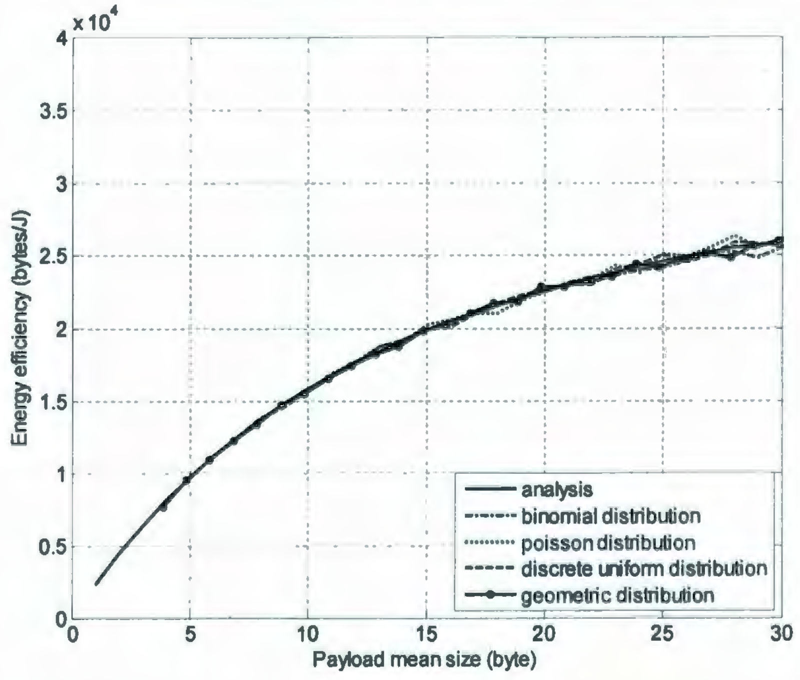


Fig. C.4. Periodic IV with ACK scheme with different distributions.

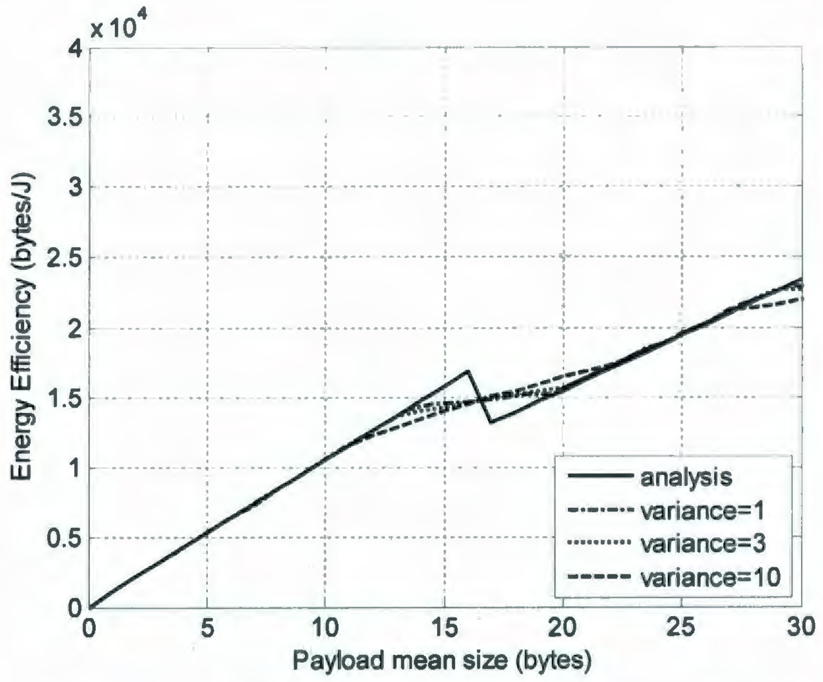


Fig. C.5. Implicit IV scheme with discrete uniform distribution.

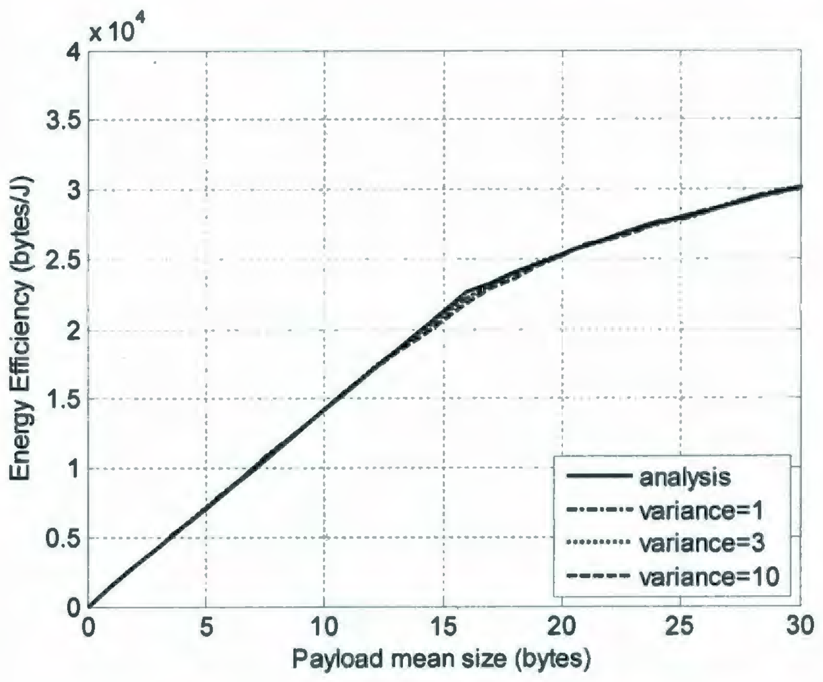


Fig. C.6. TinySec scheme with discrete uniform distribution.

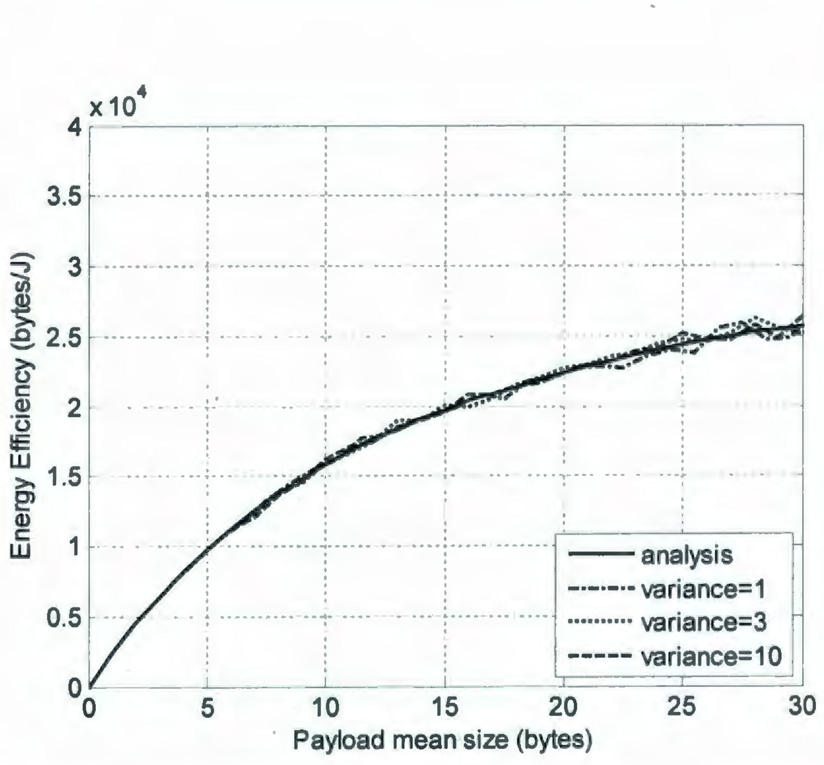


Fig. C.7. Periodic IV without ACK scheme without discrete uniform distribution.

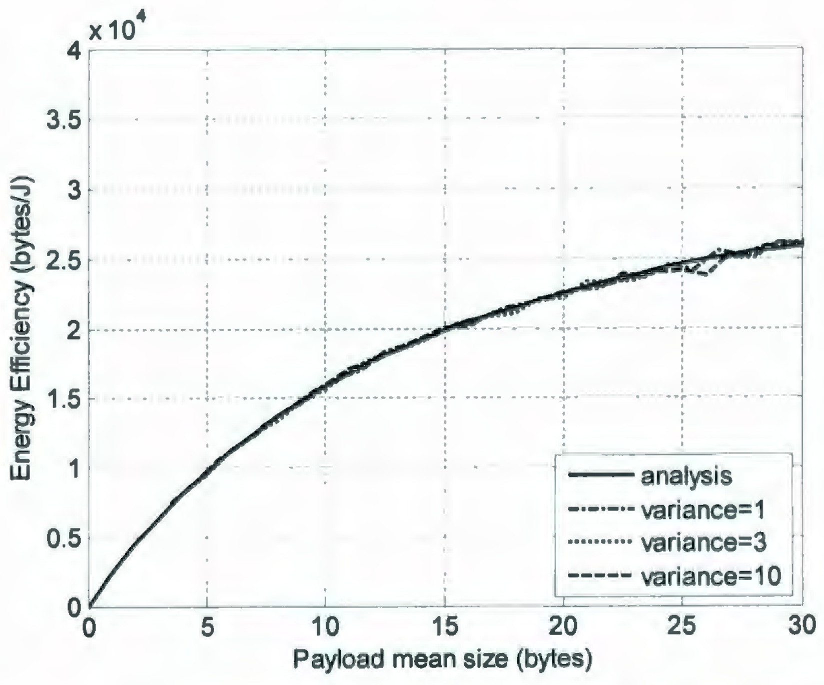


Fig. C.8. Periodic IV without ACK scheme with discrete uniform distribution.

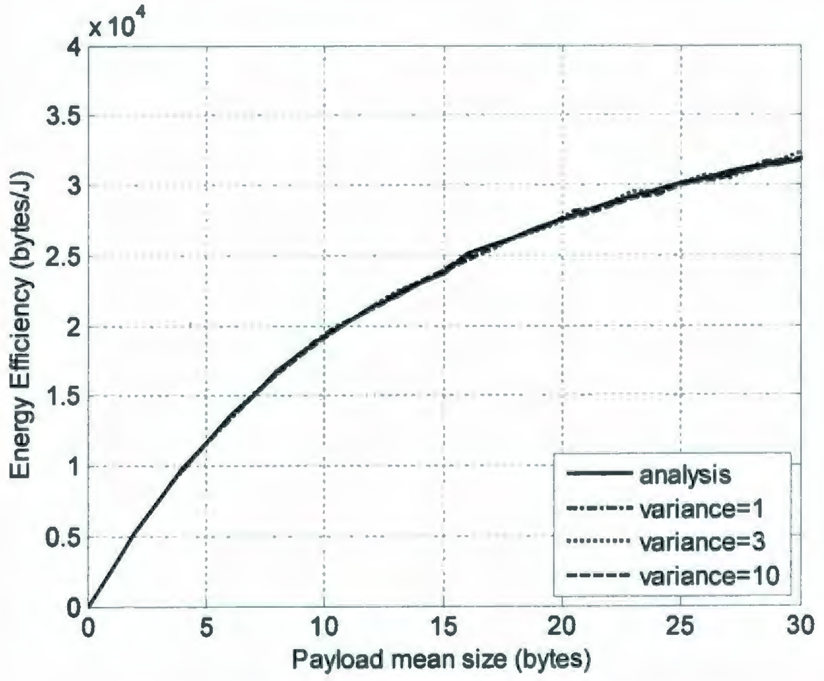


Fig. C.9. CFB scheme with discrete uniform distribution.

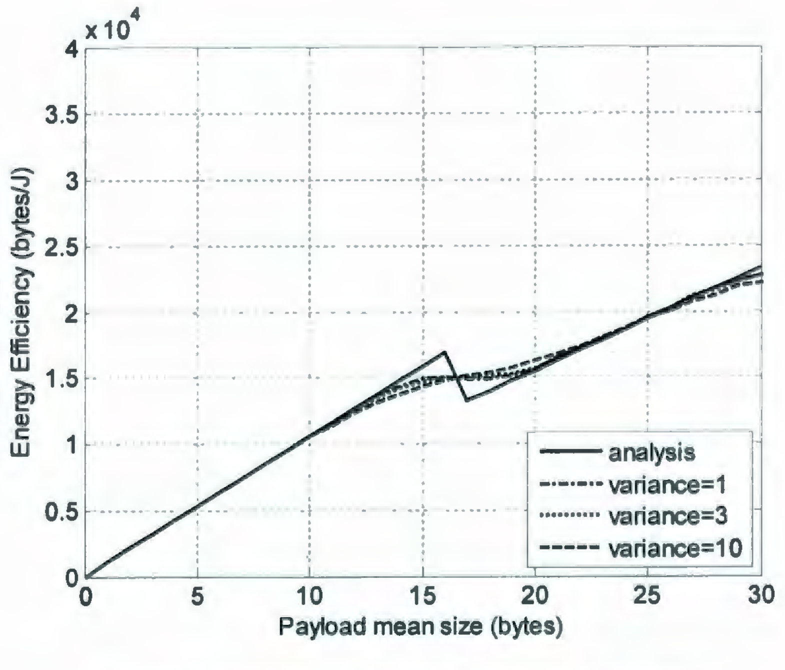


Fig. C.10. Implicit IV scheme with Poisson distribution.

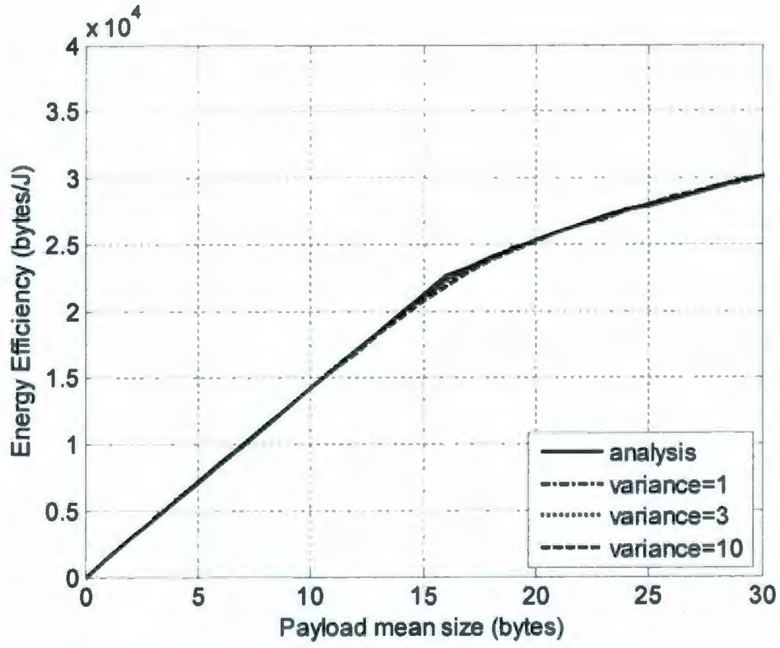


Fig. C.11. TinySec scheme with Poisson distribution.

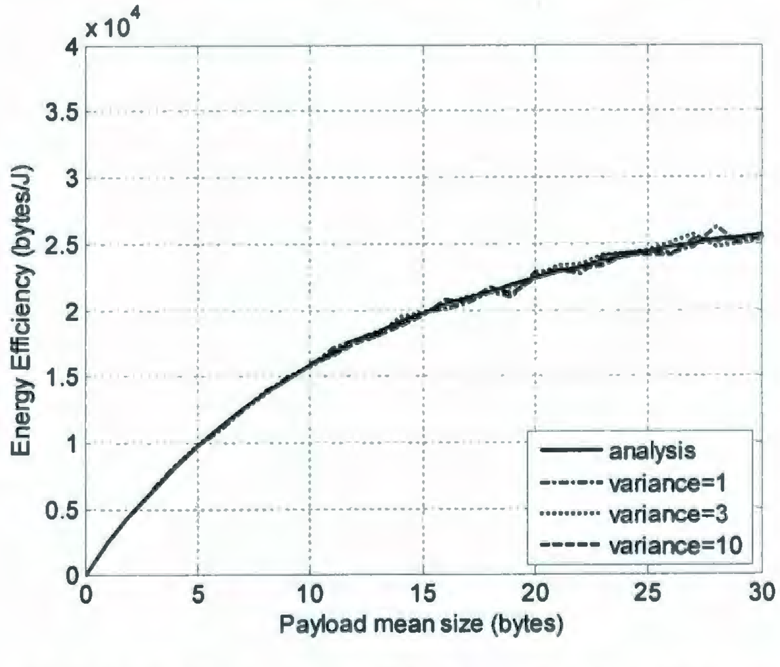


Fig. C.12. Periodic IV without ACK scheme with Poisson distribution.

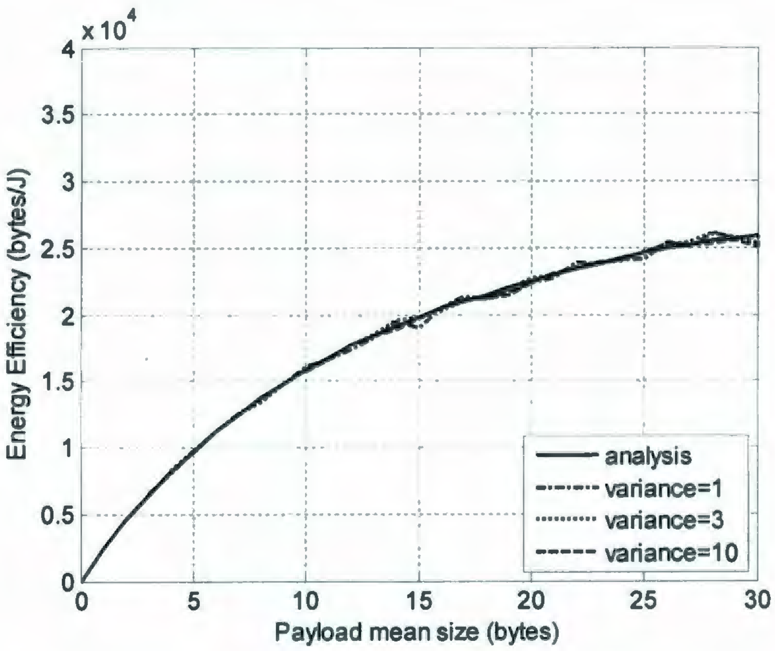


Fig. C.13. Periodic IV with ACK scheme with Poisson distribution.

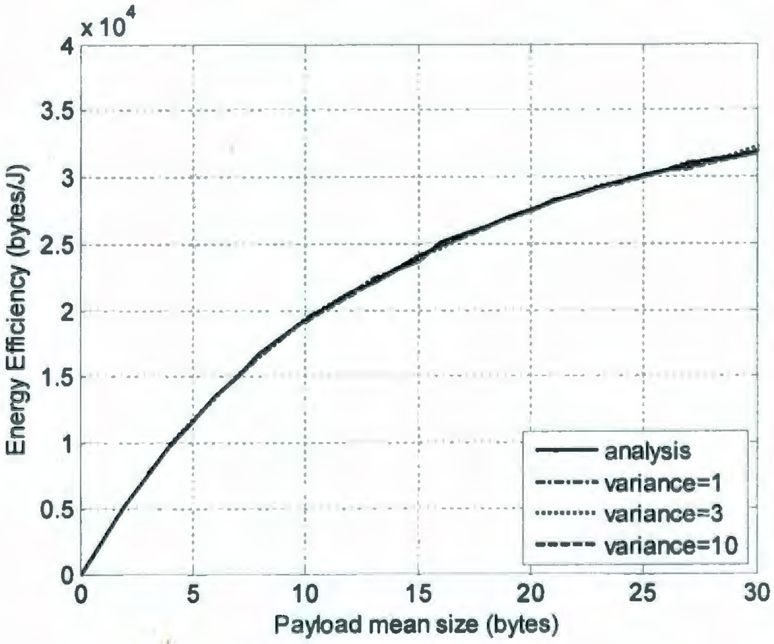


Fig. C.14. CFB scheme with Poisson distribution.

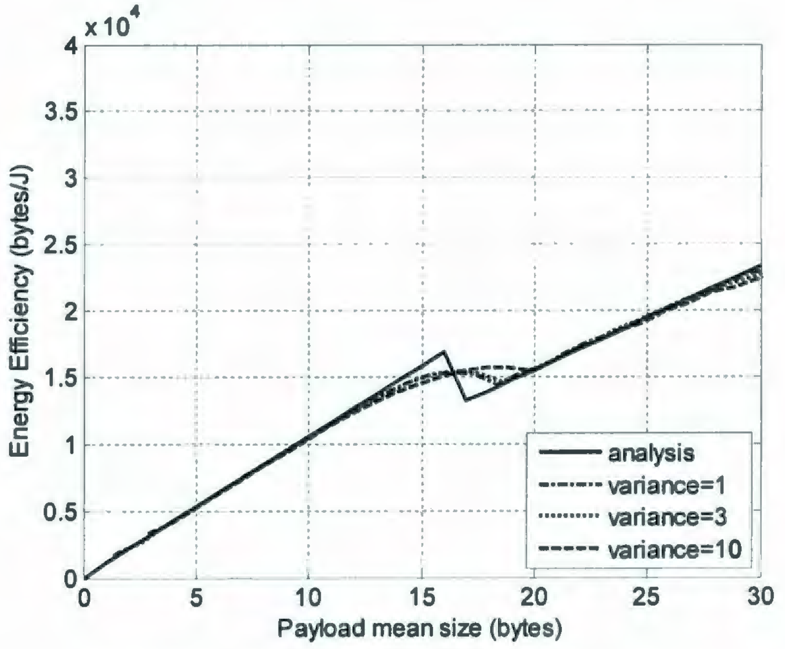


Fig. C.15. Implicit IV scheme with geometric distribution.

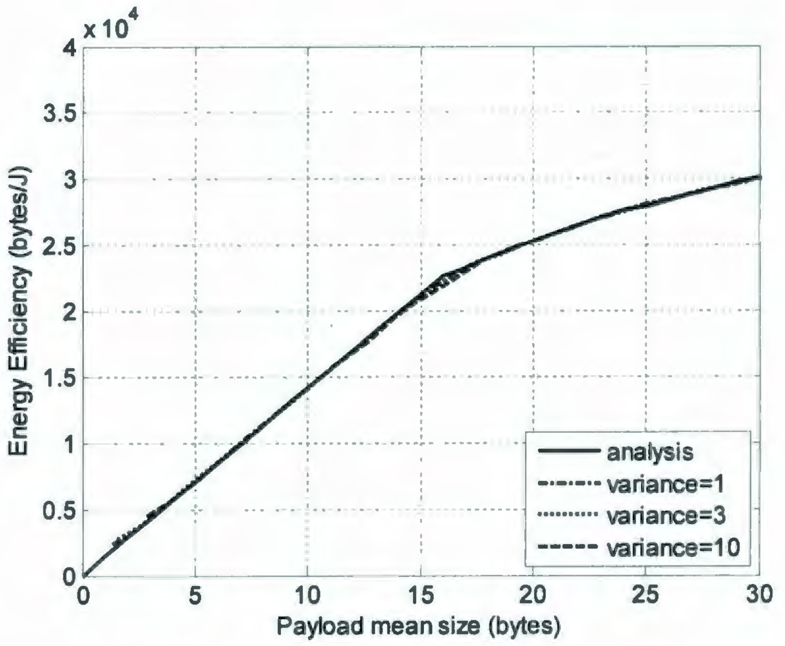


Fig. C.16. TinySec scheme with geometric distribution.

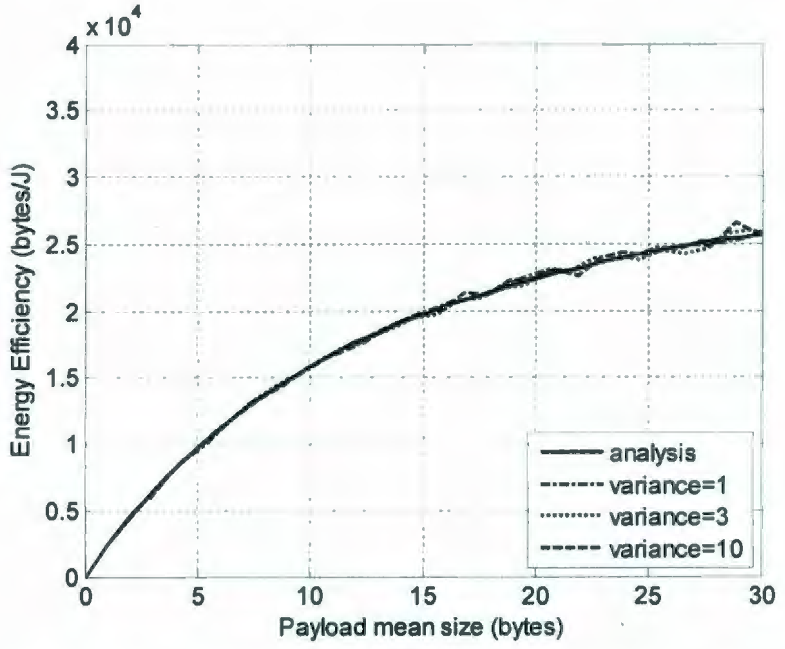


Fig. C.17. Periodic IV without ACK scheme with geometric distribution.

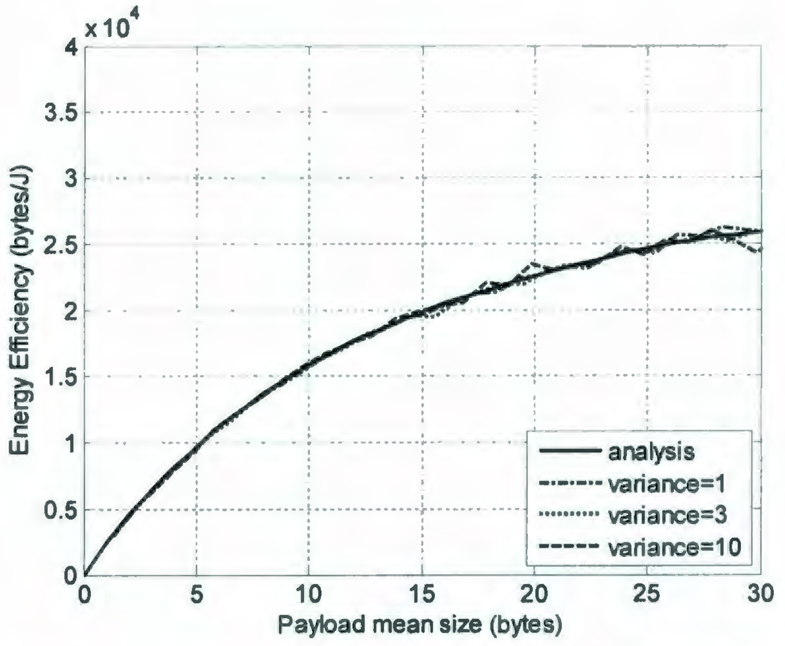


Fig. C.18. Periodic IV without ACK scheme with geometric distribution.

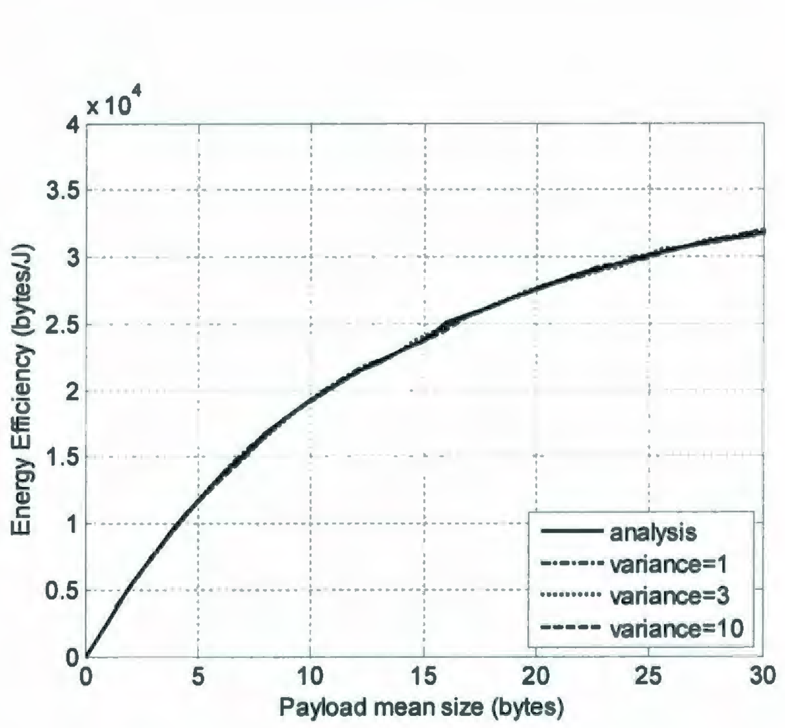


Fig. C.19. CFB scheme with geometric distribution.