Energy Consumption of Encryption Schemes in Wireless Devices

by

Sohail Hirani

Bachelor of Engineering in Electronics and Telecommunications, University of Pune, 1999

Bachelor of Technology in Computer Systems, Open University of British Columbia, 2001

Master of Science in Telecommunications, University of Pittsburgh, 2003

Submitted to the Graduate Faculty of

School of Information Science in partial fulfillment

of the requirements for the degree of

Master of Science in Telecommunications

University of Pittsburgh

[2003]

UNIVERSITY OF PITTSBURGH

SCHOOL OF INFORMATION SCIENCE

DEPARTMENT OF INFORMATION SCIENCE AND TELECOMMUNICATIONS

This thesis was presented

by

Sohail Hirani

It was defended on

April 9, 2003

and approved by

Dr. Richard Thompson, Director and Professor

Dr. Joseph Kabara, Assistant Professor

Thesis Advisor: Dr. Prashant Krishnamurthy, Assistant Professor

Energy Consumption of Encryption Schemes in Wireless Devices

Sohail Hirani, MST

University of Pittsburgh, 2003

Resources in the wireless environment are limited. The processor has limited capacity and there is limited battery power available. The increasing demand for services on wireless devices has pushed technical research into finding ways to overcome these limitations. As the penetration of wireless devices increases and applications become more critical every day, security of wireless networks has come under heavy scrutiny. However since most of the current communication algorithms are designed and tested for use in the wired environment, they cannot be used directly in Wireless LANs.

Encryption, which is the backbone of security protocols, is computationally intensive and consumes energy and computational resources that are limited in wireless devices. The current encryption standards used in wireless systems are not very secure. Also, the wireless network interface draws a significant fraction of total power consumed by the mobile device. Collisions and retransmissions lead to additional consumption of power. To design energy efficient secure protocols for wireless devices there is a need to understand how encryption affects the consumption of battery power with and without data transmission.

The research work carried out in this thesis, provides results that encourage having encryption schemes as software implementation in Wireless LANs and provides results reflecting the advantages of doing so. Various symmetric key and asymmetric key algorithms have been evaluated with different key sizes and on different devices. Effect of varying signal to noise ratio and varying packet sizes has been studied. Further, some suggestions for design of secure communications systems to handle the varying wireless environment have been provided.

ACKNOWLEDGEMENT

I would like to take this opportunity to extend my heartfelt gratitude to my Graduate Advisor Dr. Prashant Krishnamurthy for his invaluable support and encouragement. His continuous guidance and help has been instrumental throughout the progress of this research work. I would also like to thank the members of the committee Dr. Richard Thompson and Dr. Joseph Kabara for all their help. This work was partially supported by the NSF ITR/EWF grant number. 0081327 and the NIST CIP grant number 60NANB1D0120. I would specifically like to express my gratitude to both NSF and NIST for this support.

I would also like to express my appreciation to Dr. David Tipper and other members of the Wireless Information Assurance Research Group for their constructive advice, namely: Chalermpol, Kamol, Chutima, Ryan, and Harita. I would specially like to thank PhD students Phongsak Prasithsangaree and Tanapat Anusas-Amronkul for their suggestions and assistance. In addition, I wish to extend my thanks to my unending list of friends in the MST program, specifically Eric and Sopan for their comments and help in preparing for the defense.

I would like to express my heartfelt gratitude to Aga Khan Foundation for supporting my and many other deserving students' Graduate studies and making it possible through its International Scholarship Program (AKF-ISP). Above all, I would like to thank my mother Farida, my father Abdulla, brother Adil, sister Runa and my close friends in India for their love and support that has been a constant source of motivation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1
Introduction

In recent years, wireless connectivity has been gaining increasing attention with devices like laptops, PDAs, Pocket PCs and handhelds [1]. Features like nomadic access, rapid network configuration, and lack of wires make wireless networks particularly attractive [2]. Individuals are using wireless technology for storing private communications, for mobile commerce, emails and business interactions. The increasing importance of wireless systems provides malicious entities greater incentives to step up their efforts to gain unauthorized access to the information being exchanged over the wireless link [1]. The security risks in the wireless environment are particularly important because the wireless devices in the recent past have not been developed with security of the systems in mind. The problem posed by potential breaching of the systems by passive observations and masquerading is further complicated by the varying nature of the wireless environment and its limited resources.

Security is provided through security services [3]. Confidentiality of data ensures that only the authorized person reads the data and others are prevented from doing so. Confidentiality in wireless communications is achieved by transmission of encrypted secure data and maintaining the secrecy of the keys used for encryption. Authentication involves ensuring that the source of the received message is identified correctly. Integrity enables being sure that the data has not been modified by an unauthorized entity. Access control restricts the information availability to allowed parties. Availability requires the system to be available to the authorized entities whenever needed and non-repudiation ensures that neither the origination nor the receptor of the information can deny the transaction.

Studies indicate that the growth of wireless networks is being restricted by their perceived insecurity [4]. The amount of security required by the system may depend on the organization using the wireless network. A financial company would require very strong security mechanisms to prevent unauthorized users and maintain information confidentiality. The hot-spots networks

may require that only legitimate users access the network and may not require confidentiality and data integrity. On the other hand community networks may require no security at all. The protocols for wireless LAN security are still evolving to meet the needs of serious users. Until the systems provide provable security, institutional policies related to wireless network access would be based on a more cautious approach.

Wired systems are inherently more secure than the wireless systems because of the wired connectivity [3]. Eavesdropping on a wired system is relatively more difficult. The eavesdropper in a wired LAN needs to be connected to the LAN. The physical connectivity could come through a current employee, a dial up connection or through the wiring closet of the premise. However, in the wireless world the connection to a LAN is not limited by the requirement of physical connectivity. When the information on the LAN is broadcast using radio waves the vulnerability to eavesdropping and intrusion are highly increased. The wireless interface can be easily configured to listen to packets being transmitted in a promiscuous mode. With further modification malicious information can be injected into the wireless network with such a compromised system. This leads to possibility of eavesdropping and intrusion by attackers that are not even inside the premises. Wireless systems are thus prone to the vulnerabilities of the wired systems along with increased chances of security failure. An unprotected 802.11 network can be hacked in seconds while one protected by wired equivalent privacy (WEP) can still be hacked in a matter of hours [5]. Also the possibility of the small wireless devices being lost and then being found by malicious users puts additional risks in the system. Hence, the security mechanisms implemented for the wired systems cannot be used directly in the wireless environment.

Security protocols implement mechanisms through which security services can be provided. Security can be implemented at the transmission level through the means of frequency hopping and spread spectrum technologies. Such schemes would prove to be very expensive for the users and the companies employing such schemes [6]. Moreover, they primarily use Linear Feedback Shift Registers (LFSRs) [7] that are easy to break. Mathematics has been developed in the past that allows analysis of the LFSR [8]. For cost and simplicity the method that seems to be gaining acceptance is data encryption. The IEEE 802.11 standard uses the WEP protocol for security [9].

It operates at the data link layer. IP Security (IPSec) provides security at the network layer [3]. Secure Socket Layer (SSL) provides security at the transport layer for secure transactions on the Internet [10]. All these protocols rely on encryption or encryption related mechanisms to provide the security services. Encryption in this sense is thus the backbone of security services.

The above protocols have been designed for wired systems. In wireless systems, a security protocol should also consider the limited battery power, small memory and limited processing capabilities of the devices and the available bandwidth. In addition, the systems need to be able to cater to the requirements of the wide variety of wireless devices that could be used for connectivity. Study of the energy consumption of the encryption schemes in wireless devices is thus essential in design of energy efficient security protocols tailored to the wireless environment.

1.1 Motivation

Emerging trends indicate that the future wireless networks will contain a hybrid infrastructure based on fixed, mobile and ad hoc topologies and technologies [11]. The future hybrid networks would contain cellular links, high-speed access, programmable multimode radios, and wireless LANs. Devices used in such networks would be high-speed servers, desktops, handhelds, PDAs, cell phones and wireless sensors. Reliance on these networks requires that security be assured on them. Also the diversity of the devices demands that the system should be able to adapt according to the capabilities of the wireless device being used by the user. Increasing research is being done towards developing wireless systems with built in security.

The lessons learnt from the discovered insecurities of current protocols are being explored further. IEEE 802.11 WLAN and CDPD mobile data service use a 40-bit key in the encryption algorithm. IS-136 uses a 64-bit key that is more secure, but still considered weak [12]. Wireless LAN Security and its vulnerability have been explored by [5]. In designing a security system for hybrid networks, the analyst faces significant questions about the mechanisms that are to be used, the algorithms that should be used and at the layer of the communication protocol where

the mechanisms have to be placed. The security architecture in each system could be different and the protocol in such hybrid networks is required to provide intersystem security

In addition to answering such questions, issues related to the wireless devices need to be addressed. Limited resources in the wireless devices put forth certain tradeoffs that need to be considered for energy efficient secure communication systems. Generally, higher security is achieved by using larger key sizes and stronger encryption algorithm. The higher security algorithm comes at the cost of increased computational time and energy consumption. However the battery power available on the wireless devices is limited. Increasing the level of security would reduce the operation time of the device. The implications of providing security at different layers of the protocol would result in different delays.

It is thus essential to know the performance of the encryption schemes in terms of energy consumption for various options like changing key sizes, modifying the number of rounds, layers of security, amount of data processed per packet, and algorithms that can be used on the wireless devices before designing a secure wireless communication protocol. Knowledge of the tradeoffs would also help in designing systems that can adapt the security of the communication link based on the device being used and the battery power left on it. The harsh wireless environment further complicates the trade-off. There has not been any research that studies the tradeoffs between security of wireless devices and the battery consumption of the algorithms. However, there have been some studies about the energy efficiency of wireless devices and about encryption algorithms, which have been summarized below.

Some of the studies have been related to strategies for energy efficiency like reducing brightness of the monitor, adaptively switching on and off the wireless interface, and implementation on customized hardware. Energy efficient wireless networking for multimedia applications [13] by Paul J.M. Havinga and J.M. Smit talks about adapting quality of service of the system based on the dynamic wireless environment.

In the past, some researchers have concentrated on optimizing the implementation of encryption schemes for specific devices or customized hardware. Field Programmable Gate Array (FPGA)

implementation of Rijndael was studied by McLoone, W. and McCanny, J.V. [14]. Riaz, M. and Heys, H.M. [15] have studied implementation of RC6 and CAST-256 encryption schemes on FPGA. Researchers have considered code optimization for more productive implementation of the encryption schemes. Xinmiao and Parhi [16] have considered optimizing the implementation of AES algorithm. Some researchers have explored optimizing the public key encryption schemes for wireless clients [17]. However, employing platform specific optimization of the software leads to high increase in the complexity and cost of the effort. Also, the program analysis is too expensive in many cases. Thus most developers tend to use libraries that are optimized to a satisfactory level and the function of optimized generation is implemented on the compiler.

There have been studies that compare the performance of some of the encryption and decryption schemes in terms of bytes processed per unit time or time for operation [18,19]. However, there haven't been any studies related to the energy consumed by the encryption schemes in every day communication environment on wireless devices. Also it has been observed that transmissions consume much more power than computations. Hence it is essential to evaluate the energy consumption trade-offs with and without transmission of data. This research work is a directed study of the battery consumption of the encryption schemes in practical application environment.

1.2 Scope of Research

The thesis concentrates on evaluating the performance of encryption schemes in terms of the energy consumed when implemented at the application layer through standard encryption libraries on wireless devices. The goal is to aid the design of energy efficient secure communication schemes for the wireless environment in the future. The research work has been divided into following tasks to achieve this goal. First, gain knowledge and understanding of security of information systems. Secondly, study the performance and energy consumption of popular symmetric key schemes AES, CAST and IDEA and asymmetric schemes RSA, ElGamal, and ECIES. Third, study the effect of changing key size for AES and RSA. Fourth, study the effect of encryption and key size variation with transmission of data. Fifth, study the

relationship between encryption at the link layer and at the application layer. Finally, study the effect of signal to noise variation and packet size variation in wireless communications.

Optimization of implementations for specific devices and hardware implementation of schemes are beyond the scope of the thesis. This research also does not provide any specific design optimized for the wireless environment and this task is left to the discretion of the systems engineer.

1.3 Thesis Outline

The research focuses on the energy consumption characteristics of various encryption schemes under varying environmental condition in various devices such as laptops and palm-sized computers. The next four chapters of the thesis have been organized in the following order. Chapter 2 covers the literature related to the thesis. It covers different encryption schemes from secret key to asymmetric key systems and their security. It also covers topics like wireless security, IP Security, and battery technology. Chapter 3 explains the experiment design. It explains how choices were made for the experiments and the measurements were taken. Chapter 4 explains the results obtained during the research work and provides some analysis of the results. Finally, Chapter 5 presents the summary of the results and literature and provides pointers for future work.

Chapter 2
Literature Review

This chapter presents the theoretical background essential for the thesis. Section 2.2 explains the encryption schemes briefly. It explains symmetric and asymmetric keys schemes. It also provides details of the popularly used symmetric and asymmetric encryption schemes. In Section 2.1 the significance of security and the mechanisms used to achieve security are explained. Section 2.3 provides the details of Wired Equivalent Privacy and its insecurities. Sections 2.4 and 2.5 explain IP Security and Secure Socket Layer respectively. Different ways of achieving energy efficiency are summarized in section 2.6. Section 2.7 provides an overview of the battery technology currently in use.

2.1 Network Security

Measures are taken in an organization to secure its data from attackers. The measures taken are generally not as simple as they appear to be. In developing the security of the system the designer has to look at the possible ways in which the systems security mechanisms would fail. The design of such systems also needs to consider where to place the mechanism both physically and logically. As explained in Chapter 1 the security services can be classified into the following: confidentiality, authentication, integrity, non-repudiation, access control, and availability [3].

Security services are designed to prevent attacks that compromise the security policy of the organization. The attacks may be passive attacks or active attacks. In the case of passive attacks the attacker monitors the network connections. By way of monitoring the connections the attacker can get the private information of the organization and can do traffic analysis in case the content of the message cannot be decoded that easily. In active attacks the attacker modifies the communication in some ways to his advantage. Masquerade involves the attacker assuming the

identity of someone else. Masquerade is thus an attack on the authentication service. Replay attacks involve the replay of information from previous valid connections. The replay attacks can be extended further by means of modification of the information. The information content is so modified that it appears to be from a legitimate source. Denial of service attack prevents or inhibits the normal use of the communications services. It involves disruption in the flow of information either by disabling the network resources of overloading them with meaning less data.

2.1.1 Confidentiality

Confidentiality is intended to prevent passive attacks. To make the information confidential, the data is modified in such a way that it would be infeasible for the attacker to guess the data. It is achieved by means of encryption algorithms. Encryption is done based on shared secret information between communicating parties. Only the receiver and in some cases the sender know how to decrypt the data after it has been encrypted. The data is generally encrypted with an encryption key and can be decrypted by using a decryption key. For a symmetric key scheme, the encryption and the decryption keys are the same. For public key schemes, they are different. The key used for encryption is called public key while the key for decryption is called the private key. Encryption is further explained in the next section. Confidentiality in some cases also requires hiding the process of communication itself to avoid traffic flow analysis. Raju Ramaswamy [20] explains different ways in which traffic flow analysis can be achieved at various levels of the OSI layer.

2.1.2 Authentication

In authentication services, it is required that a pair of communicating entities establishes its identity. Essentially, the authentication service tries to establish the identity by means of making sure that a secret is shared between the involved entities. Some protocols establish the authentication through the means of symmetric key schemes while others establish it through the means of public key schemes. For the users of a symmetric key authentication system the communication systems share a secret key between the two communicating parties. Authentication is generally achieved based on challenge and response procedure. Lets say a user

A wants to authenticate user *B*. *A* would send a random number to *B* which *B* would encrypt by using the shared information and sends it back to *A*. *A* would authenticate user *B* by ensuring that the random number decrypted is actually the number it had sent to *B*. Obviously this is a very naïve authentication scheme and is susceptible to various attacks including man in the middle attacks.

In the man in the middle attack, there is a middleman lets say *C* who poses to be *B* and receives the random number from *A*. It then sends this information to *B* posing to *B* as *A*. *B* encrypts the information with the shared secret and sends it back to *C* and *C* relays it back to *A*. *A* thinks that the entity it is communicating with is *B*. This is clearly a situation where the authentication service has failed. A popularly used symmetric key authentication scheme is Kerberos. Kerberos was a part of the Athena Project at MIT [3, 21]. Rather than building an elaborate authentication for each communicating entity, Kerberos makes use of a centralized authentication server. Let us assume the authenticating server is *S*. *TSs* and *TSa* are timestamps by user A and server *S*. $\{M\}_K$ represents encryption of message *M* by key *K*. *Kas* is the key shared between *A* and *S*, *Kbs* between *B* and *S* and *Kab* between *A* and *B*. ID_A and ID_B represent the identities of *A* and *B* which could be their IP addresses or MAC addresses. The concept behind the Kerberos scheme is explained below in an over simplified form, although the actual scheme is much more complicated:

1. A sends S; ID_A, ID_B
2. S sends A; $\{TSs, Lifetime, Kab, ID_B, \{TSs, Lifetime, Kab, ID_A\}_{Kbs}\}_{Kas}$
3. A sends B; $\{TSs, Lifetime, Kab, ID_A\}_{Kbs}, \{ID_A, TSa\}_{Kab}$
4. B sends A; $\{Tsa + 1\}_{Kab}$

The use of timestamps assures the freshness of the messages exchanged. Even if a man in the middle was to pose as any of the entities *A*, *B*, or *S* he would not be able to authenticate himself without possessing either *Kas* or *Kbs*. Further the use of messages 3 and 4 leads to mutual authentication of *A* and *B*. Only *B* can decrypt $\{TSs, Lifetime, Kab, ID_A\}_{Kbs}$ since the key *Kbs* is known just to the server and *B*. Since *B* knows *TSa* it has *Kab* that was decrypted, assuring that it is *B* at the other end. *B* is sure its communicating with *A* because $\{TSs, Lifetime, Kab, ID_A\}_{Kbs}$ has the identity of *A* and servers timestamp to assure it freshness.

Public Key schemes are slightly different from the symmetric key authentication systems. X.509 [3,22] is one such public key authentication scheme. Here the sender *A* can encrypt the data with a public key of *B* and sends it to *B*. *B* decrypts the message and sends it back to *A* after applying some transformation. The use of public key schemes allows us to get rid of the dependency on the authentication server. However, the assurance of the public key of *B* needs to be established. The assurance of the public key is established by means of a certificate. In a certificate, trusted authorities endorse the validity of the user certificate by means of digital signatures. Digital signatures are explained in the section 2.1.4.

2.1.3 Integrity

For data integrity, assurance is needed that only legitimate entities can modify the message. Encrypting the message to some extent ensures that the attacker cannot modify the message. However there is a possibility of some malicious user sending random data to the receiver. The receiver would decrypt these messages to some incomprehensible data, which posses the possibility of some damage. One method of avoiding such situations is to add a checksum to the message before encrypting it. If the decrypted message and the checksum match then the received message can be assumed valid otherwise it is considered invalid. Such a scheme would provide authentication and confidentiality along with message integrity.

A variation to the use of checksums is the use of encrypted hash functions. A hash function takes a variable length of message, *M*, and produces a hash code, *H(M)*, of fixed size. The hash code closely depends on the message. Small changes in the message result in a completely different hash code. The hash codes are designed to have high collision resistance. This implies that given *H(M)* it is computationally infeasible to produce *M* or *H(M')* where *M'* represents some other message. The popular hash functions are MD5 and SHA [3].

2.1.4 Non repudiation

Non-repudiation involves the ability to prove to someone, the source of the document [22]. The originator then cannot deny that he is the author of the document. In this sense non-repudiation involves both authentication and integrity. Symmetric keys are however inadequate in providing absolute non-repudiation even though they can provide authentication and integrity. The sender generates a hash code over the data and transfers it along with the data to the receiver. The receiver is able to check for the integrity of the document and it can authenticate the sender. However, the receiver cannot deductively assert that the data was sent but the sender and that it was not modified by the receiver since the receiver also possesses the same key and can generate the same hash and encrypt it. A reliable signature scheme with symmetric key scheme would require a trusted authority that can sign the document and check the document signatures.

With asymmetric key schemes non-repudiation can be achieved much more elegantly. Only the owner of the key knows the private key while any one can use the public key. Exploiting this fact has led to the evolution of the concept of digital signatures. Digital signatures allow the sender to generate a unique signature on the message that can only be generated by the owner of the private key. Everyone else including the receiver can verify the owner by using the public key but it is computationally infeasible for receiver to produce a similar signature for any other message.

The digital signatures make use of public key encryption and secure hash functions. A secure hash algorithm produces a hash value of the message. The hash value is then encrypted by the private key of the user using the public key algorithm. At the receiving end the receiver uses the public key to decrypt the hash value. It also generates a hash value from the message it has just received. If the hash value generated by the receiver and the received hash values match, the message is authenticated.

2.2 Encryption Algorithms

Encryption forms the basic building block for various security services [3]. There are two types of cryptosystems: secret key and public key systems. In secret key schemes the same key is used for encryption as well as decryption. Most of the popular secret key algorithms are based on the Feistel Cipher Structure [3]. Encryption schemes like DES, IDEA, CAST, and AES use different kinds of transformation and rounds to achieve confusion and diffusion. In diffusion the statistical structure of the plaintext is dissipated into long-range statistics of ciphertext. Confusion on the other hand seeks to make the relationship between the ciphertext and the key as complex as possible. Although each one provides different mechanisms for encrypting data the basic security provided by the algorithm in today's context is decided by the brute force attack and is directly related to the key size.

Public key systems provide a radical departure from the secret key schemes. The public key scheme offered an elegant solution to the key distribution and authentication problems while using secret key mechanisms. Public key schemes are asymmetric involving the use of different keys for encryption and decryption process. They use mathematical functions known as the trap-door functions to achieve encryption. The trap-door functions are based on some difficult mathematical problem. The IEEE 1363 [23] document recognizes three distinct families of problems upon which the asymmetric key schemes can be based: integer factorization (IF), discrete logarithms (DL) and elliptic curves (EC). The popular schemes that use these methods are RSA, ElGamal and ECC respectively. Due to the computational cost of these public key schemes, they are generally used in conjunction with secret key schemes for secure data communications.

2.2.1 Symmetric Key Cryptosystems

The symmetric key algorithms also known as the conventional or one-key algorithms have the same key for encryption and decryption. For the communication between a sender and a receiver to remain secret the key should be kept secret. It can be denoted as:

$E_k(M) = C$

$D_k(C) = M$

where

E: Encryption function

D: Decryption function

M: Message

C: Cipher

k: Shared key

Symmetric Key algorithms can be classified into two categories: stream cipher that operates on a single bit at a time and block ciphers that operate on group of *n* bits at a time.

2.2.1.1 Stream Cipher

In stream cipher a bit stream, which is pseudo-random in behavior, is generated from the key and is XORed with a stream of plaintext to produce the ciphertext stream [24]. The receiving end produces the same bit stream that is XORed with the ciphertext to recover the plaintext

Stream Cipher encryption can be represented as:

$c_i = p_i \oplus k_i$

where

c_i: ith ciphertext bit

p_i: ith plaintext bit

k_i: ith key bit

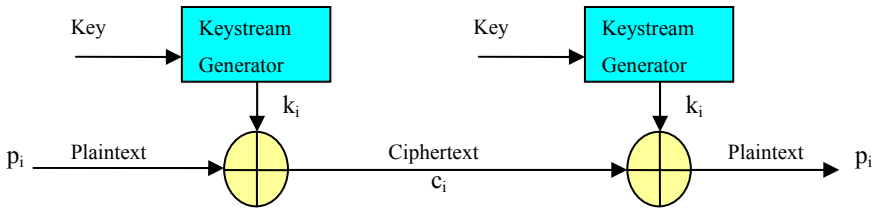


Figure 1: Stream Cipher

Security of a stream cipher depends on the period of the bit stream produced by the keystream generator. Small periods lead to an insecure XOR operation. If the keystream generation algorithm produces an endless bit stream we would have perfect security. In reality the period lies between the two extremes.

2.2.1.2 Block Cipher

A block cipher operates on plaintext block of n bits to produce ciphertext block of *n* bits. In substitution cipher there is an input to output mapping of plaintext and cipher text. The substitution cipher for small block size is however vulnerable to statistical analysis of the plaintext. A large block size would make the statistical characteristics of the result infeasible for cryptanalysis but is not practical. The key for such a scheme is the substitution itself hence for plaintext size of *n* bits the key size would be $n*2^n$ bits [3].

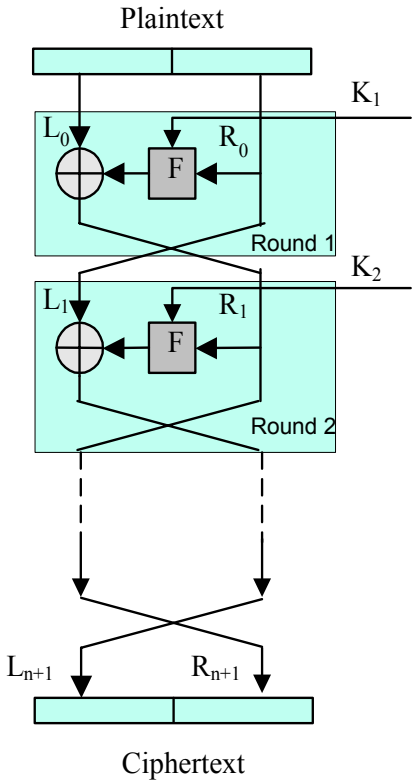


Figure 2: Feistel Cipher Scheme

Feistel proposed the concept of product cipher where two or more basic cipher functions are used sequentially such that the final product is cryptographically stronger than any of the basic ciphers. Feistel proposal of a cipher that alternates substitution and permutations was actually an implementation of Claude Shannon's proposal to develop a product cipher that alternates confusion and diffusion functions. Shannon introduced the diffusion and confusion techniques to thwart statistical analysis on cipher text. The Feistel Ciphers achieve a reversible mapping between the plaintext and the cipher text based on a key by diffusion and confusion functions.

In Feistel Cipher scheme the plaintext is split into two halves L_0 and R_0. The two halves pass through rounds of transformation and then combined to produce the ciphertext. Each round uses the output from the previous round and a sub key K_i derived from the main key K. Substitution is performed by round function F on the right half R_i of the data and then it is XORed with the left half L_i data. Interchanging the left and right halves of data performs permutation.

The process of decryption is essentially the same as encryption. Here the cipher text is the input the keys are used in reverse order and the output is the plaintext.

2.2.1.2.1 CAST ENCRYPTION

Carlisle Adams and Stanford Tavares designed the CAST-128 encryption scheme [3, 24]. CAST-128 has been published as RFC 2144 in May 1997. CAST has a classical Feistel network with 16 rounds and operates on 64 bits of plaintext to produce 64 bits of cipher text [3]. CAST makes use of keys that can vary from 40 bits to 128 bits. CAST Encryption scheme employs two subkeys 32 bit Km_i and 5 bit Kr_i in each round derived from the key. For key sizes less than 80 bits there are 12 rounds and for key size greater than 80 bits there are 16 rounds. Decryption is essentially the same with the key employed in reverse order. The structure of the *F* function used by CAST encryption scheme is given below.

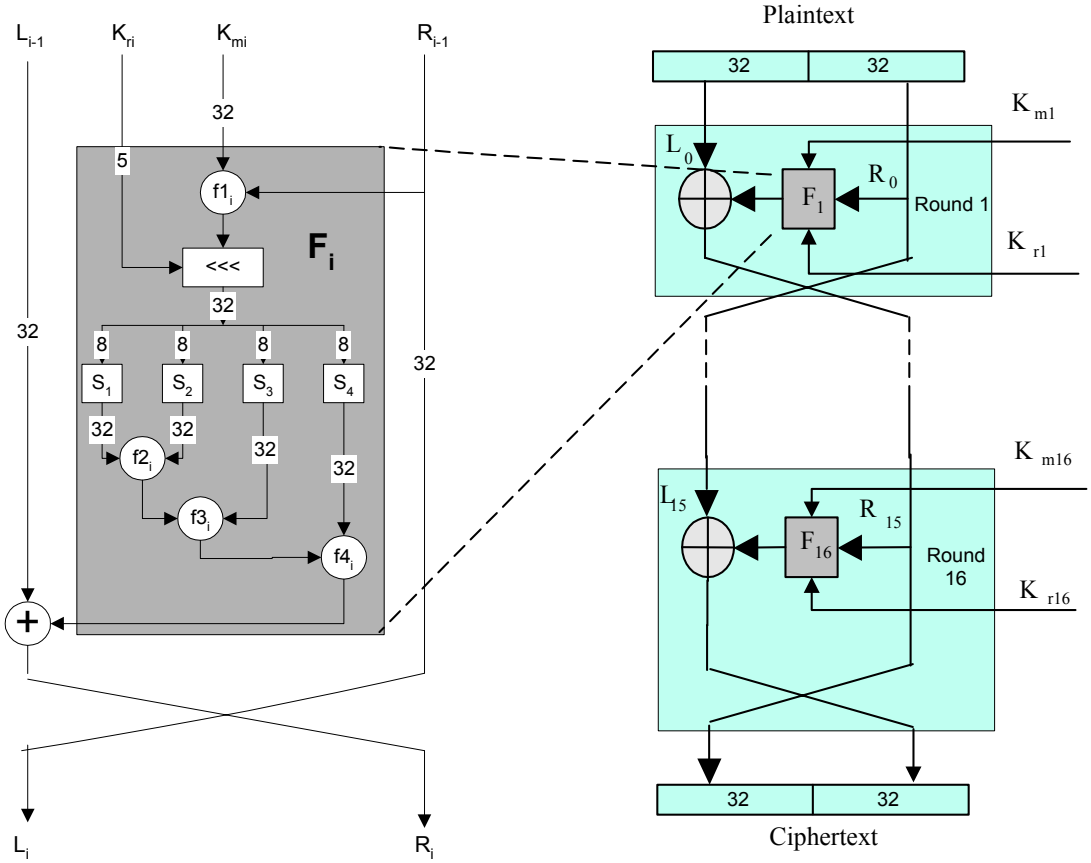


Figure 3: CAST-128 Encryption Scheme

Round (i)	$f1_i$	$f2_i$	$f3_i$	$f4_i$
1,4,7,10,13,16	+	⊕	-	+
2,5,8,11,14	⊕	-	+	⊕
3,6,9,12,15	-	+	⊕	-

Table 1: Functions f1, f2, f3, and f4 in CAST based on rounds

Here <<< represents circular left shift operation based on the value of Kr_i. +,-, and ⊕ represent modulo 32 addition, subtraction and XOR. The 32-bit value obtained after the circular shift operation is then split into 4 8-bit inputs to *S*-boxes *S1, S2, S3,* and *S4*. The *S*-boxes take 8 bits input and produce 32 bit outputs. Functions *f1*, *f2*, *f3*, and *f4* play different roles in different rounds as listed by table 1

There is no multiplication operation in CAST. The performance of CAST can be approximately summarized by the following equation.

$$\text{Processing time} = 21 * t_s + 22 * t_a + 21 * t_{xor} + 16 * t_{lshift}$$

Here t_s is time for subtraction, t_a time for addition, t_{xor} is time for XOR operation and t_{lshift} is the average time for circular left shift operation.

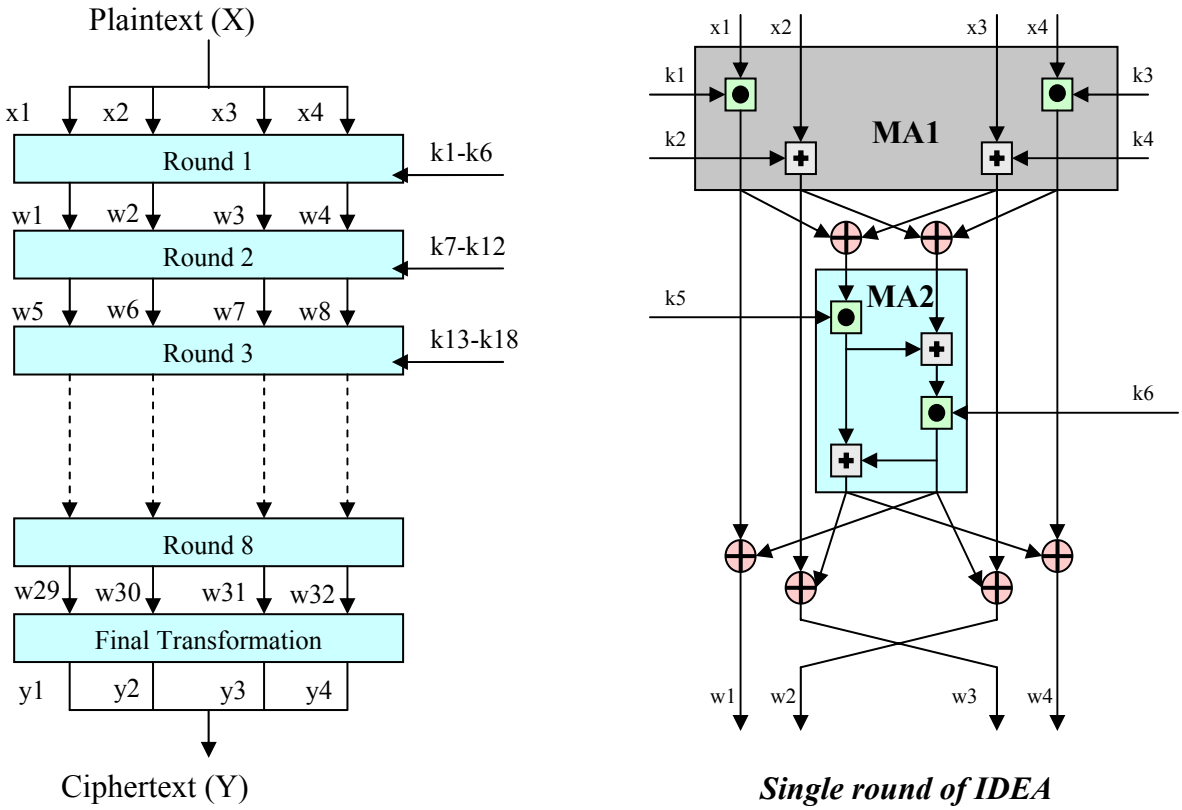
While implementing CAST the addition and subtraction operations can be easily achieved by using word size of 32 bits. Thus it can be considered equivalent to single operation. The shift operation may be available in processors like Pentium III and Celeron belonging to the IA-32 Intel® Architecture have these functions as single operations although the execution time may extend over several timing cycles. The *S*-boxes are implemented as two-dimensional arrays and the input data maps to columns and rows of the array.

2.2.1.2.2 IDEA ENCRYPTION

Xuejia and James Massey proposed International Data Encryption Algorithm (IDEA) encryption scheme in 1990 [3]. IDEA was designed as a stronger encryption scheme to replace the then existing DES encryption scheme. The IDEA encryption scheme consists of eight identical rounds of processing on the blocks and then a final transformation function.

The inputs to the scheme are 64-bit plaintext and 128-bit encryption key. The algorithm divides the input plaintext into four 16-bit blocks. Each round makes use of six 16-bit sub-keys to process the four 16-bit plaintext blocks and produces four 16-bit blocks. The final transformation round uses four sub-keys and has only the gray portion shown for a single round.

In the *MA1* blocak, two of the four 16-bit blocks input to the rounds undergo modulo addition with two sub-keys and other two undergo modulo multiplication. The four intermediate 16-bits blocks are then combined to produced two 16 bit blocks which go as inputs to *MA2* structure.The *MA2* structure has uses two keys to transform these two blocks using addition and multiplication operations and produces two 16-bit ouput blocks. These two blocks are combined with the intermediate four blocks using XOR function to produce the round output. In every round the second and the third 16-bit blocks are switched at the ouput to make the algorithm more resistant to cryptanalysis. The final transformation consists of only the *MA1* block of the rounds. This is done to make the algorithm symmetric so that the same code can be used for encryption and decryption with the keys applied in reverse order.



Basic Structure of IDEA

Single round of IDEA

Figure 4: IDEA Encryption Scheme

Subkeys *k1-k8* are directly taken from the 128 bit key with *k1* being the first 16 bits to *k8* being the last 16 bits of the key. Then a 25 bit circular left shift is performed on the key and next 8 sub-keys are generated. The procedure is repeated till 52 sub-keys are generated

The process of decryption is essentially the same as encryption. The only difference is the input to decryption is the ciphertext and the sub-key generation is slightly changed. The first four keys are generated from the keys that were input to the transformation phase. The first and the fourth subkeys are multiplicative inverse of *k49* and *k52* while second and third subkeys are additive inverse of *k50* and *k51*. For every other round the subkeys derived are multiplicative and additive inverse of the corresponding 10-*i* round

Computationally the most intensive operation in IDEA is the multiplication. Every round requires 4 16-bit multiplication operations in addition to 4 addition and 6 XOR operations.

$$\text{Processing time} = 34 * t_m + 34 * t_a + 48 * t_{xor}$$

Here t_m is time for single multiplication; t_a time for addition; and t_{xor} is time for XOR operation. For processors not capable of 32-bit multiplication the multiplication step can be improved by building a log table for the multiplication but it would take a lot of memory.

2.2.1.2.3 AES ENCRYPTION

Rijndael algorithm designed by Joan Daemen and Vincent Rijmen was selected on October 2, 2000 by NIST as AES standard [25] to replace the previous DES scheme for symmetric key encryption. Although the original submission had provisions for variable length input blocks, AES takes 128-bits plaintext input and produces 128-bit cipher text output. The scheme can have three key sizes 128, 192 and 256. The key scheduling algorithm of AES takes the cipher key and produced *4 * (Nr + 1)* words, where *Nr* represents number of rounds. These words are called the key schedule words. The plaintext goes through 10 rounds for 128-bit key, 12 rounds for 192-bit key and 14 rounds for 256-bit key. All except for the last round are identical. Figure 5 below depicts the AES Encryption Scheme for 128 bits key.

In AES, the state word represents the two-dimensional array of bytes to be processed by the round. The bytes of the block are arranged in a two dimensional array of bytes. Since for AES the block size is fixed the bytes are arranged in two-dimensional array of four bytes and four columns. A 32-bit word maps to the columns of the state, where the most significant byte maps to the first row and the least significant to the fourth row.

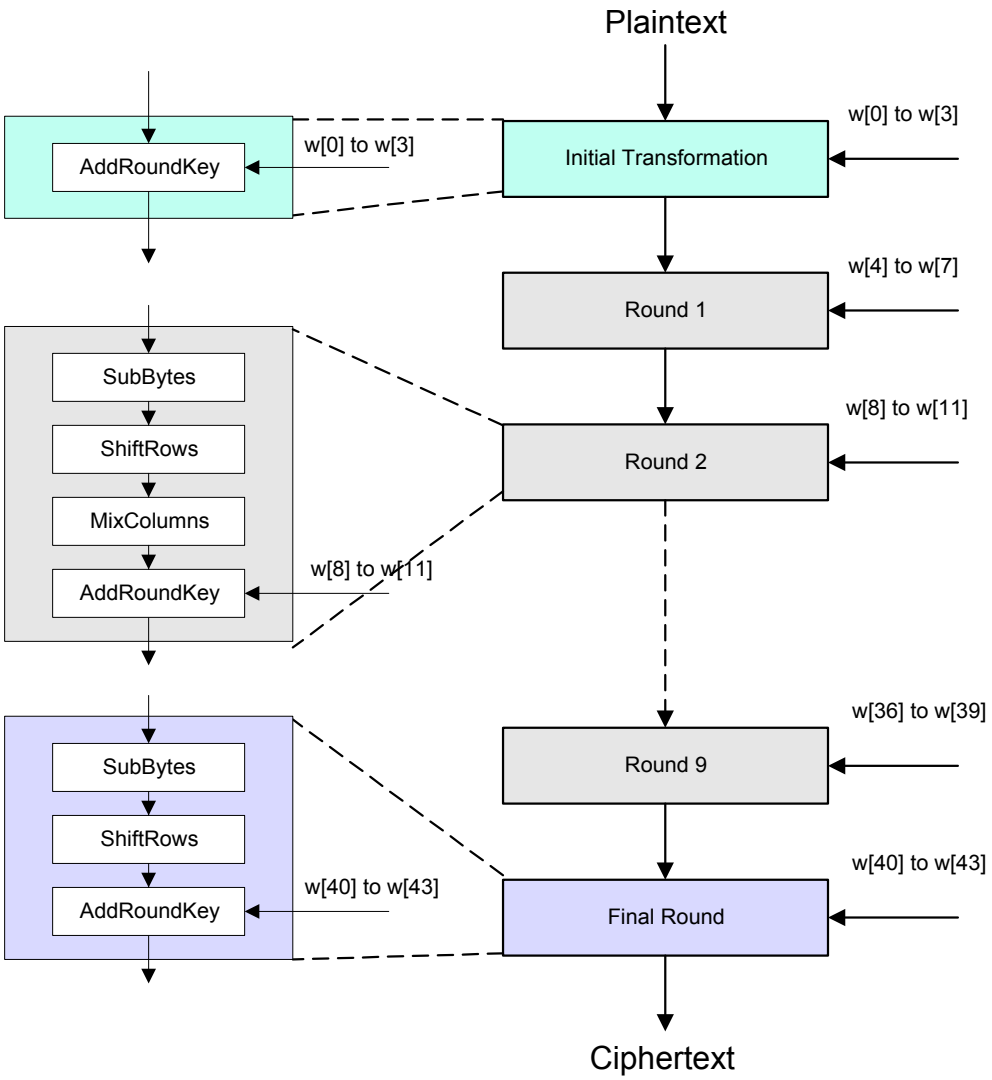


Figure 5: AES Encryption Scheme with 128 bits key

The addition and multiplication operations in AES are slightly different from the conventional operations. The bytes are treated as polynomial where the bits represent the coefficients of the polynomial. Addition represents modulo 2 additions i.e., the XOR function. The multiplication is performed by polynomial multiplication modulo an irreducible polynomial of degree 8. For AES the irreducible polynomial is represented as:

$$m(x) = x^8 + x^4 + x^3 + x + 1$$

The modular reduction ensures that the result will be a binary polynomial of degree less than 8 and can be represented as a byte.

AES also makes use of four term polynomials of the form:

$$a(x) = a_3x^3 + a_2x^2 + a_1x^1 + a_0$$

Note that the polynomial here is different from the one before as the coefficients represent bytes of a word rather than bits of bytes. The multiplication of four-term polynomial is achieved but the addition and multiplication operations explained above and then reduce the result modulo a polynomial of degree 4. The polynomial for AES is $x^4 + 1$. The polynomial $x^4 + 1$ is not an irreducible polynomial in $GF(2^8)$; however, AES specifies fixed four term polynomial that do have and inverse.

The AES scheme employs 4 fundamental functions to achieve confusion and diffusion of data. The functions are SubBytes, ShiftRows, MixColumns, and AddRoundKey. SubBytes is substitution table (*S*-box) transformation of the state. The *S*-box, which is invertible, operates independently on each byte of the state. In ShiftRows the rows of the state are cyclically shifted by *r-1* bytes, where *r* represents the row number. Thus the first row is not shifted at all; the second row is shifted by one byte; the third by two bytes and the fourth by 3 bytes.

In MixColumns the columns of the state are considered are four term polynomials and multiplied modulo $x^4 + 1$ with the fixed polynomial $a(x)$ given as:

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$$

In AddRoundKey the columns of the state are treated as words and are XORed with the key schedule words. Since it is an XOR operation AddRoundKey function is inverse of itself when applied again to the transformed word. For AES the function can be represented as:

$$[s'_{0,c}, s'_{1,c}, s'_{2,c}, s'_{3,c}] = [s_{0,c}, s_{1,c}, s_{2,c}, s_{3,c}] \oplus [w_{round*4+c}]$$

The decryption process is similar to encryption. Here inverse functions InvSubBytes, InvShiftRows, and InvMixColumns are used in place SubBytes, ShiftRows, MixColumns respectively. The InvShiftRows operated before InvSubBytes and AddRoundKey before InvMixColumns in the rounds. This assures that the order of operations applied during encryption process is applied in reverse.

The proposal for Rijndael [26] provides details on the implementation of AES for a 32-bit processor. Accordingly different steps of the round can be combined in a single set of table lookups, allowing for very fast implementation. Each cell of the state can be separated and treated differently. Accordingly the transformations in the rounds can be expressed as

$$e_j = T_0[s_{0,j}] \oplus T_1[s_{1,j-1}] \oplus T_2[s_{2,j-2}] \oplus T_3[s_{3,j-3}] \oplus w_{round*4+c}$$

Here T_0 to T_3 represent look up tables and e_j represents the jth column of the output state. Other notations represent the same terms as explained above. Thus the AES implementation can be done by means of about 12 rotate byte operations and 16 XOR operations per round. Processing time can be summarized as follows:

Processing time for AES 128 bit key = 120 $t_{rotByte}$ + 164 t_{xor}
Processing time for AES 192 bit key = 144 $t_{rotByte}$ + 196 t_{xor}
Processing time for AES 256 bit key = 168 $t_{rotByte}$ + 226 t_{xor}

Here $t_{rotByte}$ is time for single rotate byte operation and t_{xor} is time for XOR operation.

2.2.2 Asymmetric Key Cryptosystems

Public key Schemes are called asymmetric key systems because they use two separate keys: one for encryption the data and other for decrypting it. The public schemes help a long way in solving the key distribution problem of Conventional or Symmetric key systems. In symmetric

key systems, the same key must be shared by the sender and the receiver and must be protected from others. Also, frequent changes in the key are advised to limit the amount of the data that would be compromised in case a key is revealed. Hence the security of the system is highly restricted by the security of the key distribution scheme. For Conventional scheme the possible way of key distribution is through physical transmission of the key or through the use of a key distribution center. The problem is further magnified by the fact that each connection should have a separate set of keys. This means that if there are *n* hosts communicating with each other than they would require *n(n-1)/2* keys for independent secure communication between hosts.

Public key schemes solve this problem. The structure of public key systems allows two different keys for encryption and decryption. Hence for a sender to communicate with a receiver, the sender uses the public key of the receiver to encrypt the data being sent. Only the receiver has the key to decrypt the data. Thus we no loner need *n(n-1)/2* key between n hosts and the number of keys required comes down to n. The receiver can publish his public key openly and any sender can use it to send secure data to the receiver. The receiver can at any time change its private key and publish the corresponding new public key in place of the old one to maintain the security of the system.

The question that comes to mind with this solution is why then should we use the conventional encryption systems. The reason why the conventional schemes are still very popular is because the public key cryptosystems due to their nature are very slow in encrypting and decrypting data. In fact, they are so slow that most of the modern day systems make use of a combinational scheme, which makes use of the public key schemes to exchange the secret key for the symmetric key scheme, which is actually used to transfer secure data.

A public key crypto system relies on Non-deterministic Polynomial-Time algorithms. During encryption the message is converted from an easy instance to difficult instance through encryption key. Decryption converts the difficult instance to an easy instance by using the decryption key. The algorithms are so designed, given a difficult instance, the only way to find the easy instance without the decryption key can be best evaluated through non-deterministic methods. A non-deterministic algorithm involved guessing the solution and then verifying that

solution to be correct. There is no known polynomial time solution for the problems and it is assumed that a polynomial time solution doesn't exist. In other words it is assumed that there is no deterministic solution to the difficult problem. Such non-deterministic problems can only be solved in exponential time. The RSA, ElGamal and elliptic curve cryptography systems' way of using these mechanisms respectively have been explained below.

2.2.2.1 RSA

The Rivest-Shamir-Adleman (RSA) scheme is a block cipher scheme in which the plaintext and the ciphertext are integer between 0 and *n-1* for some n. RSA scheme is developed using the exponentiation. In RSA, the keys are generated by selecting two large prime numbers *p* and *q* and their product is calculated as:

$$n = p * q$$

We calculate then Ø(n) as:

$$Ø(n) = (p-1) * (q-1)$$

Ø(n) is also called the Euler's totient function [3], which is the number of integers less than *n* and relatively prime to *n*. Next a number, *e*, is selected is selected that is relatively prime to *Ø(n)* i.e., GCD of *e* and *Ø(n)* is 1. Then the multiplicative inverse of *e* is calculated using the Euclidean algorithm such that:

$$e*d = 1 \bmod Ø(n) \qquad \text{or} \qquad d = e^{-1} \bmod Ø(n)$$

The numbers *e*, *n* form the public key and *d*, *n* forms the private key. *p*, *q* and *Ø(n)* are discarded but never revealed. Size of *n* refers to the size of the key in RSA. *e* is generally chosen to be small, as the value of *e* is not known to affect the security of the scheme when proper encoding schemes are used [23]. Typical values of *e* in use are *3*, *17* and $2^{16}+1$. Although no particular

attack with the context of RSA with Optimal Asymmetric Encryption Padding (OAEP) [27] has been detected with the use of *e=3* the more conservative users prefer using public exponents larger than 3. For more details on the security of short keys in RSA refer [28]. The RSA problem can be solved fastest by the integer factorization method. For current status of factorizing refer [29]. Moreover systems can be designed to have a constant value for *e* so that *e* need not be transmitted. Size of *d* is approximately the same as the size of *n*.

To encrypt a message, the message is broken into small numbers, M_i, less than n. Let us call these message blocks. These numbers are raised to power *e* modulo *n* to obtain the cipher block, C_i:

$$C_i = M_i^e \bmod n$$

This is assuming the fact that it is mathematically difficult to determine M_i given C_i. To decrypt the cipher blocks are again raised to power *d* mod *n* to obtain the corresponding message blocks. For more details on number theory and Euler's theorem refer Cryptography and Network Security [3] chapter 7.

$$\begin{aligned} C_i^d \bmod n &= (M_i^e)^d \bmod n \\ &= M_i^{ed} \bmod n \\ &= M_i^{1 \bmod \emptyset(n)} \bmod n \\ &= M_i \end{aligned}$$

Normally an encoding scheme is applied before encrypting the message for security purpose. The encoding scheme maps the message to and encoded message with some randomness that can be reversed for decryption of message based on the encoding parameters. The recommended encoding mechanism of RSA is Optimal Asymmetric Encryption Padding (OAEP) [27].

Both encryption and decryption in RSA involve raising an integer to another integer mod n. Since the integers are large numbers, if the exponentiation is done before modulo

operation the size of the intermediate result would be very large. To make it practical to implement the RSA algorithm the following property of modular arithmetic is exploited.

$$(a \bmod n) * (b \bmod n) = (a * b) \bmod n$$

Using this property along with successive multiplication scheme it is possible to compute x^e with less than *(e-1)* multiplications. For example x^{32} can be computed by computing the following intermediate results: $x, x^2, x^4, x^8, x^{16}, x^{32}$. Here the result could be obtained in 5 multiplications instead of 31.

Successive multiplication can be applied to any exponent and can be evaluated on an average in *1.5 * k* multiplications, where *k* is the size in bits of the exponent. For multiplication of to larger prime numbers with odd modulus the Montgomery Algorithm is used [30]. The runtime for the multiplication algorithm is proportional to $O(N^2)$ where *N* is the size in bits of modulus *n*.

Thus total run-time of encryption can be estimated as follows:

$$\text{Processing time for encryption} = 1.5 * k * O(N^2)$$

For decryption operation is made further efficient by using the Chinese Remainder Theorem (CRT) [3]. According to CRT, $M = C_i^d \bmod n$ can be calculated from the residues $Mp = C_i^d \bmod p$ and $Mq = C_i^d \bmod q$ since $n = p*q$. This allows reducing the computational time but reducing the modulus bit size to half. Further reference on CRT can be obtained from [31].

2.2.2.2 ElGamal Encryption Scheme

The ElGamal Scheme [3,18,23,24] is based on discrete logarithm problem proposed by T. ElGamal in 1985. It is based on the Diffie-Hellman scheme. The ElGamal scheme uses randomization hence the same message block can produce different ciphertext block for a given public key.

For generating a key pair, first a large prime number p is chosen along with two other random numbers, *g* the generator and *x*, both less than *p*. Next we calculate *y* as:

$$y = g^x \bmod p$$

The discrete logarithm problem states that given *y*, and *p* it is computationally difficult to determine the value of *x*. The public key is *y*, *g*, and *p*. *x* is the private key. The key size refers to size of *p* in bits. For ElGamal scheme it is possible to use a generator *g* such that the group generated by *g* belongs to the subgroup of the order *r*. For details of groups, prime field and order of group refer [3,23]. For the purpose of our discussion subgroup with order *r* represents the maximum unique modulo *p* numbers that can be generated by successive exponentiation of the generator *g*. *r* is a prime divisor of *p-1* by definitions of subgroup. The private key *x* is chosen to be in the range 1 to *r-1*. The use of subgroup allows the use of short keys for exponentiation in ElGamal improving the overall performance of the scheme. For more details on security of short keys refer [32]. The best know solution to the Discrete logarithm problem is the Generalized Number Field Sieve (GNFS) [23, 33] which has an asymptotic run time of $exp(((1.923 + o(1))\ ln(p)^{1/3}\ (ln(ln(p))^{2/3})$ where *o(1)* is a number that goes to zero as *p* increases. The exponent is thus chosen such that the brute force on the exponent takes more time than GNFS algorithm. The sizes of exponent chosen for security purpose are as follows:

Modulus (in bits)	**Exponent (in bits)**
512	120
1024	164
2048	226

Table 2: Short Exponent size for ElGamal encryption and decryption [18]

To encrypt a message block, the message block is mapped to a number in the range 2 to *p-1*. Then a random number, *k*, in the range 2 to *r-1* is chosen (in case short exponents are not used *k* and are in the range 1 to *p-1*). We then compute the following two values

$$C1_i = g^k \bmod p$$

$$C2_i = (y^k * M_i) \bmod p$$

$C1_i$ and $C2_i$ form the ciphertext. To decrypt the message the following calculation is done:

$$C1_i^{-x} * C2_i = g^{-kx} * (y^k * M_i) \bmod p$$
$$= g^{-kx} * g^{kx} * M_i \bmod p$$
$$= M_i$$

The processing time for ElGamal for RSA is about the same equation as for RSA. However, in case of ElGamal it is possible to do pre-computation of values of the intermediate results: $x, x^2, x^4, x^8, x^{16}, x^{32}$ for generator g and public key y for the successive multiplication operation based on k to speed up the process of encryption. This can be done because the value of the base i.e., g and y remain the same.

2.2.2.3 Elliptic Curve Cryptography (ECC)

The standardized elliptic curve cryptography scheme considered in this document is Elliptic Curve Integrated Encryption Scheme (ECIES). It is also Diffie-Hellman based scheme. The complete standard is specified in the IEEE P1363a [23] draft. The underlying scheme is similar to DHAES scheme [34] and makes use of basic cryptographic structures like elliptic curve arithmetic along with symmetric key schemes, message authentication code, and cryptographic hash functions to achieve a hybrid asymmetric encryption scheme.

In this document it is intended to introduce the cryptosystem. The entire mathematical background of elliptic curve cryptosystems is beyond the scope of this document. Elliptic curves are not ellipses but are called so because they are described by cubic equations similar to those used for calculating the circumference of an ellipse. Elliptic curves of use in cryptosystems are normally of the form:

$$y^2 = (x^3 + a*x + b) \bmod p \quad ; \text{where } (4a^3 + 27b^2) \neq 0$$

Here *a* and *b* are all real numbers and *p* is large prime number. For elliptic curves we are interested in points in the first quadrant from (0, 0) upto (p-1, p-1) that satisfy the mod *p* equation. Elliptic curves also include a point, *O*, called the identity point or zero point. The number of point on the curve is called the order of the curve and is represented as *#E*. A point, *G*, is selected on the curve which is the generator point. Let the order of this point be *r*. Order of a point is different from the order of the curve. The order is the minimum integer number with which the point has to be multiplied to obtain *O*, in other words $rG = O$. The curve parameters are thus completely defined by *a*, *b*, *p*, *r*, and *G*. For further details on determining the curve parameters refer the Appendix in IEEE P1363a [23].

The properties of importance for the elliptic curves [3] used are:

1. Point *O* serves as an additive identity such that $O = -O$. If P is a point on the curve then $P + O = P$.
2. Negative of a point *P*, *-P*, is a point on the curve with the same *x* coordinate as *P* but negative y coordinate.
3. If $P(x_1,y_1)$ and $Q(x_2,y_2)$ are two points on the curve then $P + Q = R(x_3, y_3)$ can be determined by the following equation

$$x_3 = (\lambda^2 - x_1 - x_2) \bmod p$$
$$y_3 = (\lambda(x_1 - x_3) - y_1) \bmod p$$
$$\text{where } \lambda = (y_2 - y_1)/(x_2 - x_1) \quad \text{for } P \neq Q$$
$$= (3 * x_1^2 + 9)/ (2 * y_1) \qquad \text{for } P = Q$$

4. Multiplication is defined as repeated addition i.e., $4P = P + P + P + P$.

The private key in ECIES is an integer *s* in the range 1 to *r-1* and the public key is $W = sG$. The elliptic curve key pairs are closely related to the curve parameters and can only be used with them.

Step involved in encryption using ECIES are as follows:

1. Select a random number, *u*, in the range 1 to *r-1*

2. Compute an elliptic curve point $C_1 = u * G$
3. Compute an elliptic curve point $P = u * W$
4. Convert the x coordinate of P to bit stream z
5. In case block cipher is to be used,
    a. Use key derivation function (KDF) to get a derived key, K, from z such that length of $K = macLen + encLen$, where *macLen* is the size of the message authentication code (MAC) Key and *encLen* is the size of the symmetric key. Let leftmost *encLen* bits form the secret key, *encKey*, and rightmost *macLen* bits for the *macKey*
    b. Encrypt message M with the key *encKey* to form encrypted message C_2
6. In case block cipher is to be used,
    a. Use KDF to get a derived key, K, from z such that length of $K = macLen + l$, where *macLen* is the size of the MAC Key and l is the size of the message. Let leftmost *encLen* bits form the *encKey* and rightmost *macLen* bits for the *macKey*
    b. Let $C_2 = encKey \oplus M$
7. Apply the MAC function over the encrypted message C_2 to produce authentication tag C_3
8. $(C_1||C_2||C_3)$ form the ciphertext

Decryption process is recovers the plaintext from the cipher text. The inputs to the decryption process are ciphertext $C = C_1||C_2||C_3$, private key s, and curve parameters. Steps involved in decryption process are as follows:

1. Validate Point *C1* belongs to the group with generator G of the order r.
    a. Validate *C1* is a point on the curve defined by the parameter
    b. Compute $C1 * r$ to verify it is an identity point. If not return 'invalid public key' and stop (since order of the subgroup to which G belongs is r, rG equals identity point by definition)
2. Compute the elliptic curve point $s * C1 = s * u * G = P$. If P is identity point return 'invalid public key' and stop
3. Convert the x coordinate of P to bit stream z
4. In case block cipher is to be used,

a. Use key derivation function (KDF) to get a derived key, K, from z such that length of $K = macLen + encLen$, where *macLen* is the size of the message authentication code (MAC) Key and *encLen* is the size of the symmetric key. Let leftmost *encLen* bits form the secret key, *encKey*, and rightmost *macLen* bits for the *macKey*
b. Decrypt message M with the key *encKey* to form encrypted message C_2

5. In case block cipher is to be used,
   a. Use KDF to get a derived key, K, from z such that length of $K = macLen + l$, where *macLen* is the size of the MAC Key and l is the size of the message. Let leftmost *encLen* bits form the *encKey* and rightmost *macLen* bits for the *macKey*
   b. Decrypt message $M = encKey \oplus C_2$
6. Apply the MAC function over the encrypted message C_2 to produce authentication tag T and verify with C_3. If authentication is invalid return 'invalid message' and stop.
7. Output the message M as plaintext

In case of ECIES, as in ElGamal, it is possible to do pre-computation of values of intermediate results: *x, 2x, 4x, 8x, 16x, 32x* for generator G and public key W for the successive addition operation based on u to speed up the process of encryption.

2.2.3 Security of encryption algorithms

Determining the security of an encryption algorithm is perhaps a difficult task to do. In determining whether the algorithm is secure a cryptanalyst studies the mathematical characteristics of the encryption scheme to find shortcut methods of decrypting the ciphertext. Mathematical treatment to find flaws is generally hard and the shortcut solutions to break the schemes may not be discovered. In order to ensure that the algorithm is secure, the algorithms are shared in communities and forums for open public scrutiny.

The symmetric key algorithms described in section 2.1 are considered secure against differential cryptanalysis and linear cryptanalysis. The security of most symmetric key algorithms that have

survived public scrutiny is evaluated in terms of size of the encryption key assuming that the best solution is only to do an exhaustive search on the possible keys. There are no known weaknesses of the algorithms. Daemen [35] however discovered some 2^{51} weak keys in IDEA which if used for encryption could be easily detected and recovered. However the likelihood of these keys being selected out of the 2^{128} keys possible is very low and can be prevented in the implementation of the algorithm itself. IDEA is generally considered very secure and its theoretical basis has been widely and openly discussed since 1990. CAST has been around since 1997 and the Rijndael algorithm used in AES was proposed in 1999. Table 3 below summarizes the characteristics of the symmetric key schemes discussed in this chapter.

Algorithm	Year Published	Key Size (bits)	Plaintext Size (bits)	Number of Rounds	Operations
IDEA	1990	128	64	8	34 mult + 34 add + 48 XOR
CAST	1997	88-128	64	16	21 sub + 22 add + 21 XOR + 16 circular shift
		40-80		12	16 sub + 16 add + 16 XOR + 12 circular shift
AES	1999	128, 192, 256	128	10, 12, 14	120 Rotate Byte + 164XOR, 144 Rotate Byte + 196 XOR, 168 Rotate Byte + 226 XOR

Table 3: Characteristics of Symmetric Key Encryption schemes

For RSA the best-known attack is based on integer factorization problem. If it is possible to factorize in into p and q, then the secret key d and be easily derived. The attack against ElGamal scheme is solving the discrete log problem. The best-known algorithm for factoring and discrete log problem is Generalized Number Field Sieve (GNFS) [23, 33]. The solution to the GNFS algorithm has an asymptotic run time of $exp(((1.923 + o(1))\ ln(n)^{1/3}\ (ln(ln(n))^{2/3})$ where $o(1)$ is a number that goes to zero as n increases and n is the modulus. This means the equation to obtain the same level of security as symmetric curve schemes can be written as follows:

$$k = ((1.923 + o(1))\ ln(n)^{1/3}\ (ln(ln(n))^{2/3} \quad ; k \text{ is equivalent size of symmetric key}$$

For elliptic curves the best attack known is the Pollard-ρ algorithm [36]. The asymptotic runtime of the algorithm is $O(q^{1/2})$ where q is the order of the curve. IEEE 1363 [23] states that the order

of an elliptic curve is approximately equal to the prime *p* used for modulus. This implies size of be *p* should be atleast *2 * k* for security equivalent to *k* bits of symmetric key scheme.

Table 4 below summarizes the relative strength of keys required for different types of algorithms. These are rough estimates taken from [37] based on the best-known solution to the problems on which the encryption is based.

Year	Symmetric Key Scheme (key size in bits)	Asymmetric Key Schemes (RSA/DL) (key size in bits)	Elliptic Curve Cryptography (modulus in bits)
1982	56	512	112
2013	80	1024	160
2055	112	2048	224
2075	128	3072	256
2159	192	7680	384
2242	256	15360	512

Table 4: Key Sizes recommended for Security [37]

It can be seen that the size in bits of RSA/DL schemes increases at a much rapid pace compared to elliptic curve schemes. The increasing key size leads to an increase in the size of the numbers being used in the system and hence the computational cost of operations on them. For the same reason is anticipated that elliptic curves with out perform RSA and ElGamal in the near further. Hence a lot of attention is being paid towards elliptic curve as a viable alternative for public key cryptography.

2.3 Wired Equivalent Privacy (WEP)

In the wired environment only the terminals connected to the LAN can listen to the traffic but in the wireless environment that is not the case. In 802.11b environment [9] defined for Wireless LANs, any station that is 802.11b compliant can hear all the traffic within its range. Thus a

connection without privacy services seriously degrades the security of the system. To provide wireless environment with the level of security of wired LAN, IEEE 802.11 specifies an optional algorithm, Wired Equivalent Privacy (WEP). The encryption of data is restricted between station-to-station communications only. If the privacy is not implements all the messages are unencrypted.

In wireless LANs the authentication service is used by all station to establish their identity with the stations with which they will communicate. IEEE 802.11 offers two types of authentication schemes: Open System and Shared Key System. Open system authentication is a simple authentication algorithm. Here any station that requests to be authenticated from a station with open system authentication gets authentication. It is basically a two-step process. The first step involves identity assertion and the second step involves the authentication result.

Shared key authentication requires WEP privacy mechanisms. Here it is assumed that a key is shared between two stations trying to establish communication. The shared key algorithm is a four-step challenge-response based authentication scheme. The steps are explained as follows:

1. The requesting station asserts its identity to the responder.
2. The responder then sends a challenge text, which is a pseudorandom number of fixed length of 128 octets, to the requester.
3. The requester encrypts the challenge text with the shared key using WEP algorithm and sends it to the responder.
4. The responder validates the message and checks if the decrypted challenge text is the same as it had sent to the responder. The responder replies with an authentication result, which is either successful or unsuccessful based on the previous validation.

The protocol used in Shared Key system is WEP. WEP intends to protect the IEEE 802.11 networks from the eavesdropping. It provides confidentiality, access control and data integrity. WEP algorithm is based on stream cipher scheme of encryption. The plaintext is XORed with a pseudo random sequence of equal length generated by the WEP Key Sequence Algorithm. Running an IEEE 802.11 without WEP leaves the system open and susceptible to significant security risks. The figure 6 below explains the operation of the WEP algorithm.

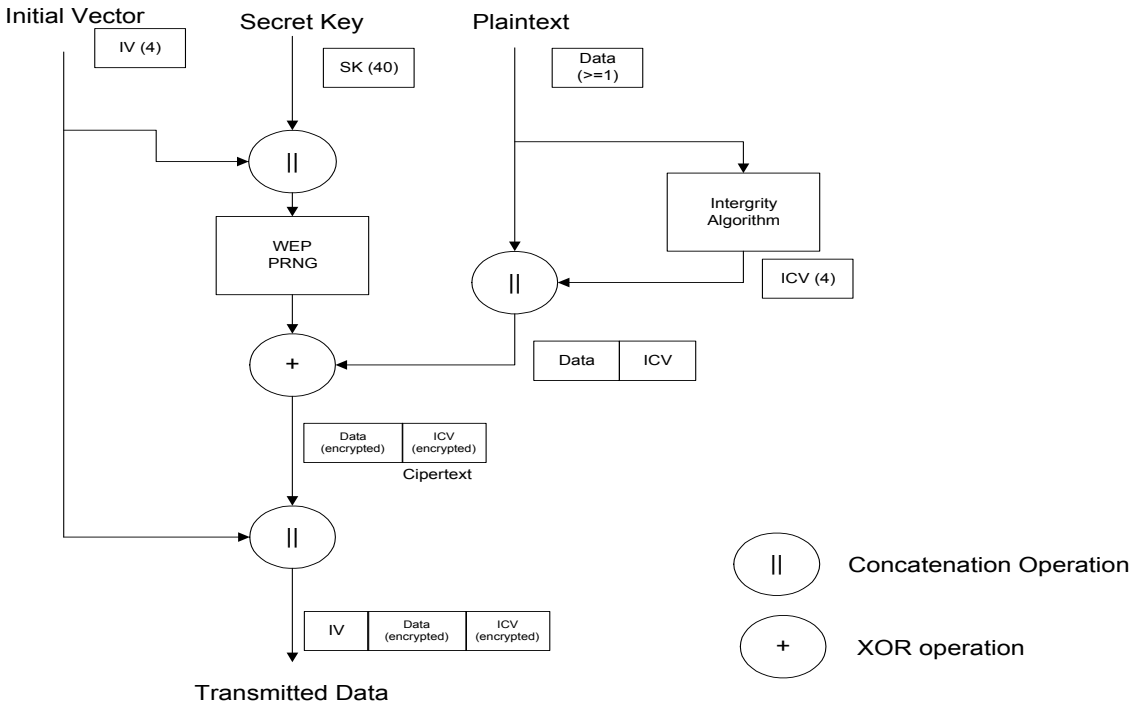


Figure 6: Encryption in Wired Equivalent Privacy (WEP)

The integrity algorithm operates over the plaintext to produce Integrity Check Value (ICV), which is 4 bytes, to assured that the data is not modified on the fly based on CRC-32 scheme. The WEP PRNG is a pseudorandom number generator that generates a key sequence k of length equal to the plaintext and the ICV. The key sequence generation algorithm is from RC4. The ciphertext produced by the XOR operation with the key sequence is combined with the initialization vector to produce the output message to be transmitted.

It is important to note the critical components of the protocol. The WEP PRNG converts a small key into a long key sequence, k, which is combined with the plaintext and ICV to produce the ciphertext. The inputs to the WEP PRNG algorithm are the IV and secret key. Although the secret key remains constant, the IV changes frequently with every new message. Thus the pseudorandom number used for every message is different. The IV is transmitted in clear as is does not provide any information about the secret key and its value is essential for the process of decryption at the other end. The WEP protocol also allows the possibility of using four different

secret keys. The Key Id in the message specifies which key was used for encrypting the message. The WEP algorithm thus self synchronizes with every message that is transmitted.

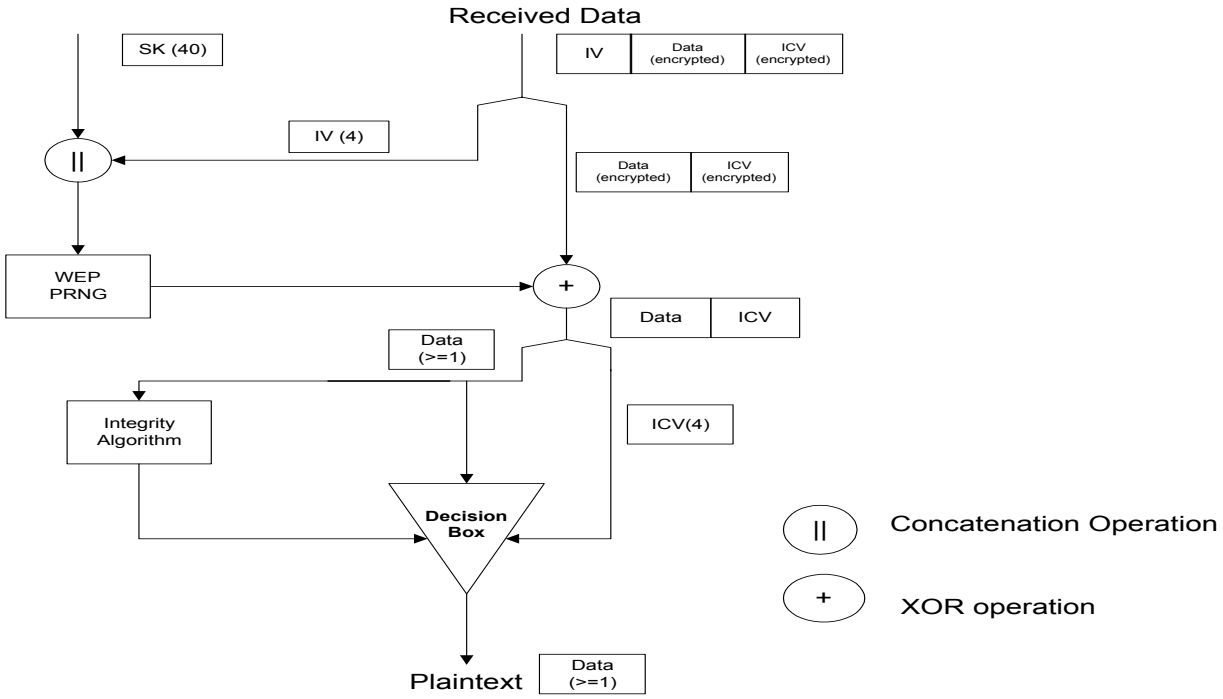


Figure 7: Decryption in Wired Equivalent Privacy (WEP)

During decryption the secret key and the IV from the received message are used to generate the key sequence. XORing this key sequence with the ciphertext gives the plaintext and the ICV. The CRC-32 integrity algorithm is used on the plaintext to see if the ICV received matched with the ICV produced by the plaintext. The erroneous messages for which the ICV do not match are simply ignored and not passed to the upper layers.

The primary goal of WEP is to provide privacy of data in the wireless environment, which is equivalent to the wired environment. However, WEP falls short of its goal. The algorithm is vulnerable to various passive and active attacks. The WEP protocol specifies the use of 40-bit secret key length. This key length is short enough to make brute force attack practical for organizations with fairly good computing resources. Some vendors have extended the WEP protocol to use 128 bit keys, which actually has a 104-bit key and 24-bit IV. The extended

version rules out the possibility of brute force attack on the key. However, the 128 bit key version is also not very secure because of the inherent weakness of the algorithm.

First attack comes from XORing two ciphertexts that have the same IV. If the IV is same the key sequence, *K*, for the two would also be same hence we could present the encryption of plaintexts *P1* and *P2* to produce ciphertexts *C1* and *C2* as follows:

$$C1 = P1 \oplus K$$
$$C2 = P2 \oplus K$$

Then,

$$C1 \oplus C2 = P1 \oplus P2$$

Partial Knowledge of *P1* or *P2* could reveal the information content of the other. The use of new IV per packet is intended to defeat this attack. The WEP standard recommends use of the IV with every packet but it does not specify how. In some PCMCIA cards analyzed the IV is reset to 0 every time the card is reinitialized and then is incremented with every packet. This leads to high use of the IV's of lower values.

Second attack is on the integrity of data. The ICV field in WEP is based on CRC32. It is possible to flip bits of the message such that the CRC32 checksum would still be valid for the data. The CRC32 algorithm is based on parity and flipping bits on the result is simple because the keystream is just XORed to the message and CRC32 value.

2.4 Internet Protocol Security (IPSec)

Virtual Private Network (VPN) provides the leverage to use the Internet as a private LAN. However, with the flexibility offered, they also expose the internal network to outside attacks. Hackers could access confidential data by exploiting accounts or by simply exploiting bugs in the access protocol. To prevent the problem of direct Internet access VPNs are often associated with mechanisms that provide authentication and encryption services. IP Security (IPSec) provides the means to implement security at IP level. It ensures security not just for applications

that have security mechanisms but also to other ignorant applications that have not implemented any protection schemes.

IPSec provides security against open access, data manipulation, man in the middle attack and passive monitoring. IPSec has two functions by which it provides the security: Authentication Header (AH) to provide authenticity and Encapsulating Security payload (ESP) to encrypt the data portion of the IP packet [3]. IPSec allows systems to chose the algorithms and select the protocols used for services. The use of AH or ESP is defined by Security Association (SA). A SA is a one-way relationship between sender and receiver. Thus for two-way security two SAs are required.

The AH is based on the message authentication code (MAC). The MAC is generated by a shared secret between two terminal parties. The AH has a sequence number which is incremented every time the SA is used. This prevents replay attack. After all the sequence numbers for the SA are used a new SA needs to be established. The AH also has a value called the Integrity Check value (ICV). The ICV is a message MAC or a truncated version of a code produced by a MAC algorithm, which may be either SHA-1 or MD5. The MAC is calculated over the IP header fields that either do not change or have predictable change, AH fields other than the ICV field and the upper layer protocol data.

With Encapsulating Security Payload (ESP) the transmitter encrypts the payload of the IP packet. With ESP the data is padded to ensure the plaintext to be multiple numbers of bytes required by the encryption scheme. Number of algorithms can be used with ESP. These include the following: TripleDES, RC5, IDEA, CAST, Blowfish.

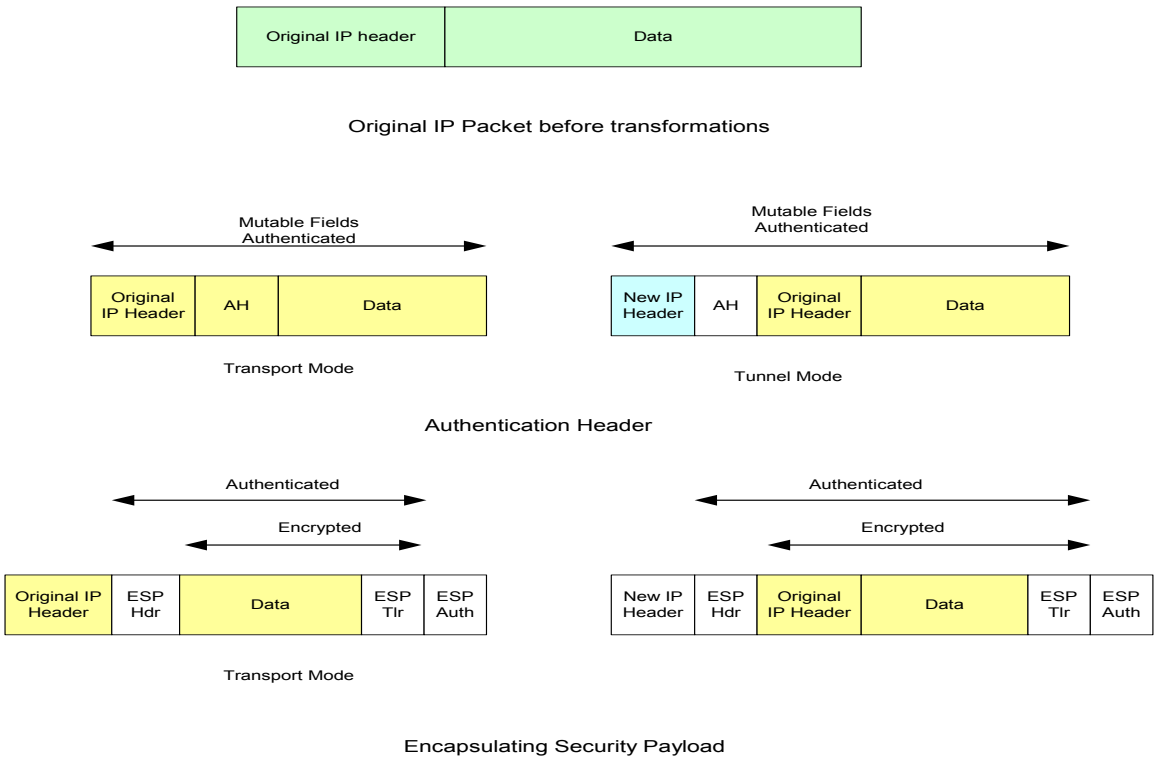


Figure 8: IP Security Packet formats

IPSec can operate in two modes: Transport mode and the Tunnel mode. The Transport Mode provides protection primarily to the upper layer protocols. It is used for secure authenticated communication between two IP connected hosts. The ESP in Transport Mode encrypts the IP payload and optionally authenticates it while the AH in transport mode authenticates the IP Payload and selected portions of the IP Header.

In Tunnel Mode the complete original IP Packets are encapsulated inside the AH or ESP to provide a secure tunnel. Thus the original IP packet is treated as payload and is not examined in the intermediate stages. Tunnel mode is essentially established to create secure communication between two network endpoints, which could be routers or gateways. In tunnel mode the ESP encrypts and optionally authenticated the entire IP packet including the inner header. AH in tunnel mode authenticates the entire inner packet and selected portion of the outer IP header.

2.5 Secure Socket Layer (SSL)

SSL has been originally developed by Netscape and has been accepted widely by the World Wide Web for Internet secure communication between client and server [3]. SSL operates above about some reliable transport protocol like the TCP layer that is the standard for connection-oriented communications over the Internet. It comes below the application layer protocols like HTTP, IMAP. SSL facilitates the use of Server Authentication, Client Authentication and secure encrypted communication between the client and the server. SSL fragments the message to be transmitted, optionally compresses it, applies message authentication code, encrypts it, and transmits the message using services provided by TCP. At the receiving end the messages are received, decrypted, verified, decompressed, reassembled and delivered to the upper layers [10].

A SSL session is an association between the client and the server. The SSL session protocol is layered and has two layers. One of the upper layers, the SSL Handshake protocol, allows the client and server to authenticate each other, negotiate the encrypting algorithm and the keys for encryption before the applications starts transmitting or receiving data. The upper layer is encapsulated by the SSL Record protocol. SSL session allows the use of multiple transient connections between the client and the server within on SSL session, which allows avoiding the renegotiations of the protocol parameters for each connection.

The SSL Record Protocol provides encapsulation to the upper layer protocols. It fragments the data into data blocks of 16384 or less. A lossless compression algorithm then compresses the fragmented blocks. The compression algorithm may be specified as null in case compression is not desired. The compression algorithm cannot expand the data above 1024 bytes in the worst-case scenario. In the next step a message authentication code (MAC) is computed over the compressed data. The compressed message and the MAC are then encrypted using a symmetric key encryption scheme. The encryption may not increase the content length by more than 1024 bytes. The algorithm for MAC and symmetric key encryption can be selected during the handshake protocol. Initially the algorithm for MAC and symmetric encryption are null.

The SSL session begins with the handshake. Handshake protocols helps in negotiating the parameters for secure communication using SSL sessions in addition to client and server authentication. The following steps are involved in SSL handshake protocol:

1. The client sends a client hello message to the server. This specifies the SSL version, random number used as nonce, session identifier in case the client wants to reuse the parameters from a previous session, CipherSuite list containing the algorithms supported by the client, and the compression method.
2. The server responds to the handshake message by either a handshake failure or a server hello message. The server hello contains the protocol version that is the lower value of the versions supported by the client and the server, randomly generated data, session id that may be a new session id or an old one if the client had sent a previous session id and the server is willing to reuse that session, the cipher suite selected by the server from the client CipherSuite list and the compression method selected.
3. If the server is to be authenticated, the server sends its certificate after the server hello message. In case anonymous Diffie-Hellman scheme is used server certificate is not required.
4. In case the server has no certificate, has certificate only for signing, or FORTEZZA KEA [3] key exchange the server key exchange message is sent. The server key exchange message has the public key for the algorithm chosen in case RSA or DH schemes are used or a secure random number for FORTEZZA KEA and a hash of the parameter with the client and server nonce using SHA or MD5 signed by the signature scheme.
5. In case the client authentication is desired the server sends the client a Certificate request. The certificate request contains the certificate type and the certification authorities supported.
6. The server hello done message is then sent to indicate the end of messages associated with server hello. After sending the message the server waits for a response from the client.
7. If the server has requested a client certificate then the client now responds with the Client certificate message. If no client certificate is available the client sends a no client certificate message to which a server can respond by a handshake failure.

8. The client then sends a client key exchange message to the server. If the KeyExchange algorithm was RSA, the client generates and 48 byte pre-master secret and encrypts it with the server's public key. For DH or FORTEZZA the client's public parameters are sent to the server to calculate the pre-master secret. The Server and Client generate a master secret from the 48 byte pre-master secret, random number from client hello and random number from server hello by using the MD5 and SHA schemes.
9. The client may send a Certificate verify message to the server for verification of the certificate. This message contains hash code of padded master secret and handshake messages signed by the client. The purpose of this message is to verify client's ownership of the private key for the client certificate.
10. The finish message is immediately send after the change cipher spec message from the client. The finish message is the first protected message with the just negotiated algorithms, keys and secrets. It is a concatenation of hash over master secret and handshake protocol messages exchange before with SHA and MD5 schemes.
11. In response to the above two messages the server sends its own change cipher spec and finish messages to the client. At this point the handshake process is completed and the client and server begin to exchange the application data.

Other than Handshake protocol there are two other upper layer protocols in SSL: Change Cipher Spec protocol and Alert protocol. The Change Cipher Spec protocol consists of a single message called the change cipher spec message that is used by the client and the server to notify the receiving party that the subsequent messages will be protested by the just-negotiated CipherSpec and keys. The Alert protocol is used convey alert messages to the peer entities. Alert messages convey level of severity and the description of the alert. The alert messages are encrypted and compressed like other messages. The following alert messages are included in the protocols: close notify, unexpected message, bad record mac, decompression failure, handshake failure, no certificate, bad certificate, unsupported certificate, certificate revoked, certificate expired, certificate unknown, and illegal parameter.

2.6 Energy Efficiency

Wireless communication is difficult to achieve compared to wired communication because of the time varying nature of the RF carrier. In general wireless communication is dominated by high error rates, varying signal to noise ratio, noise variations, limited bandwidth and multipath delays. These factors lead to retransmissions and effectively higher power consumptions.

Recent trends show increased demand for multimedia capable wireless devices. However, they are limited by the size, battery power and communication bandwidth constraints. The ever-increasing demand of networked services will aggravate the problem even further. Energy constraint poses restrictions on the functionality that can be implemented on wireless devices. Unfortunately, the battery performance improves at a very slow rate in terms of energy available per unit size or weight. An alternative approach of handling the problem is through energy efficiency at various levels of the system.

Energy efficiency can be achieved at the physical level by improving the communication protocol to make it more efficient. Quality of service trade offs is another mechanism to achieve energy efficiency. The requirement for energy efficiency is that the systems need to adapt so that the communication consumes relatively less energy.

Traditionally, energy efficiency has been achieved through focus on low-powered VLSI design. Most of the components are currently fabricated using CMOS technology. The power consumption of CMOS devices is directly proportional to the switching activity due to computations, capacitance, square of voltage, and frequency of operation. Reducing the frequency would reduce the power but the energy consumed would remain the same cause the same operation may be executed over a longer period. Most of the concentration hence is directed towards reducing voltage and capacitive load.

Dynamic power management schemes that adapt the system are required to achieve energy efficiency. They concentrate on optimizing the throughput and the performance of the system by reducing idle time and deactivation units when not required. For example while communicating

the wireless card could be turned off till there is no data to be transmitted since the energy consumption is significantly different during the transmit/received mode and sleep mode. However, the time to shutdown and wake-up the card should be much less than the data to be transmitted. The ways of achieving efficiency could be by queuing data and transmitting it all together or by using an adaptive wake up scheme.

Different devices have different energy utilization patterns. In laptop it has been observed that energy use of a laptop is dominated by the backlight of the display, the disk and the processor. However, generally the network interface consumes the same amount of energy as the rest of the system. The effective impact of each component on the overall battery life may be different for different devices due to their difference in operating systems, processor speed, memory size, battery size and other components.

In secure communication environment factors of concern are communication and computation efficiency. Computation efficiency depends on the hardware and the software used for encrypting the data. It would certainly be inefficient to provide customized hardware for encryption since it would mean lot of additional gates and hence more power consumption. Programmability of encryption scheme is essential in today's environment because of the advancing computational power and increasing key size requirements. Also, for energy efficiency we should be able to adapt encryption techniques to meet the varying data rates and quality of service requirement of applications. Another problem of providing customized encryption hardware is the differential power analysis attack where the electrical activity of the device is monitored to determine the secret information.

2.7 Battery Technology

Batteries used in wireless devices are an essential component. The batteries used with current technology provide a lifetime of 1.5 hours to 4 hours. With people using more and more of the wireless devices there is a heavy demand for longer lasting batteries of less weight and smaller size. Unfortunately, the battery technology is not progressing as fast as the digital technology or

the increasing user demand. On an average there is 2% increase in battery efficiency every year [38]. In this section we will look at various types of batteries that are being used in today's market.

Various technologies exist today for batteries. Table 5 provides the characteristics of some of the existing battery technologies. Voltage presents the amount of power that can be delivered by the battery and a milli-amperes hour (mAh) is the time for which that power can be delivered. Higher voltage is preferred since if battery has lower output voltage then additional circuit is required to convert the voltage to higher values.

Battery type	**Li-ion**	**Li polymer**	**NiCd**	**NiMH**
Rated voltage	3.6 V	3.6 V	1.2 V	1.2 V
Capacity (mAh)	to 1800	to 800	to 700	to 1100
Weight energy density(Wh/kg)	to 150	to 180	to 60	to 80
Volume energy density(Wh/liter)	to 300	to 260	to 200	to 300
Memory effect	No	No	Yes	Yes
Self discharge	Small 5%/Month	Small	Large 15%/Month	Large 20%/Month
Internal impedance	High	High	Low	Low

Table 5: Characteristics of Major Battery Systems [39]

Rechargeable Nickel-Cadmium (NiCd) batteries are popular for use in cordless phones and other moderate power devices. These batteries have a 'memory effect', which means they remember the previous discharge level when recharged and may play dead when they reach the state again. Hence it is required that they should be recharged after fully discharging. Nickel-metal hydride (NiMH) batteries are less susceptible to the 'memory effect' and can hold more charge for the same size. They are popular as rechargeable AA and AAA batteries. One down side of NiMH is they have a higher self-discharge rate than NiCd.

Lithium-ion (Li-ion) batteries are most preferred for devices like laptops and the newer cell phones and digital cameras. They have higher energy density figure, low self-discharge rate and quick recharge time compared to NiCd and NiMH. There are two types of negative electrodes

used in Li-ion batteries: coke and graphite. Li-ion batteries clearly have many advantages but they also bring in certain complications in the device circuitry. They require protection circuits to ensure linear charge and discharge and to prevent overcharge and overdischarged. If the user overcharges the electrolyte solution is decomposed and gasses are released increasing battery pressure and lithium is precipitated causing a rise of fire and explosion. If overdischarge occurs the electrolyte decomposes and the characteristics deteriorate.

In improving battery technology, there is greater focus on power management of these battery packs. An increasing number of today's Li-ion batteries are smart. They come equipped with a microchip that control charging, discharging and conditioning of the batteries. The power management system generally has circuitry for voltage detection, voltage regulation, switching regulation and battery protection [39]. For battery protection, it detects overcharge or overdischarge conditions and disconnects the battery. Overcharge is detected by over-voltage and overdischarge by under-voltage. Charging begins with checking for presence of good batteries. For good batteries if the voltage is low (below 3.2V) the charging is done with low current. Once the batteries reach a set value they are charged through maximum charging current. The discharge circuit generally has the cells in parallel with a voltage detection and voltage regulator to maintain constant output voltage.

Chapter 3
Experiment Design

The setup for the experiment is as shown in figure 9. The server used was an IBM compatible PC with Athlon processor. The server had Windows XP professional installed on it. The server is connected to the access point with 100 Mbps Ethernet link. The access point used for the experiments is Cisco AP1200 series with a 2.4 GHz wireless module. The transmission power of the wireless module for the access point was set to 15 dBm as the Orinoco cards used in the system for wireless interface had 15 dBm output power configured. The wireless devices considered in the experiment are laptop, Pocket PC and Handheld PC. The wireless devices are used one at a time and never all at the same time. Transmission of data is from the wireless devices to the server through the wireless link using TCP/IP protocol. For encryption the libraries considered were Crypto++ and OpenSSL. OpenSSL was only used for RSA on Windows CE. These are explained further below in section 3.1.

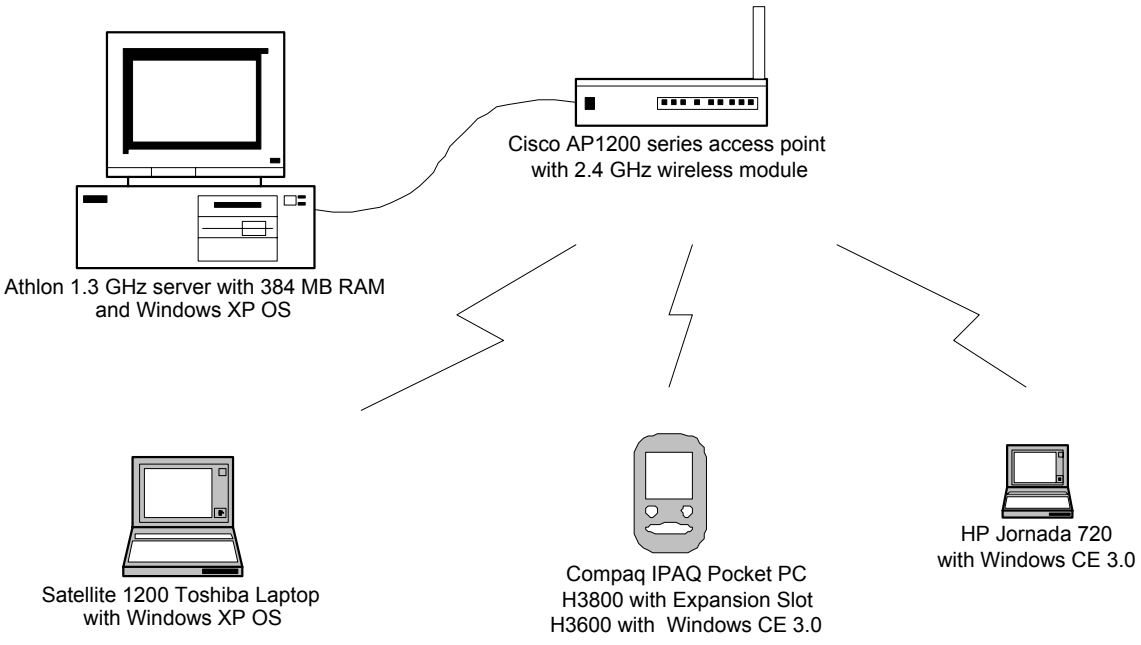


Figure 9: Configuration of the Experimental Setup

3.1 Encryption Libraries

We could have used applications like PGP for encrypting the file, but we chose using encryption library to avoid the process startup and close overheads, which include memory allocation, loading program, configuration, and cleanup. Numbers of libraries are available supporting most of the cryptographic algorithms. Some of popular libraries are as follows:

1. **Crypto++** [18] is a free C++ class library. It has implementation for most of the popular cryptographic algorithms. It has implementation for AES, IDEA, DES, Triple-DESRC2, RC5, Blowfish, Diamond2, TEA, SAFER, 3-WAY, GOST, SHARK, CAST-128, Square, Skipjack, Panama, ARC4, SEAL, WAKE, WAKE-OFB, BlumBlumShub, RSA, ElGamal, Nyberg-Rueppel (NR), Rabin, Rabin-Williams (RW), LUC, LUCELG, DLIES (variants of DHAES), ESIGN, ECIES. The dedicated author Wei Dai actively maintains the library.
2. **OpenSSL** [40] is an open source development project written in C for a full-fledged toolkit that implements the SSL version 3 and TLS version 1 protocols along with a fully functional cryptographic library. It has implementations for AES, DES, Triple-DES, IDEA, RC2, RC4, RC5, SAFER, DH, RSA, and DSA.
3. **Cryptlib** [41] is another open source library in C and implements most of the popular algorithms for encryption and authentication. It has implementation for Blowfish, CAST, DES, triple DES, IDEA, RC2, RC4, RC5 and Skipjack conventional encryption, MD2, MD4, MD5, RIPEMD-160 and SHA hash algorithms, HMAC-MD5, HMAC-SHA, and HMAC-RIPEMD-160 MAC algorithms, and Diffie-Hellman, DSA, ElGamal, and RSA public-key encryption.
4. **Cryptix** [42] is Java cryptographic library implemented in Java and Perl. It has implementations for Blowfish, CAST5, DES, IDEA, MARS, RC2, RC4, RC6, Rijndael (AES), Serpent, SKIPJACK, Square, TripleDES, Twofish, RSA, DH, and ElGamal.
5. **MIRACL** [43] is a big numbers library, which implements all primitives necessary for Cryptography. It has implementations in C with C++ wrapper for Linux and Windows based systems. It offers full support for elliptic curve cryptography and has a multithreaded wrapper for the IEEE 1363 Public Key Cryptography specification.

All libraries are generally up-to-date with the latest development in number theory that helps faster mathematical operation. Crypto++ was the basic library selected for encryption since is has a sample benchmark published on its website. The choice was primarily based on ease of use, availability of benchmarks for verification, algorithms supported. A comprehensive comparison of the features and performance of the libraries was not done because it is outside the scope of this thesis. The Crypto++ code for AES and CAST was slightly modified so that the same code for those schemes could be run on the devices with Windows CE operation system. In case of implementing RSA on WinCE environment OpenSSL was used as it supports WinCE because porting the RSA code in Crypto++ to WinCE was not considered feasible.

3.2 Methodology

The flow chart for the software code used in the experiment is presented in figure 10. The battery and computational trade-off of encryption schemes under different scenarios are considered in various experimental setups but the underlying setup remains the same. Initialization in case of encryption would be to establish the keys required while in case of data transmission it would include establishing a connection with the server. For the purpose of comparing their performance, a big file of size 5MB unless explicitly mentioned is processed multiple number of times.

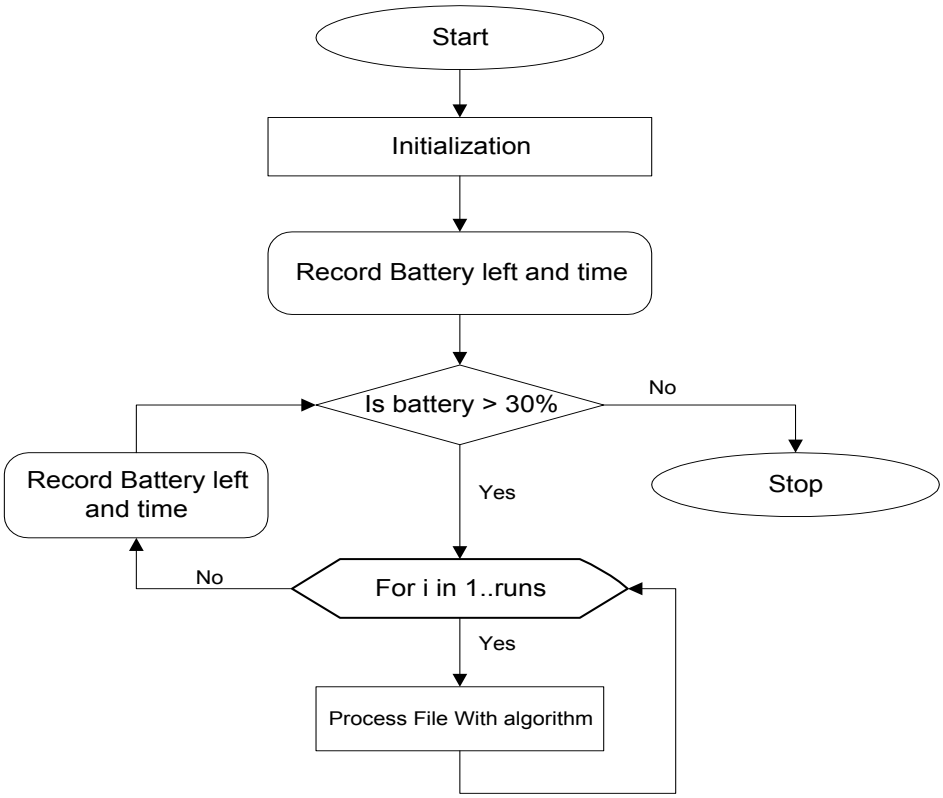


Figure 10: Flow Chart for Driver Program in the Experiments

Processing in experiment for encryption without data transmission is to read data from the file encrypt the data and put it in another file. In case of encryption with data transmission the data is read from the file encrypted and the send to the server through the connected socket. This is done till the battery drains to 30% of the lifetime left. We stop at 30% because after that the systems alarm and data recovery mechanisms become active and the performance of the schemes change. The socket connection used is TCP socket for reliable transmission of the data. After a few runs of processing on the file the battery life left and the system time is recorded. The average battery life consumed per run and the time taken to do so is the calculated for the results.

It is expected that the computation time would be closely related to the battery requirements; however, since the CPU utilization of power depends on parameters like voltage supply and capacitive load. The capacitive load on the CPU depends on the switching demand, which again

depends on the instructions being executed. Hence, measurements for both the parameters are considered.

The battery capacity keeps reducing with the number of times it is charged and discharged. Hence, care is taken during the experiments that the battery is not unnecessarily discharged too many times. Also to ensure that this effect does not affect the results, the measurements for a group of readings for one section are taken together. In all about 100 discharge cycles have been completed on the laptop and 10-20 cycles on the Pocket PC and Handheld PC. 100 discharge cycles would lead to reduction in capacity of 5% for Li-ion batteries that the laptop uses from the characteristics published by Toshiba [44].

The experiments note the number of iteration or runs over the file and the battery life. So to restrict the number of measurements statistical properties were gathered by taking the difference in battery life left. Change in battery life divided by the number of runs gives the battery life consumed in percentage for one run. It was observed that since the charge and discharge cycles of the laptop were more or less linear and the memory effect that leads to non-linear discharge characteristics is negligible for Li-ion batteries.

To demonstrate how the calculations were done consider Table 6 as an example of the observations for a particular scheme.

Observation Number	Number of Iterations	Percentage Battery Left
1	10	98
2	20	97
3	30	95
4	40	94
5	50	92

Table 6: Sample table of observations

Next we calculate the difference in iterations between different observations as follows.

Observations	Diff in Interation (a)	Change in % battery left (b)	Battery Consumed Per Iteration (b/a)
1,2	10	1	0.1
2,3	10	2	0.2
3,4	10	1	0.1
4,5	10	2	0.2

Table 7: Sample table for calculations

The average battery consumed per iteration can thus be calculated as follows

Average battery Consumed per iteration = (0.1 + 0.2 + 0.1 + 0.2)/4 = 0.15

Thus the statistics for reliability was gathered by taking every battery life change divided by runs to get battery life per run over the period of the experiment.

3.3 Comparison of algorithms

Purpose of this section of experiments is to see the trade-off in choosing secure algorithms and their computational and battery requirement costs.

3.3.1 Symmetric Key Schemes:

We have reviewed of symmetric key schemes: AES128, CAST128, IDEA in chapter 2. All of these schemes provide 128 bits key encryption schemes and provide the same level of security as discussed earlier in section 2.2.3. However they have different performance in terms of time and battery requirements. Crypto++ library provides implementation of AES, CAST and IDEA. Programs are written with each scheme that record battery drain and time after encrypting a 5MB file 10 times. Also a case where just the file is accessed and written to another file is considered to differentiate between the file access load and encryption load. In case of No Encryption and

AES128 the minimum data block size accessed is 128 bits (16 bytes) while for CAST128 and IDEA the minimum data block size accessed is 64 bits (8 bytes).

In secure communications architecture it should be useful to change the key size based on the battery power remaining in the system to reduce the load. Hence effect of varying key size to 128, 192, and 256 bit keys is studies for AES algorithm.

AES has 10 rounds for 128-bit key. It should be possible to change the number of rounds of encryption to make a quality to battery utilization compromise. Reducing the number of rounds would make the algorithm less secure but would have the advantage of being able to use the same key and adaptively switching the number of rounds based on the channel and the device battery status.

3.3.2 Asymmetric Key Schemes:

We have reviewed asymmetric key schemes: RSA, ElGamal, ECIES in chapter 2. Programs are written with each scheme that record battery drain and time after encrypting a 5MB file 10 times. Also a case where just the file is accessed and written to another file is considered to differentiate between the file access load and encryption load. In case of RSA the test is further extended. The effect of varying key sizes from 512 bits to 2048 bits is studied.

3.3.3 Different Devices

The wireless environment is dominated by various kinds of devices and all devices have different performance because of their varying architecture: CPU, memory, and battery. In our study the devices selected for study are Satellite Toshiba 1200 Celeron Notebook, Compaq IPAQ Pocket PC, and HP Jornada 720 Handheld PC 2000. The all devices are tested in windows environment. Majority of the performance study is done on the laptop. Study on Compaq has been restricted to AES and RSA while for Handheld PC just AES is studied. To test the performance of AES and CAST on Pocket PC and handheld the crypto++ implementation of AES and CAST was ported

to WinCE environment. To test the working of the port the encrypted data by AES on laptop and WinCE device were compared and found to be the same. Doing so ensured that the same AES code was being used on the laptop and the WinCE devices. For RSA the OpenSSL implementation of RSA for WinCE environment was used since porting RSA from crypto++ to WinCE proved to be a prodigious task.

3.4 Wireless Environment

Some perspective of the effect of changing wireless environment may serve to be useful while designing wireless communication systems. With this in mind factors like changing the signal to noise ratio, packet size, and layer where encryption is performed are considered in the experiments.

3.4.1 Data Transmission

All algorithms considered in above need to be considered when the data that is encrypted by them is transmitted over the wireless network. Initially the effect of signal to noise ratio is excluded by keeping the device very close to the access point to have excellent signal conditions. We evaluate the effect of the algorithm and key size variations under data transmission

3.4.2 Signal to Noise Ratio

Reduced signal to noise ration causes retransmission and transmissions under lower signal to noise ratio are more demanding on the battery. To understand the comparison of battery cost for encryption and transmission with varying signal to noise conditions measurements are taken transmission for data transfer with and without encryption under different signal to noise conditions.

3.4.3 Layer of Encryption

One of the goals of this study is to determine the effect of moving the encryption process to application level. So data transmission is done with encryption enabled at the link level by WEP and then same data is transmitted with WEP disabled and encryption at application level using AES.

3.4.4 Changing Packet Size

In 802.11b environment the TCP transmissions involve transmission of packets over which the TCP, IP and Ethernet headers are added. So the transmission of 128 bits data leads to and inefficient transmission mechanism. Also waiting for acknowledgements leads to reduction in effective throughput. The purpose of this set of experiments is to determine the change in performance observed by switching from 128 bits transmission to 1024 bytes transmission.

Chapter 4
Results and Analysis

4.1 Comparison of algorithms

We consider various symmetric and asymmetric key schemes with different key sizes and on different devices. The graphs for the results obtained are plotted in this section. The performance of the schemes and their implications has been analyzed in this chapter and suggestions have been made for designing energy efficient secure communication systems. The appendix section can be checked for actual numerical values that were obtained during the experiments.

4.1.1 Symmetric Key Schemes

The figure 11 below shows the battery consumed and the time consumed for iteration over a 5MB file. The comparison is done between AES encryption, CAST encryption and IDEA all with 128 bits key size using Crypto++ library. The 'No Encryption' scheme represents the case where the file was just accessed and the data from this file was put into another file. This was included to see actually how many extra resources are put into encrypting over accessing the data.

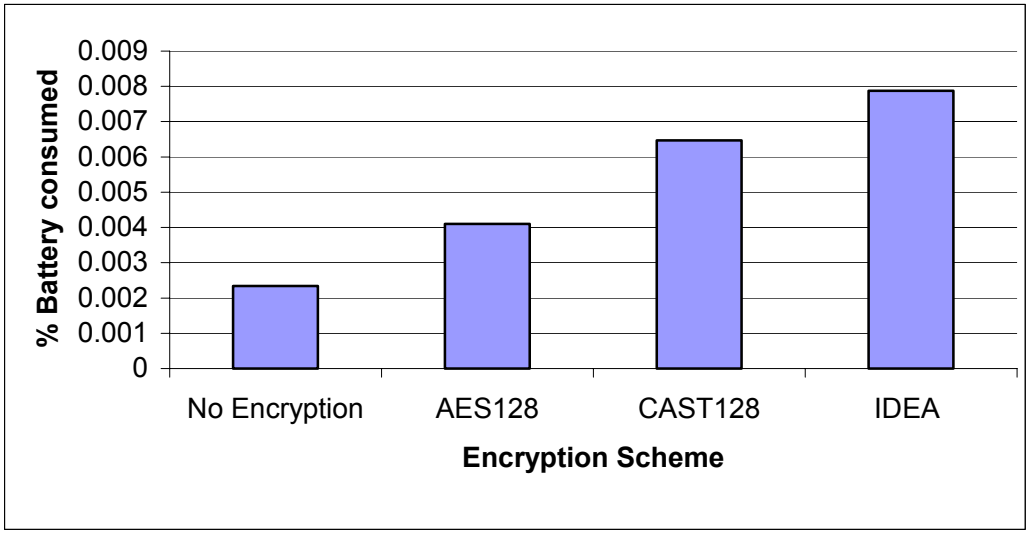


Figure 11: Percentage Battery Consumed by symmetric key schemes without transmission

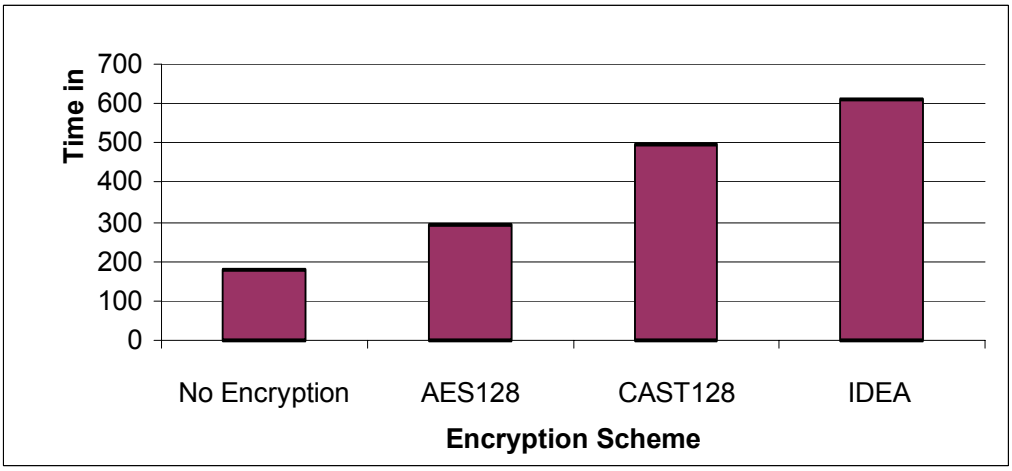


Figure 12: Time Consumed per iteration by symmetric key schemes without transmission

From the figure 11 CAST128 and IDEA are more power hungry than AES128. The reason for this is that the algorithms employ different diffusion and confusion techniques. If we recall the implementation details of the AES, CAST and IDEA algorithms it can be seen that the major computationally intensive operations in them were 120 rotate bytes, 21 circular left shift and 34 multiplication operations per encryption assuming that addition, subtraction and XOR operations are relatively less intensive. CAST128 and IDEA process 64 bits at a time while AES128 processes 128 bits at a time. The time and the battery consumed are proportional to the complexity of operations per round and the number of data moves required making it difficult to

state how much effect each type of operation has in the performance of the schemes. This is because data moves are difficult to analyze with different register allocations that can be done by the compiler and state of the operation system.

Encryption of the plaintext data with AES128 causes increase in battery consumption by 75% and time consumption by 65%, when compared to the No Encryption scenario. CAST128 consumes 58% and IDEA consumes 92% more battery than AES128. CAST128 takes 70% and IDEA takes 110% more time than AES scheme. The total energy of the Toshiba battery is 58Wh. A naïve approach to calculate the energy consumed per iteration over the file would be to multiply the percentage of battery consumed to 58Wh. For example, battery consumed by AES128 would be *0.00499 * 58 = 0.2377 Wh* for processing 5 MB of data. In terms of security, there are no known attacks on the unmodified version of CAST and AES. IDEA however has a set of 2^{51} weak keys [35] hence AES and CAST can be considered more secure than IDEA.

4.1.1.1 Key Size variation

We compare the change in performance by using different key sizes for AES algorithm. For AES we consider the three different key sizes possible i.e., 128 bit, 192 bits and 256 bit keys.

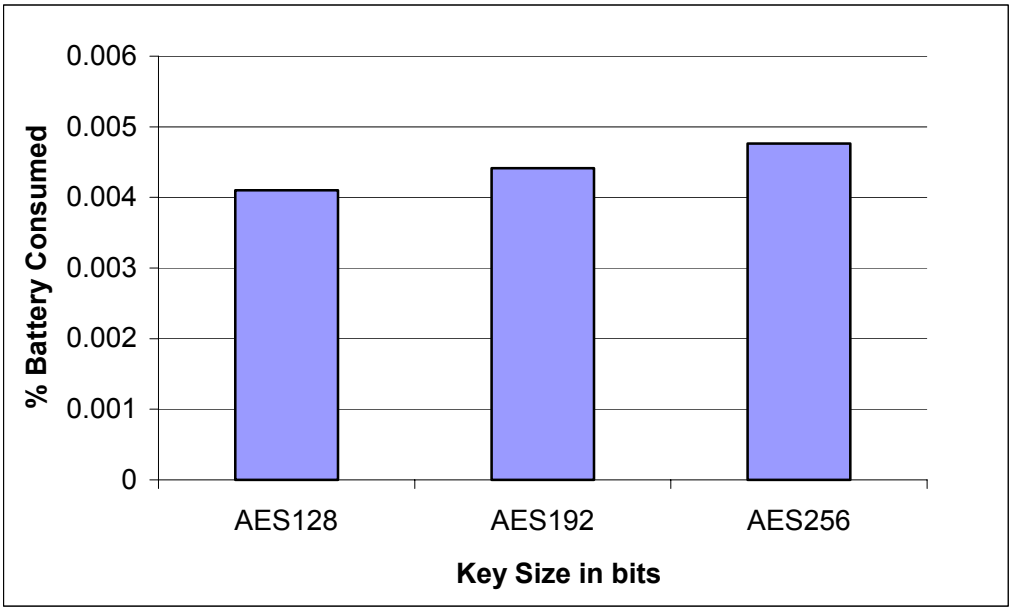


Figure 13: Percentage Battery Consumed with different Key Sizes for AES

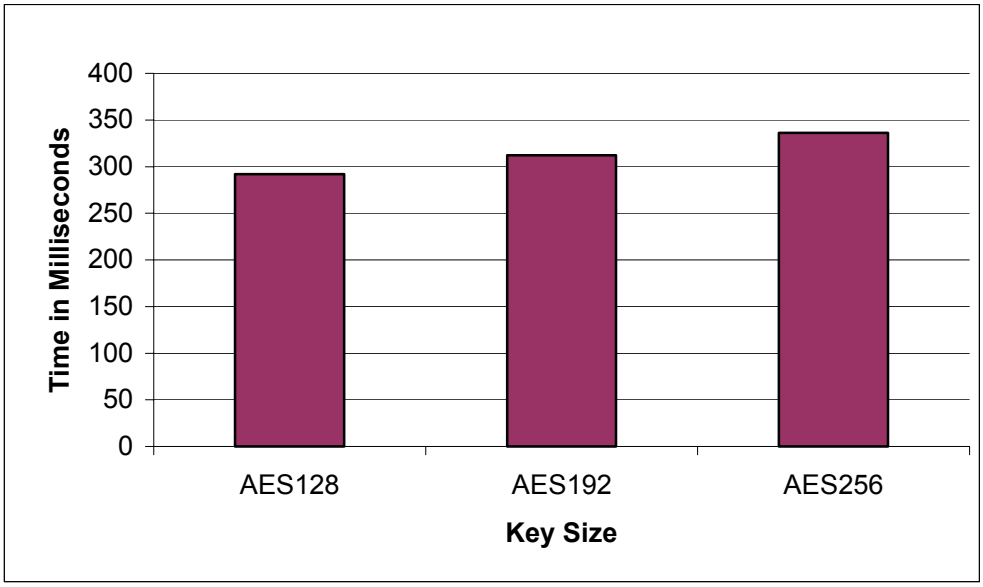


Figure 14: Time Consumption with Different Key Sizes for AES

Increased key size leads to increased security as shown in table 4. In case of AES it can be seen that higher key size leads to noticeable change in the battery and time consumption. It can be seen that going from 128 bits key to 192 bits causes increase in power and time consumption by about 8% and to 256 bit key causes an increase of 16%. AES128 has 120, AES 192 has 144, and AES256 has 168 rotate byte operations, which implies 20% and 40% more operations for AES 192 and AES256 when compared to AES128. Although there seems an increase in power consumption that is directly proportional to the increased operations, the increase is less amplified. This can be attributed to the fact that the data access from the file over which the operations are performed. However, the increased power consumption of higher key size posses a compromise that should be considered before choosing the size of the key. For normal application 128 bits key is considered very secure hence going for higher key sizes would mean unnecessary wastage of resources for the added security that is actually not required.

4.1.1.2 Changing Number of Rounds

The AES encryption scheme has 10 rounds for 128 bits key. It should be possible to reduce the number of rounds so that the amount of battery and time consumed while encrypting the data

could be reduced. Figures 14 and 15 below show the comparison of energy and time consumed by the reduced round version of AES 128 bits key encryption.

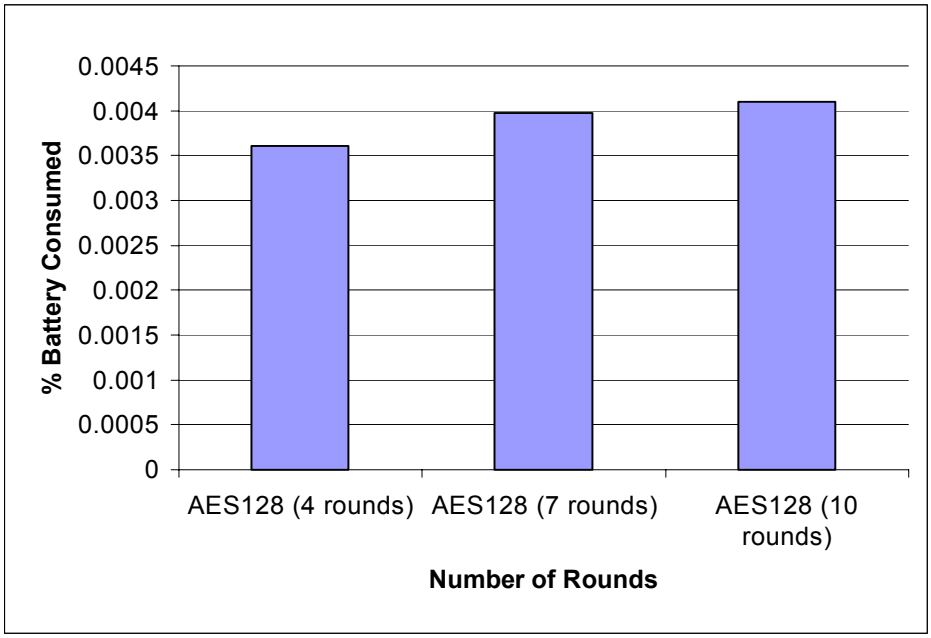


Figure 15: Percentage battery consumed by different number of rounds for AES 128 bit-key encryption

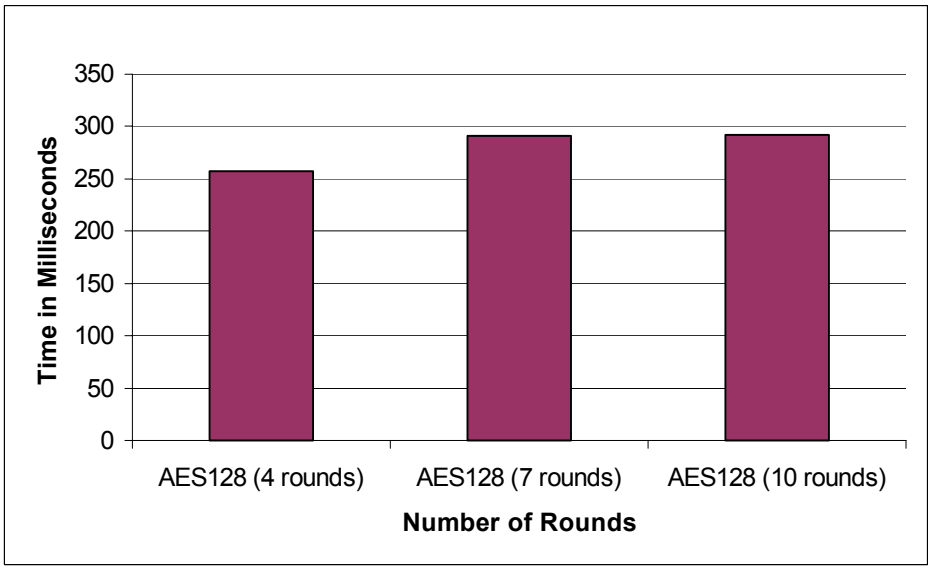


Figure 16: Time Consumed by different number of rounds for AES 128 bit-key encryption

As can be seen from the graph by reducing the number of rounds from 10 to 4 rounds it is possible to save 13% of battery and time consumption and from 10 to 7 rounds is 3%. Reducing the number of rounds would decrease the security of the encryption. The NIST report on AES [36] provides the work factor or operations and the memory required to attack the scheme under reduced number of rounds for 128 bits key size. No known attacks have been reported against the actual AES scheme with 10 rounds and the only way is through brute force which would take 2^{127} encryptions on an average.

Rounds	Operations	Memory
4	2^9	Small
5	2^{40}	Small
6	2^{44}	$7*2^{32}$
7	2^{120}	2^{61}
10	2^{127}	Not Known

Table 8: Attacks reported on reduced round variants of AES with 128 bits key [36]

It can be seen that 4 round encryption with AES is not very secure and under most cases would not be preferred. Compromising the number of rounds leads to highly reduced security for 4 rounds. As the number of rounds increase the security of the scheme improves, as can be seen from the table 8 above. There is likelihood of more attacks on the reduced round version of AES being discovered which increases the risk in using these schemes. The designer should consider the level of security required and the battery saving resulting from the performance improvements by reducing the number of rounds with extreme caution.

4.1.2 Asymmetric Key Schemes

In asymmetric key scheme RSA, ElGamal and ECIES are considered. The ECIES and ElGamal are hybrid schemes used in combination with symmetric key scheme in this case AES. The implementation of RSA used has OAEP standard for encoding then message with SHA for hashing. The RSA standard specifies the asymmetric key encryption but doesn't specify how it

can be used in conjunction with symmetric key schemes. The implementation is open to user. ElGamal scheme like RSA is also just encryption. It does not use any encoding or hash functions. ECC used here has XOR for symmetric key scheme and SHA [3,45] as hash function. The secp160k1 curve provided by Certicom [46] is considered for ECIES. For the purpose of the performance measurement the encryption is considered in units of 16 bytes of data. The ElGamal and ECIES encryption schemes employ pre-computation with storage space of 16 locations. In the case of Asymmetric key schemes the file size considered is 1MB instead of 5MB as the processing with asymmetric keys is much more time consuming.

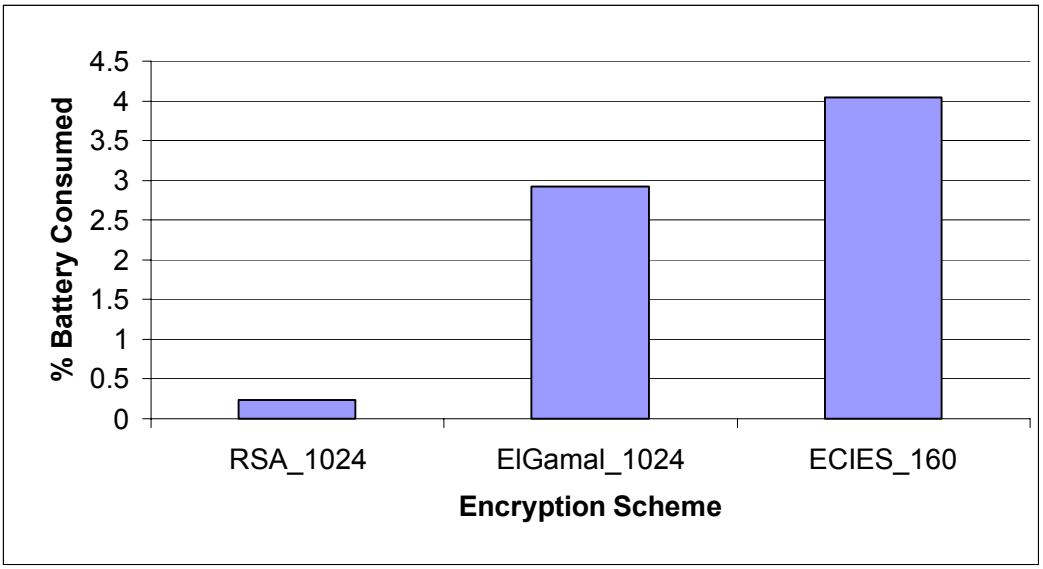


Figure 17: Percentage Battery Consumption of Asymmetric Key Schemes

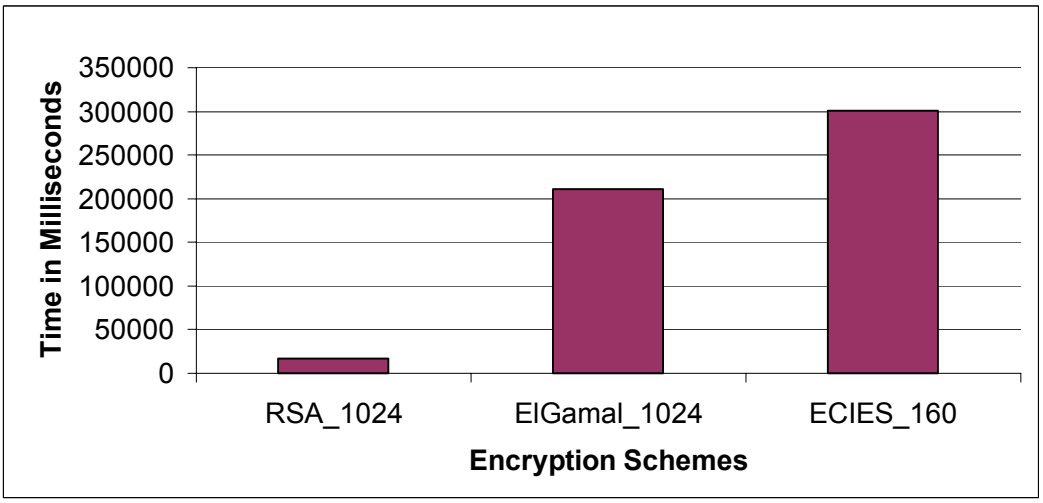


Figure 18: Time Consumption of Asymmetric Key Schemes

The figures 17 and 18 above show the performance of the schemes in terms of the time and the battery consumed without data transmission. We can see that for encryption RSA is much more efficient than ElGamal and ECIES schemes. Infact, RSA 1024 bits encryption is as much as 17 times better than ECIES 160 bits curve and 12 times better than 1024 bit ElGamal scheme. RSA encryption is much faster than the other two schemes because it uses short key for encryption, which is much smaller than the short exponent used in ElGamal. The exponent for RSA used in these experiments is 17 and can be represented in 5 bits while the short exponent in ElGamal is 164 bits. ElGamal encryption requires two exponentiation operations while RSA requires one. Also, the 160 bits elliptic curve multiplication in ECIES is costlier than the exponentiation of the short exponent in ElGamal. The assumption while making the statement is that the encoding schemes have a relatively low impact computational requirement as the RSA and ECIES schemes employ SHA functions before encrypting the message.

In our experiments it is also essential that we consider decryption performance of the algorithms. Figure 19 below gives the graphs for decryption process of these algorithms.

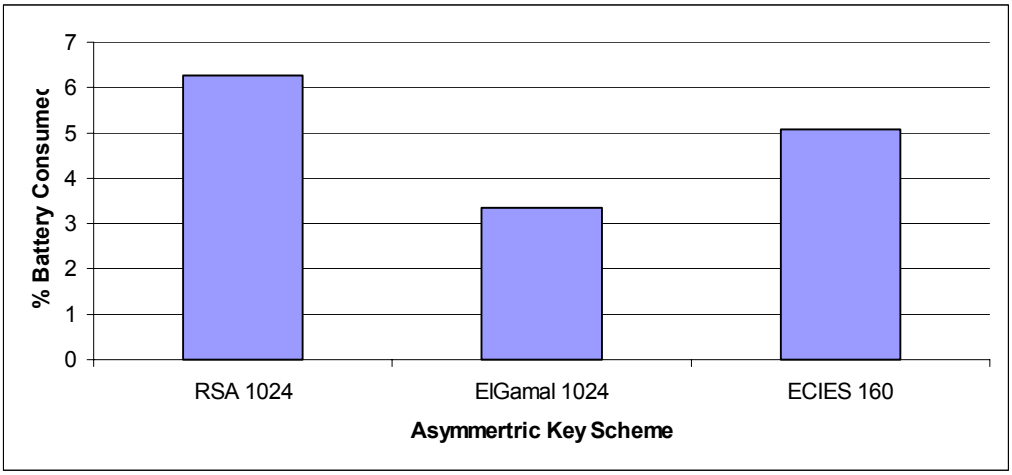


Figure 19: Percentage Battery Consumed by Asymmetric key decryption

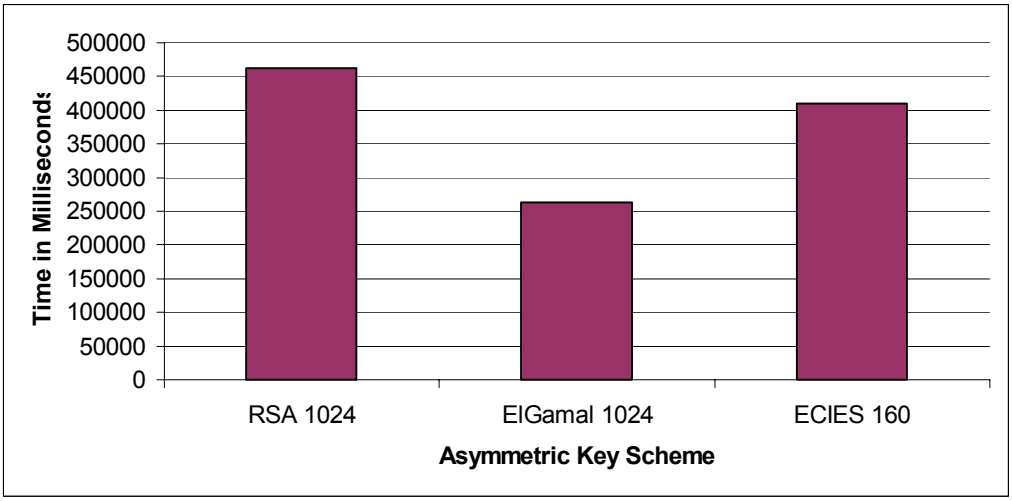


Figure 20: Time Consumption of Asymmetric Key Decryption

It can be seen that in the decryption process ElGamal 1024 is the fastest scheme. ElGamal decryption requires only one exponentiation operation with a 164 bit short exponent. RSA also employs one exponentiation for decryption but it does not employ short keys, as short decryption key would make RSA insecure [23]. The exponent in RSA is 1024 bits and in ElGamal is 164 bits. Even after application of the CRT [31] for RSA, which reduces the modulus size to 512 bits, ElGamal is still more efficient. ECIES decryption requires two elliptic curve multiplication operations. Also ECIES 160 bits decryption is about 20% more efficient than RSA 1024 decryption.

Looking at the performance of the Asymmetric Key Schemes RSA is very efficient for encryption while ElGamal is the most efficient for decryption process and should be chosen to save the energy consumed. ECIES is the most energy hungry of all the schemes when energy consumption with both encryption and decryption is considered. However, RSA and ECIES used here are more secure than ElGamal because they employ message authentication functions to ensure that messages decrypted are not invalid.

4.1.2.1 Key Size variation

We compare the change in performance by using different key sizes for RSA algorithm. The next figures show the performance of different key sizes of RSA. 512, 1024 and 2048 key sizes are

considered. For encryption the file being encrypted was 1MB and for decryption the decrypted file was of 1MB.

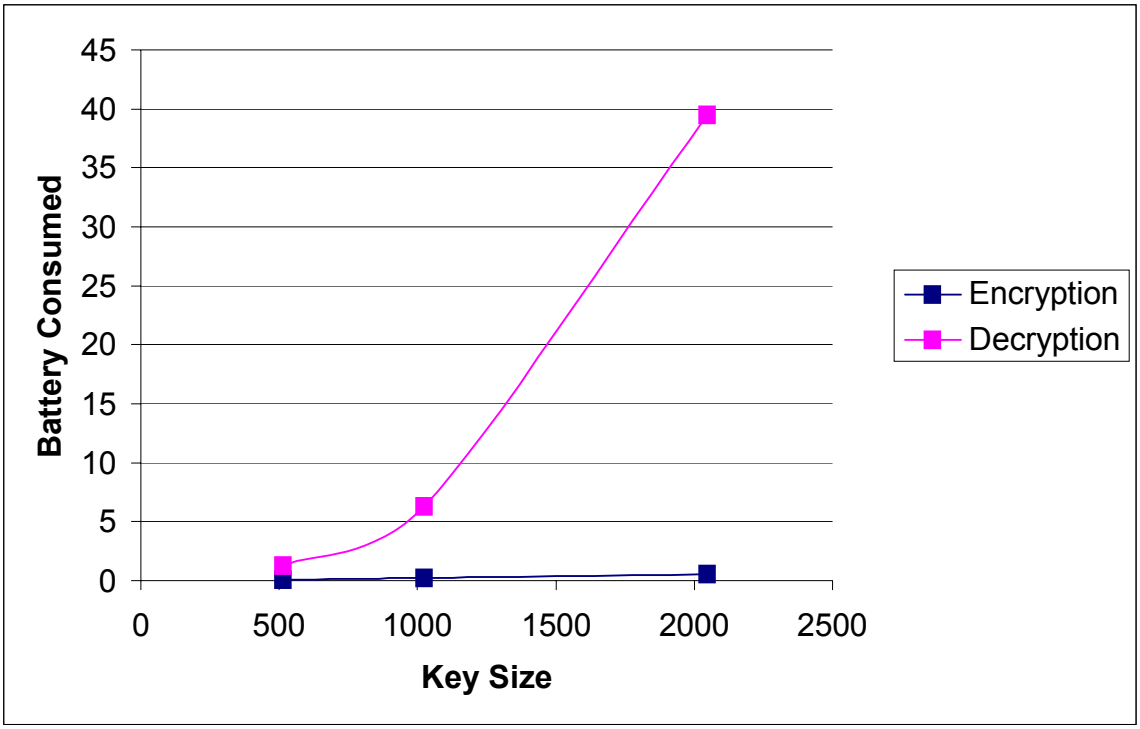


Figure 21: Percentage Battery Consumed with different Key Sizes for RSA without data transmission

Increasing the security by increasing the key size in case of RSA has a higher cost. The curve shows that as the key size doubles the battery and time consumption requirements are more than double indicating a nonlinear growth in consumption with increase in key size. In Chapter 2 we have seen that the Processing time for RSA is proportional to $1.5 * k * O(N^2)$, where k is the exponent and N is the size of the modulus. The exponent remains constant for encryption the processing time is proportional to $O(N^2)$. However, for decryption the exponent depends on the key size and follows a third order growth proportional to $O(N^3)$ as $k \alpha N$. Choosing 2048 bits security over 1024 bits security results in three times more energy consumption for encryption and 6.6 times more energy consumption incase of decryption. By current standards 1024 bit key is considered secure for normal applications till 2013. 2048 should only be opted for highly secure applications.

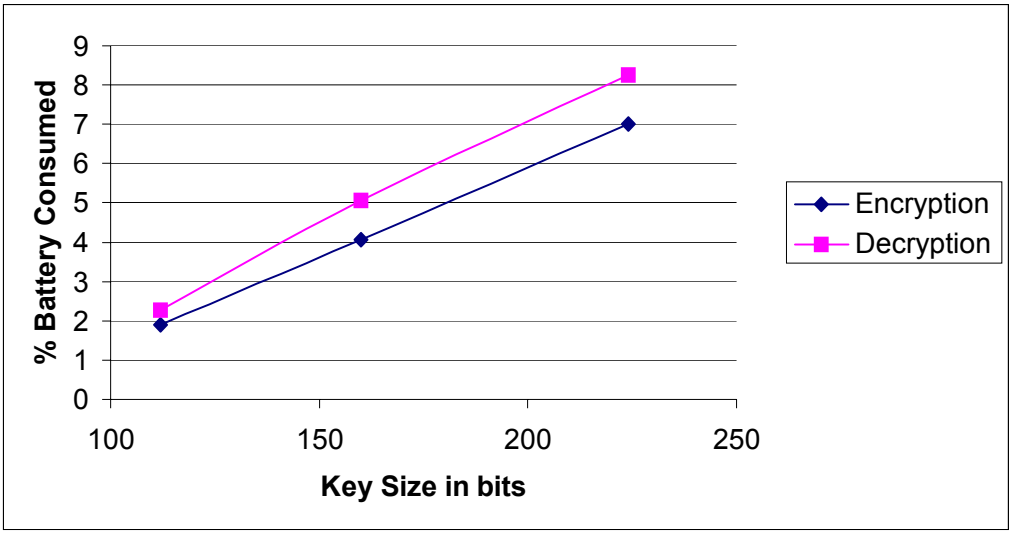


Figure 22: Percentage Battery Consumed with different Key Sizes for ECIES without data transmission

The figure above shows the increase in battery consumption with increasing key size. For encryption the file being encrypted was 1MB and for decryption the decrypted file was of 1MB. The curves used for ECIES are the ones defined by Certicom [46]. For 112 bits secp112r1 is used, for 160 bits secp160k1 is used and for 224 bits secp224k1 is used. There is increase in battery consumption by 114% with 160 bits key and 270% with 224 bits key.

Security (year)	**RSA**			**ECIES**		
	Key (bits)	**% Battery (Encrypt)**	**% Battery (Decrypt)**	**Key (bits)**	**% Battery (Encrypt)**	**% Battery (Decrypt)**
1982	512	0.1047	1.2727	112	1.8918	2.258
2013	1024	0.2372	6.2727	160	4.05	5.0769
2055	2048	0.5833	39.5	224	7.0000	8.25

Table 9: Comparison of percentage battery and time consumed by RSA and ECIES for different key sizes

112 bits key of ECIES is equivalent to 512 bits key of RSA and 224 bits key is equivalent to 2048 bits key of RSA. We have seen that ECIES decryption with 160 bits key is more efficient than RSA with 1024 bits key. For higher security requirements ECIES proves to be much more efficient for combined encryption and decryption performance. This is so because the key size

grows much gradually for ECIES the key size of RSA for the same increase in the level of security.

4.1.3 Different Devices

The Pocket PC and Handheld PC are considered in this experiment. The size of the file processed is 1 MB since these devices have much lower resources than the laptop. Results for Pocket PC are as follows:

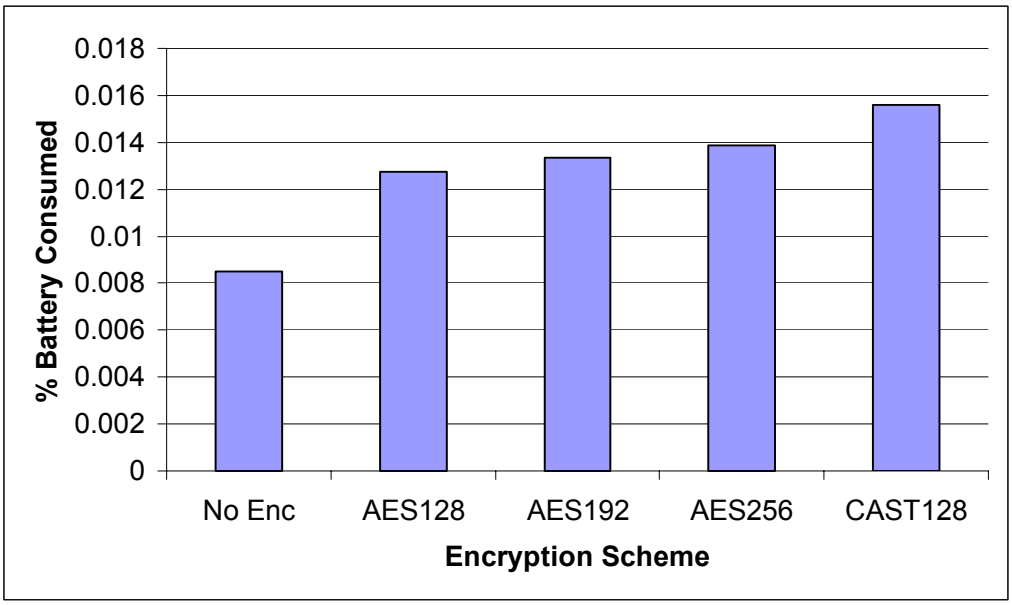


Figure 23: Percentage Battery Consumed by symmetric key schemes without transmission on Pocket PC

On the Pocket PC, CAST128 consumes 22.14 % more power than AES128. Also encrypting data with AES128 is 50% more costly then just processing plaintext data. AES is still more efficient than CAST. AES256 consumes 9% more energy than AES128 and AES192 consumes 5% more energy. RSA 1024 encryption is 29 times more costly than AES 128 encryption the Pocket PC. The results for RSA were shown separately in a table for better comparison of the symmetric key schemes in the graph.

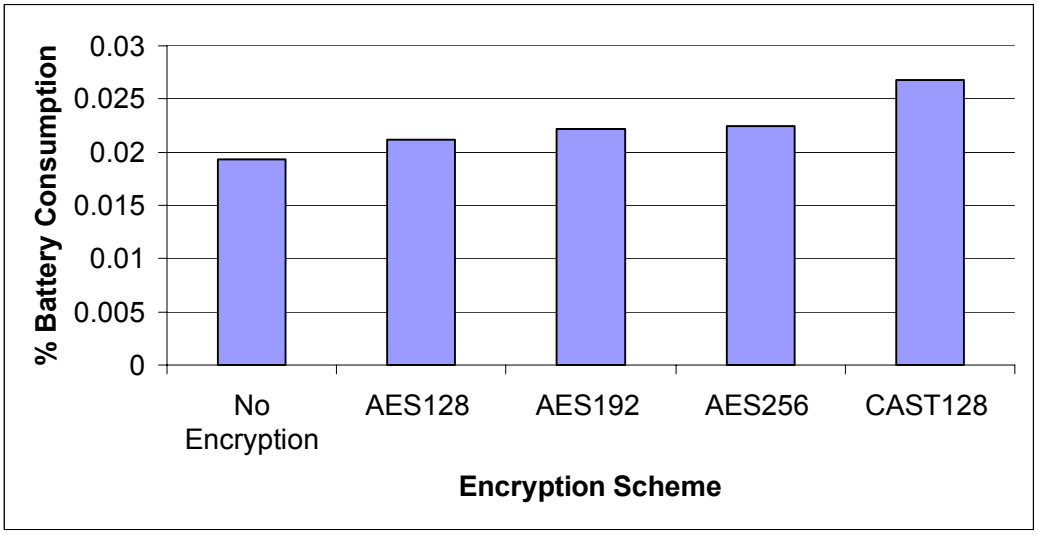


Figure 24: Percentage Battery Consumed by symmetric key schemes without transmission on Handheld device

On Handheld HP Jornada 720 CAST128 consumes 26% more battery than AES128. Encryption is 10% more costly than access just accessing data. AES192 consumes 4% and AES256 about 6% more battery than AES128.

Encryption	**Satellite1200 (laptop)**		**IPAQ HP3870 (Pocket PC)**		**HP Jornada 720 (Handheld)**	
	% Battery	**Time (ms)**	**% Battery**	**Time (ms)**	**% Battery**	**Time (ms)**
No Encryption	0.0004678	35.3016	0.008502	1632.53	0.019363	2253.16
AES128	0.00082	58.3904	0.012755	2391.46	0.021205	2266.93
AES192	0.0008826	62.4436	0.013367	2348.32	0.022182	2313.89
AES256	0.0009526	67.2373	0.013860	2391.46	0.022429	2358.4
CAST128	0.001293	99.4639	0.015578	2748.45	0.026782	2729.58
RSA 1024 (encrypt)	0.2372	17039.9	1.944444	361459.46	2.273268	357280.0
RSA 1024 (decrypt)	6.2727	462858.09	27.5	5114000	33.5	4453000

Table 10: Performance of Encryption Schemes on Laptop, Pocket PC and Handheld

When compared to a laptop encryption 1MB on the laptop with AES128 would consume 0.00082% battery, which implies that on the Pocket PC AES128 is 16 times costlier operation and 26 times costlier on handheld HP Jornada 720. The comparison made here is just in terms of the percentage battery and not the amount of energy consumed. The energy consumed on each device may give very different results but for practical applications the percentage battery consumed figure should be a much relevant figure in making a decision about system design. In terms of time consumed Pocket PC takes 41 times and handheld takes 38.82 times more time to encrypt with AES128 encryption. The time consumed by handheld and Pocket PC is nearly the same. This is because the IPAQ was used with the expansion pack, which effectively provides the IPAQ with more battery power while that is not the case with the Handheld computer. Without the expansion pack the performance is expected to be similar for Pocket PC and Handheld device. The HP Jornada 720 has a 206 MHz 32-bit StrongARM SA1110 processor with 32 MB SDRAM that operates at 51 MHz with display area of 82.5 square centimeters. IPAQ HP 3870 also has a 206 MHz 32-bit StrongARM SA1110 processor but with 64 MB SDRAM at 100 MHZ [47] and a display area of 48 square centimeters. It can bee seen that handheld has about twice the display area as Pocket PC and half the memory with the same processor. The access to data in Handheld computer in much slower than that of Pocket PC because of the slower RAM and hence there is a difference in the performance of the data access with no encryption scenario.

4.2 Wireless Environment

We consider the effect of changes in the communication environment by changing packet sizes, signal to noise ratio, and layer where encryption occurs.

4.2.1 Data Transmission

Figures below show comparison graphs for the same schemes with transmission of the data over the network. It has been noticed that the size of the data unit transmitted per unit time affected the results hence to remove that factor of variability from consideration two units of data were

combined in the case of CAST and IDEA so that same amount of data is transmitted for all the scheme i.e., a data unit of 16 bytes was maintained for all schemes.

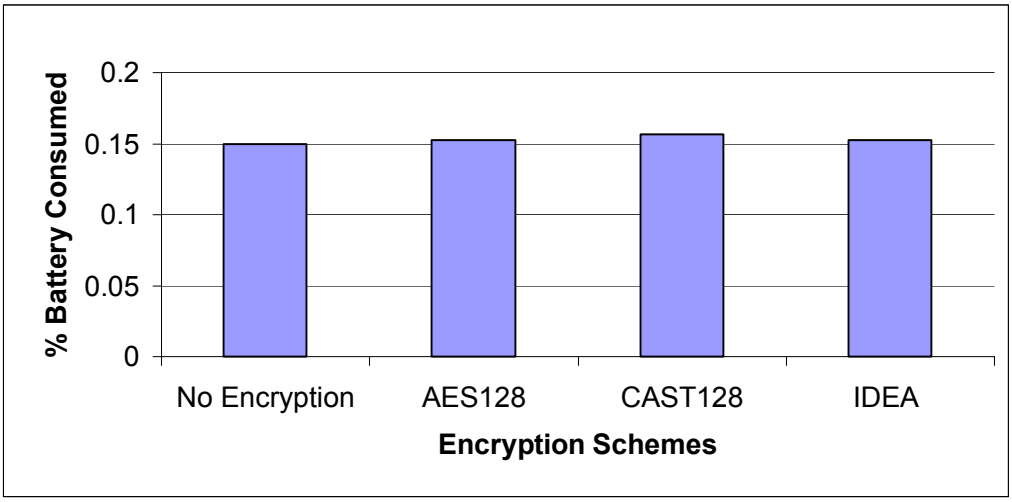


Figure 25: Percentage Battery Consumed by symmetric schemes with data transmission

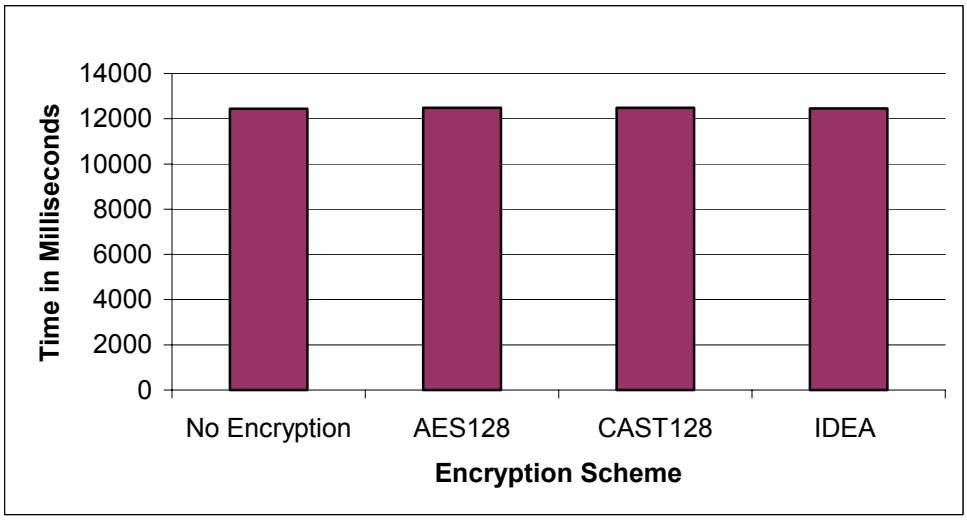


Figure 26: Time Consumed by symmetric key schemes with data transmission

The above graphs show the battery and time consumed per iteration over a 5MB file. From the chart it may be concluded that the encryption scheme used makes almost no significant difference in the energy and time consumption of encryption schemes. It can be seen that if we consider the 90% confidence interval for each of the case above we cannot say that either of the case is better than other. Most of the energy and time goes into transmission of data rather than encryption and since encryption and transmission are dependent on different resources the task that takes max time and resources dominates the results i.e., in this case transmission.

The significance of the particular algorithm used would only come into picture when there are other computation intensive processes running parallel with the application using secure communication at application level, which by the data should be negligible. It is estimated that in a multi-threaded environment when the contention for computation resources reaches a stage where data transmission can become faster than data access and encryption the importance of the encryption scheme used will be realized. In any case AES128 appears to be a better choice.

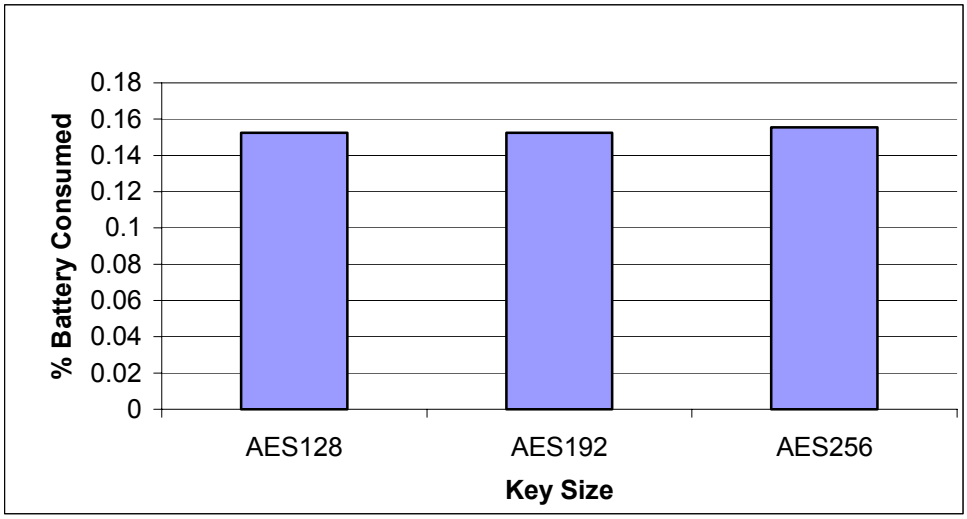


Figure 27: Percentage battery consumed by different AES Key Sizes with data transmission

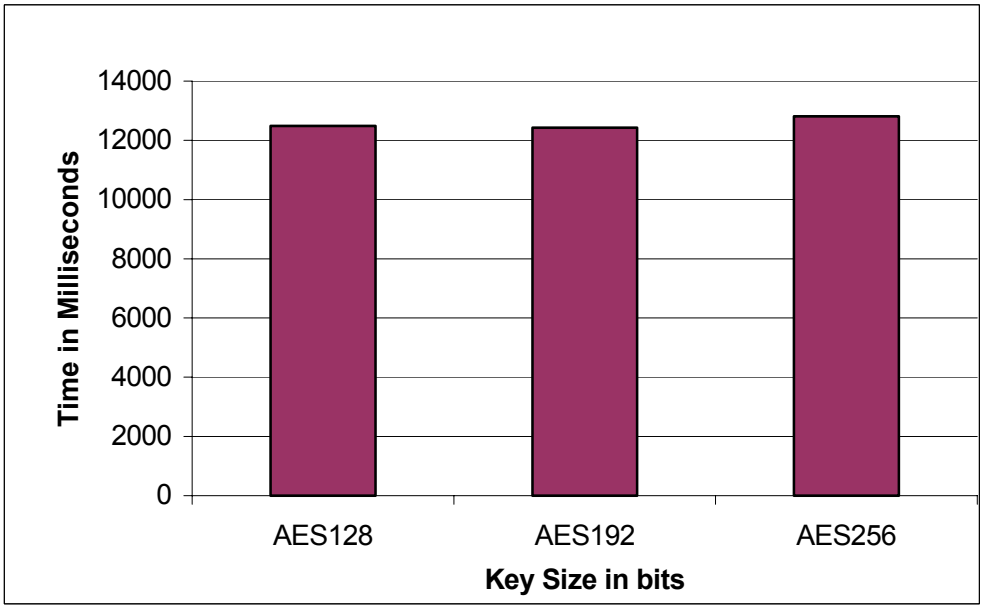


Figure 28: Time Consumed by different AES Key Sizes with data transmission

It can also be seen that with data transmission there is no significant difference between different key sizes for the AES scheme. The reason for this is attributed to the fact that data transmission requires much more battery than encryption. Hence, the additional computational cost due to increased key size, casts less effect on the battery and time consumption.

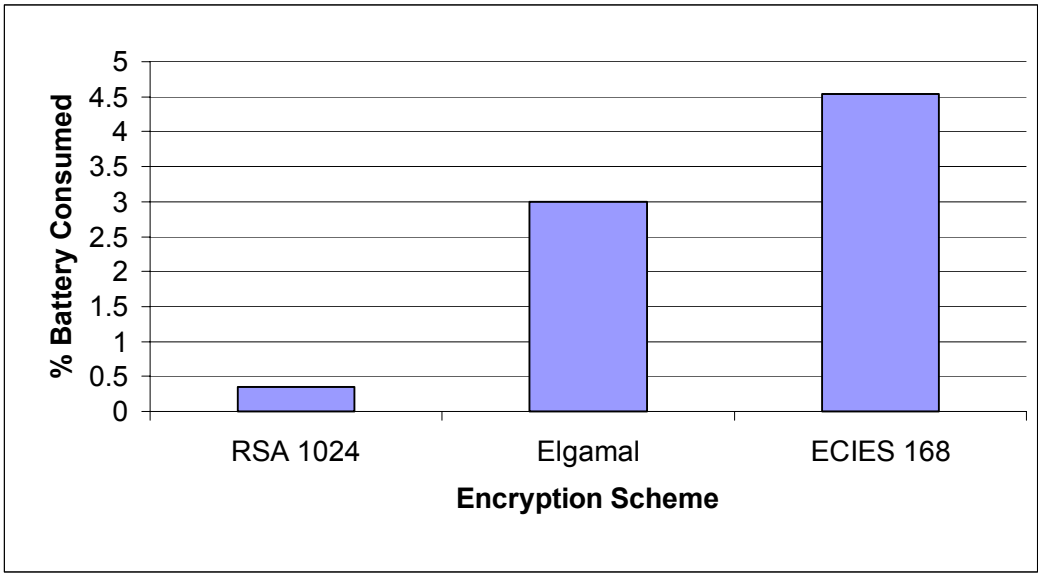


Figure 29: Asymmetric Key Schemes Percentage Battery Consumption with data transmission

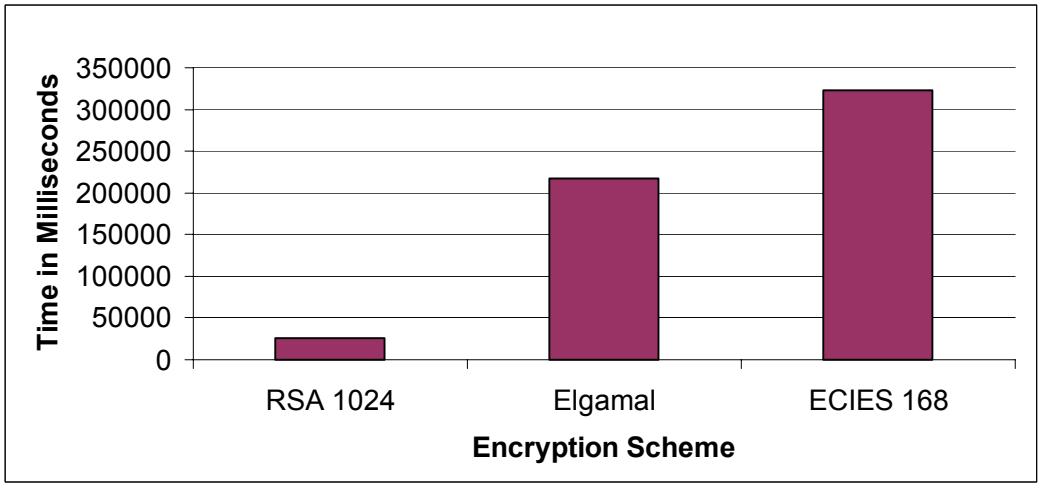


Figure 30: Asymmetric Key Time Consumption with data transmission

The figures 29 and 30 show the performance of the RSA, ElGamal and ECIES asymmetric key schemes with data transmission with 1 MB file. It follows that there is a slight increase in the

battery and time consumption requirements when compared to encryption without data transmission. It can thus be concluded that the asymmetric key scheme for data transmission is an important choice. In practical applications the asymmetric key schemes are expected to be used in parallel with symmetric key schemes. ECIES has inherent hybrid implementation. In case of RSA and ElGamal the user has to decide a hybrid scheme to be used with the asymmetric key scheme. The cipher size is 128 bytes for RSA 1024, 256 bytes for ElGamal 1024 and 62 bytes for ECIES 168. Also the public key is 129 bytes for RSA, 408 bytes for ElGamal and 121 bytes for ECIES [19].

The impact of choice of the asymmetric key scheme in such hybrid asymmetric key schemes for key setup is expected to be small when hue amount of data is being transferred. The reason for this statement is that the graphs show the encryption of a 1 MB file but practically the amount of data that would be encrypted would be 16 bytes during the symmetric key set up for 128 bits key.

The asymmetric key scheme can however play an appreciable role in a hybrid scheme used for secure wireless communications when there is frequent key set up required. Also, not all applications transfer 1 MB of data. Many sessions would just require kilobytes of secure data transfer. In such cases the asymmetric key scheme would be a crucial choice. In is important that we consider the decryption performance of the schemes too. Because in setting up a secure communication session the encryption as well as the decryption process would be used. It is seen that the process of RSA decryption is 40 times more time consuming than data transfer with AES encryption. Added to that in protocols where the sender would have to wait till the decryption is acknowledged it leads to increased battery consumption on the sender's computer due to idle time.

In such cases when the communications is between the wireless device and a wired PC or Server where the public key decryption does not have to be acknowledged immediately before data transfer, it is preferred that we have RSA based encryption on the wireless device for transmission as ElGamal encryption with data transmission consumes 8 times more battery compared to RSA. While for decryption, we should use the ElGamal scheme as RSA scheme

consume about two times more battery power. When the communications is between two wireless devices ElGamal would be preferred because both devices would then use equal amount of resources.

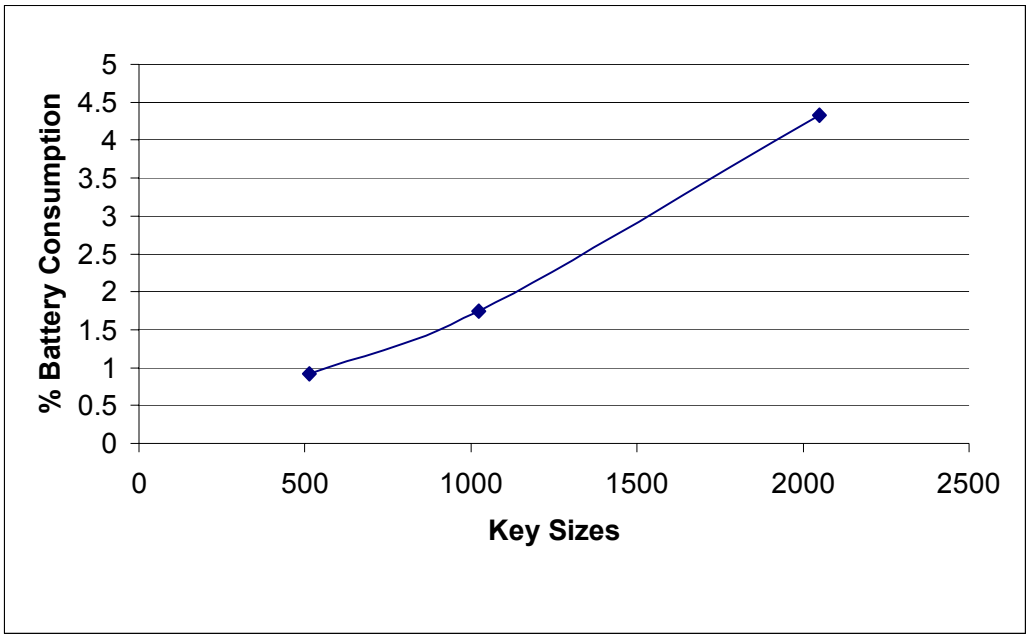


Figure 31: Percentage Battery Consumed with Different Key Sizes for RSA with data transmission

When data transmission is also considered in case of RSA key sizes that the increase in battery and time consumption is much more compared to without data transmission. This is because as the key size increases the size of the encrypted data also increases. The encrypted data block is 64 bytes for 512 bits key, 128 bytes for 1024 bits key and 256 bytes for 2048 bits key. This increase cipher block and requires more energy and time for transmission of the encrypted information. The choice between 2048 and 1024 bits key becomes even more significant in case of data transmission.

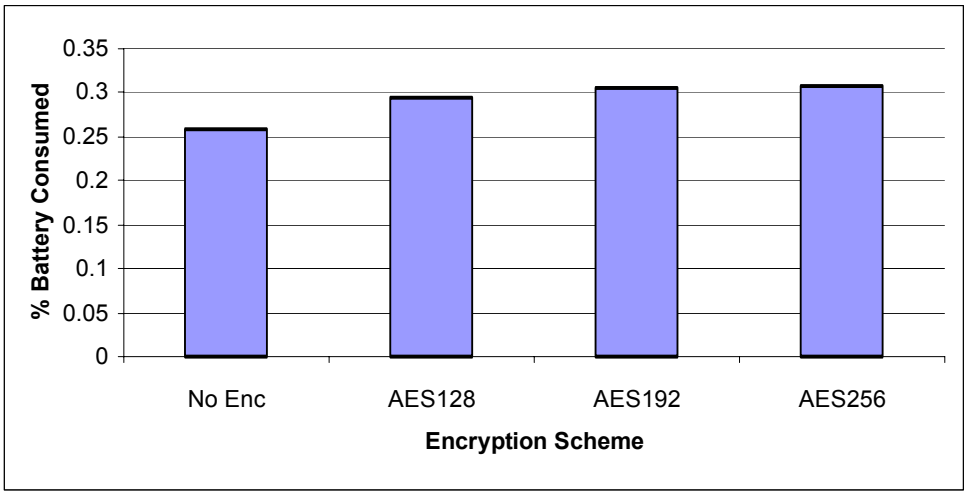


Figure 32: Percentage Battery Consumed by symmetric key schemes with data transmission on Pocket PC

In case of encryption with data transmission some amount of variation can be seen with increasing key size but the variation is less than the standard deviation of variance of the obtained results. Thus statistically it cannot be concluded that battery consumption increases by noticeable percentage with increasing key size with data transmission for Pocket PCs. However, it can be observed that encryption of data does have some impact on the amount of battery consumed even thought the impact is small.

4.2.2 Signal to Noise Ratio

The figures below show batter consumption when we transmit data under different signal to noise conditions. As it was not possible to maintain a constant signal to noise ratio the signal to noise ratio shown in the graph is the time average value of the signal to noise ratio over the period of the experiment.

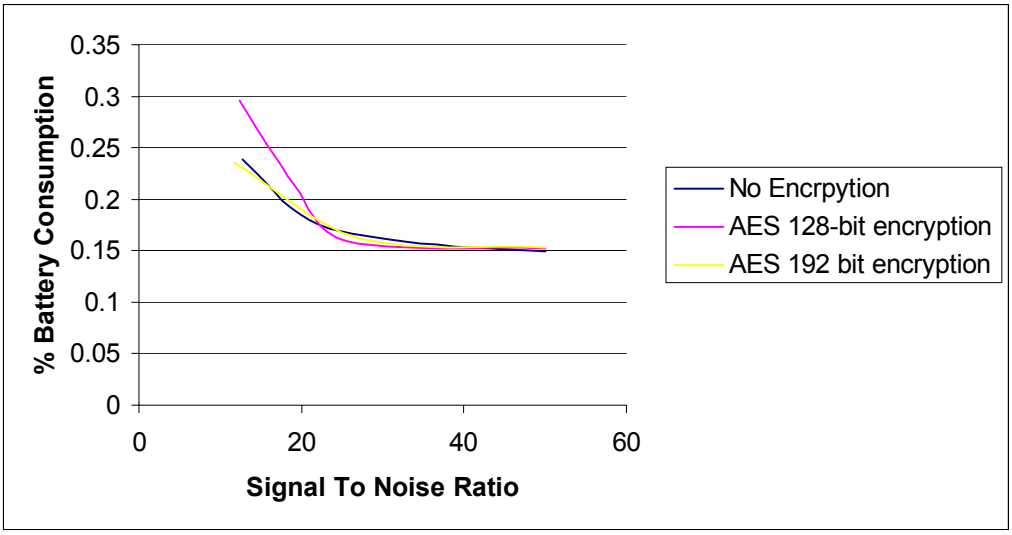


Figure 33: Percentage Battery Consumed with different signal to noise ratio

From the graph it can be concluded that as signal to noise ratio reduces the battery required for data transmission increases in a nonlinear fashion. Above 25dB signal to noise ratio the battery consumption is nearly constant. Results below an average of 10 dB SNR were not collected because it was difficult to maintain the TCP sessions at such a low signal to noise ratio. No definite conclusion can be made that the process of symmetric key encryption actually has some impact on the power consumption under different signal to noise ratios.

4.2.3 Layer of Encryption

Current implementation of WEP requires encryption at the hardware layer. The figure below shows battery and time consumption if the encryption is moved to application layer. WEP encryption, which is a symmetric key stream cipher algorithm, has been explained in the literature review chapter. WEP algorithm code is implemented in the hardware. AES encryption has been considered at the application software layer. Further AES encryption is 128 bits block cipher that is much more secure than 64 bits key encryption used in WEP.

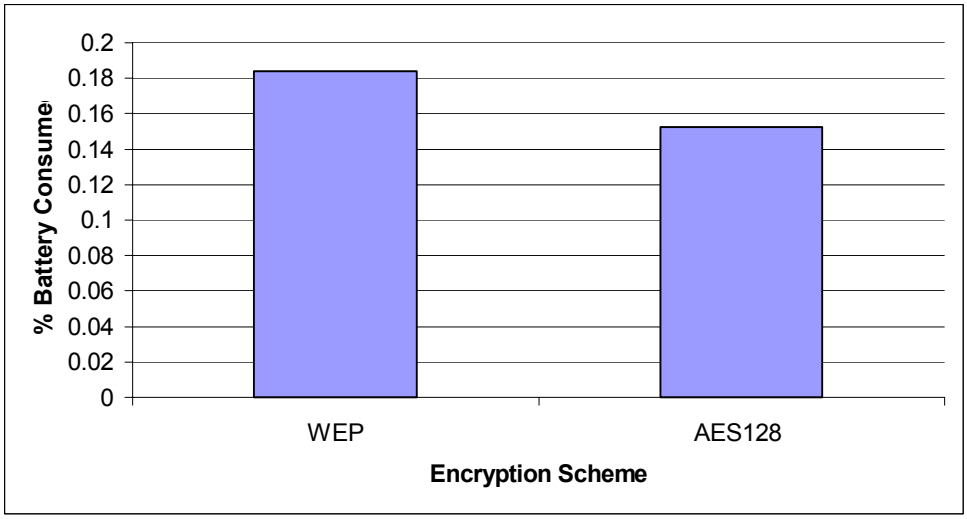


Figure 34: Percentage Battery Consumed by WEP and AES at application software level

Encryption with AES scheme at application software level yields in 18% less battery consumption and 20% less time consumption, when compared to WEP. The reason for this is that WEP transfers an additional header per packet indication the initial vector. Furthermore, encrypting with AES at application software level gives the system the control to switch encryption ON of OFF depending on the requirements. Although this feature would increase the complexity of the system but it would be highly desirable in resource conservative environments.

The two options are run at different layers in the OSI stack. The security provided by the two schemes is different. At the application layer or the transport layer where we expect the software code to be executed, it is possible to secure the data information that is being communicated between the two systems. However a passive observer can always detect that information is being communicated between the two systems, as the IP address part of the packets being transferred would not be encrypted. This allows the possibility of traffic analysis. WEP provides encryption at the link layer thus encrypting all the data from the above layers. It can thus provide access control facilities as explained in the literature review, which cannot be provided by application layer security.

To ensure traffic analysis cannot be done on the information being communicated it would be necessary to ensure that link encryption is set between all the links between the sender and the

receiver. Thus packet must be encrypted and decrypted each time it enters a packet switch. It is possible that some of the intermediate nodes may have compromised security levels. Some applications like credit care transactions, banking operations and financial services would want to make sure that such a possibility does not exist. In such case applications would prefer encrypting and decrypting the essential information between end points such that it cant be decrypted on the intermediate nodes. In such cases the block encryption provided by application layer encryption with AES may be preferred. Application layer encryption provides assurance to the receiver that the data that it has received was actually sent from the intended sender and assures the sender that only the intended recipient can decrypt the data.

In fact, some applications may desire encryption at both layers for higher security. The sender first encrypts the information at the application layer. Then this information is again encrypted at the link layer. On its way to the recipient the information is decrypted and re-encrypted at the link layer. The actual information inside the packets is never revealed at the switches. The receiver then decrypts the packet at the link and the application layer. The information transferred is thus secure between communication ends.

WEP has overheads of message authentication and IV added to the packets that are transferred. The security provides by WEP with 64 bits is much less compared to 128 bits AES encryption. Also WEP encrypts data by the XOR operation, which is inherently less secure. It is possible to flip data packet bits and predictably change the information such that the same encryption key is in use. Using AES at the link layer may prove to be useful in increasing the security level of the systems. However, arrangement would have to be made to provide a better level of security. The use of message authentication code intends to ensure that the packet has not been modified in the air before reaching the receiver. The message authentication based on CRC-32 is not very secure as it is possible to make controlled modifications to the data without affecting the checksum. WEP protocol requires the use of IV that acts as a nonce every time the packets are transferred between wireless stations. This ensures that the replay attack can be prevented. The added bits, however, do result in increased battery consumption for secure information transfer especially in case of smaller packets. Designer of a system using AES at the link layer would need to consider similar mechanism to obtain equal or better level of security.

4.2.4 Changing Packet Size

In wireless communications the packets have headers of each layer. For smaller packet sizes the efficiency of information transfer is low because the ratio of the number of bits in header to number of bits of information is lower compared to the ratio in case of larger packets. Figure below shows the graph of battery consumption to packet size that was sent from application layer to the underlying TCP layer.

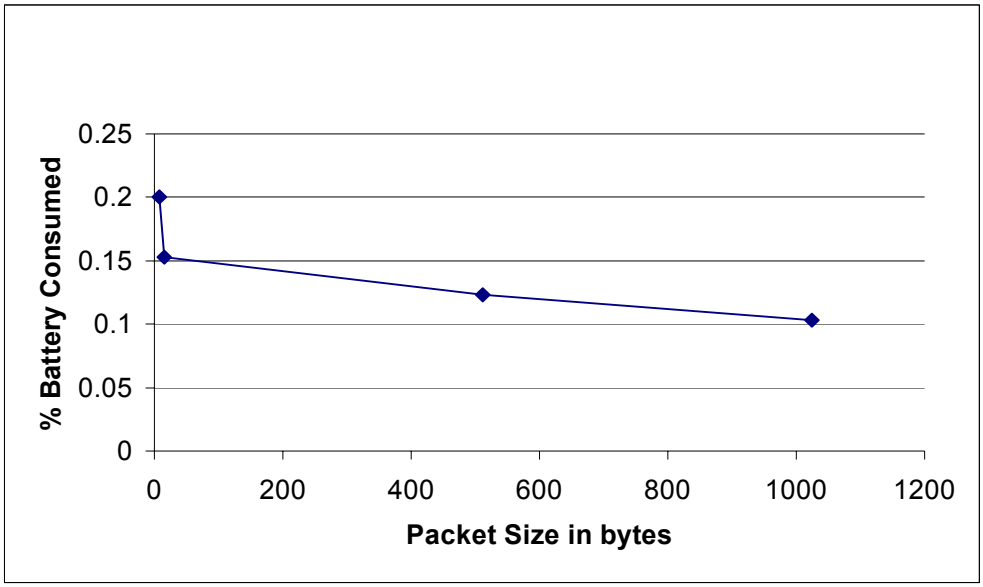


Figure 35: Percentage Battery Consumption with different Packet Size

As can be seen the battery consumption is much high for packet sizes below 10 bytes. As we go to larger packet sizes the battery consumption starts to normalize. It is hence desired that applications aggregate data as much as possible before transferring it to the underlying layer for transmission. The performance in Figure 35 was obtained for signal to noise ratio greater than 45 dB. It is expected that the performance characteristics will be different with less signal to noise ratio as the probability of error increase. Such a study is however suggested for future work and is not covered in this thesis.

Chapter 5
Conclusions and Future Work

It is seen that AES is faster and more energy efficient than IDEA and CAST. When transmission of data is considered there is negligible difference in performance of different symmetric key schemes as most of the resources are consumed for data transmission rather than computation. Even under the scenario of data transfer it would be advisable to use AES scheme incase the encrypted data is stored at the other end and decrypted multiple number of times. Increasing the key size by 64 bits of AES leads to increase in energy consumption by about 8% without any data transfer and with data transfer the difference is not noticeable. Thus real time applications were data is just transferred between systems and not stored for future retrieval may prefer to have higher security provided by larger key size. Reducing the number of rounds leads to power savings but it makes the protocol insecure for AES and should be avoided. Seven or more rounds can be considered fairly secure and could be used to save energy in some cases.

In case of asymmetric key systems for currently recommended key sizes, RSA encryption is the most efficient compared to ElGamal and ECIES. ElGamal is better than ECIES for encryption. For decryption ElGamal is the most efficient. Thus in applications that require decryption multiple numbers of times ElGamal would prove to be useful. ECIES decryption is better than RSA. For higher security requirements in the future ECIES would be much more efficient than RSA and ElGamal when combined performance of encryption and decryption is considered. Even with data transfer the asymmetric key scheme used dominates the energy consumption. Thus the choice of asymmetric key scheme for wireless communication is important. The process of encryption and decryption with Asymmetric key schemes is atleast 100 times more costly than symmetric key schemes.

On handheld and Pocket PC devices the encryption operation is much more expensive than on the laptop. It is 20 times more expensive to do AES encryption on Pocket PC and about 26 times on Handheld in terms of battery consumption. The size of the key has lesser effect on these devices. Data transmission is 20 times more expensive than just encrypting the data without

transmission on Pocket PC while for laptop it's 200 times more expensive than without transmission. Data transmission is however less expensive in laptops than Pocket PC in terms of percentage battery consumed per data byte transferred although the actual energy consumed may be the same.

Implementing encryption at the software level results in considerable power savings upto 18%. Encrypting the message at software level with AES rather than WEP helps us get rid of the requirement to transmit the initial vector and also overcome the inherent insecurities of stream cipher. However, the security provided by the message integrity check of WEP was not included in the comparison. The design of a protocol utilizing block cipher encryption is left for future work.

Signal to noise ratio doesn't have any significant impact on the amount of battery consumed by the encryption scheme as can be seen from the result. When signal to noise ratio falls below 20dB the battery consumption starts to increase rapidly. Wireless LAN communications systems should try to maintain an SNR of atleast 20dB in order to achieve significant power sayings. Also the packet size should be kept large to make the communication efficient. Strategies like aggregation should be adopted whenever possible. Packet sizes should be atleast more than 10 bytes. Applications should aggregate data and transmit it as one packet as far as possible.

In the future, optimizing the encryption schemes for wireless devices can be considered. The security of encryption at various levels of the OSI stack and its performance implications can be studied in further details. The performance characteristics of encryption schemes obtained in this research can be used to modify the existing protocols like SSL and IPSec for the wireless environment. Such a study could further be extended to develop security protocols for hybrid wireless networks. The performance of elliptic curve on handheld and pocket PCs could be studied, as it is expected that the performance of RSA exponentiation operation and elliptic curve multiplication operations would be different on the smaller devices. The performance equations derived for various encryption schemes could be used to develop simulation models to study the performance of encryption schemes on wireless devices and verify the results obtained in this research.

APPENDIX A

Results on the Laptop

Algorithm	File Size	Battery Life per 5MB (%)			Time per 5MB (msec)		
		Average	Stand. Dev	Sample Size	Average	Standard Dev	Sample Size
No Encryption without transmission	5 MB	0.002339	0.000110	70	176.507996	6.55092	3003
AES128 without transmission	5 MB	0.004099	0.000202	70	291.952271	16.543851	1713
AES192 without data transmission	5 MB	0.004413	0.000216	70	312.218018	11.336117	1591
AES256 without data transmission	5 MB	0.004763	0.000266	70	336.186768	15.503484	1476
CAST128 without data transmission	5 MB	0.006468	0.000423	70	497.319611	13.704752	1089
IDEA without data transmission	5 MB	0.007871	0.000426	70	612.023499	12.488589	893
No Encryption with data transmission	5 MB	0.149573	0.052860	52	12438.7636	158.216370	53
AES128 with data transmission SNR=50	5 MB	0.152506	0.053720	51	12483.6132	85.205696	52
CAST128 with data transmission	5 MB	0.156463	0.054610	49	12483.5263	59.184216	50
IDEA with data transmission	5 MB	0.152174	0.049953	46	12463.3681	59.967556	47
AES192 with data transmission SNR=50	5 MB	0.152506	0.053720	51	12433.8417	61.833210	52
AES256 with data transmission	5 MB	0.155556	0.054433	50	12806.3662	68.565895	51
RSA1024 without data transmission	5 MB	1.186441	0.389462	59	85194.5156	178.712845	60
RSA2048 without data transmission	5 MB	2.916667	0.276385	24	208738.516	601.349976	25

Algorithm	File Size	Battery Life per 5MB (%)			Time per 5MB (msec)		
		Average	Stand. Dev	Sample Size	Average	Standard Dev	Sample Size
RSA1024 with data transmission	5 MB	1.75	0.433013	40	129654.24	1517.48437	41
RSA2048 with data transmission	5 MB	4.333333	0.471404	9	328717.123	14.262969	9
RSA512 without data transmission	5 MB	0.523809	0.120468	70	37583.035	162.182083	139
RSA512 with data transmission	5 MB	0.913043	0.189517	69	65370.950	432.719	82
No Encryption SNR=12.762	5 MB	0.238462	0.08356	13	26709.87	2657.26	13
No Encryption SNR=20.938	5 MB	0.18	0.07024	30	17395.81	5782.815	30
No Encryption SNR=32.81	5 MB	0.159091	0.049167	44	13560.41	13560.41	45
No Encryption SNR=50	5 MB	0.149573	0.052860	52	12438.76	158.2163	53
AES128 SNR=12.365	5 MB	0.295652	0.085863	23	33727.58	6676.237	23
AES128 SNR=19.063	5 MB	0.21406	0.04146	32	21406.93	2412.26	33
AES128 SNR=25.81	5 MB	0.159091	0.049167	44	13329.43	206.54847	45
AES128 (4 rounds) without data transmission	5 MB	0.003606	0.000219	70	257.257874	24.025738	1951
ElGamal encryption	1 MB	2.916667	0.41574	9	210631.234	2206.277	10
ECIES 168 encryption	1 MB	4.533333	0.49888	15	323163.437	4770.0473	16
AES192 SNR=11.869	5 MB	0.236000	0.055714	25	24391.7949	3744.0517	25
AES192 SNR=27.3	5 MB	0.16207	0.042133	39	14686.2	211.98414	40
AES128 (7 rounds)	5 MB	0.003972	0.000436	70	291.27449	43.4701	1784
ECIES with secp160k1 curve encryption	1 MB	4.0588	0.235294	17	300085.375	232.84497	18
ECIES with secp224k1 curve encryption	1 MB	7.0000	.447214	10	514744.718	172.63105	11
ECIES with secp112r1 curve encryption	1 MB	1.891892	0.310551	37	140375.26	392.91397	38
ECIES with secp160k1 curve decryption	1MB (decrypted)	5.076922	1.54	13	409530.37	1565.04	13

Algorithm	File Size	Battery Life per 5MB (%)			Time per 5MB (msec)		
		Average	Stand. Dev	Sample Size	Average	Standard Dev	Sample Size
RSA 1024 decryption without data transmission	1 MB (decrypted)	6.272727	0.445362	11	462858.09	262.0228	12
RSA 1024 decryption without data transmission	1 MB (decrypted)	39.5	NA	2	2945625.5	NA	2
ElGamal 1024 decryption without data transmission	1 MB (decrypted)	3.526316	0.499307	19	262782.34	361.68991	20
ECIES with secp224k1 curve decryption	1MB (decrypted)	8.25	0.433	8	614629.43	148.3981	9
RSA 512 decryption without data transmission	1MB (decrypted)	1.272727	0.445363	55	93602.109	490.0623	56
ECIES with secp224k1 curve decryption	1MB (decrypted)	2.2580	0.43757	31	165857.56	117.1206	32

Results on the Pocket PC

Algorithm	File Size	Battery Life per 5MB (%)			Time per 5MB (msec)		
		Average	Stand. Dev	Sample Size	Average	Standard Dev	Sample Size
AES192 without data transmission	1 MB	0.013367	0.001822	71	2348.32	87.9847	1045
AES256 without data transmission	1 MB	0.013860	0.00188	71	2391.46	51.6263	1008
AES128 without data transmission	1 MB	0.012755	0.0007	70	2311.39	99.348	1106
RSA1024 without data transmission	1 MB	1.944444	0.328671	36	361459.468	2890.7551	37
CAST without data transmission	1 MB	0.015578	0.007187	70	2748.4582	87.976646	908
No encryption without data transmission	1 MB	0.008502	0.02349	74	1632.5339	73.813	1543
AES192 with data transmission	1 MB	0.304348	0.099905	46	54561.70	354.036	47
AES256 with data transmission	1 MB	0.306667	0.099778	45	55460.871	747.017	46
AES128 with data transmission	1 MB	0.295157	0.0986	39	52710.00	1964.66	40
No Encryption with data transmission	1 MB	0.259259	0.091325	54	48000.00	439.834	55
RSA 1024 decryption	1 MB (decrypted)	27.5	NA	2	5114000.0	NA	2

Results on the Handheld

Algorithm	File Size	Battery Life per 5MB (%)			Time per 5MB (msec)		
		Average	Stand. Dev	Sample Size	Average	Standard Dev	Sample Size
AES128 without data transmission	1 MB	0.021205	0.006746	4	2266.93	94.376945	735
AES192 without data transmission	1 MB	0.022182	0.007496	4	2313.89	99.030289	727
AES256 without data transmission	1 MB	0.022429	0.007468	4	2358.40	81.871033	702
CAST128 without data transmission	1 MB	0.026782	0.010479	4	2729.58	95.524307	622
No Encryption without data transmission	1 MB	0.019363	0.005631	4	2253.16	96.423126	805
RSA 1024 Encryption without data transmission	1 MB	2.273268	0.465741	4	357280.00	1183.8918	25
RSA 1024 decryption without data transmission	1 MB (decrypted)	33.5	NA	2	4453000	NA	2

APPENDIX B

Driver Code for Encryption

```
#include <afxinet.h>
#include <iostream.h>
#include <rijndael.h>
#include <stdio.h>
#include <string.h>
#include <io.h>

#define block 16
// The block size for AES is 128 bits
USING_NAMESPACE(CryptoPP)
USING_NAMESPACE(std)

void main(int argc, char *argv[]) {

        unsigned char data[block],cipher[block];
        static const byte *const key=(byte
*)"abcdef01234567899876543210fedcba0123456789";
        char logFile[20];
        int i,rep=10,size=16;
        long fileSize = 0;
        long cnt =0;
        FILE *fp;
        SYSTEM_POWER_STATUS pStatus;
        SYSTEMTIME stime;
        FILE *fin, *fout;

        GetSystemTime(&stime);
```

```
	if( argc != 5 ) {
		printf("Usage: %s <-e/-n> <keySize> <noOfRepetition> <filename>",argv[0]);
		exit(1);
	}

	if( argv[1][1] != 'e' && argv[1][1] != 'n') {
		printf("Usage: %s <-e/-n> <keySize> <noOfRepetition> <filename>",argv[0]);
		exit(1);
	}

	fin = fopen(argv[4], "rb");
	if( fin == NULL ) {
		printf("%s could not be opened",argv[4]);
		exit(1);
	}
	fclose(fin);
	// Open a excel file with the current day and
	// time as its name so we know when it was made
	sprintf(logFile,"%02d%02d%02d%02d.csv",stime.wMonth,stime.wDay,stime.wHour,sti
me.wSecond);
	fp = fopen(logFile,"w");
	fprintf(fp,"%s %s %s %s\n",argv[0],argv[1],argv[2],argv[3]);

	rep=atoi(argv[3]);
	// Get the number of repetition passed by thr user
	size=atoi(argv[2]);
	RijndaelEncryption encrypt(key,size);
	//Initialize the encrypting software with the passed key size

	GetSystemPowerStatus(&pStatus);
	GetSystemTime(&stime);
```

```
fprintf(fp,"%ld,%ld,%d.%d.%d.%d\n",cnt,pStatus.BatteryLifePercent,stime.wHour,
stime.wMinute,stime.wSecond,stime.wMilliseconds);

while(pStatus.BatteryLifePercent>=30) {
        // Execute the loop till battery power falls to 30%
        for(i=0; i<rep; i++) {
                // Repeat iterations

                fin = fopen(argv[4], "rb");
                fout = fopen("aes.enc","wb");

                while( !feof(fin) ) {

                        fread(data,sizeof(unsigned char),block,fin);
                        if( argv[1][1] == 'e' )
                                encrypt.ProcessBlock(data,cipher);
                        // This is where the data is encrypted. One block at a time
                        fwrite(cipher,sizeof(unsigned char),block,fout);
                }
                fclose(fin);
                fclose(fout);
        }
        cnt+=i;
        GetSystemPowerStatus(&pStatus);
        GetSystemTime(&stime);

fprintf(fp,"%ld,%ld,%d.%d.%d.%d\n",cnt,pStatus.BatteryLifePercent,stime.wHour,
stime.wMinute,stime.wSecond,stime.wMilliseconds);
        //Print the status into excel log file
        printf("Runs %ld,Battery life left %ld\n",cnt,pStatus.BatteryLifePercent);
        fflush(fp);
```

```
            fflush(stdout);
            //Do this so that we dont lose data
        }

        fclose(fp);
}
```

BIBLIOGRAPHY

[1] WLANS: Wireless Dream, Security Nightmare, Dermot McGrath, Broadband Wireless Business Magazine, Vol. 3, No. 8, January/February 2003

[2] Wireless LANs: Global Market Demand and Opportunity Assessment, InfoTech, PBI Media, Jan 2002

[3] William Stallings, 'Cryptography and Network Security', Prentice Hall Publication, 1999

[4] Wireless LAN Security: A Short History, Matthew Gast. 2002

[5] Nikita Borison (UC Berkeley), Ian Goldbery (Zero-Knowledge Systems), and David Wagner (UC Berkeley), 'Intercepting Mobile Communications: The Security of 802.11', 2001

[6] McMurry Mike, Wireless Security, Jan 22, 2001

[7] What's an LSFR, Texas Instruments, December 1996

[8] On the linear span of binary sequences finite geometries, IEEE Trans. Info. Theory 36 (1990), pg 548-552, A. H. Chan and R. A. Games

[9] IEEE 802.11b, Standard for Wireless LAN

[10] SSL 3.0 specification, http://wp.netscape.com/eng/ssl3

[11] Information Assurance in Wireless networks, Joseph Kabara, Prashant Krishnamurthy and David Tipper, University of Pittsburgh, Fourth Information Survivability Workshop, 2001

[12] Over the Air Security in Wireless Networks: Current Approaches, Technical Report, Tanapat Anusas-amornkul and Prashant Krishnamurthy, University of Pittsburgh, May 2002

[13] Energy-efficient wireless networking for multimedia applications, Paul J.M. Havinga and Gerard J.M. Smit University of Twente, Department of Computer Science, Wireless Communications and Mobile Computing, Wiley, 2001: 1:165-184.

[14] Rijndael FPGA implementation utilizing look-up tables McLoone, W.; McCanny, J.V.; 2001 IEEE Workshop on Signal Processing Systems, 2001 Page(s): 349 –360

[15] The FPGA implementation of the RC6 and CAST-256 encryption algorithms Riaz, M.; Heys, H.M.; Electrical and Computer Engineering, 1999 IEEE Canadian Conference on , Volume: 1 , 1999 Page(s): 367 -372 vol.1

[16] Implementation approaches for the Advanced Encryption Standard algorithm, Xinmiao Zhang; Parhi, K.K.; IEEE Circuits and Systems Magazine , Volume: 2 Issue: 4 , Fourth Quarter 2002 Page(s): 24 –46

[17] Optimizing public-key encryption for wireless clients, Potlapally, N.R.; Ravi, S.; Raghunathan, A.; Lakshminarayana, G.; Communications, 2002. ICC 2002. IEEE International Conference on , Volume: 2 , 2002, Page(s): 1050 -1056 vol.2

[18] Crypto++ Library, http://www.eskimo.com/~weidai/cryptlib.html

[19] PGP in Constrained Wireless Devices, Michael Brown, Donny Cheung, Darrel Hankerson, Julio Lopez Hernandez, Michael Kirkup, Alfred Menezes' 9th USENIX Security Symposium Paper 2000, Pp. 247–262 of the Proceedings

[20] Traffic flow confidentiality security service in OSI computer network architecture, Ramaswamy, R.;TENCON 90. 1990 IEEE Region 10 Conference on Computer and Communication Systems, 24-27 Sep 1990, Page(s): 649 -652 vol.2

[21] A Logic of Authentication, Michael Browns, Martin Abadi, Roger Needham, SRC Research Report, Feb 22, 1990

[22] Tom Austin, PKI: A Wiley Tech Brief, John Wiley & Sons, Inc., 2001

[23] IEEE P1363a and IEEE 1363: Standard Specifications for Public-Key Cryptography.

[24] Bruce Schneier, Applied Cryptography, John Wiley & Sons Inc.

[25] Announcing the Advanced Encryption Standard (AES), Federal Information Processing Standards Publication 197, 26 November 2001

[26] AES Proposal: Rijndael, Joan Daemaen and Vincent Rijmen, March 9, 1999

[27] RSA-OAEP Encryption Scheme, RSA laboratories, 2000.

[28] Wiener, M. J. "Cryptanalysis of Short RSA Secret Exponents," IEEE Transactions on Information Theory 36 (1990), pp. 553-558

[29] The New RSA Factoring Challenge, http://www.rsasecurity.com/rsalabs/challenges/factoring/index.html

[30] Modular multiplication method, J.-H Oh, S.-J. Moon, Computers and Digital Techniques, IEEE Proceedings, Volume: 145 Issue: 4 , Jul 1998, Page(s): 317 –318

[31] The Chinese Remainder Theorem and its application in a high-speed RSA crypto chip, J. Grossschadl, Computer Security Applications, 2000. ACSAC '00. 16th Annual Conference , Dec 2000, Page(s): 384 –393

[32] American National Standards Institute, Accredited Standards Committee X9 Working Draft: ANSI X9.44: Public Key Cryptography Using Reversible Algorithms for the Financial Services Industry: Transport of Symmetric Algorithm Keys Using RSA, American Bankers Association, 1994.

[33] J. Buchmann, J. Loho, and J. Zayer, An implementation of the general number field sieve, Advances in Cryptology - Crypto '93, Springer-Verlag (1994), 159-166.

[34] Michel Adballa, Mihir Bellare, and Philip Rogaway, 'DHAES: An Encryption Scheme Based on the Diffie-Hellman Problem'

[35] J.Daemen, R.Govaerts, J.Vandewalle, "Weak Keys for IDEA", 1993

[36] Report on Development of the Advanced Encryption Standard (AES), Computer Science Division, NIST, U.S Department of Commerce, October 2, 2000

[37] Practical cryptography – the key size problem: PGP after years, Lenka Fibikova and Jozef Vyskoc, 21 December 2001

[38] End User Perspective – Industrial Consumer Electronics Power, Jerry Hallmark, Motorola labs, January 15, 2002

[39] Seiko Instruments Inc., Applications of battery power supply devices and battery drive circuit design

[40] 'OpenSSL Project', http://www.openssl.org/

[41] 'Cryptlib', http://www.cryptlib.orion.co.nz

[42] 'Cryptix[tm]', http://www.cryptix.org

[43] 'Multiprecision Integer Rational Arithmetic C/C++ Library', http://indigo.ie/~mscott

[44] Lithium Ion Cells, Toshiba America Electronics Components, Inc.

[45] FIPS Publication 180-1: Secure hash Standard, National Institute of Standards and Technology (NIST), 1994.

[46] SEC2: Recommended Elliptic Curve Domain Parameters, Standards for Efficient Cryptography, Certicom Research, September 20, 2000

[47] Compaq Computer Corporation devices, http://www.handhelds.org/Compaq

[48] SSH Communications Security, 'Public Key Cryptosystems' http://www.ssh.com/support/cryptography/algorithms/asymmetric.html

[49] Robert Sedgwick, 'Algorithms in C++', Addison-Wesley publishing company

[50] Scott Flubrer, Itsik Mantin, and Adi Shamir, 'Weakness in the Key Scheduling Algorithm of RC4', 2001

[51] David Linden, Handbook of batteries, second edition. McGraw-Hill, Inc. 1994.

[52] Eric Maiwald, Network Security: A Beginner's Guide, Osborne-McGraw Hill, 2001

[53] Wireless Network Security: 802.11, Bluetooth and Handheld Devices, Tom Karyginannis and Les Owens, NIST, U.S. Department of Commerce, November 2002

[54] Improved Cryptanalysis of Rijndael, N. Ferguson, J. Kelsey, S. Lucks, B. Schneier, M. Stay, D. Wagner, and D. Whiting, Seventh Fast Software Encryption Workshop, Springer-Verlag, 2000

[55] Pollard's Algorithm for Discrete logarithm problem, Professor Dr. D. J. Guan, September 2, 2001