ENERGY EFFICIENT SECURITY FRAMEWORK FOR WIRELESS LOCAL AREA NETWORKS

by

Phongsak Kiratiwintakorn

B.Eng. of Electrical Engineering,

King Mongkut's Institute of Technology, Ladkrabang, 1996.

M.S. of Electrical Engineering,

University of Kansas, 2000.

Submitted to the Graduate Faculty of

the Telecommunications Program, Department of Information

Sciences and Telecommunications in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2005

UNIVERSITY OF PITTSBURGH

TELECOMMUNICATIONS PROGRAM,

DEPARTMENT OF INFORMATION SCIENCES AND TELECOMMUNICATIONS

This dissertation was presented

by

Phongsak Kiratiwintakorn

It was defended on

April 15, 2005

and approved by

Dr. Prashant Krishnamurthy, Department of Information Sciences and Telecommunications

Dr. Richard Thompson, Department of Information Sciences and Telecommunications

Dr. James B. D. Joshi, Department of Information Sciences and Telecommunications

Dr. Sujata Banerjee, Hewlett-Packard Lab

Dr. Daniel Mossé, Department of Computer Science

Dissertation Director: Dr. Prashant Krishnamurthy, Department of Information Sciences and Telecommunications

Copyright © by Phongsak Kiratiwintakorn
2005

ENERGY EFFICIENT SECURITY FRAMEWORK FOR WIRELESS LOCAL AREA NETWORKS

Phongsak Kiratiwintakorn, PhD

University of Pittsburgh, 2005

Abstract

Wireless networks are susceptible to network attacks due to their inherent vulnerabilities. The radio signal used in wireless transmission can arbitrarily propagate through walls and windows; thus a wireless network perimeter is not exactly known. This leads them to be more vulnerable to attacks such as eavesdropping, message interception and modifications compared to wired-line networks. Security services have been used as countermeasures to prevent such attacks, but they are used at the expense of resources that are scarce especially, where wireless devices have a very limited power budget. Hence, there is a need to provide security services that are energy efficient.

In this dissertation, we propose an energy efficient security framework. The framework aims at providing security services that take into account energy consumption. We suggest three approaches to reduce the energy consumption of security protocols: replacement of standard security protocol primitives that consume high energy while maintaining the same security level, modification of standard security protocols appropriately, and a totally new design of security protocol where energy efficiency is the main focus. From our observation and study, we hypothesize that a higher level of energy savings is achievable if security services are provided in an adjustable manner. We propose an example tunable security or TuneSec system, which allows a reasonably fine-grained security tuning to provide security services at the wireless link level in an adjustable manner.

We apply the framework to several standard security protocols in wireless local area networks and also evaluate their energy consumption performance. The first and second methods show improvements of up to 70% and 57% in energy consumption compared to plain standard security protocols, respectively. The standard protocols can only offer fixed-level security services, and the methods applied do not change the security level. The third method shows further improvement compared to fixed-level security by reducing (about 6% to 40%) the energy consumed. This amount of energy saving can be varied depending on the configuration and security requirements.

Keywords: Network Security, Wireless Networks, Energy Efficiency, Security Strength, Tunable Security.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

PREFACE

This dissertation would not be successful without the inspiration, encouragement, and support from my advisor, Dr. Prashant Krishnamurthy. He has guided me through the problems in my research, and been there when I needed most. I am also thankful for his help regarding my study. He has always been my inspiration to pursue a Doctoral degree here at University of Pittsburgh since I first met him.

I would like to thank my committee members, Dr. Richard Thompson, Dr. Sujata Banerjee, Dr. James Joshi, and Dr. Daniel Mossé for their time and effort in reading and giving comments to improve the quality of my work. I would like to specially thank to Dr. Mossé for lending me his instrument for my experiments and also guiding me in the research related to energy efficiency.

My dissertation research is supported mainly by Dr. Prashant Krishnamurthy's research funding, including National Institute of Standards and Technology (NIST) Critical Infrastructure Protection (CIP) grant No. 60NANB1D0120 and National Science Foundation (NSF) Federal Cyber Service - Scholarship for Service grant. I would also like to thank Dr. Lewis for his partial financial support during my Ph.D. years. Finally, I would like to thank the Royal Thai Government for giving me a 6-year scholarship that initiates the possibility of having my Ph.D. in the United States.

Five years in Pittsburgh would have not been wonderful without friends, Tanapat, Wasan, Gwyn, Peerapon, Chatree, Wiklom, Kamol, Saowanee, Tavida, Wirun, other Thai students, and volleyball folks in Pittsburgh. They gave me warm feeling like my second home is in Pittsburgh. I would like to thank them for their friendship and supports. I would also like to thank my colleagues in Computer Science Department, Matt Craven and Cosmin Rusu, for their time and help in power measurement. I would like to give my special thank to my beloved Pornpen for her immutable love and care, which gave me strength in the last two years.

Finally, I would like to thank my parents, Chairat and Prinda, my sisters, Laddawan, Tippawan, and Tanyarat, as well as my only brother, Pradya, for their invaluable love. I feel so grateful having them all in my life.

I. INTRODUCTION

In the last two decades many wireless networking projects have been started as initiatives towards a network of a future world without wires. Wireless networks have been rapidly adopted and widely deployed around the world. The most successful wireless network is the cellular phone network which started as a very low speed link with minimum features. Nowadays, it can provide up to 2 Mbps, and it has become the largest and most rapidly grown network with millions of subscribers. Currently, wireless networks have expanded to include a variety of devices such as laptops, personal digital assistants (PDAs), pagers, sensors and wearable computers. Wireless networks are rapidly expanding due to their ability to provide communications with ubiquity and mobility. Without wires, users with wireless devices can move freely and are able to access wireless services anywhere any time. More and more people now rely on small wireless devices to fulfill their tasks. Hence, both the devices and the underlying communications need to be robust to provide reliable services and need to be secure to protect the information they carry.

A. NETWORK SECURITY

Generally, network security is divided into four main categories: confidentiality, authentication, integrity, and non-repudiation [110]. Confidentiality services ensure that exchanged information is accessible only to authorized parties by using *encryption*. It is used to protect against eavesdropping from attackers who overhear transmissions over a wireless channel. Authentication services verify the validity of identity of an intended party, and protect against masquerading or identity spoofing which are attacks to gain unauthorized access to a network. Integrity services are used to confirm that the information has not been modified, duplicated, or re-ordered during a transfer. We can provide information integrity by adding additional information, called *keyed message digests* or *keyed hashes*, that are cryptographically related to the transferred information. The keyed message digests also provide message authentication, where a receiver is able to verify the authenticity of the message origin or generator. The keyed message digest is sometimes called Message Authentication Code (MAC). Non-repudiation services ensure that a receiver can verify the *unique* origin of a message and its creator, or a sender can guarantee to any party that the message has originated from himself, and it has not been modified during transmission. Non-repudiation services are also a combination of message authentication and integrity services.

B. SECURITY IN WIRELESS NETWORKS

Security in wireless networks is of paramount importance. Due to the broadcast nature of the wireless radio signals, wireless networks are implicitly vulnerable to several network attacks. Anyone within the wireless transmission range of a device (including malicious users or attackers) is able to passively listen to or eavesdrop on the signals and could potentially access information from the signals. It is also possible to actively transmit signals that can attack the network. Wireless networks are therefore extremely vulnerable to many kinds of security threats and they essentially need strong countermeasures to overcome those threats.

In the past few years, wireless data networks have been exponentially growing. Many business and information technology applications have relied on wireless data networks such as IEEE 802.11 Wireless Local Area Networks (WLANs). The threats to those networks are also growing. Due to the discovery of vulnerabilities of WLANs in 2001, many business and government sectors have temporarily ceased to adopt WLANs in their networks because they increase threats to their businesses [40].

The new cybersecurity policy from the Department of Homeland security has identified many threats to our national networks and many business and government units have been actively aware of these threats. The threats in wireless networks have also been identified as major threats to our national information security [34]. Subsequently, the National Institute of Standards and Technology (NIST) released a special publication as a guideline for wireless security geared primarily towards wireless local/personal area networks [62].

Providing security is not an option for wireless operators. It is imperative to provide security services to wireless networks. However, providing security for wireless devices is a challenging research topic. Wireless devices have limited resources such as low-speed CPUs, small-sized memory, and importantly limited battery power. Making efficient use of battery power alone has been an interesting research topic [55, 71, 106]. Enabling the efficient use of security services in battery-powered devices is even more interesting and challenging. Security services rely on cryptographic and mathematical functions that are known to be computationally intensive. To deliver security to such resource-limited, battery-powered devices, there is a need for methodologies to efficiently utilize a "reasonable" amount of resources to supply a "reasonable" amount of security.

C. THE MAJOR CHALLENGES

Much research has focused on energy-efficient *communication* protocols for wireless devices. Several protocols have been designed at different layers in the communication protocol stack to save energy. At the link layer, for example, an adaptive transmission control protocol was designed to save energy as described in [121]. In the standard protocol, after sending a data packet, an acknowledgment packet is sent by the receiver to confirm the packet delivery. However, when wireless channel conditions are degraded during transmission, the acknowledgment packet may not be received due to the loss of itself or the transmitted data packet. The adaptive transmission control protocol stops sending data packets, but instead sends short packets or probes to the receiver. Until an acknowledgment of the probe packet is received, the sender keeps sending probes. Once an acknowledgment is received, the sender resumes the normal transmission mode. The idea here is that short probes consume less energy than longer data packets for transmission.

Another example is in high error rate environments where a sender may adjust its error correcting code rate depending on the channel conditions encountered. Using lower rate error correcting codes in low error environments can reduce energy while maintaining channel integrity. By using these methods, energy saving is possible because sending short packets or adjusting the error correcting code reduces energy consumption. These are examples of how a protocol can be adapted or adjusted to dynamic environments in wireless networks.

To our knowledge, there are no adaptive or adjustable *security* protocols or systems that have been suggested for use in wireless networks. Some protocols may offer adjustability or can be used adaptively [43, 63], but none has really been used for the purpose of performance optimization and energy efficiency. This may be due to the fact that these security protocols are implemented for wired-line network systems and thus they do not consider variable network environments and limited resources as factors in their design. Some security systems have also been proposed to offer security adjustability, but none has considered the energy efficiency [21, 22, 90].

For example, let us consider *Kerberos* which is aimed at providing authentication and key exchange services in wired-line distributed networked environments [80]. A client device needs to exchange at least six messages to be authenticated, to get a ticket for access, and to access an application server. Transmission of these message exchanges could considerably deplete the battery power of devices and would also substantially consume the network bandwidth. Therefore, security services provided in wired-line networks can no longer be directly applied as-is to wireless networks due to the difference in network characteristics. Security in wired-line networks also has several assumptions about the networks (such as: the transmission line is assumed to be somewhat physically protected, or devices operating in wired-line networks have unlimited-power and are not resource-limited, etc.). These assumptions cannot be applied to wireless networks and systems. The following are some unique characteristics of wireless networks that are relevant to security protocol design.

First, wireless networks are known to be "open", in which their physical perimeter is unknown. A person trying to access a wireless network may not be inside an authorized perimeter. Additionally, since there is, of course, no wire in wireless networks, anyone including an attacker can easily deploy his own wireless network which can be malicious. An example is the case where an attacker deploys a rogue access point outside a company building to lure company employees to his malicious network to steal valuable information. This attack is known as the "parking lot" attack [25]. Thus, the assumption of trust of any network access point no longer applies to the design of a security protocol for wireless networks.

Second, bandwidth is a limited resource in wireless networks. In wired networks, bandwidth can be expanded by deploying more communication lines. This is not the case in wireless networks and so a wireless network protocol should use bandwidth efficiently. A security protocol that sends too many messages over a wireless link would waste bandwidth and consume unnecessary energy for message transmission.

Third, radio signals are randomly degraded due to environment dynamics. Transmission via radio signals requires protocol synchronization that can deal with the randomly changing environment. A simple security protocol synchronization may not be a solution due to random loss of data. A complex synchronization may be too expensive for small limited devices. Asynchronous transmission may be a suitable solution to security protocol design for limited-resource networks.

Fourth, roaming is a unique service present primarily in wireless networks. Users with wireless devices often roam from one access point to another access point, requiring mobility management mechanisms such as location updates and session handoffs or re-associations. Security service management is also required for such scenarios. Roaming can complicate the process of security service provision, or can probably be too expensive for small devices since the newly associated access point may or may not be as secure as the previously associated one. Wireless devices may need to start over a security process to securely access the new access point, hence, utilizing large amounts of energy.

Last, wireless devices are powered by a battery which is a very limited resource. From Moore's law, the number of transistors in a chip will be doubled approximately every 18 months [78]. It implicitly says that the power of computing will grow exponentially. However, the capacity of batteries is growing linearly, and this introduces a "power gap" which is the difference between the power required by computing and the battery capacity [67]. Thus, battery power tends to be a very seriously limited resource for small wireless devices, and a security protocol should utilize energy to the minimum extent possible. This aspect is the primary focus of this dissertation.

D. MOTIVATION

From the aforementioned characteristic differences compared to wired-line networks, wireless networks need a security system or protocol that is properly designed to account for these inevitable differences. While there are several issues related to wireless networks as described above, the focus in this dissertation is to design *adjustable security protocols that are energy efficient. The main thesis of our work is to employ the concept that it is possible utilize a "reasonable" amount of energy to provide a "reasonable" amount of security.* We will focus this work only on WLANs; however, this concept of adjustable security can be applied to other wireless networks as well.

Providing a reasonable amount of energy using the adjustable security concept may be possible due to the following hypotheses:

The first hypothesis is as follows. The importance of information exchanged in wireless networks should be a factor in designing a security protocol. Instead of being fixed, the security protocol should be adjustable in term of the degree of security for network packets. For example, A wireless user may use a medium level of security for normal information exchange on his laptop. Then, when he needs to exchange important information such as for on-line banking, he may increase the security level of his wireless protocol. Then, after the exchange, he may decrease the security level to save energy for his laptop. Therefore, there should be a way to allow wireless users to adjust their level of security as needed.

The second hypothesis is that the degree of security should also be related to the wireless technology. In WLANs, there are 3 main types of packets, Management, Control and Data packets. Management packets are for network association, authentication, and discovery. They may not require encryption since the content is not secret. However, they may need message authentication for packet integrity. Control packets are for traffic and access control, and require similar security services as Management packets. Data packets carry user's data and require encryption for privacy. In each packet type, there are several subtypes and each subtype may require different security levels. Therefore, a security protocol should be adjustable to the wireless technology to provide optimal security services for energy efficiency.

The last but most important hypothesis for this dissertation is that *all encryption algorithms or ciphers may not be energy-efficient all the time* as they are currently used in wireless networks. There may be some ciphers that could provide the same security level, but consume less energy under some circumstances. Therefore, adjustable performance of energy consumption is possible by utilizing different ciphers for different services under different circumstances.

From these hypotheses, we propose a security framework that utilizes different security algorithms with different properties to provide tunable security to limited, battery-powered wireless devices in dynamic wireless environments so as to reduce the overall energy consumption.

E. CONTRIBUTIONS

- Performed exhaustive measurements of time and energy consumption of cipher primitives as a function of a variety of parameters such as packet size, key size, number of operational rounds, size of secret etc. and developed mathematical models for energy consumption

- Developed a method to estimate the robustness of a cipher and extended it to include security protocols so as to have quantitative measures for evaluating security strength

- Proposed three methods for saving energy in security protocols in general and applied them to wireless local area networks

- Developed an example security protocol for WLANs at both session and packet levels that saves energy up to 40% compared to standard security protocols

F. THE OUTLINE OF THE THESIS

The rest of this dissertation is organized as follows. Chapter II explains related material that is used as background knowledge in our work. In Chapter III, we perform a detailed study and propose a performance model of energy consumption of cryptographic functions used to provide security services. In Chapter IV, we discuss the level of security strength and propose a performance model of security strength. In Chapter V, we propose an energy efficient security framework, and show the application of the framework to standard security protocols to improve their energy efficiency. The application of the framework shows that we can save energy by changing or modifying the standard protocols. In Chapter VI, we propose a greenfield approach in which a new security protocol, called TuneSec (Tunable Security), is designed and evaluated. The concluding remarks and our future work are presented in Chapter VII.

II. BACKGROUND

In this chapter, we present an overview of topics that are closely related to our work. First, we explain cryptography in general and describe some cryptographic algorithms in detail. Then we describe network security services such as encryption, message integrity and authentication, and digital signatures. Finally, we explain standard security protocols for WLANs.

A. CRYPTOGRAPHY

Cryptography plays a very important role in providing network security services such as securing information exchange, authenticating users and validating users' identities in a communications network. Generally, cryptography is categorized into two kinds, symmetric and asymmetric cryptography or secret-key and public-key cryptography respectively [109].

1. Symmetric Key Cryptography

Symmetric key cryptography uses only one key to provide security services such as confidentiality and authentication as described later. The key is often referred to as a *secret* key; therefore, symmetric key cryptography is often called *Secret Key Cryptography* (SKC).

SKC is widely used for data encryption due to its fast operation and portability [102]. Although the cipher is widely used to provide encryption, it is also used to deliver authentication or integrity services by modifying the usage of the cipher. The cipher could be one of two kinds, a stream cipher or a block cipher.

A stream cipher encrypts each bit or byte of a message at a time. Its most important advantage is the speed and so it is commonly used for delay-sensitive applications or within a device that has limited memory or computing resources. The stream cipher generally works as follows. Using a key and an initial vector (IV), a stream of random numbers is generated which is called the *key stream*. A different IV is required to produce a different key stream with the same key. Then, each bit or byte of the key stream is XORed with each bit or byte of the message which produces the ciphertext. Examples of stream ciphers are RC4 [95] and SEAL [98].

A block cipher encrypts a message on a block-by-block basis. For each block of a message, the bits in the block are typically diffused, permuted, and manipulated with a secret key to produce a block of ciphertext. With the secret key, the block cipher produces a unique ciphertext corresponding to a particular block of plaintext. This makes it easy to identify part of the message that has a known ciphertext. To eliminate the problem of identifying blocks by this uniqueness or to provide different security needs such as producing a key stream similar to a stream cipher, a block cipher can operate in different modes such cipher block chaining (CBC) mode or Counter mode [109].

There are many examples of block ciphers, but well-known ones are DES [12], RC5 [96], IDEA [102], CAST [20], Blowfish [101], Rijndael [42], Serpent [23], and Twofish [103]. The Rijndael cipher is now a new National Institute of Standards and Technology (NIST) standard called the Advanced Encryption Standard (AES) [14]. Some of these algorithms are explained in following sections.

A drawback of SKC is that *key distribution and key management* are difficult. To use SKC in a communication, two peers need a *shared secret key* to encrypt a message before transmission and to decrypt the message after reception. The distribution of the shared secret key to all associated peers is not a problem if the number of peers is small. However, in a large communication network, key distribution and management is difficult. With N peers in a communication network, $N(N-1)/2$ pairs of secret keys need to be generated for communication. Thus, SKC is not scalable.

a. RC4 RC4 [95] is a stream cipher designed by Ronald Rivest in 1987 and it is widely used in many applications today and in wireless networks such as IEEE 802.11 Wired Equivalent Privacy (WEP) protocol [13], Cellular Digital Packet Data (CDPD) system [82], and Transport Layer Security (TLS) protocol. Before generating a key stream, a key expansion and an initial permutation process are required for each encryption. Hence, they introduce an overhead for each packet before encryption. These processes can be cached into memory for efficiency but the algorithm still requires more memory storage and it increases complexity. It should also be noted that computation with RC4 does not depend on the key size or operational rounds (a repeated operation that is common among SKC block ciphers).

Due to its simplicity, RC4 is fast and efficient once the key stream is generated, and it can be written using only a few lines of code. It requires only 256 bytes of RAM. It is also very fast since it uses only 7 CPU clock cycles per byte of output on a Pentium CPU architecture [104]. Hence, it was one of the best encryption schemes during the past decade. However, Fluhrer and many researchers have discovered several vulnerabilities in the RC4 algorithm [48] that make it unsafe for any key size although increasing the key size commonly increases the security strength. However, the weakness of RC4 can be mitigated from potential attacks if the first 256 bytes of a key stream are discarded [93].

b. AES AES [14] (previously called Rijndael) is a block cipher designed by Joan Daemen and Vincent Rijmen that has a variable key length of 128, 192, or 256 bits to encrypt data blocks of 128, 192, or 256 bits long [42]. Both block and key lengths are extensible to multiples of 32 bits. AES encryption is fast and flexible, and it can be implemented on various platforms especially in small devices and smart cards. Also, AES has been rigorously reviewed for security loopholes for more than two years before it was standardized by NIST in 2001. AES is considered to be very secure.

The security of the AES algorithm depends on the number of “Rijndael” rounds. The more the number of rounds, the more the security strength. Each Rijndael round is composed of four operations, Byte Substitution, Shift Rows, Mix Columns, and Add Round Key with some exceptions for the last round. The last round does not include the Mix Columns operation. Also before the first round, an Add Round Key is required and this could be considered as an overhead for each encryption or for each data packet in communications. Compared to the computational overhead with RC4, AES has a much smaller overhead. AES is also known to be easily expandable. Its security strength can be increased by increasing the number of Rijndael rounds.

c. Blowfish Blowfish [101] was created by Bruce Schneier as another cipher for 32-bit microprocessors that is fast, simple, and has variable key length. It is composed of a key expansion process and a data encryption process. Its key expansion process is very computationally intensive because it converts a key (up to 448 bits) into several subkeys totaling 4168 bytes. This can be a considerable overhead. Hence, Blowfish is only suitable for applications without frequent key changes such as a file encryptor, and not suitable for data packets in packet switching communications [102].

d. RC5 RC5 [96] was also created by Ronald Rivest. Like RC4, the RC5 algorithm includes a key expansion process, and this is considered as an overhead for each packet encryption. Unlike RC4, it is a block cipher and it is much more flexible in that it works with variable parameters such as the input block size, the number of operational rounds, and the key size to offer a great deal of flexibility for different applications. The security strength of RC5 depends on the combination of block size and the number of operational rounds.

2. Asymmetric Key Cryptography

Asymmetric key cryptography was first introduced by Diffie and Hellman in 1976. It was invented to solve the scalability problem of SKC in distributed networks. Asymmetric key cryptography uses two different keys with two different methods for encryption and decryption. One key needs to be known only to its owner (called a *private key*) and the other is not secret and can be distributed to other parties (called a *public key*). The public key is typically used to encrypt a message and the private key is typically used to decrypt the message. Asymmetric key cryptography is also known as *Public Key Cryptography* (PKC) because one of the keys can be made public.

The PKC system is commonly used for distributed computer networks. For one party to communicate with N other parties in a network, it needs to know only the N public keys of the other parties and its own (private key, public key) pair. The number of keys in PKC does not exponentially increase as the number of communicating parties increases, and this solves the scalability problem in large distributed communication networks.

PKC is also important for certain network security services such as non-repudiation which is provided by using a technique called *digital signature*. An owner of a message signs the message with his digital signature, and the signature is used for verification of the message authenticity and also integrity. With the digital signature, the owner cannot deny having created or transmitted the digitally-signed message with his signature. The non-repudiation service and the digital signature are commonly used in email and e-commerce transactions.

However, PKC is not as efficient as SKC in wireless networks in two ways. Because PKC algorithms are based on mathematically hard problems, the key size has to be much larger than that of SKC to provide the same security levels as SKC [69]. More importantly, the encryption schemes are known to be computationally intensive [75]. Therefore, this makes PKC undesirable for use in wireless networks where wireless devices have limited resources. However, with a new type of PKC, based on *elliptic curves*, it may be possible to implement security services based on PKC for wireless devices [36, 26].

A PKC algorithm makes use of a *trapdoor one-way function*. A one-way function is a method used to produce a "digest" from a message and it is computationally impossible to reverse the digest to get the corresponding message (as described later in the context of hash functions). On the other hand, the trapdoor one-way function is a one-way function that has the property that it is possible to reverse the function by simply using a secret trapdoor (a key). Consider a public key y and a function $f(y, x)$ that maps the plaintext x to a ciphertext $x' = f(y, x)$. With the one-way property, it is hard to find x from x'. However, by using a private key or a secret trapdoor, z, it is possible to reverse x' to find x such that $x = f(z, x')$.

A mathematical example of the trapdoor one-way function is integer factorization. Given a product of two very large primes (2^{1024}-digit primes), it is hard to find each prime from the product. However, if either one of the primes is known, it is easy to find the other prime. Either of the prime numbers is then the trapdoor. Some other examples of trapdoor one-way functions are the knapsack problem, the discrete logarithm, and the lattice finding problem [75]. In this work, we consider only popular PKC algorithms namely RSA [97] which is based on the integer factorization and discrete logarithm problem and Elliptic Curve Cryptography (ECC) [74] which is based on the discrete logarithm problem on an elliptic curve.

a. RSA The PKC algorithm that is widely used to provide security services today is the Rivest-Shamir-Adleman code or just RSA [97]. In the example below Alice (A) wants to send an encrypted message, to create a digital signature of the message, and to send both to Bob (B). For this, she employs RSA. In RSA, first, a key pair needs to be generated. The process begins with the selection of two large prime numbers p and q, and then calculating their product $n = p \cdot q$. Then, we choose an integer e between 3 and $(n-1)$ such that $GCD(e, p-1) = GCD(e, q-1) = 1$ where $GCD(x, y)$ is the *greatest common divisor* of x and y. Then, we compute d which is the multiplicative inverse of e which satisfies $e \cdot d = 1 \ mod \ \phi(n)$ where $\phi(n) = (p-1)(q-1)$ is the *totient function*. The public key is (n, e) and the private key is (p, q, d). The security of RSA lies in the fact that given (n, e) and any encrypted message using (n, e), it is computationally impossible to find (p, q, d), that is necessary for decryption [75].

To encrypt a message m, Alice uses Bob's public key $ku_b = (n_b, e_b)$ to produce the ciphertext $c = m_b^e \ mod \ n_b$. To decrypt the ciphertext, Bob will use his private key $kr_b = d_b$ to compute $(c)_b^d = (m_b^e)_b^d = m \ mod \ n_b$. The integer factorization of n into p and q is the trapdoor one-way function (where calculating d which is necessary to reverse c to m is easy if p and q are known). Thus, (p, q, d) is the trapdoor secret

In contrast to encryption, to sign a message m, Alice uses her private key d_a and her public modulus n_a to encrypt a message to produce a signature $sig = m_a^d \ mod \ n_a$. Bob, the message receiver, uses Alice's public key (e_a, n_a) to verify the signature as $(sig)_a^e = (m_a^d)_a^e = m \ mod \ n_a$. The signature is valid if the message m (calculated from the signature) is the same as the one received from the sender.

A detailed explanation of RSA encryption and decryption as well as signature generation and verification algorithms can be found in [110]. Note that usually RSA encryption and signature verification are not as computationally intensive as decryption and signature generation. This is because the exponentiation of m to the public integer e for encryption or signature verification can be made easy by choosing e to be a small number such as 3 [75]. On the contrary, RSA decryption or signature generation is much more computationally intensive since the private key d is typically a very large integer.

b. Elliptic Curve Cryptography (ECC) The security of the ECC is based on a mathematically hard problem called the *Elliptic Curve Discrete Log Problem* (ECDLP) [74]. Given P and $Q = k \cdot P$ where P and Q are discrete points (x, y) on an elliptic curve and k is a random integer, it is computationally hard to find k given P and Q. The operation $k \cdot P$ is called *scalar multiplication* which is a trapdoor one-way function, and k is a trapdoor secret. To prevent a successful attack, k must be a large integer, and the *number* of discrete points on the elliptic curve must also be large.

In ECC, private and public keys are generated as follows. The private key k is an *integer number* randomly selected from $[2, 2^n]$ where n is the number of bits of the key k. The public key R is *a point on an elliptic curve* where $R = k \cdot G$ and G is the base point which is fixed for an elliptic curve. In this case, the asymmetry between the public and private keys can be clearly observed. Even after making R and G public to any party, it is extremely hard for someone who is not the owner to determine k. Several algorithms (such as *Elliptic Curve Digital Signature Algorithm* (ECDSA) for digital signatures and *Elliptic Curve Diffie-Hellman* (ECDH) for key exchange) have been developed based on ECDLP [74].

The ECDH algorithm is commonly used to generate, for example, a pre-master key for a session between Alice (A) and Bob (B). It is generated by "combining" a public key and a private key of two peers. To generate the pre-master key, Alice uses her private key that she combines with Bob's public key, or Bob uses his private key combined with Alice's public key. The following explains how a pre-master key is generated for both Alice and Bob. Given Bob's public key R_B, Alice uses her private key k_A to generate $k_s = k_A R_B = k_A k_B G$. On Bob's side, given Alice's public key R_A, he uses his own private key k_B to generate the same pre-master key as $k_s = k_B R_A = k_B(k_A G) = k_A k_B G$. Finally, both Alice and Bob have the same shared key k_s. Note that it is impossible for anyone else to generate the same pre-master key when k_A and k_B are secret.

The ECDSA signing and verification algorithms are somewhat similar to RSA in that a party uses its private key to generate the signature of a message. The signature can be verified using the party's public key. A detailed explanation of the ECDH algorithm and the ECDSA algorithm can be found in [122] and in [59] respectively.

B. NETWORK SECURITY (REVISITED)

Using techniques from SKC and PKC described in previous section, we can provide several network security services. In this section, we discuss network security services in more detail.

1. Confidentiality

Confidentiality is a countermeasure to eavesdropping attacks. It utilizes encryption algorithms such as RC4 or AES to manipulate or encrypt a message in such a way that anyone who does not know the "secret" is computationally unable to reverse the manipulation to recover the message. Note that the secret is typically called *a key*.

Figure II.1 shows the process of providing confidentiality. The scenario is that Alice (sender) wants to send a message or *plaintext* to Bob (receiver) over an insecure channel. She encrypts the plaintext using an encryption algorithm with a key k_A to produce an encrypted message or a *ciphertext*. Bob decrypts the ciphertext using a decryption algorithm with a key k_B to retrieve the plaintext. If the keys k_A and k_B are the same and the encryption and decryption algorithms are the same, it is called *symmetric key encryption* or *secret key encryption*, and the keys are called *secret keys*. If the key k_A and k_B are different, and so are the algorithms for encryption and decryption, it is called *asymmetric key encryption* or *public key encryption*. In the case of Alice sending Bob a message, the key k_A is called Bob's *public key* and the key k_B is called Bob's *private key*. In the reverse case, Bob uses Alice's public key to encrypt and send a message to Alice, and she uses her private key to decrypt it.

By employing symmetric encryption, two entities at both ends of a link need to share

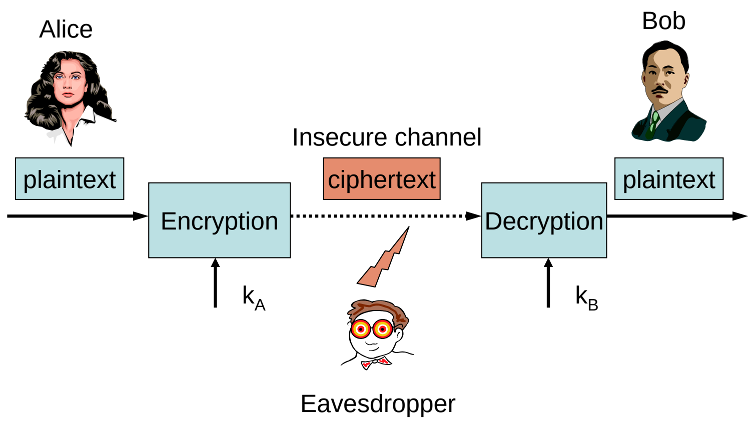


Figure II.1: An Encryption/Decryption Scheme

a key beforehand. Alternatively by employing public key cryptography, an entity (Bob) may use a trusted third party (TTP) to distribute its public key. Any other entity (such as Alice) that wants to send data to Bob obtains his key from the TTP beforehand and uses his public key to encrypt data. Only Bob who owns the private key paired to the public key can decrypt the data.

2. Entity Authentication

Entity authentication is also an important security service for wireless networks. Due to the unknown coverage of wireless networks, an authorized area, such as a corporate building, limited by a physical perimeter can no longer be used to authorize users in wireless networks. Radio signals can propagate outside the authorized area. Hence, without strong authentication, a wireless network is vulnerable to unauthorized access which may lead to other security problems.

A generic way that a peer authenticates itself is to provide to another peer its *credentials* that only it can produce from a message and a secret key. A common scheme for entity authentication is call a *Challenge-Response* scheme as shown in Figure II.2.

An authenticator (Bob) sends a message which is a challenge, C, to Alice who needs to

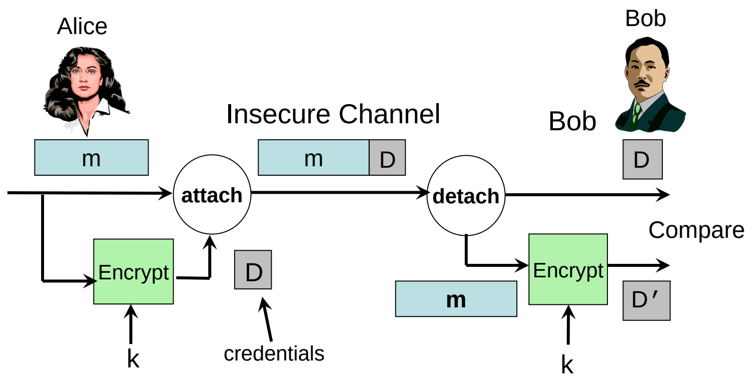


Figure II.2: A Symmetric Key Challenge-Response Authentication Scheme

be authenticated. Alice replies with a response, R', or the credentials associated with the challenge to Bob. Bob verifies the response to his version of Response, R, which he generates with his shared secret with Alice (k). Alice is authenticated if both R and R' are the same. This can only be the case if they are generated from the same key k and no one else can produce R or R' like Alice and Bob do. However, R' and C can be recorded and reused by any attacker. Therefore, the message R should be used only once to prevent so-called *replay attacks*. The challenge should be unique to each entity, or it is vulnerable to *impersonation attacks*. R is called a nonce, since it is a Number used only ONCE.

Different cryptographic functions, for example, SKC encryption (such as RC4 and AES) or PKC encryption (such as RSA) can be used to create the credential which is required to be uniquely associated with the key and the challenge.

3. Message Authentication

Transmission over wireless links is also susceptible to message modification or message injection attacks. For instance, an attacker may capture packets during a bank withdrawing transaction and modify them that the withdrawal goes to his bank account. Also an attacker may inject malicious packets that look like they are from an authorized user to gain access to the network. Message authentication is aimed at protecting such active attacks.

Message authentication is similar to entity authentication in that it uses the credentials produced from a message to prove the authenticity of the message. It shows not only that the message is generated by an entity that claims to be the originator, but also that the message has not been modified during transmission. The message modification or message fabrication can be detected by verifying whether or not the credentials associated with the message are valid.

The credentials of the message can be produced by using a *one-way function*. The function f is a one-way function if it has a following property. Given a message m, it is easy to compute $f(m) = D$. Given D, it is hard to find an m such that $f(m) = D$ by any means. D is called the credentials of the message m. This is because the authenticity of the message m relies on the fact that if the message is modified to be m', the corresponding credentials will be D' that is not the same as D. The functions that have the one-way property are *Hash functions* and *Message Authentication Codes* (MACs)

The Hash function is a light-weight function that can be applied to a message m of any size, and produces a fixed length output D called *a message digest* or *a hash* as shown in Figure II.3 (a). There is no key for the Hash function, and that means anyone can produce a pair of m and D. A hash provides only the detection of message modification. However, applications often require the ability to prove who sent the message or to limit messages only to some entities. In such cases, we need to use the MAC function.

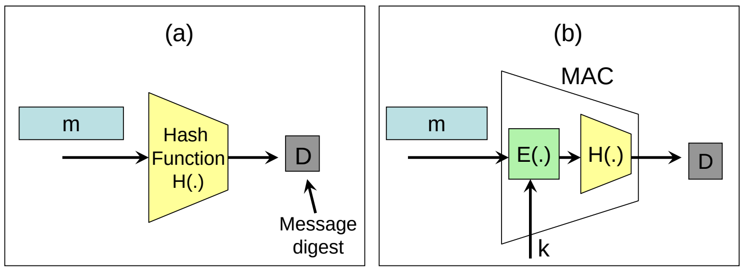


Figure II.3: (a) A Hash Function, (b) A Message Authentication Code (MAC) function

The MAC is different from a Hash function in that it uses a key or a secret k to produce the message digest. Thus, only entities who possess the key can generate the message digest.

The MAC is a compound function that is commonly composed of an encryption function $E(.)$ and a Hash function $H(.)$. Figure II.3 (b) shows one example of how to use $H(.)$ and $E(.)$ to create a MAC function. The MAC gives more security than the Hash function alone. Without the key k, it is computationally impossible to produce a digest D from a message m. The examples of Hash functions are Secure Hash (SHA) [81], Message Digest 5 (MD5) [94], and RIPEMD-160 [75]. The examples of MAC functions are Hashed MAC (HMAC) [66] and Cipher-Block-Chaining MAC (CBC-MAC) [28].

4. Digital Signature

Another kind of an attack is repudiation. A bank client may deny his/her transaction with an on-line banker since there is no guarantee that the transaction was actually made by the bank client. To protect against such an attack, a *digital signature*, which serves as a written signature, for such an on-line transaction is introduced. The digital signature ensures that only the concerned person can "sign" a transaction or a document. It is also used to protect modifications to the transaction or the document. Unlike the written signature, the digital signature is somewhat different in that it is message-dependent; it is unique to each document or each message, and this provides message integrity as well.

Unlike the authentication scheme shown in Figure II.2 where SKC is used, the digital signature is created by using asymmetric key cryptography or PKC techniques such RSA or ECDSA. As shown in Figure II.4, Alice wants to send a message m to Bob. She uses her private key kr_A to "sign" the message, and gets a digital signature D. Alice transmits m and D to Bob. Bob then detaches the message m and uses Alice's public key ku_A to "verify" the message which outputs D'. The message is valid if and only if the D' is the same as D.

The digital signature provides two different security services for Alice and Bob. First, Alice can use her digital signature to authenticate herself to Bob because no one else can produce the signature without knowing kr_A (which is known only to Alice). In addition, with the knowledge of only m, D, and ku_A, it is computationally impossible to find kr_A. Second, Bob also can use Alice's digital signature as a guarantee that Alice is the one who has signed the message m. This provides the non-repudiation service. Alice cannot deny the

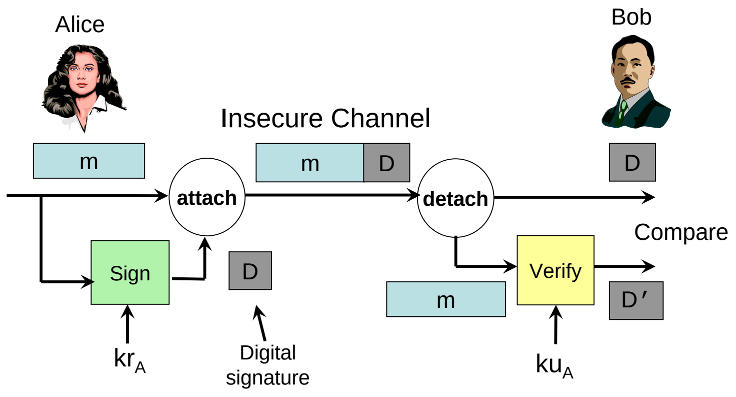


Figure II.4: A Digital Signature Scheme

fact that she did sign the message because no one else can forge her signature.

5. Key Agreement Protocol

In symmetric key encryption, the key used for encryption is assumed to be securely distributed beforehand. The key may be manually distributed via a secure channel. In wireless networks with many peers, it would cause scalability problems. Therefore, key agreement protocols are used to securely distribute a key in a scalable manner.

The key agreement protocol is usually performed after authentication to establish a session key for security services such as encryption and message authentication. Many techniques can be used in the key agreement protocol. Two examples of key agreement protocols are described. In Figure II.5 (a), Alice and Bob want to establish a session key k_s by using an existing key k which is called a *master key*. Alice simply encrypts the session key k_s and sends $E_k(k_s)$ to Bob, and Bob decrypts it to get k_s as $D_k(E_k(k_s)) = k_s$ where $E_k()$ and $D_k()$ are encryption and decryption functions with the master key k, respectively. The session key k_s is not known to any other person because it is encrypted during transmission. However, this method is not scalable since the master key k still has to be distributed beforehand.

In Figure II.5 (b), another technique based on public key encryption is used to exchange a session key k_s. This technique uses two keys to generate the key k_s, and it is called the

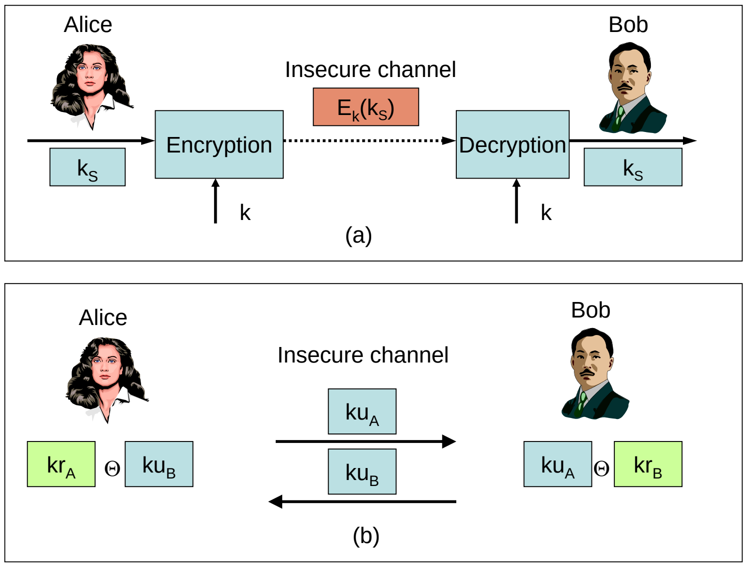


Figure II.5: Key exchange protocol using (a) symmetric key, (b) asymmetric key algorithm

Diffie-Hellman key exchange protocol similar to ECDH.

Alice gives her public key ku_A to Bob, and Bob also gives Alice his public key ku_B. Alice calculates the key k_s from her private key kr_A and Bob's public key ku_B as $k_s = ku_B \odot kr_A$. However, Bob calculates $k_s = ku_A \odot kr_B$. Here $\odot$ is a mathematical operation that is a trapdoor one-way function. By using a function such as the *discrete logarithm*, both Alice and Bob can produce the same k_s. For example, if Alice's public key is $ku_A = G^a \ mod \ n$ where G is a common base integer and n is the modulus, it is computationally impossible to find Alice's private key $kr_A = a$ given ku_A, G, and n. Also Bob's private key and public key are $kr_B = b$ and $ku_B = G^b \ mod \ n$, respectively. The operation $\odot$ is simply modular exponentiation in this case. Therefore, Alice calculates $k_s = ku_B^{kr_A} = (G^b)^a = G^{ba} \ mod \ n$. Bob calculates $k_s = ku_A^{kr_B} = (G^a)^b = G^{ab} \ mod \ n$ which is essentially the same as the k_s Alice calculates. A detailed mathematical explanation of the discrete logarithm problem can be found in [110].

C. STANDARD SECURITY PROTOCOLS FOR IEEE 802.11 WLANS

As shown in II.6, there are several standard security protocols such as TLS, IPsec, and WEP for use with WLANs. Each of them (considered as an upper-layer protocol) is composed of one or more security services as described in Section II.B. Each security service relies on low-level components which are called *security primitives* such as ciphers, hash functions, and trapdoor one-way functions as described in II.A.

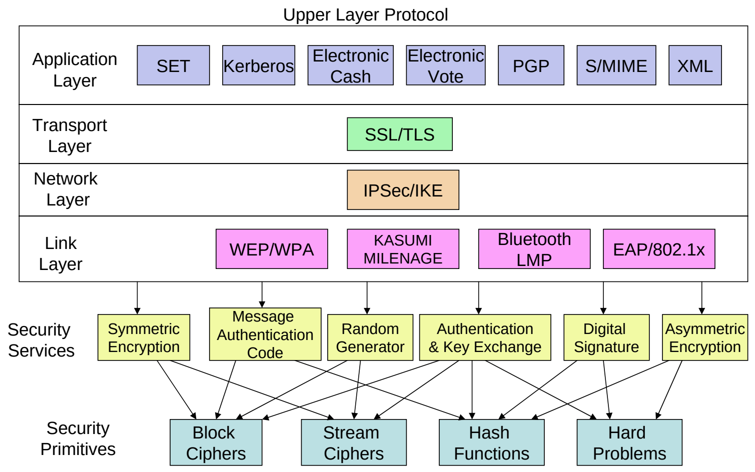


Figure II.6: An Overview of Standard Security Protocols

For example, confidentiality or encryption service is provided using a block or stream cipher. Authentication and key exchange services may be composed using both block ciphers and hash functions. One or more first-level protocols are combined to provide an upper layer security protocol. For example, the TLS protocol uses encryption, hash function, authentication & key exchange, and digital signatures to provide security services at the transport layer [91].

Security protocols for WLANs of interest in this thesis include:

- Extensible Authentication Protocol (EAP) or IEEE 802.1x for providing authentication & key exchange services, and
- Wired Equivalent Privacy (WEP) and Wi-Fi Protected Access (WPA) for providing data encryption and authentication.

1. IEEE 802.11 WLAN Architecture

Security services in WLANs can only prevent attacks on link level traffic between two entities that are wirelessly connected. The entities are often referred to as mobile station or MS and access point or AP. As shown in Figure II.7, one or more MSs are wirelessly connected to a network via an AP in its a coverage area called Basic Service Set (BSS). The AP may connect to a local area network (LAN) via a router to provide access to other network resources. In one WLAN, there may be an extended service set (ESS) of more than one AP, which allows MSs to roam across the ESS area. When a MS wants to access the network, it sends a request to an authentication server (AS) via its associated AP. The AS then performs authentication and may authorize the access request to the MS. During the authentication process, the AP acts as a proxy to relay messages between the MS and AS.

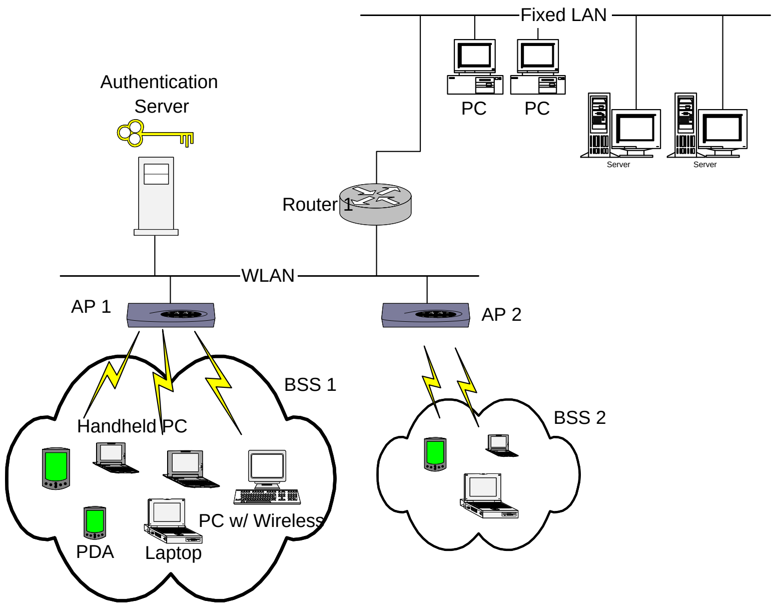


Figure II.7: A WLAN architecture

As an MS roams from its home network to a public access WLAN or hotspots as shown in Figure II.8, it may have to participate in authentication processes that may be quite different. The authentication process may be executed locally in the hotspot network to access the Internet, or remotely in the MS home network. As the confidentiality service in this scenario is only provided on the wireless link between the MS and the hotspot network, an upper-layer security protocol, such as IPsec, is needed to provide security services on the link between a MS and its home network. Therefore, a security protocol for this scenario needs to be carefully designed; otherwise, it is not efficient for use in small wireless devices. We do not consider this problem in the thesis.

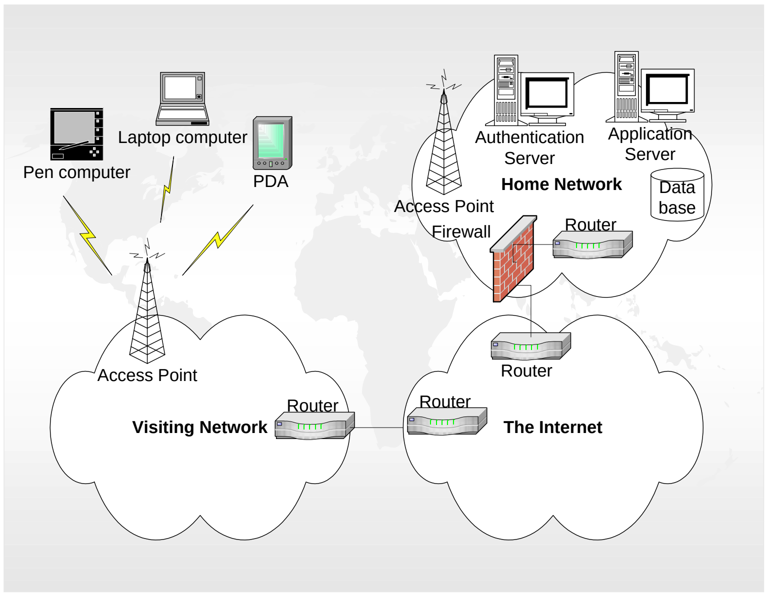


Figure II.8: A WLAN architecture connected to the Internet

2. Data Encryption & Authentication Protocol

The first data encryption and authentication protocol used in WLANs was called Wired Equivalent Privacy (WEP). It was originally created for use in IEEE WLANs or IEEE 802.11 link technology in 1999. However, the WEP protocol has been identified to have several security flaws in the following years [32]. The Wireless Ethernet Compatibility Alliance (WECA) that later changed its name to Wireless Fidelity (Wi-Fi) alliance, released a new security protocol standard in 2002, called Wi-Fi Protected Access (WPA), which aims to fix the flaws [11]. A year later, another version of the WPA standard, WPA version 2 (WPA2), was released to provide advanced security services. WPA is backwards compatible with WEP; however, WPA2 is not backwards compatible with either WEP or WPA.

The WPA and WPA2 are part of a new standard IEEE 802.11i that is an amendment to the IEEE 802.11 standard to provide stronger encryption services and secure methods of authentication [16]. The 802.11i standard provides two data encryption services called Temporal Key Integrity Protocol (TKIP) and Counter Mode (CTR) Encryption with AES cipher (CTR-AES), and two data authentication services called *Michael* and *Cipher Block Chaining Message Authentication Code* (CBC-MAC). The WPA standard is composed of the use of TKIP and Michael together to provide data encryption & authentication services while WPA2 is composed of CTR-AES and CBC-MAC. Together with CBC-MAC and CTR-AES, it is called CCMP (Counter Mode CBC-MAC Protocol).

The WEP protocol utilizes RC4 [95] as the underlying cipher. WEP has several flaws due its poor design and also the weakness of the RC4 algorithm [48]. There is a need to replace WEP with a stronger protocol. However, many 802.11 network interfaces and infrastructures are already deployed and it is expensive to migrate to a totally new standard protocol. Therefore, TKIP and Michael offer intermediate security fixes while utilizing the same hardware that resides in those network interfaces and infrastructures. TKIP offers a software-based solution which adds a two-tier key mixing process that generates a random key for the RC4 cipher to mitigate the IV-weakness attack in WEP [111]. Michael offers a message authentication service which was not included in WEP. Due to the limited capacity of WEP-based hardware, Michael needs to be lightweight. Therefore, Michael is composed of only shift-and-rotate algorithms, and it is known to be a weak algorithm [46].

Due to the weakness of the RC4 cipher itself and the weak Michael algorithm, a new standard, IEEE 802.11i, was proposed and ratified in 2004. It offers a new cipher, AES, which is the new standard for data encryption proposed by NIST [14], as the underlying cipher for CCMP protocol. Therefore, WPA2 employing CCMP protocol will not be backwards compatible; however, it provides high-class security services. RC5 is another cipher that is considered to be secure and efficient although it is not a standard cipher [61]. Due to its flexibility features, we will consider both AES and RC5 instead of RC4 as underlying ciphers in our work.

The WPA and WPA2 standards have two operation modes, Enterprise and Personal modes. The Enterprise mode provides security services that fit an enterprise network which already has a network infrastructure including an authentication server. The Personal mode favors users of small offices or home networks which has no authentication server. The difference between these two modes is the protocol used for authentication and authorization of network users. The Personal mode employs a password-based authentication, called Pre-Shared Key (PSK), to authenticate users. In contrast, The Enterprise mode utilizes the standard 802.1x protocol to provide a stronger authentication and key exchange protocol. The 802.1x protocol will be discussed in the next section. Table II.1 summarizes the WLAN security protocol standards.

Table II.1: WLAN Security Protocol Standards

Mode	Service	IEEE 802.11	WPA	WPA2
Enterprise	Authentication	WEP	IEEE 802.1x	IEEE 802.1x
	Encryption	WEP	TKIP/Michael	AES-CCMP
Personal	Authentication	WEP	PSK	PSK
	Encryption	WEP	TKIP/Michael	AES-CCMP

3. Access Authentication & Key Exchange Protocol

a. EAP/IEEE 802.1x Extensible Authentication Protocol (EAP) or IEEE 802.1x is a standard for Local and Metropolitan Area Networks (LAN/MAN) which is used to provide *port-based network access control* [15]. The EAP itself is not an authentication protocol. It provides an encapsulation for any entity authentication protocol such as the TLS Handshake protocol [43] to perform actual authentication. The EAP was first used for point-to-point network access with password-based authentication such as CHAP [107], and later proposed for use with TLS for WLANs. Now EAP-TLS is part of the WPA, WPA2, and IEEE 802.11i standards.

Within the EAP framework, three entities are involved in the user authentication process: Supplicant, Authenticator, and Authentication Server. In a typical scenario, Supplicant is a mobile station (MS) requesting network access, Authenticator is the MS's access point (AP) that is a bridge to the rest of the network, and Authentication Server (AS) is an Authentication, Authorization, and Accounting (AAA) server such as RADIUS (Remote Authentication

Dial In User Server) [92]. For the authentication process, the current EAP standard mandates only asymmetric authentication such that it only requires a MS to authenticate an AS (or server authentication) but not vice versa. It leaves the MS authentication (or client authentication) to vendors who can specify their own method using their RADIUS server. For example, the client authentication could be a traditional password-based authentication that it is widely used for dial-up and authentication in hotspot networks.

The scheme of the EAP authentication protocol is based on a challenge-response scheme. In this scheme, there are four types of messages: Request, Response, Success, and Failure. A typical EAP message exchange is shown in Figure II.9. First, a supplicant initiates an authentication process by sending an EAPOL-start message. Then a series of Request and Response messages are exchanged. The number of the Request and Response pairs depends on the underlying authentication scheme (such as TLS or passwords). The underlying authentication protocol is enclosed in a box in Figure II.9. There are many authentication schemes that are proposed to be used with EAP, but the mandatory one is EAP with TLS or EAP-TLS [19].

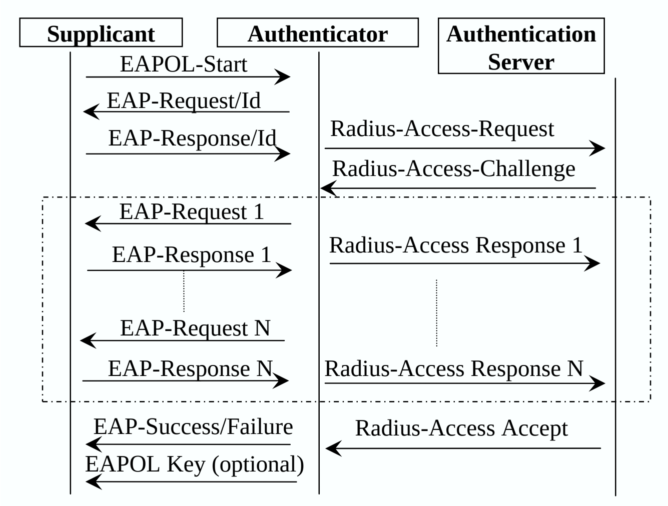


Figure II.9: A typical EAP authentication protocol

EAP-TLS can provides mutual authentication, but the client authentication is indicated as optional in the standard. Figure II.10 shows a typical EAP-TLS protocol with only the server authentication between a MS and an AS server. After initiating the authentication by exchanging EAP Request/Response Id messages with an AP, a normal TLS handshake protocol between the MS and the AS is started. After the authentication process, the key generated during the Handshake protocol between the MS and the AS is forwarded to the AP by the AS.

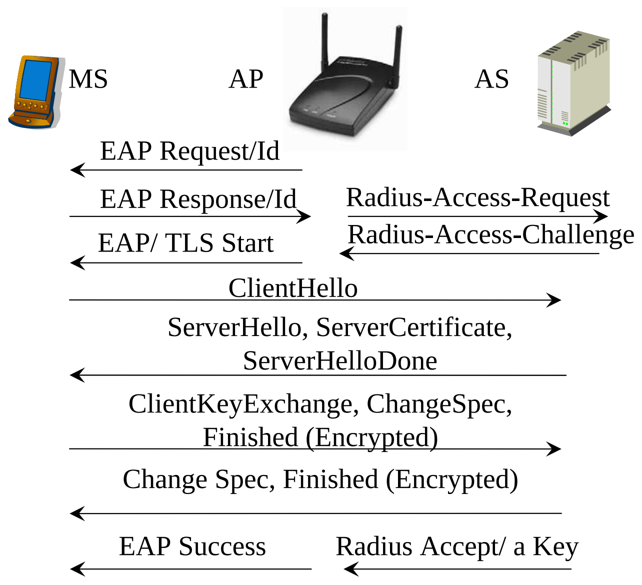


Figure II.10: An EAP-TLS with only server authentication

Client authentication is also possible using EAP-TLS. This process is usually optional because it requires each client to have a valid public key certificate that will be used to authenticate the client. It also requires a Public Key Infrastructure (PKI) to issue and revoke the certificates that additional investment is needed. Instead, a traditional password-based authentication protocol is often preferred to use with EAP Tunneled TLS (EAP-TTLS) to avoid the expensive PKI deployment [50].

b. WPA-PSK WPA-PSK is the standard security protocol in Personal mode for home or SOHO (Small Office Home Office) networks where there is no authentication server such as RADIUS. The WPA-PSK provides user authentication and session key management.

The authentication is based on a shared secret or a paraphrase that is known between an access point (AP) and a mobile station (MS) prior to authentication. Based on the

ESS identity (ESSID) of the networks and the paraphrase, a pairwise master key (PMK) is generated. From the PMK, nonces and MAC addresses of the AP and MS, a session key is generated. The nonce is for preventing the replay attack. The session key has to be renewed after a time interval to increase the security and prevent attacks on a fixed key. The WPA-PSK authentication relies solely on the secret of the paraphrase to generate a session key; thus, it may be subject to dictionary-based attacks as shown in [113].

4. IEEE 802.11 Authentication and Key Agreement

Generally, authentication and key agreement (AKA) occurs in one protocol. The authentication is performed between a mobile station (MS) and an authentication server (AS) via an access point (AP). If the user has the right to access the network, a session key is generated with an agreement between MS and AS. Figure II.11 shows the standard AKA protocol for 802.11 WLANs.

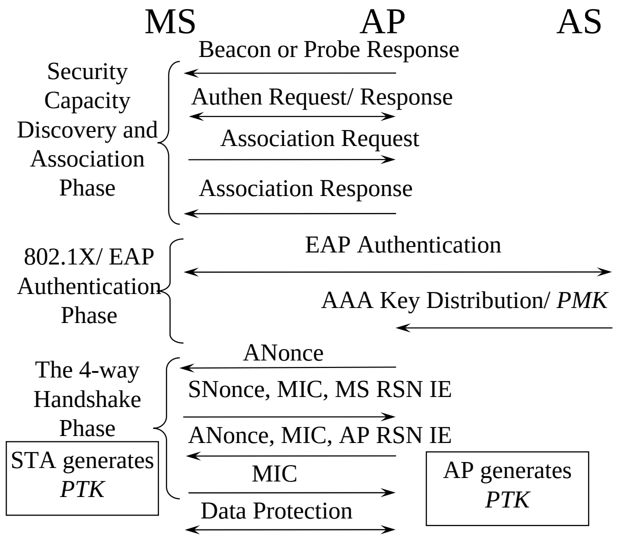


Figure II.11: The 802.11i Standard Authentication and Key Agreement Protocol

In the figure, the AKA protocol can be divided into three phases. During the Service & Security Discovery and Associate phase, the MS looks for an AP that provides capacities and security services to which it is compatible. Then, the MS selects the capability and services. The capacities and services of an AP is described in a beacon packet which is periodically broadcast. The MS then requests an association to the AP. The AP responds to the request and initiates an authentication process in the second phase, which now occurs between the MS and AS. In this phase, the AP acts as a proxy between the MS and AS. After the authentication is successful, a session key called Pre-Master Key (PMK) is generated and distributed from the AS to AP. The PMK is also generated by the MS without key transportation from the AS or AP. This way the key is never revealed on the air.

In the third phase, the AP initiates a 4-way handshake. The purpose of the standard 4-way handshake is twofold. It is used between the MS and AP to confirm the possession of the PMK and to derive a pairwise transient key (PTK) from the PMK for freshness. The PTK is then used to derive three other keys: a key confirmation key, a key encryption key and a temporal key for data encryption during a session. Additionally, the 4-way handshake is used for agreement upon the cipher suite, and for transportation of an encrypted group transient key from the AP to the MS for secure broadcasting.

When a MS wishes to disassociate from an AP, the MS may request deauthentication to destroy the PTK, but the PMK may be cached. With the cached PMK, the MS can re-associate to the same AP without authentication, and only the 4-way handshake is required to generate a new PTK (see [16] Section 5.4.3.2). To reuse the PMK, the MS needs to include a list of cached PMKs in a (Re)Association Request frame, and the AP may reply with which PMK is selected in the first message of the 4-way Handshake (See [16] section 5.9.5). This requires both the MS and AP to cache PMKs and to provide for secure key caching.

D. SECURITY ATTACKS IN WLANS

Wireless networks are inherently vulnerable to several security attacks due to the nature of open medium in the networks. In this section, we give an overview of security attacks in WLANs. The following attacks are also common to other types of wireless networks.

1. Brute-Force Search Attack

The brute-force search attack is the attack trying to find a match of two messages by going through every possible message. For example, for a message that is 64 bits long with an equally long key, the attack requires 2^{64} operations at most or 2^{63} operations on average. The attack is commonly used for key search; thus it is commonly called key search attack. From a pair of known plaintext and a its corresponding ciphertext, the attack tries to find the key using the brute-force search. The key is found if the encryption of the plaintext matches the ciphertext. This attack is simple, and requires only the pair of plaintext-ciphertext. It however requires more operations as the message or the key size increases.

2. The Dictionary-based Attack

The dictionary-based attack is similar to the brute-force attack. However, the searching space is smaller. The attack is also called the password-guessing attack. Based on the known words in a dictionary, the table of pairs of the known words (or plaintexts) and the encryption of the words (or ciphertexts) is pre-computed. Then, an encrypted password is captured and is brute-force searched through the table to find the match of the plaintext. To prevent the dictionary-based attack, the paraphrase needs to be long and random. The WEP and the WPA-PSK could be vulnerable to this attack since they utilize a password or a paraphrase as a shared secret. The non-password-based EAP such as EAP-TLS is not vulnerable to this type of attack. However, EAP-MD5 which is the password-based authentication is vulnerable to this attack.

3. The Eavesdropping Attack

Due to the nature of the radio signal that can propagate through walls and ceilings, it is possible for an outsider to eavesdrop traffic sent or received by a mobile station. To increase the privacy of WLANs, encryption is required, which is provided by using a cipher with a secret key. The WEP assumes that all mobile stations in the entire network share the same key. Therefore, with WEP any mobile station, which can access the network, can eavesdrop on traffic and this is called the insider attack. The WPA and WPA2 are not vulnerable to eavesdropping and insider attacks since each pair of communication utilizes a pairwise key.

4. The Replay Attack

The replay attack is where an attacker tries to access or claim to be legitimate to the network or other parties without possessing the secret key. The attack relies on previously recorded conversations of other legitimate parties and replaying the conversation. The attack can be eliminated if a session key is freshly generated for each conversation and for each session. It is common to use a nonce as a countermeasure with a shared key to generate a fresh session key for each conversation. WEP is vulnerable to this attack as it uses a 24-bit IV with a shared key as the RC4 key for encryption. Since the IV is short and can be repeated, it is possible that the RC4 key is not freshly generated. WPA and WPA2 are not vulnerable to replay attacks since each session key is generated from nonces which guarantee the key freshness.

5. The Session Hijacking Attack

It is an attack that tries to steal a session from a legitimate user to access the network. This attack happens after the authentication between users and the network, and the MS is authorized or is allowed a session to access the network. Therefore, if a malicious user can hijack the session, he can access the network without authentication.

The session hijacking attack often includes the packet injection attack and the impersonation attack. The injection attack is to disassociate an associated user from the network without detection by the network; hence, the session created by the legitimate user is still available. This means a malicious user can impersonate the legitimate MS, which is already disassociated from the injection attack, and is able to transmit or receive messages as a legitimate user.

This attack can be prevented by using message authentication codes (MACs). WEP is vulnerable to this attack since it does not utilize any MAC algorithm. code. WPA employs Michael as the MAC algorithm, but it is known to be weak [46], which means the attack is still possible although requiring some effort. WPA2 employs CBC-MAC with AES cipher, and it is known to be strong and immune against this attack.

6. The Man-in-the-Middle Attack

It is an attack whose goal is to steal valuable information from a legitimate MS of the network. The information can be the username and password, or a shared secret which may later be used to access the same network. In this attack, an attacker sets up a device to be a "man in the middle" between the legitimate MS and AP. The device acts as a (legitimate) access point to the legitimate MS, and a (legitimate) user to the legitimate AP. The device tries to convince the legitimate MS to associate with it to steal valuable information. The convincing may require communication with the legitimate AP. This attack can be prevented if a mutual authentication between MS and AP is employed.

WEP as well as WPA and WPA2 that use PSK are not vulnerable as long as the shared secret or paraphrase is not compromised. WPA and WPA2 (that use IEEE 802.1x/EAP) *may be* vulnerable to this attack. For example, EAP-TLS with mutual authentication is strong against this attack. However, EAP-MD5 is subject to the dictionary-based attack since the encapsulated protocol which is password-based authentication is already vulnerable to the dictionary-based attack [31].

III. MEASUREMENT AND MODELING OF ENERGY CONSUMPTION OF CRYPTOGRAPHIC PRIMITIVES

Security can be provided at different levels using different settings with different security primitives, which can consume different levels of energy. The security settings can be different in many factors, but the main factors are the choice of ciphers used to provide security functions, the key length, and the number of operational rounds. These factors also have a substantial impact on the energy consumption for providing security.

A security algorithm or a cipher is a function that is commonly used to provide security services such as encryption and message authentication. Many ciphers have been created, but only few are known to be strong and secure, which means no loophole or backdoor is known. That a cipher is strong is practically hard to prove unless it has been rigorously reviewed by cryptographers and experts for many years (such as in the case of AES). RC4 was known to be very efficient in term of computation, but has some loopholes [48]. RC5 and Blowfish are known to be strong. One may however expect these ciphers to consume different levels of energy since the way they operate are different.

With a strong secret key cipher, the length is another factor to increase the strength of encryption. A key size of 128 bits would able to resist an exhaustive key search or brute-force attack until the year 2075 [69]. This belief is based on the argument that the strong cipher has no known short-cut attacks, e.g. attacks that are more efficient than the brute force attack. Using the brute force search with an 128-bit key requires 2^{127} searches on average. This is considered to be infeasible with today's technology. Thus, a large key size would provide more strength against a brute force attack. On the contrary, a long key may increase computation and hence energy consumption.

The strength of a cipher depends not only on the key size but also on the number of operational rounds (which normally is a routine of data diffusion and manipulation). The number of rounds is an important factor that provides security strength against attacks such as linear cryptanalysis [73] and differential cryptanalysis [29]. Unlike the brute-force attack where 2^{k-1} trials (on average for a k bit key) are required to break the key, cryptanalytic attacks can be more efficient. Depending on the algorithm, a high number of rounds may be required for robustness against such attacks. A good security algorithm should not use too few or too many operational rounds. Too few rounds would make the encryption vulnerable to cryptanalysis attacks, and too many rounds would be unnecessary, and importantly can consume more energy. Thus, to provide the same security strength, each algorithm may require a different number of operational rounds.

For example, Rijndael needs at least 6 rounds for a 128-bit key, 8 rounds for a 192-bit key, and 10 rounds for a 256-bit key to provide enough security strength against all known attacks [41]. However, the proposed standard adds more rounds for providing a security margin. Thus, AES performs 10 rounds for a 128-bit key, 12 rounds for a 192-bit key and 14 rounds for a 256-bit key [14]. RC5 is a block cipher with a variable key size, a variable input size, and variable numbers of rounds of operations. There is a differential attack that requires 2^{53} chosen plaintext for 12 rounds and 2^{68} for 15 rounds [61]. However, the input of RC5 is only 64 bits long; therefore, such an attack is impossible with at least 16 rounds since collectiong 2^{53} chosen plaintexts is virtually impossible. The linear attack, which is less powerful than the differential attack, is impossible after 6 rounds. The recommendation from Rivest, the creator of RC5, is that RC5 should have at least 12 rounds to provide enough security strength, or 16 rounds to provide complete security strength [96]. Blowfish is another flexible block cipher with a variable key length and a customizable S-box (a non-linear diffusion box). The key for Blowfish can be up to 448 bits. There is a differential attack which requires 2^{4r+1} chosen plaintexts where r is the number of rounds. Therefore, with 16-round Blowfish, such an attack is completely ineffective because the input size in the case of Blowfish is 64 bits long.

Clearly, different settings of algorithms, key sizes and rounds of a cipher will have different amounts of energy consumption. In wireless communications, as we will see in this chapter, the data packet size is also another factor that impacts the energy consumption. For instance, some security algorithms are more efficient only if they are used to encrypt longer packets. In the next sections, we explore the performance of different security settings in terms of energy consumption. We also propose performance models for energy consumption of the security settings. Additionally, we also use different implementations of cryptographic functions to confirm the difference in energy consumed. We consider three commonly used cryptographic libraries, OpenSSL, Cryptlib, and Crypto++. We also use different methods for measuring energy consumption to confirm the validity of the difference in energy consumed.

A. MEASUREMENT OF ENERGY CONSUMPTION

Energy consumption of security primitives can be measured in many ways. The obvious mechanism is to directly measure the energy consumption of each component of a device such as a laptop or a PDA. This method gives the actual value of energy consumption due to each hardware component of the device. Components involved in operations of security primitives would be the CPU and its memory unit. However, to measure only the energy consumption of a security primitive, this method is probably too complex. This method is used in the research work by Viredaz and Wallach to measure energy consumption in a pocket personal computer device [115]. A less complex method is to measure the total energy consumption of the device, which can be measured by measuring the input current drawn by the device from its power supply.

A second method used to measure energy consumption is to assume that an average amount of energy is consumed by normal operations and to test the extra energy consumed by an encryption scheme [56]. This method simply monitors the level of the percentage of remaining battery by using the standard API of Advanced Power Management (APM) or Advanced Configuration and Power Interface (ACPI). This method has some limitations. The extra energy consumed by one encryption can be very small compared to that by normal operations especially for SKC algorithms. Thus, the granularity of power measurement is very coarse; hence, the amount of measured energy consumption is unlikely to be precise. However, the accuracy of this method could be improved by using an external multi-meter that can read the level of current drawn by one encryption.

The energy consumption of security primitives can also be measured by *counting the amount of computing cycles* which are used in computations related to cryptographic operations. The number of cycles is usually used by cryptographers to evaluate new encryption algorithms such as those which were proposed for the new Advanced Encryption Standard (AES) [105]. Each cycle of CPU spent for an instructional set consumes some amount of energy. In [79], details of the numbers of cycles taken up by each instruction of Intel 486DX2 processor and also various amounts of current drawn from a battery are provided. By averaging the amount of energy consumed in each cycle, we can convert the number of cycles used by a security primitive into its amount of energy consumed. This technique is also used in research work by Carmen et al. [38].

While the first method shows the real energy consumption and gives the overall energy performance and the second method actually reads out the battery level in a mobile device, in this work, we use the third method to determine the energy consumption of a security algorithm. This provides us simplicity and fine-grained measurement. We also use the first method for comparison with the third method in Section III.G.

1. Cycle Counting Energy Measurement

To convert cycles to energy consumption, we need to know approximately how much current is drawn for one computation cycle, the operating voltage, and the operating speed of a processor. For example, an Intel Mobile Pentium III processor has two working modes, full power mode and low power mode. In the full power mode, the processor is normally operating at 1.60 volts with a maximum current of 16.6 A at the speed of 800 MHz. In the low power mode, it is running at 650 MHz, and operating at 1.35 volts with a maximum current of 12.0 A [9]. From the above variables, the energy consumption (E) of each security algorithm can be computed as $E = (C \times V \times I)/F$ where C is the number of cycles, V is the operating voltage, I is the average current for each cycle, and F is the CPU frequency or speed.

However, we do not have a benchmark tool that can measure exactly how much current is drawn for each instruction or each cycle. Also, this approach assumes that the current is constant over the cycle. We need to estimate this value. On average, each cycle consumes approximately 270 mA on an Intel 486DX2 processor [79] or 180 mA on an Intel Strong ARM chip [108]. Based on these, we assume that the average current drawn for our Mobile Pentium M is close to 200 mA. This number is fixed for all energy consumption calculations, and it could be changed easily for energy calculations if the true average current for the processor is known. The key variable here is the number of cycles used for each security algorithm.

For our experiments, we use a laptop with a mobile Pentium III 800 MHz CPU, in which performance data are collected. We use the cryptographic software library from OpenSSL version 0.9.7a [1] to implement security test programs in our work. While other libraries do exist [114], we choose OpenSSL because it has been widely used in the research community and in many open-source research projects. It has been rigorously reviewed by many cryptographers and programming experts for its correctness and performance. We also use the widely-used Cryptlib [2] and Crypto++ [3] libraries for performance comparison as described later. All of the three crypto libraries are certified under the NIST FIPS-140-2 specifications.

B. ENERGY PERFORMANCE OF SECRET KEY CRYPTOGRAPHY

Before transmission, a message is often divided into packets. Packets are encrypted individually to provide data confidentiality. Encryption of different packets should not be related because a loss of one encrypted packet means a loss of all related encrypted packets. Especially in wireless networks where the packet loss rate is high, it is common to provide a per-packet key to encrypt each packet independently so that the decryption can tolerate packet losses.

Since each packet requires a per-packet key, before the packet encryption takes place, there may be need to expand the single shared key. The key expansion process can be considered as an overhead for each packet encryption. Thus, the first parameter for encryption that may affect the security performance is the packet size. While the packet size is often statistically distributed and it depends on what communication layer produces the packet, a long packet would usually reduce the encryption overhead and thus improve the performance.

The second important parameter for the encryption is the size of the encryption key. A large key size would provide more strength against a brute force attack as well as other types of attacks. On the contrary, a long key may also increase computation and hence energy consumption. The third factor that would affect the security performance is the number of operational rounds. A larger number of rounds would prevent successful cryptanalytic attacks, but would also require more computation and hence energy.

In this section, we describe the effect of these three parameters, data packet size, key size, and the number of operational rounds, on the performance of different secret key encryption algorithms in terms of energy consumption. These three parameters are chosen as they tend to be significant factors of energy consumption for encryption.

1. Encryption with Different Packet Sizes

The packet size has been known to have significant effect on wireless transmission. Transmitting large sized packets improves the network utilization because the ratio of the overhead to data payload size is low. On the contrary, large size packets are more susceptible to errors during transmission than smaller sized packets [70]. The packet size also affects the energy consumption due to transmission [45].

The packet size also has an effect on energy consumption due to encryption. From the need for per-packet keying, each packet may require a key expansion process which requires a fixed amount of energy for computation and is independent of the packet size. Our study has shown that encryption of long packets consumes less energy than that of short packets using the same key length and operational rounds [88]. For example, with 128-bit key RC5 algorithm, the key expansion takes about 65 % of the total energy for encrypting a 16-byte packet, and that can be decreased to 14 % if used for encrypting a 256-byte packet. Some encryption algorithms need to expand the key that requires more subtle data diffusion and manipulation such as the case of the Blowfish algorithm. The 128-bit key expansion process in Blowfish takes about 98 % and 94 % of the total energy to encrypt a 16 byte long packet and a 256 byte long packet respectively. Hence, Blowfish is only appropriate for encrypting a large file rather than individual packets.

After key expansion, a packet can be encrypted. In the case of a block cipher, the packet that is larger than the input size of the cipher is divided into several blocks, and then encrypted. The overhead in this process is the block division. However, block division consumes very little energy since it can be done using instructions which take few CPU cycles.

Figure III.1 shows the energy consumption of different encryption algorithms using 128-bit keys with different packet sizes. It can be observed that AES encryption consumes far less energy than others when encrypting smaller packets. The AES key expansion consumes much less energy than others. Hence, the packet size only slightly affects the AES algorithm. RC5 consumes more energy than AES, but less than others when the packet is small. RC4 consumes the least energy when the packet is large. Blowfish has a significant key expansion overhead and it is not suitable for packet transmission although it is supposed to be a light-weight encryption scheme with sufficient security [102]. Although, we can use the same key for all packets to reduce the key expansion overhead of Blowfish, this requires an efficient cache, state and key management, and hence increases the system complexity which is not recommended.

2. Encryption with Different Key Sizes

In this measurement, we study whether the key size has an effect on energy consumption. From previous section, we know that the packet size has an impact on the energy consumption due to key expansion. In this study, we consider energy consumption due to encryption with different key sizes for two cases, with and without the key expansion process. Without the key expansion, we start to count the number of cycles for encryption after the key expansion process.

Figure III.2 shows the amount of energy consumed per byte by each encryption with different key sizes, packet sizes and ciphers *without* the key expansion. From the figure, we can see that AES consumes different levels of energy when different key sizes are employed.

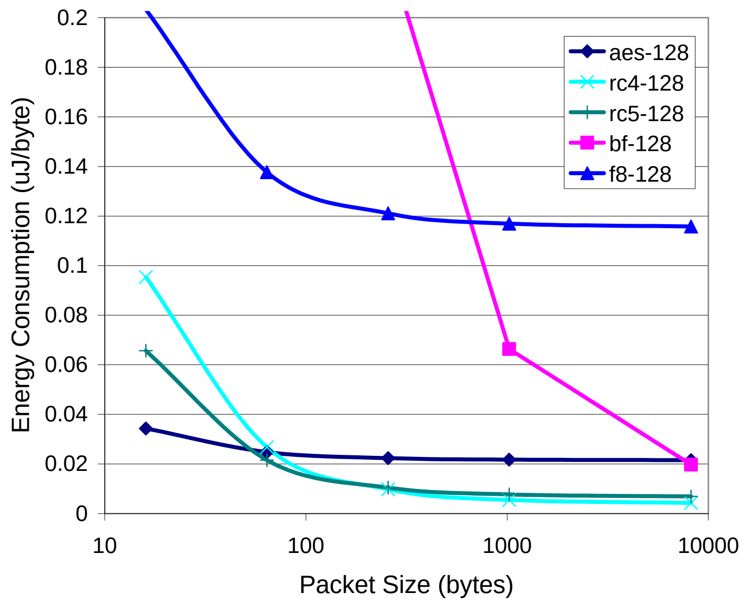


Figure III.1: Energy consumption of encryption with different packet sizes

As we will see later, the reason for this is not the key size itself. With a longer key, the encryption algorithm needs to do more “Rijndael” rounds according to the AES standard which results in more computation and more energy consumption. The RC4 encryption relies on a random number generator or a key stream generator to produce a series of keys. Generating a key from the stream consumes very little energy; therefore, its performance is almost independent of the key size. Thus, for RC4, a larger key size would only make a successful brute-force key search harder without increasing the energy consumption.

In a manner similar to that of RC4, the energy consumption of RC5 encryption is almost independent of the key size. With a longer key size, the encryption just needs a few more additions and rotations of 32-bit words according to the algorithm. These operations are very simple; hence, increasing the key size is unlikely to increase the energy consumption. The computational load of Blowfish is also independent of the key size. No matter what the key size is, Blowfish always runs 16 rounds of a Feistel-like network. However, Blowfish has a heavy key expansion process before encryption as shown in Section III.B.1

We also study energy consumption of encryption *with* the key expansion process and

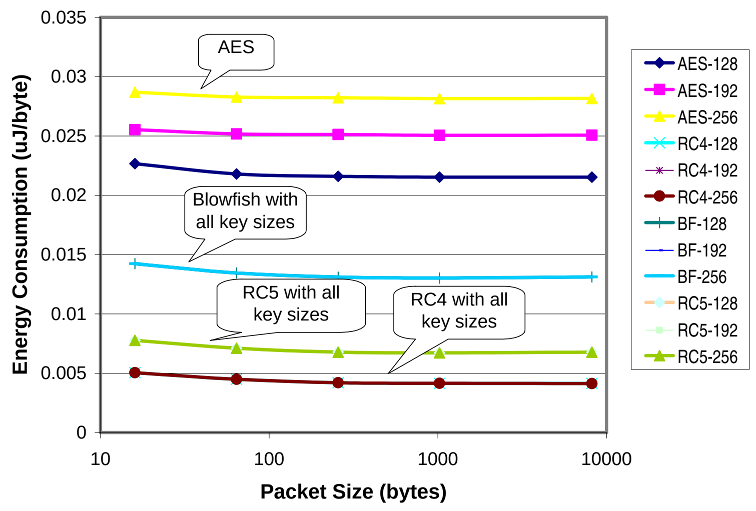


Figure III.2: Energy consumption of encryption without key expansion

different key sizes. Figure III.3 shows the energy consumption. We can see that the energy consumption of both RC4 and RC5 is not affected by the key size while AES is slightly affected when the key size increases and the key expansion process is present.

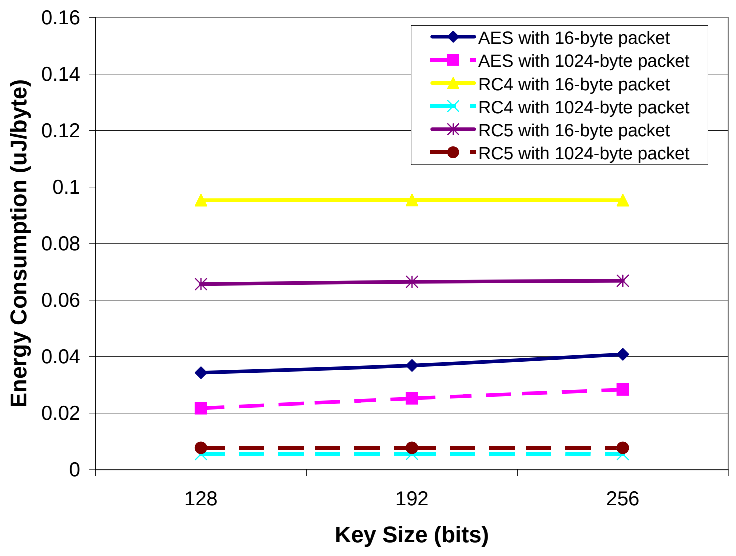


Figure III.3: Energy consumption of encryption with key expansion

3. Encryption with Different Operational Rounds

The energy consumption of SKC algorithms or ciphers is also largely based on the difference in operations or nature of the algorithms. Especially for block ciphers, the energy consumption heavily depends on the recursive cryptographic algorithm which is normally a routine of data manipulation or operational rounds. Block ciphers like AES and RC5 have a recursive operation in which an input to the cipher goes through operational rounds. As the number of the rounds increases, the energy consumption of the cipher increases.

From the previous study of the key size and the energy consumption, we saw that the increase of key size only increases the energy consumption of AES. In fact, the increase of key size changes the number of operational rounds of AES, but not RC5 or Blowfish. As shown in Figure III.2, the energy consumption of RC5 is independent of the key size. This is because RC5 has a fixed key expansion process that does not depend on the key size. In contrast, the energy consumption of AES is increased mainly due to the increase of the operational rounds as suggested by its standard. The key size has an impact on the AES key expansion process, but the impact is much less compared to the AES round operations. The standard AES requires the operational rounds to be 10, 12, and 14 for AES with 128, 192, and 256-bit keys, respectively. Ignoring the standard, we can operate AES with 256-bit keys with a lower number of rounds, but it may be subject to cryptanalysis attacks. In summary, it can be seen that the energy consumption of block ciphers heavily depends on the number of operational rounds, and the key size of a cipher may or may not be a factor in increasing the energy consumption for SKC algorithms. It is known that a long key size has a large impact on preventing the brute force search attack; however, it has only a slight effect in increasing the energy consumption.

The number of operational rounds also has an impact on the security strength of the cipher against attacks such as linear cryptanalysis [73] and differential cryptanalysis [29]. Depending on the algorithm, some algorithms may require higher numbers of rounds to be robust against such attacks. We discuss this further in Chapter IV. A good security algorithm should not perform too little or too many of operational rounds.

In this section, we study the performance of energy consumption of block ciphers, AES, RC5, and Blowfish. RC4 is not studied here since its algorithm is not based on operational rounds. We consider the energy consumption due to encryption with a per-packet key scheme. This means that for each data packet we need to perform key expansion and encryption, and a different key is used for each packet. This scheme is commonly used in wireless networks.

Figure III.4 shows the amount of energy consumption of encryption algorithms with different numbers of rounds. The algorithms are used to encrypt a packet of 1024 bytes with a 128-bit key. The energy consumption shown does *not* include that from the key expansion process. We see that AES consumes energy at a higher rate than others as the number of rounds increases. RC5 is probably a good cipher for constrained devices in terms of energy consumption. However, it has a high overhead due to the key expansion as shown in the previous section.

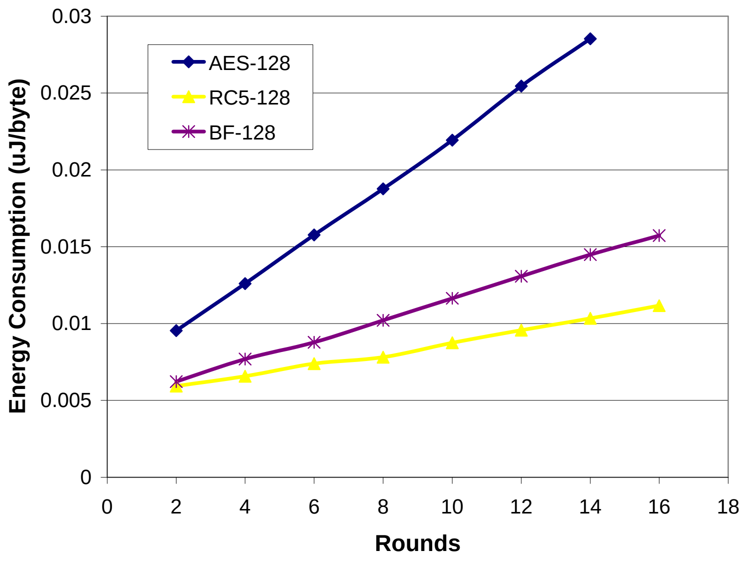


Figure III.4: Energy consumption of encryption with different rounds

Figures III.5 and III.6 shows the energy consumption (using cycle counting approach) of AES and RC5 as a function of packet size and operational rounds. We see that the energy consumption of AES increases as the number of operational rounds increases regardless of the packet size. The energy consumption of RC5 also increases as the number of rounds increases. However, as the packet size increases, the increase of the energy consumption with the number of rounds becomes less significant.

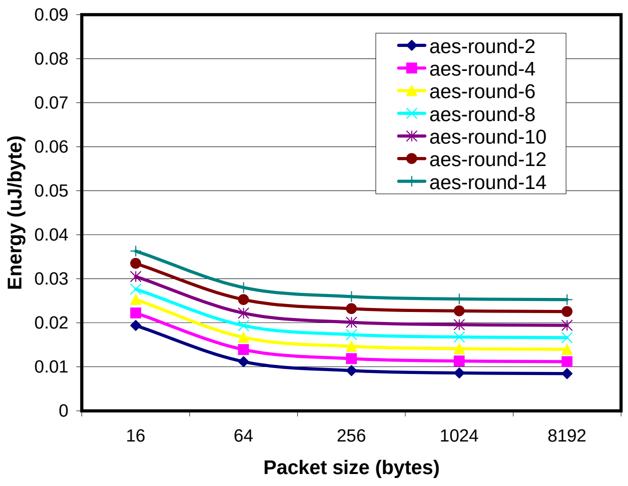


Figure III.5: AES Energy Consumption for Different Packet Size and Rounds

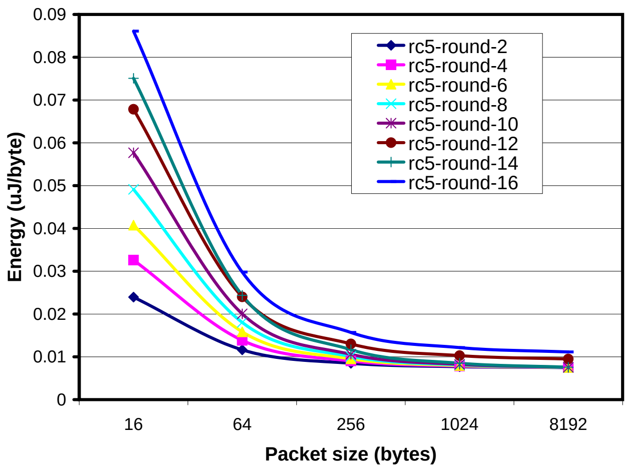


Figure III.6: RC5 Energy Consumption for Different Packet Size and Rounds

4. A Summary of SKC Performance

From the previous sections, we summarize the impact of the factors that affect the performance of SKC encryption in Table III.1. It can be seen that AES and RC5 tend to be energy efficient encryption algorithms for wireless devices since the impact of variable packet and key sizes is low to medium. The AES has advantage over RC5 that the energy consumption is slightly variable with the packet size. Despite its known vulnerability, RC4 is also likely to be very energy efficient due to the medium impact of the packet size variation. Blowfish is not suitable for wireless networks and devices because the variation of packet size has high impact on the energy consumption.

Table III.1: A Summary of Impact of Factors on SKC Performance

Factors	RC4	AES	RC5	Blowfish
Packet Size	medium	low	medium	high
Key Size	no impact	low	no impact	no impact
Round	-	medium	low	low
Vulnerability	known	unknown	unknown	unknown
Suitable for Wireless	yes	yes	yes	no

C. ENERGY PERFORMANCE OF PUBLIC KEY CRYPTOGRAPHY

Security algorithms based on public key cryptography (PKC) are computationally intensive. They are unlikely to be used for data encryption for every packet. However, they are commonly used in exchanging a shared secret or for key exchange. The performance of the key exchange algorithms are different based on what the underlying PKC algorithm is. PKC is also used for digital signatures, which can only be implemented using PKC algorithms. The performance of digital signature algorithms are also different based on the underlying PKC algorithm used. In this section, we show the performance of the key exchange and digital signature algorithms.

1. Shared Secret or Key Exchange Algorithms

Figure III.7 shows the energy consumption of RSA to exchange a secret. Basically, the secret is encrypted using RSA by a sender and it is decrypted by a receiver. It is shown that the energy consumption does not depend on the size of the secret. However, it depends on the key size of RSA and whether it is encryption or decryption. RSA with a 2048-bit key consumes more than twice the energy than 1024-bit RSA. However, doubling the key size does not double the security of the algorithm [69]. Therefore, RSA may not be appropriate for the future where more security strength may be needed with linearly increasing energy consumption. Unlike secret key encryption, it is also clear that RSA encryption and decryption consume different amounts of energy. RSA encryption is quite simple and consumes very little energy. This is suitable for constrained wireless devices in an infrastructure network where there is a powerful device to do decryption.

Figure III.8 shows the amount of energy consumption of secret exchange using the ECDH [122] algorithm. It is shown that exchanging different secret sizes affects the energy performance. The ECDH algorithm can also be used with different elliptic curves which yield different performances. An ECDH with a prime curve is based on prime number operations such as addition and multiplication which can easily be coded in software. An ECDH with binary curves (Koblitz or Random) is based on binary operations which can easily be embedded into a microprocessor. From the figure, it is shown that ECDH with a prime curve consumes less energy than one with the Koblitz and Random curves respectively. Note that energy consumption shown in the figure is only of the key exchange algorithm excluding that of transmission. The performance of ECDH with binary curves can be improved if efficient techniques based on binary algorithms are used [54].

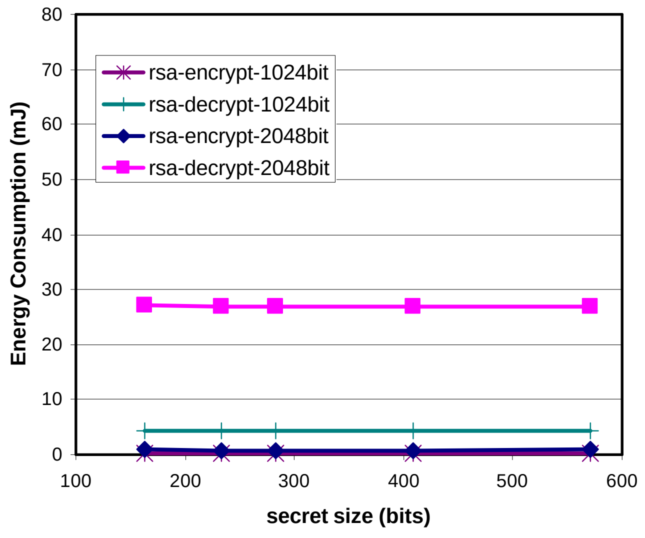


Figure III.7: Energy Consumption of Key Exchange with RSA

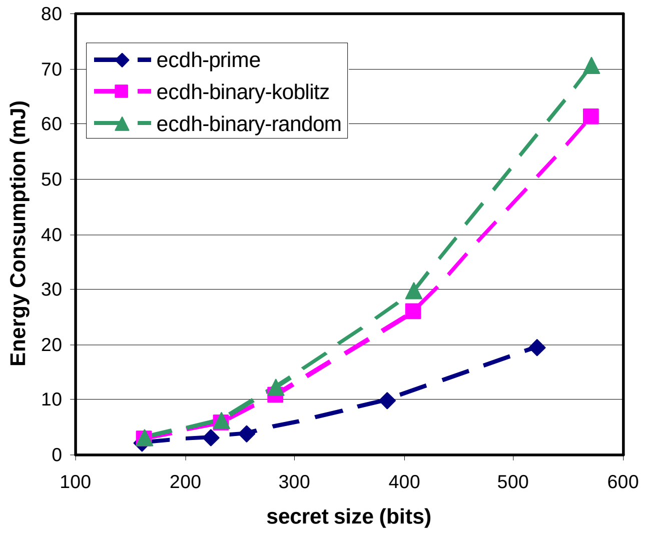


Figure III.8: Energy Consumption of Key Exchange Using ECDH

2. Digital Signature Algorithms

Figure III.9 shows the energy consumption of the RSA digital signature algorithms. The digital signature is generated on 160 bits of hash generated from a message. Since the RSA digital signature is based on the RSA encryption/decryption technique, it also has an asymmetrical performance. The RSA signature verification algorithm (which is similar to RSA encryption) consumes much less energy than the signature generation algorithm (which is similar to RSA decryption) especially when the key size is large. This implies that the verification algorithm is appropriate for constrained devices, but not the signing algorithm.

Figure III.10 shows the energy consumption of the ECDSA [59] algorithm. With increasing key size, the ECDSA also consumes more energy. The level of energy consumption also depends on the underlying elliptic curve. With the prime curve, the verification and signing algorithms consume approximately the same energy and also consume less energy than other curves. However, the energy consumption of the verification and signing algorithms with binary curves (Koblitz and Random) are much different and much higher than that with the prime curve. Note that the ECDSA with a prime curve can be simply implemented in software, and the ECDSA with a binary curve can be easily implemented in an embedded microprocessor.

3. A Summary of PKC Performance

From previous sections, we summarize the impact of several factors on the performance of PKC functions in Table III.2. We note that RSA encryption and signature verification functions are very efficient since the variation of secret size has no or very low impact on the energy consumption. However, RSA decryption and signature generation functions are very inefficient for wireless devices. They are not energy efficient mainly due to the high computation although the variation of secret or key size has no or minimal impact on the energy consumption. ECC-based functions such as ECDH and ECDSA are efficient and are usable for small wireless devices. This is because they are not sensitive to the variation of the factors such as the secret or key size.

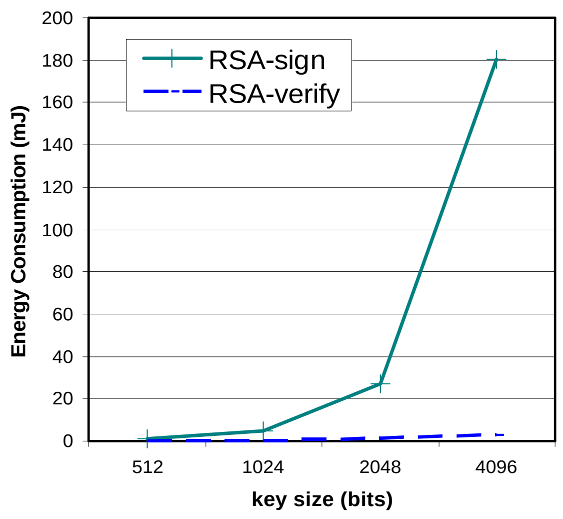


Figure III.9: Energy Consumption of RSA Digital Signature

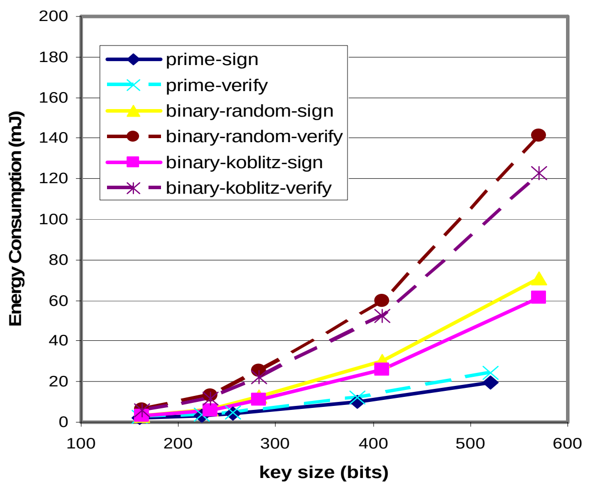


Figure III.10: Energy Consumption of ECDSA Digital Signature

Table III.2: A Summary of Impact of PKC Performance Factors

Factors			Secret Size	Key Size	Energy Efficiency	Complexity
RSA	Encrypt		no impact	-	high	low
	Decrypt		no impact	-	low	high
ECDH	Prime		low impact	-	high	medium
	Random		medium impact	-	medium	low
RSA	Sign		-	very low impact	low	high
	Verify		-	very high impact	high	low
ECDSA	Prime	Sign	-	low impact	high	medium
		Verify	-	low impact	high	medium
	Random	Sign	-	medium impact	high	low
		Verify	-	medium impact	high	low

D. ENERGY PERFORMANCE OF HASH FUNCTIONS

Hash functions are also as important as other cryptographic functions. They are used to provide data integrity in digital signatures and public key certificates. They are also used in conjunction with a secret key to provide message authentication and integrity to prevent malicious message injection into or modification of messages in a wireless network. In this section, we study the performance of two popular hash functions (SHA-1 [81] and MD5 [94]).

Figure III.11 shows the energy consumption of SHA-1 and MD5 with different packet sizes. It can be seen that SHA-1 consumes much more energy than MD5 especially when the packet size is small. To create a digest of a packet, SHA-1 processes 512 bits as input at a time, and continues for as many as 512-bit blocks as in the packet. Each 512-bit block is passed through four rounds, each of which are identical and have 20 operations. Each operation performs a non-linear function, adding, and shifting.

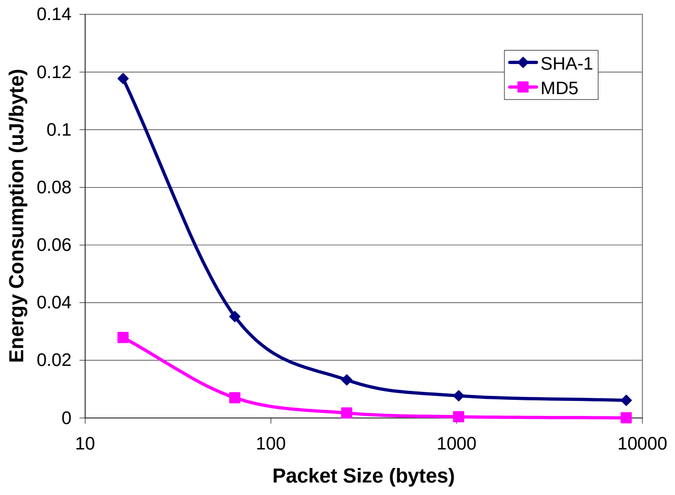


Figure III.11: Energy Consumption of Hash Functions

MD5 also has four rounds of operations, but each round has only 16 operations. In addition, each operation in MD5 is much simpler than that in SHA-1. This results in less computation and energy consumption of MD5 than those of SHA-1. Note that the SHA-1 produces a 160-bit hash which is longer than a 128-bit hash produced by MD5. MD5 is known to have security loopholes [44] unlike SHA-1 which has better security strength.

E. ENERGY PERFORMANCE OF MAC FUNCTIONS

In section III.D, we have shown the energy consumption of hash functions such as SHA-1 and MD5. The security service provided by hash functions is the message integrity in which a message receiver can only detect whether a message is modified. Unlike hash functions, a Message Authentication Code (MAC) provides both message integrity and message authentication in which the receiver can verify not only the integrity, but also the authenticity of the message. Therefore, the MAC is commonly used in security protocols to verify the sender identity and message integrity.

The MAC is an algorithm that is composed of a secret key and a one-way function such as a cipher or a hash function. Examples of MAC functions with a cipher function and a hash function are Cipher Block Chaining MAC (CBC-MAC) [28] and Hashed MAC (HMAC) [66], respectively. The output size of the MAC depends on the output of the one-way function. For example, using CBC-MAC with AES produces an output of 128 bits, and using HMAC with SHA-1 produces an output of 160 bits.

In this section, we study the energy consumption of MAC functions as a function of packet size. We study the HMAC function with SHA algorithm (HMAC-SHA) which produces variable output sizes of 160, 256, 384, and 512 bits as well as CBC-MAC with AES which produces a 128-bit output.

Figure III.12 shows the energy consumption of HMAC-SHA with variable output sizes and 128-bit CBC-MAC-AES. It shows that the CBC-MAC uses much less energy than HMAC-SHA. However, HMAC-SHA produces a larger output size than CBC-MAC-AES. Additionally, with HMAC-SHA, we can produce variable output sizes, such as 160, 256, 384 and 512 bits. Note that we do not show the performance of 384-bit HMAC-SHA because it uses the algorithm of 512-bit HMAC-SHA, and truncates the output of 512 bits to 384 bits; therefore, both 384-bit and 512-bit HMAC-SHA have identical performance. From the figure, we can also see that the energy consumption per byte is reduced as the packet size is increased. This shows that CBC-MAC-AES has much less overhead per packet than HMAC-SHA.

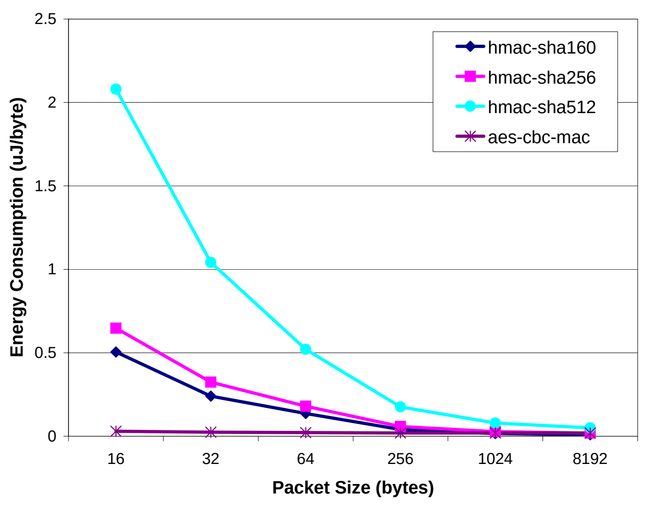


Figure III.12: Energy Consumption of HMAC-SHA and CBC-MAC

F. COMPARISON OF CRYPTOGRAPHIC LIBRARIES

Many cryptographic libraries that are commercially graded are freely available for implementing security functions for software-based applications. They are also available in different application programming interfaces (APIs), programming languages, and platforms. Due to the variety of the libraries, performance of cryptographic functions in terms of energy consumption is probably different. In this section, we perform a comparative study of three different cryptographic libraries, OpenSSL [1], Cryptlib [2], and Crypto++ [3]. Their recent versions are known so far to be efficient and without known vulnerabilities, and widely used in the security research community.

1. Cryptographic Libraries

OpenSSL is probably the most commonly used library to implement cryptographic functions since it is free for both commercial and non-commercial use. It was first implemented by Eric A. Young and was known as SSLeay. The library was originally written for SSL protocol transactions which employ several cryptographic functions. Later, it has been extended to include many more new cryptographic algorithms such as AES, RC4, and RC5. It also includes some hand-tuned assembly code for performance optimization. The OpenSSL's primary API is for C programming, and it supports different processor architectures.

Cryptlib is another efficient and robust cryptographic library. It is written by a security expert, Peter Gutmann. It can be easily integrated into any application using its easy-to-code API. The API supports C programming on a variety of processor architectures. Like OpenSSL, it contains some hand-tuned codes for performance improvement. However, it is free only for non-commercial use.

Crypto++ is known to have the largest list of cryptographic functions available for developers. Its API supports C++ programming with hierarchical class structure and templates. By using the C++ object-oriented scheme, it can be easily integrated into any C++ program. However, the library is not tuned for optimal performance. It is free for any use. More comparative descriptions of these three libraries can be found in [114].

Using these three cryptographic libraries, we conduct a comparative study on a laptop PC platform with an Intel Pentium III 800 MHz CPU. We use the cycle-counting method for energy measurement.

2. Comparative Results

Figure III.13 shows the energy consumption of 128-bit AES with different libraries. The Cryptlib and OpenSSL libraries show results that are close to each other because both of them are based on C programming language. However, Crypto++, which is based on C++ programming, consumes more energy for extra processing of memory/resource management and error checking. When the packet size is small, Crypto++ tends to consume even more energy. Due to memory management in C++, the overhead of allocating and de-allocating small memory becomes significant. However, all of them show similar trends in that the encryption consumes more energy per byte when it is used for small packet sizes, and the consumption is decreased when the packet size is larger.

Figure III.14 and III.15 show results of RC4 and RC5 encryption. We can see that the overhead of using C++ in Crypto++ becomes more significant when the packet size is small.

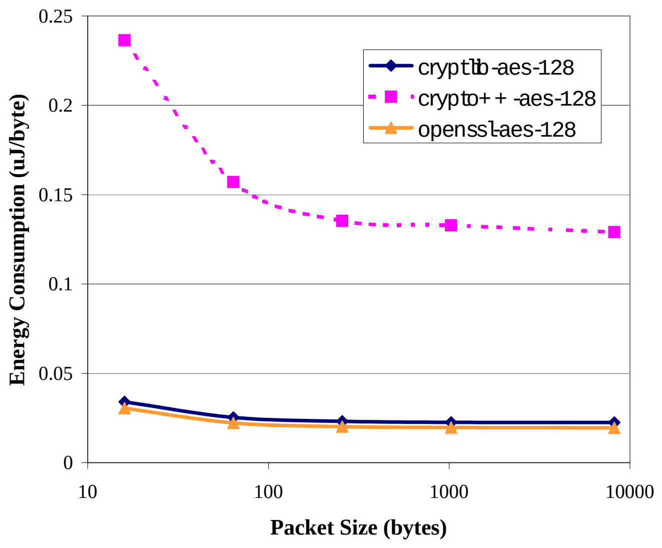


Figure III.13: Energy Consumption of AES with Different Cryptographic Libraries

Compared to that of AES, the algorithms of RC4 and RC5 during the key expansion require higher memory manipulation while AES utilizes a very simple key expansion.

The comparison of only OpenSSL and Cryptlib when used for 128-bit cryptographic functions is shown in Figure III.16. As shown, they both have similar performance in terms of energy consumption.

G. COMPARISON OF ENERGY MEASUREMENT METHODS

Three different methods that are commonly used for energy measurement for cryptographic functions have been described in section III.A. We have already shown the performance of cryptographic functions using the cycle-counting energy measurement method. Using other methods may have different complexity and accuracy. In this section, we perform a comparative study of different energy measurement methods. We compare the cycle-counting method to that using external hardware that is specialized for energy measurement.

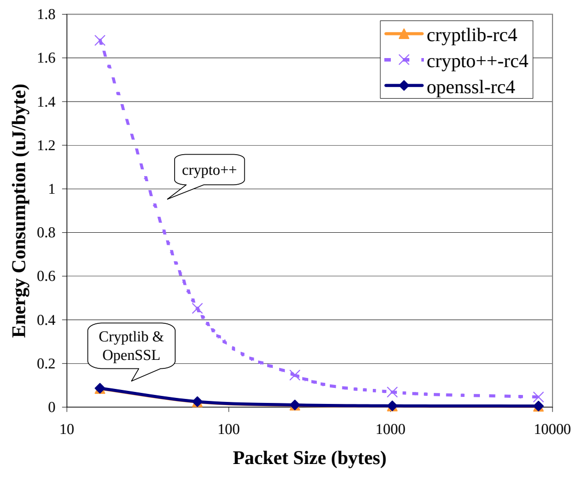


Figure III.14: Energy Consumption of RC4 with Different Cryptographic Libraries

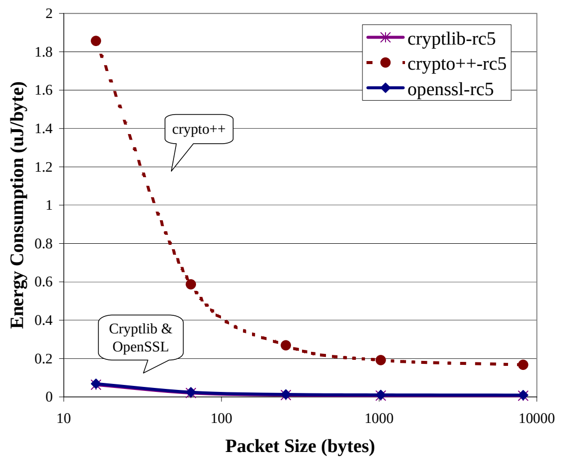


Figure III.15: Energy Consumption of RC5 with Different Cryptographic Libraries

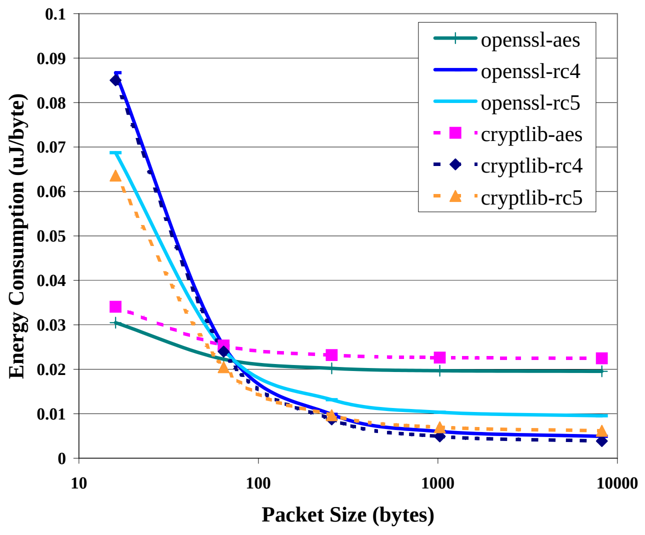


Figure III.16: A Comparison of Energy Consumption between OpenSSL and Cryptlib

1. Hardware-based Energy Measurement

Energy consumption of a cryptographic algorithm can also be measured using a dedicated hardware or a data acquisition (DAQ) system. We use the DAQ system which includes SCXI-1000 module chassis and SCXI-1100 32-channel analog input module from National Instruments, Inc. to measure the amount of current drawn to a laptop while running cryptographic operations. The DAQ system is capable of measuring 240,000 samples/sec. Based on the experiment setting for our experiments and the specification of the SCXI-1100, the absolute accuracy of our measurement is between ±1.265 mA (±0.0614%) and ±1.429 mA (±0.0632%) (See Appendix for absolute accuracy calculation). We used at least 260,000 samples for each measurement, and we obtained a standard deviation of less than 0.82%.

We have set up our measurement test bed as shown in Figure III.17. We measure the power drawn by the laptop during cryptographic operations. We use the same test codes that are used in the cycle-counting method to run the cryptographic operations. We add a programming interface to trigger the DAQ machine to start measurement just before and to stop after the operation. The energy measured by the DAQ is the energy consumed by all components of the laptop, including LCD, hard drive, and etc. We do 1000 experiments for each test and find the average amount of energy consumption. We calculate the standard deviation of energy consumption of all experiments, and find that it is less than 1%.

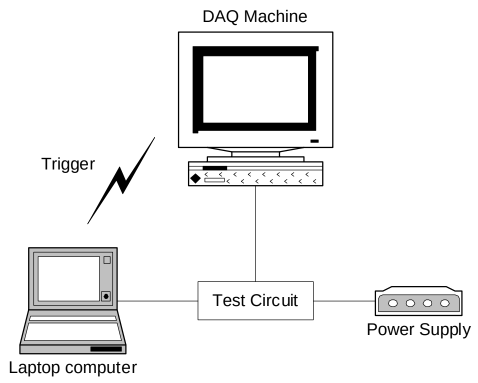


Figure III.17: A Hardware-based Measurement Testbed

2. Comparative Results

Figure III.18 shows the energy consumption of 128-bit cryptographic functions using the OpenSSL library. It shows similar characteristics of energy consumption to those when using the cycle-counting method in Figure III.1. However, the amount of energy consumed is higher when using the DAQ system. Although it is proportional to the amount of computation, it may include energy for LCD, hard disk, memory, and other devices during the cryptographic operation. Figure III.19 shows the comparison between OpenSSL and Cryptlib libraries for 128-bit encryption. It shows results similar to those when we apply the cycle-counting method as shown in Figure III.16.

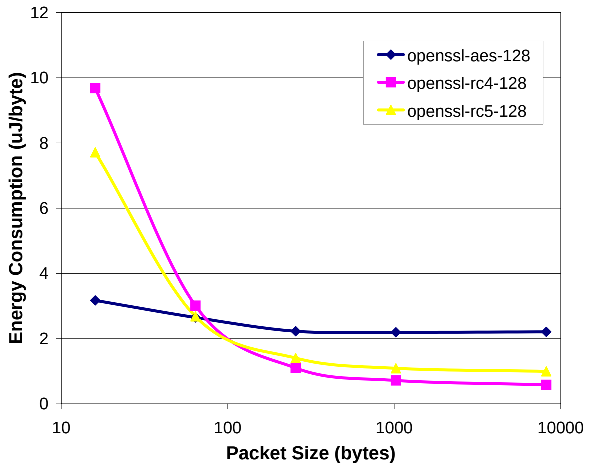


Figure III.18: Energy Consumption of 128-bit Crypto-Functions with OpenSSL

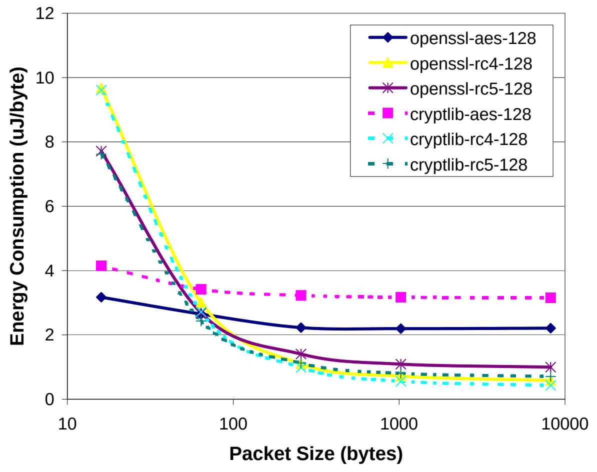


Figure III.19: A Comparison of Energy Consumption between OpenSSL and Cryptlib

The comparison of using different methods for energy measurement is shown in Figure III.20. The measurement of energy consumption uses the cycle-counting (on the left y-axis with solid lines) and the DAQ-system (on the right y-axis with dashed lines). Both methods show close performance, but on a different scale. The DAQ system method which measures the total energy consumed by the device shows much higher energy consumption.

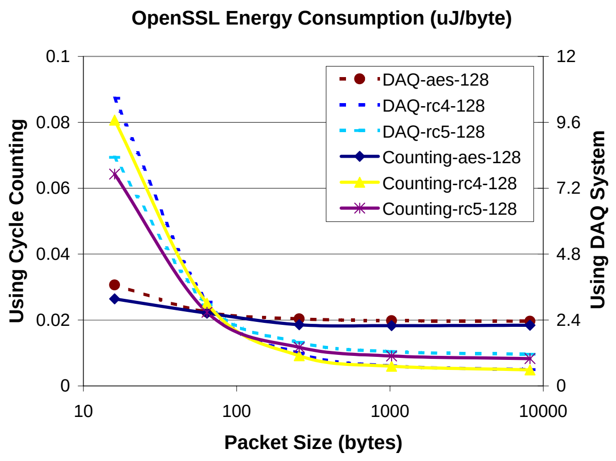


Figure III.20: A Comparison of Using Different Energy Measurement Methods

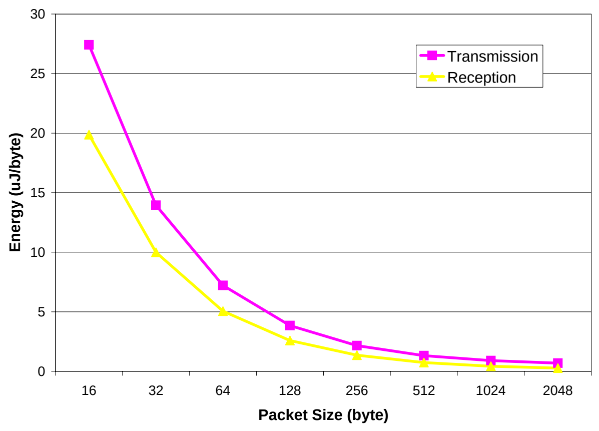
H. COMPARISON OF TRANSMISSION AND ENCRYPTION ENERGY

In this section, we compare the the energy consumption due to transmission with that due to encryption. To calculate the transmission energy, we use the energy transmission model for point-to-point transmission in IEEE 802.11b WLANs proposed by Feeney and Nilsson [45]. The transmission energy model is a linear model given by:

$$\text{Tx Energy} = 431\mu J + 0.48\mu J/bytes \tag{III.1}$$

$$\text{Rx Energy} = 316\mu J + 0.12\mu J/bytes \tag{III.2}$$

The transmission and reception energy have a fixed cost that is similar to that in encryption for key expansion process. We compare the transmission energy with our encryption energy model as proposed in Section III.I. Figure III.21 shows the energy consumption per byte of transmission and encryption as a function of packet size. It can be seen that the transmission energy is higher than the encryption energy. This is because the transmission requires a higher fixed cost or overhead to turn on the radio circuit on for transmission or reception. Note however that PKC algorithms consume around an order of magnitude higher energy than even transmission and reception.



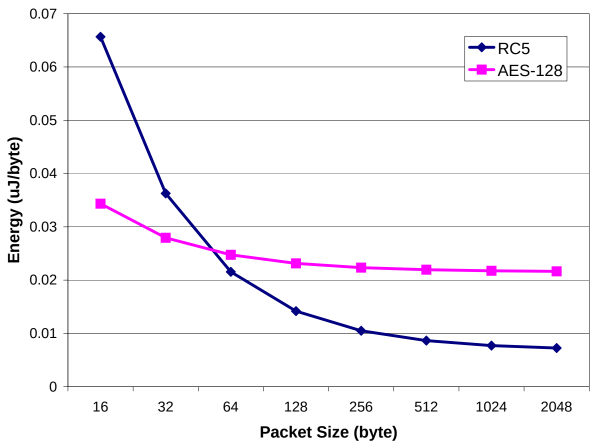


Figure III.21: A Comparison of Transmission and Encryption Energy

I. MODELING ENERGY CONSUMPTION

In our study in previous sections, we show the performance of cipher functions in term of energy consumption. In this section, we come up with models of energy consumption of the cipher functions in various settings. The energy consumption model is important in that it can be used for simulation of mobile devices to estimate the energy consumption for encryption. For example, for a given packet size and number of operational rounds, we can estimate how much the energy consumption will be for securing packets using a cipher such as AES or RC5. Additionally, from the packet size, we can estimate the energy consumption of Hash and MAC functions.

1. The Energy Model

As we have shown in previous sections, energy consumption per byte of cryptographic functions is a function of packet size. The characteristics of the energy consumption per byte is closed to the linear-in-parameter exponential function of as a function of the natural logarithm of the packet size. Therefore, we model the energy consumption per byte as a function given by:

$$\epsilon = \alpha + \beta e^{-\ln x} + \gamma \ln x e^{-\ln x}, \tag{III.3}$$

$$\tag{III.4}$$

where ϵ is the energy consumption per byte (in μJ/byte), and α, β, and γ are the model arguments, and x is the number of bytes.

To find the model arguments, we use curve fitting techniques to find the curve that best fits the energy consumption from measurement. By trial-and-error, we found that the above equation yields the best approximation or best fit for the measured energy consumption for all cryptographic functions. We use the least-mean-square fitting [119] technique with the above equation to find the energy model of each cryptographic function. We use the R-square or Coefficient Correlation to find the goodness of fit [118] and we show this for each function in later sections.

We can reduce the above equation into a short form, and we have the energy consumption per byte as follows:

$$\epsilon = \alpha + \frac{\beta}{x} + \frac{\gamma \ln x}{x}. \tag{III.5}$$

To calculate the energy consumption per packet, E, we multiply ϵ by x, the packet size in bytes, to get:

$$E = \beta + \alpha x + \gamma \ln x. \tag{III.6}$$

From this model, we see that the energy consumption per packet of cipher functions is almost linear as a function of packet size if we ignore the last term. The energy consumption per packet, E, depends on three costs, the fixed cost, the first variable cost, and the second variable cost. The fixed cost (β) is the energy overhead independent of the packet size, which is from the computation required for the key expansion process. The first variable cost, α, is dependent on the packet size. The second variable cost, γ, has a lower dependency on packet size than the first variable cost.

To make the encryption energy model simpler, one may ignore the second variable cost since it is not significant compared to the first two costs as the packet size increase. The model can be reduced into a simple form for encryption energy consumption per packet as shown below:

$$E = \beta + \alpha x. \tag{III.7}$$

2. Modeling Energy Consumption of Ciphers

As we have seen in Section III.B.1, standard ciphers such as AES, RC4, and RC5 consume different energy levels for different packet sizes. In this section, we use a curve fitting technique and the energy model in the previous section to find the model arguments (α, β, and γ) from the data in Figure III.1.

Table III.3 shows the arguments of the energy model for the cipher functions and the goodness of curve fit in term of R-square values for each cipher. We can see that the fixed cost, β, of RC4 is higher than that of other ciphers, and AES has a much lower fixed cost than RC4 and RC5. However, AES has a high variable cost, α, compared to RC4 and RC5.

Table III.3: An energy model of fixed-round ciphers

Cipher	R-Square	α	β	γ
RC4	0.99999999	0.005352241	1.570481104	-0.019608391
RC5	0.99999996	0.010556583	1.052393166	0.006131408
AES-128	0.99999824	0.021548394	0.205036074	-0.000142957
AES-192	0.99999619	0.025066828	0.188190654	0.000362155
AES-256	0.99999616	0.028178031	0.201417806	0.000399011

We use the arguments in the table and plot the energy consumption per byte, ϵ, of the cipher functions with different packet sizes as shown in Figure III.22. The resulting figure shows a good fit to the energy consumption from measurement in Figure III.1. We also show the energy consumption per packet, E, in Figure III.23. From the figure, we can see that the AES cipher consumes energy for encrypting a packet at a higher rate than RC4 and RC5 as the packet size increases.

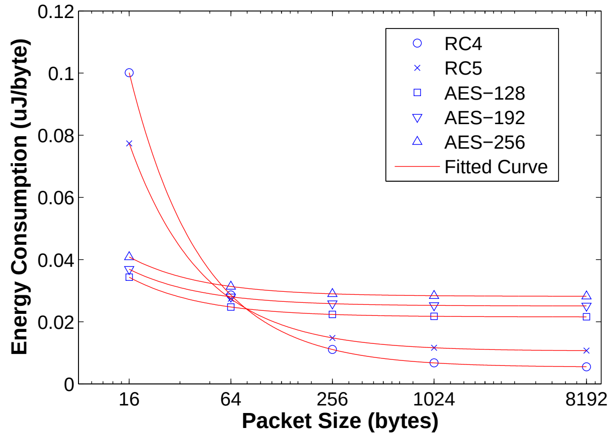


Figure III.22: The energy per byte model of encryption algorithms

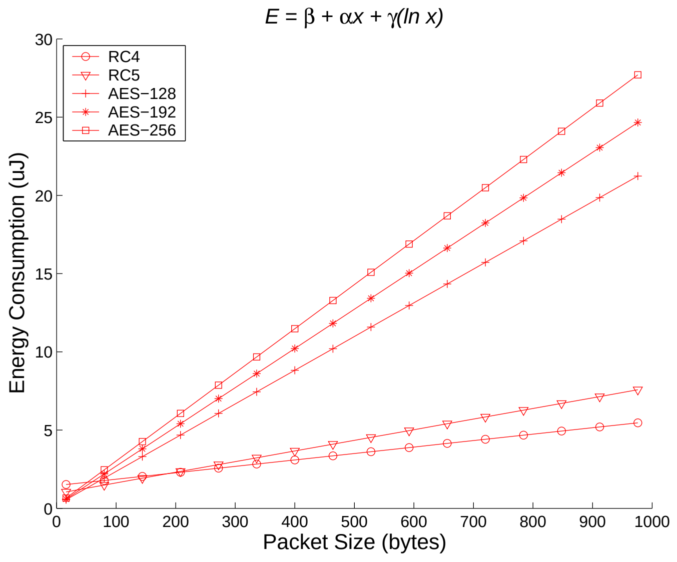


Figure III.23: The energy per packet model of encryption algorithms

From the model shown in Figure III.22, AES-128 performs better than RC4 and RC5 when the packet size is less than 80 bytes. However, this may not be true since AES and RC5 are block ciphers. When input data is not equal to an input block size, the data needs to be padded to the input block size. The input block size of AES and RC5 is 16 bytes and

8 bytes, respectively. Therefore, when a packet size is not of 16 bytes for AES or 8 bytes for RC5, the energy consumption per byte is increased as it needs to encrypt more bytes than the actual packet bytes.

Figure III.24 shows more granular measurements of energy consumption between packet size of 56 and 88 bytes. We see that AES actually performs better than RC4 and RC5 when the packet size is smaller than or equal to 64 bytes. Therefore, the above proposed model can only be used to estimate the energy consumption of AES and RC5 when the packet size is equal to the input block size. If the packet size is not equal to the input block size, we need to calculate the energy consumption of the packet using the input block size that is larger than the packet size using the energy model. For example, encrypting a packet of size 65 bytes consumes energy that equals the energy for encrypting an 80-byte packet using AES or a 72-byte packet using RC5.

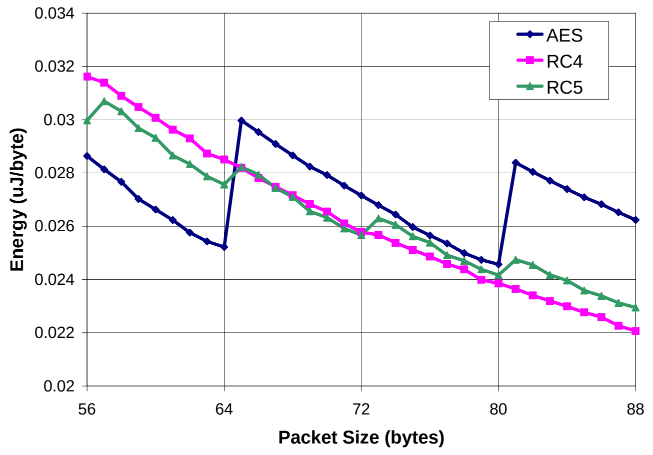


Figure III.24: The granular measurement of energy per packet

Figure III.25 shows the difference of energy consumption per byte between the granular measurement and using the energy models of AES, RC4 and RC5. The model provides a lower bound on the energy value.

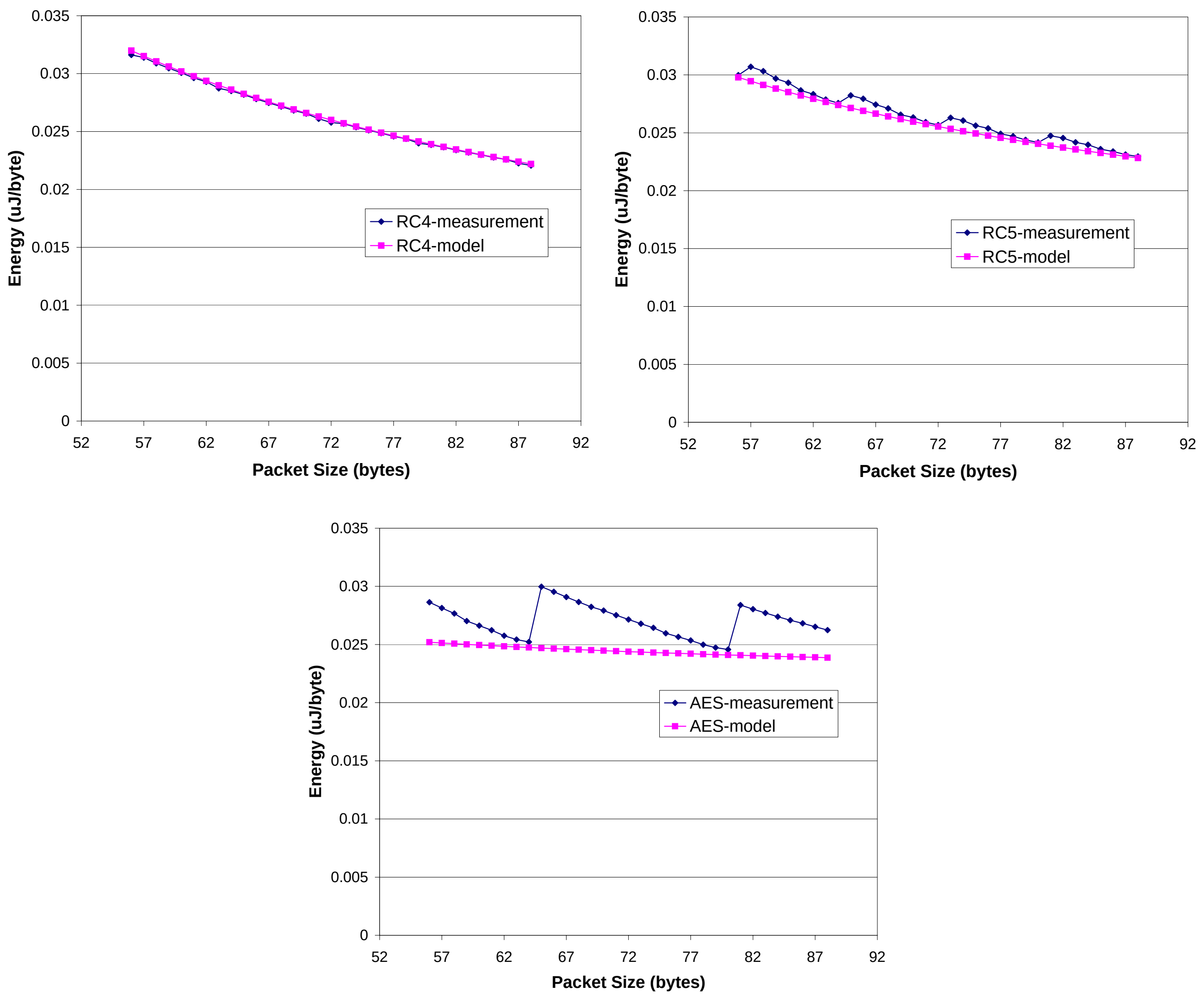


Figure III.25: Comparing the energy model to the granular measurement

3. Modeling Energy Consumption of Variable-Round AES and RC5

In Section III.B.3, we show the energy consumption of block ciphers such as AES, RC5, and Blowfish. From the results, we can model the energy consumption per byte based on the number of operational rounds and the packet size. We do not model the energy consumption of Blowfish since it will not be used later in our work. In any case, it will not be hard to model the energy consumption of Blowfish by using the same curve fitting technique.

From Figures III.5 and III.6, we use the energy model to find the model arguments and the goodness of fit for different numbers of operational rounds and packet sizes. Table III.4 shows the R-square values and the model arguments. From the table, we can see that the fixed cost component, β, of AES is fairly constant, while that of RC5 is increased as the number of operational rounds increases. This is because RC5 needs more computation for key expansion as the number of operational rounds increases. The increase of the number of rounds also increases the computation in the encryption process, but it is insignificant compared to the key expansion process. The RC5 encryption process is only composed of one XOR, one rotation and one addition operation for each round.

Using AES and RC5 with 14 rounds from Table III.4, we show the goodness of fit in Figure III.26.

Table III.4: An energy model for variable-round AES and RC5

Cipher	Round	R-Square	α	β	γ
AES	2	0.99999878	0.008447073	0.17643700	-0.000372095
	4	0.99999901	0.011159903	0.17790412	-0.000295925
	6	0.99999389	0.013974894	0.19367355	-0.004574693
	8	0.99999923	0.016615498	0.17672169	-9.27346e-06
	10	0.99999901	0.019417545	0.17720414	0.000132047
	12	0.99999865	0.022534813	0.17499212	8.82427e-05
	14	0.99999978	0.025258314	0.17694732	-0.000175299
RC5	2	0.99999829	0.007433666	0.25628641	0.002915492
	4	0.99999860	0.007427554	0.38893572	0.005036759
	6	0.99999960	0.007435403	0.52541720	0.002830619
	8	0.99999860	0.007424251	0.64547943	0.007890188
	10	0.99999986	0.007439175	0.79930360	0.001637456
	12	0.99999993	0.009364852	0.93294880	0.000983088
	14	0.99999997	0.007443185	1.07992994	0.000756165
	16	0.99999989	0.010977162	1.19542920	0.002287954

From Table III.4, we can model energy consumption of variable-round AES and RC5. Figure III.27 shows the figure of α and β cost components as a function of the number of operational rounds (r). We do not consider γ since it has very little significance in the energy model; therefore, we simplify our variable-round model to only α and β. From the figure, it is clear that α of AES and β of RC5 are linear functions of the number of operational rounds, and β of AES and α of RC5 are likely to be constant values. Using the curve fitting technique, we compute α of AES and β of RC5 as follows in terms of r, the number of operational rounds.

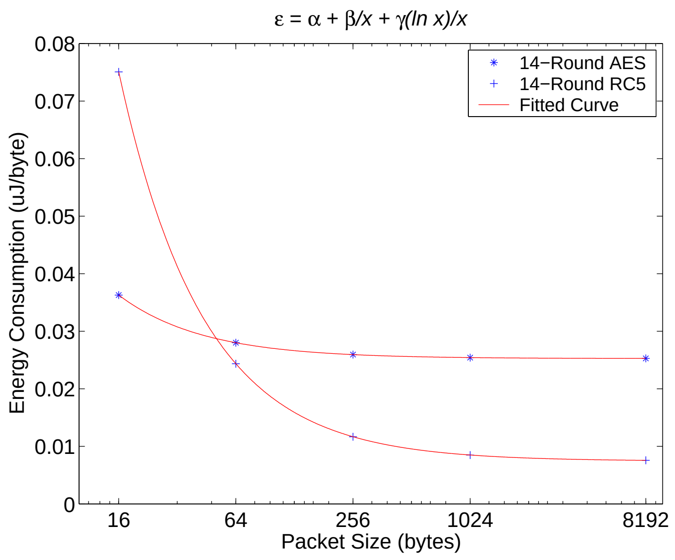


Figure III.26: Goodness of fitting for AES and RC5 energy consumption

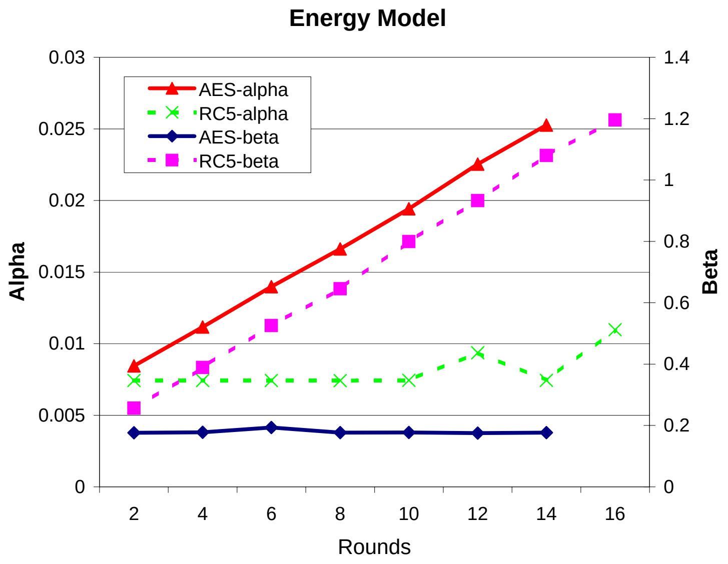


Figure III.27: The Energy Model of Variable Round AES and RC5

$$\text{AES:} \quad \alpha = 0.005582889 + 0.001398641r, \quad \text{R-square} = 0.999591509 \tag{III.8}$$

$$\beta = 0.176701065 \tag{III.9}$$

$$\text{RC5:} \quad \alpha = 0.007433872 \tag{III.10}$$

$$\beta = 0.116988204 + 0.067887565r, \quad \text{R-square} = 0.999409297 \tag{III.11}$$

By using α and β of AES and RC5 into Equation III.5, we have variable-round energy models of AES and RC5 as follows.

$$\text{AES:} \quad \epsilon = 0.005582889 + 0.001398641r + \frac{0.176701065}{x} \tag{III.12}$$

$$\text{RC5:} \quad \epsilon = 0.007433872 + \frac{0.116988204 + 0.067887565r}{x} \tag{III.13}$$

From the above equations, we can clearly see that AES has a smaller overhead than RC5 in terms of energy consumption as we discuss in Chapter II.A. As shown in Section III.I.2, the overhead of RC5 is smaller as the packet size increases for each encryption when operating using the standard number of operational rounds. When operating using more rounds, as shown in Figures III.6 and III.5, the overhead of using RC5 becomes greater while that of using AES is only slightly increased.

This can be clearly seen in Figure III.27. The overhead cost component, β, of RC5, linearly increases as we increase the number of operational rounds while that of AES is slightly increased. In contrast, the variable cost component, α, of RC5 is fixed while that of AES is linearly increased as we increase the number of operational rounds.

In summary, we conclude two rules for RC5. First, RC5 should only be used for encrypting large-size packets due to its high overhead cost. Second, when using RC5 for large packet size, we may increase the number of operational rounds to increase security strength *while the energy consumption is only slightly increased.*

4. Modeling Energy Consumption of Hash Functions

In Section III.D, we show the performance of hash functions such as SHA-1 and MD5 in terms of energy consumption per byte, and we use this data to find an energy model for them. From Figure III.11, we use the energy model and the curve fitting technique previously described to find the energy arguments and the goodness of fit.

Table III.5 shows the model arguments and the goodness of curve fit indicated by the R-Square values. We can see that SHA-1 has a much higher overhead (a fixed cost) than MD5. SHA-1 also has a higher variable cost since it performs more computations than MD5. Note that MD5 is known to have vulnerabilities.

Table III.5: An energy model for hash functions and packet size

Hash	R-Square	α	β	γ
SHA-1	0.99999324	0.00570248	1.61598213	0.06370296
MD5	0.99999999	-0.00000086	0.44510441	0.00036898

Using the arguments, we show the energy per byte model of hash functions in Figure III.28, and it shows a good fit to that in Figure III.11.

5. Modeling Energy Consumption of MAC Functions

In Section III.E, we study the energy performance of MAC functions such as HMAC and CBC-MAC. In this section, we model the energy consumption of the MAC functions using the results from the study. We use the curve fitting technique and find the model arguments and the goodness of fit in Table III.6.

From the table, we can see that the CBC-MAC-AES (using AES as cipher) has a much lower fixed cost, β, than HMAC-SHA. The fixed cost of the HMAC-SHA also increases as the size of the SHA output or digest is increased. The variable cost, α, of the HMAC-SHA is also increased as the digest size increases; however, it is increased at a lower rate than the fixed cost. The high overhead cost is due to the fact that SHA function already has a high overhead cost, HMAC function multiplies the overhead cost in that it performs two SHA functions internally to produce an output.

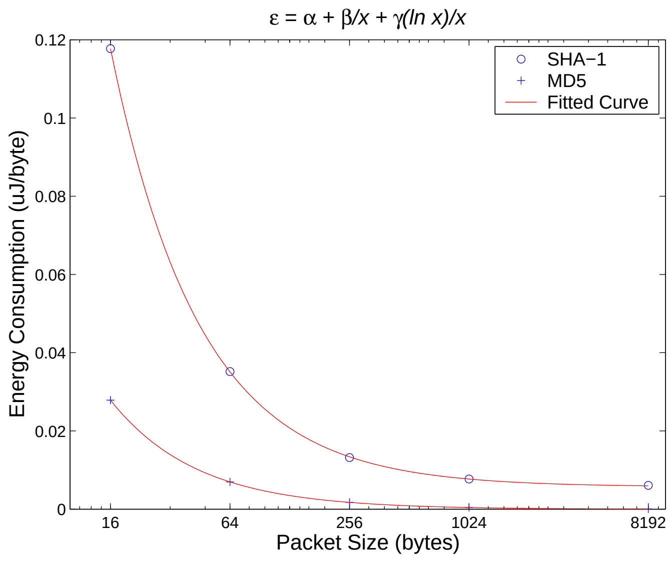


Figure III.28: The energy per byte model of hash functions

Table III.6: An energy model for HMAC-SHA and CBC-MAC

MAC	Output (bits)	R-Square	α	β	γ
HMAC-SHA	160	0.99892989	0.01097168	8.69571394	-0.30388744
	256	0.99973079	0.01898750	10.23319142	-0.07599955
	512	0.99991585	0.05268896	36.20744206	-1.35579240
CBC-MAC-AES	128	0.99552824	0.01912965	0.20543614	-0.01385172

We use the model arguments from the table and create a plot as shown in Figure III.29. The plot shows the goodness of fit for energy consumption per byte of HMAC-SHA with variable output sizes and 128-bit CBC-MAC-AES for different packet sizes.

J. CONCLUSION

It has been shown that several factors have an impact on the performance of security primitives of both SKC and PKC. The packet size can significantly affect the key expansion (or preparation) process of secret key encryption algorithms, although the key can be expanded

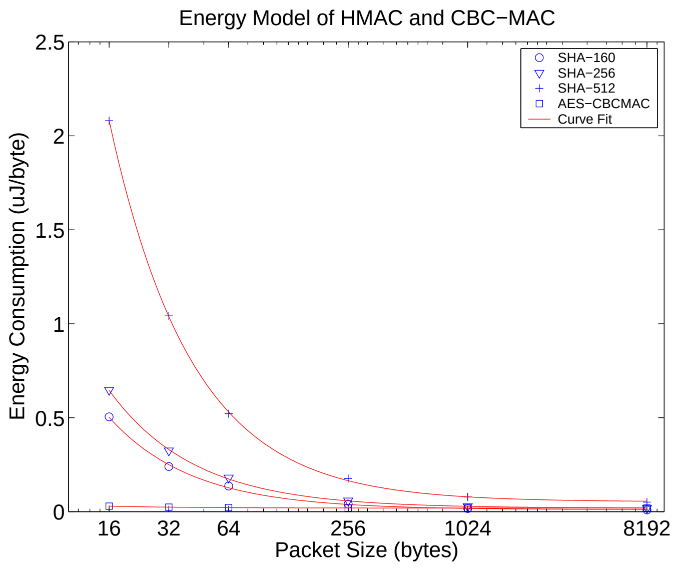


Figure III.29: Goodness of Fitting for MAC Energy Consumption

and cached for later use to reduce the computational load. However, cache management may not efficiently utilize the memory space especially in packet switching networks where data are bursty. It also increases the system complexity and may introduce a security loophole because of the buffered key. In addition, in wireless networks where packet error rates can be high, a per-packet key scheme is often used to provide security services to each packet. Thus, the packet size is a significant factor in determining the energy consumption.

The key size has only a slight impact on the performance of SKC algorithms such as AES. As suggested by the AES standard, increasing the key size of AES requires an increase of the number of operational rounds; hence, the computational load is increased. Thus, the impact for increasing AES operational rounds is more significant than increasing the key size. Compared to SKC, the key size of PKC algorithms can significantly affect the energy consumption. It is because increasing the key size implies an increase of mathematical operations, and hence increases the computation and energy consumption. The number of operational rounds can only affect the performance of block ciphers whose encryption process is recursive. Increasing the number of rounds does magnify the computation but increases security strength. Hash functions are often light-weight, and hence they are not of much concern. Although, the packet size may impact the performance of hash functions, it is insignificant compared to SKC and PKC functions.

We have shown that using different cryptographic libraries could also affect the performance of energy consumption. It is shown that OpenSSL and Cryptlib show similar results because they are both based on a C-programming interface which is lightweight. On the contrary, Crypto++ uses the C++ programming interface which requires more computation and more energy for error checking and memory/resource management.

We also use different methods of energy measurements to compare the performance of different cryptographic libraries for cryptographic operations. We use the hardware-based approach to measure the amount of current drawn to the laptop performing cryptographic operations to be compared with the cycle-counting approach. The results show that both methods can be used to measure the energy consumption of cryptographic operations. However, the cycle-counting approach yields only the amount of energy used by the CPU of a mobile device such as a laptop. On the other hand, the hardware-based approach shows the total energy consumed by the device including CPU, hard disk, LCD monitor and etc.

We also study and propose energy models for variable round AES and RC5 as well as the MAC as a function of packet size. The models show that the energy consumption is a linear-in-parameter logarithmic function with three components. These models can be useful in performance evaluation of energy consumption of security protocols, and will later be used in our study of security protocol performance.

IV. SECURITY STRENGTH: MEASUREMENT AND MODELING

In the following sections, we discuss how we can characterize the strength of a security protocol and how it can be estimated. Then, we propose a security strength model for use in security strength estimation. The study in this chapter has been largely extended from our work in [85].

A. MEASUREMENT OF SECURITY PROTOCOL STRENGTH

Fundamentally, the strength of a security protocol can be evaluated based on the underlying security properties and primitives used in the protocol: such as ciphers, key size, operational rounds, and mechanism of the protocol.

1. Strength from Cipher Algorithms

A cipher is a core component used in many security protocols. Each cipher is a unique algorithm which mostly includes data manipulation, permutation, and diffusion which builds the cipher strength. A cipher is known to be weak or insecure when it is possible to perform shortcut attacks, e.g., attacks that are more efficient than exhaustive key search. Such attacks are cryptanalysis attacks that analyze the cipher algorithm and try to reverse the ciphertext to plaintext or discover the key. Examples of cryptanalysis attacks are differential and linear cryptanalysis attack, weak key attack, related-key attack and Square attack [102]. Among these attacks, the differential and linear cryptanalysis attacks are most efficient especially for a block cipher like AES, RC5 and Blowfish. Details of these attacks are discussed later in this chapter.

2. Strength from Key Size

The size of the cipher key is also another factor that impacts the cipher security strength. When a cipher used is strong and there is no short-cut attack, the most efficient attack is a key search attack. The key search attack is simple and is most practical in that it can be mounted to attack almost all ciphers. In the key search attack, attackers exhaustively search for a key that matches a pair of known plaintext and ciphertext. Therefore, to strengthen the protocol security, not only do we need a strong cipher, but also a key size that is long enough to prevent the key search attack with a reasonable amount of time and resources. The shorter the key size, the more vulnerable the cipher becomes to exhaustive key search attacks.

Basically, the desired key size depends on how long we want our encrypted data to be safe. The longer the key size, the longer the encrypted data is safe. It would approximately take 5,300,000 years using a PC with a Pentium IV (3 GHz) to break a message encrypted with a 80-bit symmetric cipher or about 1 year using 5 million PCs. Using a 128-bit key with a provably secure cipher (e.g., AES) implies that the data can be protected until the year 2075 [69]. Increasing the key size does increase the security strength, but may also increase the amount of computation need by devices. Small devices such as wireless PDAs may be resource constrained and hard pressed to perform such computation. This can be more problematic if the devices have limited small battery capacity [99, 56]. Therefore, we may need to reduce the security level or strength to an optimal level so that such devices can operate longer.

3. Strength from Operational Rounds

The security strength of any secret key cipher is mainly based on the specific design of the encryption algorithm. Most encryption algorithms follow a design where there are repeated round operations. In each operational round, the algorithm manipulates data in two ways – scrambling it to obscure the relationship between the plaintext and ciphertext and dissipating the effects of each plaintext bit over several bits of ciphertext. As more rounds are used, the cipher tends to be more secure since it leaves no trails of the original data. The number of operational rounds is often used to determine the strength of a cipher against cryptanalysis attacks [102]. By employing the right number of operational rounds, one can ensure that there is no shortcut attack that can be performed faster than an exhaustive key search. A cipher with a smaller number of rounds than specified is weak against a shortcut attack such as a linear or differential cryptanalysis attack [73, 29]. For example, it is required to use 10 rounds in AES with 128-bit block size as specified in the AES standard [14]. However, the minimum number of rounds to provide adequate security is 6 rounds as described by the author of AES [41].

4. Strength of Protocol Design

Besides the first three factors, the protocol design itself is most important. Security protocols often have flaws due to improper design which leads to loopholes in the protocol. To evaluate a security protocol, we can use formalisms. The formalism is a mathematical method using logic to verify whether the protocol is secure [75]. However, the use of formalism is tedious and often has limitations that restrict the use to only security protocols with certain conditions and assumptions [18, 51]. A modern and common way to evaluate the protocol security is to use reasoning and arguments based on lessons learned from security protocol design [24, 17]. A quantitative way to assess the strength level of a protocol is to assess the strength of protocol security primitives such as the cipher and message authentication code (MAC) used in the protocol. In our study, we only assess strength of security protocols based on a quantitative method since formalisms are not the focus of our work. Note that while it is important to ensure that protocol flaws do not exist, the objective of this work is to demonstrate the effects on energy consumption rather than designing better security protocols.

5. Strength of Message Authentication Code

The security strength of a MAC or a hash function is often based on the output size or the hash size. An attack on a hash function is successful if there are two identical hashes an attacker can produce for two different messages. The equivalent exhaustive search is faster due to the birthday paradox [110]. The attack requires computation of about $2^{x/2}$ hash function outputs where x is the size of the output in bits in order to be successful with a 50% probability. Therefore, to be as secure as a 128-bit cipher, a hash function is required to produce an output of at least 256 bits.

In IV.1, we provide a categorization of security strengths of protocols that make use of different cryptographic primitives with different key sizes for the cipher and a hash output size for the MAC. For example, a protocol with low security strength uses a cipher with a 64-bit key and a MAC with a 128-bit output. This security setting is minimally secure because researchers have proved that the primitives can be practically compromised using a brute-force attack. For example, a 64-bit cipher (RC5) was broken using freely available computing power from about 330,000 PCs (involving distributed.net) in about 4 years [4]. Successively larger key and hash output sizes make a security protocol stronger.

Table IV.1: Typical Key Sizes and Hash Sizes for Different Security Strengths

Strength	Key size for cipher	Hash size
Low	64 bits	128 bits
Medium	128 bits	256 bits
High	192 bits	384 bits
Very High	256 bits	512 bits

6. Estimation of Security Strength

The measure of security strength is usually computational – the amount of time and money an attacker will need in order to break the security service. The computation which also implies the security strength is often measured in MIPS-years, defined as the amount of computation that can be performed in one year by a single DEC VAX 11/780. MIPS (million instruction per section) is widely accepted as a unit to approximate computing power of any processor (such as Intel Pentium) even if it uses instruction sets different from VAX. For example, in one year, a single PC with a 3 GHz Intel Pentium processor approximately provides the computing power of 3,000 MIPS-years or 0.003 MMYs.

The security strength or the MIPS-years required to break a cryptographic primitive indicates how long the data protected by the security protocol is secure. In the study by Lenstra and Verheul [69], the time to break a cipher is quantitatively compared to the time to break a DES cipher, the first standard block cipher internationally used since 1976 [5]. To break 56-bit DES, Lenstra and Verheul estimate that it requires about 0.5 MMYs. In 1997, it was first discovered that DES can be cracked using 3500 PCs and 4 months (using an exhaustive search for a key) [69]. In 1999, DES was cracked in less than 24 hours using a specialized hardware, “Deep Crack” [49], and computing resources from distributed.net [4]. From these evidences, it shows that 0.5 MMYs for cracking DES may be an overestimate.

From the MMYs required for DES cracking, we can estimate the amount of MMYs required to crack any cipher using an exhaustive key search, based on the key size used. The MMY estimation considers the increase of computing power which is approximately doubled every 18 months according to Moores Law [78]. The estimation also considers the amount of budget that an adversary may have and the declined price of CPU and memory to build an infrastructure for the brute-force attack.

In summary, the estimation using MMY is somewhat practical and can be used as a reference for selecting the key size of a cipher or the hash size based on what level of security strength is needed to prevent a brute force key search attack. Although, this estimation is applicable to a cipher or a crypto-primitive, the strength of a security protocol which is based on its underlying crypto-primitive can be associated to be the same as that of the crypto-primitive if protocol flaws do not play a huge role in the weakness of the protocol.

B. STRENGTH MEASUREMENT BASED ON SECURITY ATTACK

There are several types of security attacks, but only few are known to be effective and even practical. In this section, we describe some common security attacks and analyze the strength of ciphers used in our work against such attacks.

1. Brute Force Key Search Attack

The easiest and most effective attack is a brute force key search attack. In such attack, one obtains a plaintext and its corresponding ciphertext under a secret key, and simply tests each of all keys until a match is found. If the key size is n bits long, there are 2^n possible keys to test. This attack requires minimal storage/memory and it can be parallel. Therefore, the attack can be more effective by incorporating more than one computer. This attack is known to be the most practical since it can be mounted against any cipher. However, this attack is only effective when the key size is small. For example, RC5 with a 64-bit key was broken using the key search attack which takes 1,757 days with 331,252 participating computers. However, it would take only 790 days with approximately 45,998 2GHz AMD Athlon XP machines running 24 hours a day [6].

The efficiency of this attack depends heavily on the key size used. As recommended by NIST, a key size of 128 bits is safe for today's security and it would be safe until 2075 using the approximation proposed by Lenstra and Verheul [69]. In Figure IV.1, we estimate the amount of computing required in MMY (on the left y-axis), and the year (on the right y-axis) that it would be broken for a given key size using the same approximation technique.

2. Cryptanalysis Attacks

A cryptanalysis attack is an attack that uses a mathematical model to analyze the pattern of how a plaintext is "changed" to a corresponding ciphertext. It is a shortcut attack where the attack can be faster than the key search attack if it can be practically used. The goal of the attack is to find the "correlation" among a set of plaintexts, a set of ciphertexts and the key. There are two common cryptanalysis attacks, differential and linear cryptanalysis.

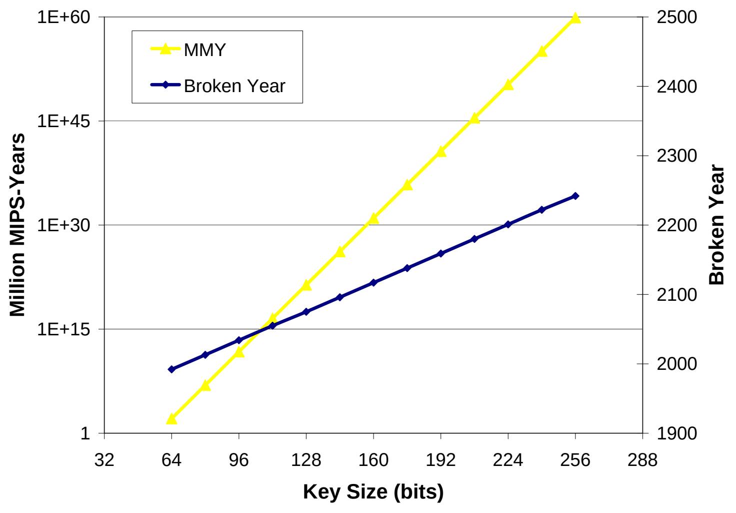


Figure IV.1: Required Computing Power for Brute Force Key Search Attack

Differential cryptanalysis was first introduced by Biham and Shamir [29], and has been widely used to analyze most block ciphers such as RC5 and AES. The basic idea in this technique is to find the "difference" between two chosen plaintexts and their corresponding ciphertexts. The plaintexts, with difference ΔP, are chosen such that the difference of the ciphertexts, ΔC, has a specific value with better than average probability. The tuple $(\Delta P, \Delta C)$ is called a *characteristic*. Depending on the technique used in the analysis and the cipher, certain bits of the key can be derived by analyzing the behavior of the characteristic.

The second common cryptanalysis attack is linear cryptanalysis pioneered by Matsui [73]. The basic idea is to find correlation among certain bits of plaintext, ciphertext and key that has a probability of less than or more than 0.5, which is called *bias*. With a certain bias, the correlation or *linear approximation* can lead an attacker to find information about the key. The cryptanalysis attack can be incorporated with a related key attack in which a set of keys is used for the attack, and the set is chosen with a particular property.

The cryptanalysis attack is often used with variants of a cipher, which can be a reduced-round cipher. It is easy to attack a cipher with a small number of rounds. As the number of operational rounds increases, the correlation complexity among plaintexts and ciphertexts grows exponentially. The attack commonly works for a small number of rounds. Increasing the number of rounds until it is impossible to attack or the attack is less effective than the key search attack is a common way of securing a cryptosystem. Fundamentally, the cryptanalysis attack is ineffective if it requires 2^{n-1} operations or more for an n-bit key.

From the above analysis, we propose a new security parameter to measure the capability of ciphers against the cryptanalysis attack. The *Cipher Robustness* or simply called the *Robustness* is the strength of the cipher to prevent the cryptanalysis attack. The Robustness is the product of the number of plaintext-ciphertext pairs and the number of operations required from the cryptanalysis attack.

$$\text{Cipher Robustness} = \text{Plaintext-Ciphertext Pairs} \times \text{Operations}$$

As the number of required plaintext-ciphertext pairs grows, the possibility that an attacker can collect those is decreased or impractical; thus, increasing the cipher robustness. In addition, the number of required operations is another factor that impacts the possibility of the attack. With a certain amount of available computing power, increasing the number of required operations decreases the ability of attackers from analyzing the cipher. Therefore, increasing the required number of operations does increase the cipher robustness.

Figure IV.2 shows the Robustness of AES(k,m) and RC5(k,m) where k is the key size in bits and m is the plaintext size in bits. The Robustness of AES is derived from [72, 41, 47], and that of RC5 is derived from [61, 30, 33]. From the figure, we see that the attack is more difficult as the number of operational rounds increases. Second, we also see that attacking AES is more difficult than RC5 as the number of rounds increases. However, one increased round of AES requires about 60% to 175% more energy than RC5 as shown in Figure III.4. Third, we can see from the figure that the attack of 256-bit AES with 9 rounds is more efficient than that with 8 rounds because the attack incorporates the Related Key attack of the long key size (256 bits) [47]. Therefore, as also recommended by AES authors, the number of operational rounds should be increased as the size of keys increased. Lastly, to make such attacks impossible, more than 10, 12, and 14 rounds are recommended for AES with 128-, 192-, 256-bit keys, respectively, and 16 rounds for RC5.

In comparison with the key search attack, Figure IV.3 shows the required computing

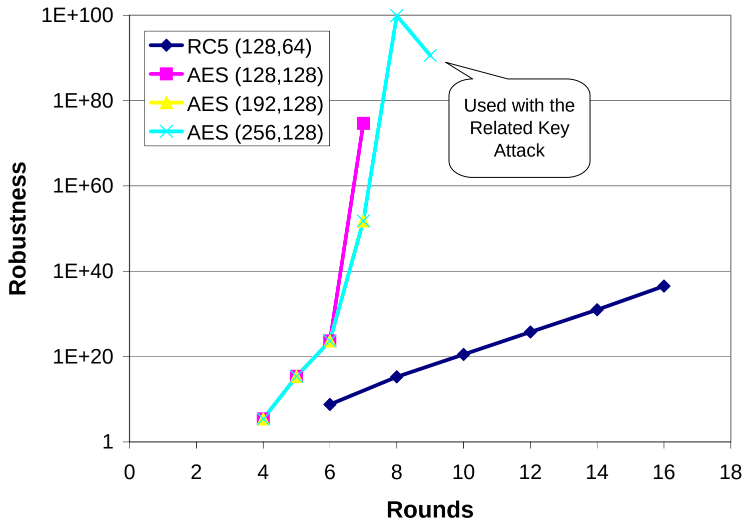


Figure IV.2: Cipher Robustness to Cryptanalysis Attack

power for a typical cryptanalysis attack against AES and RC5. If we assume that the amount of available plaintext-ciphertext pairs is unlimited (which is unlikely to happen in the real world), such attacks can be much more powerful than the key search attack since they require much less computational effort.

3. Modeling the Cipher Robustness

In the previous section, we described the cryptanalysis attack for block ciphers such as AES and RC5. The cryptanalysis attack relies on the number of operational rounds and the amount of available plaintext-ciphertext pairs. From this two parameters, we proposed a new parameter called *Cipher Robustness*. In this section, we model the Cipher Robustness based on the the number of operational rounds of a cipher. From the data, we use curve fitting techniques called least-mean-square (LMS) [119]. We derive the model of the Robustness. We use the R-square or the Correlation Coefficient value as a parameter to evaluate the goodness of the fitting [118].

The model is necessary for the calculation of the number of operational rounds based on

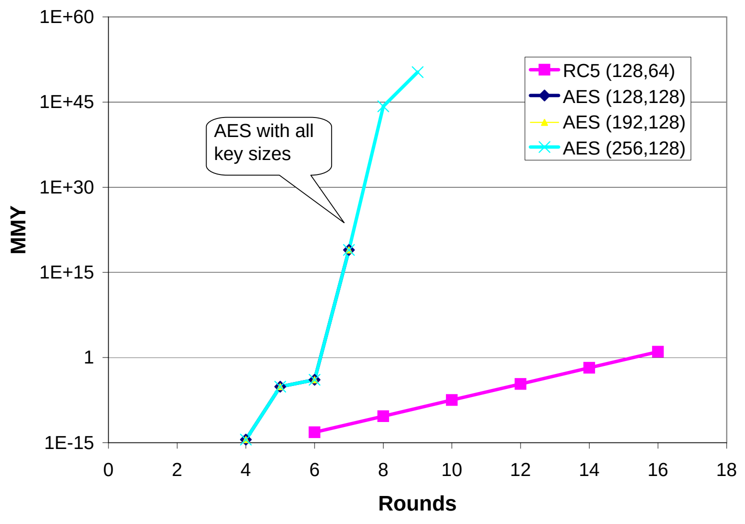


Figure IV.3: Required Computing Power for Cryptanalysis Attack

the required Robustness of a particular security level. For example, for a given security level, we calculate the Robustness, and we use the Robustness to derive the required number of operational rounds. We then operate the cipher with this number of rounds with the guarantee that the cipher will protect information with the given security level. The definition of the security level will be described in Chapter VI in more detail.

In Figure IV.2, we show the Robustness of AES and RC5 against cryptanalysis attacks. We can see that the Robustness of RC5 is exponential with the number of operational rounds. Thus, the Robustness model for RC5 is

$$\text{RC5 Robustness} = \alpha e^{\beta r}$$

where α and β are the model arguments, and r is the number of operational rounds.

We also model the Robustness of AES with a 128-bit key (AES-128) and a 256-bit key (AES-256). The Robustness of AES with a 192-bit key is similar to that with a 256-bit key. We can see from Figure IV.2 that the log function of the Robustness tends to be exponential as a function of the number of operational rounds. Thus, we can model the Robustness of AES as

$$\text{AES Robustness} = exp(\alpha e^{\beta r}).$$

We have tested several curve functions, and we have found that the aforementioned functions yield the highest R-square value. Using the LMS fitting on the above functions, we find the arguments for RC5, AES-128, and AES-256 and the R-square value as shown in Table IV.2.

Table IV.2: Robustness Model for AES and RC5

Cipher	α	β	R-Square
RC5	8.45715666e-8	6.28948775	0.99886109
AES-128	0.18749891	0.97279156	0.98547977
AES-256	1.00967057	0.67885200	0.99790237

From the table, we can summarize as follows. An increase of one round of AES-128 would increase the Robustness more than one round of RC5 since the Robustness of AES is exponentially increased as the number of rounds increases. One round of AES-256 increases the Robustness less than AES-128 due to the fact that attacks on AES-256 are more efficient than those on AES-128 and AES-192 (the β of AES-128 is higher than that of AES-256).

In Figure IV.4 and IV.5, we show the cipher robustness from the cryptanalysis and our robustness models for RC5, AES-128 and AES-256, respectively. We will later use these models to derive the number of operational rounds used by a cipher to provide a given security level.

4. RC4 Cryptanalysis Attack

RC4 is a stream cipher and the cryptanalysis of RC4 is different from that of block ciphers such as AES and RC5. The cryptanalysis attacks are related-key attack, "tracking" attack, and statistical analysis attack [52, 77, 10]. The attacks try to predict the states of the key stream; however, it is not efficient if the RC4 word size is more than 8 bits which is widely used for RC4. The most successful and practical attack is the Fluhrer-Mantin-Shamir (FMS) attack, where they found a weakness in the key scheduling algorithm (KSA) of RC4 [48].

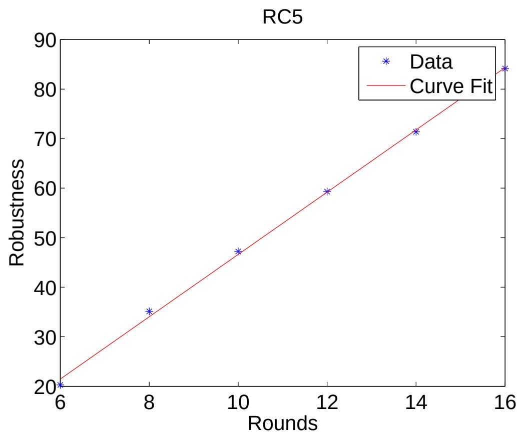


Figure IV.4: The Models of Cipher Robustness for RC5

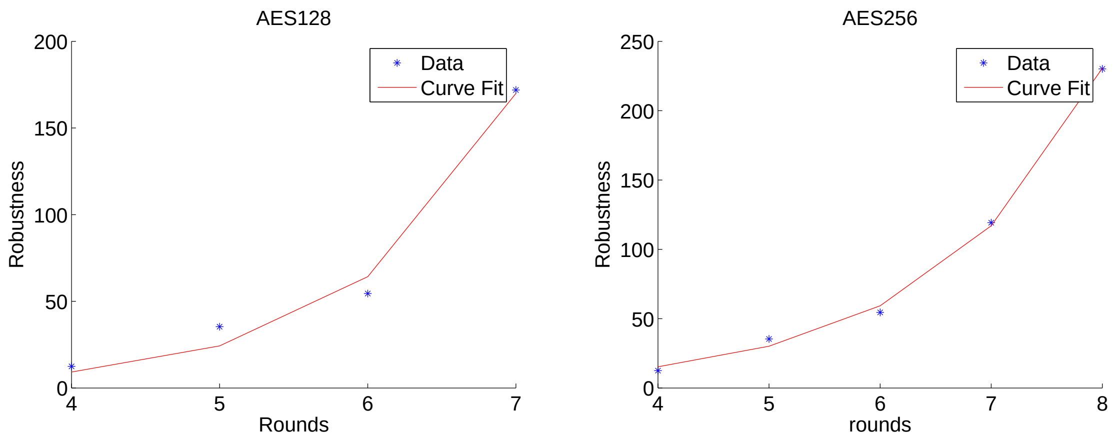


Figure IV.5: The Models of Cipher Robustness for AES

The FMS attack is practical as demonstrated by Stubblefield et.al. [111], and has been implemented in many attack tools, such as Airsnort [7] and WEPcrack [8]. The FMS attack has discovered two weaknesses in RC4, the *invariance* weakness and the initial vector (IV) weakness.

The invariance weakness is the existence of specific patterns, which are invariant with respect to the KSA. That means if we can find some pattern in RC4 keys, we can also find a similar pattern in the key stream output with high probability. The RC4 keys that have such patterns are weak keys. Using this weakness, attackers only find the desired pattern in the ciphertext (which is commonly an XOR between the key stream and a plaintext), and they can determine the secret key. The complexity of this attack is an $\Theta(2^{n+l})$, where n is the word size in bits and l is the key size in bytes. Typically, RC4 has 8-bit word size, $n = 8$ and a 128-bit key, $l = 16$, and the complexity of the attack is $\Theta(2^{24})$.

The IV weakness occurs when RC4 is used in a common mode in which an IV is simply concatenated to a secret key to produce an RC4 key. In this mode, the secret key can be used on a long term basis, and one can only change the IV to generate a different RC4 key. The IV is not a secret; therefore, it is exposed to attackers. With some known IVs to produce RC4 keys, an attacker can analyze key stream outputs, and determine the rest of the RC4 keys whose part is the long-term secret key. The complexity of this attack is $\Theta(2^{n+8})$, which is independent of the key size (l). This attack has been used against the WEP protocol, which uses the aforementioned common mode, and it makes WEP unsafe for any key size. The complexity of the WEP attack is $\Theta(2^{16})$, where the word size (n) is 8 bits long.

Rivest, the creator of RC4, recommends the prevention of these attacks by discarding the first 256 bytes of the key stream output since these bytes can be used by the key-related analysis. Also, the key and the IV should be pre-processed before they can be used as the RC4 key [93]. The discarding and pre-processing can prevent the IV weakness attack; however, as described by Fluhrer et.al., the invariance attack is still possible with lower probability [48]. The more the first several bytes are discarded, the less the successful rate of the invariance attack.

5. Hash and MAC Function Attacks

There are three common attacks for hash functions, which are based on brute force search, the *pre-image* attack, the *second pre-image*, and the *collision attack*.

The pre-image attack is that given the hash of the message M, $h(M)$, one is trying to find M' such that $h(M') = h(M)$. With the brute force search, the attack tries to find all possible outputs of hash function or digests. With the output size of n bits, the required number of operations is about 2^{n-1} on the average. The second pre-image attack is that given

$h(M)$ and message M, one is trying to find M' such that $h(M') = h(M)$. The difficulty of the finding is as hard as that of the pre-image attack.

The collision attack is more efficient than the first two attacks. Instead of trying to find a match from a given hash, one is trying to find a pair of messages, M and M', such that $h(M) = h(M')$. This is similar to the problem of how many people would be in a room so that two people have the same birthday. The answer is 23 people as opposed to 183 (if we use the brute-force search). This attack is also called *birthday paradox attack*. Thus, with a n-bit output, the attack approximately requires only $2^{n/2}$.

Due to the birthday paradox attack, maintaining the security level of a hash function requires the output or hash size to be at least twice as much as the size of the secret key. Table IV.3 shows hash sizes of common hash functions, the required operations, the MMYs of the operations, and the collision time[1] of the birthday paradox attack.

Table IV.3: The estimated collission time of hash functions under the birthday paradox attack

Hash	Output Size	Required Operations	MMY	Collision Time
MD5	128 bits	2^{64}	128	15.57 days
SHA-1	160 bits	2^{80}	8.38E6	2796 years
SHA-256	256 bits	2^{128}	2.36E21	7.87E17 years
SHA-384	384 bits	2^{192}	4.35E40	1.45E37 years
SHA-512	512 bits	2^{256}	8.03E59	2.67E56 years

From the table, it is suggested that we select the right hash function to generate the hash output according to our security need. If our security need is that we can prove the integrity of a messsage for the period of 16 days since the hash of the message is generated, we should not use MD5. However, a recent attack has discovered a shortcut attack which makes the collision time even less [116], and it is proved by an implementation that we can find the MD5 collision within 8 hours using a 1.6 GHz Intel Pentium PC [64]. SHA-1 also has a known shortcut attack which requires about 2^{69} operations to find the SHA-1 collision [117].

[1]The collision time is calculated using one 3 GHz PC capable of having about 3000 MIPS

The attack for MAC functions is a bit different from hash functions since the MAC functions requires a key to generate a message digest. In addition to those on hash functions, attacks on MAC functions are related to and limited by the key size. Sometimes the attack is specific to the algorithm of the MAC function. However, the common attacks on MAC functions can be divided into two categories: the *forgery attack* and the *key recovery attack* [89].

The forgery attack tries to predict the output of the MAC function of the message M, $Mac_K(M)$, without knowing the k-bit key K. The simplest forgery attack is the *MAC guessing* attack that one selects an arbitrary message and a randomly chosen MAC output, and hopes that the verification of the message matches the MAC output. Ideally, the probability of success is $1/2^n$, where n is the MAC output bits. The attack needs on the average 2^{n-1} operations for correct verification of the forgery. Such attacks can be avoided by making n sufficiently large and/or limiting the number of forgery attempts on a given key. Without knowing the key, the birthday paradox attack can also be used which requires about $2^{n/2}$ forgery attempts for a given key.

The key recovery attack to trying to find the key K from a number of the pairs $(M, Mac_K(M))$. The simplest key recovery attack is the brute-force search attack which can be used to recover the key from a few known $(M, Mac_K(M))$ pairs (about k/n pairs). The key recovery attack can be specific to how the MAC fucntion is constructed. More details of attacks on MAC functions can be found in [89, 76, 35].

Table IV.4 shows the security level of MAC functions, WPA Michael, 802.11i CBC-MAC, and HMAC, equivalent to that of ciphers, and the number of forgery attempts[2] which an adversary may need to perform for the forgery attack, and the estimated key lifetime[3].

From the table, it is suggested that the key used with the MAC functions be changed before the number of the plaintext-MAC processing reaches the forgery attempts. The number of forgery attempts can also be used with, for example, a failure counter as a countermeasure to limit the number of attempts from an adversary.

Note that Michael is a 64-bit MAC used in Wi-Fi Protected Access (WPA) standard while

[2]We assume the MAC key is not changed during the period of the forgery attempt.

[3]To calculate the key lifetime, we assume one forgery attempt takes about 200 msec

Table IV.4: The estimated key lifetime of key in MAC functions under the forgery attack

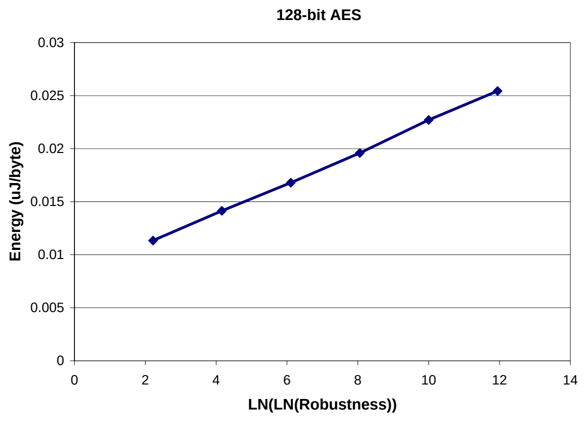
MAC	Output Size	Equivalent Secret Size	Forgery Attempts	Estimated Key Lifetime
Michael	64 bits	20 bits	2^{20}	58 hours [4]
CBC-MAC	64 bits	32 bits	2^{32}	9942 days
HMAC-SHA1	160 bits	80 bits	2^{80}	2.798E18 days
HMAC-MD5	128 bits	64 bits	2^{64}	4.270E13 days

CBC-MAC (Cipher Block Chaining-MAC) is used in WPA version 2 (WPA2) standard. The equivalent secret size of Michael is lower than 32 bits because it has a known weakness [46]. The weakness is further described in the next section. The WPA2 CBC-MAC output is the first 64 bits of the output of an 128-bit AES with CBC-MAC.

C. ENERGY AND ROBUSTNESS

In previous sections, we show that the increase of security level, which increases the Cipher Robustness, increases the amount of energy consumption. In this section, we compare the energy consumption and the Cipher Robustness of AES and RC5. AES is a newer cipher than RC5, and it is known to be stronger from our study. In Section IV.B.3, we show that the Cipher Robustness of AES is higher than that of RC5 as one operational round is added. Figure IV.6 shows the energy consumption as a function of the Cipher Robustness of AES and RC5.

From the figure, we see that one additional operational round of RC5 slightly increases the energy consumption and linearly increases the Cipher Robustness. However, one additional operational round of AES linearly increases the energy consumption, but *exponentially* increases the Cipher Robustness.



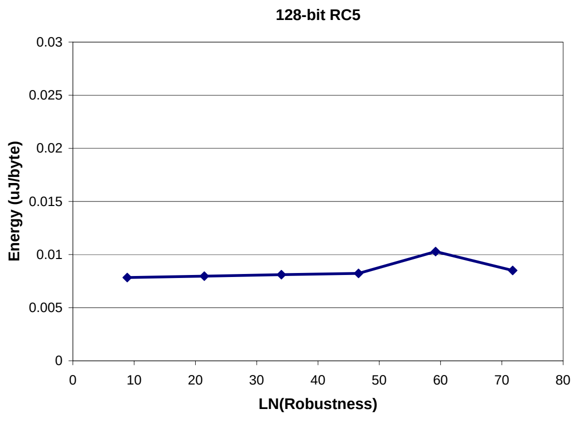


Figure IV.6: Energy Consumption VS Cipher Robustness

D. SECURITY STRENGTH ANALYSIS OF WLAN SECURITY PROTOCOLS

In this section, we show an example of how we can estimate the security strength of a wireless security protocol. We will evaluate the security of Wired Equivalent Privacy (WEP) protocol, Wi-Fi Protected Access (WPA) and the recently standardized IEEE 802.11i or WPA version 2 (WPA2). These are standard protocols for providing security services in Wireless Local Area Networks (WLANs).

1. WLAN Security Protocols

WEP was first used as the standard security protocol in WLANs. WEP employs RC4 cipher as the underlying crypto engine for data encryption and authentication. WEP has known weaknesses partly due to the improper use of RC4 and the key generation which simply concatenates a secret to a counter (an initial vector). This has lead to a practical attack on WEP [111]. WPA was then standardized to fix the WEP weaknesses. WPA introduces TKIP (Temporal Key Integrity Protocol), a key mixing process to mitigate the weakness of the key generation, but still uses RC4 as a cipher. RC4 is known to be weak due to its key scheduling algorithm, the process that expands the secret key of any size to 256 bytes before generating a stream of random numbers for encryption [48]. Therefore, the WPA2 standard was released in July 2004 to be used as a strong, flawless security protocol for WLANs. WPA2 is totally re-designed and utilizes AES as the underlying cipher in Counter with CBC-MAC (CCM) mode [120].

For message integrity, WEP employs the Cyclic Redundancy Check-32 (CRC-32) which is not a MAC and cannot be used for strong message authentication. WPA utilizes a MAC function known as *Michael* that produces a hash output of 64 bits. Due to its simplicity and lightweight, Michael has been subject to active attacks that actually reduces the security of the output from 64 bits to 20 bits (instead of 32 bits due to the Birthday Paradox attack) [46]. In the attack, one simply tries to test two different messages that have similar corresponding outputs. As more tries are tested, one can find a collision where two different messages yield the same output for a given Michael key. This Michael design is intended to have low security since Michael is needed to be compatible and affordable for legacy devices which have low computing power.

The re-designed WPA2 employs a better MAC, CBC-MAC which is stronger and has no known weakness. It utilizes the AES cipher in CBC mode to generate MAC from the upper 64 bits of ciphertext. This enables a verification process from ciphertext received before actually decrypting it. The summary of standard security protocols for 802.11 WLANs is shown in Table IV.5.

Table IV.5: Summary of Security Protocols for WLANs

Security Services	WEP		WPA	WPA2/IEEE 802.11i
	WEP-40	WEP-104		
Encryption	40-bit RC4	104-bit RC4	RC4 128-bit TKIP	128-bit AES CCM
MAC	N/A	N/A	64-bit Michael	64-bit CBC-MAC

2. WLAN Security Protocol Analysis

Based on the specification of WLAN security protocols, we calculate the time required to break a security protocol with computing power equivalent to a typical high-performance PC available today which is an Intel Pentium IV 3 GHz CPU. The security strength of ciphers are shown in Table IV.6. The time can be reduced if multiple PCs are used together to break the security in a distributed network. From the table, we see that 40-bit WEP can be broken in less than an hour using one PC. As the key size increases, more computations are needed to break the protocol. The MAC of WPA can be quickly broken since Michael is susceptible to active attacks where an attacker tests two messages sent to a receiver with the hope that the messages have the same output, although more tests are needed for the attack to be successful. Therefore, it is required by the IEEE 802.11i standard to use countermeasures such as an integrity check failure counter to monitor such attacks when Michael is used [16]. The maximum number of the counter should not exceed the number of forgery attempts as shown in the previous section. WPA2 is the strongest protocol which has been rigorously reviewed for many years by the 802.11i working group and security experts.

Table IV.6: Strength of WLAN Security Protocols

Protocol	MAC Strength	Cipher Strength		Overall
	Forgery Attempts	MMY	Broken Year	Strength
WEP-40	N/A	$7.629E-6$	Now	Very Low
WEP-104	N/A	$1.407E14$	2044	Low
WPA	2^{20}	$2.361E21$	2075	Medium
WPA2	2^{32}	$2.361E21$	2075	High

In this analysis, we have not considered the protocol flaws in measuring the security strength. WEP uses ICVs as MACs and they are known to be useless. Consequently we do not estimate the strength of the MAC here. Attacks against ciphers that are better than exhaustive search have also not been considered primarily because of the difficulty of estimating the security strength. For example, the FMS attack could drastically reduce the amount of time to break 104-bit RC4. The attack is based on the usage of weak keys and this is hard to estimate. Additionally, TKIP is assumed to be as secure as AES although it employs the weak RC4 algorithm since no attack has known to be faster than the exhaustive key search attack at the time of this writing.

E. CONCLUSION

The strength of a security protocol can be measured based on the underlying crypto-primitives used, key size, the number of operational rounds, and the protocol design. Commonly used primitives are the cipher and the hash function or message authentication code. The strength of the cipher often relies on the key size, operational rounds and algorithm used. The key size is a countermeasure to the brute-force search or the key search attack while the operational rounds and the algorithm are countermeasures to cryptanalysis attacks. The strength of hash or MAC functions only relies on the output sizes. To have the same security level of the cipher against the brute-force search attack, the output size should be twice as much as the cipher key size. Applying the security strength analysis to existing WLAN standard protocols shows that the current WEP and WPA are not so secure while the newest standard, WPA2 or IEEE 802.11i, is fairly secure against all known attacks.

V. AN ENERGY EFFICIENT SECURITY FRAMEWORK

We have shown that providing security services can consume different amounts of energy depending on a variety of security parameters such as number of operational rounds, packet sizes per encryption, key sizes, and choice of ciphers. We have also shown that security services can be provided at different levels, each of which correspond to different security strength or robustness. A requirement of high security strength needs a high amount of energy for computation. From these observations, it is possible that we can trade off security strength to provide higher energy efficiency to wireless and limited battery-power devices. We may lower the security strength to a sufficient level in order to conserve more energy, and hence to prolong the devices' battery life.

In this chapter, we propose an energy efficient security framework which proposes methods to reduce the amount of energy consumption. We also apply such methods to security protocols to achieve energy savings while maintaining the security level of the protocols.

A. METHODOLOGIES

Based on the evaluation of energy consumption of security primitives, we see that there is potential for reducing the energy consumption of security protocols for wireless networks. We propose three different methods by which energy can be conserved that can be applied towards the design of energy efficient security protocols.

1. Method 1: Eliminating/Replacing Most Energy Consuming Parts

The first and obvious way to reduce energy consumption is to eliminate or replace the most energy consuming part of existing standard protocols. Besides transmission and reception of packets, the most energy consuming part of the security protocols is due to cryptographic functions such as encryption, key generation, and digital signatures. From our findings as described earlier, different primitives consume different amounts of energy. Looking at several standard protocols, we also find that most of them allow different primitives to be used during a protocol transaction, but the protocols are not adaptive after a primitive has been selected for use for a whole session. However, we believe it is possible to carefully select a combination of primitives to be used in a single session, which is more energy-efficient than a standard one. As an example, assuming that both RC4 and AES with 128 bit keys provide similar security levels, it is better to use AES for shorter packets and RC4 for longer packets to provide confidentiality as it is in current protocols.

2. Method 2: Modifying Protocol Primitives and Transactions

The second method to reduce energy consumption of security protocols is to modify the protocols themselves. Beyond changing the security primitives, we can modify the protocol primitives (messages) and transactions. Some protocols are not energy-efficient since they have to exchange several primitives and messages; for example, a Kerberos client needs to exchange at least six messages before it can access network services [80] and employing it on a WLAN may result in a larger consumption of energy than required. However, the protocol modification should not change the security level already defined in the existing protocol. To ensure security equivalence, formal methods are often used to verify security strength and to find flaws of a protocol. A well-known formal method is BAN logic [37]; however, it has limitations in that it does not capture the aspects of real systems [18, 51]. In a different way, we can also show security equivalence by reasoning or arguments based on lessons learned [24, 17]. To implement protocol modifications and achieve energy savings, we can make use of security protocol standards that allow us to ”plug-in” any security transaction. One example of such a protocol for authentication is the Extensible Authentication Protocol (EAP) [31].

3. Method 3: Using an Entirely New Security Framework

The last method we suggest is to take the greenfield approach where we start from scratch and come up with security protocols that consume the least energy. Such a method should offer tunable and adaptive security services. As an example, consider data integrity for packets over a WLAN. MD5 is more efficient than SHA-1 for shorter packets, but it is also less secure. MD5 may still be used for shorter packets that do not need a lot of security (it would be sufficient if the security is not breached for a few days instead of a few years). Another example is a modified communications/security protocol that adapts itself to channel conditions. For instance, knowing that the channel is bad, short probe packets can be transmitted with a low level of security (simply providing confidentiality for the duration of the probes). Both of these examples require the development of a policy for determining how much security is required for a given communication session (what is reasonable) and how this can be mapped into the appropriate primitives. We believe this method that tunes the security protocol based on a policy and known performance measures would significantly reduce energy consumption due to adaptive security provisioning and may also reduce the energy consumed by signal/message transmission. We develop an example in the next chapter, but do not consider channel conditions there. We also do not address the development of policies in this dissertation nor do we consider optimality of the solution.

4. A Summary of Methodologies

A comparison of the three energy saving methods for security protocols is shown in Table V.1. From the table, the first method improves the energy saving by simply analyzing several existing security protocols and changing some protocol components. This method will still be compliant with existing standards; however, the energy saving is probably small. The second method probably offers more energy saving. However, it may or may not be backwards compatible with existing protocols, but it is within the scope of the standard. This method may also introduce a little complexity. The third method starts from scratch and creates a security protocol that utilizes a new energy efficient framework. This method introduces complexity ranging from medium to high and probably will not comply with any existing standards; however, we believe it has the potential to conserve the most energy.

Table V.1: A Comparison of Different Energy Saving Methods

Methodology	Method 1	Method 2	Method 3
Complexity	low	low	medium-high
Compliance	yes	maybe	no
Adaptability	no	no	yes
Energy Savings	low	medium	high

Another method has been proposed by many researchers as one of the energy-efficient methods, although this method is not related to security level adjustment [84]. This method aims to reduce the size of transmitted messages of a security protocol. It aims at reducing the energy consumed by the message transmission using several compression techniques. Though the compression method probably offers some energy saving, we will *not* use this method to reduce energy consumption of security protocols as it is not specific to security protocols alone.

5. Applying the Framework

In next sections, we show two simple examples of using the proposed energy saving framework described in Section V.A to conserve energy for security provisioning in WLANs. In the first example, we use Method 1, where we replace the most energy consuming part with something that consumes lesser energy. This example looks at a very simple variation of the handshake protocol for SSL where the energy consuming RSA signature algorithm is replaced by ECDSA, but other RSA processes are left as they are. In the second example, we employ a technique that is a part of Method 2. We adaptively change the encryption algorithm in a simple home WLAN. In the current WLAN standard, every packet transmitted over a wireless channel can be encrypted with RC4 algorithm in the secure mode [13]. However, to provide ”reasonable” security, the encryption needs not be equally applied to all packets. As an example, if only confidentiality services are required for some packets, encryption should only be applied to confidential data packets, and not to control and management packets such as the beacon and acknowledgment packets. Also if the packet size is small, we may use

AES instead of RC4 since AES has less overhead [88]. Employing both RC4 and AES and switching between them are not possible in the current WLAN standard, but this example shows the potential energy savings that can be achieved under a good framework.

B. A VARIATION OF TLS AUTHENTICATION FOR ENERGY SAVING

In this section, we utilize our proposed framework to an existing standard security protocol to be more energy efficient. As we described in Section V.A, one of the methods suggested by our framework is a replacement of security primitives that consume more energy than it is necessary.

Based on the framework, we propose the replacement method to the authentication protocol of Transport Layer Security (TLS) protocol or the Handshake protocol. The Handshake protocol negotiates a fixed set of cipher (called a ciphersuite) and a session key that will be used for a TLS session. The ciphersuite is commonly based on RSA for authentication and key exchange. However, from our study, ECC has some advantages over RSA for authentication. Therefore, the goal in this work is to study the replacement RSA algorithms that consume high energy with ECC in the handshake protocol. We use *both* RSA and ECC in the ciphersuite to make the standard TLS handshake protocol more energy efficient. This work is extended from our work in [87].

1. TLS Standard Protocol

The TLS protocol or previously called Secure Socket Layer (SSL) protocol is widely used to secure various network application protocols. It is the de facto standard for web-based transactions for e-commerce (such as online banking and shopping). It is used to secure application protocols such as email (IMAP, POP3, and ACAP) and network file systems (NFS). Recently, it has been used as part of the Extensible Authentication Protocol (EAP) standard and IEEE 802.1X standard for authentication in WLANs [19, 15].

Figure V.1 shows the protocol primitives of the TLS Handshake protocol [43]. First, the Mobile Station (MS) client and the server exchange random numbers (or nonces) and negotiate a "cipher suite" with *ClientHello* and *ServerHello*. The cipher suite specifies which cryptographic algorithms can be used to provide confidentiality, authentication and data integrity. The server then sends its certificate signed by a Certificate Authority (CA) which is trusted by the MS and the server. The server also requests client authentication with *CertRequest* message. The MS uses *PKC_Verify* process to validate the server's certificate by using the CA's public key (which is distributed beforehand). If it is valid, the MS obtains the server's public key from the server's certificate.

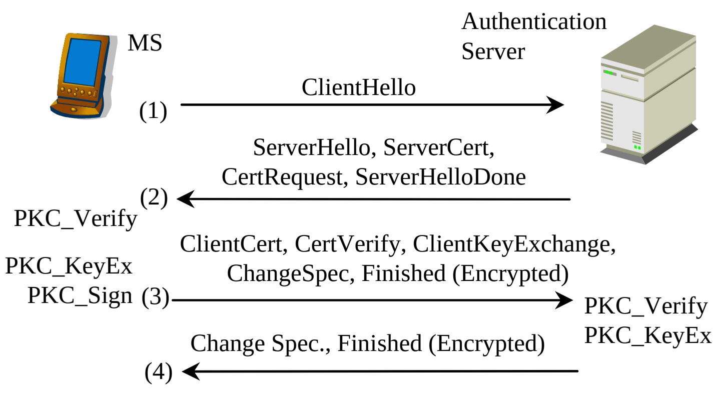

MS
Authentication Server
(1)
ClientHello
ServerHello, ServerCert, CertRequest, ServerHelloDone
(2)
PKC_Verify
ClientCert, CertVerify, ClientKeyExchange, ChangeSpec, Finished (Encrypted)
PKC_KeyEx
PKC_Sign
(3)
PKC_Verify
PKC_KeyEx
Change Spec., Finished (Encrypted)
(4)

Figure V.1: The Handshake Protocol with Mutual Authentication

The MS then sends its certificate (*ClientCert*) signed by the CA to the server. The server validates the MS's certificate (using *PKC_Verify*). This process is usually optional for web browsers. The client also initiates a key exchange process (*PKC_KeyEx*) to exchange a pre-master secret. During this process, the MS sends an encrypted key in the *ClientKeyExchange* message to the server so that the server can generate the same pre-master secret by decrypting it using its private key. Additionally, the MS needs to authenticate itself to the server by showing possession of its private key. The MS produces a message digest of previously exchanged messages and signs the digest with its private key (using *PKC_Sign*), and sends the signed digest within the *CertVerify* message. Using the MS's public key obtained from the MS's certificate, the server verifies the received digest (using *PKC_Verify*). If it is valid, the MS truly possesses the private key that pairs with the public key sent to the server.

At this point, the MS and the server are authenticated to each other, and they already share a master key derived from the key exchange process. To complete the Handshake protocol, the MS and the server send the encrypted message digest of all previously exchanged messages to ensure the integrity of the messages and to show possession of the shared master key. Note that the three cryptographic processes, *PKC_Verify*, *PKC_KeyEx*, *PKC_Sign*, are different depending on which PKC algorithm (RSA or ECC) is used.

Fundamentally, at the MS, the Handshake protocol is composed of three processes: server authentication, key exchange, and MS authentication. In the server authentication process, a MS uses a digital signature verification (*PKC_Verify*) process to validate a server's certificate. This process uses either RSA or ECDSA verification algorithm but not both. The key exchange (*PKC_KeyEx*) process establishes a shared key which uses either RSA encryption/decryption or ECDH. The MS authentication (*PKC_Sign*) process enables an MS to prove its identity by digitally signing a message and sending it to a server. This process uses either RSA or ECDSA digital signing algorithm.

2. The Variation of Handshake Protocol

From our experiments (described in Chapter III), we see that ECDSA has advantages over the RSA signature algorithm in terms of energy consumption. However, signature verification using ECDSA consumes more energy than the RSA signature verification algorithm. RSA verification, like encryption is quite cheap in terms of energy consumption. In the key exchange process, the RSA encryption algorithm consumes much less energy than the ECDH algorithm to exchange a 48-byte pre-master secret. This asymmetrical nature of energy consumption of the RSA and ECC algorithms is exploited in our proposed variation of the Handshake protocol.

The goal of the proposed variation of the Handshake protocol is to reduce energy consumption at only the MS side where the mobile device is energy constrained. The variation introduces additional computational load at the fixed server-side device, but this would be tolerable at the server. In this variation, the MS needs to hold an ECC-based certificate and the server needs to hold an RSA-based certificate. The variation also requires more memory space to store both RSA and ECC modules than the traditional TLS protocol. However, it would not be a constraint since the cost of memory is smaller than the cost of battery.

The variation is that, in Figure V.1, the MS uses the RSA verification algorithm to validate the server's RSA certificate in the PKC_Verify process. Upon the certificate request from the server, the MS sends its ECC certificate and not an RSA certificate. Then, in the PKC_KeyEx process, the MS generates and encrypts a pre-master secret using RSA encryption algorithm along with the server's public key (obtained from the server's RSA certificate). To prove its identity in the PKC_Sign process, the MS signs all previously exchanged messages using ECDSA. The security strength of the proposed protocol is the same as that of the ECC and RSA algorithms if the proper key sizes are selected (160 bit keys with ECC are equivalent to 1024 bit keys in RSA).

3. Results

Table V.2 shows the total average energy consumption on the client wireless device using different Handshake protocols and different key sizes. These results are from the comparison of RSA, ECC with the random binary curve (RBC), and the proposed variation of the Handshake protocol. From the table, we see that by using the variation of the protocol, we can save energy up to 90% compared to 4096-bit RSA or up to 70% compared to the 283-bit ECC Handshake protocol. Note that the energy savings is small for smaller key sizes.

Table V.2: The Average Energy Consumption of a TLS Handshake Session

ECC Key Size	Energy Consumption (mJ)			Percent Saving Compared to	
(RSA Key Size)	RSA	ECC RBC	Proposed	RSA	ECC RBC
163 (1024)	5.07	13.02	3.82	24.79%	70.70%
233 (2048)	28.51	25.86	8.14	71.46%	68.54%
283 (4096)	186.05	50.13	18.27	90.18%	63.55%

C. AN ENERGY EFFICIENT STANDARD SECURITY PROTOCOL FOR WLANS

From our framework described in Chapter V.A, we offer a method of modification of protocol primitives of a standard security protocol to be more energy efficient. In this section, we propose a modification of a security protocol in WLANs for encryption and message authentication services. The standard WLAN security protocol utilizes either RC4 or AES as an underlying cipher. From our previous work in Section III.B.1, RC4 and AES have different tradeoffs, and they may be used together to provide more energy saving. The goal of this work is to use both AES and RC4 to provide more energy efficient encryption protocol. This work is extended from our work in [86].

1. WLAN Standard Security Protocols

Currently, security protocols for WLANs, i.e., WEP, WPA, and WPA2 have been proposed to provide security services such as encryption, authentication and data integrity. WEP has been shown to be failed to provide such services. WPA and WPA2 have been proposed to replace WEP. WPA utilizes RC4 cipher as a core cryptographic engine to provide the services while WPA2 employs AES cipher. Both AES and RC4 have different pros and cons as described in Section II.A.

From our study, we have shown that not only RC4 and AES have different pros and cons, but also have different characteristics of energy consumption in data communication networks such as WLANs. We know that for a small packet size, AES consumes less energy than RC4. However, RC4 is more energy efficient for a large packet size. Based on this simple fact, we propose an security protocol for WLANs based on the selection of AES or RC4 as a cipher for packet encryption. The selection is based only on the size of the packet to be encrypted. The result will yield more energy saving than than the standard protocols.

2. The Proposed WLAN Security Protocol

We propose (see Figure V.2) an energy efficient security protocol for WLANs with a simple modification. We use different cryptographic primitives to provide security to packets based on their size and type. We show the results of how much energy can be saved by using such a security protocol in WLANs.

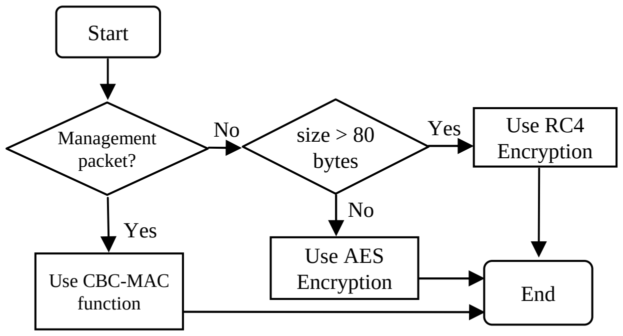


Figure V.2: A Simple Energy Efficient Adaptive Security Protocol

We use CBC-MAC [28] to provide 128-bit message authentication to management packets such as an 802.11 ACKs (14 bytes long), beacon packets (40 - 546 bytes long), and other short 802.11 management packets. The assumption here is that we need to provide only message authentication to the control and management packets to prevent packet modification. Additionally, the contents of these packets are not private; hence, encryption is unnecessary unless complete privacy is required (and perhaps to thwart traffic analysis). For data packets, we use RC4 or AES to provide confidentiality. To be more energy efficient, we propose to use RC4 only with the packets whose sizes are more than 80 bytes; otherwise AES is used. The packet size of 80 bytes is determined based on our study in Section III.B.1 where we saw that the AES performs better than RC4 with the packet size of 80 bytes or more. However, there are security implications beyond the simple mechanisms used here. For instance, if both RC4 and AES use the same key, if RC4 was broken and the key compromised, AES would also be broken. In this study, we assume both AES and RC4 use two different keys. From the results of the security performance in previous sections, we can compute the energy consumption of RC4, AES, and CBC-MAC as a function of packet sizes.

3. Experimental Design

In our study, we measure the energy consumption of cryptographic protocols in an emulated environment. To create the emulated environment, we used a sniffer to capture packets in two different network topologies, home and campus networks. Based on the collected packets, we created an empirical distribution of the packet sizes in these networks. The home network topology consisted of one access point and two mobile stations. The campus topology consisted of 10 access points in an eight-story building and many mobile stations. The packet size distributions for the two networks are shown in Figure V.3. The distributions of the packets are used to compute the mean energy consumed by a mobile station in running the security protocol. We can make the following observations from this figure. Most of the packets in either network have small sizes. These are the management frames of IEEE 802.11 (probes, associations, reassociations, dissociations, and beacons) or acknowledgment frames. As we will see below, this has an impact of how much energy is consumed in the security process.

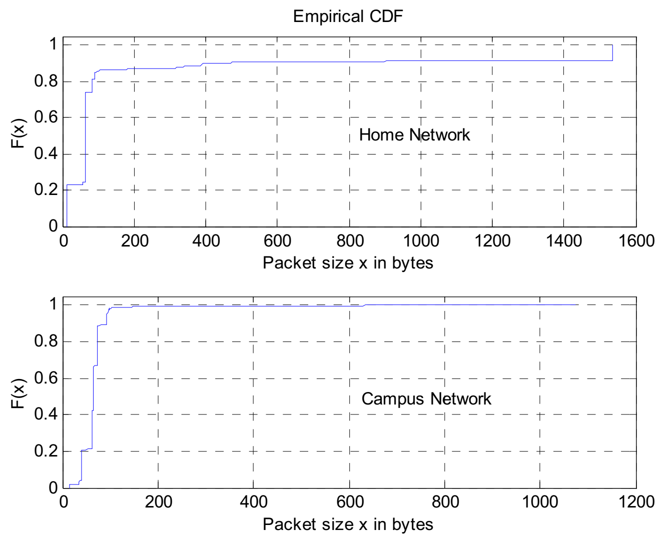


Figure V.3: Distribution of packet sizes in home and campus networks

Most encryption algorithms operate in the following manner. Initially, there is a process in which the key is manipulated in a certain way. In RC4, there is a key expansion process and an initial permutation process that requires several swaps of bytes between vectors. In AES, a key schedule has to be created [110]. In IEEE 802.11, each frame is self-synchronized. That is, each frame needs to be received independently of other frames. Consequently, a mobile station encrypts and decrypts each frame independently. The key manipulation processes constitute an overhead for encryption and can form the significant part of energy consumption while encrypting short packets. In [88] and in Chapter III, we showed that it is better to use RC4 for long packets and AES for shorter packets for this reason.

4. Results

For the scenario where we captured packets (10 MB in total for the home network and 4 MB in total for the campus network), we calculate the energy consumed for each packet based on its packet size and type. Table V.3 shows the results of energy consumption using a fixed algorithm (AES or RC4) and the proposed algorithm. For the fixed algorithm, only one selected encryption algorithm is used to encrypt all packets. We use a 128-bit keys for both RC4 and AES functions. The key size is selected as the minimum requirement for today's security [69]. From the table, we see that by using the proposed algorithm we can save about 0.01% and 57% of energy for the home network compared to using fixed RC4 and AES encryption algorithms, respectively. By using the proposed algorithm, we can also save about 0.03% and 5% for the campus network compared to using fixed RC4 and AES, respectively. The energy saving compared to using RC4 is not significant because most data packet sizes are long. The energy saving comparing to RC4 use will be improved when the average size of data packets is more than 80 bytes.

To provide stronger security by using AES and save energy, we could also fragment a long packet into smaller packets and use AES to encrypt them. Additionally, smaller packets are likely to be less susceptible to wireless channel errors, and hence, we could save much more energy. Although the fragmentation would give significant energy efficiency, it will also lower transmission throughput (thereby increasing the energy consumed) and increase

Table V.3: A Comparison of energy consumption using the fixed and proposed security protocols

Algorithm	Energy Consumption (mJ)			Percent Saving Compared to	
	RC4	AES	Proposed	RC4	AES
Home	99.7654	236.8924	99.7486	0.0168%	57.8928%
Campus	88.9532	93.8633	88.9279	0.0284%	5.2581%

latency. The tradeoffs in this approach needs a more detailed investigation.

D. LIMITATIONS OF THIS WORK

This work demonstrates the benefits of applying energy saving techniques in the design of security protocols for wireless networks. A major limitation of this work is that we do not tie it with the security requirements of the wireless network and the implications it may have on the security provided to a wireless network. For a given network or application we need a security policy - a statement of what is allowed and what is not allowed. The security services that implement the security policy can then be defined (e.g., confidentiality for data packets, only authentication for probes, ACKs and management packets). Then the way these security services use security mechanisms and primitives adaptively to conserve energy can be determined. Wherever possible, we have pointed out the pitfalls of blindly adopting the approaches presented in the chapter. In the case of the variation of the handshake protocol for SSL, there needs to be a new protocol version that would support mixed RSA/ECC algorithms. In the home WLAN example, we have ignored the way the communicating parties would identify the encryption scheme that is used with a particular packet. We have also ignored how different keys would be used with different encryption schemes, how they are exchanged and so on. The assumption here is that all MSs share the same secret with an AP and that secret is used for encryption and message authentication. Instead of using the shared secret, there should be a key management that generates a pairwise key from the shared secret for each station, and it should be subject to be renewed after an interval. Finally, we have not investigated in detail, the impact of the radio channel nor have we designed protocols that can exploit knowledge of good and bad states of the channel to save on the energy consumed.

E. CHAPTER SUMMARY

This chapter describes a framework for designing energy efficient security protocols. We use our study in Chapter III to suggest three approaches for designing efficient security protocols. We apply two of these approaches in simple examples to demonstrate the potential for saving energy.

From the methodologies, one can modify the existing standard security protocols to be energy efficient. It has been shown that the saving can be significant. The replacement of some RSA with ECC algorithms in the TLS Handshake protocol can typically save about 25% to 70% compared to plain RSA and ECC, respectively. For the protocol modification in WLANs, the saving ranges approximately 0.01% and 57% compared to using plain RC4 and AES ciphers. However, these savings are limited due to the existing structure of the standard security protocol design. The re-design can only improve the savings to some extent. For example, we can only change the underlying ciphers or algorithms in order to make the standard protocol more energy efficient. We cannot specify a security level for the protocol.

Due the limitations, we need a security system that is more flexible in which we can finely tune our security needs to a just enough level and save even more energy. One can say that the level of the energy saving depends on the level of protocol tuning. A more fine protocol tuning can save more energy. In the next chapter, we propose a tunable security system to support a fine-grained tuning mechanism for more energy saving.

VI. TUNESEC: AN ENERGY EFFICIENT SECURITY PROTOCOL FOR WLANS

From the proposed methodologies in Chapter V to save energy due to security services for resource-limited mobile devices, we showed that Methods 1 and 2 can reduce energy consumption while trying to comply with existing standards. In Methods 1 & 2, we utilized different security algorithms in a standard protocol to reduce its energy consumption. In Section V.B, we use both RSA and ECC public key algorithm to provide authentication and key establishment service for the WTLS protocol while the standard one utilizes one or the other. The use of RSA and ECC for different purposes during the protocol transaction yields energy savings of approximately 20% to 70% for authentication and key establishment. In Section V.C, we use AES and RC4 differently to provide encryption service based on packet size while the standard WLAN utilizes either one. The result is that we can save approximately from 0.01% to 57% of energy consumption of data encryption.

These two methodologies have shown that security algorithms can be combined differently to reduce the energy consumption while providing a fixed security level to provide the same security strength. These methods tend to be compatible with standard protocols which support security services at a fixed level.

We however believe that additional energy savings can be achieved if we can provide a fine-grained method for security provision to adjust not only the combination of the algorithms, but also the security level needed for individual messages. Based on Method 3, the greenfield approach, we propose an example security system to provide energy-efficient security in which the security level of security services can be tunable. Our example security system is called Tunable Security or *TuneSec*. This is NOT an optimally energy efficient security protocol, but it provides an example of how one can design a security protocol keeping energy consumption in mind.

TuneSec is divided into two parts to provide packet-level security and session-level security. The packet-level security service comprises of encryption and message authentication and the session-level security service comprises of authentication and key establishment (AKE). Thus, we propose two different strategies to provide tunability to packet-level and session-level security.

In this section, we will describe a high-level TuneSec architecture in Section VI.A. We describe the TuneSec protocol for packet-level security in Section VI.B and for session-level security in Section VI.C. Then, we apply the TuneSec framework to IEEE 802.11 WLANs to provide energy efficient security services with the packet-level and session-level security, and show performance results in Section VI.D. We conclude our work in Section VI.E.

A. TUNESEC ARCHITECTURE

The goal of the TuneSec is to be a tunable, simple, high performance and secure system that offers energy efficient security services that depend on available resources and changes in the dynamic environment in wireless networks. The scenario we consider is that of a mobile terminal as a client connecting to an access point to access a network and to request and utilize services in a secure and energy-efficient manner.

First, the access point initiates negotiation of a security association (SA) for a client based on the available resources through a control channel. The control channel is also protected using confidentiality and authentication techniques. The negotiated security association should comply with the security policy established by the client or a system administrator. The security association describes which security primitives are used and at what level. Once the SA is establish, network applications can use the security services which can be dynamically switched to fit the client's available resource and security policy. Then, the data from network applications are securely transmitted through a data channel.

The architecture of TuneSec system is shown in Figure VI.1. It is composed of five components: (i) Security Policy and Association Manager (SPAM), (ii) Authentication and

Key Manager (AKM), (iii) Resource Manager, (iv) Tuner, and (v) Security Service Manager (SSM).

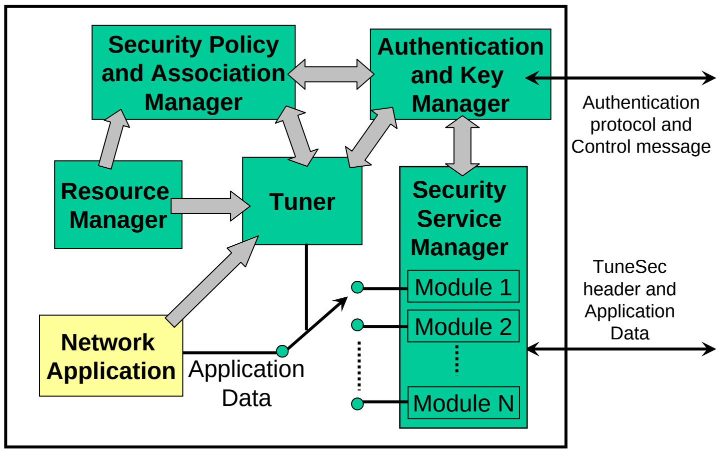

Security Policy and Association Manager
Authentication and Key Manager
Authentication protocol and Control message
Resource Manager
Tuner
Security Service Manager
Module 1
Module 2
Module N
TuneSec header and Application Data
Network Application
Application Data

Figure VI.1: The TuneSec Architecture

The main purpose of the Security Policy and Association Manager (SPAM) is to maintain security policies and SAs, and to solve conflict between user and system policies. The policy may be active in which it can be temporally changed according to available resources and environments. Examples of the active policy scheme are Role-based Access Control (RBAC) with an extension to active security [27] and Generalized Temporal RBAC (GTRBAC) [60]. In this work, we assume no security policy is active and does not pose any conflict.

The main purpose of the Authentication and Key Manager (AKM) is for authentication and session key establishment. The AKM may also be used to negotiate a security association and to maintain the session key. It is also required to create and maintain a suite of SAs corresponding to a security module for a given connection. In some case, control messages are sent from the receiver to the sender through the AKM component to update or re-negotiation the SA. An example of the AKM component is the IKE module of IPsec [39]. The AKM is involved only in the session-level security services. Before a session is started, AKM gathers necessary information and establishes security modules needed for the session, which are used by the Tuner to provide packet-level security services.

The resource manager reports the availability of resources to other components. Such resources are remaining battery level, the CPU load information, the transmission rate/ channel quality, possible location of the hosting device, and maybe alarms from the host or the network system. The location of the device is possibly used for increasing or decreasing the level of security when the device roams across different kinds of wireless networks. After it gathers needed information, the resource manager summarizes and reports to Tuner for making a decision.

The Tuner is responsible for making decisions about the security module used for each application data set. In making the decision, the Tuner relies on the information about system, network and application environments from other components such as the resource manager, the SPAM, and the AKM. Based on the set of application data, the Tuner moves the "switch" between different security modules to provide different security services. The Tuner is only involved in the packet-level security services.

The main purpose of the security service manager (SSM) is to load and unload a security module according to the SA suite provided by the AKM. The module is responsible to perform packet-level security services such as encryption/decryption and message authentication code. It also reports any error or suspicious packet to AKM for further adjustment or making a session.

In a system that supports the TuneSec system, the network application may need to communicate with the Tuner for its specific security and resource policy for minimum security level for its connection. The application may also provide information about how important its data are and hence, security level may need to be adjusted according to the policy.

B. PACKET-LEVEL TUNESEC

The goal of packet-level TuneSec is to make the given security protocol be tunable to a sufficient level of security at the packet level. To be able to determine the security level, we need to identify "sufficient security" in a quantitative manner that can be converted to parameters that are understandable to cipher algorithms. Upon having the parameters, we need to apply them to an "adjustable security-level" cipher so that it can provide different security levels as needed, yet saving energy. In this section, we provide a mechanism to interpret an abstract security level to some quantitative parameters, which are later applied to existing ciphers such AES and RC5 to provide different security levels.

1. Flow Chart of TuneSec Mechanism

Figure VI.2 shows the flow chart of TuneSec mechanism. At the first step, a user may initiate an information exchange session and asks for a specific security requirements. Then, interpretation of these security requirements is needed to translate the requirements into a set of quantitative parameters. Then, we use the parameters to create security modules and to provide different levels of security. If there is a need to adjust the security services, such as when the security need is changed or when the remaining energy budget is low, the parameter interpretation is required again to create a new cipher module and to destroy the unused one. The TuneSec is stopped when the user's session is terminated.

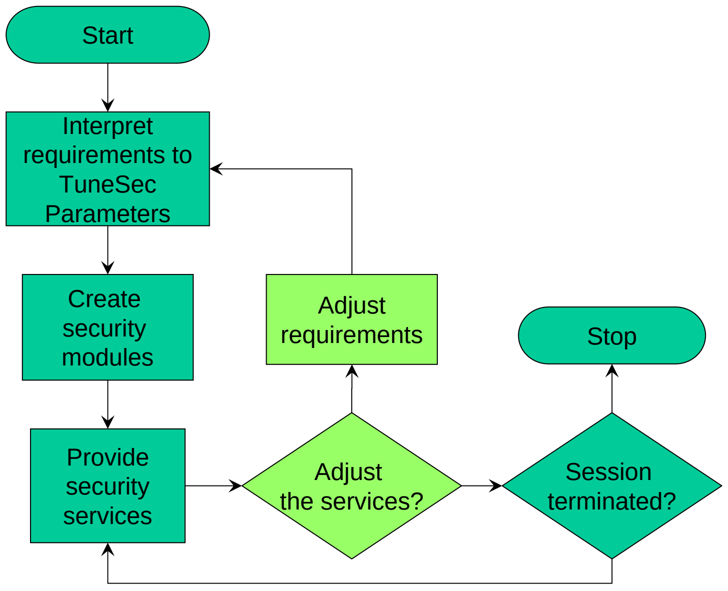


Figure VI.2: The TuneSec Mechanism Flow Chart

2. Interpretation of Security Level

Generally, the term “security level” seems to be very abstract in which it can only be qualitatively determined. Other previous works have proposed the way to qualitatively indicate the security level. In [57], the security level is simply classified as low, medium, and high. For each level, the security is numerically assigned based on fixed key size and integrity rate for security. For example, low security means using 56-bit key and providing message integrity to 60% of all packets. Similarly, in [112], the security level is known as a class, each of which uses different algorithms and protocols. The qualitative classification of the security level seems to be vague. For example, The 56-bit key encryption provides “low” security level. The security class does not give a clear meaning of security level. Since the low security can be interpreted in many ways with respect to individuals’ judgment.

In fact, we cannot provide security based on an abstract level, and we need a common, understandable, and quantitative way to specify the security level. We can quantitatively provide security based on the cost of data we want to protect. Since it is impossible to provide “unbreakable” security, we often provide the security such that the cost of breaking it is more than the benefit gained from breaking it or the value of the data being broken.

The value of data is naturally time-sensitive; for example, data we want to protect may be worth a lot today, but may be worthless after 20 or 100 years. Thus, we propose to provide the security level based on the time we need to protect any information or data.

Additionally, the security level is dependent on the packet type in wireless networks. In IEEE 802.11 WLANs, there are 22 types of packets [13]. Each type definitely requires a different security level. For example, we may only want to protect any Beacon packet which contains network parameters such as SSID (Service Set Identifier), supported rates, CF (contention free) period, and etc. for a few years, and to protect “Data” packets for more years. Table VI.1 shows some examples of the number of years that may be required for 802.11 WLAN packets for a low security level.

Table VI.1: An Example of A Low Security Level for 802.11 Packets

Packet	Years	Description
Assoc Req	5	A request for association to an access point
Assoc Resp	5	A response to a client for association
Beacon	2	A packet to give network information
Authen	10	A request for authentication to an access point
Data	25	A payload carrying client's data

3. Calculating TuneSec Parameters

In the previous section, we proposed to use the number of years as a security level to provide security services to data. Then, we need to interpret the number of years to a security level. We propose to employ not only the key size, but also the number of operational rounds as the security level as described in Chapter IV.

Commonly, the security level is only defined in term of the key size. The longer the key size, the higher the security level. However, this assumes that the number of operational rounds is fixed at a level that the cryptanalysis attack is hard or completely impossible. Using only the key size, we may not be able to provide a "true" different security level, which in fact also relies on the number of operational rounds of a cipher. Therefore, we propose to utilize both the key size and the number of operational rounds as parameters for adjusting the security level. We have shown in Chapter IV that increasing the number of rounds provides a higher cipher robustness; thus, yielding a higher security level against cryptanalysis attacks. For easy understanding, we call both key size and the number of operational rounds as *TuneSec Parameters*.

Figure VI.3 shows how we calculate the TuneSec Parameters. From the number of years and data needed for protection, we calculate the needed key size using the security model proposed by Lenstra and Verheul [69]. The needed key size (KS) is calculated using the following formula:

$$\mathrm{KS} = 56 + (y + y' - 1982) \times (12/m + 1/b)$$

where y is years needed for security, y' is the current year ($y' = 2005$). Here m is the average number of months that the CPU speed and available memory is doubled. According to Moore's law, we define $m = 18$. The b is the number of years that the available budget for attacking is doubled, and we define $b = 10$ as described in detail in [69]. This formula is derived from incidents of breaking DES with a 56-bit key that it could happen, in theory, in 1982 although it was first broken in 4 months in 1997 by Rocke Verser and in 56 hours in 1998 by Electronic Frontier Foundation [49]. However, it is believed that DES could be trusted to secure information only until 1982.

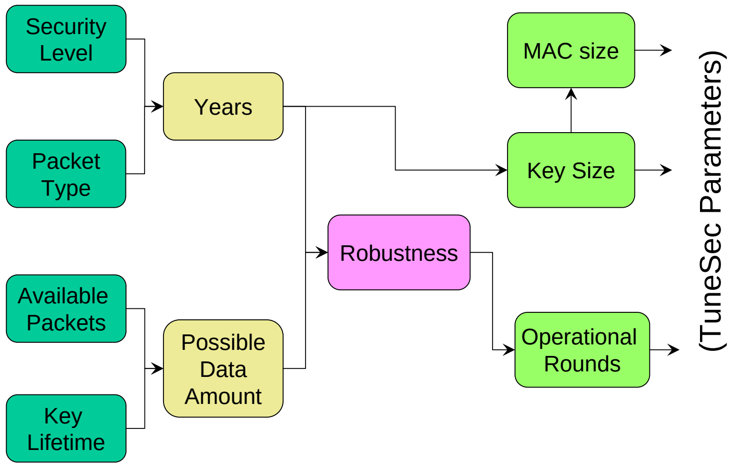


Figure VI.3: The TuneSec Parameter Space

After having the needed key size for a cipher, it is easy to compute the MAC size. The needed MAC size can be derived using the Birthday Paradox attack where only $2^{n/2}$ operations are needed to break the MAC, where n is the MAC size in bits. Note that we require 2^n operations for key searching attack, where n is the number of bits of the key. Therefore, to provide an equivalent security level of MAC to the cipher, the MAC size should be twice as long as the cipher key size.

To calculate the number of operational rounds, first we need to compute the security robustness which is the multiplication of the number of operations and the amount of data required for breaking. Based on the number of years, we can calculate how much computing power is available in terms of MMYs. As derived in Lenstra and Verheul [69], the available

MMY can be computed as:

$$\text{AMMY} = 0.5 \times 2^{(y'+y-1982)(12/m+1/b)}$$

where AMMY is the available MMY that is possible to have in year $y' + y - 1982$. The calculation is based on the MMY margin (0.5 MMY in this case) which is possibly available in year 1982, which can be used to break DES [69].

Then, we need to compute the possible amount of data available for cryptanalysis attack. The amount of data depends on the available packet rate and the key lifetime. The key lifetime says how long a session key will be used to provide security services. The key lifetime can be specified using the session-level TuneSec. A long key lifetime makes a security system more vulnerable.

Based on the key lifetime, which is the time a key will be used to provide security services, we estimate the available data as the multiplication of the packet rate and the key lifetime. Using the possible amount of data available and the MMY as the number of possible operations, we calculate the robustness as their multiplication. From the robustness, we derive the number of operational rounds using the robustness model that we have in Section IV.B.3.

At this point, we have the needed key size and MAC size (derived from the key size) as well as the number of operational rounds. These parameters, the TuneSec Parameters, will later be used by a cipher to provide just needed security level.

4. Creating Security-Level Adjustable Modules

Once we have the TuneSec parameters, TuneSec creates one or more security modules to provide security services. The security module is an adjustable cipher such as AES or RC5 or a MAC algorithm such as HMAC. There is no limitation on the number of security modules created. For example, we can create different modules of AES, or one AES module that can provide different security levels. Additionally, we may use both AES and RC5 modules. However, the number of cipher modules can be limited if devices have limited memory resources, or it is defined by users or applications. However, we believe that the memory resources are not as limited as the available energy for small devices as the memory capacity could be doubled approximately every 18 months according to advanced memory chip technologies and Moore's Law [58, 83].

We choose AES and RC5 as our adjustable ciphers because of two reasons. First, AES and RC5 provide flexibility in changing the parameters such as key size, rounds, and block size. Second, they are known to be efficient in term of computation and energy efficiency. The current standard cipher such as AES does not provide the adjustable ability. However, it is not hard to modify the AES to be adjustable and also to provide a security level as needed. Like other block ciphers, AES employs several operational rounds to manipulate input data into output data. As the input goes through more rounds, it is harder to trace back from the output to input; hence, providing a strong encryption.

In the standard, AES uses 10, 12, and 14 rounds for keys of 128 bits, 192-bits, and 256 bits. The increased number of rounds provides strength against cryptanalysis attacks while the increased key size provides strength against brute-force key attacks. As intended by the Rijndael creators, AES can be very flexible. The key size can be increased from 128 bits to a multiple of 64 bits such as 192 and 256 bits without increasing the number of rounds. In contrast, we can increase the number of rounds without increasing the key size. To use AES as an adjustable cipher, we only need to provide different key schedules for each round. The key scheduling is a process that translates a user key (a fixed-size secret key) into a longer key, and part of the long key is used for each round. Thus, for more rounds, we need a longer key schedule.

RC5 is also designed to be flexible. It has a variable number of operational rounds from 0 to 255, key sizes from 0 to 2040 bits, and block size of 32, 64, or 128 bits. The suggested parameter set is 12 rounds, 128-bit key, and 64-bit block size. However, 12-round RC5 was discovered to be weak against an improved cryptanalysis attack in early 1998 [30]. Thus, RC5 is suggested to be operated at 16 rounds to be completely secure [61]. However, we can utilize the lower number of rounds as we need a lower security level.

We choose CBC-MAC to provide 128-bit MAC outputs and HMAC with SHA (HMAC-SHA) to provide MAC outputs of 160, 256, 384, and 512 bits. The CBC-MAC algorithm generates a MAC output using a block cipher. In our study, we choose to use 128-bit AES with the CBC-MAC which generates an 128-bit output. To generate longer outputs, we use HMAC with SHA algorithm. HMAC can be used with any hash function such as MD5 or SHA. The MD5 algorithm, known to be more efficient than SHA, but it is not as secure as SHA [44]. A recent attack on MD5 claims to find a collision of MD5 outputs within an hour [116] while such collision can happen in 15 days using the birthday paradox attack. A real implementation of the shortcut attack shows that the collision can be found in 8 hours using a 1.6 GHz Intel Pentium PC [64]. The SHA algorithm is also known to have weakness, but the weakness is not as severe as that in MD5. A recent attack on SHA-1 provides a shortcut attack that needs a number of operations less than that of the birthday paradox attack. Compared to the number of operations needed for the birthday paradox attack, 2^{80}, the shortcut attack requires only 2^{69} operations [117]. Despite this attack, we choose SHA because SHA is standardized by NIST, and it is widely used. However, a recent debate suggests that the use of MD5 and SHA with HMAC is secure in spite of the weakness [65]. Second, SHA can produce variable output sizes of 160, 256, 384, and 512 bits.

C. SESSION-LEVEL TUNESEC

Session-level security services often serve two purposes, for authentication and for session key agreement. The session-level security services are required before a packet-level service can begin. Before a session is started, a user is needed to be authenticated before being authorized to access network resources. This step is performed by the Authentication and Key Manager as shown in the TuneSec architecture. Once a user is authenticated, the user and a network server need to agree on a session key used to provide packet-level security services for this session.

Due to the very specific characteristics of a wireless network, the Authentication & Key Agreement (AKA) protocol is needed to be specifically designed to fit those characteristics. For example, the AKA protocol for WLANs is totally different from that for GSM networks. Due to their specific design, it is difficult to generalize the TuneSec concept for any AKA protocol for all kinds of wireless networks. Therefore, in this study, we only propose a new

AKA protocol based on TuneSec framework for only IEEE 802.11 WLANs to be tunable and hence energy efficient.

1. The Problem with the IEEE 802.11 AKA Protocol

As we described in Section II.C.4, the IEEE 802.11 AKA protocol is composed of three phrases, the Discovery and Association, the Authentication, and the 4-way Handshake protocol. At first, a mobile station (MS) discovers an access point (AP) that has capabilities to support the MS's security requirements. Then, the MS and the AP authenticate each other, and the MS is granted an access to the network if the authentication is successful. Before accessing the network, the MS and AP needs to agree on a master key, called Pairwise Master Key (PMK), which may be used to re-associate to the network without the authentication. From the PMK, the MS and the AP needs to agree on a temporary key or a session key, called Pairwise Transient Key (PTK), using the 4-way handshake protocol. The PTK key is subject to be used for a period of time until the MS disassociates from the network.

The problem with the IEEE 802.11 AKA protocol is that it does require the 4-way handshake protocol when the MS wants to re-associate to generate a new PTK. Additionally, the handshake protocol can be expensive for small limited-power devices. Therefore, we propose to use one-time pad keys or one-time passwords for lightweight re-association to replace the expensive 4-way handshake protocol. In our scheme, a series or a chain of one-time passwords is recursively generated using a hash function to hash from one root key to several child keys [68]. For WLANs, the root key is the PMK, and the child keys are the PTKs.

The advantages of the proposed scheme are two-fold. First, from one PMK we generate a series of PTKs using the hash chain. The length of the chain can be varied based on the security level. A long PTK chain may be vulnerable to key collision; however, it can save energy from performing a full association and the 4-way handshake. Second, by using the hash chain, each PTK generated can be used as an authentication token, and MS does not require a re-authentication process as long as the generated PTKs are not used up, yet saving even more energy.

The security level here is the inverse of the length of the PTK chain. A high security level requires the chain length to be short to prevent from the key collision, which must never happen in the one-time password scheme. In the following sections, we explain our proposed scheme in more details and we show the performance comparison of the standard AKA scheme to the proposed one.

2. One-time Secret Key Authentication Protocol

In this section, we propose a new authentication protocol that does not require the 4-way handshake repeatedly, and it is able to provide a fresh key. It also provides a lightweight (re)authentication to save energy for small wireless devices. The proposed protocol is based on the concept of the one-time password scheme such as S/Key [53], and we call it One-time Secret Key Authentication (OSKA) protocol.

After the first full AKA, the MS and AP possess a PMK. Then, the 4-way handshake protocol is used to exchange nonces and to generate a *root key*, RK, from the PMK as shown in equation VI.1 where the PRF (pseudorandom function) could be HMAC-SHA [66]. Then, we use the RK to generate a sequence of one-time keys, K_i where $i = n-1$ to 0 and n is the length of the sequence, as shown in equation VI.2. The whole sequence needs not be stored at the MS and AP if we cache the RK since we can always generate it from the cached RK.

$$RK = PRF(PMK, \text{ANonce}||\text{SNonce}||\text{APMAC}||\text{MSMAC}) \tag{VI.1}$$

$$K_{i-1} = f(K_i), \quad i = n-1 \text{ to } 0 \tag{VI.2}$$

For example, we generate $K_{n-1} = f(K_n)$ where K_n is the RK, and $f(.)$ is a secure hash function such as SHA [81]. Each K_i is used to generate PTK_i for each re-association as described later. The summary of the key derivation is shown in Figure VI.4. We may provide up to 256 keys in a sequence if we define an 8-bit parameter for n. The length of the sequence can be adjusted based on the tradeoff between energy and security which is studied in the latter section. A shorter sequence yields better security, but more energy is consumed because a full 4-way handshake protocol is required to generate a new RK for a new sequence. In contrast, a long sequence is probably susceptible to collisions (there may be two similar keys in the sequence). However, using a secure hash such as SHA-512 which produces an output of 512 bits, a long collision-free sequence can be obtained.

$$\begin{tikzcd}
PMK \arrow[r, "PRF(\,)"] & RK \arrow[d, "f(\,)"'] & \\
 & K_{n-1} \arrow[r, "PRF(\,)"] \arrow[d, dashed, "f(\,)"'] & PTK_{n-1} \\
 & K_1 \arrow[r, "PRF(\,)"] \arrow[d, "f(\,)"'] & PTK_1 \\
 & K_0 \arrow[r, "PRF(\,)"] & PTK_0
\end{tikzcd}$$

Figure VI.4: The Key Derivation for OSKA

a. The OSKA protocol After a full authentication and the key sequence derivation, the MS and AP start using the first key, K_0, to generate a pairwise transient key, PTK_0, of length m, using the HMAC-SHA algorithm [66]. The PTK is composed of keys for encryption and for message authentication, and it is generated as shown in Figure VI.5.

```
$PTK_i \leftarrow$ NULL
  For $j \leftarrow$ 0 to ($m$/128)
    $PTK_i \leftarrow PTK_i$ || HMAC-SHA-1 (**PMK, $K_i$** ||
     ANonce || SNonce || AP MAC || MS MAC || $j$)
```

Figure VI.5: The PTK derivation

PTK_0 is used until the MS disassociates from the AP, or there is a need for a key change. The MS requests a new key by simply re-associating with the AP again, and in the re-associate message, the MS attaches the K_0, which is no longer used. The attachment of K_0

serves two purposes. First, it is used for MS authentication. By verifying that $K_0 = f(K_1)$, AP knows that the MS is legitimate because only legitimate MS knows the key sequence. The K_0 needs not be encrypted since it is no longer a secret after once used. Additionally, from the PTK derivation, an adversary who knows K_0 cannot generate PTK_0 without the secret PMK; hence, it provides forward secrecy.

Second, it is used to prevent man-in-the-middle attacks by providing mutual authentication. As shown in Figure VI.6, for $i = 0$, after receiving K_0, the AP authenticates the MS. Then the AP replies whether it accepts the re-association. If not, the full authentication is performed. Otherwise, the AP generates a MIC using HMAC-SHA(K_1,K_0), and replies to MS for AP authentication. The MS verifies the MIC using K_1. After authentication, both MS and AP discard K_0, update the currently used key to be K_1, and derive PTK_1. As the re-association number grows, the one-time key in the sequence may be used up. Then, a full authentication is required.

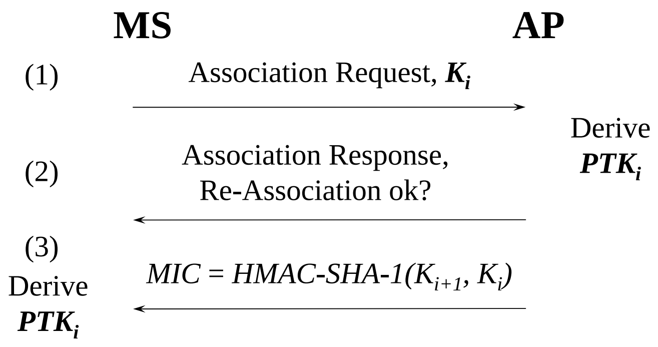

MS
AP
(1)
Association Request, K_i
Derive PTK_i
(2)
Association Response, Re-Association ok?
(3)
Derive PTK_i
$MIC = HMAC\text{-}SHA\text{-}1(K_{i+1}, K_i)$

Figure VI.6: The OSKA Protocol

b. OSKA Security analysis We analyze OSKA security protocol based on security reasoning and argument. Despite a formal verification, the reasoning and argument are a practical and common method of analyzing a security protocol based on past incidents of discovered flaws in security protocol design [24, 17].

Key Recovery:

The security of the OSKA depends on the strength of the hash function, $f(.)$, such as SHA which is standardized by NIST. SHA is known to be (so far) a strong hash function while MD5, another popular hash function, is known to have collisions for some inputs [116]. From the properties of the SHA, it is computationally impossible to reverse the key sequence. For example, an adversary cannot derive K_j though he knows $K_{j-1}, K_{j-2}, \ldots, K_0$.

Key Reusing or Key Replay Attack:
An adversary can replay the key if a key sequence is not freshly generated. In the OSKA protocol, we generate a key sequence based on new nonces from MS (SNonce) and AP (ANonce), and this guarantees the freshness.

Man-in-the-Middle attack:
The attack can be prevented by mutual authentication. To authenticate a MS, an AP needs to prove that $K_i = f(K_{i+1})$ where K_i is attached in the (Re)Association message. To authenticate the AP, the MS needs to verify that the MIC in Message (3) in Figure VI.6 is valid.

Birthday paradox attack:
A hash function is often subject to a birthday paradox attack [75]. To use a hash function to generate a key which provides a resistance of the birthday paradox attack, the hash output should be twice as long as the key length. For example, to generate a 128-bit K_i, we use SHA-256 which produces a 256-bit output. Then, we can truncate it to use only 128 bits. We use SHA-384 and SHA-512 to generate a 192-bit and 256-bit key, respectively. The use of SHA algorithm is strictly deployed as specified in the FIPS 180-1 standard [81].

3. OSKA Performance Results

In this section, we show the performance of OSKA compared to the standard protocol. We use the cycle counting method as described in Section III.A to measure energy consumption used by any cryptographic process. We calculate the transmission/reception energy based on the amount of data exchanged during a protocol transaction. By using the linear energy model for WaveLAN wireless interface [45], we can approximate the energy consumption of transmission and reception. It requires (431 μJ + 0.48 μJ/bytes) for point-to-point transmission of messages, and (316 μJ + 0.12 μJ/bytes) for receiving messages.

Figure VI.7 shows the total energy used for AKA protocols as the number of re-associations increases. The total energy includes energy from both transmission and cryptographic processes. Despite the existing standard re-association protocol, most commercial wireless devices do not support the re-association since it is not mandatory in the standard. The full association needs to start over from the association and discovery phase to the 4-way handshake phase, which consumes a great amount of energy. Using the standard re-association, the energy consumption is reduced. Compared to them, our proposed protocol, OSKA with a 256-bit key, performs much better in terms of total energy consumption since it reduces the energy from message handshaking.

Figure VI.8 shows the energy consumption from only the cryptographic process. It is shown that the OSKA outperforms the standard re-association when the number of re-associations exceeds about 30 re-associations. We do not show the performance of the full association since its energy consumption linearly grows at a much higher rate.

Figure VI.9 shows the energy consumption of 1000 re-associations as the length of the key chain increases. The increase of key chain means that we require less full associations to generate PMKs. For example, the key chain of 20 keys requires 50 out of 1000 full associations to generate 50 PMK keys, each of which is used to generate 20 PTKs.

For a small chain length, OSKA consumes high energy; however, as we increase the key chain length, the energy consumption is exponentially reduced. This is because we infrequently need a full associations, an expensive operation, to generate a new key chain as we set the key chain length to be long. The energy consumption of using standard re-associations is also reduced as the frequency of full associations decreases. However, its energy consumption is much higher compared to OSKA.

To reduce its energy consumption, OSKA uses a long key chain to avoid an expensive full association process. However, the long key chain may be subject to a collision in which two PTK keys are generated from one PMK key. Figure VI.10 shows the probability of

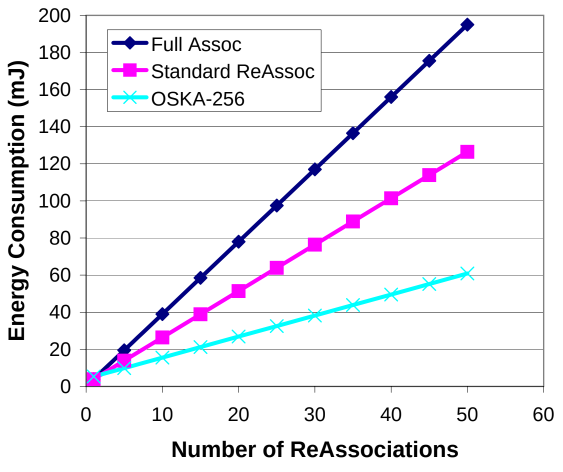


Figure VI.7: Energy Consumption of Transmissions and Cryptographic Functions

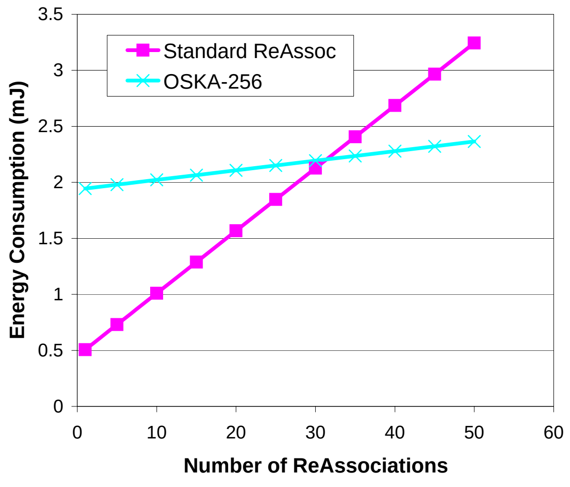


Figure VI.8: Energy Consumption of Only Cryptographic Functions

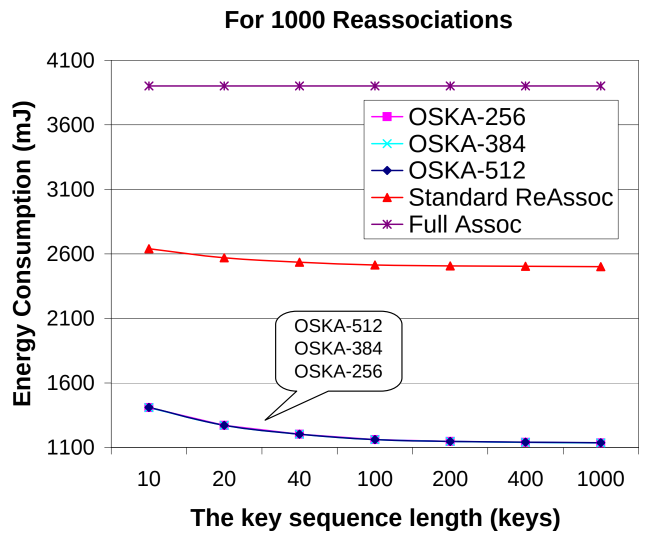


Figure VI.9: Energy Consumption and OSKA Key Length

key collision of OSKA key chain. As the length of the key chain increases, the collision probability is slightly increased. It shows that using a 512-bit hash function to generate the

OSKA key chain highly reduces the probability of collision.

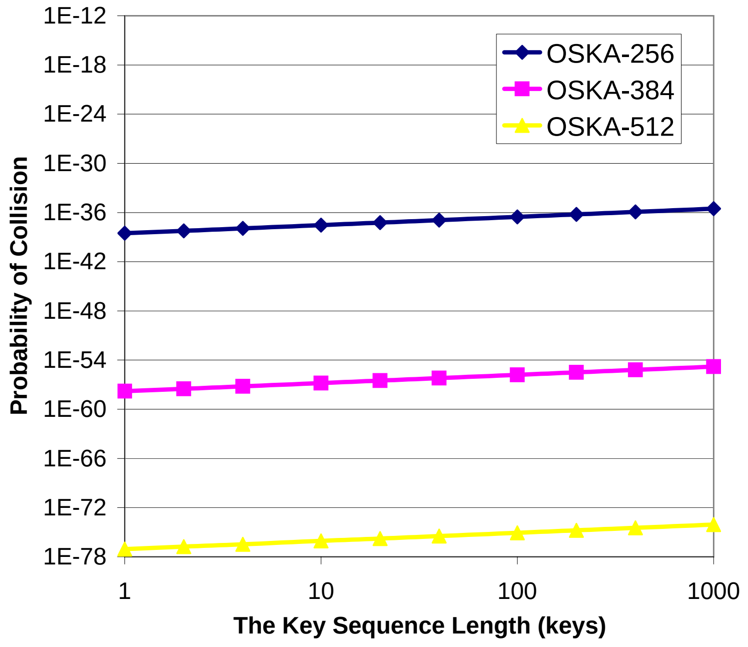


Figure VI.10: The Probability of Key Collision in a Key Sequence

D. EXPERIMENTS AND RESULTS

1. Pre-Defined Security Levels

In Chapter III, we have shown that the energy consumption of security services such as encryption and message authentication is different due to security properties such as key size, operational rounds, and ciphers as well as the data packet size. A set of properties can provide different security level and consume different energy level. To provide energy efficient security services, we need to determine a sufficient security level to provide security services to yield more energy savings. The goal of packet-level TuneSec here is to find a sufficient security level based on network packet types and to provide security services according to the security level.

In this section, we only consider IEEE 802.11 WLANs. In this study, we define 5 security schemes, each of which differently defines a security level for each packet type defined in IEEE 802.11 standard. Table VI.2 shows these five security schemes and levels in term of the number of years the data should be protected for each scheme for each packet type. In this table, we define three levels: low, medium and high, that provides different security levels for different packet types. These three schemes are supported by TuneSec system, in which a security level can be fine-tuned for each packet type. We also define two fixed-security schemes, fixed-low and fixed-high, in which all packet types have the same security level. From the years needed for data protection in the table, we interpret the number of years into the TuneSec parameters (rounds, key size, and MAC size) for ciphers and MAC functions to provide just enough security level.

Table VI.2: Years to protect data based on IEEE 802.11 packet types

		Years Protected					
Type	Packet	Services	High	Medium	Low	Fixed Low	Fixed High
Management	Assoc Req	(1)	20	10	5	25	100
	Assoc Resp	(1)	20	10	5	25	100
	ReAssoc Req	(1)	20	10	5	25	100
	ReAssoc Resp	(1)	20	10	5	25	100
	Probe Req	(1)	5	3	2	25	100
	Probe Resp	(1)	5	3	2	25	100
	Beacon	(1)	5	3	2	25	100
	ATIM	(1)	5	3	2	25	100
	DisAssoc	(1)	5	3	2	25	100
	Authen	(1)	40	20	10	25	100
	DeAuthen	(1)	5	3	2	25	100
Control	Action	(1)	5	3	2	25	100
	PS-poll	(1)	5	3	2	25	100
	RTS	(1)	5	3	2	25	100
	CTS	(1)	5	3	2	25	100
	Ack	(1)	5	3	2	25	100
	CF-End	(1)	5	3	2	25	100
	CF-End+Ack	(1)	5	3	2	25	100
Data	Data	(1,2)	100	50	25	25	100
	Data+CF-Ack	(1,2)	100	50	25	25	100
	Data+CF-Poll	(1,2)	100	50	25	25	100
	Data+CF-Ack/Poll	(1,2)	100	50	25	25	100
	Null	(1)	20	10	5	25	100
	Null+CF-Ack	(1)	20	10	5	25	100
	Null+CF-Poll	(1)	20	10	5	25	100
	Null+CF-Ack/Poll	(1)	20	10	5	25	100

(1) = Message Authentication services, (2) = Encryption service

We also provide different security services to different packet types. In the table, Service (1) means Message Authentication service, and Service (2) means Confidentiality or Encryption. For example, we may provide both encryption and message authentication

Table VI.3: The security level of session-level security protocols

Security Level	Low	High	Fixed Low	Fixed High
# of Full Associations	Using OSKA for every 50 associations	Using OSKA for every 5 associations	Using standard re-association for every 50 associations	Using standard re-association for every 5 associations

to Data packets which may contain sensitive data. However, we may provide only message authentication to those Management and Control packets such as Beacon, Association/Disassociation, etc. since they contain no secret. However, it depends on the users or application requirements to define such services. We only provide here an example for what we can expect in a typical WLAN.

Table VI.3 defines four security levels for session-level security for associations in IEEE 802.11 WLANs. For fixed security levels, we use the standard re-association as defined in the 802.11 standard [13]. For TuneSec, we use OSKA as the security protocol for re-associations. For the low security level, we infrequently perform the full authentication to refresh the master key, PMK. For a higher security level, the interval of the master key refreshment needs to be shorter to reduce the probability of a successful security attack.

2. Performance Study

In this performance study, our goal is to find how much energy can be saved in typical WLAN environments. We collect packet traces using an 802.11 sniffer. Each packet trace contains packets that have been sent and received by a client or a mobile station. It also includes packets that are broadcast to clients for network management purpose. Such packets are Beacon and CTS/RTS packets for collision avoidance, etc. The traces include all kinds of packets, Management, Control and Data packets according to the 802.11 standard [13].

We collect the traces from 3 different network locations to provide diversity in our test environments. We collect traces in a home network, at Hillman library, and in the School of

Information Science (SIS) building, which have three different characteristics. In the home network, there is only one access point (AP) and few mobile devices. At the library, there are several APs and more client devices, and its physical environment is a big hall room in which received signal strength (RSS) at client devices is probably high, and a typical packet loss rate is low. At the SIS building, its physical environment is a multi-floor office building which also has several APs and client devices. We have collected 12 traces for each location and the summary of traces is shown in Table VI.4. Figure VI.11 shows the Cumulative Distribution Functions (CDF) of the packet size of the traces at three locations. It can be seen that all three locations have similar packet size distributions in which there are more short packets than long packets.

Table VI.4: A Summary of Packet Traces

Location	Traces	Total Packets	Total Bytes	Association Requests
Home	12	286,307	65,669,596	57
Library	12	2,831,383	292,323,060	53
Building	12	1,248,565	375,997,034	41

In addition to five different security schemes, we also use three different cipher schemes for providing security services. The three scheme are using only AES cipher (ALLAES), only RC5 cipher (ALLRC5), and using both AES and RC5 (BOTH). The last scheme is the feature of TuneSec in which we can add more than one security module such as AES and RC5 cipher modules. We use AES and RC5 because from our study, it is shown that for packet sizes of less than 80 bytes, AES consumes less energy than RC5. Therefore, we use AES for packets whose size is less than 80 bytes; otherwise, we use RC5.

For providing message authentication service, we use one MAC algorithm, HMAC-SHA, for the scheme that uses only one cipher, ALLAES and ALLRC5. However, when using BOTH scheme, we use both CBC-MAC and HMAC-SHA. We utilize CBC-MAC to provide 128-bit MAC output, and HMAC-SHA to provide 160, 256, 384, and 512 bits of MAC outputs.

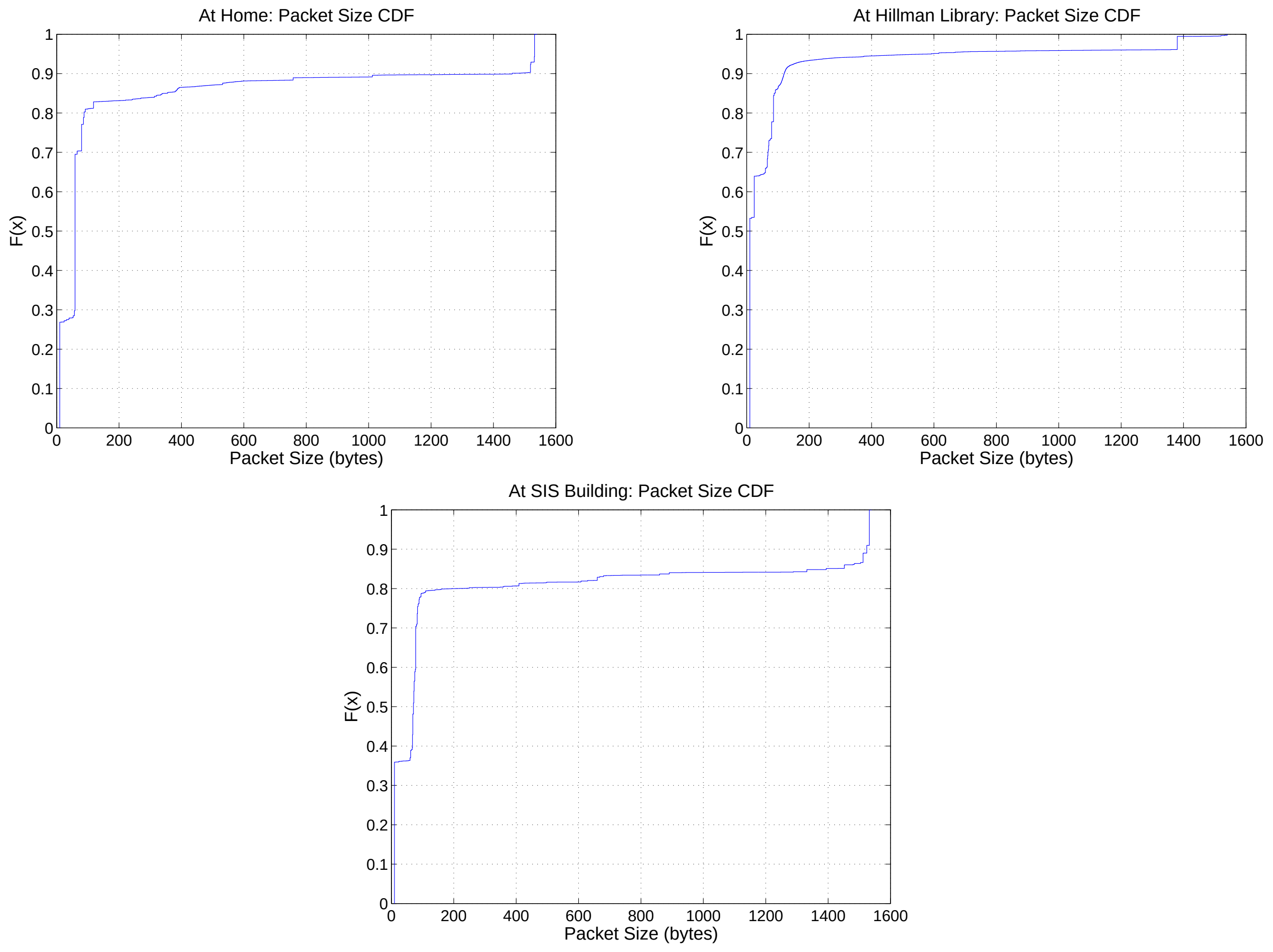
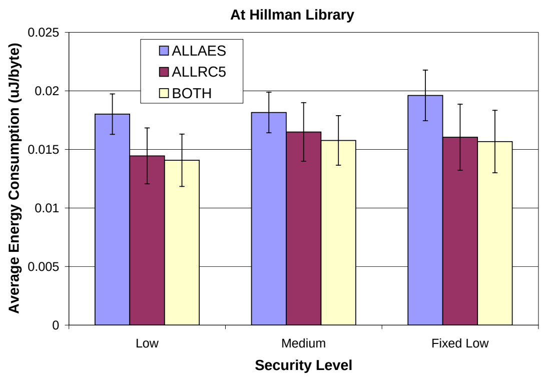


Figure VI.11: The CDF of Packet Size in Different Locations

3. Packet-Level Security Performance Results

In this section, we show the performance results of using TuneSec with five security schemes, three cipher schemes, and at three locations. Figure VI.12 shows the average energy consumption of using fixed and fine-tuning security levels with different cipher schemes for the Hillman library network. The bar height shows the average of normalized energy consumption (in μJoule/byte), and the lines on the bars show the 95% confidential interval.

From the figure, it is shown that using TuneSec can save energy in both the low and high security scenarios compared to the fixed low and high scenarios. The amount of savings using TuneSec in the high security scenario is more than that in the low security scenario. This is



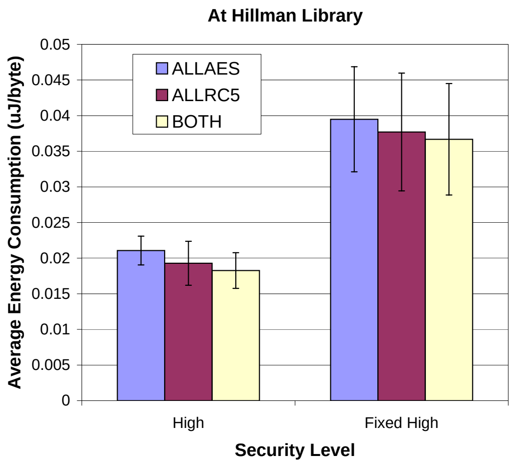


Figure VI.12: Comparison of Different Security Levels for the Hillman Library Network

the case that in the high security scenario, the amount of computation for security is intensive due to the high security requirement, and using TuneSec can leverage the computation and hence save energy.

When comparing between the cipher schemes, ALLAES scheme consumes more energy than ALLRC5 scheme. This shows that in the network, the size of data packets is more than 80 bytes on the average. Note than both schemes use the same MAC function, HMAC-SHA, for authentication of management and control packets whose size is typically smaller than that of data packets. However, BOTH scheme, which utilizes both AES and RC5 ciphers as well as both CBC-MAC and HMAC-SHA, can save more energy although it is not significant in this case.

Figures VI.13 and VI.14 show the average energy consumption of using fixed and fine-tuning security levels for the SIS building network and the Home network, respectively. When comparing three different networks or locations, the energy consumption at the Hillman library location as shown in Figure VI.12 is higher than that in the SIS building and in the Home network because there are more users in the same network at the Library. We show all performance results as the mean and standard deviation in Table VI.5.

Table VI.6 shows the amount of energy consumption (in μJ/byte) which is compared

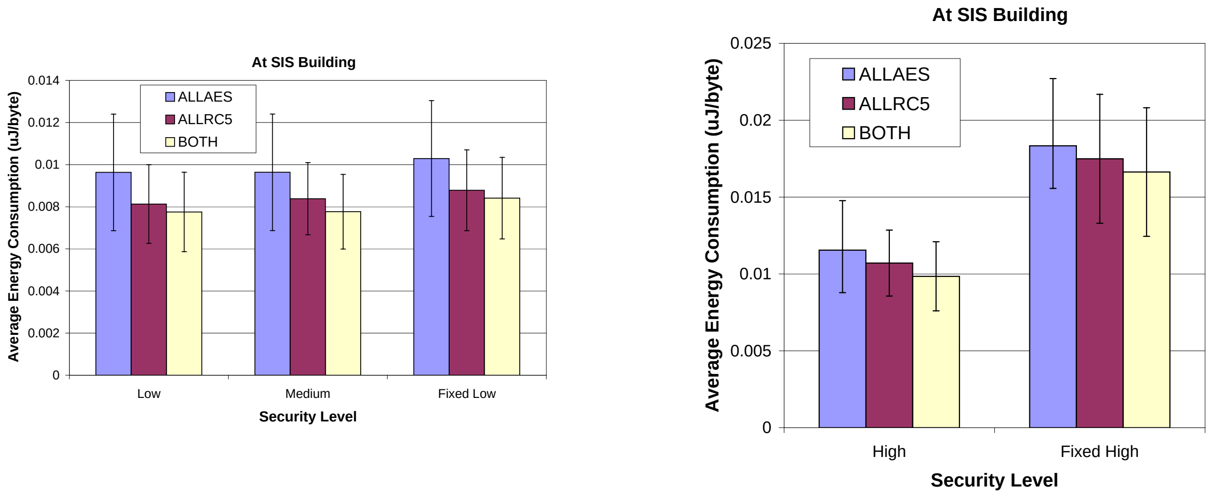


Figure VI.13: Comparison of Different Security Levels for the SIS Building Network

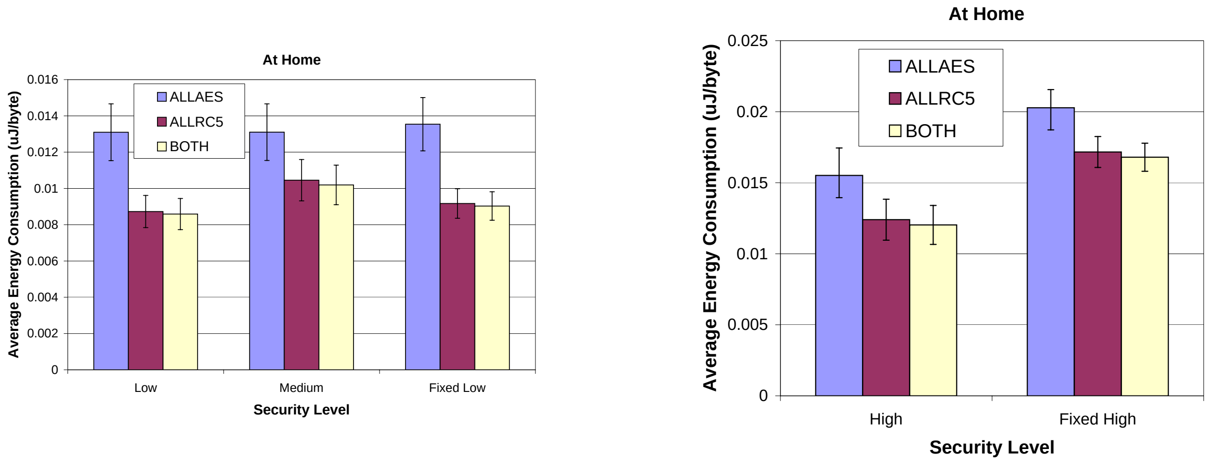


Figure VI.14: Comparison of Different Security Levels for the Home Network

between using fixed security levels and using TuneSec and the percentage of energy saving of different settings. It is shown that we can save up to 8% for low-level security, and up to 43 % for high-level security.

Table VI.5: A Comparison of Energy Consumption

Scheme	Level	SIS Building		At Home		At Library	
		Avg. Energy (μJ/byte)	STDEV	Avg. Energy (μJ/byte)	STDEV	Avg. Energy (μJ/byte)	STDEV
ALLAES	Low	0.00963087	0.00489169	0.01309751	0.00276620	0.01801368	0.00305351
	Medium	0.00963633	0.00488853	0.01310135	0.00275733	0.01815302	0.00306116
	High	0.01155064	0.00568788	0.01551342	0.00341330	0.02106827	0.00356431
	FixedLow	0.01028754	0.00485831	0.01353913	0.00259690	0.01961033	0.00381954
	FixedHigh	0.01663180	0.00742264	0.01679438	0.00175423	0.03667975	0.01400912
ALLRC5	Low	0.00812484	0.00329794	0.00872728	0.00156698	0.01444745	0.00422319
	Medium	0.00838126	0.00302813	0.01045390	0.00201349	0.01649165	0.00441110
	High	0.01070796	0.00379244	0.01239773	0.00254900	0.01928582	0.00545369
	FixedLow	0.00878151	0.00339058	0.00916890	0.00143555	0.01604409	0.00497666
	FixedHigh	0.01749185	0.00740832	0.01716274	0.00192737	0.03770560	0.01460972
BOTH	Low	0.01833452	0.00773463	0.02027844	0.00225421	0.03948805	0.01302484
	Medium	0.00775267	0.00333283	0.00858821	0.00151851	0.01407481	0.00397385
	High	0.00776370	0.00312620	0.01019322	0.00192935	0.01576759	0.00390110
	FixedLow	0.00984792	0.00395622	0.01202937	0.00242636	0.01825997	0.00469288
	FixedHigh	0.00840934	0.00342177	0.00902983	0.00138323	0.01567145	0.00473783

Table VI.6: The Percent Saving of Fixed Security and TuneSec

Scheme	Energy Consumption (μJ/byte)				% Saving	
	Fixed		TuneSec			
	Low	High	Low	High	Low	High
ALLAES	0.01447900	0.02603367	0.01358069	0.01604411	6.204	38.372
ALLRC5	0.01133150	0.02412007	0.01043319	0.01413050	7.928	41.416
BOTH	0.01103688	0.02336864	0.01013856	0.01337908	8.139	42.748

4. Session-level Security Performance Results

In this section, we show the performance results of session-level security (it occurs during the association or re-association phase). We use the traces from three different locations and determine the possible number of associations in the typical WLAN environment. Then, we calculate the amount of energy consumption for each association, which can be a full association, a standard re-association, and the proposed OSKA re-association as shown in Table VI.3. The calculation of energy consumption is similar to that described in Section VI.C.3.

Table VI.7 shows the amount of energy consumption in millijoules in the three typical WLAN environments for two different security levels as described in Section VI.D.1. We

Table VI.7: Session-level Security: Energy Consumption (mJ)

Scheme	Level	SIS Building	At Home	At Library
Full Associations	Low	159.9070	222.3097	206.7090
	High	159.9070	222.3097	206.7090
Standard ReAssociation	Low	103.9164	145.3226	135.321036
	High	113.7147	157.9205	146.519156
OSKA-256	Low	49.2392	70.14151	65.6076139
	High	68.6060	95.04177	87.7411726

can see that the energy consumption of session-level security in the Home environment is higher than that in other environments because there are more associations and possible re-associations as shown in Table VI.4.

It is shown in Table VI.8 that by using OSKA with 256-bit key chain, we can reduce the energy consumed by a significant amount. The reduction is up to 68% for low security level and up to 57% for high security level. The saving is less for the high security level due to the fact that we require a higher number of full associations (which includes expensive authentication) to reduce the probability of a successful attack. Otherwise, we can reduce the energy used for the session-level security as we reduce the number of full associations as well as reduce the security level.

Table VI.8: The percentage of average energy saving using OSKA

Scheme	Energy Consumption (mJ)		% Energy Saving in TuneSec	
	Low	High	Low	High
Full Assoc	196.3085	196.3085	68.58%	57.31%
Standard Reassoc	128.1866	139.3848	51.89%	39.88%
OSKA-256	61.6627	83.7963	-	-

5. TuneSec Performance Summary

In this section, we look at the overall performance of TuneSec security protocols (including session-level and packet-level security protocols) compared to the fixed-level security protocol that is suggested by IEEE 802.11 or WPAv2 standards. We use the traces described in Section VI.4 and combine the average energy consumption at the packet-level and session-level security as already described in Sections VI.D.3 and VI.D.4, respectively. In Table VI.9, we see that by using TuneSec security protocols, we can save energy up to 10% for low-level security and up to 42% for high-level security compared to the standard WLAN security protocol.

Table VI.9: The Average of the Total Energy Consumption (mJ) of Packet Traces

Scheme	Security Level	
	Low	High
Fixed	2825.6109	5831.6779
TuneSec	2539.3036	3332.0114
% Saving	10.13%	42.86%

E. CONCLUSION

We have proposed the concepts of tunable security that the level of security services should be adjustable to support security requirements, yet providing a sufficient security level. We explain the TuneSec architecture that is designed to support tunable security features. We have divided the TuneSec security into packet-level and session-level services. The packet-level security services are at the packet level to provide encryption or confidentiality and message authentication. In contrast, the session-level security services provides authentication and key agreement for a session to be started. The agreed key will be used for the packet-level security.

We have proposed to use the number of years as a quantitative indicator for an abstract security level. From the number of years, we proposed an interpretation of security robustness which is later used to determine the number of operational rounds used by a security module. A high security level requires a security module to operate with a high number of rounds. We used RC5 and AES ciphers as the security modules due to their flexibility, well-known secure design, and their wide usage.

In our previous study in Section V.C, we could save up to 57% by using RC4, AES and CBC-MAC to provide security services at the fixed level of 128-bit key or 70 years of security. By using the packet-level TuneSec which provides a fine-grained security tuning to the packet level, it has been shown that the amount of energy saving can be further reduced by about 6% to 40% compared to using a fixed level security protocol as in our previous study.

The deployment of TuneSec in session-level services is difficult due to the necessity of it being specific to the design of the session-level protocol. Thus, we propose to use the TuneSec concepts to the session-level services only in WLANs. We have proposed One-time Secret Key Authentication (OSKA) as an adjustable authentication and key agreement protocol. OSKA provides a key chain which can be used not only for the packet-level security, but also as an authentication token for a lightweight re-authentication process. Hence, we reduce the number of full re-authentication process which is expensive, and increase the amount of energy saving. OSKA is secure and also energy efficient.

VII. CONCLUSION

Information security services for today's computing is no longer a choice. It must be deployed to prevent information abuse and to protect malicious attacks. Basic information security services are confidentiality or encryption, authentication, and integrity. To provide the services, we need to leverage our computing resources. Small wireless devices often have very limited resources especially battery power. A high and intensive security service is not always the best for wireless devices. We need a way to provide security services that are suitable for the small and wireless devices.

In this work, we proposed the concept of providing just enough security in which the security level is determined and security services are provided with the minimum use of energy resources. We used the concept of tunability of security strength to account for different security needs instead of a fixed level as provided in today's security services.

The tunable security or TuneSec provides a fine-grained security level specification. For example, a fixed security service may prevent an unauthorized access to all of the information for 100 years. However, some part of the information may not require the 100-year protection. By using the TuneSec mechanism, security levels can be specified to each part of the information that requires different security strength. Hence, energy is saved by using appropriate ciphers for appropriate packet sizes and content.

A. SUMMARY

In our work, we have proposed an energy efficient security framework. We suggest three methodologies for energy efficient security protocol design. The first method suggests the replacement of the security algorithm with one that consumes less energy. From our study, we have found this is possible due to the variety of cryptographic algorithms and their differences in performance. We applied this method to the TLS Handshake protocol to use both RSA and ECC public key algorithms to achieve a more energy efficient TLS Handshake protocol that is still able to provide the equivalent level of security strength and still be compatible to the TLS standard. Using this method with the handshake protocol, we can save about 25% to 70% energy compared to the plain handshake protocol.

The second method in our proposed framework suggests the modification to standard protocols in order to achieve energy efficiency. We apply this method to a standard WLAN security protocol. We modify the protocol such that two ciphers are used to provide packet encryption based on the packet size. From our study, we have shown that AES cipher performs better than RC4 cipher when the packet size is about 80 bytes or smaller. We have also utilized the MAC to provide message authentication based on the packet type. In our study, we have achieved approximately up to 57% energy reduction. Using this method, the security level provided to each packet is still fixed.

In the last method, we suggest the greenfield approach, where a new system is designed to provide an energy efficient security protocol. In this work, we developed an example TuneSec system where the security level can be adjusted to provide different levels to different packets. We have shown that the use of TuneSec can save even more energy, from approximately 6% up to 40%, *on the top of the fixed security level system.* The percentage of energy saving may vary depending on the security specification. TuneSec is designed based on a *very simple, yet practical concept.* It can be applied to any security scheme related to any small and limited-resource devices. Especially in wireless networks where a cell phone is built to be smaller and smarter and security services are of paramount importance, the TuneSec framework can be deployed to provide security services, and yet be secure at the very minimum amount of energy consumption.

B. FUTURE RESEARCH

The TuneSec framework is a proof-of-concept that security can be provided at different levels and hence consumes different level of energy. It is not optimal, nor does it use specific benchmarks of available resources to pick cipher suites and so on. One approach to achieve an optimal method of saving energy is to use approaches like [100]. TuneSec also ignores communication protocols. Combining energy efficient communication protocols with security protocols could potentially save more energy.

Future work can also build a prototype utilizing the TuneSec framework and apply it to be used with multimedia applications on small and limited devices over a wireless network. Potential applications are Voice over Internet Protocol (VoIP) over Wireless LAN network, or video-on-demand over WLANs. We expect a tremendous saving on such applications due to their intensive use of the resources.

APPENDIX

ABSOLUTE ACCURACY CALCULATION

The absolute accuracy shows the specification of a measurement of data to show the actual range of the measurement error in a given environment. There are five variables for absolute accuracy calculation:

- Percent of Reading – is uncertainty gain that is multiplied by the actual input voltage for the measurement.
- Offset – is a constant value applied to all measurements.
- System Noise – is a natural random noise that may occur in the system.
- Temperature Drift – is the drift due to the ambient temperature of measurement. If the temperature is between 15 to 35°C, the temperature drift is already compensated by the system. Otherwise, the calculation of additional temperature drift is required.
- Input Voltage – the value of input voltage being measured.

Below is the formula of absolute accuracy and its relative to input (RTI) percentage.

$$\begin{aligned}\text{Absolute Accuracy} &= \pm[(\text{Input Voltage} \times \%\text{ of Reading}) + && (.1)\\ &\quad \text{Offset} + \text{System Noise} + \text{Temperature Drift}] && (.2)\end{aligned}$$

$$\text{Absolute Accuracy RTI} = \pm\frac{\text{Absolute Accuracy}}{\text{Input Voltage}} \tag{.3}$$

In our measurement, we use a data acquisition (DAQ) system which includes an SCXI-1100 module of National Institute to measure the voltage dropped across an 1 ohm resistor which is calculated as an input current used by our laptop or a device-under-test. Our ambient temperature is between 15 to 35°C; hence, there will be no drift. From the specification of SCXI-1100 when using at $\pm 10V$ range, we have the percent of reading of 0.05%, the offset of $250\mu V$ and the noise of $15\mu V$.

From our experiment, we found a range of the voltage between 2.001194 and 2.328655 volts. Therefore, we calculate the absolute accuracy for the minimum and the maximum voltages as follows. The minimum absolute accuracy and its RTI are

$$\begin{aligned}
\text{Min. Absolute Accuracy} &= \pm[(2.001194 \times 0.05\%) + 250\ \mu V + 15\ \mu V + 0] && (.4)\\
&= \pm 1.265\ mV. && (.5)\\
\text{Min. Absolute Accuracy RTI} &= \pm\frac{1.265\ mV}{2.001194\ V} && (.6)\\
&= \pm 0.0632\%. && (.7)\\
& && (.8)
\end{aligned}$$

The maximum absolute accuracy and its RTI are

$$\begin{aligned}
\text{Max. Absolute Accuracy} &= \pm[(2.328655 \times 0.05\%) + 250\ \mu V + 15\ \mu V + 0] && (.9)\\
&= \pm 1.429\ mV. && (.10)\\
\text{Max. Absolute Accuracy RTI} &= \pm\frac{1.265\ mV}{2.001194\ V} && (.11)\\
&= \pm 0.0614\%. && (.12)\\
& && (.13)
\end{aligned}$$

BIBLIOGRAPHY

[1] OpenSSL Software Distribution, http://www.openssl.org/ [last access Dec. 1, 2004].

[2] Cryptlib Software Distribution, http://www.cs.auckland.ac.nz/~pgut001/cryptlib/ [last access Dec. 14, 2004].

[3] Crypto++ Software Distribution, http://www.eskimo.com/~weidai/cryptlib.html [last access Dec. 14, 2004].

[4] RSA Cryptographic Challenges, http://www.rsasecurity.com/rsalabs/challenges/ [last access Dec. 14, 2004].

[5] Wikipedia, Data Encryption Standard, http://en.wikipedia.org/wiki/DES [last access Dec. 22, 2004].

[6] RC5 Project, Distributed.net webpage, http://www1.distributed.net/pressroom/news-20020926.txt [last access Dec. 24, 2004].

[7] Airsnort Project, http://airsnort.shmoo.com/ [last access Dec. 25, 2004].

[8] WEPcrack Project, http://wepcrack.sourceforge.net/ [last access Dec. 25, 2004].

[9] "The datasheet of mobile intel pentium III processor in BGA2 and Micro-PGA2 packages."

[10] "Statistical analysis of the alleged RC4 key stream generator."

[11] "Wi-Fi protected access," Wireless Fidelty (Wi-Fi), http://www.weca.net.

[12] "Data encryption standard (DES)," National Institute of Standards and Technology (NIST), Tech. Rep. FIPS PUB 46-2, December 1993, federal Information Processing Standards Publication.

[13] *IEEE 802.11 Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification*, 1999th ed., 1999, iEEE Standard 802.11.

[14] "Advanced encryption standard (AES)," National Institute of Standards and Technology (NIST), Tech. Rep. FIPS PUB 197, November 2001, federal Information Processing Standards Publication.

[15] *IEEE 802.1x-2001 Standards for Local and Metropolitan Area Networks: Port-Based Network Access Control*, 2001st ed., 2001, iEEE Standard 802.1x.

[16] *Wireless LAN Medium Access Control (MAC) and physical layer (PHY) specifications: Specification for Enhanced Security*, April 2004, iEEE Standard 802.11i.

[17] M. Abadi, "Security protocols and their properties," in *In F.L. Bauer and R. Steinbrueggen, editors, Foundations of Secure Computation*, Marktoberdorf, Germany, 2000, pp. 39–60.

[18] M. Abadi and M. Tuttle, "A semantics for a logic of authentication (extended abstract)," in *Proceedings of the tenth annual ACM symposium on Principles of distributed computing.* Montreal, Quebec, Canada: ACM Press, 1991, pp. 201–216.

[19] B. Aboba and D. Simon, "PPP EAP TLS authentication protocol," October 1999, iETF Request for Comments RFC 2716.

[20] C. Adams, "The CAST-128 encryption algorithm," May 1997, iETF Request for Comments RFC 2144.

[21] J. Al-Muhtadi, D. Mickunas, and R. Campbell, "A lightweight reconfigurable security mechanism for 3G/4G mobile devices," *IEEE Wireless Communications*, vol. 9, no. 2, pp. 60–65, 2002.

[22] J. Al-Muhtadi, A. Ranganathan, R. Campbell, and M. D. Mickunas, "Cerberus: A context-aware security scheme for smart spaces," in *The IEEE International Conference on Pervasive Computing and Communications (PerCom 2003)*, Dallas-Fort Worth, Texas, March 23-26, 2003.

[23] R. Anderson, E. Biham, and L. Knudsen, "Serpent: A proposal for the advanced encryption standard," in *First Advanced Encryption Standard (AES) Conference*, Ventura, California, USA, 1998.

[24] R. Anderson and R. Needham, "Programming Satan's computer," in *J. van Leeuven, editor, Computer Science Today: Recent Trends and Developments, volume 1000 of Lecture Notes in Computer Science Series.* Berlin: Springer-Verlag, 1995.

[25] W. A. Arbaugh, N. Shankar, and J. Wang, "Your 802.11 network has no clothes," in *in Proceedings of the First IEEE International Conference on Wireless LANs and Home Networks*, Singapore, December 2001, pp. 131 – 144.

[26] M. Aydos, T. Yanik, and C. Koc, "An high-speed ECC-based wireless authentication protocol on an ARM microprocessor," in *Proceedings of the 16th Annual Computer*

Security Applications Conference, New Orleans, Louisiana, December 11-15, 2000, pp. 401–409.

[27] J. Bacon, K. Moody, and W. Yao, "A model of OASIS role-based access control and its support for active security," *ACM Trans. Inf. Syst. Secur.*, vol. 5, no. 4, pp. 492–540, 2002.

[28] M. Bellare, J. Kilian, and P. Rogaway, "The security of the cipher block chaining message authentication code," in *Advances in Cryptology - Crypto 94 Proceedings, Lecture Notes in Computer Science Vol. 839*, Y. Desmedt, Ed. Springer-Verlag, 1994.

[29] E. Biham and A. Shamir, "Differential cryptanalysis of DES-like cryptosystems," *Journal of Cryptology*, vol. 4, no. 1, pp. 3–72, 1991.

[30] A. Biryukov and E. Kushilevitz, "Improved cryptanalysis of RC5," in *EUROCRYPT*. Springer-Verlag, 1998, pp. 85–99.

[31] L. Blunk, J. Vollbrecht, B. Aboba, J. Carlson, and H. Levkowetz, "Extensible authentication protocol (EAP)," February 2004, iETF draft-ietf-eap-rfc2284bis-08.txt.

[32] N. Borisov, I. Goldberg, and D. Wagner, "Intercepting mobile communications: The insecurity of 802.11," in *Rome, Italy*, July 2001.

[33] J. Borst, B. Preneel, and J. Vandewalle, "Linear cryptanalysis of RC5 and RC6," in *FSE '99: Proceedings of the 6th International Workshop on Fast Software Encryption*. Springer-Verlag, 1999, pp. 16–30.

[34] P. Boutin, "Feds label Wi-Fi a terrorist tool," December 6, 2002, news from Wired, http://www.wired.com/news/wireless/0,1382,56742,00.html.

[35] K. Brincat and C. J. Mitchell, "New CBC-MAC forgery attacks," in *Information Security and Privacy, ACISP 2001, LNCS 2119*, V. Varadharajan and Y. Mu, Eds. Sydney, Australia: Springer-Verlag, July 2001, pp. 3–14.

[36] M. Brown, D. Cheung, D. Hankerson, J. Hernandez, M. Kirkup, and A. Menezes, "PGP in constrained wireless devices," in *Proceedings of the 9th USENIX Security Symposium*, Denver, Colorado, USA, August 2000.

[37] M. Burrows, M. Abadi, and R. Needham, "A logic of authentication," *Transactions of Computer Systems*, vol. 8, no. 1, pp. 18–36, February, 1990.

[38] D. Carmen, P. Kruus, and B. Matt, "Constraints and approaches for distributed sensor network security," NAI Labs, Tech. Rep. Technical Report 00-010, September 2000.

[39] D. Carrel and D. Harkins, "The internet key exchange (IKE)," November 1998, iETF Request for Comments RFC 2409.

[40] J. Cox, "Serious security weakness in 802.11b wireless LANs exposed," August 6, 2001, news from Network World Fusion, http://www.nwfusion.com/news/2001/0806ieee.html.

[41] J. Daemen and V. Rijmen, "AES proposal: Rijndael," September 1999, http://csrc.nist.gov/encryption/aes/rijndael/Rijndael.pdf.

[42] ——, *The Design of Rijndael: AES - The Advanced Encryption Standard.* Springer-Verlag, February 2002.

[43] T. Dierks and C. Allen, "The TLS protocol version 1.0," January 1999, iETF Request for Comments RFC 2246.

[44] H. Dobbertin, "The status of MD5 after a recent attack," *RSA Labs' CryptoBytes*, vol. 2, no. 2, 1996.

[45] L. Feeney and M. Nilsson, "Investigating the energy consumption of a wireless network interface in an ad hoc networking environment," in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, Anchorage AK, April, 2001.

[46] N. Ferguson, "Michael: an improved MIC for 802.11 WEP," IEEE 802.11i Working Group, Tech. Rep. IEEE 802.11-02/020r0, Jan 17 2002.

[47] N. Ferguson, J. Kelsey, S. Lucks, B. Schneier, M. Stay, D. Wagner, and D. Whiting, "Improved cryptanalysis of Rijndael," in *FSE '00, Proceedings of the 7th International Workshop on Fast Software Encryption, Lecture Notes in Computer Science No.1978.* Springer-Verlag, 2000, p. 213230.

[48] S. Fluhrer, I. Mantin, and A. Shamir, "Weaknesses in the key scheduling algorithm of RC4," in *The Eighth Annual Workshop on Selected Areas in Cryptography (SAC2001)*, Toronto, Ontario, Canada, August 2001.

[49] E. F. Foundation, *Cracking DES - Secrets of Encryption Research, Wiretap Politics & Chip Design.* San Francisco, CA: O'Reilly, 1998.

[50] P. Funk and S. Blake-Wilson, "EAP tunneled TLS authentication protocol (EAP-TTLS)," August 2003, iETF Internet Draft, draft-ietf-pppext-eap-ttls-03.txt, work in progress.

[51] V. Gligor, R. Kailar, S. Stubblebine, and L. Gong, "Logics for cryptographic protocols - virtues and limitation," in *Proceedings of the IEEE Computer Security Foundations Workshop IV*, Franconia, New Hampshire, June, 1991, pp. 219–226.

[52] A. Grosul and D. Wallach, "A related-key cryptanalysis of RC4," Rice University, TX, Tech. Rep. TR00-358, June 2000.

[53] N. Haller, "The S/KEY one-time password system," in *Proceedings of the ISOC Symposium on Network and Distributed System Security*, San Diego, CA, 1994.

[54] D. Hankerson, J. Hernandez, and A. Menezes, “Software implementation of elliptic curve cryptography over binary fields,” in *Cryptographic Hardware and Embedded Systems (CHES), Lecture Notes in Computer Science 1965*. Springer-Verlag, 2000, pp. 1–24.

[55] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, “Energy-efficient communication protocol forwireless microsensor networks,” in *Proceedings of the 33rd Hawaii International Conference on System Sciences*, Maui, Hawaii, 2000.

[56] S. Hirani, “Energy efficiency of encryption schemes in wireless devices,” Master’s thesis, Telecommunications Program, University of Pittsburgh, Pittsburgh, Pennsylvania, 2003.

[57] C. Irvine and T. Levin, “Quality of security service,” in *NSPW ’00: Proceedings of the 2000 workshop on New security paradigms*, Ballycotton, County Cork, Ireland, 2000, pp. 91–99.

[58] R. D. Isaac, “The future of cmos technology,” *IBM Journal of R&D*, vol. 44, no. 3, pp. 369–378, 2000.

[59] D. Johnson and A. Menezes, “The elliptic curve digital signature algorithm (ECDSA),” Center for Applied Cryptographic Research, University of Waterloo, Canada, Tech. Rep. Technical Report, CORR 99-34, 1999.

[60] J. B. D. Joshi, B. Shafiq, A. Ghafoor, and E. Bertino, “Dependencies and separation of duty constraints in GTRBAC,” in *Proceedings of the eighth ACM symposium on Access control models and technologies*. ACM Press, 2003, pp. 51–64.

[61] B. S. Kaliski and Y. L. Yin, “On the security of the RC5 encryption algorithm,” RSA Laboratories, Technical Report TR-602, Version 1.0, September 1998.

[62] T. Karygiannis and L. Owens, *Wireless Network Security: 802.11, Bluetooth and Handheld Devices*, November 2002, special Publication 800-48.

[63] S. Kent and R. Atkinson, “Security architecture for the internet protocol,” November 1998, iETF Request for Comments RFC 2401.

[64] V. Klma, “Finding MD5 collisions a toy for a notebook,” 2005, http://cryptography.hyperlink.cz [last access Mar. 29, 2005].

[65] H. Krawczyk, August, 22 2004, iETF Email Archive, Available at http://www1.ietf.org/mail-archive/web/cfrg/current/msg00527.html.

[66] H. Krawczyk, M. Bellare, and R. Canetti, “HMAC: Keyed-hashing for message authentication,” February 1997, iETF Request for Comments RFC 2104.

[67] K. Lahiri, A. Raghunathan, S. Dey, and D. Panigrahi, “Battery-driven system design: A new frontier in low power design,” in *Asia South Pacific Design Automation*

Conference (ASP-DAC) / International Conference on VLSI Design, Jan. 2002, pp. 261–267.

[68] L. Lamport, "Password authentication with insecure communication," *Communications of the ACM*, vol. 24, 1981.

[69] A. Lenstra and E. Verheul, "Selecting cryptographic key sizes," *Journal of Cryptology*, vol. 14, no. 4, pp. 255–293, 2001.

[70] P. Lettieri, C. Schurgers, and M. Srivastava, "Adaptive link layer strategies for energy efficient wireless networking," *Wireless Networks*, vol. 5, no. 5, pp. 339–355, 1999.

[71] L. Li and J. Halpern, "Minimum energy mobile wireless networks revisited," in *Proc. IEEE International Conference on Communications (ICC)*, June 2001.

[72] S. Lucks, "Attacking seven rounds of Rijndael under 192-bit and 256-bit keys," in *The Third AES Candidate Conference, printed by the National Institute of Standards and Technology (NIST)*, Gaithersburg, MD, 2000, pp. 215–229.

[73] M. Matsui, "Linear cryptanalysis method for DES cipher," in *Advances in Cryptology–EUROCRYPT'93 Proceedings.* Springer-Verlag, 1994, pp. 386–397.

[74] A. Menezes, *Elliptic Curve Public Key Cryptosystems*, 1st ed. Kluwer Academic Publishers, June 1993.

[75] A. Menezes, P. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography.* CRC Press, October 1996.

[76] C. J. Michell, "On the security of XCBC, TMAC, and OMAC," Royal Holloway, University of London, Mathematics Department, Technical Report RHUL-MA-2003-4, August 2003.

[77] S. Mister and S. Tavares, "Cryptanalysis of RC4-like cipher," in *Fifth Annual Workshop on Selected Areas in Cryptography SAC*, Kingston, Ontario, Canada, 1998, pp. 136–148.

[78] G. E. Moore, "Cramming more components onto integrated circuits," *Electronics*, vol. 38, no. 8, April 19, 1965, ftp://download.intel.com/research/silicon/moorespaper.pdf.

[79] K. Naik and D. Wei, "Software implementation strategies for power-conscious systems," *Mobile Networks and Applications*, vol. 6, no. 3, pp. 291–305, June 2001.

[80] B. C. Neuman and T. Ts'o, "Kerberos: An authentication service for computer networks," *IEEE Communications*, vol. 32, no. 9, pp. 33–38, September 1994.

[81] N. I. of Standards and T. (NIST), "FIPS PUB 180-1: Secure hash standard," April 1995.

[82] K. Pahlavan and P. Krishnamurthy, *Principles of Wireless Networks - A Unified Approach.* Prentice Hall, 2002.

[83] L. Paulson, “Will fuel cells replace batteries in mobile devices?” *Computer*, November 2003.

[84] N. Potlapally, S. Ravi, A. Raghunathan, and G. Lakshminarayana, “Optimizing public-key encryption for wireless clients,” in *International Conference on Communications (ICC)*, New York City, New York, May 2002.

[85] P. Prasithsangaree and P. Krishnamurthy, “Analysis of tradeoffs between security strength and energy savings in security protocols for WLANs,” in *IEEE Fall Semi-annual Vehicular Technology Conference (VTC)*, Los Angeles, CA, 2004.

[86] ——, “On a framework for energy-efficient security protocols in wireless networks,” *Elsevier Computer Communications*, vol. 27, no. 17, pp. 1716–1729, 2004.

[87] ——, “A variant of WTLS authentication protocol for reducing energy consumption in wireless devices,” in *the 7th IEEE International Conference on High Speed Networks and Multimedia Communications (HSNMC’04)*, Toulouse, France, 2004.

[88] ——, “Analysis of energy consumption of RC4 and AES algorithms in wireless LANs,” in *IEEE Global Communications Conference (Globecom’03)*, San Francisco, CA, December 1-5, 2003.

[89] B. Preneel, “Cryptanalysis of message authentication codes,” in *Information Security, 1st International Workshop, ISW 1997, Lecture Notes in Computer Science 1396.* Springer-Verlag, 1997, pp. 55–65.

[90] T. Qian and R. Campbell, “Dynamic agent-based security architecture for mobile computers,” in *Proceedings of the International Conference on Parallel and Distributed Computing and Networks (PDCN’98)*, Australia.

[91] E. Rescorla, *SSL and TLS Designing and Building Secure Systems.* Addison-Wesley, 2000.

[92] C. Rigney, S. Willens, A. Rubens, and W. Simpson, “Remote authentication dial in user service (RADIUS),” June 2000, request for Comments, RFC 2865.

[93] R. Rivest, rSA Technote, Revised September 1, 2001, http://www.rsasecurity.com/rsalabs/node.asp?id=2009 [last access Dec. 25, 2004].

[94] ——, “The MD5 message-digest algorithm,” April 1992, iETF Request for Comments RFC 1321.

[95] ——, “The RC4 encryption algorithm,” March 1992, rSA Data Security, Inc.

[96] ——, "The RC5 encryption algorithm," in *Fast Software Encryption, Lectures Notes in Computer Science LNCS 1008*. Leuven, Belgium: Springer-Verlag, 1995, pp. 86–96.

[97] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[98] P. Rogaway and D. Coppersmith, "A software-optimized encryption algorithm," *Journal of Cryptology*, vol. 11, no. 4, pp. 273–287, 1998.

[99] N. Ruangchaijatupon and P. Krishnamurthy, "Encryption and power consumption in wireless LANs," in *The Third IEEE Workshop on Wireless LANs*, Newton, Massachusetts, September 2001.

[100] C. Rusu, R. Melhem, and D. Mossé, "Maximizing the system value while satisfying time and energy constraints," *IBM Journal of R&D*, vol. 47, no. 5/6, pp. 689–702, 2003.

[101] B. Schneier, "Description of a new variable-length key, 64-bit block cipher (Blowfish)," in *Proceedings of Fast Software Encryption, Cambridge Security Workshop*. Springer-Verlag, December 1993, pp. 191–204.

[102] ——, *Applied Cryptography: Protocols, Algorithms and Source Code in C*, 2nd ed. Addison-Wesley, 1996.

[103] B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, and N. Ferguson, *The Twofish Encryption Algorithm: A 128-Bit Block Cipher*. Wiley-Addison, March, 1999.

[104] B. Schneier and D. Whiting, "Fast software encryption: Designing encryption algorithms for optimal software speed on the Intel Pentium processor," in *Lecture Notes in Computer Science LNCS 1267*. Springer-Verlag, 1997, pp. 242–259.

[105] ——, "A performance comparison of the five AES finalists, third AES candidate conference," 2000, http://www.counterpane.com/aes-comparison.html.

[106] E. Shih, S. Cho, N. Ickes, R. Min, A. Sinha, A. Wang, and A. Chandrakasan, "Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks," in *The ACM Annual International Conference on Mobile Computing and Networking (MobiCom)*, Rome, Italy, July 2001.

[107] W. Simpson, "PPP challenge handshake authentication protocol (CHAP)," August 1996, request for Comments 1994.

[108] A. Sinha and A. Chandrakasan, "JouleTrack - a web based tool for software energy profiling," in *Proceedings of the 38th Design Automation Conference*, Las Vegas, NV, June 2001.

[109] W. Stallings, *Cryptography and Network Security: Principles and Practice*, 3rd ed. Prentice Hall, 2002.

[110] D. Stinson, *Cryptography: Theory and Practice*, 2nd ed. Chapman & Hall, 2002.

[111] A. Stubblefield, J. Ioannidis, and A. D. Rubin, "Using the fluhrer, mantin, and shamir attack to break WEP," in *Proceedings of 2002 Network and Distributed System Security Symposium Conference*, San Diego, California, 2002.

[112] S. W. Tak and E. K. Park, "Adaptive secure software architecture for electronic commerce," *Software: Practice and Experience*, vol. 33, no. 14, pp. 1343 – 1357, 2003.

[113] T. Takahashi, "Wpa passive dictionary attack overview," November 2004, http://www.tinypeap.com/ [last access: Mar. 21, 2004].

[114] J. Viega and G. McGraw, *Building Secure Software: How to Avoid Security Problems the Right Way*. Addison-Wesley, 2001.

[115] M. Viredaz and D. Wallach, "Power evaluation of a handheld computer: A case study," Compaq Western Research Lab, Tech. Rep. 2001/1, 2001.

[116] X. Wang, D. Feng, X. Lai, and H. Yu, "Collisions for hash functions MD4, MD5, HAVAL-128 and RIPEMD," 2004, http://eprint.iacr.org/2004/199/ [last access: Mar 30, 2005].

[117] X. Wang, Y. L. Yin, and H. Yu, "Collision search attacks on SHA1," February 2005, http://theory.csail.mit.edu/~yiqun/shanote.pdf [last access: Mar. 30, 2005].

[118] E. W. Weisstein, "Correlation coefficient," from MathWorld–A Wolfram Web Resource, http://mathworld.wolfram.com/CorrelationCoefficient.html.

[119] ——, "Least squares fitting," from MathWorld–A Wolfram Web Resource, http://mathworld.wolfram.com/LeastSquaresFitting.html.

[120] D. Whiting, R. Housley, and N. Ferguson, "Counter with CBC-MAC (CCM)," September 2003, iETF Request for Comments RFC 3610.

[121] M. Zorzi and R. Rao, "Energy constrained error control for wireless channels," *IEEE Personal Communications*, vol. 4, no. 6, Dec 1997.

[122] R. Zuccherato and C. Adams, "Using elliptic curve diffie-hellman in the SPKM GSS-API," August 1999, iETF Internet Draft draft-ietf-cat-ecdh-spkm-00.txt.