Analyzing the Energy Consumption of Security Protocols

Nachiketh R. Potlapally†, Srivaths Ravi‡, Anand Raghunathan‡ and Niraj K. Jha†
† Dept. of Electrical Engineering, Princeton University, Princeton, NJ 08544
‡NEC Laboratories America, Princeton, NJ 08540

Abstract

Security is critical to a wide range of wireless data applications and services. While several security mechanisms and protocols have been developed in the context of the wired Internet, many new challenges arise due to the unique characteristics of battery powered embedded systems. In this work, we focus on an important constraint of such devices – battery life – and examine how it is impacted by the use of security protocols.

We present a comprehensive analysis of the energy requirements of a wide range of cryptographic algorithms that are used as building blocks in security protocols. Furthermore, we study the energy consumption requirements of the most popular transport-layer security protocol SSL (Secure Sockets Layer). To our knowledge, this is the first comprehensive analysis of the energy requirements of SSL. For our studies, we have developed a measurement-based experimental testbed that consists of an iPAQ PDA connected to a wireless LAN and running Linux, a PC-based data acquisition system for real-time current measurement, the OpenSSL implementation of the SSL protocol, and parametrizable SSL client and server test programs. We investigate the impact of various parameters at the protocol level (such as cipher suites, authentication mechanisms, and transaction sizes, etc.) and the cryptographic algorithm level (cipher modes, strength) on overall energy consumption for secure data transactions.

Based on our results, we discuss various opportunities for realizing energy-efficient implementations of security protocols. We believe such investigations to be an important first step towards addressing the challenges of energy efficient security for battery-constrained systems.

Categories and Subject Descriptors

E.3 [**Data**]: Data Encryption; C.2.0 [**Computer Systems Organization**]: Computer-Communication Networks- *General (Security and protection)*; D.4.6 [**Software**]: Operating Systems- *Security and Protection*; C.2.1 [**Computer Systems Organization**]: Computer-Communication Networks- *Network Architecture and Design (Wireless Communication)*; C.4 [**Performance of Systems**]: Measurement Techniques

General Terms

Security, Performance, Measurement, Algorithms

Keywords

Security, Energy analysis, Low-power, Handheld, Embedded system, Security protocols, Cryptographic algorithms, DES, 3DES, AES, RSA, ECC, DSA, Diffie-Hellman, SSL

1. INTRODUCTION

Today, an increasing number of battery-powered embedded systems – PDAs, cell phones, networked sensors, and smart cards, to name a few – are used to store, access, manipulate, or communicate sensitive data, making security an important issue. Security concerns in such systems range from user identification, to secure information storage, secure software execution, and secure communications. Most battery-powered systems contain wireless communication capabilities for untethered operation, introducing new security concerns due to the public nature of the physical communication medium or channel.

With the evolution of the Internet, network and communications security has gained significant attention [1, 2, 3, 4]. Secure communication across wired and wireless networks is typically achieved by employing security protocols at various layers of the network protocol stack (*e.g.*, WEP [5] at the link layer, IPSec [6] at the network layer, TLS/SSL [7] and WTLS [8] at the transport later, SET at the application layer, *etc.*). The building blocks of a security protocol are cryptographic algorithms, which are selected based on the security objectives that are to be achieved by the protocol. They include asymmetric and symmetric encryption algorithms, which are used to provide authentication and privacy, as well as hash or message digest algorithms that are used to provide message integrity.

While security protocols and the cryptographic algorithms they contain address security considerations from a functional perspective, many embedded systems are constrained by the environments they operate in and the resources they possess. For such systems, there are several challenges that need to be addressed in order to enable secure computing and communications. For battery-powered embedded systems, perhaps one of the foremost challenges is the mismatch between the (energy and performance) requirements of security processing [1] and the available battery and processor capabilities. Rapid increases in communication data rates and security levels required, together with slow increases in battery capacities, threaten to widen this "battery gap" to a point where it will impede the adoption of applications and services that require security.

In this work, we demonstrate that security processing can have a significant impact on battery life. *Addressing the battery gap in secure communications requires that we first analyze and understand the energy consumption characteristics of security protocols and cryptographic algorithms. This paper presents a comprehensive energy measurement and analysis of the most popular transport-layer security protocol used in the Internet, the Secure Sockets Layer (SSL) or Transport Layer Security (TLS) protocol. To our knowledge, this is the first comprehensive energy analysis of the energy requirements of SSL/TLS.* The energy analysis in this study is performed by executing secure data transactions on a battery-powered system (a Compaq iPAQ PDA [9]), measuring the current drawn from the power supply, and calculating the energy consumed during the time intervals in which the security protocol or its constituent cryptographic algorithms are executed. Our results can be used to explore the impact of various parameters, at the protocol and cryptographic algorithm levels, on overall energy consumption for secure data transactions. Based on our analysis, we discuss various opportunities for energy-efficient implementations of security protocols.

The rest of this paper is organized as follows. Section 2 motivates the need to address energy consumption issues in security protocols. Section 3 introduces the reader to pertinent security terms and concepts. Section 4 describes the experimental research testbed used in our work to execute and analyze secure wireless transactions, and provides details of the energy measurement testbed. Section 5 presents the results of our energy measurements, applies this information to analyze the SSL protocol, and suggests ways of optimizing the energy requirements of SSL. Section 6 summarizes the insights gathered in this work, and enumerates future avenues of research.

2. MOTIVATION

We consider the following example system to motivate the need to address energy consumption issues in security protocols: a sensor node, using a Motorola "DragonBall" MC68328 processor and operating at a data rate of 10Kbps, consumes 21.5mJ and 14.3mJ, for transmitting and receiving 1024 bits of data, respectively [10]. In secure mode, when RSA encryption is used as part of a security protocol, encrypting 1024 bits of data on the node was observed to consume 42mJ of energy. Thus, given a typical battery capacity of 26KJ in sensor nodes, it can be shown that with encryption on the battery runs out more than twice as fast as when there is no encryption. This example motivates us to investigate techniques to facilitate energy-efficient execution of security protocols. This objective can be achieved in multiple ways. For example:

- By making the execution of underlying cryptographic algorithms efficient through a combination of hardware and software techniques [11, 12, 13, 14, 15], we can improve the performance and energy requirements of security protocols. Usually, there is an overhead, in the form of increase in silicon area

[1] We use the term *security processing* to refer to any computations performed for the sake of security, including the execution of security protocols and cryptographic algorithms.

ISLPED'03, August 25–27, 2003, Seoul, Korea.

or more complex software, associated with these techniques.

- We can make the security protocols energy-cognizant, by allowing them to alter their operation depending on the operating environment. This adaptation of behavior is guided by rules, which determine the best possible alternative with respect to energy efficiency, under any given input conditions. These changes may involve a conscious tradeoff between the level of security and energy.

The challenges of energy-efficient secure communications can be better addressed if energy requirements and bottlenecks are well understood. In this work, we perform a detailed analysis of the energy requirements of various cryptographic primitives, with the intention of using this data as a foundation for devising energy-efficient security protocols. We performed several experiments where we varied several protocol and cryptographic algorithm level parameters and observed the impact on energy. We use the results of our experiments to suggest ways for making the execution of the SSL protocol energy-efficient.

Security protocols and cryptographic algorithms are known to have significant computational requirements, and studies have indicated that they stretch the processor capabilities available in many embedded systems [16, 17, 18, 19, 20, 21]. While researchers have quantified and addressed the performance overhead of security, the energy implications are relatively less understood. Nevertheless, researchers have recently proposed interesting approaches to the design of lightweight security protocols. Low-power key management protocols have been devised for sensor nodes by analyzing the impact of security algorithms on the energy consumption of sensor nodes [10]. The work in [22] evaluated the energy consumption of selected key-exchange protocols on a WINS sensor node, and proposed energy-efficient ways for exchanging cryptographic keys, while custom protocols for low-power mutual authentication were proposed in [23, 24]. Energy tradeoffs in the network protocol and key management design space of sensor nodes were explored in [25]. Techniques to minimize the energy consumed by secure wireless sessions have also been proposed in [26]. We believe that comprehensive energy analyses of security protocols, such as the one performed in our work, will facilitate identification of energy bottlenecks and development of energy-efficient security mechanisms.

3. PRELIMINARIES: THE SECURE SOCKETS LAYER (SSL) PROTOCOL

In this section, we provide a brief overview of the popular security protocol SSL, which is widely used for secure connection-oriented transactions. SSL offers the basic security services of encryption, source authentication, and integrity protection, for data exchanged over underlying unprotected networks.

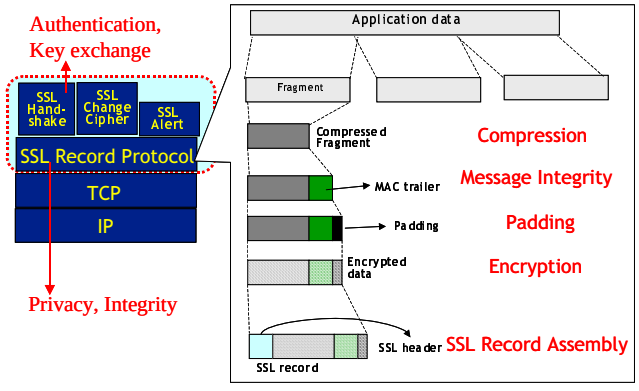


Figure 1: The SSL protocol, with an expanded view of the SSL record protocol

The SSL protocol is typically layered on top of TCP/IP layers of the protocol stack, and is either embedded in the protocol suite or is integrated with applications such as browsers. The SSL protocol consists of two main layers as shown in Figure 1. The SSL record protocol provides the basic services of privacy and integrity to the higher-layer protocols: SSL handshake, SSL change cipher and SSL alert. Let us now examine how the SSL record protocol is used to encrypt application data. The first step involves breaking the application data into smaller fragments. Each fragment is then compressed, if compression options are enabled. The next step involves computing a message authentication code (MAC), which facilitates message integrity. The compressed message plus MAC is then encrypted using a symmetric cipher. If the symmetric cipher is a block cipher, then a few padding bytes may be added. Finally, an SSL header is attached to complete the assembly of the SSL record. The header contains various fields including the higher-layer protocol used to process the attached fragment.

Of the three higher-layer protocols, SSL handshake is the most complex and consists of a sequence of steps that allows a server and client to authenticate each other and negotiate the various cipher parameters needed to initiate a session. For example, the SSL handshake is responsible for negotiating a common suite of cryptographic algorithms (cipher-suite), which can then be used for session key exchange, authentication, bulk encryption and hashing. The cipher-suite RSA-3DES-SHA1, for example, indicates that RSA can be used for key agreement (and authentication), while 3DES and SHA1 can be used for bulk encryption and integrity computations, respectively. More than 30 such cipher suite choices exist in the OpenSSL implementation [27] of the SSL protocol, resulting from combinations of various cipher alternatives for implementing the individual security services.

Finally, the SSL change cipher protocol allows for dynamic updates of cipher suites used in a connection, while the SSL alert protocol can be used to send alert messages to a peer. Further details of the SSL protocol can be found in [3].

4. EXPERIMENTAL SETUP

Figure 2 describes the experimental setup used to execute secure client-server interactions, and the testbed developed to quantify the energy consumption of the various constituent security protocols.

The experimental setup for secure client-server communication consists of a client that connects to a LAN through a wireless access point, while the server is a PC that is wired to the LAN. The handheld used in the experiment is a Compaq iPAQ H3670, which contains an Intel SA-1110 StrongARM processor clocked at 206MHz. It is provided with 64MB of RAM and 16MB of FlashROM, and has an expansion sleeve which allows for memory expansion using compact flash cards. It connects to the wireless access point using a Cisco Aironet 350 series WLAN card. The handheld also supports additional communication capability through a serial port, a USB port and IrDA at 115.2 Kbps. It is powered by a Li-Polymer battery with a 950 mAh rating. The handheld uses the Familiar distribution [28] of Linux as its OS. The server is a PC equipped with a 700MHz Intel Pentium III having 256MB of RAM and running the RedHat Linux OS. The security of client and server interactions is provided by the SSL software from the OpenSSL [27] open source project.

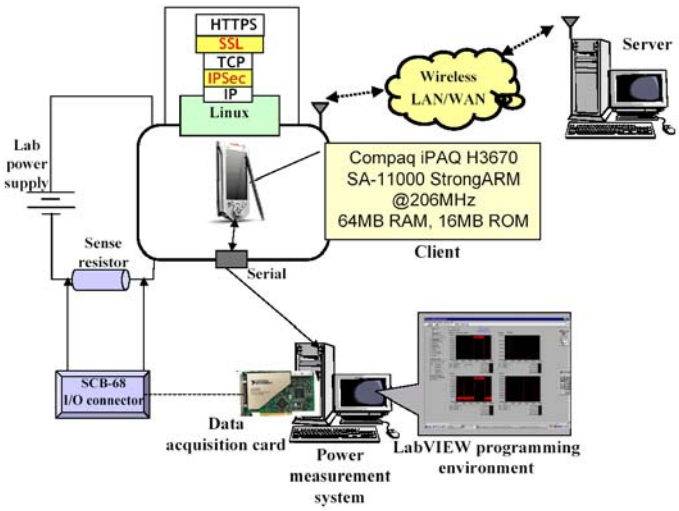


Figure 2: Secure client-server configuration and the energy measurement testbed

The energy consumption values for individual cryptographic algorithms are obtained by running their implementations on the client, and measuring the current drawn from the power supply. Figure 2 also shows the arrangement used for measuring the energy consumption of the cryptographic algorithms. The energy measurement is done using

LabVIEW [29], a GUI-based data acquisition, measurement analysis, and presentation software. The data acquisition software runs on a PC (called a power measurement system), which is also directly connected to the handheld through its serial port. This enables the handheld to send synchronization signals to the data acquisition unit to start and stop the energy measurements. This signaling mechanism allows us to precisely measure the energy dissipated by the chosen software kernels. The current drawn by the client is measured by connecting a sense resistor in series between the handheld and the energy source, *i.e.*, the battery. The voltage drop across the sense resistor is measured using an SCB-68 I/O connector block [29]. This block interfaces to the data acquisition software, LabVIEW, through a data acquisition (DA) card in the PC running the LabVIEW software. LabVIEW is used to calculate the energy supplied to the handheld by integrating power over the time interval between the start and stop synchronizing signals.

5. RESULTS

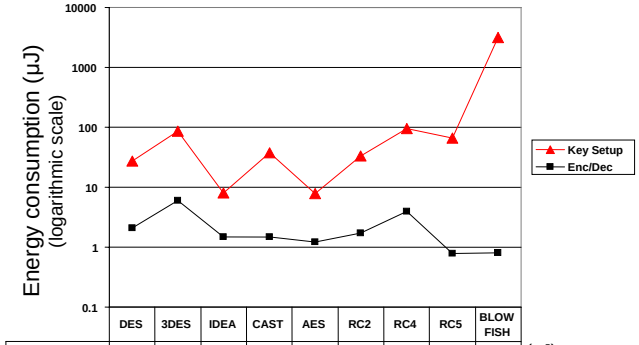
In this section, we present a comprehensive empirical analysis of the energy consumption characteristics of cryptographic algorithms (Section 5.1) using the experimental set-up described in Section 4. We also present a comprehensive energy analysis for various stages of the SSL protocol (Section 5.2).

5.1 Energy Analysis of Cryptographic Algorithms

We first analyze how the choice of a cryptographic algorithm for a given function (privacy, message integrity and authentication) and the choice of settings for various cipher parameters (key size, cipher mode) can lead to varying levels of energy consumption (Sections 5.1.1 - 5.1.4). While the energy results were evaluated in the context of the SSL protocol, the conclusions are broadly applicable since the same underlying cryptographic algorithms are used in other protocols such as WTLS, IPSec, *etc*. The last part of this section (Section 5.1.5) illustrates the energy consumption versus security trade-offs possible by identifying and varying cipher parameters in a cryptographic algorithm.

5.1.1 Symmetric Ciphers

Symmetric ciphers can be chosen from two classes for use in a security protocol - *block* and *stream* ciphers. Block ciphers operate on similar-sized blocks of plain-text and cipher-text. Examples of block ciphers include DES, 3DES, AES, *etc*. Stream ciphers such as RC4 convert a plain-text to cipher-text one bit (or byte) at a time. Before a block or stream cipher starts the encryption/decryption operation, the input key (usually, 64 bits) is expanded in order to derive a distinct and cryptographically strong key for each of the rounds (*key setup*). Encryption or decryption in symmetric algorithms then proceeds through a repeated sequence (rounds) of mathematical computations.



	DES	3DES	IDEA	CAST	AES	RC2	RC4	RC5	BLOW FISH
Key Setup	27.53	87.04	7.96	37.63	7.87	32.94	95.97	66.54	3166.3
Enc/Dec	2.08	6.04	1.47	1.47	1.21	1.73	3.93	0.79	0.81

(µJ)
(µJ/byte)

Figure 3: Energy consumption data for various symmetric ciphers

Figure 3 shows variations in energy consumption due to the use of different symmetric ciphers. Energy numbers for the key setup phase and energy-per-byte numbers for encryption/decryption phases are shown for each cipher. The results are reported for one specific mode of each block cipher - ECB or electronic code book, where a given plain-text block always encrypts to the same cipher-text block for the same key (the impact of different modes on energy is explored later in Section 5.1.4). The only exception is RC4, which is a stream cipher. Taking into account both the key setup and encryption/decryption costs, we see from Figure 3 that AES has the least energy cost and BLOWFISH the greatest. The large cost of BLOWFISH is primarily due to its very high key setup cost, since the expanded key in BLOWFISH consists of sub-keys totaling 4168 bytes which delivers very robust security. The cost of BLOWFISH encryption/decryption is quite small. In case of sufficiently large data transactions, one would expect the cost of key setup to be amortized by the low encryption cost. It is interesting to note that the energy costs of IDEA, for both encryption/decryption and key-setup, compare well with those of AES. However, the cryptanalytical strength of AES is superior to that of IDEA, making the former an attractive option.

5.1.2 Hash Algorithms

Table 1 summarizes the energy cost of commonly used hashing algorithms. In general, hash algorithms are the least complex of the cryptographic algorithms, and should intuitively incur the least energy cost. From Table 1, MD2 and HMAC are observed to be more compute-intensive than the rest of the hash algorithms. HMAC is a keyed hash, and as the bit-width of the key is increased from 0 (no key) to 128 bits, the energy cost varies by a very small amount. SHA and SHA1 are newer hash algorithms, and have more number of steps than MD4 and MD5. Also, SHA and SHA1 are supposed to have better collision resistance, *i.e.*, probability of two inputs mapping to the same hash value, than MD4 and MD5. These benefits of SHA (and SHA1) come at the cost of a slightly higher energy cost than MD4 and MD5.

Table 1: Energy consumption characteristics of hash functions

Algorithm	MD2	MD4	MD5	SHA	SHA1	HMAC
Energy (μJ/B)	4.12	0.52	0.59	0.75	0.76	1.16

5.1.3 Asymmetric Algorithms

Table 2 compares the energy consumed by the three federal information processing standard (FIPS)-approved asymmetric algorithms for generating and verifying signatures in security protocols: RSA, digital signature algorithm (DSA) and elliptic curve digital signature algorithm (ECDSA). Note that we use a 163-bit key for ECC computations, which is proven to be equivalent to a 1024-bit key for RSA [30]. The energy values are reported for the three main steps associated with digital signature algorithms: key generation, signature creation (Sign) and signature verification (Verify). We assume *a priori* generation of the parameters used in the key generation process, as is the case in resource-constrained devices. The results show that ECDSA consumes less energy than DSA. However, ECDSA and RSA digital signature algorithms have complementary energy costs. RSA performs signature verification efficiently, while ECDSA imposes a smaller cost for signature generation. The difference between the energy costs of signature generation and verification in RSA is much greater than in ECDSA. If a mobile client is required to perform frequent signature generation, then it seems preferable to use ECDSA for low-power reasons. On the other hand, if the frequency of signature verification is greater than signature generation, then RSA digital signature algorithm should be employed.

Table 2: Energy cost of digital signature algorithms

Algorithm	Key size bits	Key generation (*mJ*)	Sign (*mJ*)	Verify (*mJ*)
RSA	1024	270.13	546.5	15.97
DSA	1024	293.20	313.6	338.02
ECDSA	163	226.65	134.2	196.23

Asymmetric algorithms are also widely used for performing key exchange. Table 3 compares the standard algorithms used for key exchange, Diffie-Hellman (DH) and its elliptic curve analogue (ECDH). We observe that a 163-bit ECDH consumes much lesser energy than a 1024-bit DH key exchange. The energy cost of the DH algorithm can be drastically reduced by decreasing the size of keys from 1024 bits to 512 bits. However, this benefit does come at the cost of reduced security.

5.1.4 Impact of Cipher Parameters

The energy analysis of cryptographic algorithms is not complete without considering the several modes of operation and tunable parameters associated with each cipher, which can result in algorithmic variants with significantly different energy consumption characteristics. We illustrate this fact by studying the energy consumption

Table 3: Energy cost of key exchange algorithms

Algorithm	Key size (*bits*)	Key generation (*mJ*)	Key exchange (*mJ*)
DH	1024	875.96	1046.5
ECDH	163	276.70	163.5
DH	512	202.56	159.6

characteristics of DES, which can be used in several valid operating modes. The simplest is the ECB. Other modes (cipher block chaining (CBC), cipher-feedback mode (CFB), and output-feedback mode (OFB)) employ a feedback mechanism so that the encryption of a plain-text block is made dependent on the results of encryption of previous plain-text blocks. Due to the feedback mechanism, even for the same key, a given plain-text will not always map to the same cipher-text. In addition to variants due to operating mode considerations, there are variants of the basic DES algorithm such as DES-X. The corresponding energy consumption profile is plotted in Figure 4. The plot shows that the OFB and PCBC modes for DES encryption differ by a factor of nearly 2X in terms of their energy consumption.

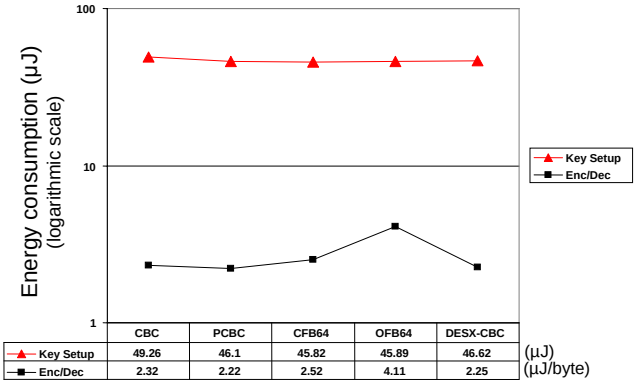


Figure 4: Energy consumption data for various operating mode variants of the symmetric cipher DES

In addition to the operating mode, parameters such as key size have a strong impact on the energy consumption of cryptographic algorithms. Table 4 presents the energy consumption of the AES algorithm for various operating modes and key sizes. From the table, we can see that (i) the energy consumption for the key set-up phase increases with the key size, and (ii) the CFB mode is the most expensive (energy-wise) operating mode for AES encryption, while the ECB mode is the most energy-efficient.

Table 4: Energy costs of AES variants

Key size	Key setup (μJ)	ECB (μJ/B)	CBC (μJ/B)	CFB (μJ/B)	OFB (μJ/B)
128	7.83	1.21	1.62	1.91	1.62
192	7.87	1.42	2.08	2.30	1.83
256	9.92	1.64	2.29	2.31	2.05

5.1.5 Energy Consumption Versus Security Trade-offs

If different security levels can be provided in a cryptographic algorithm, each with associated energy consumption characteristics, a security protocol can be adapted to a level of security commensurate with the current state of the battery of the system. Table 5 identifies different security levels for the RC5 cipher, obtained by changing the number of rounds used in the cipher, for a given key and block size (128 bits). Each entry indicates the data (number of attempts) needed for a successful attack against RC5 using differential and linear cryptanalysis techniques. The symbol $>$ denotes the case when the attacks are deemed impossible even theoretically. We measured the energy consumption of RC5 for various security levels, and the detailed energy versus security trade-off curve is shown in Figure 5. This shows a scheme for lowering the energy consumption by adjusting the security level from high to mid to low, achieved by changing the number of RC5 rounds from 20 to 16 to 8, respectively.

Table 5: Multiple levels of cryptanalytic difficulty in RC5 [31]

Rounds	4	6	8	10	12	14	16
DC-C	2^{19}	2^{42}	2^{58}	2^{83}	2^{106}	2^{123}	$>$
DC-K	2^{74}	2^{86}	2^{94}	2^{106}	2^{118}	$>$	$>$
LC	2^{47}	2^{95}	2^{119}	$>$	$>$	$>$	$>$

** DC-C: Differential cryptanalysis (chosen plain-text), DC-K: Differential cryptanalysis (known plain-text), LC: Linear cryptanalysis (known plain-text)*

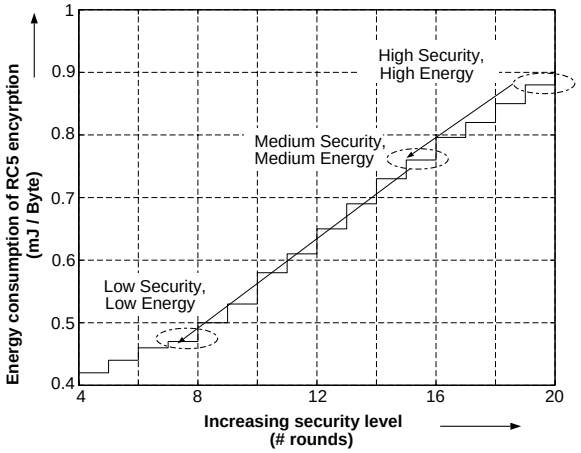


Figure 5: Energy consumption versus security trade-off for RC5 encryption

We also analyzed the effect of key size and number of rounds on the energy cost of key setup for RC5. From Figure 6, we can see the cost of key setup steadily increasing with key size and number of rounds.

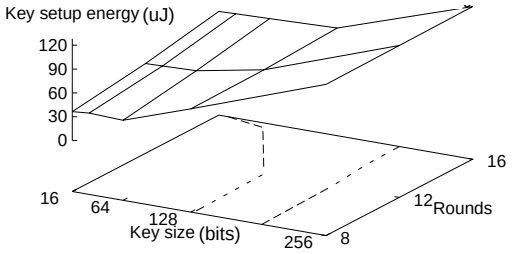


Figure 6: Energy consumption profile for RC5 key setup

5.2 Energy Analysis of the SSL Protocol

Figure 7 shows the typical (client-side) sequence of operations for a secure session that uses the SSL protocol. The first stage involves loading the client certificate from local storage, decrypting it using a symmetric cipher and performing an integrity check. Once the SSL handshake initiates a session, the client and server begin a sequence of exchanges which result in the client-side operations shown in the figure. The operations include (i) *Server authentication*, where the client verifies the digital signature of the trusted certificate authority (CA) on the server certificate through decryption using the public key of the CA, followed by an integrity check, (ii) *Client authentication*, where the client generates a digital signature by hashing some data using MD5 and SHA-1 algorithms, concatenating the digests, and encrypting the result with its private key, and (iii) *Key exchange*, where the client generates a 48-byte pre-master secret (used to generate the secret key for the record stage) and encrypts it with the public key of the server. Once the connection is established, secure transmission of data proceeds through the SSL record stage.

Figure 8 examines the energy consumption contributions from the handshake and record stages of the SSL protocol for various transaction sizes. We can see that for small transaction sizes (upto 256KB), the SSL handshake protocol dominates the overall energy consumption (*e.g.*, 98.9% for 1KB transactions), while for large transactions, the energy consumption of the SSL record protocol is significant (*e.g.*, 80.4% for 1MB transactions). Since both the handshake and

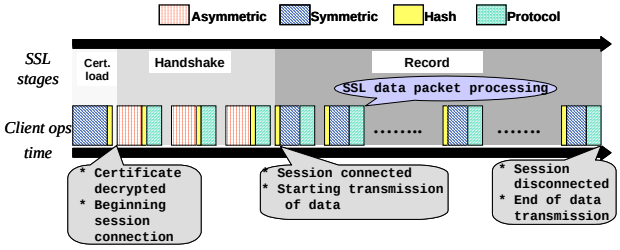


Figure 7: Sequence of client-side operations for an SSL session

record stages of the SSL protocol include various cryptographic operations (asymmetric, symmetric and integrity operations) and non-cryptographic processing (protocol processing), we also present a fine-grained breakup of energy consumption into these components. Figure 9 summarizes our findings for three different transaction sizes (1KB, 100KB, 1MB). From the figure, we can see that the contribution to overall energy consumption due to protocol processing increases with the transaction sizes. The energy consumption of cryptographic processing is dominated by asymmetric ciphers for small transactions and symmetric ciphers for large transactions.

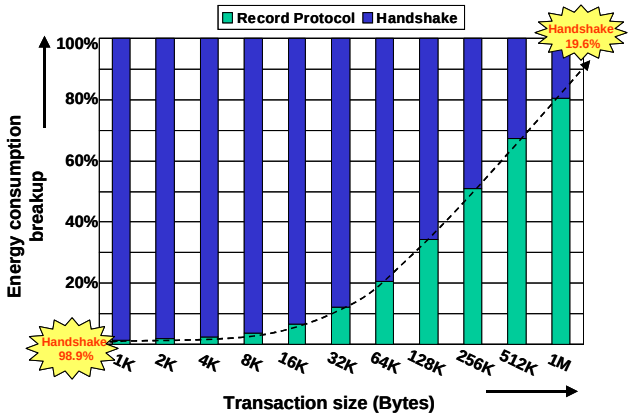


Figure 8: Variation of energy consumption contributions from SSL handshake and record stages with increasing transaction sizes

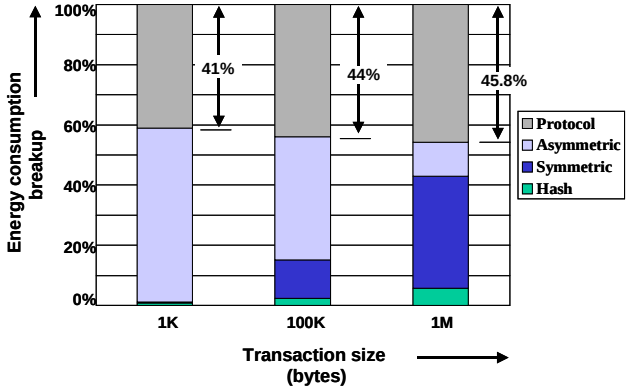


Figure 9: Break-up of SSL energy consumption into cryptographic and non-cryptographic components

Having examined the energy consumption characteristics of the SSL protocol, we now analyze how the energy consumption of the handshake and record stages is affected by various protocol-level services as well cryptographic algorithm parameters. Specifically, we describe how the use of client authentication impacts the energy consumption due to SSL handshake and how the choice of cipher-suite affects the energy consumption of both SSL handshake and record stages.

5.2.1 Impact of Client Authentication and Asymmetric Cipher Choice on SSL Handshake

We investigated the energy cost of the SSL handshake protocol using the RSA algorithm and the ECC algorithms (ECDSA/ECDH) to implement various public-key operations. The results of our analysis are presented in Figure 10. The SSL handshake can be performed between a server and a client with or without client authentication. In the case of handshake without client authentication, the following operations are performed by the client and the server:

- **RSA-based handshake**: The client performs two RSA public key operations (verify and encrypt), and the server performs an RSA private key operation (decrypt).
- **ECC-based handshake**: The client performs verification using ECDSA, and a ECDH operation is performed to compute the shared secret. The server performs an ECDH operation to calculate the shared secret.

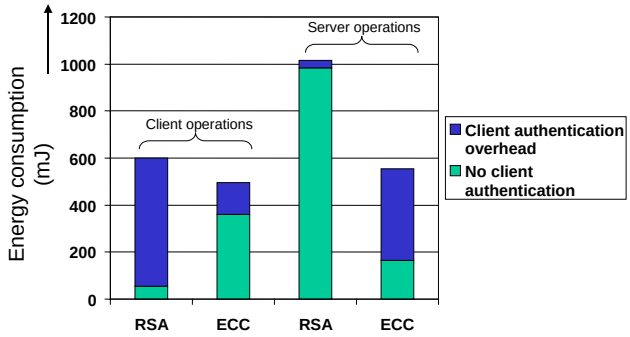


Figure 10: Energy consumption for client and server operations in SSL handshake under the presence or absence of client authentication

If client authentication is required, some extra operations need to be performed. The extra operations to be performed by the client and the server are:

- **RSA-based handshake**: The client performs an RSA private key operation (sign). The server performs two extra RSA public key operations (verify).
- **ECC-based handshake**: The client performs an additional signing operation using ECDSA, and the server performs two extra verification operations using ECDSA.

Figure 10 shows the energy consumed by the SSL handshake process using RSA or ECC algorithms for the handheld functioning as a client or a server. Though the handheld typically behaves as the client in a majority of transactions, it may sometimes be required to play the part of the server. In order to investigate this scenario, we allowed the handheld to perform the server operations for collecting the corresponding energy data. Energy data were also collected for studying the impact of client authentication in all the cases. With respect to the client energy cost, we can see from the figure that RSA-based handshake is much more efficient than ECC-based handshake, when there is no client authentication in the SSL handshake stage. However, in the presence of client authentication in SSL handshake, ECC-based handshake consumes less energy than RSA-based handshake. Thus, depending on whether client authentication is performed or not, either RSA-based handshake or ECC-based handshake should be chosen by the client for optimizing its energy consumption. In general, we believe that various protocol-level parameters have interdependent effects on energy, leading to many interesting trade-offs.

5.2.2 Impact of Cipher-suite Choice on SSL Energy Consumption

The energy cost of the SSL record stage is mainly determined by the amount of bulk data that is transmitted. Analysis of the cipher suites shows that careful choices of cryptographic algorithms need to be made, in order to optimize energy during the record stage. Consider the following two cipher suites, ECC-BLOWFISH-SHA1 and ECC-AES-SHA. A cursory examination would conclude that the second cipher suite is more energy-efficient, given the very high cost of key setup in BLOWFISH. However, Figure 11 shows that if the

amount of data transacted is greater than 7.9 KB, then, in fact, the first cipher suite is more efficient. This is because the cost of key setup in BLOWFISH is gradually amortized, and the advantages of BLOWFISH come into play.

Figure 11 illustrates the energy consumption of two cipher suites, RSA-RC5-SHA and ECC-3DES-SHA. The public-key algorithm (RSA or ECC) is used in the SSL handshake stage and the symmetric-key algorithm (RC5 or 3DES) is used for bulk encryption in the SSL record stage. The figure shows that for data sizes smaller than 21 KB, ECC-3DES-SHA is more energy-efficient because ECC is simpler than RSA (and asymmetric energy consumption dominates that of small data transactions). However, for transactions where there are significant bulk data (greater than 21 KB) to encrypt, RSA-RC5-SHA consumes less energy, because for large data transfers energy consumption of symmetric ciphers dominates the total energy spent, and RC5 is much simpler than 3DES. This shows that a judicious choice of cryptographic algorithms can greatly reduce the amount of energy consumed.

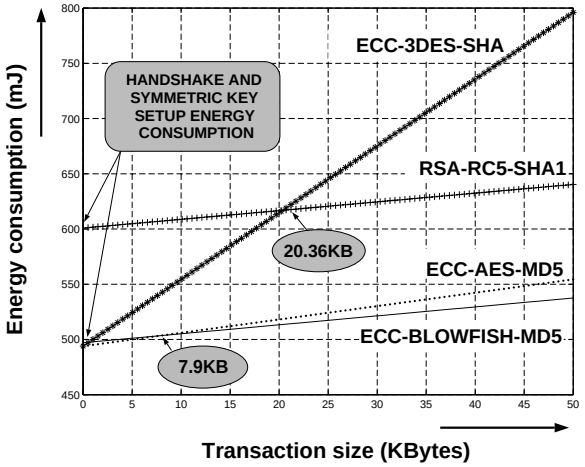


Figure 11: The impact of cipher suite selection on energy consumption during the SSL handshake and record stages

5.2.3 Scope for Optimizing SSL

The energy analyses of the SSL protocol and cryptographic algorithms allow us to explore various options for optimizing the energy consumption of the SSL protocol. The SSL handshake protocol can, for example, be optimized depending on whether client authentication is performed or not, by choosing ECC algorithm in the former case, and RSA algorithm in the latter case. Usually, applications which require a high degree of security need client authentication. In case of applications, where security requirements are not stringent, further energy savings can be obtained by switching to smaller keys. Energy savings can be obtained in the SSL record protocol, by choosing a symmetric algorithm depending on the size of the data to be transacted, such that the overall energy consumption is reduced. In order to account for all the factors on which the energy consumption of the SSL protocol depends, we propose the formulation of an energy cost function, which can be parametrized on a number of factors, such as (i) use of client authentication in handshake, (ii) asymmetric algorithm used in handshake, (iii) key size of the asymmetric algorithm, (iv) symmetric algorithm used in the record stage, (v) hash algorithm used in the record stage, (vi) size of the data to be transmitted, *etc*.

The cost function can be used to decide the best performing among possible alternatives, depending on the input conditions. Such high-level macro-models are the subject of future work, and would allow static, as well as dynamic, optimization of the SSL protocol for energy efficiency.

6. CONCLUSIONS

In this work, we presented a framework for analyzing the energy consumption of security protocols. Asymmetric algorithms have the highest energy cost, symmetric algorithms come second, and at the bottom are the hash algorithms. The energy cost of asymmetric algorithms is very much dependent on the key size, while that of symmetric algorithms is not affected to the same extent by the key size. The cost of symmetric algorithms is made up of two parts, namely, the key set-up (key expansion) and encryption/decryption cost. There is a wide variation in the energy costs within the same family of cryptographic algorithms, *i.e.*, among asymmetric, symmetric and hash algorithms. The energy costs of the handshake and record stages of the SSL protocol vary depending on parameters like functionality desired in the handshake, size of bulk data transacted, *etc*. These conditions reveal the opportunity for making the execution of security protocols dynamic in nature. The protocol execution can be altered depending on the input conditions, such that security of transactions is provided with optimal energy consumption. Future research needs to be done towards this end.

7. REFERENCES

[1] U. S. Department of Commerce, *The Emerging Digital Economy II*. `http://www.esa.doc.gov/508/esa/TheEmergingDigitalEconomyII.htm`, 1999.

[2] W. W. W. Consortium, *The World Wide Web Security FAQ*. `http://www.w3.org/Security/faq/www-security-faq.html`, 1998.

[3] W. Stallings, *Cryptography and Network Security: Principles and Practice*. Prentice Hall, 1998.

[4] B. Schneier, *Applied Cryptography: Protocols, Algorithms and Source Code in C*. John Wiley and Sons, 1996.

[5] *LAN MAN Standards Committee of the IEEE Computer Society*. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification: IEEE standard 802.11, 1990.

[6] *IPSec Working Group*. `http://www.ietf.org/html.charters/ipsec-charter.html`.

[7] *SSL 3.0 Specification*. `http://wp.netscape.com/eng/ssl3/`.

[8] *Wireless Application Protocol 2.0 - Technical White Paper*. WAP Forum (`http://www.wapforum.org/`), Jan. 2002.

[9] *Compaq Ipaq Pocket PC*. `http://h20022.www2.hp.com`, 2002.

[10] D. W. Carman, P. S. Kruus, and B. J. Matt, *Constraints and Approaches for Distributed Sensor Security*. Network Associates Labs Tech. Rep. 00-010, 2000.

[11] J. Goodman, A. Chandrakasan, and A. Dancy, "Design and implementation of a scalable encryption processor with embedded variable DC/DC converter," in *Proc. Design Automation Conf.*, pp. 855–860, June 1999.

[12] Z. Shi and R. Lee, "Bit permutation instructions for accelerating software cryptography," in *Proc. IEEE Int. Conf. Application-specific Systems, Architectures and Processors*, pp. 138–148, July 2000.

[13] J. Burke, J. McDonald, and T. Austin, "Architectural support for fast symmetric-key cryptography," in *Proc. Int. Conf. ASPLOS*, pp. 178–189, Nov. 2000.

[14] N. Potlapally, S. Ravi, A. Raghunathan, and G. Lakshminarayana, "Optimizing public-key encryption for wireless clients," in *Proc. IEEE Int. Conf. Communications*, pp. 1050–1056, May 2002.

[15] S. Ravi, A. Raghunathan, N. Potlapally, and M. Shankaradass, "System design methodologies for wireless security processing platform," in *Proc. Design Automation Conf.*, pp. 777–782, June 2002.

[16] D. Boneh and N. Daswani, "Experimenting with electronic commerce on the PalmPilot," in *Proc. Financial Cryptography*, pp. 1–16, Feb. 1999.

[17] W. Freeman and E. Miller, "An experimental analysis of cryptographic overhead in performance-critical systems," in *Proc. Int. Symp. Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pp. 348–357, Oct. 1999.

[18] S. K. Miller, "Facing the challenges of wireless security," *IEEE Computer*, pp. 46–48, July 2001.

[19] G. Apostolopoulos, V. Peris, P. Pradhan, and D. Saha, "Securing electronic commerce: Reducing the SSL overhead," *IEEE Network*, pp. 8–16, July 2000.

[20] D. S. Wong, H. H. Fuentes, and A. H. Chan, "The performance measurement of cryptographic primitives on Palm devices," in *Proc. Annual Computer Security Applications Conf.*, pp. 92–101, Dec. 2001.

[21] S. Ravi, A. Raghunathan, and N. Potlapally, "Securing wireless data: System architecture challenges," in *Proc. Int. Symp. System Synthesis*, pp. 195–200, Oct 2002.

[22] A. Hodjat and I. Verbauwhede, "The energy cost of secrets in ad-hoc networks," in *Proc. IEEE CAS Wkshp. Wireless Communication and Networking*, Sept. 2002.

[23] M. Jakobsson and D. Pointcheval, "Mutual authentication for low-power mobile devices," in *Proc. Financial Cryptography*, pp. 178–195, Feb. 2001.

[24] D. S. Wong and A. H. Chan, "Mutual authentication and key exchange for low power wireless communications," in *Proc. IEEE MILCOM Conf.*, pp. 39–43, Oct. 2001.

[25] Y. W. Law, S. Dulman, S. Etalle, and P. J. M. Havinga, *Assessing Security-Critical Energy-Efficient Sensor Networks*. Univ. of Twente, The Netherlands, Tech. Rep. TR-CTIT-02-18, July 2002.

[26] R. Karri and P. Mishra, "Minimizing energy consumption of secure wireless session with QoS constraints," in *Proc. Int. Conf. Communications*, pp. 2053–2057, May 2002.

[27] *OpenSSL Project*. `http://www.openssl.org`.

[28] *Familiar Project*. `http://familiar.handhelds.org`.

[29] *National Instruments Corp*. `http://www.ni.com`.

[30] V. Gupta, S. Gupta, S. Chang, and D. Stebila, "Performance analysis of elliptic curve cryptography for SSL," in *Proc. ACM Wkshp. Wireless Security*, pp. 87–94, Sept. 2002.

[31] Y. L. Yin, "The RC5 encryption algorithm: Two years on," *RSA Laboratories' Cryptobytes*, vol. winter, pp. 14–15, 1997.