

# MINIMIZING THE SECURE WIRELESS SESSION ENERGY

Ramesh Karri

Piyush Mishra

[ramesh@india.poly.edu](mailto:ramesh@india.poly.edu)

[pmishr01@utopia.poly.edu](mailto:pmishr01@utopia.poly.edu)

718-260-3596

718-260-4011

Department of Electrical and Computer Engineering

Polytechnic University, Brooklyn

6 Metrotech Center, NY, US 11201

*Abstract: In this paper we identified the various sources of energy consumption during the setup, operation and tear down of a secure wireless session. Our analysis showed that the data transfers during a secure wireless transaction, the number and size of messages exchanged during secure session establishment and the cryptographic computations used for data authentication and privacy during secure data transactions in that order are the main sources of energy consumption during a secure wireless session. We developed techniques based on compression, protocol optimization and hardware acceleration to reduce the energy consumed by a secure session. A mobile test bed was developed to verify our energy management schemes and to study the energy consumption vs. security trade-offs. Using our proposed schemes we were able to reduce the session establishment energy by more than 85% and the secure data transaction energy by more than 40% during data transmission and by more than 65% during data reception*

*Keywords: Mobile, Security, Wireless, Energy-efficient, WTLS*

## 1. Introduction

According to the industry projections, mobile computing and communication device market is poised to overtake the desktop computing market [1]. The widespread adoption of Internet combined with the “anytime-anywhere” access of mobile devices is driving a huge growth in mobile e-commerce applications such as on-line shopping, stock trading, web-based banking and electronic bill payment. Such confidential transactions over wireless public networks demand end-to-end secure connections to ensure data authentication, privacy and integrity [2].

Energy consumed by such secure wireless sessions on mobile devices is very significant. There are two main sources of energy consumption during a secure wireless transaction: (i) cryptographic computations used to establish the secure session and to support encryption and authentication during data transaction, and (ii) message exchanges during secure session establishment and data transfers during secure data transactions. We considered a

Symbol PPT2800™ Pocket PC device running Windows CE™ 3.0 operating system and equipped with an 11 Mbps Spectrum24™ wireless LAN adapter card as the mobile test bed<sup>1</sup> to measure the energy consumed by a secure wireless session while transmitting 64 KB data. Session negotiation used Diffie-Hellman key exchange protocol [29] and data transaction used 3DES encryption [32] and SHA-256 message authentication code [33].

Figure 1 shows that message exchanges consume more than 90% of the system energy during session negotiation while during data transaction idle system consumes 44% of the system energy, followed by data transmission that consumes 35% and cryptographic computations that consume 21% of the system energy respectively.

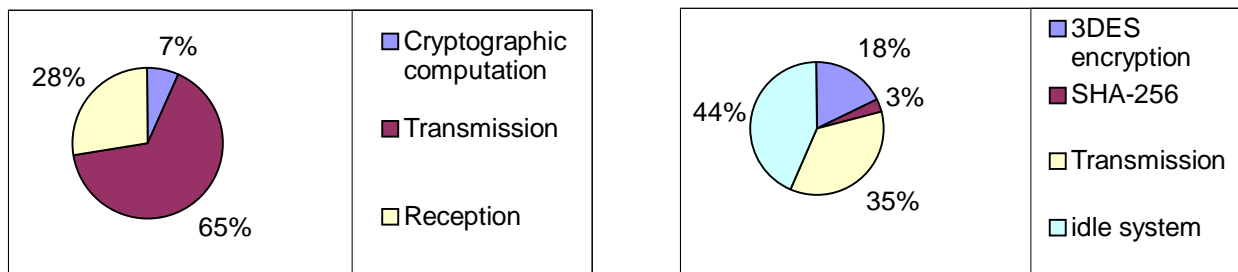


Figure 1: Energy consumed by a secure wireless session during (a) session negotiation and (b) data transaction while transmitting 64 KB data

There has been substantial research in the field of circuit, logic and architecture level design techniques for low power [4, 5], instruction level power models of processors [15], estimation and minimization of software energy [16], and energy-efficient software compilation techniques [17]. At the application level, adaptive applications have been developed to reduce energy [13,14]. Techniques to minimize the energy consumed by a communication unit include modulating the energy used by the mobile transmitter during active communication [18]. All of these techniques reduce the energy consumed when the system is active. Operating system level energy management techniques include switching between low power processor modes [6,7], adaptive and controlled spin downs of hard disks [8,9,10] and control of the device display. Energy aware network protocols either reduce energy-expensive retransmission of lost messages or suspend device operation during idle periods [19] or transition between different modes of operation [20,11,12]. All of these techniques reduce the energy consumed by a system by increasing the idle periods.

<sup>1</sup> Refer to section 3.1 for a detailed description of the mobile test bed.

In this paper we will measure the energy consumed by various components of a secure wireless session, present techniques to minimize their energy consumption and investigate energy vs. security trade-offs. We will use Wireless Transport Layer Security (WTLS) [3] of the Wireless Application Protocol (WAP) suite as the example protocol. WAP is a set of protocols in transport, security, transaction, session and application layer as shown in Figure 2 that enables creation of advanced mobile services.

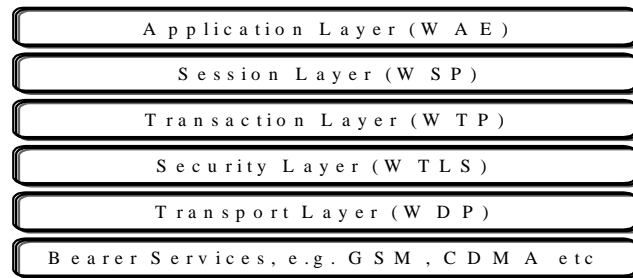


Figure 2: Wireless Application Protocol architecture

A secure wireless transaction between a web server and a mobile device is a two-step activity using WAP. While the secure transmission between the web server and WAP gateway uses Secure Session Layer (SSL) protocol, the wireless transaction between the WAP gateway and the mobile device is secured using WTLS protocol. WTLS operates above the transport protocol layer and provides the higher level layers with a secure transport service interface.

In section 2 we will describe the setup, operation and tear down of a secure wireless session using WTLS protocol. In section 3 we will describe the mobile test bed and summarize the energy consumed during the setup, operation and tear down of a secure wireless transaction. In section 4 we will present techniques to reduce the energy consumed by a mobile device during a secure wireless session. Finally, in section 5 we will summarize the results of this study.

## 2. Wireless Transport Layer Security

A handshake protocol is used to establish a secure session between the client and the server before carrying out secure data transactions. The client initiates the handshake by sending a **client\_hello** message to the server. This message contains session id, key refresh rate, private key encryption algorithm and its mode of operation, message authentication code (MAC) algorithm and a random number used to generate the encryption and MAC keys from the master\_secret. The message also identifies the master\_secret exchange protocol and its parameters. For

example, if the Diffie-Hellman master\_secret exchange protocol is used [29], the client sends the generator (g) and the prime modulus (n) used to generate the master\_secret<sup>2</sup>.

The server responds with a **server\_hello** message accepting the security association proposed by the client, another random number (to be used during encryption key and MAC key generation), the **server\_certificate** for authenticating itself to the client, its share  $X = (g^x \text{ mod } n)$  of the master\_secret, where x is a large random number (**server\_key\_exchange**) and a **certificate\_request** from the client.

The client replies with its **client\_certificate**, its share  $Y = (g^y \text{ mod } n)$  of the master\_secret (**client\_key\_exchange**), where y is a large random number and a **certificate\_verify** message to verify its certificate.

Finally, the client and the server exchange **change\_cipher\_spec** message to activate the session with the negotiated security association (the encryption algorithm and its mode of operation, the MAC algorithm, the session id and the key refresh rate) and a **finished** message to indicate successful key exchange. The client and the server independently compute the master\_secret  $g^{xy} \text{ mod } n$  as  $(Y^x \text{ mod } n)$  and  $(X^y \text{ mod } n)$  respectively and generate the secret keys for encryption and MAC from the master\_secret, the random numbers exchanged and the message sequence number. Figure 3 summarizes the handshake protocol.

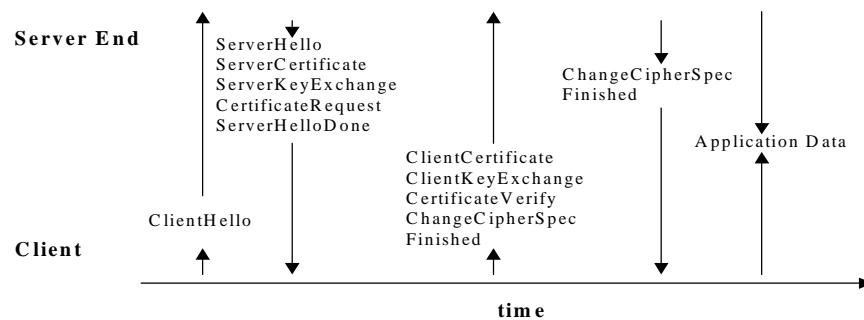


Figure 3: Messages exchanged during the handshake protocol

After successfully establishing the secure session WTLS (either at the client or at the server) accepts plain text messages, computes the MAC, encrypts the data and transmits it. At the other end, the received data is decrypted and verified. The security of a session is enhanced by periodically refreshing the encryption and MAC keys; for a key\_refresh\_rate of n, the client and the server generate new encryption and MAC secret keys after exchanging  $2^n$  data blocks. Either side can terminate the session by sending a **closing** message.

<sup>2</sup> In this paper we use the Diffie-Hellman master\_secret exchange protocol. RSA and EC protocols can be analyzed in a similar fashion.

### 3. Energy consumption of a secure wireless session

#### 3.1 Test-bed for energy measurements

The mobile test bed shown in

Figure 4 consists of a PPT2800™ Pocket PC device equipped with an 11 Mbps Spectrum24™ wireless LAN Adapter card (IEEE 802.11b), both from Symbol Technologies Inc. [31, 21], and a Wildcard™ FPGA board from Annapolis Micro Systems® [26]. The Pocket PC is running WindowsCE™ operating system on a 32-bit, 206 MHz StrongArm™ SA-1110 processor with 16 MB flash ROM, 16 MB RAM, 16 KB instruction cache and 8 KB data cache. The WLAN card is operating in polling mode P1 [21]. The Wildcard™ board supporting a Xilinx® Virtex™ XCV300E field programmable gate array (FPGA) with 400,000 system gates and 130,000 block RAM bits is used to accelerate the cryptographic computations.

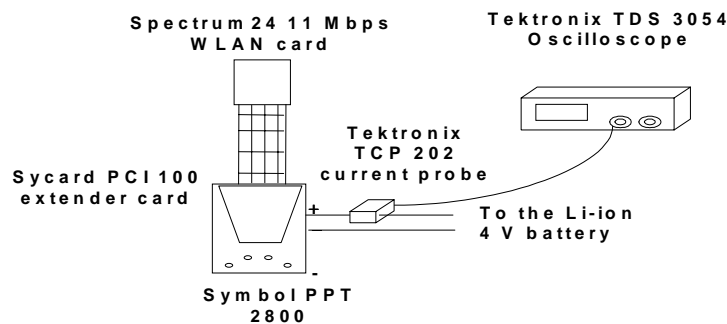


Figure 4: Mobile test bed for performance and energy measurements

We measure the current drawn by an application executing on the mobile test bed using a Tektronix TCP202 current probe and a Tektronix TDS 3054 oscilloscope [30]. Voltage is held constant by repeatedly charging the battery. In order to ensure consistency and accuracy of our results, we averaged each of our results over several thousand iterations for each application with the data residing in the main memory and data and instruction caches enabled.

#### 3.2 Sources of energy consumption

Energy consumed while establishing a secure session is the sum of the energy consumed during message exchanges, energy consumed to compute the master\_secret, the encryption keys and the MAC keys, and the energy consumed to sign and verify the certificates. Energy consumed during a secure wireless data transaction is the sum of energy consumed by data authentication, data encryption and data transmission. Energy consumed to terminate a secure session is negligible.

### 3.2.1 The wireless transceiver subsystem

According to the specifications of the 11 Mbps Spectrum24<sup>®</sup> LA-4121 wireless LAN card operating at 5 V, the current consumption depends on its mode of operation [21], as shown in Table 1. The WLAN card support a continuous access mode (CAM) and five polling modes, P1 to P5. Transmission energy is the product of the power consumed in the transmit mode and the time to transmit the data.

	CAM		Polling	
	Current (mA)	Power (W)	Current (mA)	Power (W)
Sleep	-	-	10	0.05
Idle	170	0.85	30	0.15
Receive	190	0.95	190	0.95
Transmit	410	2.05	410	2.05

Table 1: Power consumed by 11 Mbps Spectrum24<sup>®</sup> LA-4121 WLAN card

### 3.2.2 Message Authentication Code

We used SHA-256, a variation of a 256-bit symmetric block encryption algorithm, as the message authentication code (MAC) [33]. SHA-256 encrypts the intermediate hash values using the message blocks as the keys. Figure 5 shows the energy consumed by optimized C implementation of SHA-256 for different data sizes.

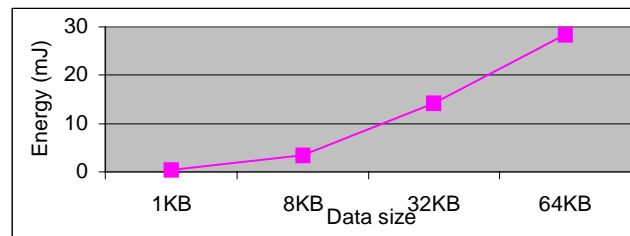


Figure 5: Energy consumed by SHA-256 MAC

### 3.2.3 Encryption

Data encryption standard (DES) and 3DES are private key encryption algorithms that have been widely used for data privacy for more than 20 years now [32]. Recently National Institute of Standards and Technologies (NIST) selected Rijndael as the Advanced Encryption Standard (AES) to replace DES in systems with higher security requirements [22]. AES supports multiple user key lengths (128, 192, or 256 bits) and multiple data block sizes (128, 192 or 256 bits) [23]. Security level and the energy consumption of a private key encryption algorithm increase with the number of encryption rounds and the length of the user key. Table 2 summarizes the energy consumed by the round key generation and encryption of optimized C implementations of AES (128, 192, 256 - bit key). Energy consumed by AES key generation increases with the key size at a rate greater than the energy

consumed by its encryption due to an increase in the complexity of key generation and an increase in the number of round keys.

AES key generation			
	128-bit key	192-bit key	256-bit key
Energy ( $\mu$ J)	10.44	13.70	17.44
Time ( $\mu$ S)	7.48	9.82	12.47

(a)

AES encryption			
	128-bit key	192-bit key	256-bit key
Energy/bit ( $\mu$ J)	0.067	0.07	0.075
Throughput (Mbps)	25.96	24.58	24.1

(b)

Table 2: Energy consumed by AES (a) key-generation and (b) encryption

### 3.3 Energy consumed by a secure wireless session

#### 3.3.1 Negotiating a security association

Energy consumed by D-H based handshake protocol message exchanges and associated cryptographic computations are summarized in Table 3. The energy values in bold corresponds to the client. Energy consumed by the handshake protocol depends on the master\_secret exchange protocol, the level of security (size of certificates, size of the master\_secret, size of the encryption and MAC keys) and the number and size of messages exchanged.

A client establishing a secure session consumes approximately 1062 milli Joules, 7% of which is consumed by the cryptographic computations and 93% is consumed by the message exchanges. When encryption and MAC keys are periodically refreshed only 12.33 milli Joules are expended (at the client and at the server),  $\frac{1}{85}$

Joules for establishing a new secure session.

Messages exchanged		Energy consumed (mJ)		
		Cryptographic computations	Message exchanges	
			Transmit	Receive
M1	Client_hello	<b>0.01</b>	<b>15.7</b>	6.8
M2	Server_hello	0.01	2.6	<b>1.3</b>
	Server_certificate	-	337.2	<b>145</b>
	Server_key_exchange	60.21	337.2	<b>145</b>
	Request_certificate	-	5.3	<b>2.3</b>
M3	Client_certificate	-	<b>337.2</b>	145
	Client_key_exchange	<b>60.74</b>	<b>337.2</b>	145
	Change_cipher_spec,finished	-	<b>2.6</b>	1.3
M4	Change_cipher_spec,finished	-	2.6	<b>1.3</b>
	Encryption and MAC secret key computation at the client	<b>12.33</b>	-	-
	Certificate verify at client	<b>1.21</b>	-	-
	Certificate verify at server	1.21	-	-
	Encryption and MAC secret key computation at the server	12.33	-	-
	<b>CLIENT</b>	<b>74.3</b>	<b>692.7</b>	<b>295</b>
	SERVER	73.8	685	298

Table 3: Energy consumed by the handshake protocol using Diffie-Hellman key exchange protocol

### 3.3.2 Secure data transaction

Table 4 summarizes the energy consumed during secure wireless data transmission assuming the following security association: 128-bit key AES encryption, SHA-256 MAC, key refresh rate that entails re-computing the encryption and MAC keys every 128 KB of data, and session refresh every 2 MB of data.

Mobile-to-server transaction energy (mJ)		
	2560 KB data	8 KB data
SHA-256 MAC	1130	3.53
AES-128 encryption	1372	4.29
Transmission	13480	42.13
Key-refresh	245	-
Idle system	16604	51.87
Total	32831	101.82

Table 4: Energy consumed by secure wireless data transmission

Energy consumed by the idle system is inversely proportional to the sustained throughput of the system, where sustained throughput of a system is determined by a variety of factors, including the network conditions, efficiency of the wireless protocols and the requirements of the application. Energy consumed by data transmission and reception increases linearly with the input data size and the energy consumed by the cryptographic computations increases linearly with both data size and security level. For example, while transferring an 8 KB of data does not entail any key refresh overhead, transferring 2560 KB of data entails 19 key refreshes, consuming 245 milli Joules at the client and at the server. Refreshing the secret keys entails generating new encryption and MAC keys and generating round keys from the new encryption key.

Similarly, large encryption key and higher number of encryption rounds also improve the level of security at the cost of extra energy consumption and performance degradation. Cryptographic computations (key refresh, data encryption and MAC authentication) consume 7.7% of the total energy for transferring 8 KB data, and this increases to 8.4% of the total energy for transferring 2560 KB data.

## 4. Reducing the system energy consumption

Securely transferring 2560 KB of data consumes 33893 milli Joules. The handshake protocol consumes 3% of this energy, data transfers consume 40%, data encryption, authentication and key refreshes consume 8%, and the idle system consumes 49%. On the other hand, securely transferring 8 KB data consumes 1164 milli Joules, of which 91.2% is consumed by the handshake protocol, 3.6% by data transfers, 0.7% by data encryption and authentication and 4.5% by the idle system. Therefore, for small data transactions energy consumed by the handshake protocol is most dominant, but as the data size increases energy consumed by data transmission,

encryption and authentication become more significant. In this section we will investigate compression, handshake protocol optimization and hardware acceleration of cryptographic computations to target these sources of energy consumption.

#### 4.1 Impact of compression

Compressing the data before encryption and transmission and decompressing it after reception and decryption reduces the energy consumed by a secure wireless transaction if the energy savings due to the reduced data size are more than the extra energy consumed by compression and decompression. Similarly, compressing the messages exchanged during session negotiation reduces the energy consumed by the handshake protocol.

To study the impact of compression we used optimized C implementation of DEFLATE loss-less data compression algorithm [24]. Compression level, history window size and memory-level are the three main parameters that affect the energy consumed by DEFLATE compression. Table 5 summarizes the energy consumed by DEFLATE while compressing 1KB, 8 KB and 64 KB size benchmarks from Calgary corpus [25].

Data size		Compression level =9 Memory level =9	Compression level =1 Memory level =9	Compression level =9 Memory level =1	Compression level =5 Memory level =5
64 KB	Energy (mJ)	1004.15	132.59	2785.15	395.79
	Compression ratio	4.482	3.5884	4.0042	4.3256
8 KB	Energy (mJ)	55.71	46.11	65.52	31.69
	Compression ratio	3.5085	3.1059	3.1035	3.4782
1 KB	Energy (mJ)	26.62	34.57	14.24	14.76
	Compression ratio	2.8679	2.7861	2.4805	2.8254

Table 5: Energy consumed by DEFLATE compression

From Table 5 it can be seen that matching the compression block size to the data cache size saves significant energy (8 KB for our mobile test bed). Further, progressively increasing either the compression level or the memory level results in a proportional increase in the energy consumed without a corresponding increase in the compression ratio. Progressively increasing the size of the history window yields a proportional increase in the compression ratio without a corresponding increase in the energy consumed. We found that medium compression level (level 5), a medium memory level (level 5) and a maximum history window size (15 bits) combination achieves a compression ratio close to the best, while consuming significantly less energy. In this paper we will use DEFLATE with these parameters.

We derived an expression for the energy consumed by DEFLATE compression as a function of data size and the selected parameters using the measure values and the MATLAB simulation:  $12.9 + 1.83 D + 0.065 D^2$  milli

Joules, where D denotes the data size in kilobytes. Energy consumed by decompression is approximately one-tenth of the energy consumed by compression since decoding is very simple and fast.

Therefore, while transmitting data the compression block size should be matched to the data cache size of the device and while receiving data large compression block size (larger the better) should be used to reduce the client energy. Such an asymmetric compression arrangement can be agreed upon during the secure session negotiation.

#### 4.1.1 Compressing the handshake protocol messages

Generation and exchange of the client and server certificates (server\_certificate, client\_certificate) and the master\_secret (client\_key\_exchange, server\_key\_exchange) consumes more than 90% of the handshake protocol energy.

	Energy consumed by handshake protocol (mJ)	
	Uncompressed	Compressed
Transmit	692.7	199.1
Receive	295	84.8
Cryptographic computations	74.3	74.3
Compress/Decompress	-	568.7
Total	1062	926.9
Energy saving factor		1.15 ×

Table 6: Energy consumed by the handshake protocol

Since decompression energy is very small compared to energy for compression, it is always energy-efficient for a mobile device to receive compressed data. Hence, if the WAP gateway compresses all messages following the **server\_hello** message, energy consumed by the client during handshake protocol can be reduced. Table 6 shows a 1.15× reduction in the energy consumed by client during the handshake by compressing the 64-Kbit certificates and the 64-Kbit master\_secret messages exchanged by the client and the server.

#### 4.1.2 Compression of secure wireless transactions

Energy consumed during transmission and reception of uncompressed and compressed, Rijndael encrypted 2560 KB and 8 KB data are summarized in Table 7. Energy consumed by data transmission is 10× the energy consumed by AES encryption which in turn is 0.15× the energy consumed by DEFLATE data compression for 8 KB compression block size<sup>3</sup>.

Besides reducing the size of the data to be encrypted and transmitted, compression also reduces the energy consumed by the key refreshes. For example, if encryption and MAC keys are refreshed after exchanging 128 KB of data, an uncompressed 2560 KB data transfer requires 19 key refreshes and consumes 495 milli Joules, while the

compressed 736 KB data (=2560 KB÷compression ratio of 3.4782) requires only 5 key refreshes. Therefore, data compression reduces (a) the transmission, reception, encryption and decryption energy during a secure data transaction (b) the number of key refreshes required and the corresponding energy, and (c) the energy consumed by the idle system.

	Energy consumed (mJ)	
	2560 KB data	
	Uncompressed	Compressed
Compress	-	10141
AES-128 encrypt	1372	394.6
AES-128 decrypt		
SHA-256 sign	1130	324.9
SHA-256 verify		
Transmit	13480	3876
Receive	5803	1668
Decompress	-	1014
Key-refresh	245	64.45
Idle system	16604	4773
Total transmit energy	32831	19574
Transmit energy saving factor	-	1.68 ×
Total receive energy	25154	8239
Receive energy saving factor	-	3.05 ×

Table 7: Energy consumed during a secure wireless transaction

## 4.2 Optimizing the handshake protocol

The handshake protocol discussed until now consumes significant energy, with the generation and exchange of certificates and master\_secret material consuming > 90% of the total client energy during handshake.

As shown in section 4.1.1, compressing the client and the server certificates reduces the client energy consumption by 12.7%. Client energy can also be reduced by 32% by modifying the handshake protocol such that the server looks up the client's certificate from its own source (**handshake variant 1**). Embedding the client's share of master\_secret in its certificate can further reduce its energy by another 41%. When establishing a new session, a client-server pair can exchange new client and server random numbers and combine these with previously negotiated security association (**handshake variant 2**). Finally, implanting the master secret in the server and mobile device eliminates the energy consumed by master\_secret exchange messages (**handshake variant 3**) and reduces the energy consumed by the client by 97%. Table 8 shows the energy consumed by these various handshake protocols.

<sup>3</sup> We used the energy consumed by wireless transmission from Section 3.2.1, energy consumed by keyed-MAC from Section 3.2.2, energy consumed by 128-bit key Rijndael encryption from Section 3.2.3 and energy consumed by DEFLATE compression from Section 4.1

	Handshake protocol energy consumption (mJ)			
	Basic	variant 1	variant 2	variant 3
Transmit	692.7	101	18.3	18.3
Receive	295	84	2.6	2.6
Master_secret computation + authentication	62	62	62	-
Key generation	12.3	12.3	12.3	12.3
Total	1062	259.3	75.2	33.2
Energy saving factor		4.1 ×	14.12 ×	32 ×

Table 8: Energy consumed by various handshake protocols

We propose an **adaptive handshake** protocol that uses **handshake variant 1** for establishing a new session and **handshake variant 2** to refresh the security association by exchanging new client and server random numbers if the session lasts beyond a certain number of messages determined by the security requirements of the session. Allowing the client and the server to use compression immediately following the **server\_hello** message minimizes the energy consumed during the **handshake variant 1** protocol. Since **handshake variant 3** is inflexible (master\_secret is implanted permanently into the server and the client) and vulnerable to physical and side-channel attacks to recover the implanted master\_secret, we did not include it in the adaptive handshake protocol.

### 4.3 Impact of FPGA implementation of AES (Rijndael)

Implementing cryptographic computations in hardware improves the throughput and reduces the energy consumption. Since hardware resources are limited in small mobile devices, all cryptographic operations cannot be implemented in hardware. We implemented the Rijndael encryption on Wildcard™ board. Results of the FPGA implementation are summarized in Table 9 (a). Area is computed as the number of Virtex slices used by the design where a Virtex slice contains two look-up tables and one look-up table can implement four input-one output logic functions. Throughput of encryption defined as the number of bits encrypted per second is calculated as

$$\frac{\text{\# of bits encrypted}}{\text{\# of clock cycles for encryption} \times \text{clock period}}$$

We used the Xilinx® FPGA power estimation tool Xpower™ to estimate the energy consumed by our FPGA implementation of Rijndael encryption. Table 9 shows that the FPGA implementation improves the throughput and reduces the energy consumed by Rijndael encryption. Since hardware resources are limited, a trade-off can be made between the area overhead, performance and energy benefits. The mix-column operation in Rijndael is the performance bottleneck in its software implementation, consuming approximately 10× more time and energy than any other operation. Therefore, implementing the mix-column operation in FPGA and the rest of Rijndael in software improves performance and reduces energy while consuming limited FPGA resources (~250 Virtex slices).

	Rijndael-128 Encryption	
	Software	Hardware
Area (# of slices)	-	3973
Throughput (Mbps)	25.963	124.8
Energy/bit ( $\mu$ J)	0.067	0.0012

(a)

	Energy consumed by secure wireless data transaction of 2560 KB data (mJ)	
	Uncompressed	Compressed
Total transmit (software encryption/decryption)	32831	19574
Total transmit (hardware encryption/decryption)	31484	19186
Transmit energy saving factor	1.04 $\times$	1.02 $\times$
Total receive (software encryption/decryption)	25154	8239
Total receive (hardware encryption/decryption)	23807	7851
Receive energy saving factor	1.06 $\times$	1.05 $\times$

(b)

Table 9: (a) Software vs. Hardware implementation of Rijndael encryption and (b) the impact of hardware implementation of encryption on the total session energy

## 5. Summary

WTLS offers three levels of security: renewing session after exchanging certain amount of data, refreshing the encryption and MAC keys after  $2^n$  data transactions and changing the key length and number of rounds of encryption. Our proposed schemes affect WTLS secure sessions at all three security levels, as shown in Table 10.

Security level	Scheme
Session renewal	Protocol optimization
Key refresh	Compression
Key length, encryption rounds	Hardware implementation of encryption

Table 10: Improving the session energy consumption at various security levels

Let us examine the impact of these schemes on the energy consumption during a secure wireless session. Consider a client supporting AES-128 encryption, SHA-256 MAC, key refresh every 128 KB of data and session renewal every 2 MB of data while transmitting 20 MB data over an 11 Mbps wireless channel. We assume a compression ratio of 3.4782 corresponding to the data block size of 8 KB.

Table 11 compares the energy consumed by an un-optimized scheme using **basic handshake** protocol, supporting encryption in software and no compression scheme with the energy consumed by an optimized scheme using **adaptive handshake** protocol, supporting encryption in hardware and DEFLATE data compression. Figures in braces ‘()’ refer to the corresponding frequency of operation. For example, Un-optimized scheme requires 9 **basic handshakes** for the entire data transaction. As shown in table 11, optimized scheme results in more than 1.5 $\times$  energy savings during data transmission and more than 3 $\times$  savings during data reception. Since energy saved by

encrypting the compressed data in hardware is not significant, adopting a performance, energy consumption and area tradeoff based software-hardware co-design approach, as discussed in section 4.3, will be more beneficial.

	Un-optimized secure session		Optimized secure session	
	Session parameters	Energy (mJ)	Session parameters	Energy (mJ)
Handshake	Basic (9)	9558	Version 1 (1), Version 2 (1)	335
Compress	-	-	DEFLATE	79225
SHA-256 sign	Software	8828	Hardware	2538
SHA-256 verify				
AES-128 encrypt	Software	10719	Hardware	3082
AES-128 decrypt				
Transmit	-	105313	-	30278
Receive	-	45336	-	13034
Decompress	-	-	DEFLATE	7988
Key refresh	(150)	1934	(43)	554
Idle system	-	129719	-	37295
Total transmit	-	266071	-	153307
Transmit energy saving factor				1.74 ×
Total receive	-	206094	-	64826
Receive energy saving factor				3.18 ×

Table 11: Energy savings for the client due to secure session optimization

## 6. References

1. J. A. Senn, "The emergence of M-Commerce", IEEE Computer, December 2000, pp. 148-150.
2. D. Clark, "Encryption advances to meet Internet challenges", IEEE Computer online magazine, December 2000.  
<http://www.computer.org/computer/articles/August/technews800.htm>
3. Wireless Application Protocol: Wireless Transport Layer Security Specifications, Feb 2000.  
<http://www.wapforum.org>
4. G. Lakshminarayana, A. Raghunathan, K. S. Khouri, N. K. Jha, S. Dey, "Common case computation: A high level technique for power and performance optimization", Proceedings, ACM/IEEE DAC, June 1999.
5. T. D. Givargis, J. Henkel, F. Vahid, "Interface and cache power exploration for core based embedded system design", Proceedings, ICCAD, 1999.
6. J. R. Lorch, A. J. Smith, "Reducing processor power consumption by improving processor time management in a single-user operating system", Proceedings, ACM MOBICOM, 1996.
7. K. Govil, E. Chan, H. Wasserman, "Comparing algorithms for dynamic speed-setting of a low-power CPU", Proceedings, ACM MOBICOM, 1995.
8. Adaptive power management for mobile hard drives, IBM, August 1995.  
<http://www.storage.ibm.com/storage/oem/tech/ableback.htm>

9. K. Li, R. Kumpf, P. Horton, T. Anderson. "A quantitative analysis of disk drive power management in portable computers", Proceedings, Winter USENIX, 1994.
10. F. Douglass, P. Krishnan, B. Bershad, "Adaptive disk spin-down policies for mobile computers", Proceedings, USENIX, April 1995, pp. 121-137.
11. R. P. Dick, G. Lakshminarayana, A. Raghunathan, N. K. Jha, "Power analysis of embedded operating system", Proceedings, DAC, June 2000.
12. S. Udani, J. Smith, "The power broker: Intelligent power management for mobile computers", Technical Report MS-CIS-96-12, CS Department, University of Pennsylvania, May 1996.
13. J. Flinn, M. Satyanarayanan, "Energy-aware adaptation for mobile applications", Proceedings, ACM SOSP, 1999, pp. 48-63.
14. C. S. Ellis, "The case for higher level power management", Proceedings, HTOS, 1999.
15. V. Tiwari, S. Malik, A. Wolfe, "Power analysis of embedded software: A first step towards software power minimization", IEEE Transactions on VLSI Systems, December 1994.
16. Y. Li, J. Henkel. "A framework for estimating and minimizing energy dissipation of embedded HW/SW systems", Proceedings, DAC, June 1998, pp. 188-193
17. C.T. Hsieh, M. Pedram, G. Mehta, F. Rastgar, "Profile-Driven Program Synthesis for Evaluation of System Power Dissipation", Proceedings, IEEE DAC, June 1997, pp. 576-581.
18. J. M. Rulnick, N. Bambos, "Mobile power management for maximum battery life in wireless communication networks", Proceedings, IEEE INFOCOM, 1996.
19. R. Kravets, P. Krishnan, "Power management techniques for mobile communication", Proceedings, ACM/IEEE MOBICOM'99.
20. A. Kamerman, L. Monteban, "WaveLAN-II: A high performance wireless LAN for the unlicensed band", Bell Labs Technical Journal, 1997.
21. Spectrum24® High Rate LA 41X1 PC Card.  
<http://www.symbol.com/products/wireless/la41x1.html>
22. <http://csrc.nist.gov/encryption/aes>
23. J. Daemen, V. Rijmen, "AES proposal: Rijndael".  
<http://www.esat.kuleuven.ac.be/~rijmen/rijndael/rijndaeldocV2.zip>

24. DEFLATE Compressed Data Format Specification version 1.3.

<http://www.kblabs.com/lab/lib/rfc/1900/rfc1951.txt.html>

25. T.C. Bell, "Text compression", Prentice Hall, Englewood Cliffs, NJ, 1990

26. Annapolis Micro Systems Wildcard FPGA board

<http://www.annapmicro.com/products.html>

27. [ftp://dspftp.ece.ubc.ca/pub/tmn/qcif\\_source](ftp://dspftp.ece.ubc.ca/pub/tmn/qcif_source)

28. <http://www.ee.ubc.ca/image/>

29. W. Diffie, M. E. Hellman, "New directions in cryptography", IEEE Transactions on Information Theory, v. IT-22, n. 6, November 1976, pp. 644-654

30. <http://www.tektronix.com>

31. Symbol                    PPT2800                    series                    portable                    pen                    terminal.

[http://www.symbol.com/products/mobile\\_computers/mobile\\_ppc\\_ppt2800.html](http://www.symbol.com/products/mobile_computers/mobile_ppc_ppt2800.html)

32. <http://csrc.nist.gov/encryption>

33. <http://csrc.nist.gov/encryption/tkhash.html>