Available online at www.sciencedirect.com
SCIENCE DIRECT
Microprocessors and Microsystems xx (xxxx) 1–10
MICROPROCESSORS AND MICROSYSTEMS
www.elsevier.com/locate/micpro

Energy cost analysis of IPSec on handheld devices

Peifeng Ni, Zhiyuan Li*

Department of Computer Science, Purdue University, West Lafayette, IN 47907, USA

Abstract

We analyze the computation overhead of the IP security protocol (IPSec) on a handheld device. We design experiments to quantify the energy consumed by the individual components in IPSec. We then experiment with several measures which can potentially cut the energy consumption without compromising security. Our results show that by replacing 3DES with Advanced Encryption Standard as the encryption algorithm, the total energy consumption is reduced by up to 38%. On the other hand, MD5 and SHA-1 consume about the same amount of energy for message authentication, which makes SHA-1 more preferred because of its known higher strength against birthday attacks. We also find that the power-saving mode of the wireless LAN interface does not reduce (and may even increase) the total energy cost despite the substantial amount of network idle time. Finally, we find that the data compression option must be used carefully. When sending data from the handheld device, the lossless compression used in IPSec can actually increase the energy consumption until the network bandwidth drops down to 2 Mbps or lower. However, compression is found to save energy when the device receives compressible data under normal bandwidths. This result suggests that the compression option must be dynamically changed in IPSec between data transmission and reception.
© 2004 Published by Elsevier B.V.

Keywords: Handheld devices; Wireless LAN; Security protocols; Low power computing

UNCORRECTED PROOF

1. Introduction

Handheld computing devices with wireless network connections have the potential to become powerful mobile tools to access information and software resources from anywhere at any time. Wireless networks, however, are vulnerable to intrusion. Therefore, securing wireless data transmission is highly critical. On the other hand, the computation and energy cost to achieve security can be high. Since the resource on handheld devices is limited, it is important to evaluate the cost of data security on such devices and to find efficient ways to implement security protocols. In this paper, we evaluate the cost of running the IPSec protocol.

Most of current wireless LANs are based on the widely adopted IEEE 802.11 standard. Unfortunately the security specification, called WEP, in this standard has been shown to be insecure and is thus inadequate for protecting a wireless network from eavesdropping or abuse [1]. As a result, users who have high security concerns turn to other protection measures, e.g. secure socket layer (SSL) and IP security protocol (IPSec), for more reliable encryption and authentication.

Like many other secured transmission techniques, IPSec provides high security based on symmetric encryption algorithms and one-way hash functions. The main advantage of IPSec is the transparency of its security services to both applications and users. The application programs using IPSec do not need to be modified. This is particularly important when the source code of the application programs are not available. This transparency sets IPSec apart from security protocols that operate above the Internet layer.

However, IPSec slows down data access and hence may degrade application performance when compared with unsecured transmissions. This is because the IP stack must be either changed or extended [2] by adding time-consuming functions for authentication and encryption/decryption. Adding headers to IP packets for authentication

* Corresponding author. Tel.: +1 765 494 7822; fax: +1 765 494 0739.
E-mail addresses: npf@cs.purdue.edu (P. Ni), li@cs.purdue.edu (Z. Li).

0141-9331/$ - see front matter © 2004 Published by Elsevier B.V.
doi:10.1016/j.micpro.2004.08.010

and encryption increases the size of IP packets, which in turn may lower the effective bandwidth. Furthermore, executing the additional protocol for security, including Internet Key Exchange protocol, incurs extra computation cost and network traffic.

A handheld device normally has a much slower CPU and a smaller memory than a desktop machine. It has also a very limited battery supply. For example, current high-end pocket PCs with wireless connection can stay active for just a few hours. Therefore, it is important to evaluate the resource, especially the energy, consumed by IPSec and to study energy-efficient implementations without sacrificing security.

In this paper, we examine the effect of different components of IPSec on the energy consumption on handheld devices. This work is a part of a project which investigates the energy issue on handheld devices. A previous paper studies the impact of IPSec on *computation offloading* [6]. Computation offloading is a method to offload computational tasks from a handheld device to a desktop computer or a server, thereby to reduce the energy consumed by the handheld device. The previous paper also examines the impact of the compression facility in IPSec on computation offloading. However, the encryption algorithm used in that study is 3DES, not the new Advanced Encryption Standard (AES) algorithm. Moreover, the study does not investigate the effect of the individual components of IPSec on the data communication cost. Another previous paper [16] studies the impact of data compression on energy consumption of handheld devices. However, the wireless network used in those previous experiments does not run any secure protocols such as IPSec.

The rest of the paper is organized as follows. In Section 2, we analyze the computation overhead of IPSec and measure the timing results for its individual components. In Section 3, we study the computation cost of cryptographic algorithms, which constitute the most expensive operations in IPSec. We show a breakdown of operation counts. In Section 4, we first examine the energy consumed by different components. We then experiment with different energy saving strategies and show the results. In Section 5, we draw conclusions.

2. IPSec overview

IPSec has two modes of operations, namely the transport mode and the tunnel mode. The transport mode provides upper layer (transport layer) protection to direct data traffic between a pair of peer hosts. The tunnel mode, in contrast, provides a tunneled IP protection such that the traffic must pass through a gateway before reaching the ultimate destination. Generally, the tunnel mode is more expensive than the transport mode, because an additional IP header must be appended and more must be protected by encryption and authentication. Our experiments in this paper focus on the transport mode, but most of the observations also apply to the tunnel mode.

IPSec maintains three types of network integrity:

1. Connectionless integrity which detects modifications of a single IP packet by the use of an algorithm specific integrity check value.
2. Anti-replay integrity which detects duplicate IP packets at the receiver end by the use of a sequence number.
3. Connection-oriented integrity which detects lost or reordered IP packets at the receiver end, also by the use of sequence numbers.

These kinds of integrity are provided by two types of security protocols, namely IP authentication header (IPSec AH) [3] and IP encapsulating security payload (IPSec ESP) [4]. We briefly describe these protocols next.

2.1. AH and ESP protocols

IPSec AH protocol provides connectionless integrity, data origin authentication, and anti-replay integrity. The anti-replay integrity is optional and is not enforced at the receiver's end. Depending on the cryptographic algorithm and the method of keying, the AH protocol may also support digital signatures to provide nonrepudiation services. Since the packet carries a digital signature, the sender will not be able to deny having sent it.

The AH protocol provides an additional header (Fig. 1) between the IP and the transport layer headers. This new header includes authentication data which the receiver can use to ascertain that the source of data is as claimed. A keyed one-way hash function, such as MD5 or keyed SHA, is used to compute and verify the AH data [7,8]. Computing and verifying authentication data in this way is much more efficient than encrypting and decrypting the entire IP packet.

IPSec ESP protocol provides confidentiality and anti-replay integrity. It also provides connectionless integrity and data origin authentication. The ESP protocol encrypts (using symmetric encryption algorithms [5,15]) and encapsulate either the transport layer payload or the entire IP packet, depending on the mode of use. The IP module must insert an IP header and encrypt parts of the IP packet accordingly. Encryption is performed on the sender side and decryption on the receiver side. The precise format of

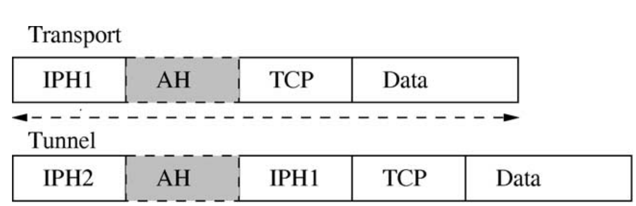


Fig. 1. IPSec AH authentication protection (dotted arrows).

UNCORRECTED PROOF

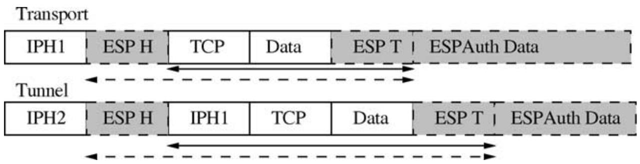


Fig. 2. IPSec ESP authentication protection (dotted arrows) and encryption protection (solid arrows).

the payload data depends on the particular encryption algorithm and transformation in use. The ESP protocol has a narrower authentication scope than AH protocol (Fig. 2) because the IP headers below the ESP header are not protected. If the network environment only requires authentication at the upper layer, then it is more storage-efficient to use only the ESP authentication than using both ESP and AH.

2.2. IPSec components

To study the energy cost of running IPSec, we choose FreeS/Wan IPSec 1.99, which is the most popular implementation of IPSec in the open source community. During data communication under this implementation, the following components are executed:

1. Routines which perform IP version check, time to live (TTL) decrement, routing check, IPSec check and so on.
2. The compression function (invoked when the IPCOMP option is chosen).
3. The encryption function.
4. The authentication function.
5. The ip_send I/O routine which sends the packets.

Running FreeS/Wan IPSec on an HP iPAQ on a 11 Mbps wireless LAN, we found that it takes 158% more time to send data than without IPSec. For each IP packet communicated without compression, about 63% of the time is spent on encryption and authentication. (Details about the experimentation setup can be found in Section 3.) Under most circumstances, longer processing time means higher energy consumption. Thus, our first focus is on the encryption and authentication functions. We wish to study the impact of replacing encryption algorithms on the energy cost. Furthermore, as the CPU is busy executing the encryption function, the wireless LAN interface on the handheld device idles for a substantial amount of time. Setting the power-saving mode for the network interface can potentially reduces energy consumption. We wish to examine its actual effect. Finally, we wish to examine the impact of IPCOMP, the option for data compression.

3. Experimentation setup

In this section, we briefly describe the experimentation setup. The IPSec setup is the same as in a previous study of computation offloading [6] except that we add the AES encryption algorithm in this work. Furthermore, we compare different authentication algorithms and examine the effect of different components of IPSec.

3.1. Hardware setup

The handheld device used in our experiments is a HP iPAQ 3650 which has a 206 MHz Intel StrongArm SA1110 processor and 32 MB RAM. It communicates, through a wireless access point, with a Dell Dimension 4100 desktop computer which has a 1 GHz P-III processor. Both machines run the Linux operating system. The wireless network device of the iPAQ is a Lucent Orinoco (WaveLAN) Golden PCMCIA card which follows the IEEE 802.11b standard. The access point used is LinkSys BEFW1154 which also follows the IEEE 802.11b standard. Under the 11 Mbps nominal peak rate, the effective data rate of the WaveLAN card is measured as about 5 Mbps. *Unless stated otherwise, all experimental results are obtained under this network bandwidth setup*.

3.2. IPSec setup

The implementation of IPSec used in our experiments is Linux FreeS/Wan 1.99. During the experiments, the focus is on the energy consumption for each IP packet. The overhead of constructing a secure session and key refreshment (IKE) [9] is not measured, because its overhead depends more on the security setting than the complexity of the cryptography algorithm. The frequency of re-negotiating the security association and refreshing the key depends on the security requirement. For example, key refreshment performed once per hour is sufficient in many cases, but it may not be good enough for users who want to take extreme caution. Since in different security settings, the proportion of energy spent on IKE may change dramatically, it is more reasonable not to compare IKE's energy cost with that of packet-level cryptographic algorithms. In our experiments, we set the key refreshing rate to once each hour, and we build up the secure connection in advance. Thus, in all the experiments, there is neither key refreshment nor renegotiation of security association.

3.3. Energy measurement setup

To measure the energy consumption of the handheld device, we use an HP 3458a low impedance (0.1 Ω) digital multimeter which take several hundred samples per second and record maximum, minimum and average values between two sample periods. Another feature of the multimeter is that it can be controlled remotely by a computer connecting to it using HPIB connection. By using this mode, the multimeter can take readings at a high speed, which is critical for some of the experiments that require accurate point readings instead of average readings.

UNCORRECTED PROOF

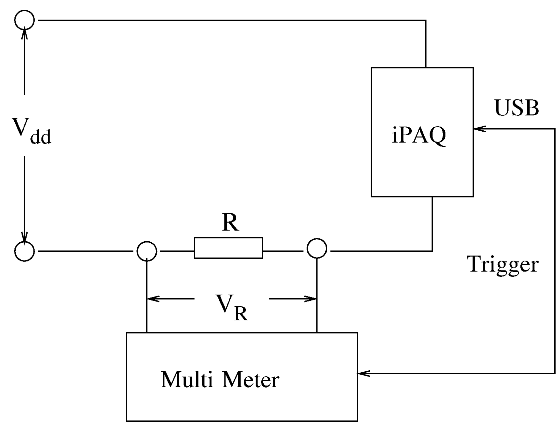


Fig. 3. Measurement setup.

The action of taking readings is controlled by a trigger mechanism built in the multimeter. The device driver for the aforementioned mechanism is taken from PowerScope [11]. However, it is modified to reduce the measurement overhead and to fit in our platform. We install the driver as a kernel loadable module (KLM) which can send and receive signals through USB. After receiving a signal from the iPAQ, the multimeter takes readings and replies a signal which triggers iPAQ's interrupt. The interrupt handler sends another signal to the multimeter to repeat the same procedure. Only three lines of assembly code is needed to send a signal. Therefore, the overhead of invoking the interrupt handler is quite small.

In order to get reliable and accurate readings, we disconnect the batteries from both the iPAQ and the extension pack, using an external 5 V DC power supply instead. The setup of the experiment is shown in Fig. 3. To reduce the error induced by the multimeter, we measure the voltage on a small resistance and use the formula below to calculate the energy consumed by the iPAQ

$$E = (V_{dd} - V_R)V_R/RT, \quad (1)$$

where T is the execution time of the program, $V_{dd}=5$ V is the external power supply voltage. V_R is the measured voltage drop on the small resistance $R=0.2\ \Omega$.

3.4. Workload setup

In order to prepare the workload for experiments, we need to find out whether the types of data payload make any difference. Without IPSec and without data compression, transmitting different types of data always consumes same amount of energy. FreeS/Wan IPSec includes a data compression feature. Therefore, the energy consumption is different for different types of data. In order to separate the effect of compression from that of security algorithms, we disable data compression in the experiments until we specifically study the effect of compression.

Conceptually, the data types should no longer make any difference, because the amount of computation is performed by the encryption and authentication algorithms and the same amount of output is generated. We have verified this by experimenting with files of various types and found little difference in energy consumption. Therefore, unless we specifically study the effect of compression, we use arbitrary types of payload in the experiments.

4. Cryptographic algorithms

IPSec uses two kinds of cryptographic algorithms, namely symmetric encryption algorithms and one-way hash functions. We analyze these algorithms next.

4.1. Symmetric encryption algorithms

Until very recently, most IPSec implementations, including FreeS/Wan, provided 3DES as the default encryption algorithm, partly because it is enforced in Ref. [4]. The 3DES algorithm applies DES encryption to each data block three times with different keys. Let E and D represent DES encryption and decryption, respectively. 3DES encryption is defined as

$$C = E_{k3}(D_{k2}(E_{k1}(M)))$$

where M is the plain text, C is the cipher text, and $k1$, $k2$, $k3$ are three DES keys. Thus, the key length of 3DES is in effect 168 bits long if $k1$, $k2$ and $k3$ are independent. It is 112 bits long if two of $k1$, $k2$ and $k3$ are equal.

In 2001, NIST selected the Rijndael algorithm as the AES both for its strength against cryptanalysis and for its lower computation complexity than the other candidates [17]. Hardware implementations of the Rijndael cipher can encrypt and decrypt at disk transfer speeds.

Rijndael is a block cipher. The block size and the key length can be chosen independently to be 128, 192 and 256 bits. This cipher performs 10, 12, or 14 steps called rounds, depending on the block length and the key length. It was designed to be simple, to be resistant against all known attacks and to have fast and compact coding on many platforms. Each round consists of four layers, which operate either on bytes or 32-bit words.

In order to compare 3DES and AES at the instruction level, we encrypt and decrypt a data block of 16 bytes 10,000 times, using 3DES and AES192, respectively. AES192 is AES with a 192-bit key which is longer than the 168-bit key length of 3DES. We compile FreeS/Wan using GNU's GCC compiler for Pocket Linux to generate the machine code on the HP iPAQ and collect the instruction trace. Tables 1 and 2 list the operation counts.

From these two tables, we see that AES 192 executes a much smaller number of instructions than 3DES of each instruction type. The total number of instructions executed by AES192 is only 38.7% of that of 3DES. Moreover, the

UNCORRECTED PROOF

Table 1
Encryption using 3DES and AES192

Instruction type	3DES		AES192	
	Count	Percentage	Count	Percentage
Load	19,831,838	32.04	8,683,135	36.31
Store	6,670,513	10.78	840,968	3.52
Unconditional branch	370,173	0.60	110,373	0.46
Conditional branch	120,737	0.20	51,651	0.22
Int computation	34,895,064	56.38	14,225,512	59.49
Fp computation	0	0	0	0
Trap	12	0	12	0

ratio among different types of executed instructions are extremely similar for both algorithms, which indicates that neither algorithm uses more power-intensive instructions than the other. This result suggests that the AES consumes less energy than 3DES in IPSec. In a later section, we will examine the difference in the *total* energy consumption.

4.2. Keyed hash functions

Verifying the integrity and authenticity of information is a prime necessity for network security. In particular, two parties communicating over an insecure communication medium require a method to validate the information received as authentic and unmodified by intruders. Most commonly, such a mechanism is based on a secret key shared between the two parties. When party A transmits a message to party B, it appends to the message a value called the *authentication tag*. This value is computed by the Message Authentication Code algorithm which is a function of the transmitted information and the shared secret key. When party B receives the message, it recomputes the authentication tag using the same mechanism and the same key. The recomputed tag must equal the tag attached to the received message. Otherwise the information received is considered altered on the way from A to B. The goal of tag matching is to prevent forgery of the message or the authentication tag.

Previously, the Message Authentication Code algorithm was constructed out of block ciphers such as DES. More recently, however, the preference of the Internet community is to construct the algorithm from cryptographic hash functions which simpler and more efficient when implemented in software. Cryptographic hash functions map strings of different lengths to short string of fixed size. These functions are primarily designed to be collision resistant. This means that if we represent such a hash function by F, then it should be infeasible for an adversary to find two strings x and x' such that $F(x)=F(x')$. Such functions are applied to digital signatures to make them efficient to generate and yet difficult to forge. (More background information on collision-resistant hash functions can be found in Ref. [12].)

IPSec requires mandatory message authentication coding using either Message Digest (MD5) [7] or Secured hash Algorithm (SHA-1) [8] as the one-way keyed hash function. MD5 has shorter output (128 bits) than SHA-1 (160 bits), making it more vulnerable to birthday attack than SHA1. Therefore, unless the total energy cost of running IPSec is much lower under MD5 than under SHA-1, the latter is preferable on the handheld device.

In the FreeS/Wan instruction trace, we obtain an operation-type breakdown for both MD5 and SHA-1 when hashing 64 bytes of data 10,000 times. From Table 3, we see that for each type of instructions, MD5 executes about only 38% as many instructions as SHA-1. The question remains, however, whether this difference translates to a reduction in the *total* energy cost. We examine this issue next.

UNCORRECTED PROOF

Table 2
Decryption using 3DES and AES192

Instruction type	3DES		AES192	
	Count	Percentage	Count	Percentage
Load	20,264,744	32.28	8,685,007	36.24
Store	7,111,463	11.33	841,396	3.51
Unconditional branch	340,284	0.54	110,421	0.46
Conditional branch	121,447	0.19	61,759	0.26
Int computation	34,930,447	55.65	14,268,510	59.53
Fp computation	0	0	0	0
Trap	16	0	16	0

Table 3
MD5 and SHA1

Instruction type	MD5		SHA1	
	Count	Percentage	Count	Percentage
Load	21,372,502	39.75	55,552,591	40.59
Store	6,990,945	13.00	16,711,010	12.21
Unconditional branch	870,314	1.62	2,680,331	1.96
Conditional branch	1,151,712	2.14	4,431,769	3.24
Int computation	23,375,140	43.48	57,495,305	42.01
Fp computation	0	0	0	0
Trap	16	0	16	0

5. Energy cost evaluation

The common configuration used in current IPSec products is ESP(3DES+MD5) or ESP(3DES+SHA-1), which means the ESP protocol is used in conjunction with the 3DES encryption algorithm and the MD5 (or SHA-l) authentication algorithm. FreeS/Wan IPSec contains two kernel-mode functions. Before sending an IP packet, the *transmitting function* encrypts the packet and generates the hash code of it. After receiving an IP packet, the *receiving function* verifies the hash code and decrypts the packet. Naturally, IPSec can be divided into three components, namely encryption/decryption, authentication and the I/O operation for transmitting and receiving.

IPSec alternately applies different functions to each IP packet. Each function is a small piece of code. Most multimeters do not read fast enough to accurately measure the power consumption of a small block of code. In order to evaluate the energy cost of individual IPSec components, we measure the overall energy consumption under different IPSec configurations. By activating different components in different configurations, energy consumption of individual components can be estimated.

The following is the list of configurations used in our experiments:

1. IPSec Module On: executing the IPSec Module from the kernel without activating any authentication or encryption functions.
2. AH(MD5/SHA): using MD5 or SHA as the hash function for IPSec AH authentication without performing encryption.
3. 3DES Only: using 3DES for encryption without message authentication.
4. ESP(3DES/AES)+AH(SHA/MD5): using IPSec AH with either SHA or MD5 as the hashing function and using ESP with 3DES or AES as the encryption algorithm.
5. ESP(3DES/AES+SHA/MD5): using IPSec ESP for both authentication and encryption (with either 3DES or AES as the encryption algorithm, and either MD5 or SHA as the hashing function).

We should note that configurations 1, 2, and 3 listed above are impractical, because both authentication and encryption are highly important for security. These configurations cannot be activated in common settings, and we set up these only for experimental purposes.

We transmit 1 MB of data on the wireless network. Tables 4 and 5 show the voltage (U_R), the consumed time and energy calculated from Eq. (1), under different configurations. These results show that after using IPSec, transmitting and receiving time and energy increase significantly. For example, using ESP(3DES+SHA-1), transmitting time increases by as much as 169%, and so does energy consumption. Using ESP for authentication is slightly less energy consuming than using AH, but the difference is quite small. Using SHA-1 is just slightly more expensive than using MD5. Hence, SHA-l is more preferable because it is less vulnerable to birthday attacks.

In order to get a deeper understanding of the relative weights of each component in IPSec, we calculate a breakdown of energy consumption based on Tables 4 and 5: Let $E_{\mathrm{W/OIPSec}}$ be the total energy cost of data communication without using IPSec, $E_{\mathrm{AH(MD5)}}$ be the total energy consumption when using IPSec under AH(MD5), and E_{All} be the total energy consumption of using IPSec

Table 4
Results of transmitting 1 MB data

Configurations	Voltage (mV)	Time (s)	Energy (J)
W/O IPSec	76.6237	1.6862	3.1806
IPSec module on	76.6485	1.6998	3.2073
AH(MD5) only	76.1816	2.0076	3.7652
AH(SHA-1) only	75.2747	2.2582	4.1855
3DES only	73.6057	4.1712	7.5628
3DES+AH(MD5)	73.6726	4.396	7.9773
3DES+AH(SHA-l)	73.2261	4.5486	8.2049
ESP(3DES+MD5)	73.3591	4.3546	7.8691
ESP(3DES+SHA-1)	73.3203	4.5432	8.2055

UNCORRECTED PROOF

Table 5
Results of receiving 1 MB data

Configurations	Voltage (mV)	Time (s)	Energy (J)
W/O IPSec	70.0225	1.7044	2.9420
IPSec module on	70.3559	1.7054	2.9575
AH(MD5) only	71.0451	1.8188	3.1841
AH(SHA-l) only	71.4109	1.9064	3.3534
3DES only	70.2684	3.7456	6.4870
3DES+AH(MD5)	70.5269	3.8176	6.6360
3DES+AH(SHA-1)	70.3383	3.9132	6.7845
ESP(3DES+MD5)	69.9901	3.7966	6.5506
ESP(3DES+SHA-1)	70.2531	3.8716	6.7039

under AH(MD5)+ESP(3DES). The value of $E_{\mathrm{All}}-E_{\mathrm{AH(MD5)}}$ should be close to E_{3DES}, the energy consumed by 3DES. The value $E_{\mathrm{AH(MD5)}}-E_{\mathrm{W/OIPSec}}$ should be close to E_{MD5}, the energy consumed by MD5. $E_{\mathrm{SHA\text{-}1}}$ can be derived similarly. Fig. 4 shows the ratios of energy consumed by 3DES, MD5 and the network I/O. From the figure, we see that the energy consumed by the hashing function (MD5 in this case) is a small percentage of the total, which explains why the choice between MD5 and SHA-1 does not have much effect on the total energy cost.

From Fig. 4, we see that 3DES consumes the largest part of energy. If this part of energy can be reduced, the total energy consumption will be reduced significantly. After replacing 3DES with AES in the FreeS/Wan IPSec package we obtain the results shown in Tables 6 and 7, with key lengths of 128, 192, and 256 bits, respectively. There is little difference in the total energy consumption for different key lengths. In all cases, the total energy consumption is significantly lower than using 3DES. Notice that after replacing 3DES with AES, encryption is no longer the most consuming part of IPSec. It consumes about 13% of energy when receiving data and about 25% when sending data.

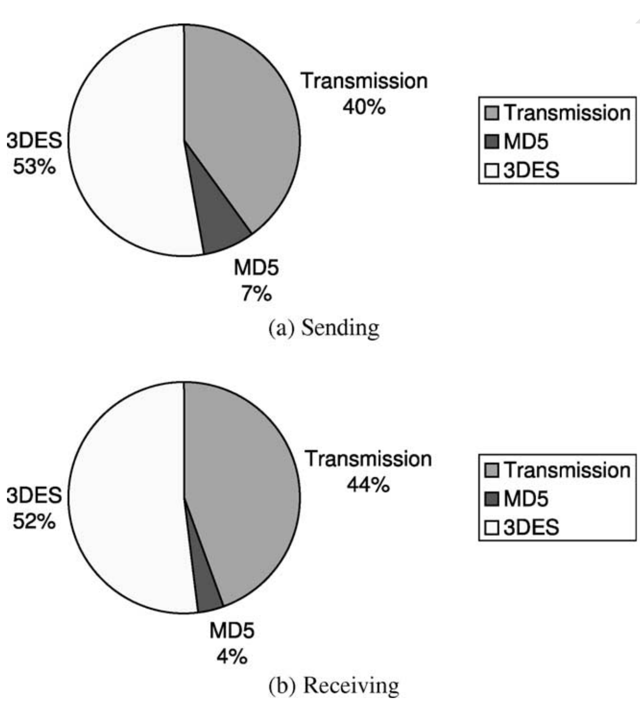


Fig. 4. Break down energy consumption ESP(3DES+MD5).

Table 6
Results of transmitting 1 MB data using AES

Configurations	Voltage (mV)	Time (s)	Energy (J)
AES128+AH(MD5)	75.1965	2.6654	4.9352
ESP(AES128+MD5)	75.0595	2.6496	4.8973
ESP(AES128+SHA-1)	75.0205	2.8452	5.2562
ESP(AES192+MD5)	75.0948	2.7174	5.0251
ESP(AES192+SHA-1)	74.0961	2.9416	5.3685
ESP(AES256+MD5)	74.8867	2.7448	5.0618
ESP(AES256+SHA-1)	74.7351	2.9446	5.4193

6. Power-saving mode and compression

IEEE 802.11 [13] supports two power modes: active and power-saving (PS). Under the so-called *infrastructure network*, the access point (AP) monitors the mode of each mobile host. A host in the active mode is fully powered and thus may transmit and receive at any time. On the contrary, a host in the PS mode only wakes up periodically to check for possible incoming packets from the AP. A host always notifies its AP when changing modes. Periodically, the AP

Table 7
Results of receiving 1 MB data using AES

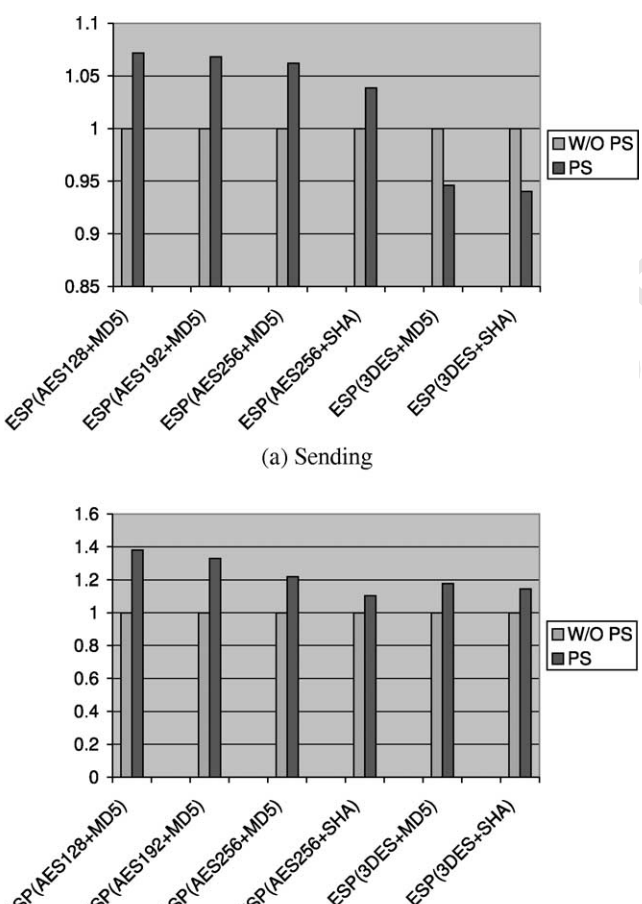
Configurations	Voltage (mV)	Time (s)	Energy (J)
AES128+AH(MD5)	70.9881	2.0796	3.6377
ESP(AES128+MD5)	70.8596	1.9726	3.4443
ESP(AES128+SHA-1)	70.8032	2.1814	3.8072
ESP(AES192+MD5)	70.7578	2.0984	3.6587
ESP(AES192+SHA-1)	70.9390	2.2522	3.9377
ESP(AES256+MD5)	70.7607	2.1166	3.6918
ESP(AES256+SHA-1)	79.7405	2.3568	4.1064

transmits beacon frames spaced by a fixed beacon interval. A PS host should monitor these frames.

When in the PS mode, the WaveLAN card utilizes the hardware mechanism that periodically switches between the sleep mode and the idle mode. It significantly reduces the electrical current when there exist no network activities. It reduces the effective data rate by about 25% when there exists network communication, due to the overhead of switching between states. Heuristics have been proposed in literature to predict the optimal timing to wake-up from sleep mode [14]. However, the success rate of such methods highly depends on event predictability.

From the timing data in Table 4 through Table 7, we see that the encryption and decryption operations decrease the network throughput significantly. These operations increase the idle time on the WaveLAN card. This seems to suggest an opportunity for reducing the energy consumed by the WaveLAN card by enabling the power-saving mode. One must realize that, however, changing the power mode requires a costly operating-system call. It requires a hardware reset of WaveLAN card to switching between the normal mode and the PS mode and to change the polling period. For our WaveLAN card, e.g. it takes about 40 ms to finish the switching, about the same amount of time to process more than 10 IP packets. (The time to process a single IP packet of about 1500 bytes ranges from 2 to 4 ms on our handheld device, depending on whether compression is performed and what kind of encryption and authentication algorithm is used.) Inserting such a mode-switching function during data communication can cause many packets not to be processed on time and eventually be dropped. Therefore, a more sensible way to use the PS mode is to enable it throughout the data communication session.

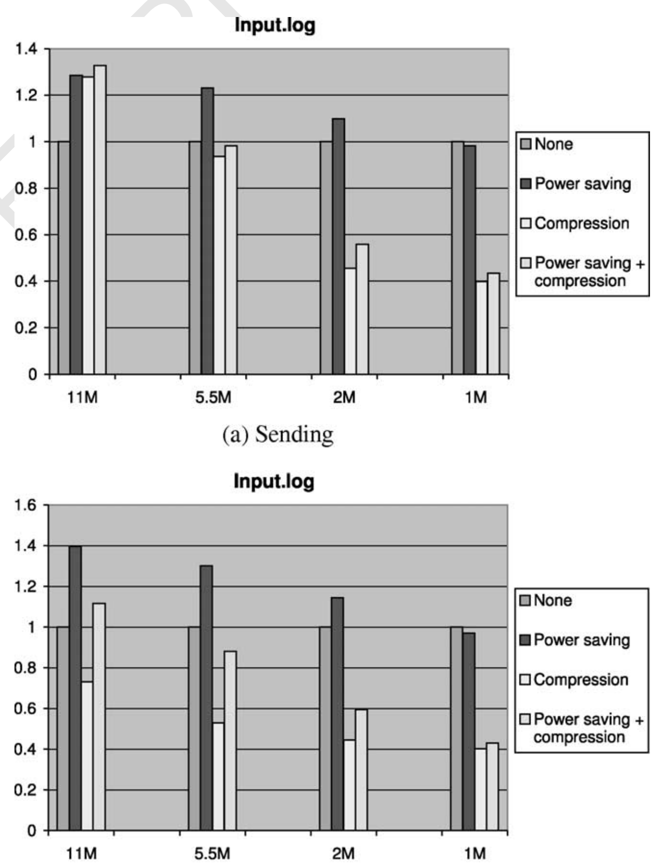
Setting the polling period to the default value 100 ms, we measured the effect of power-saving mode on communication time, power rate, and total energy consumption, using various combinations of encryption algorithms, key lengths and authentication hashing functions. Note that the total energy consumption is the product of the average power consumption rate and the total communication time. We found that, under the PS mode, the power consumption rate is decreased by 6–11% when sending data, regardless of the encryption algorithm being AES or 3DES. (The power consumption rate actually increases slightly when *receiving* data under AES, but it decreases by about 14% under 3DES.) On the other hand, in the PS mode, the total communication time is increased by 5–15% when sending data and by 10–40% when receiving data, depending on the



(a) Sending

(b) Receiving

Fig. 5. Effect of power saving mode on energy.



(a) Sending

(b) Receiving

Fig. 6. Effect of bandwidth (Input.log).

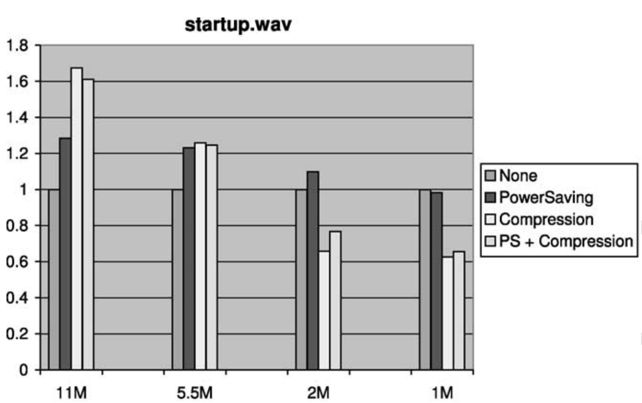
algorithms. (This shows that the overhead of hardware mode-switching can be quite high.) Unfortunately, the time-power product is higher in the PS mode except when sending data under 3DES encryption. Fig. 5 compares the total energy consumption.

The polling period is the time between the WaveLAN card's wakeups. An increase of the period normally leads to an increase in the communication latency. A decrease, on the other hand, will cause more frequent switching. We found that increasing the polling period from 100 ms tends to increase the energy consumption, despite the decreased power rate, because of the increased latency. Moreover, we did not find any substantial difference in the result when varying the polling periods in the range of 1–100 ms.

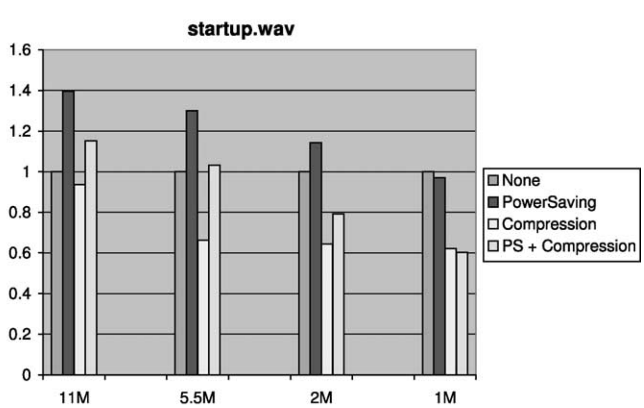
FreeS/Wan IPSec has an option called `IPCOMP` which causes the protocol to invoke the `zlib` routine for lossless compression. This routine implements the Lempel-Ziv compression algorithm. When IPCOMP is turned on, every IP packet is compressed before being encrypted. Although a compressed file takes less time to send or receive, the compression/decompression operations can be quite expensive. We choose three files of different types to examine the effect of IPCOMP on the total energy consumption of IPSec under the configuration of ESP(AES128 + MD5).

`Input.log` is a text taken from a log file which contains many repeated patterns and its compression ratio is 11.24:1. `Startup.wav` is a raw audio file whose compression ratio is 2.24:1. `Input.random` is a random data file which cannot be compressed. We vary the network bandwidth from 1 to 11 Mbps. In practice, the bandwidth may vary due to the change in the distance to the AP, noises, contentions, and so on. We examined the effect of different bandwidths both on the power-saving mode and on compression. Figs. 6–8 show the experimental results. For each network-bandwidth value, the results are relative to the case of no power-saving mode and no compression. In a previous study [16], our group compares different compression schemes which are applied to a large number of different data types.

From these figures, one can see that as the bandwidth decreases to as low as 1 Mbps, the PS mode starts to help reduce the energy consumption because of the increased idle time. However, the reduction due to the PS mode is very little. At a higher bandwidth, the PS mode actually increases the total consumption. For files that are compressible, energy is saved by using decompression when the bandwidth is low. The reason is obvious. When the bandwidth drops, the cost of network transmission begins to outweigh the cost of compression/decompression. In a

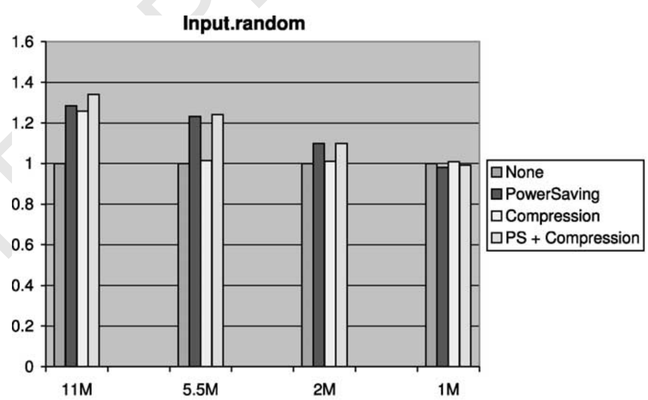


(a) Sending

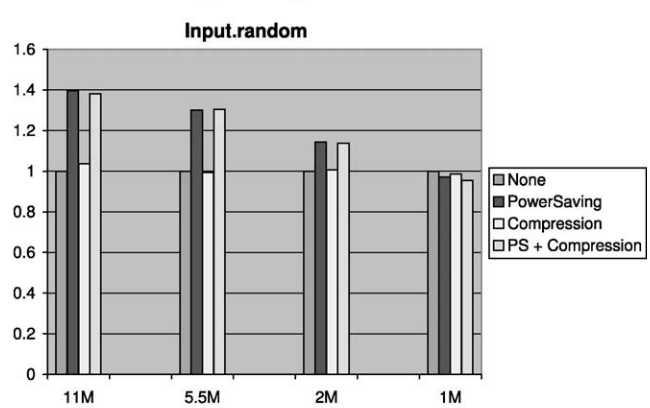


(b) Receiving

Fig. 7. Effect of bandwidth (startup.wav).



(a) Sending



(b) Receiving

Fig. 8. Effect of bandwidth (Input.random).

previous study of data compression, our group presents an adaptive mechanism to dynamically determine whether to transmit compressed or uncompressed data. Although that study does not use any secure protocols such as IPSec, we believe that its idea of adaptive compression can still be beneficial in the IPSec environment. We leave this issue for future investigation.

7. Conclusion

In this paper, we have shown experimental results indicating that the 3DES cipher algorithm consumes more than half of the energy in the typical use of IPSec for data transmission. After replacing 3DES with AES, the replacement reduces the energy consumption by up to 38%. This replacement, therefore, not only makes IPSec less vulnerable to cryptanalysis, but also make the protocol much more energy efficient for handheld devices.

The network card remains idle during a considerable amount of time when the processor executes the cipher algorithms. Nonetheless, we found that, in most cases, the power-saving mode of the IEEE 802.11 does not save the energy for data transmission under IPSec. In fact, power-saving mode results in a quite substantial energy loss when the handheld device receives data. We believe that this is mainly due to the fragmented idle time of the network card as the cipher algorithm is applied to separate packets, making it difficult to offset the overhead to switch between the power-saving and the normal modes. Our result suggests that the power-saving mode should not be used during data communication under IPSec in spite of the existence of network card idle time.

We found that the data compression option in IPSec must be used carefully. When sending data from the handheld device, the lossless compression used in IPSec may actually increase the energy consumption until the network bandwidth drops down to 2 Mbps or lower. When receiving data, however, the energy is almost always saved if the data is compressed. This contrast is mainly due to the fact that compression requires much more operations than decompression. Our result suggests that the compression option must be dynamically changed in IPSec between data transmission and reception.

Acknowledgements

This work is sponsored in part by National Science Foundation through grants CCR-0208760, ACI/ITR-0082834, and CCR-9975309, and by Indiana 21st Century Science and Technology Fund. The authors acknowledge Rong Xu's contribution to the infrastructure building of the handheld device project. The authors thank the careful reviews from the anonymous reviewers.

References

[1] N. Borisov, I. Goldberg, D. Wagner, Intercepting mobile communications: the insecurity of 802.11, Seventh Annual International Conference on Mobile Computing and Networking, Rome, Italy, July (2001).

[2] R. Oppliger, Security at the internet layer, Computer 31 (9) (1998) 43–47.

[3] S. Kent, R. Atkinson. IP AuthenticationHeader. IETF RFC 2402, 1998.

[4] S. Kent, R. Atkinson. IP Encapsulating Security Payload (ESP). IETF RFC 2406, 1998.

[5] S. Kent, R. Atkinson. Security Architecture for the Internet Protocol, IETF RFC 2401, 1998.

[6] Z. Li, R. Xu, Energy impact of secure computation offloading on a handheld device, IEEE Fifth Annual Workshop on Workload Characterization (WWC-5), IEEE Computer Society Press, Los Alamitos, CA, November 2002.

[7] C. Madson, R. Glenn. The Use of HMAC-MD5-96 within ESP and AH. RFC 2403, November, 1998.

[8] C. Madson, R. Glenn. The Use of HMAC-SHA-l-96 within ESP and AH. RFC 2404, November, 1998.

[9] D. Harkins, D. Carrel. The Internet Key Exchange (IKE). RFC 2409.

[10] M. Bellare, J. Kilian, P. Rogaway, The security of cipher block chaining in: Y. Desmedt (Ed.),, Proceedings of Advances in Cryptology (Crypto 94), Lecture Notes in Computer Science 839, Springer, Berlin, 1994.

[11] J. Flinn, M. Satyanarayanan, PowerScope: a tool for profiling the energy usage of mobile applications, Proceedings of the Second IEEE Workshop on Mobile Computing Systems and Applications, New Orleans, LA, February (1999).

[12] J. Nechvatal, Public key cryptography in: G. Simmons (Ed.),, Contemporary Cryptography, the Science of Information Integrity, IEEE Press, New York, 1992.

[13] S. Narayanaswamy, V. Kawadia, R.S. Sreenivas, P.R. Kumar, Power control in ad-hoc networks: theory, architecture, algorithm and implementation of the COMPOW protocol, European Wireless February (2002).

[14] M. Stemm, R.H. Katz, Measuring and reducing energy consumption of network interfaces in handheld devices, Proceedings of the Third International Workshop on Mobile Multimedia Communications (MOMUC-3), September (1996).

[15] A.S. Tanenbaum, Computer Networks, third ed, Prentice Hall, Englewood Cliffs, NJ, 1996. pp. 588–595.

[16] R. Xu, Z. Li, C. Wang, P. Ni, Impact of data compression on energy consumption of wireless-networked handheld devices, Proceedings of the 23rd IEEE International Conference on Distributed Computing Systems (ICDCS'03), IEEE Computer Society Press, Los Alamitos, CA, 2004 in press.

[17] S.S. Wagstaff Jr., Cryptanalysis of Number Theoretic Ciphers, Chapman & Hall/CRC Press, London/Boca Raton, FL, December, 2002.

UNCORRECTED PROOF