Design and Performance Analysis of Electronic Seal Protection Systems Based on AES

Dong Kyue Kim, Mun-Kyu Lee, You Sung Kang, Sang-Hwa Chung, Won-Ju Yoon, Jung-Ki Min, and Howon Kim

A very promising application of active RFID systems is the electronic seal, an electronic device to guarantee the authenticity and integrity of freight containers. To provide freight containers with a high level of tamper resistance, the security of electronic seals must be ensured. In this paper, we present the design and implementation of an electronic seal protection system. First, we propose the eSeal Protection Protocol (ePP). Next, we implement and evaluate various cryptographic primitives as building blocks for our protocol. Our experimental results show that AES-CBC-MAC achieves the best performance among various schemes for message authentication and session key derivation. Finally, we implement a new electronic seal system equipped with ePP, and evaluate its performance using a real-world platform. Our evaluation shows that ePP guarantees a sufficient performance over an ARM9-based interrogator.

Keywords: Active RFID, electronic seal (eSeal), eSeal protection protocol, AES, pseudorandom function, message authentication code.

Manuscript received Mar. 05, 2007: revised Aug. 06 2007.

The preliminary versions of this paper were presented in WISA 2006 [1], [2].

This work was supported by the Korea Research Foundation Grant funded by the Korean Government (MOEHRD) (The Regional Research Universities Program/Research Center for Logistics Information Technology) and grant no. R01-2006-000-10957-0 from the Basic Research Program of the Korea Science & Engineering Foundation.

Dong Kyue Kim (phone: + 82 2 2220 2312, email: dqkim@hanyang.ac.kr) is with the Division of Electronics and Computer Engineering, Hanyang University, Seoul, Rep. of Korea.

Mun-Kyu Lee (email: mklee@inha.ac.kr) is with the School of Computer Science and Engineering, Inha University, Incheon, Rep. of Korea.

You Sung Kang (email: youskang@etri.re.kr) and Howon Kim (email: khw@etri.re.kr) are with Information Security Research Division, ETRI, Daejeon, Rep. of Korea.

Sang-Hwa Chung (phone: + 82 51 510 2434, email: shchung@pusan.ac.kr), Won-Ju Yoon (email: anospirit@pusan.ac.kr), and Jung-Ki Min (email: jkmin81@gmail.com) are with the Department of Computer Engineering, Pusan National University, Busan, Rep. of Korea.

I. Introduction

1. Radio Frequency Identification

Radio frequency identification (RFID) is an automatic identification method, in which identification data is stored on electronic devices called RFID tags (or transponders), and the information stored in RFID tags can be retrieved by an interrogator (or reader) using radio waves. An RFID tag only stores information to identify an object, such as a commercial product, animal, or a person. More detailed information on that object is stored in a backend server, which communicates with the interrogator through another channel. According to their power supply, RFID tags can be classified into two categories. Active tags have their own internal power source, while passive tags receive their energy from the electromagnetic field of the interrogator. In this paper, we are interested in active tags.

Although RFID systems are now used in many applications including supply chain management, transport systems, animal identification, access control, and so on, security issues must be properly addressed for RFID systems, since they often deal with secret and private information. There has been extensive research on many aspects of RFID security, including blocker tags [3], hash lock schemes [4], hash chains [5], pseudonyms [6], re-encryption [7], block-cipher-based authentication [8], and so on.

2. Electronic Seal and Security

An electronic seal, or eSeal [9], is an electronic device to guarantee the authenticity and integrity of freight containers. It is an improved version of the manual cargo seal [10] which provides physical protection like a lock and indicates whether

or not the sealed entrance has been compromised. An eSeal can also contain identification data for containers and shipment information; therefore, it can be seen as a kind of active RFID tag. While there are several kinds of eSeals including infrared seals and remote reporting seals supporting satellite or cellular communications, the most popular ones are RFID-based eSeals. There are already many commercial RFID eSeal products operating at 433.92 MHz (UHF band). There are also ongoing standardization activities such as ISO 18185 drafts by ISO [9], [11]-[14].

According to ISO 18185-1 [9], the communication between an eSeal and an interrogator is performed using a command-response protocol, that is, the interrogator always initiates a session using a pre-defined command, and the eSeal responds to it with appropriate data. The only exception is the alert message, which is initiated by an eSeal. Originally, an eSeal was defined as a read-only, non-reusable freight container seal conforming to the high security seal defined in ISO/PAS 17712 [10] and conforming to ISO 18185 or revision thereof that electronically evidences tampering or intrusion through container doors. However, the concept of the eSeal is evolving into a tamper-resistant device with rewritable memory to store user-defined confidential information.

Although eSeals can provide freight containers with a high level of tamper resistance by immediate alert, error condition reporting, and event logging, the security of the eSeal itself must be ensured [1]. There are many possible attacks against the authenticity and integrity of an eSeal. For example, an attacker can erase the tamper event log inside an eSeal, plant a fake event in the log, generate a fake alarm to deceive the interrogator, and so on [15]. Unfortunately, the current specifications for RFID eSeals do not provide any robust solution to these problems.

3. Previous Works

The draft standard ISO 18185, established by ISO TC104/SC4/WG2, defines application requirements, environmental characteristics, and various protocols for eSeals [9], [11]-[14]. However, a recent report by Motorola indicated that major deficiencies in the current ISO 18185 draft standard will lead to delayed or missed reads, inadequate security, and a lack of interoperability [16]. There was a particularly extensive vulnerability assessment for eSeals in early 2005, and spoofing and cloning were identified as potential data integrity threats to eSeals [17]. Hence, device authentication is believed to be the highest priority solution to mitigate those identified risks, and the eSeal standard-setting work (ISO 18185-4 [13]) is being expanded to meet that objective. As related works, we recently proposed a challenge-response protocol for mutual authentication between an eSeal and an interrogator [15]. We also implemented a 433 MHz active RFID system [18].

4. Contribution

In this paper, we present the design and implementation of an eSeal protection system to protect confidential information and provide mutual authentication between an eSeal and its associated interrogator. Our contribution is three-fold.

- We implement various cryptographic primitives as building blocks for our system. First we implement the standard block cipher AES [19] to guarantee the confidentiality of packets as well as several message authentication schemes for message integrity, authentication, and session key derivation.
- We propose the eSeal Protection Protocol (ePP)[1], which is based on AES. The ePP provides mutual authentication between an eSeal and its corresponding interrogator, supplementing the existing communication protocol with new security commands. It also provides several security functionalities such as data confidentiality, data integrity, immunity to DoS and replay protection.
- We implement software modules for ePP, embed them into an eSeal and an interrogator, and evaluate their performances on a real-world eSeal system. Our evaluation shows that security protocols based on standard block cipher AES-128 over a low-cost and low-power processor, such as ATmega128L, cannot conform to the turn-around timing requirements of the current ISO 18185-7 standard [14], even if it is fully optimized. Therefore, it seems that the turn-around time in the current ISO 18185-7 standard is too tight, and it should be loosened to support secure operations.

5. Organization

The remainder of this paper is organized as follows. In section II, we describe security properties required for the communication protocol between an eSeal and an interrogator. Section III introduces cryptographic primitives, which will be the building blocks for our eSeal protection system, and analyzes their performance. In section IV, we propose ePP and design new commands conforming to the existing eSeal standards. Section V provides experimental results for performance evaluation of our system on a real-world platform. Finally, we conclude in section VI.

II. Requirements for eSeal Data Protection

In April 2005, ISO investigated threats and vulnerabilities in

1) This protocol was presented as a form of report at an ISO TC104 meeting [20].

the air interface between an eSeal and an interrogator. According to this survey, the vulnerabilities are classified into the following three categories [17].

- Gather (loss of confidentiality): A malicious party tries to gain information about an eSeal by intercepting radio signals between eSeals and interrogators, or by probing a valid eSeal electrically.
- Mimic (loss of integrity and authenticity): A malicious party clones an eSeal (cloning) or creates a device that cannot be discerned from a legitimate eSeal (spoofing).
- Denial of service (loss of availability): A malicious party disrupts the communication channel by jamming or shielding, or disrupts the eSeal by power consumption or physical destruction.

In order to solve the above problems, a communication protocol between an eSeal and an interrogator should guarantee the following security functions:

- Mutual authentication between an eSeal and an interrogator
- Data confidentiality and data integrity
- Non-repudiation of stored data
- Immunity to denial of service
- Replay protection

III. Cryptographic Primitives for eSeal Protection

1. Cryptographic Algorithms

In this section, we give preliminary information on cryptographic primitives, which are the main building blocks for our eSeal protection system.

A. Encryption Algorithms

To guarantee data confidentiality, an encryption algorithm should be used. Since it is natural to assume that an eSeal has only limited computing power, we decided to use symmetric key encryption. In our eSeal system, we use the standard block cipher AES [19]. The AES cipher is a substitution-permutation network composed of iterative rounds, where each round except the last contains four different transforms; SubBytes, ShiftRows, MixColumns, and AddRoundKey (No MixColumns transform is performed in the last round). While there are three allowable key lengths, namely, 128 bits, 192 bits and 256 bits, we only consider the first one and call it AES-128.

For realization of our eSeal protection system, it is necessary to find an efficient way to implement AES and other related algorithms because cryptographic operations are resource-consuming operations in general. There have been extensive studies on the efficient implementation of AES [8], [21]-[25], where most of the optimization is done on the SubBytes transform, since this is the most complex transform that involves a finite field inversion operation on $GF(2^8)$. A typical approach to implementing this transform is to use a pre-computed table called an S-box. An S-box maps an 8-bit input to an 8-bit output; thus, it requires 256 elements in total. Note that if we use AES-128, the input and output of every transform are 128 bits long. Therefore a SubBytes transform requires sixteen table look-ups. For hardware implementation, we can perform multiple SubBytes transforms in parallel. As a general rule, the more S-boxes are used in parallel, the less clock cycles are needed for encryption [8]. For software implementation, most of the opportunity for improvement lies in manual code optimization, such as register reallocation and loop unrolling. Also, there is a novel technique in which an isomorphic composite field $GF((2^4)^2)$ is used instead of the original field $GF(2^8)$ [21].

B. Pseudorandom Functions and Message Authentication Codes

In addition to an encryption algorithm, we need a pseudorandom function (PRF) to derive various keys from the master key, and a message authentication code (MAC) for authentication and data integrity. Actually, PRF and MAC are closely related to each other, and there are many practical implementations of PRF and MAC in various international standards, many of which use hash functions or block ciphers for the building blocks of PRF as follows.

- IKE (Internet Key Exchange) [26], [27], which is a component of IPsec used for mutual authentication and security association management, defines MACs and PRFs based on hash functions and AES, such as HMAC-MD5, HMAC-SHA1 and AES-XCBC-MAC [28], [29].
- TLS (Transport Layer Security) protocol [30] for communication security over the Internet defines a PRF using HMAC-MD5 and HMAC-SHA1.
- IEEE 802.11i [31] for Wireless LAN security defines a PRF as a concatenation of HMAC-SHA1 outputs.
- IEEE 802.16e [32] standard defines a key derivation function as iterations of CMAC or SHA-1.

In this paper, we consider hash-based MAC (HMAC) schemes [33] using standard hash functions, such as MD5 and SHA-1. We also consider block-cipher-based MACs, such as CBC-MAC [34], CMAC [35], and XCBC-MAC [36], [37] using AES. Detailed description of these schemes can be found in [1].

Note that the cryptographic strength of the above schemes is based on the properties of the underlying primitives, namely, block ciphers and hash functions. Since several weaknesses were recently found in MD5 and SHA-1 [38], [39], the use of AES-based PRFs and MACs would be preferable from the viewpoint of security.

2. Efficient Implementation

In this subsection, we present various experimental results to implement AES and PRFs. An eSeal is equipped with a low-end microcontroller so that it can process identification data, shipment information, and tamper event logs. Therefore, we can provide security functions as a form of software code optimized for this microcontroller. Our target device for eSeal is Atmel's ATmega128L microcontroller[2], which is a RISC processor with 32 general purpose 8-bit registers. It has a program memory of 128 KB and a data memory of 4 KB, and it operates at various clock speeds. We chose a speed of 8 MHz, and we used WinAVR (release 20060421) as a cross compiler. Because an interrogator has to deal with many packets which are sent to and from numerous eSeals around it, its computing power should be much better than that of an eSeal. For this reason, we used Samsung's S3C2440A processor for the interrogator, which is a 16/32-bit RISC microprocessor based on the ARM920T core and has a maximum 400 MHz operating speed.

A. Efficient Implementation of AES

In our software implementation of AES-128, as in typical software implementations, we concentrated on maximizing the throughput. By loop unrolling ten rounds of AES, we could obtain some improvement in throughput at the expense of program memory. Also we constructed a pre-computation table for the xtime operation, namely, a multiplication by x over $GF(2^8)$, as well as a pre-computed S-box. Thus, we could obtain the data given in the first two rows of Table 1.

According to our analysis of this initial implementation, the compiled code had many load/store instructions, since it stores the 128-bit state into the data memory. Also, loading pre-computed values for the S-box and xtime operations consumes many machine cycles to compute the addresses of these values. This problem seems more serious on the ATmega128L processor because it operates at a very low clock speed. Therefore, we wrote assembly programs for ATmega128L, in which each state is stored in registers, not in the memory, and the addresses of pre-computed tables are fixed so that they are not computed repeatedly. Additionally, we tried the following modifications.

1) First, we convert the C code into a hand-written assembly code, which is a common technique of optimization.
2) AES is composed of ten rounds, each of which has a similar structure. In the initial version previously explained, we unrolled this loop to speed up computation. However, loop unrolling requires more memory to program each round independently. In the second modification, we removed loop unrolling, recovering the loop structure, to reduce the required program memory.
3) Each round of AES, except the last round, is composed of four transforms, namely, SubBytes, ShiftRows, MixColumns, and AddRoundKey, where SubBytes maps each input byte into another byte according to an S-box, and ShiftRows permutes the output bytes of SubBytes in a pre-defined order. Hence, a common optimization technique merges SubBytes and ShiftRows into a single transform, which performs substitution and permutation together. Note that this method improves both the throughput and the usage of program memory.
4) If we represent each element in $GF(2^8)$ as a polynomial with degrees up to 7 and whose coefficients are in {0, 1}, the MixColumns transform can be written as a matrix multiplication as

$$\begin{pmatrix} o_0 \\ o_1 \\ o_2 \\ o_3 \end{pmatrix} = \begin{pmatrix} x & x+1 & 1 & 1 \\ 1 & x & x+1 & 1 \\ 1 & 1 & x & x+1 \\ x+1 & 1 & 1 & x \end{pmatrix} \begin{pmatrix} i_0 \\ i_1 \\ i_2 \\ i_3 \end{pmatrix},$$

where i_j, o_j ($j = 0,\cdots, 3$) are polynomials with degrees up to 7. Since this transform requires many polynomial multiplications, a common acceleration technique for MixColumns is to pre-compute $x \times a$ for every $a \in GF(2^8)$. Thus, the initial version of our AES implementation has two pre-computation tables, one for the xtime operation and the other for the S-box, as previously mentioned. However, the xtime table guarantees less speed improvement than the S-box table. In our fourth modification, we removed the pre-computation table for xtime to reduce the required data memory. This modification also reduces the program memory slightly, since there is no routine to construct an xtime table.

The lower part of Table 1 shows the results of these

Table 1. Performance of software modules for AES-128.

		Memory (B)		Time (μs)	
		Program	Data	Key expansion	Encryption
C on S3C2440A		15,204	436	3.9	5.4
C on ATmega128L		8,334	554	268.0	604.0
Assembly on ATmega128L	Method 1	4,270	512	105.0	277.1
	Method 2	1,688	512	105.0	302.7
	Method 3	1,660	512	105.0	293.7
	Method 4	1,528	256	103.7	339.2

2) We do not use a more powerful processor, such as ARM 9, because ATmega128L is a low-power microcontroller. Power consumption is very important for an eSeal system, since an eSeal has to survive the lifetime of a freight container. Also, ATmega128L has an internal RAM and an internal EEPROM, which saves more power.

Table 2. Performance of various MAC modules written in C on ATmega128L and S3C2440A.

	S3C2440A			ATmega128L		
	Memory (B)		Throughput* (Mbps)	Memory (B)		Throughput* (kbps)
	Program	Data		Program	Data	
HMAC-MD5	5,509	632	7.71	10,498	982	18.35
HMAC-SHA1	7,741	632	7.22	5,950	1,003	13.33
AES-CBC-MAC	15,633	568	16.18	8,762	554	154.22
AES-CMAC	15,897	584	11.72	9,018	570	111.30
AES-XCBC-MAC	15,757	616	9.03	8,938	602	83.39

* Key scheduling time is included.

Table 3. Performance of AES-based MAC modules written in assembly over ATmega128L.

		Memory (B)		Throughput (kbps)
		Program	Data	
AES-CBC-MAC	Method 1	4,636	688	371.01
	Method 2	2,044	688	345.95
	Method 3	2,016	688	355.56
	Method 4	1,888	432	316.05
AES-CMAC	Method 1	4,806	706	266.67
	Method 2	2,214	706	248.54
	Method 3	2,186	706	253.47
	Method 4	2,054	450	224.56
AES-XCBC-MAC	Method 1	4,776	706	195.42
	Method 2	2,182	706	181.56
	Method 3	2,154	706	186.86
	Method 4	2,022	450	164.10

experiments. Clearly, each of the four methods using hand-written assembly codes requires a smaller amount of memory and much less time than a module compiled from a C program. We can observe a time-memory tradeoff between methods 1 and 2 in the table. Method 3 slightly improves both the memory size and the execution time compared to method 2. Removal of pre-computation tables (method 4) reduces the amount of data memory, but the execution time increases.

B. Efficient Implementation of PRF

Next, we consider the implementation of MAC and, thus, PRF. As explained in section III.1, we can take two categories of MACs into account, namely, hash-based MACs and block-cipher-based MACs. We compare the following typical choices: HMAC-MD5 and HMAC-SHA1 for hash-based MACs, and AES-CBC-MAC, AES-CMAC, and AES-XCBC-MAC for AES-based MACs.

Table 2 compares the performance of MAC modules implemented using the C language, where AES-based algorithms use the AES routines given in the previous section. Because we use MAC algorithms for key derivation and authentication in our eSeal system, the input message for MAC is very short in most cases. Therefore, we set the message length to 256 bits, that is, 32 bytes, which is a sufficient value for communication protocols between an eSeal and an interrogator [9]. Table 3 shows the performance of MAC modules using hand-written assembly modules for AES-128 over ATmega128L. In these tables, we can see that AES-based algorithms are much faster than hash-based ones using a comparable amount of memory, but the gain is relatively smaller over S3C2440A than it is over ATmega128L because the MD5 and SHA-1 algorithms have been designed to fit into a 32-bit architecture, but AES was designed to perform well also on an 8-bit architecture. In particular, AES-CBC-MAC shows the best throughput and uses the smallest amount of memory; therefore, we use AES-CBC-MAC for our MAC and PRF.

IV. eSeal Protection Protocol (ePP)

In this section, we present ePP and its related commands conforming to the existing eSeal standards. We also analyze its security features. First, we begin by introducing the overall structure of an eSeal system.

1. Overview of an eSeal Protection System

Figure 1 shows the architecture of the proposed system. We assume that there are two kinds of back-end servers to support secure data management, namely, an authentication server and a digital signature server. The authentication server stores master keys to access eSeals, where every eSeal is related to a distinct master key. After the interrogator obtains the seal ID of the target eSeal using the collection method conforming to ISO 18185-1 [9], it sends a request for a master key to the authentication server, and acquires a proper key related to the identified eSeal.[3] The digital signature server generates a signature to guarantee that the data inside an eSeal is valid. Hence, the data is signed using the private

3) The problem of key distribution is important, but it is beyond the scope of this paper. The architecture of our system is similar to that of WPA2 (Wi-Fi Protected Access 2), or IEEE 802.11i [31], which is a security framework for IEEE 802.11 Wireless LAN. Hence it can be effectively implemented using a mechanism similar to WLAN.

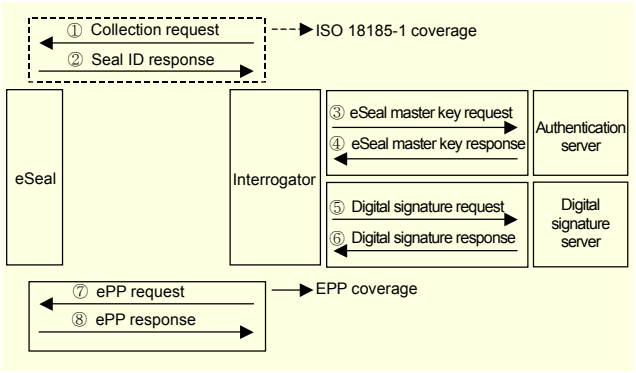

Fig. 1. Architecture of the proposed eSeal system.

key of the sender of a freight container. Although the signature is transferred together with data when it is uploaded to an eSeal, this does not mean that the eSeal has the ability to verify the digital signature. An eSeal only functions as a signature carrier. The signature is verified by another interrogator that reads the eSeal's data and checks the authenticity and integrity of the corresponding freight container. We assume that communication between the interrogator and the digital signature server is protected by IPsec, and that signing and verification are performed according to ECDSA-163 [40].

As previously explained, steps 1 to 6 in Fig. 1 can be implemented by existing protocols. In steps 7 and 8, the interrogator reads data from an eSeal or writes data to an eSeal in a secure manner. The purpose of this section is to design and analyze protocols for this part of the system. The security of this communication is supported by a unique master key which is hard-programmed into the eSeal. This master key should be identical to one stored in the authentication server's database with that eSeal's ID.

2. Design of New Commands

Since there are already many eSeal systems and related ISO standards, the new ePP should conform to these existing standards. Therefore, our ePP operations are not designed as completely new ones, rather they are embedded into three basic command types of eSeal standard: read, write, and alert. Our new command codes are defined according to the existing ISO command format [9]. Table 4 shows these new commands. The ePP Write and ePP Read commands are used for an eSeal and an interrogator to authenticate each other when the eSeal's data is accessed. These commands are always initiated by the interrogator's request. On the other hand, ePP Alert command is a broadcast command initiated by an eSeal, and it is used to generate an authentic alert to prevent a fake alert from an illegal eSeal. All of these communications are secured using the master key.

Table 4. ePP command codes.

Code	Name	Type	Direction	Description
0x51	ePP Write	Request	Interrogator→eSeal	Write into eSeal's memory
	ePP Write	Response	eSeal→Interrogator	
0x53	ePP Read	Request	Interrogator→eSeal	Read from eSeal's memory
	ePP Read	Response	eSeal→Interrogator	
0x7F	ePP Alert	Alert	eSeal→Interrogator (Broadcast)	Alert message

To guarantee the cryptographic strength of ePP, we use a well-known authenticated encryption mode of block cipher, namely, counter with CBC-MAC (CCM) mode of operation [41], which is also used in the IEEE 802.11i standard [31]. This choice is reasonable in the context of our eSeal system since CBC-MAC shows better performance than other block-cipher-based MACs and hash-based MACs. The CCM mode requires encapsulation and decapsulation procedures for each packet. We use modifications of the procedures defined in the IEEE 802.11i standard [31] and denote them as CCMe_ENC$_{Key}$ (*Rand*, *AAD*, *Message*) and CCMe_DEC$_{Key}$ (*Rand*, *AAD*, *Cipher*), respectively, where CCMe stands for *CCM for eSeal*, *Key* is a symmetric key for CCM, *Rand* is a random number which becomes a part of the nonce, *AAD* is additional authentication data, *Message* is a message to be encoded, and *Cipher* is a ciphertext to be decoded and verified. The precise description of these procedures will be given separately in the next subsection.

A. ePP Write Command

Figure 2 shows the procedure for the ePP Write command, which involves the following parameters:

- *r*: random number to guarantee freshness of a new session

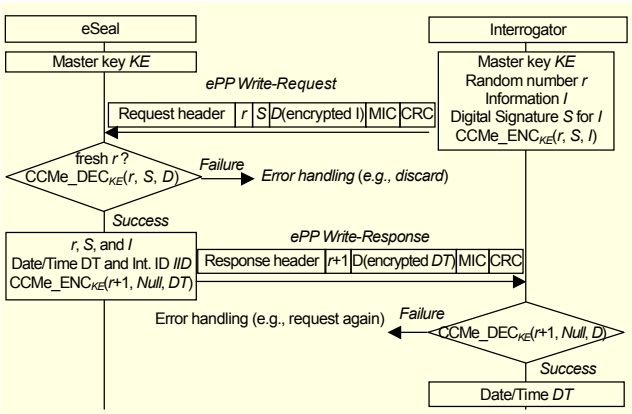

Fig. 2. ePP Write procedure.

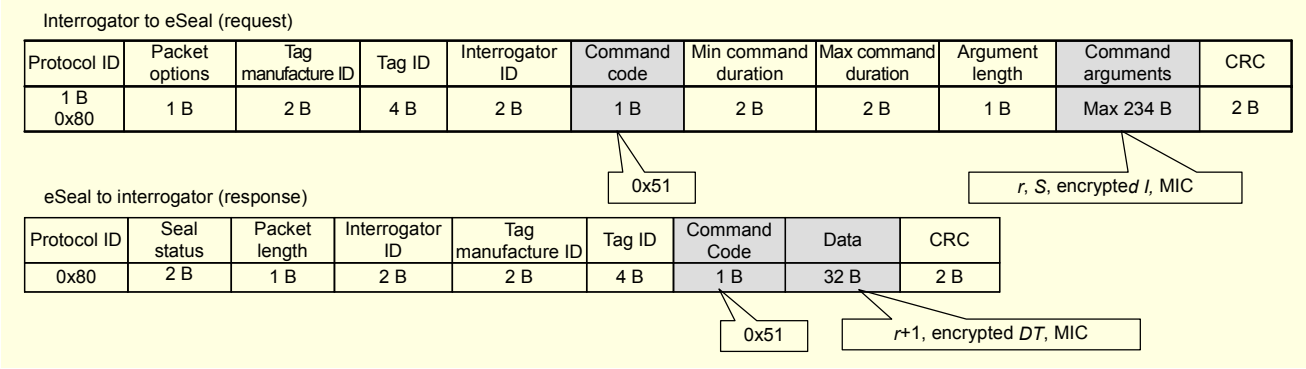

Fig. 3. Packet format for the ePP Write command.

- *KE*: master key
- *I*: confidential information to be protected
- *S*: digital signature for *I*
- *DT*: current date and time
- *D*: encrypted value for either *I* or *DT*
- *IID*: interrogator ID

The message integrity code is denoted by MIC, and CRC stands for cyclic redundancy check.

The ePP Write procedure is a two-way handshake protocol including the ePP Write-Request and ePP Write-Response. The ePP Write-Request includes the ePP packet header, which delivers the seal ID and the interrogator ID, and it also includes *r*, *S*, *D*, MIC, and CRC. Each step of the ePP Write procedure is performed as follows.

Step 1. Interrogator (request)
- Obtain the master key *KE* from the authentication server.
- Generate a random number *r*.
- Prepare confidential information *I*.
- Get the digital signature *S* for *I* from the digital signature server.
- Perform the *CCMe_ENC* function to encrypt *I* and calculate MIC for *I* and *S*. We use *S* as AAD of CCM encapsulation.
- Send the ePP Write-Request command as shown in Fig. 2.

Step 2. eSeal (response)
- Receive the ePP Write-Request command.
- Perform CCMe_DEC function to decrypt the ciphertext *D* and verify MIC. We should use the same random number *r* and the same AAD *S* as in the encapsulation procedure performed by the interrogator.
- If decryption fails or if there is already a stored random number equal to r, this command is discarded.
- Store *r*, *S*, and *I*.
- Record the current data/time, *DT*, and *IID*.
- Increase *r* by 1.
- Perform the CCMe_ENC function to encrypt *DT* and calculate MIC for *DT*.
- Send the ePP Write-Response as shown in Fig. 2.

Step 3. Interrogator
- Receive the ePP Write-Response.
- Perform the CCMe_DEC function to decrypt the ciphertext *D*.
- If decryption fails, then the response is discarded.
- Record the decrypted *DT*.

Figure 3 shows the packet format for the ePP Write Information command, which conforms to the ISO command format [9]. The gray boxes show newly defined values.

B. ePP Read Command

The second ePP operation is ePP Read, which is depicted in Fig. 4. We omit the procedure and packet format of ePP Read, since they are similar to those of ePP Write.

C. ePP Alert Command

The last ePP operation is ePP Alert, which is shown in Fig. 5. The procedure of ePP Alert is performed as follows.

Step 1. eSeal (Alert)
- Generate a random number *r*.

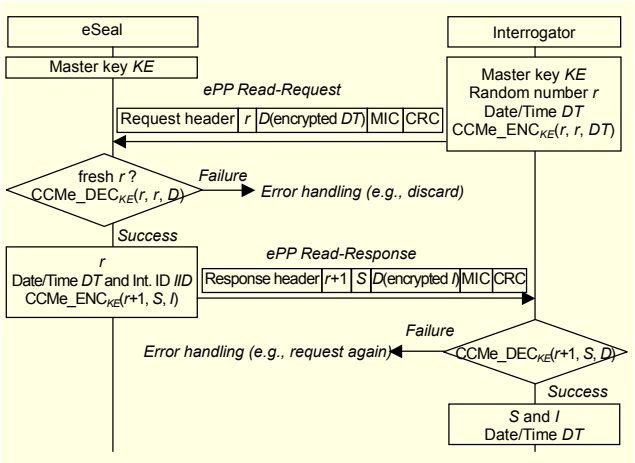

Fig. 4. ePP Read procedure.

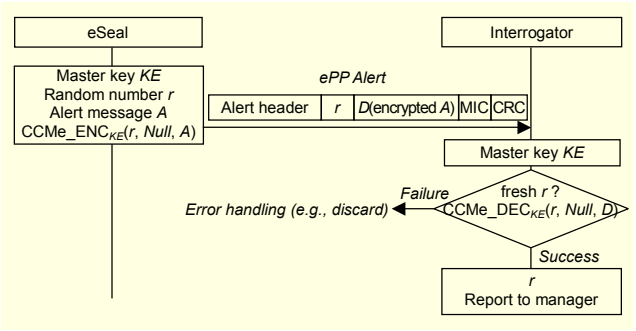

Fig. 5. ePP Alert procedure.

- Prepare an alert message *A*.
- Perform the *CCMe_ENC* function to encrypt *A* and calculate MIC for *A*.
- Send the ePP Alert packet.

Step 2. Interrogator (alert management)

- Receive the ePP Alert message.
- Obtain the master key *KE* from the authentication server.
- Perform the *CCMe_DEC* function to recover *A*.
- If the decryption fails or there is a stored random number equal to the received *r*, then the alert message is discarded.
- Store *r*, and report the alert to the manager.

3. Cryptographic Functions for ePP

Figure 6 shows an example of CCMe encapsulation which is performed by an interrogator to initiate an ePP Write operation. The interrogator generates a write request for *Message* by encrypting it using AES-CCM (part (A) in the figure) with a random number *r* and additional authentication data, namely, a digital signature *S* for *Message*. As well as an encrypted message *Ciphertext*, an MIC is generated (part (A)), expanding the original packet size by 8 bytes (part (B)). Then, CRC is appended after the MIC (part (C))[4]. The CCMe decapsulation procedure can be defined as a counterpart to the CCMe encapsulation procedure. For a more detailed description, see [2].

According to the preceding description, the following primitives should be implemented for ePP:

- pseudorandom number generator (PRNG) to generate *r*
- AES-128 and its counter and CBC modes for CCM encryption/decryption
- PRF to derive a mutual transient key (MTK) from a master key

For the first item, we can reuse the PRNG used for other eSeal operations, such as anti-collision and collection, and it can be realized using various methods based on hardware or

4) Although CRC also increases the packet size, this is not an overhead, since every eSeal command packet should contain a CRC even when we do not use ePP.

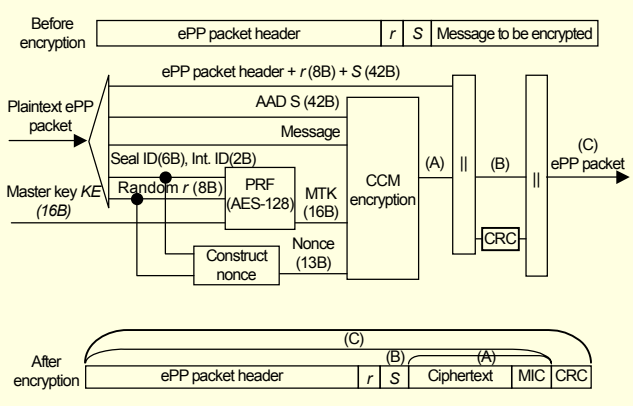

Fig. 6. Generation of an ePP Write-Request packet using CCMe_ENC$_{KE}$ (*r*, *S*, *Message*).

software. The software implementation of AES-128 using C and assembly languages was provided in section III.2, and various PRFs based on hash functions and AES modes of operations were compared. As demonstrated in section III.2, AES-CBC-MAC is the most efficient. The key derivation procedure using PRF is defined as

$$\text{MTK} = \text{AES-CBC-MAC}_{KE}(r \,\|\, \text{seal ID} \,\|\, \text{interrogator ID}),$$

where *KE* is a 128-bit master key, *r* is a 64-bit random number, and the seal ID and interrogator ID are 6 and 2 bytes long, respectively. Therefore, this is actually AES encryption of a single message block. An MTK is refreshed for every packet. Finally, a 13-byte nonce is constructed by concatenating *r*, the three most significant bytes of the seal ID, and the 2-byte interrogator ID.

4. Security Analysis

In this section, we analyze the security features of the proposed ePP using the terms given in section II.

A. Mutual Authentication

This security service is for the ePP Write and ePP Read operations. The eSeal and the interrogator can authenticate each other by proving to have driven the same MTK from the same pre-shared master key to generate the MIC.

B. Data Confidentiality and Data Integrity

Since we encrypt the private data *I* using MTK, and this MTK should be derived from the master key, an eavesdropper who does not know the master key cannot understand *I*. The encryption is done with the CCM mode of AES, which also supports message authentication. Thus, an attacker who does not know the master key can neither generate a fake message nor change the content of a message.

C. Non-repudiation of Stored Data

In our protocol, only the party who has permission to write the confidential information I should be able to generate a digital signature S using its own private key. The digital signature server shown in Fig. 1 plays the role of the owner of the private key, and delegates the interrogator to write I and the corresponding S to the eSeal. The connection between the interrogator and the digital signature server is protected through a secure channel. An eSeal does not verify a digital signature. It functions as a carrier of the signature.

D. Partial Immunity to Denial of Service

Although DoS cannot be prevented completely, it can be mitigated by ePP, since ePP will not proceed any further after an integrity check fails.

E. Partial Replay Protection

An eSeal and an interrogator hold a list of the previously used random numbers and compare the received random numbers with previously stored values. Hence, a reused message can be detected. However, since the memory of the eSeal is restricted, it cannot store the complete list of random numbers.

V. Performance Analysis

1. Testbed for eSeal Protection System

In this section, we estimate the performance of our eSeal protection system. First, we introduce an eSeal platform implemented in our previous work [18] as a test bed for performance evaluation. In this platform, the interrogator and the eSeal were implemented in accordance with ISO 18185-1 and ISO 18185-7 [9], [14]. Figure 7 shows the block diagram for the hardware of the interrogator and the eSeal.

The interrogator is equipped with a Samsung microprocessor (S3C2440A), a Chipcon RF transceiver (CC1020), a 64 MB SDRAM, a 16 MB flash memory, an Ethernet controller, and other peripheral components. The S3C2440A is a 16/32-bit RISC microprocessor based on the ARM920T core and has a maximum operating speed of 400 MHz. It embeds a real time clock (RTC) unit, and the only external RTC component required is a 32.768 kHz crystal. The RTC function is necessary for an eSeal system that includes the ePP operations. The RF transceiver, which is controlled by the microprocessor, allows the interrogator to communicate with the eSeal using the 433.92 MHz frequency. The interrogator provides a 10 Mbps Ethernet interface and an RS-232 serial interface. Through these interfaces, the interrogator receives

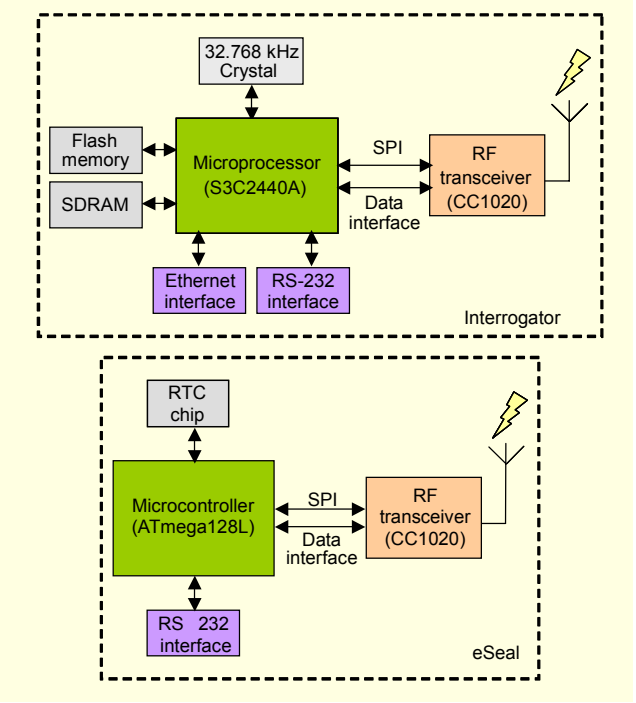

Fig. 7. Block diagram of the hardware of the interrogator and the eSeal.

commands from and sends responses to a host system. The interrogator software modules are implemented using the C language based on the embedded Linux (kernel version: 2.4.20), which was ported to the microprocessor of the interrogator.

The eSeal is equipped with an Ateml microcontroller (ATmega128L); a Chipcon RF transceiver (CC1020), which is the same as the one used in the interrogator; an RTC chip; and other components. The eSeal is powered by two 1.5 V AA batteries. It also has an RS-232 serial interface to monitor the function of the eSeal. The eSeal software modules are implemented using the C language, with the exception of cryptographic algorithms, which are implemented in part using the assembly language.

For cryptographic operations performed in the interrogator and the eSeal, we use the software modules implemented in section III. The implemented interrogator and eSeal work in accordance with ISO 18185-4 as well as ISO 18185-1 and ISO 18185-7 [13], [9], [14]. In addition, so that the interrogator may support the IPsec protocols, we ported the IPsec software modules to it, including the IKEv2 software modules.

We measured the timing for processing ePP and analyzed the processing overheads in both the interrogator and the eSeal. In our experiments, the processes for digital signatures were not included in the ePP procedures. We also assumed that an interrogator already has the seal ID of a target eSeal. Figure 8 shows the experimental environment for performance evaluation of ePP. As explained above, the microprocessor

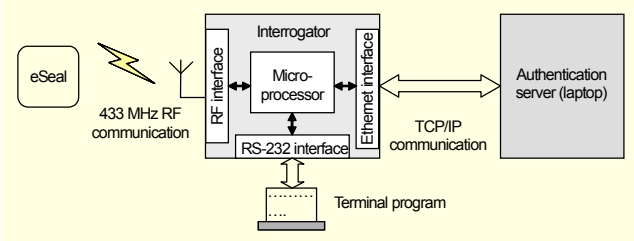

Fig. 8. Experimental environment for performance evaluation of ePP.

(S3C2440A) in the interrogator operates at 400 MHz and the microcontroller (ATmega128L) in the eSeal operates at 8 MHz.

For the experiments, we developed a simple authentication server on a laptop to provide the master key corresponding to the Seal ID. The laptop was equipped with a 2.0 GHz Intel Pentium 4 CPU and 512 MB RAM. It operated with Linux Fedora Core 3 (kernel version: 2.6.9). The interrogator was directly connected to the authentication server through 10 Mbps Ethernet without a switch. A 10 Mbps Ethernet connection was sufficient for our platform, since the amounts of transmitted data were very small, such as 6 bytes or 16 bytes.

2. Latency in the Backend Communications

Before ePP is processed, the interrogator must acquire the master key corresponding to the seal ID from an authentication server. We measured the time required for this process for two cases, one using common TCP/IP communication and the other using IPsec-enabled TCP/IP communication. The interrogator sent the 6-byte eSeal ID to the authentication server and received the 16-byte master key from the server. Table 5 shows the total latency for the interrogator to acquire the master key in the two cases.

The latency with IPsec is approximately 36 μs longer than that using common TCP/IP communication because the processing time for the IPsec protocols is added. Thus, the processing time for the IPsec protocols is much less than the latency of TCP/IP communication. In a real communication environment, the latency of TCP/IP communication is not measured in microseconds but in milliseconds. With high communication latencies, the processing time of the IPsec protocols in the interrogator can be disregarded, although the latency of TCP/IP communication itself may cause a bottleneck in the ePP performance in the interrogator. However, in future network environments with low communication latencies, the processing time for the IPsec protocols in the interrogator may become significant.

Table 5. Latency for the interrogator to acquire the master key.

Feature	Latency
Using common TCP/IP communication	592 μs
Using IPsec-enable TCP/IP communication	628 μs

3. Analyses of Processing Overheads in Interrogator and eSeal

To analyze the processing overheads of ePP, we measured the time spent in processing ePP in both the interrogator and the eSeal. For experimental purposes, we revised our ePP so that only confidential information *I* is encrypted; thus, in ePP Write-Response and ePP Read-Request, *DT* is not encrypted, but transmitted plain. There are no MICs for these packets.

Figure 9 shows the processing times for the two ePP commands in the interrogator. The processing of ePP proceeds as follows: (1) the interrogator generates the ePP Request packet (xx_Packet); (2) the interrogator performs the CCMe encapsulation of the ePP Request packet (xx_Enc); (3) the interrogator sends the ePP Request packet to the eSeal and the eSeal processes it and returns the ePP Response packet (xx_RF and xx_eSeal); (4) the interrogator performs the CCMe decapsulation of the received ePP Response packet (xx_Dec).

As shown in Fig. 9, the total processing time of ePP (xx_Total), which is the sum of xx_RF, xx_eSeal, xx_Packet, xx_Enc, and xx_Dec, is proportional to the amount of confidential data exchanged between the interrogator and the eSeal. However, as the amount of data increases, the processing times of the cryptographic operations for ePP (xx_Enc and xx_Dec) do not increase much, or stay constant. Moreover, xx_Enc and xx_Dec are much less than the time for RF communication between the interrogator and the eSeal (xx_RF) or the processing times of ePP in the eSeal (xx_eSeal). This proves that ePP is a light-weight protocol. Therefore, ePP operations in the interrogator incur very few processing overheads.

Figure 10 shows the processing time of the two ePP commands in the eSeal. The total processing time of ePP (xx_Total), which is the sum of xx_Packet, xx_Dec, and xx_Enc, is the same as xx_eSeal in Fig. 9. The eSeal performs the CCMe decapsulation of the received ePP Request packet (xx_Dec), generates the ePP Response packet (xx_Packet), and performs the CCMe encapsulation of the ePP Response packet (xx_Enc).

As shown in Fig. 10, xx_Total is proportional to the amount of confidential data exchanged between the interrogator and the eSeal. However, while the interrogator has sufficient processing power, the eSeal's processing power is limited. For that reason, as the amount of confidential data increases, the processing times of the cryptographic operations for ePP

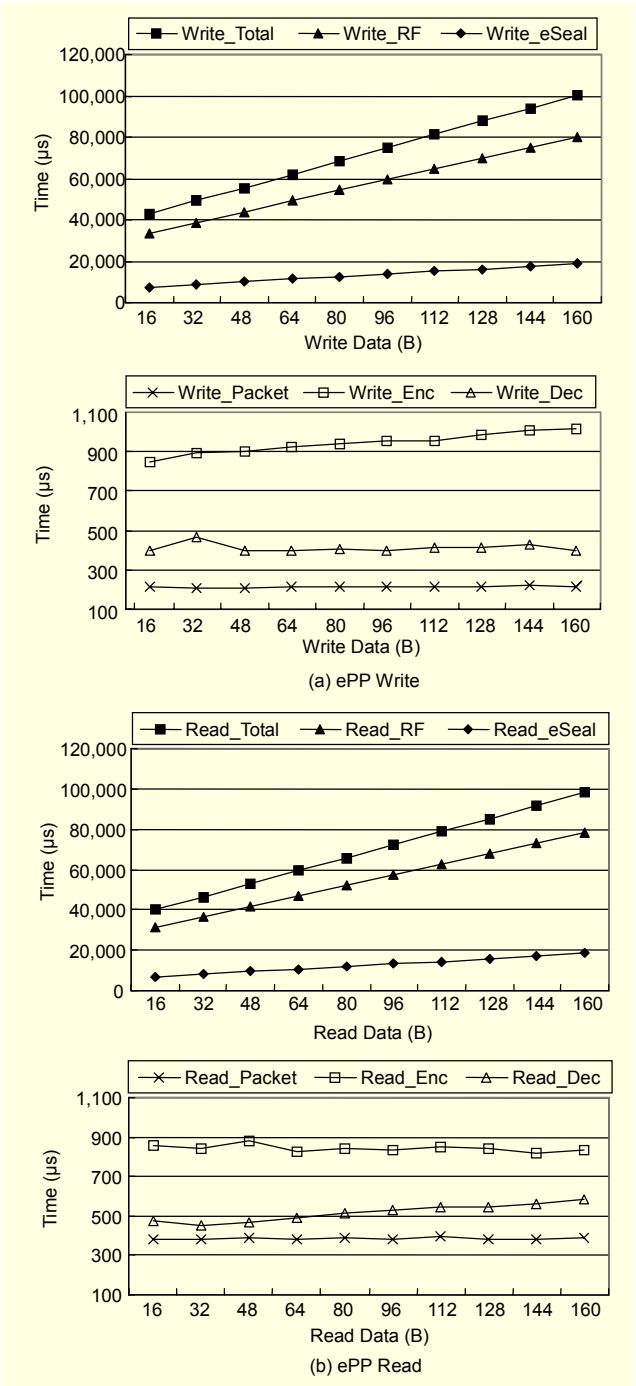

Fig. 9. Processing time of the two ePP commands in the interrogator.

(xx_Enc and xx_Dec) also increase. For ePP Write, as more confidential data is received from the interrogator, more processing time is required for the CCMe decapsulation of the ePP Write-Request packet (Write_Dec). For ePP Read, as more confidential data is sent to the interrogator, more processing time is required for the CCMe encapsulation of the ePP Read-Response packet (Read_Enc). On the other hand, the ePP Write

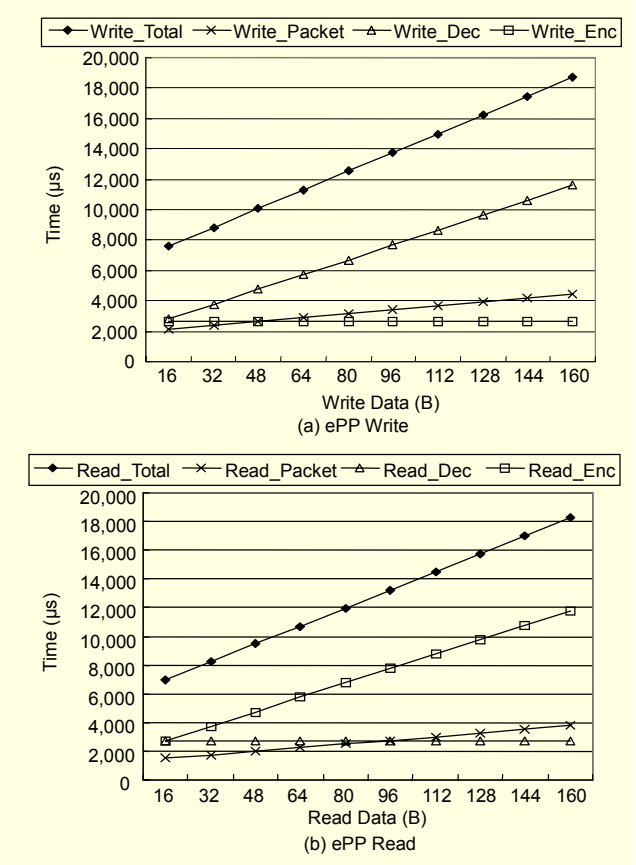

Fig. 10. Processing time of the two ePP commands in the eSeal.

Response packet and the ePP Read-Request packet contain no confidential data, and their sizes are fixed; therefore, their processing times stay constant, independently of the amount of confidential data to be written or read (Write_Enc and Read_Dec).

VI. Concluding Remarks

In this paper, we presented the design and implementation of an electronic seal protection system. We identified security requirements for the eSeal system, and proposed the eSeal Protection Protocol based on the standard block cipher AES. Our security analysis shows that ePP satisfies the security requirements. We implemented various cryptographic primitives as building blocks for our protocol, and evaluated the performance of ePP on a real-world test bed using these primitives. Our evaluation shows that ePP guarantees a sufficient performance over an ARM9-based interrogator.

However, processing ePP in an eSeal requires several milliseconds, even if we implement the cryptographic algorithms using the reasonably optimized assembly routines given in section III. The turn-around time in the eSeal system should be less than 1 ms to conform to the ISO 18185-7

standard [14]. To address this problem, we could use a more powerful processor such as ARM9, but enhancing the computing power would increase the power consumption of an eSeal. Since an eSeal has to survive for a long period, at least during the transport time of freight containers, the power consumption issue is critical; therefore, the most economic solution would be to loosen the turn-around timing requirements defined in the ongoing ISO 18185 standard.

If modification of the standard is not acceptable for other reasons, we can use a dedicated hardware for AES-128. Then, we can significantly reduce packet processing time, since it accelerates AES-CCM for encryption and MIC generation, and also accelerates the key derivation using AES-CBC-MAC. In the preliminary version of this paper [1], we provided various FPGA modules for AES, and their throughputs are over 400 Mbps, which seems to easily solve the problem. Finally, there is an extensive literature on low power implementation of cryptographic hardware. For example, [42] would be a useful reference.

References

[1] M.K. Lee, J.K. Min, S.H. Kang, S.H. Chung, H. Kim, and D.K. Kim, "Efficient Implementation of Pseudorandom Functions for Electronic Seal Protection Protocols," *International Workshop on Information Security Applications-WISA 2006*, *LNCS*, vol. 4298, Springer, 2007, pp. 173-186.

[2] Y. Kang, H. Kim, and K. Chung, "Design of Lightweight Security Protocol for Electronic Seal Data Protection," *Pre-Proceedings of WISA 2006*, 2006, pp. 517-531.

[3] A. Juels, R. Rivest, and M. Szydlo, "The Blocker Tag: Selective Blocking of RFID Tags for Consumer Privacy," *Proceedings of the 10th ACM Conference on Computer and Communications Security*, 2003, pp. 103-111.

[4] S.A. Weis, *Security and Privacy in Radio-Frequency Identification Devices*, Master's Thesis, Massachusetts Institute of Technology, 2003.

[5] M. Ohkubo, K. Suzuki, and S. Kinoshita, "Cryptographic Approach to 'Privacy-Friendly' Tags," *RFID Privacy Workshop*, 2003.

[6] A. Juels, "Minimalist Cryptography for Low-Cost RFID Tags," *The 4th Int'l Conf. Security in Communication Networks-SCN 2004*, LNCS, vol. 3352, Springer, 2004, pp. 149-164.

[7] P. Golle, M. Jakobsson, A. Juels, and P. Syverson, "Universal Re-encryption for Mixnets," *CT-RSA 2004*, *LNCS*, vol. 2964, Springer, 2004, pp. 163-178.

[8] M. Feldhofer, S. Dominikus, and J. Wolkerstorfer, "Strong Authentication for RFID Systems Using AES Algorithm," *Cryptographic Hardware and Embedded Systems-CHES 2004*, LNCS, vol. 3156, Springer, 2004, pp. 357-370.

[9] ISO 18185-1, *Freight Containers - Electronic Seals - Part 1: Communication Protocol,* ISO, 2006.

[10] ISO 17712, *Freight Containers - Mechanical Seals*, ISO, 2003.

[11] ISO 18185-2, *Freight Containers - Electronic Seals - Part 2: Application Requirements*, ISO, 2005.

[12] ISO 18185-3, *Freight Containers - Electronic Seals - Part 3: Environmental characteristic*, ISO, 2005.

[13] ISO 18185-4, *Freight Containers - Electronic Seals - Part 4: Data Protection*, ISO, 2006.

[14] ISO 18185-7, *Freight Containers - Electronic Seals - Part 7: Physical Layer*, ISO, 2006.

[15] S. Park, M.K. Lee, D.K. Kim, K. Park, Y. Kang, S. Lee, H. Kim, and K. Chung, "Design of an Authentication Protocol for Secure Electronic Seals," *Cybernetics, Informatics and Systemics 2005*, 2005, pp. 47-51.

[16] Motorola, Inc., "Second Report of Detailed Container Use Cases and Deficiencies in the ISO 18185-1, ISO 18185-7, and ISO 18000 Standard," 2005, available at http://www.autoid.org/tc104_sc4_wg2.htm (sc4wg2n0233).

[17] T. Drake and J. Reinold, "ISO Study: Vulnerabilities and Threats for Container Identification Tags and e-Seals," 2005, available at http://www.autoid.org/tc104_sc4_wg2.htm (sc4wg2n0225).

[18] W.J. Yoon, S.H. Chung, H. Kim, and S.J. Lee, "Implementation of a 433 MHz Active RFID System for U-Port," *The 9th International Conference on Advanced Communication Technology*, 2007.

[19] FIPS Publication 197, *Advanced Encryption Standard*, NIST, 2001.

[20] Electronics and Telecommunications Research Institute, "Report of ePP (eSeal Protection Protocol) for ISO 18185-4," 2005, available at http://www.autoid.org/tc104_sc4_wg2.htm (sc4wg2n0254).

[21] A. Rudra, P. Dubey, C. Jutla, V. Kumar, J. Rao, and P. Rohatgi, "Efficient Rijndael Encryption Implementation with Composite Field Arithmetic," *Cryptographic Hardware and Embedded Systems - CHES 2001*, *LNCS*, vol. 2162, Springer, 2001, pp. 171-184.

[22] P. Chodowiec and K. Gaj, "Very Compact FPGA Implementation of the AES Algorithm," *Cryptographic Hardware and Embedded Systems - CHES 2003*, *LNCS,* vol. 2779, Springer, 2003, pp. 319-333.

[23] S. Mangard, M. Aigner, and S. Dominikus, "A Highly Regular and Scalable AES Hardware Architecture," *IEEE Transactions on Computers*, vol. 52, no. 4, 2003, pp. 483-491.

[24] K. Aoki and H. Lipmaa, "Fast Implementation of AES Candidates," *Third AES Candidate Conference - AES3*, 2000, available at http://csrc.nist.gov/CryptoToolkit/aes/round2/conf3/aes3papers.html.

[25] T. Wollinger, M. Wang, J. Guajardo, and C. Paar, "How Well Are High-End DSPs Suited for AES Algorithms?" *Third AES Candidate Conference - AES3,* 2000, available at http://csrc.nist.gov/CryptoToolkit/aes/round2/conf3/aes3papers.html.

[26] RFC 2409, *The Internet Key Exchange (IKE)*, IETF, 1998.

[27] RFC 4306, *Internet Key Exchange (IKEv2) Protocol*, IETF, 2005.

[28] RFC 4109, *Algorithms for Internet Key Exchange, Version 1 (IKEv1)*,

IETF, 2005.
[29] RFC 4307, *Cryptographic Algorithms for Use in the Internet Key Exchange Version 2 (IKEv2)*, IETF, 2005.
[30] RFC 4346, *The Transport Layer Security (TLS) Protocol, Version 1.1*, IETF, 2006.
[31] IEEE Std. 802.11i, *IEEE Standard for Information Technology - Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Networks - Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, Amendment 6: Medium Access Control (MAC) Security Enhancement*, IEEE, 2004.
[32] IEEE Std. 802.16e, *IEEE Standard for Local and Metropolitan Area Networks - Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems - Amendment 2: Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands and Corrigendum 1*, IEEE, 2006.
[33] RFC 2104, *HMAC: Keyed-Hashing for Message Authentication*, IETF, 1997.
[34] FIPS Publication 113, *Computer Data Authentication*, NIST, 1985.
[35] NIST Special Publication 800-38B, *Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication*, NIST, 2005.
[36] RFC 3566, *The AES-XCBC-MAC-96 Algorithm and Its Use With IPsec*, IETF, 2003.
[37] RFC 4434, *The AES-XCBC-PRF-128 Algorithm for the Internet Key Exchange Protocol (IKE)*, IETF, 2006.
[38] X. Wang and H. Yu, "How to Break MD5 and Other Hash Functions," *Advances in Cryptology - Eurocrypt 2005*, LNCS, vol. 3494, Springer, 2005, pp. 19-35.
[39] X. Wang, Y.L. Yin, and H. Yu, "Finding Collisions in the Full SHA-1," *Advances in Cryptology - Crypto 2005*, LNCS, vol. 3621, Springer, 2005, pp. 17-36.
[40] ANSI X9.62, *Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)*, ANSI, 1998.
[41] RFC 3610, *Counter with CBC-MAC (CCM)*, IETF, 2003.
[42] S. Kumar, K. Lemke, and C. Paar, "Some Thoughts About Implementation Properties of Stream Ciphers," *State of the Art of Stream Ciphers Workshop – SASC*, 2004, available at http://www.crypto.ruhr-uni-bochum.de/imperia/md/content/texte/publications/sasc_klp.pdf.

Dong Kyue Kim received the BS, MS, and PhD degrees in computer engineering from Seoul National University in 1992, 1994, and 1999, respectively. From 1999 to 2005, he was an assistant professor in the Division of Computer Science and Engineering at Pusan National University. He is currently an associate professor in the Division of Electronics and Computer Engineering at Hanyang University, Korea. His research interests are in the areas of embedded security systems, crypto-coprocessors, and information security.

Mun-Kyu Lee received the BS and MS degrees in computer engineering from Seoul National University in 1996 and 1998, and the PhD degree in electrical engineering and computer science from Seoul National University in 2003. From 2003 to 2005, he was a senior engineer at ETRI. He is currently with the School of Computer Science and Engineering at Inha University, Korea. His research interests are in the areas of information security and theory of computation.

You Sung Kang received the BS and MS degrees in electronics engineering from Chonnam National University in 1997 and 1999, respectively. He is now pursuing his PhD in electrical and electronic engineering from Korea Advanced Institute of Science and Technology (KAIST). In November 1999, he joined Electronics and Telecommunications Research Institute (ETRI), and he is now a senior engineer. Since 2004, he has been the IT international standard expert of Telecommunications Technology Association (TTA). His research interests include the areas of RFID/USN security, wireless LAN security, cryptographic protocol and network security.

Sang-Hwa Chung received the BS degree in electrical engineering from Seoul National University in 1985, the MS degree in computer engineering from Iowa State University in 1988, and the PhD degree in computer engineering from the University of Southern California in 1993. He was an assistant professor in the Electrical and Computer Engineering Department at the University of Central Florida from 1993 to 1994. He is currently a professor in the Computer Engineering Department at Pusan National University, Korea. His research interests are in the areas of computer architecture and high-performance computer networking.

Won-Ju Yoon is a PhD candidate in the Computer Engineering Department at Pusan National University. He received his BS and MS degrees in computer engineering from Pusan National University in 2002 and 2004, respectively. His research interests include active RFID systems and wireless mesh networks.

Jung-Ki Min received his BS and MS degrees from the Computer Engineering Department of Pusan National University in 2005 and 2007, respectively. He is currently an engineer with the Gate Technologies Company. His research interests are in the areas of information security systems.

Howon Kim received the BS degree in electronic engineering from Kyungpook National University, Daegu, Korea, in 1993, and the MS and PhD degrees in electronic and electrical engineering from Pohang University of Science and Technology (POSTECH), Pohang, Korea, in 1995 and 1999, respectively. From July 2003 to June 2004, he studied at the COSY group at the Ruhr-University of Bochum, Germany. He is currently a senior member of the technical staff with the Electronics and Telecommunications Research Institute (ETRI), Daejeon, Korea. His research interests include RFID technology, sensor networks, and information security. He is a member of the IEEE, IEEE Computer Society, and IACR. He is also an editor of ISO 24791-6 and ITU-T x.rfidsec-1.