E-seal 보안 프로토콜을 위한 Pseudorandom Function의 효율적인 구현[1)]

민정기[1], 강석훈[1], 정상화[1], 김동규[2]

[1]부산대학교 컴퓨터공학과
[2]한양대학교 전자통신컴퓨터공학부

Efficient Implementation of Pseudorandom Functions for the e-seal Protection Protocol

Jung Ki Min[1], Seok Hun Kang[1], Sang-Hwa Chung[1], Dong Kyue Kim[2]

[1]Dept. of Computer Engineering, Pusan National University
[2]Dept. of Electronics and Computer Engineering, Hanyang University

요 약

e-seal은 RFID 기술을 사용하여 원격에서 자동으로 봉인상태를 확인할 수 있는 컨테이너 봉인 장치를 말한다. RFID의 특징상 반도체 칩에 기록된 정보를 제 삼자가 쉽게 판독 및 변조할 수 있다는 취약점이 있는 실정이다. ISO에서는 RFID의 취약점을 보안하기 위한 표준작업(ISO 18185)을 진행 중이다. 이 중, ISO 18185-4는 e-seal에 저장되는 자료나 리더와의 RF통신에서 데이터 보호를 위한 표준이며, 관련된 연구로는 Active-RFID 인증 프로토콜과 ISO 18185-4를 위한 보고서로 제출된 보안 프로토콜 등이 있다. 제안된 e-seal 보안 프로토콜을 적용하기 위해서는 e-seal과 리더 간의 데이터를 암/복호화 키를 Pseudorandom Function(PRF)을 이용하여 마스터 키로부터 MTK(Mutual Transient Key)를 유도하고, MTK를 암/복호화 키로 사용해야 할 필요가 있다. 본 논문에서는 현재 보안 프로토콜에서 사용되고 있는 PRF에 대해 살펴보고, e-seal 환경에서 PRF를 소프트웨어로 구현하였다. 구현 결과 해시 함수를 기반으로 하는 PFR보다 암호화 알고리즘 AES를 기반으로 하는 PRF이 더 좋은 성능을 보였으며, 블록 암호화 알고리즘인 AES-128을 어셈블리어로 구현함으로써 PRF를 최적화하였다.

I. 서론

e-seal은 RFID(Radio Frequency IDentification) 기술을 사용하여 원격에서 자동으로 봉인상태를 확인할 수 있는 컨테이너 봉인 장치를 말한다. e-seal은 미국의 911테러 이후, 국제유통 네트워크를 통한 테러위협 방지에 관심이 높아지면서 그 역할과 관심이 증가하고 있다. 그러나 RFID의 특징상 반도체 칩에 기록된 정보를 제 삼자가 쉽게 판독 및 변조할 수 있다는 취약점이 있는 실정이다.

ISO에서는 화물 컨테이너용 e-seal에 대한 표준을 제정하는 하기 위해 TC104 SC4 WG2를 두고, RFID의 취약점을 보안하기 위한 표준작업(ISO 18185)을 진행 중이다. 이 중, ISO 18185-4는 e-seal에 저장되는 자료나 리더와의 통신에서 데이터 보호를 위한 표준이다. 이와 관련된 연구로는 인증 프로토콜과 ISO 18185-4를 위한 보고서로 제출된 보안 프로토콜 등이 있다.[1, 2, 3]

e-seal 보안 프로토콜을 적용하기 위해서는 e-seal과 리더 간의 데이터를 암/복호화할 키가 필요하다. 하지만, 키 서버를 통해 전달받은 마스터 키를 데이터 암/복호화 키로 바로 사용하는 것은 보안 상의 문제점을 야기할 수 있다. 따라서 PR

1) 이 논문은 교육인적자원부 지방연구중심대학육성사업(차세대물류IT기술연구사업단)의 지원에 의하여 연구되었음.

F(Pseudorandom Function)을 이용하여 마스터 키로부터 MTK(Mutual Transient Key)를 유도하고, MTK를 암/복호화 키로 사용해야 한다.

현재 다양한 보안 프로토콜에서 PRF를 일방향 해시 함수를 기반으로 하는 HMAC[4] 및 블록 암호화 알고리즘을 기반으로 하는 MAC[5, 6, 7]을 이용하여 정의한다[8-12].

본 논문에서는 e-seal을 위한 블록 암호화 알고리즘을 기반으로 하는 MAC과 일방향 해시 함수를 기반(AES[13]-128을 선택)으로 하는 MAC을 소프트웨어로 구현하였다. 성능 측정 결과, C 프로그램에서 AES 기반의 MAC이 해시 함수 기반의 HMAC 보다 좋은 성능을 보여주었다. 따라서 본 논문에서는 AES와 그 기반의 MAC을 어셈블리어를 이용하여 최적화한 결과 C 프로그램에 비해 2배 이상의 성능 향상이 있었다..

이 후, 논문의 구성은 2장에서 HMAC과 블록 암호화 알고리즘 기반의 MAC, AES에 대해 설명하고, 3장에서 구현 내용 및 결과를 제시한다. 그리고 4장에서 결론 및 이후 연구 방향에 대해 기술한다.

II. 기본지식

이 장에서는 해시 함수 기반의 MAC과 암호화 알고리즘 기반의 MAC에 대해서 간략히 설명하고, 대표적인 암호화 알고리즘 AES에 대해 설명한다.

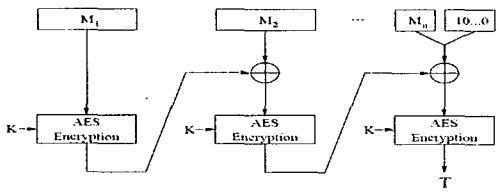

그림 1: CBC-MAC 생성 과정

1. 해시 함수 기반의 MAC(HMAC)

HMAC은 일방향 해시 함수(MD5, SHA1, 등)를 이용하여 메시지 인증 코드를 생성하는 알고리즘이다[4]. HMAC은 해시 함수의 종류에 따라 HMAC-MD5, HMAC-SHA1 등으로 표시한다.

HMAC은 다음과 같이 정의 되어진다.

$$HMAC_K(M) = H((K^+ \oplus opad) \| H((K^+ \oplus ipad) \| M)$$

여기서 H는 사용되는 해시 함수, K는 비밀키, M은 메시지를 뜻한다. 그리고 K^+는 0을 패딩하여 만들어지는 64 바이트 열을 뜻한다.

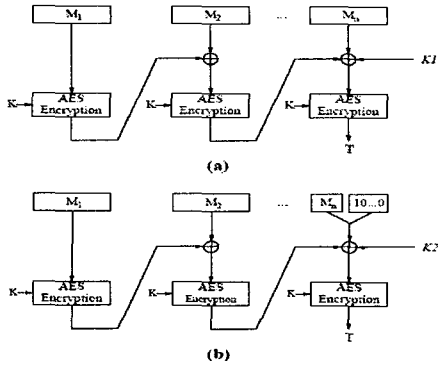

그림 2: CMAC 생성 과정

2. 암호화 알고리즘 기반의 MAC

블록 암호화 알고리즘을 이용하여 MAC을 생성하는 방법에는 여러 가지가 존재한다. 그중 가장 보편적인 방법이 CBC-MAC이다[5]. 그림 1은 AES를 이용한 CBC-MAC을 보여준다. 입력 메시지 M을 128 비트 크기의 n개 블록 ($M = M_1 \| M_2 \| \cdots \| M_n$)으로 나누고 마지막 블록이 128 비트가 되지 않으면 '10...0'으로 패팅한다.

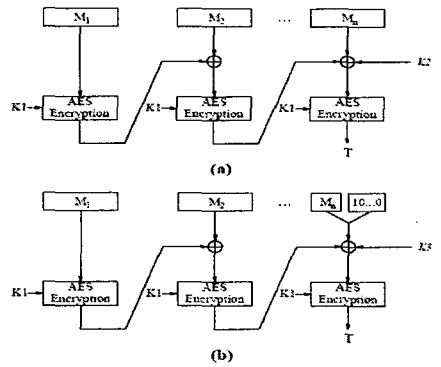

그림 3: XCBC-MAC 생성 과정

본 논문에서 고려하는 다른 두 방법으로는 CMAC과 XCBC-MAC이 있다[6, 7]. 그림 2와 그림 3은 각각의 동작 과정을 보여준다. 두 개의 알고리즘은 CBC-MAC과 유사한 반면, 미묘한 차이를 보인다. CMAC의 경우 마지막 입력 블록 M_n 또는 $M_n \| 10...0$은 이전 암호화 블록 뿐만 아니라 $K1$ 또는 $K2$도 같이 XOR 되어진다. $K1$과 $K2$는 입력 키 K로부터 유도되어지는 서브키(subkey)이다. 만약 마지막 블록이 M_n이면, $K1$이 XOR되고 (그림 2.(a)), 아니면, $K2$가 사용된다(그림 2.(b)). 유사하게 XCBC-MAC은 입력 키 K로부터, $K1$,

표 1: AES-128 소프트웨어 모듈의 성능

		Memory (Bytes)		Time(μsec)	
		Program	Data	Key Expansion	Encryption
C Language		8,334	554	268.0	604.0
Assembly Language	방법 1	4,270	512	105.0	277.1
	방법 2	1,688	512	105.0	302.7
	방법 3	1,660	512	105.0	293.7
	방법 4	1,528	256	103.7	339.2

$K2$, $K3$가 유도되어지고, CMAC의 K, $K1$, $K2$의 자리에 각각 사용되어진다.

3. AES

본 논문에서는 CBC-MAC, CMAC, XCBC-MAC의 블록 암호화 알고리즘으로 AES(Advanced Encryption Standard)를 사용하였다[13]. AES의 암호화 과정은 4개의 연산으로 구성 된다.

- SubBytes 연산 : State를 구성하는 각각의 바이트에 대해 독립적인 비선형 치환(nonlinear substitution)을 수행한다. 여기에 사용되는 치환 테이블을 S-box라 한다.
- ShiftRows 연산 : State의 각 행을 구성하는 바이트의 위치를 교환한다.
- MixColumns 연산 : State의 각 열을 유한체 $GF((2^8)^4)$상의 원소로 정의하고, 다항식 곱셈을 수행한다.
- AddRoundKey 연산 : State의 각 바이트와 라운드 키(Round Key)를 XOR 연산한다.

III. 구현 내용 및 실험 결과

본 장에서는 PRF를 위한 다양한 MAC의 소프트웨어 모듈을 설명한다. Atmel사의 ATmega128 MCU를 타깃 장치로 하였다. ATmega 128 MCU는 RISC 구조를 갖는 마이크로 컨트롤러로 8 비트 데이터 버스와 16 비트 주소 버스를 갖는다. 그리고, 32개의 8비트 범용 레지스터가 있으며, 128KB의 프로그램 메모리와 4KB의 데이터 메모리를 가지고 있다. 또한 ATmega 128의 동작 속도를 8MHz로 설정하고, 실험하였다. 소프트웨어 모듈은 WinAVR(release 20060421)을 사용하여 컴파일 하였다.

AES-128의 C 모듈에서, 처리율 관점을 두고 구현하였다. AES의 10 라운드를 Loop unrolling 하여 처리율을 증가 시켰고, xtime 연산, 즉, $GF(2^8)$상에서의 x 곱셈 연산, 결과와 SubBytes 연산 결과를 테이블 미리 데이터 메모리에 저장하였다. C 모듈의 결과는 표 1의 첫 번째 행에 나타난다.

이 초기 구현을 분석해 보았을 때, C 모듈은 State를 데이터 메모리에 저장하기 때문에 많은 Load/Store 인스트럭션이 수행되는 단점이 있었다. 뿐만 아니라, S-Box와 xtime 연산의 결과를 참조하기 위해 주소를 계산하는데, 많은 사이클을 소비하였다. 이러한 문제점들을 해결하기 위해 어셈블리 언어를 이용하여 State를 메모리가 아닌 레지스터에 저장하고, 테이블의 주소를 고정하여 테이블 참조에 필요한 주소 계산을 줄였다. 추가적으로 다음과 같은 다양한 방법의 구현을 수행하였다.

방법1. C 코드를 직접 어셈블리 코드로 수정

방법2. 프로그램 메모리의 사용량을 줄이기 위해 Loop unrolling을 제거

방법3. SubBytes 연산과 ShiftRows 연산을 하나의 연산으로 통합

방법4. 데이터 메모리의 사용량을 줄이기 위해 xtime 결과 테이블을 제거

표 1에서 어셈블리 언어를 이용한 모듈의 결과를 확인 할 수 있다. 위에서 설명한 4가지 방법 모두 C 코드에 비해 메모리 및 수행 시간 면에서 좋은 결과를 보여주었다. 방법 1과 방법 2에서는 시간-메모리 간의 Trade-off가 발생하였고, 방법 3에서는 방법 2에 비해 메모리와 수행 시간에서 조금 나은 결과를 보였다. 방법 4는 데이터 메모리의 사용량은 줄어들었지만, 수행 시간이 증가하는 결과를 보였다.

표 2은 C 언어를 이용하여 구현한 MAC 모듈의 사용 메모리 크기 및 수행 시간을 보여준다. AES 기반 알고리즘이 해시 함수 기반 알고리즘에 비해 월등히 좋은 성능을 보이는 것을 알 수 있다. MD5와 SHA-1 알고리즘은 32 비트 구조

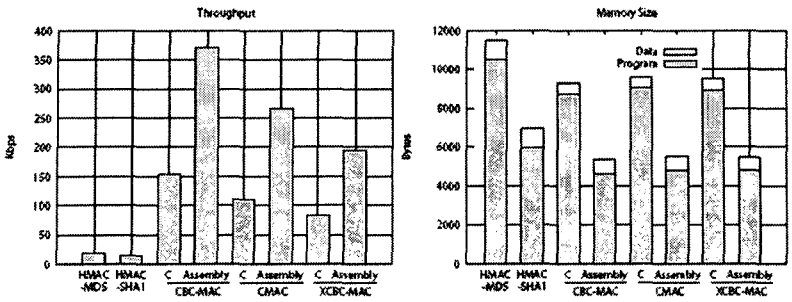

그림 4: 처리율과 메모리 사용량의 비교

에 적합하도록 디자인 되었고, AES는 8 비트 구조에서도 좋은 성능을 보이도록 디자인 되었기 때문이다.

표 2: 다양한 MAC 모듈의 C 프로그램 결과

	Memory (Bytes)		Throughput
	Program	Data	(Kbps)
HMAC-MD5	10,498	982	18.35
HMAC-SHA1	5,950	1,003	13.33
AES-CBC-MAC	8,762	554	154.22
AES-CMAC	9,018	570	111.30
AES-XCBC-MAC	8,938	602	83.39

마지막으로, 그림 4에서 다양한 MAC의 C 모듈 및 어셈블리 모듈의 메모리 사용량과 처리율의 도표로 비교하였다. 그림 4에서 어셈블리 모듈은 가장 좋은 처리율을 보인 방법 1을 선택하였다. 도표의 결과를 보면 AES-CBC-MAC을 어셈블리 어로 구현하였을 때, 가장 좋은 처리율과 가장 작은 메모리 사용량을 보였다.

IV. 결론 및 향후 연구

본 논문에서는, e-seal 환경에 적합한 PRF를 위해 기존의 다양한 보안 프로토콜에서의 PRF를 살펴보고, 각 보안 프로토콜에서 사용하는 MAC을 소프트웨어로 구현하였다. 실험 결과 AES기반의 MAC 중 CBC-MAC이 가장 뛰어난 성능을 보여주었다. PRF의 전력 사용량을 측정하여 고성능, 적은 메모리 사용, 저전력의 PRF 구현을 향후 연구 과제로 남긴다.

참고 문헌

[1] Seongsoo Park, Mun-Kyu Lee, Dong Kyue Kim, Kunsoo Park, Yousung Kang, Sokjoo n Lee, Howon Kim, Kyoil Chung, "Design of an Authentication Protocol for Secure El ectronic Seals", CIS(International Conferenc e on Cybernetics, Informatics and Systemic s)'05 (2005)

[2] 박성수, 이문규, 김동규, 박근수, 김호원, 정교일, "안전한 전자봉인을 위한 인증 프로토콜 설계", 한국정보과학회 추계학술발표대회논문집, 8권 2호, pp.279-282 (2005)

[3] ETRI, "Report of ePP(eSeal Protection Protocol) for ISO 18185-4" (2005)

[4] NIST FIPS PUB 198, "keyed-Hash Messge Authentication Code(HMAC)" (2002)

[5] NIST FIPS PUB 113, "Computer Data Authentication" (1985)

[6] NIST SP800-38B, "Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication" (2005)

[7] IETF RFC 3566: The AES-XCBC-MAC-96 Algorithm and Its Use With IPsec (2003)

[8] IETF RFC 2409: The Internet Key Exchange (IKE) (1998)

[9] IETF RFC 4306: Internet Key Exchange (IKEv2) Protocol (2005)

[10] IETF RFC 4346: The Transport Layer Security (TLS) Protocol Version 1.1 (2006)

[11] IEEE Std 802.11i: IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements (2004)

[12] IEEE Std 802.16e: IEEE Standard for Local and metropolitan area networks (2006)

[13] NIST FIPS PUB 197, "Advanced Encryption Standard(AES)", November 2001