2001년 한국정보처리학회 춘계 학술발표논문집 제8권 제1호

암호키 관리 기반구조 구축을 위한 KMI 라이브러리의 구현 및 성능분석

최강은 *, 임양규*, 오수현*, 김지연**, 권현조**, 원동호*
*성균관대학교 전기 전자 및 컴퓨터공학과
** 한국 정보보호센터 암호기술팀 연구원
e-mail:{kechoi,yklim,shoh,dhwon}@dosan.skku.ac.kr
{jykim, hjkwon}@kisa.or.kr

Implementation and Performance Analysis of KMI Library for Building Key Management Infrastructure

Kang-Eun Choi*, Yang-Kyu Lim*, Soo-Hyun Oh*, Jee-Yeon Kim**,
Hyun-Jo Kwon**, Dong-Ho Won*
*Dept. of Electrical & Computer Engineering,
Sungkyunkwan University
**Cryptographic Technology Team, KISA

요약

최근 세계 각 국에서는 암호 기술의 역기능을 방지하고 일반 사용자들 사이에 암호 사용을 활성화하기 위한 방안으로 암호키 관리 기반 구조(KMI)에 관한 연구가 활발히 진행되고 있으며 이를 구현한 시스템들도 개발되고 있다. 그러나 현재 여러 나라에서 이러한 키 복구 기술에 대한 연구와 함께 키 복구 시스템들이 개발되고 있는 것에 반해 국내에서는 아직까지 암호키 관리 기관 구조를 구축할 수 있는 실제 시스템들에 대한 연구 및 개발이 미흡한 실정이다. 본 논문에서는 암호키 관리 기반 구조 구축에 기반이 될 수 있는 KMI 라이브러리를 구현하고 성능을 분석하였다.

1. 서론

암호화 기술은 제 삼자로부터 기밀성을 유지할 수 있다는 장점이 있는 반면에 범죄 단체 등에 의해 악용되어 사회의 안전을 해칠 수 있을 뿐 아니라, 정당한 사용자라 할지라도 키를 분실하거나 손상된 경우에는 암호문으로부터 평문을 복호할 수 없는 등의 문제점을 가지고 있다. 따라서 암호화 기술이 실제 네트워크 상에서 일어나는 전자 상거래와 같은 응용에 활용되기 위해서는 이러한 문제를 해결하기 위한 연구가 반드시 필요할 것이다.

최근 세계 각 국에서는 이러한 암호 기술의 역기능을 방지하고 일반 사용자들 사이에 암호 사용을 활성화하기 위한 방안으로 암호키 관리 기반 구조(KMI : Key Management Infrastructure)에 관한 연구가 활발히 진행되고 있으며 이를 구현한 시스템들도 개발되고 있다.[1)]

그러나 현재 여러 나라에서 이러한 키 복구 기술에 대한 연구와 함께 키복구 시스템들이 개발되고 있는 것에 반해 국내에서는 아직까지 암호키 관리 기반 구조를 구축할 수 있는 실제 시스템들에 대한 연구 및 개발이 미흡한 실정이다.

따라서 본 논문에서는 KMI 구축에 필요한 요소 기술들을 분석해서 KMI 라이브러리를 개발한 후, 구현한 라이브러리의 성능을 분석하였다.

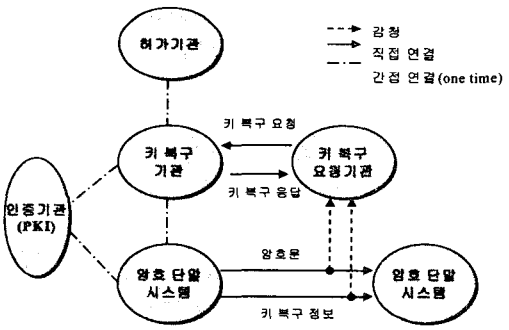
본 논문의 구성은 다음과 같다. 2장에서는 연구의 배경이 되는 KMI의 개요에 대해서 간략히 살펴본 후, 3장에서는 KMI라이브러리의 구조, 모듈 설계 및 특징에 대해 설명한다. 4장에서는 개발한 라이브러리의 성능을 분석하고, 5장에서 결론을 맺는다.

†본 논문은 한국정보보호센터의 " 암호키 관리 기반구조 구축을 위한 KMI라이브러리 개발(과제번호: 암호기술연구 00-10)" 과제의 일부로 수행된 결과임

2. KMI의 개요

PKI가 구축되고 암호 사용이 일반화되고 다양한 보안 서비스가 가능해지면서 여러 가지 문제가 발생하게 되었다. 이런 암호 사용의 부작용에 대한 해결책으로써 KMI의 필요성이 대두되었다. KMI는 암호화에 사용되는 키를 필요시에 정당한 권한을 가진 키 복구 요청기관(예 : 법 집행기관, 일반 기업, 개인 사용자)이 복구할 수 있도록 하는 키 복구 기술을 핵심으로 하며, 암호화 키를 복구하기 위한 목적으로 암호화 키에 대한 관리 업무를 수행하는 하드웨어, 소프트웨어, 인력, 정책 등의 집합체라고 정의할 수 있을 것이다. PKI가 암호에 필요한 공개키를 그 관리 대상한다면 KMI는 암호화에 사용되는 공개키와 비밀키를 관리 대상으로 한다.

KMI의 구성요소는 위탁 방식, TTP방식, 캡슐화 방식등 메커니즘에 따라서 다르게 구성할 수 있으나, 대체로 <그림 1>과 같이 구성할 수 있으며, PKI, 암호단말 시스템, 키 복구 기관 및 키 복구 요청기관은 공통적인 구성요소이다.



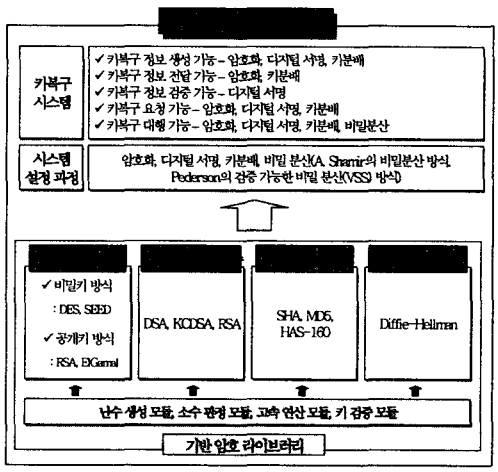
<그림 1> KMI의 구성 요소

3. KMI 라이브러리 설계

3.1 KMI 라이브러리의 구조

2장에서 언급한 키 복구 시스템의 다섯 가지 기능을 구현하기 위해서는 암호화, 디지털 서명, 키분배, 해쉬 알고리즘과 같은 기반 암호 시스템이 필요하며 이를 이용하여 실제 키 복구 시스템을 구축하기 위한 설정 과정에서 비밀 분산 프로토콜이 필요하다. 그리고 암호화, 디지털 서명, 키분배 및 해쉬 알고리즘과 같은 기반 암호 알고리즘은 난수 생성 알고리즘, 소수 판정 알고리즘, 모듈러 멱승과 곱셈의 역원 계산 등을 포함하는 고속 연산 알고리즘과 같은 공통적인 세부 기술들을 이용하였다.

따라서 KMI 구축을 위한 라이브러리 개발에 필요한 구성 요소들은 기본 기능 모듈과 기반 암호 라이브러리로 나눌 수 있다. 구현한 라이브러리의 구성요소는 <그림 2>와 같다.



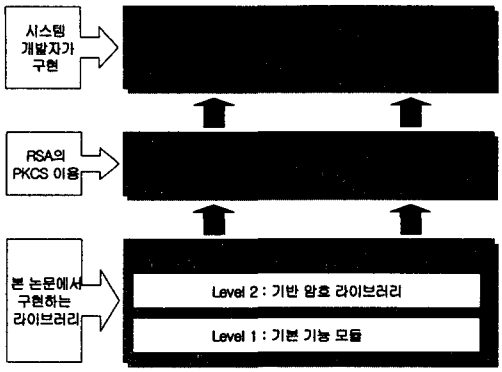
<그림 2> KMI 라이브러리의 구성요소

3.2 KMI 라이브러리의 모듈 설계

KMI 모듈 설계 시 라이브러리는 시스템 개발자들에게 사용하기 쉬운 인터페이스를 제공하기 위해 CAPI (Cryptographic Application Programming Interface)에 기반하여 개발하였다. CAPI란 시스템 개발자들이 이미 구현되어 있는 암호 라이브러리의 함수나 메소드에 대한 입출력 파라미터, 형식 선언문 및 전역 변수 등에 대한 지식만으로도 쉽게 보안 시스템을 구축할 수 있도록 하기 위한 보안 서비스 API를 말한다.

구현한 라이브러리는 CAPI 중 RSA사에서 개발한 공개키 암호 방식의 표준인 PKCS #11 Cryptography Token Interface Standard에 따라 개발하였다. <그림 3>은 시스템 개발자들이 KMI 구축시 CAPI를 통해서 구현한 라이브러리를 사용하는 구조를 나타낸 것이다.

구현한 라이브러리의 모듈 중에서 DES, RSA 암호 알고리즘, RSA, DSA 디지털 서명 알고리즘, MD5, SHA-1 해쉬 알고리즘과 Diffie-Hellman의 키 분배 알고리즘은 PKCS#11에 정의되어 있으므로 각



<그림 3> CAPI를 이용한 암호키 관리 기반구조의 구축

모듈의 함수명, 변수명 및 파라미터를 그대로 사용하였다. 그러나, PKCS#11에 정의되어 있지 않은 모듈인 SEED, KCDSA 디지털 서명 알고리즘, HAS-160 해쉬 알고리즘 및 비밀 분산 알고리즘은 PKCS#11을 참고로 하여 자체적으로 설계하였으며 난수 생성, 소수 판정, 고속 연산 모듈과 같은 기반 모듈들 역시 PKCS#11을 참고하여 자체적으로 설계하였다. 구현한 라이브러리에서 사용한 암호 알고리즘, 디지털 서명, 해쉬 알고리즘, 키 분배 알고리즘 및 비밀 분산 알고리즘의 각 모듈은 [표 1]과 같다.

[표 1] 구현한 각 모듈별 함수 및 기능 설명

	함 수 명	
해쉬 함수	· C_MD5_Init · C_SHA_Init · C_has160_Init	해쉬 함수의 초기화
	· C_MD5_Update · C_SHA_Update · C_has160_Update	데이터의 해쉬
	· C_MD5_Final · C_SHA_Final · C_has160_Final	해쉬 함수의 종료
	· C_MD5_Round · C_SHA_Round · C_has160_Round · C_has160_MesExpand	해쉬 함수의 각 라운드 과정 및 메시지 확장
키 분배	· C_DH_check	생성된 파라미터의 유효성 검사
	· C_DH_generate_key	키 분배에 필요한 비밀키와 공개키를 생성·계산
	· C_DH_compute_key	공유키를 생성해내는 함수

[표 1] 구현한 각 모듈별 함수 및 기능 설명(계속)

	함 수 명	
암호 / 복호	· C_des_cbc_encrypt · C_des_ecb_encrypt · C_seed_ecb_enc · C_seed_cbc_enc	DES 및 SEED의 암호화 및 복호화
	· C_RSA_public_encrypt · C_elgamal_enc	RSA 및 ElGamal 암호화 함수
	· C_RSA_private_decrypt · C_elgamal_dec	RSA 및 ElGamal 복호화 함수
	· C_RSA_padding_add_PKCDS1_type_1	메시지에 패딩을 하는 함수, PKCS#1에 정의
서명 / 검증	· C_RSA_private_encrypt · C_DSA_sign · C_KCDSA_sign	RSA, DSA, KCDSA에 대한 서명 함수
	· C_RSA_public_decrypt · C_DSA_verify · C_KCDSA_verify	RSA, DSA, KCDSA에 대한 검증 함수
비밀 분산	· C_GenerateShare	Shamir 방식에서 비밀정보에 대한 n개의 부분 정보 생성
	· C_InputsOK	비밀 복원 단계를 수행하기 위한 인자들을 검사하는 함수
	· C_CreatedOptput	부분 정보 파일을 생성하는 함수
	· C_RecoverSecret	비밀 정보 복원하는 함수
	· C_Process	비밀 분산 위한 연산 함수

3.3 구현한 라이브러리의 특징

구현한 라이브러리를 이용하여 시스템 개발자들이 실제 KMI를 구축하고자 할 때, 기반 암호 모듈 및 암호 알고리즘, 해쉬 함수, 디지털 서명, 키 분배 등에 관련된 내부 함수에 대한 자세한 지식 없이도 용이하게 구축할 수 있도록 함수 내부와의 인터페이스를 제공해주는 인터페이스 모듈을 별도로 만들었다.

또한, 비밀 분산 모듈을 추가함으로써 키 위탁 기관의 수와 복구 시에 필요한 위탁기관의 수를 개발자가 결정할 수 있으며, 구현한 라이브러리는 PKI와 연동 가능한 특징을 가지고 있다.

4. 구현한 라이브러리 성능 평가

구현한 라이브러리는 CPU 800MHz, 메모리 256M의 Intel Pentium Ⅲ(Windows 98)에서 실행속도를 측정하였으며. 사용 컴파일러는 Microsoft사의 Visual C++6을 사용하였다.

해쉬 함수와 관용 암호 알고리즘은 입력으로 사용된 파일의 크기와 시간 단위로 결과를 표현하였으며, 공개키 암호 알고리즘과 디지털 서명 알고리즘은 오퍼레이션 당 걸리는 시간에 중점을 두고 테스트를 수행하였다. 구현한 라이브러리의 성능 분석 결과는 [표 2], [표 3]과 같다.

[표 2] 해쉬 함수와 관용 암호 성능 분석

Algorithm	Byte Processed	Time Taken	MegaByte /Second
MD5	99,149,322	1.71 Sec	58.0
SHA-1	99,149,322	2.81 Sec	35.3
HAS-160	99,149,322	6.15 Sec	16.1
DES(암호화)	4,347,940	0.54 Sec	8.1
DES(복호화)	4,347,940	0.54 Sec	8.1
SEED(암호화)	22,690,624	11.74 Sec	1.933
SEED(복호화)	22,690,624	11.72 Sec	1.936

[표 3] 공개키 암호 및 디지털 서명의 성능 분석

Algorithm	Iterations	Total Time	MS/ Operation
RSA-512(암호화)	10000	3.13	0.313
RSA-1024(암호화)	10000	8.68	0.868
ElGamal-512(암호화)	10000	58.4	5.84
RSA-512(복호화)	10000	46.95	4.95
RSA-1024(복호화)	10000	295.5	29.55
ElGamal-512(복호화)	10000	30.6	3.06
RSA-512(서명)	10000	45.8	4.58
DSA-512(서명)	10000	50.0	5.0
KCDSA(서명)	10000	58.8	5.88
RSA-512(검증)	10000	30.2	3.02
DSA-512(검증)	10000	60.4	6.04
KCDSA(검증)	10000	74.1	7.41

5. 결론

최근 들어 인터넷을 통한 전자 상거래가 점점 활성화되고 있으며 이러한 네트워크 상에서의 거래에서 개인 정보에 대한 기밀성을 제공할 수 있는 암호 기술의 사용은 필수적인 요소가 되었다. 따라서 일반 사용자들 사이에서 암호 기술의 사용을 대중화하기 위해서는 암호키 관리 기반 구조의 구축이 필요하며 이에 기반이 될 수 있는 KMI 라이브러리 개발이 요구된다. 따라서 본 논문에서는 시스템 개발자들에게 시스템 구축시 사용이 용이한 인터페이스를 제공하며 공개키 기반 구조와도 연동 가능한 KMI 구축을 지원할 수 있는 라이브러리를 구현한 후, 성능분석을 하였다. 구현한 라이브러리가 국내 암호키 관리 기반 구조를 구축할 수 있는 실제 시스템들의 개발에 있어서 기초 자료로 활용될 수 있도록 기대한다.

참고문헌

[1] A. Shamir, "How to share a secret," Communication of the ACM, 22 pp. 612- 613, 1979

[2] "Data Encryption Standard," National Bureau of standard Federal information processing standard publication 46, 1977

[3] http://www.rsasecurity.com/rsalabs/pkcs

[4] P. Feldman, "A Practical Scheme for verifiable secret sharing," Proc. 28th Focs. pp. 427-438, 1987

[5] R. L. Rivest, A.Shamir and L. Adleman, "A method of obtaining digital signature and public key system," Communication of the ACM, pp. 120- 126, Feb. 1978

[6] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithm," IEEE Trans. on Information Theory IT-31, pp. 469-472, 1995

[7] T.P. Pederson, "Non-Interactive and Information Theoretic Secure Verifiable secret sharing ," IEEE Trans. on Information Theory IT-31, pp. 469-472, 1995

[8] 한국정보통신기술협회, "부가형 전자서명 방식 표준-제2부: 인증서 기반 전자서명 알고리즘 (KCDSA)"

[9] 한국정보통신기술협회, "해쉬 함수 표준-제2부 : 해쉬 함수 알고리즘(HAS-160)"

[10] 한국정보통신기술협회, "128 비트 블록 암호 알고리즘 표준(SEED)"