안전한 키보호 기능을 제공하는 암호 API

김명희°, 전문석
안철수연구소°, 숭실대학교
cryptohee@hanmail.net°, mjun@computing.ssu.ac.kr

A cryptographic API to provide the secure key protection function

Myung-Hee Kim°, Moon-Seog Jun
AhnLab, Inc. °, Dept. of Computer, Soong-Sil University

요 약

국내 전자상거래 제품과의 호환성과 확장성을 위하여 국내 전자서명 표준인 KCDSA(Korean Certificate-based Digital Signature Algorithm) 메커니즘을 PKCS(Public Key Cryptographic Standard) #11 암호 API(Application Programming Interface)에 기능을 추가한다. PKCS #11에서 정의한 키관리(Key Management) 함수의 입력 파라미터에 암호화할 키를 바로 입력하면 변조된 키를 전달할 수 있으므로, 본 논문에서는 안전한 키보호(Key Protection) 함수를 새로 정의하여 암호화할 키 대신 사용자 PIN(Personal Identification Number: 패스워드) 입력하여 사용자의 KCDSA 개인키와 공개키를 보다 더 안전하게 보관하고자 한다.

1. 서 론

오늘날 많은 전자상거래 제품 개발에 호환성과 확장성 표준을 제시하고 있는 PKCS는 전자상거래 인증기관과 제품간의 상호 운용이 가능하도록 데이터 형식, 통신 프로토콜, 암호 API 등에 대한 기본을 규정한다. PKCS는 #1부터 #15까지 규정되어 있으며, 그 중에서 PKCS #11은 메일 프로그램, 인증기관 소프트웨어에서의 키관리 등에 사용되고 있다. PKCS #11에 KCDSA 메커니즘에 의한 전자서명 생성과 검증 서비스를 추가하므로써, 국내 전자상거래 제품 개발에 많은 호환성과 확장성을 제공할 수 있게 된다.

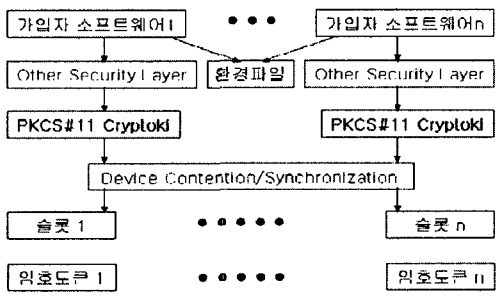
전자상거래 시스템은 키가 안전하게 보존된다는 가정하에서 시스템의 안전성이 보장되는데, 현재 키의 안전한 관리가 미흡한 상태이다. PKCS #11 암호 API에서 암호화할 키를 키관리 함수의 입력 파라미터에 바로 입력하면 변조된 키를 전달할 수 있으므로, 본 논문에서 정의한 안전한 키보호 함수에 암호화할 키 대신 사용자 PIN을 입력하여 사용자의 개인키와 공개키를 보다 더 안전하게 보관할 수 있도록 한다.

본 논문의 구성은 다음과 같다. 2장에서 관련연구로 PKCS #11과 키관리 함수를 살펴본다. 3장에서 안전한 키보호 기능을 제공하는 암호 API와 제안한 키보호 함수를 이용한 KCDSA 서명 생성 과정을 연구한다. 4장에서 분석하고, 마지막 5장에서 결론을 맺는다.

2. 관련연구

2.1 PKCS #11

PKCS #11의 기본 모델 구조는 [그림 1]과 같다.[1]



[그림 1] PKCS #11 기본 모델

암호토큰(Cryptographic Token)은 암호관련 정보를 저장하거나 암호기능을 수행할 수 있는 논리적 관점의 디바이스이며 사용자 개인키와 인증서를 여기에 저장한다. 슬롯(Slot)은 암호토큰과의 입출력을 수행할 수 있는 논리적 리더기이며, 다

중 쓰레드(Multi Thread)의 동기화는 뮤텍스(Mutexes) 기법으로 지원한다. 메커니즘(Mechanism)은 암호기능을 구현하는 절차이며, PKCS #11은 AES, RSA, DSA 메커니즘 등을 지원한다. 본 논문에서 사용하는 PKCS #11 라이브러리는 국내 전자상거래의 활성화와 호환성을 위하여 국내 전자서명 표준인 KCDSA 메커니즘도 지원하도록 한다.

2.2 키관리 함수

PKCS #11에서 정의한 키관리 함수에는 C_GenerateKey, C_GenerateKeyPair, C_WrapKey, C_UnwrapKey, C_DeriveKey 등이 있다. C_GenerateKey 함수는 비밀키(Secret Key)를, C_GenerateKeyPair 함수는 개인키(Private Key)와 공개키(Public Key)를 생성한다. C_WrapKey 함수는 비밀키 또는 개인키를 암호화하고, C_UnwrapKey 함수는 암호화된 키를 복호화한다. C_DeriveKey 함수는 기본키(Base Key)로부터 새로운 키를 생성한다. 이 중에서 키를 암호화하는 C_WrapKey 함수는 개인 데이터 노출시 가장 위험한 비밀키와 인증서 등을 암호화하여 안전하게 보관하는데 사용된다.([1], [2])

```
CK_DEFINE_FUNCTION(CK_RV, C_WrapKey)
(
    CK_SESSION_HANDLE   hSession,
    CK_MECHANISM_PTR    pMechanism,
    CK_OBJECT_HANDLE    hWrappingKey,
    CK_OBJECT_HANDLE    hKey,
    CK_BYTE_PTR         pWrappedKey,
    CK_ULONG_PTR        pulWrappedKeyLen
)
```

여기서, hSession은 세션 핸들이고, pMechanism은 암호 메커니즘에 대한 포인터이다. pWrappingKey는 암호화키의 핸들이고, pKey는 키를 암호화하기 위한 키의 핸들이다. 그리고, pWrappedKey는 암호화된 키를 보관하는 장소에 대한 포인터이고, pulWrappedKeyLen은 암호화된 키 길이를 보관하는 장소에 대한 포인터이다.

C_WrapKey 함수는, (1) 비밀키를 hWrappingKey에 입력하고 이것을 암호화하거나 복호화하는 공개키를 hKey에 입력하는 경우, (2) 비밀키를 hWrappingKey에 입력하고 이것을 암호화하거나 복호화하는 다른 비밀키를 hKey에 입력하는 경우, (3) 개인키를 hWrappingKey에 입력하고 이것을 암호화하거나 복호화하는 비밀키를 hKey에 입력하는 경우 중에서 사용할 수 있다.

3. 제안한 키보호 함수

3.1 안전한 키보호 기능을 제공하는 암호 API

2장 2.2절에서 살펴본 C_WrapKey 함수의 hKey에 키가 그대로 입력되면 개인키 및 공개키에 악의의 영향을 줄 수 있다 본 논문에서는 새로운 키보호 함수 C_SecureWrapKey를 정의하므로써 사용자 키를 보다 더 안전하게 보관하고자 한다. hKey에 키를 입력하는 C_WrapKey 함수와는 달리 사용자가 PIN을 바로 입력하여 생성된 키로 개인키 및 공개키를 암호화하는 것이다. 암호화하는 키를 제 3 자는 알지 못하므로 키를 안전하게 보관할 수 있게 된다.

```
CK_DEFINE_FUNCTION(CK_RV, C_SecureWrapKey)
(
    CK_SESSION_HANDLE   hSession,
    CK_MECHANISM_PTR    pEncryptionMechanism,
    CK_MECHANISM_PTR    pDerivationMechanism,
    CK_OBJECT_HANDLE    hWrappingKey,
    CK_UTF8CHAR_PTR     pPin,
    CK_ULONG            ulPinLen,
    CK_BYTE_PTR         pWrappedKey,
    CK_ULONG_PTR        pulWrappedKeyLen
)
```

여기서, hSession은 세션 핸들이고, pEncryptionMechanism은 키 암호 메커니즘에 대한 포인터이다. pDerivationMechanism은 사용자가 입력한 PIN으로부터 키를 생성하는 메커니즘에 대한 포인터이고, pWrappingKey는 암호화할 키의 핸들이다. pPin은 PIN에 대한 포인터이고, ulPinLen은 PIN 길이에 대한 포인터이다. pWrappedKey는 암호화된 키를 보관하는 장소에 대한 포인터이고, pulWrappedKeyLen은 암호화된 키 길이를 보관하는 장소에 대한 포인터이다.

3.2 제안한 함수를 이용한 KCDSA 서명 생성 과정

본 논문에서 사용하는 PKCS #11 라이브러리는 메커니즘의 포인터 CK_MECHANISM_PTR hMechanism에 전자서명 방식

으로 RSA, DSA, KCDSA 중 선택할 수 있도록 국내 전자서명 표준인 KCDSA 메커니즘을 추가한다.

3장에서 제안한 키보호 함수 C_SecureWrapKey를 이용한 KCDSA 전자서명 생성 과정은 다음과 같다. PKCS #11 라이브러리가 지원하는 함수 포인터들을 CK_FUNCTION_LIST 구조체에 가져온다.(C_GetFunctionList) config 파일이 있는지 검색하여 있으면 환경변수를 먼저 가져온다. 토큰 라이브러리를 초기화하고, 토큰에 대한 슬롯을 등록한다. 그리고, 객체들을 초기화한다.(C_Initialize) 시스템의 모든 슬롯 리스트를 획득한다.(C_GetSlotList) 각각의 슬롯에 대한 메커니즘 리스트를 획득한다.(C_GetMechanismList) 변수들을 초기화한 후에, Legacy 체크를 위해 CKF_SERIAL_SESSION 플래그가 TRUE로 설정되어 있는지 확인한다. CKF_SERIAL_SESSION 플래그가 TRUE로 설정되어야만 Parallel 지원이 가능하다. 동기화를 지원하기 위해 뮤텍스를 생성하고, 슬롯을 통해 토큰을 가져온다. KCDSA 메커니즘을 지원하는지 확인한 후에 새로 생성한 세션의 핸들에 세션의 정보와 데이터를 설정한다.(C_OpenSession) 현재 생성된 세션이 있는지 검색하여 그 세션에 대한 정보로 키 객체를 찾는다.(C_GetAttributeValue) KCDSA 개인키 객체 템플릿의 토큰[3]과 세션을 검색하기 위해 초기화한다.(C_FindObjectInit) KCDSA 개인키 객체 템플릿을 검색한다.(C_FindObject) KCDSA 개인키 객체 템플릿 검색을 종료한다.(C_FindObjectFinal) 사용자 PIN을 입력하여 암호토큰에 로그인한다.(C_Login) KCDSA 개인키 객체 템플릿, 3절에서 정의한 키보호 함수 C_SecureWrapKey로 암호화된 키를 복호화하기위한 사용자 PIN, 그리고 서명하고자 하는 데이터를 입력하여 KCDSA 메커니즘으로 서명하기 위해 초기화한다.(C_SignInit) 초기화한 후에, KCDSA 메커니즘으로 서명을 생성한다.(C_Sign) 뮤텍스를 생성하여 세션의 엔트리를 제거하므로써 세션을 종료한다.(C_CloseSession) 모든 세션을 해제하면서 PKCS #11 라이브러리를 종료한다.(C_Finalize)

4. 제안한 키보호 함수 추가에 대한 분석

PKCS #11 암호 API에 안전한 키보호 함수를 새로 추가하므로써 다음과 같은 장점을 가질 수 있다.[4]

- **다양한 응용프로그램 지원** : 현재 작성되고 있는 다양한 전자상거래 응용프로그램 뿐만 아니라 향후 제시될 전자상거래 응용 프로그램에서도 다양한 메커니즘을 제공할 수 있다. 국내 전자서명 표준 KCDSA 메커니즘을 추가하므로써 국내외 전자상거래 인증기관과 제품간의 상호운용이 가능하다.
- **키 보안성** : PKCS #11 암호 API는 입력 파라미터에 키를 바로 입력하므로 안전한 키가 입력되는지에 대한 확인이 필요하다. 본 논문에서 제안한 안전한 키보호 함수에서 키 대신 사용자 PIN을 직접 입력하므로써 사용자 개인키와 공개키에 대한 보안성을 강화한다.
- **API 이용한 프로그래밍 안전성** : 개발자가 암호 API를 사용하여 개발하는 경우에, 잘못된 키보호 함수를 사용하면 보안상의 문제가 야기될 수 있다. 본 논문에서 제안한 함수를 공통적으로 사용하므로써 키 보안성 뿐만 아니라 API 이용한 프로그래밍 안전성 또한 제공할 수 있다. 일관성있는 이름 부여, 부주의로 인한 정보 유출을 방지하기 위한 정보 은닉, 암호 API의 일관성있는 순서 등을 제공하여 개발자가 실수할 확률을 줄여주는 것이다.

5. 결 론

본 논문에서는 PKCS #11 암호 API에 KCDSA 메커니즘을 추가하여 국내 전자상거래 인증기관과 제품간의 호환성을 지원하였다. 키관리 함수 C_WrapKey는 키가 파라미터에 바로 입력되어 키의 노출에 대한 위험이 있으므로 사용자 PIN을 입력하는 안전한 키보호 함수 C_SecureWrapKey를 본 논문에서 제안하였다. 제안한 키보호 함수를 추가하므로써 PKCS #11 암호 API가 더욱 안전하고 기능적으로 향상되었음을 살펴보았다. 향후 연구과제로 본 논문에서 제안한 암호 API가 표준으로 성립되어, 개발자의 API 잘못 사용에 따른 위험성을 줄이고자 한다.

참 고 문 헌

[1] RSA Laboratories, "PKCS #11 v2.20: Cryptographic Token Interface Standard - Draft 5", 2004.

[2] Jolyon Clulow, "On the Security of PKCS #11", CHES 2003, LNCS 2779, 2003.

[3] 김명희, 전문석, "PKCS #11에 기반한 KCDSA 메커니즘 설계", 한국정보처리학회 춘계학술발표논문집, 제 11 권, 제 1 호, pp. 1015-1018, 2004.

[4] R.L.Rivest, M.E.Hellman, J.C.Anderson, "Response to NIST's proposal", Comm. ACM, 35(7), pp. 41-52, 1992.