通信情報保護學會誌
第8卷 第4號. 1998. 12

효율적인 정보보호를 위한 상용 암호 서비스 기술(CryptoAPI중심)

김 점 구*, 김 석 우**, 강 창 구***, 이 재 광*

Technology of Common Cryptographic Service for Efficient Information Security

요 약

본 논문은 기존 다양한 암호 알고리즘과 각종 정보 서비스용 응용 소프트웨어의 결합이 1:1로 결합됨으로써 새로운 암호 서비스의 요구나 재사용 요구에 부응할 수 없는 결합형 암호 서비스의 문제점을 해소할 수 있도록 한 CAPI들을 분석하고, 특히 Windows 환경에 적합한 CryptoAPI에 대한 기능적인 내용을 분석하였다.

1. 서 론

초고속 정보통신망 NII(National Information Infrastructure)에 의해 폭발적으로 관심을 집중시킨 정보화사회 기반 구축은 전자문서, 전자상거래 등의 정보서비스 활용을 가능케 하고 있다. 과거 수동시대에 필요했던 물리적 보안과 역기능 방지 수단은 전자 시대의 정보보호 기능 구현을 필요로 하게 되었다.

그러나, 정보 서비스 응용 프로그램이 필요로 하는 암호학적 기능을 구현하는 방법으로써 개발자들이 사용한 방법인 응용 프로그램과 암호 알고리즘의 1:1 결합은 새로운 암호서비스 요구나 알고리즘의 변경 시 재사용을 할 수 없다는 단점이 있어, 1993년 GSS-API (Generic Security Service API)가 IETF로부터 표준화 암호 응용 인터페이스로 인증됨으로써 이의 구현과 보완 발전이 진행되고 있다.[1][7]

다양한 암호 알고리즘과 각종 정보 서비스용 응용 소프트웨어, 그리고 멀티미디어 정보 자원은 과거의 결합형 암호서비스 요구로부터 개방화된 n:m의 선택적 연결을 필요로 하게 되어, 마이크로소프트, 인텔 등의 대규모 상업회사와 국가기관 등이 암호 서비스 API와 구조를 개발하여 자사 또는 고유 활용분야 용 CAPI(Cryptography API)를 제품으로 개발하여 사용하고 있다.

국내의 경우, 1994년 5월 국무총리를 위원장으로 하는 초고속 정보화 추진 위원회를 구성, 범 정

* 한남대학교 컴퓨터공학과
** 한세대학교 정보통신학과
*** 한국전자통신 연구원

부 차원에서 정보 고속도로를 구축하고 각 기관과 개인에게 새로운 서비스 실현과 첨단기술을 확보키로 하여 2010년대 초고속 정보통신망 기반 구축을 추진 중에 있다. 정보보호분야도 1995년 이후 소규모 정보보호 회사가 출범하여, 현재에는 한국 표준 암호 알고리즘으로 해시 및 디지털 서명 알고리즘이 채택되는 과정 중에 있다.[18]

이와 같은 국내외적 추세는 인터넷과 정보 서비스에 암호 서비스의 필요성을 더 더욱 증가시킬 것이며, 개방화 요구와 개별적 보안 요구가 상충하게 되는 환경하에서 각종 정보의 수검, 가공, 분배가 이루어지게 될 것이다.

21세기 개방화된 분산환경에서 이러한 정보 서비스를 수용할 수 있는 암호 서비스 API의 연구는 전자상거래 등을 구현하는 하부 커널 구조로써 우선적으로 실현되어야 하므로 국내의 실정에 비추어 시급한 연구 분야라 할 수 있다.[19] 따라서, 정보 서비스 응용 프로그램의 활발한 활용을 위해서 개발되거나 개발중인 상용 암호 API를 심도 있게 분석하여 국내 적용 모델을 개발함으로써 적기 21세기 전자 문명 시대에 대비한 정보보호 기반 구축을 마련해야 하겠다.

본 논문에서는 제 2장에 상용 암호 서비스 기술들의 종류와 특징을 비교하고, 3장에서는 마이크로 소프트사의 CryptoAPI의 대한 분석 결과를 요약하였다.

2. 상용 암호 서비스 기술

컴퓨터 네트워크를 이용한 분산 응용 서비스가 증가하고, 네트워크를 이용한 응용 서비스의 보안 위협 요소가 증가함으로써, 위협 요소를 방어하기 위한 보안 기술의 필요성이 절실하게 요구되어지고 있다. 따라서 이러한 문제점을 고려한 응용 소프트웨어는 암호학적 기능을 견고하게 결합하는 것이 무엇보다 중요하다고 할 수 있을 것이다.[2]

그러나 이와 같은 접근 방법은 응용 소프트웨어와 암호학을 결합하는데 있어 별도의 새로운 개발 노력이 필요하며, 소프트웨어의 재사용과 상업적 상품으로서 기대되는 모듈화와 내구성을 지원하지 못한다. 또한, 분산 개방 환경의 특성에 따른 보안 위협 요소 증대로 인하여 응용 소프트웨어 개발자들은 응용과 암호 모듈을 각자가 개발하여 정합함으로써 암호기능의 호환성이 결여되고, 개발 노력의 중복으로 인한 투자 손실로 이어지는 문제점을 낳고 있다.[2][12]

따라서 다양한 개발 환경과 분산 응용에 공통으로 적용할 수 있는 암호 기술과 이의 효과적인 정합 방법으로 더 많은 유연성과 강력한 대안을 제공할 수 있는 표준화된 암호 응용 인터페이스(CAPI:Cryptographic Application Program Interface)를 사용하는 것이다.[2][17]

현재 국제적인 상용 암호화 기술 기반 표준으로 1992년GSS-API 를 Linn이 IETF에 상용 정보보호 API를 제안하여 1993년에 IETF로부터 인증 받아 RFC1508 및 1509로 문서화 한 후 많은 단체 및 기업체로부터 새로운 CAPI가 개발되어 표준화가 진행되고 있다.[7]

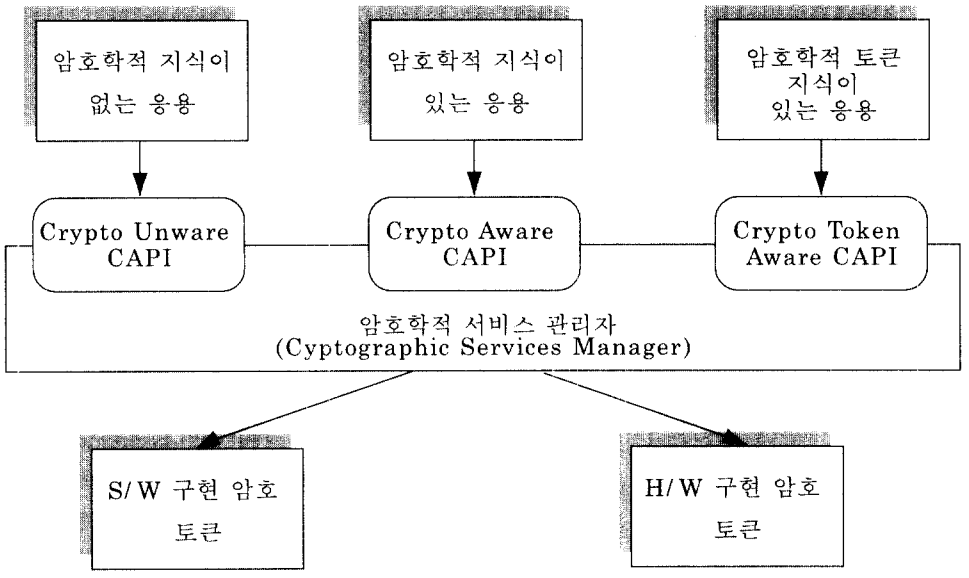
2.1 상용 암호 서비스 기술 요구사항

CAPI 개발자들은 응용이 접근을 요구하는 보안 서비스 수준을 기술하기 위하여, 암호학적 지식의 개념을 사용하였다. 이 개념은 운영체제가 허용하는 암호 서비스 접근의 수준을 포함하기 위하여 확장될 수 있다. 목표는 스프레드시트, 워드프로세서 및 전자우편 등과 같은 일반 목적의 응용들이 암호 지식이 없이도 이용할 수 있으며, 기초적인 암호학과 암호 지원 즉, 보증서 관리, 키의 관리, 데이터 분리 등에 대해 자세하게 알 필요 없이 최소한의 상위 수준 보안 요청만 알고서도 사용할 수 있도록 하는데 있다. 이와 같은 상위 수준의 요청은 암호 구문(cryptographic context)을 초기화 하고, 데이터를 보호하는 것이다.[6]

이 요청들은 암호 알고리즘 또는 암호 모듈에

대해 상세한 지식을 요구하지 않는다. 기껏해야 응용이 요구되는 보호의 질(Quality of Protection)을 표현하는 정도이다.

보증서 인증 응용과 같은 특별한 목적의 응용들은 암호학적인 지식이 필요하며, 암호 토큰을 정확하게 조정하는 것이 요구 된다. 이와 같은 요청들은 특정한 알고리즘과 동작 모드를 선택해야 하며, 호출자의 실수가 암호 시스템의 보안을 유출 시킬 수 있기 때문에 하위 수준의 인터페이스를 사용하는 데는 반드시 세심한 주의가 필요하다.



[그림 2-1] 상용 암호 서비스 구조

2.2 상용 암호 서비스 기술 종류

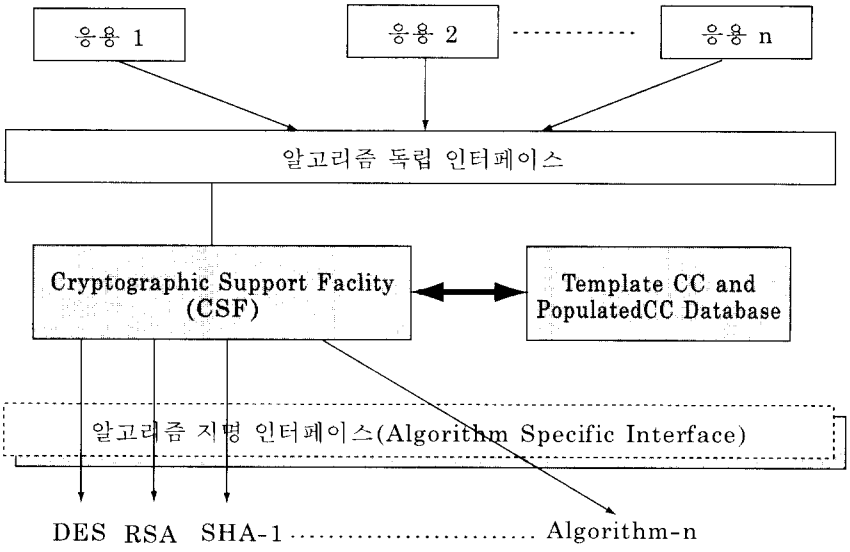
수 년 전부터 CAPI의 필요성을 인식하고, GSS-API, IDUP-GSS-API(Independent Data Unit Protection-GSS-API), GCS-API(Generic Cryptographic Service-API), CryptoAPI, CryptoKi, Forteeza 등을 이미 개발하여 현재는 표준화에 대한 노력을 경주하고 있다.[19]

2.2.1 GSS-API/IDUP-GSS-API

GSS-API는 1993년 IETF의 상용 인증 기술(common authentication technology)에 의해 RFC(Request For Comments) 1507-1511로 표준화 되었다. RFC 1508은 1997년에 RFC 2078인 GSS-API 버전 2로 개정되었으며, 이 문서들은 초기 C 언어 버전의 GSS-API를 개략적으로 설명하고 있다. GSS/IDUP가 제공하는 내용은 보안 구문(Security Context)으로 통신 쌍방 사이에 확립된 관계에 의해 통신 개체로부터 수신된 신용장을 사용하고 있다. GSS/IDUP는 특히 인증에 접근하는 방법을 제공하는 보안 서비스 API이다.[1]

2.2.2 GCS-API

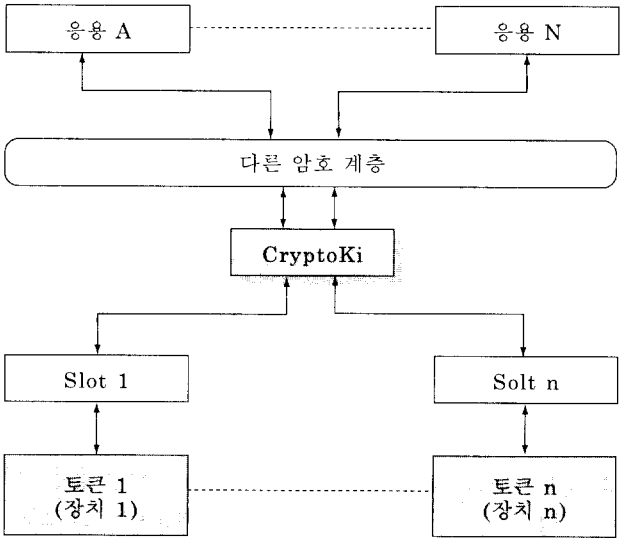
GCS-API는 X/Open 컨소시엄의 보안 활동 그룹(SWG: Security Working Group)이 응용에 대해 보안 서비스를 제공하기 위하여 만든 API와 메커니즘의 집합니다. GCS-API는 암호학적인 지식이 있는 응용과 지식이 없는 응용 모두에 대해 설계되었다. 이 응용은 최소한 알고리즘과 키 관리의 운용에 관한 기본적인 사항을 이해해야 한다.[5][19]



[그림 2-2] GCS-API 서비스 구조

2.2.3 CryptoKi

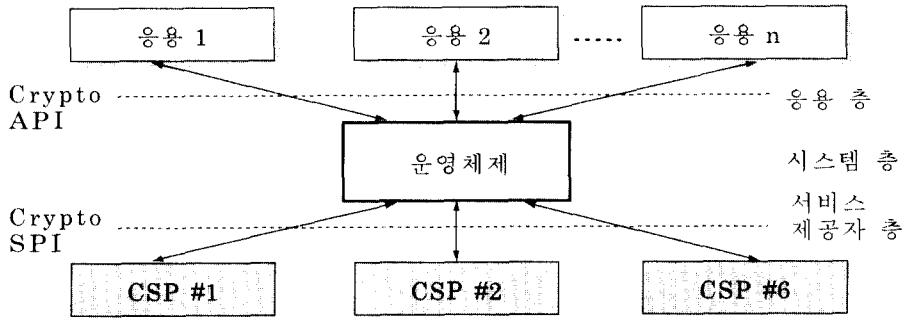
CryptoKi는 RSA사의 공개키 암호 표준인 PKCS#11이다. CryptoKi는 RSA 사에 의해서 표준화되었고, 아무런 제약없이 배포되고 있다. CryptoKi는 개인의 암호 토큰에 일차적으로 접근하는 하위 수준의 CAPI의 표준을 제공한다. RSA 사는 기존 CAPI들의 라이브러리가 응용이 암호 토큰과 함께 동작하는 것을 지원하지 못하므로 이러한 응용의 요구를 충족할 수 있는 CryptoKi를 개발하게 되었다. CryptoKi의 부가적인 목표는 휴대성, 확장성, 일반성, 자원 공유 지원 그리고 알고리즘 독립성이다.[14]



[그림 2-3] CryptoKi 서비스 구조

2.2.4 CryptoAPI(Microsoft)

마이크로소프트사가 개발한 CryptoAPI는 표준화된 프로그래밍 인터페이스가 암호 서비스 제공자(CSP:Cryptographic Service Provider)에게 통용될 수 있는 암호 모듈을 추상화하여 전달할 수 있게 하는 하위 수준의 인터페이스이다. CSP는 CryptoAPI로 하여금 객체와 데이터 구조를 실제적으로 알아야 하는 부담이 없도록 추상적인 방법으로 정의 한다. 마이크로소프트사는 CSP를 6가지 유형으로 정의하고 있는데, 각각의 형식은 운영체제에 등록되고, 하나의 CSP는 그 부류의 기본 CSP로 할당 된다. CSP형식은 하드웨어와 소프트웨어 양쪽 모두로 구현될 수 있다. 마이크로소프트사는 몇 개의 미리 정의된 CSP의 형식이 갖는 최소한의 기능을 정의함으로써 응용이 최대한의 융통성을 갖게 한다.[9]

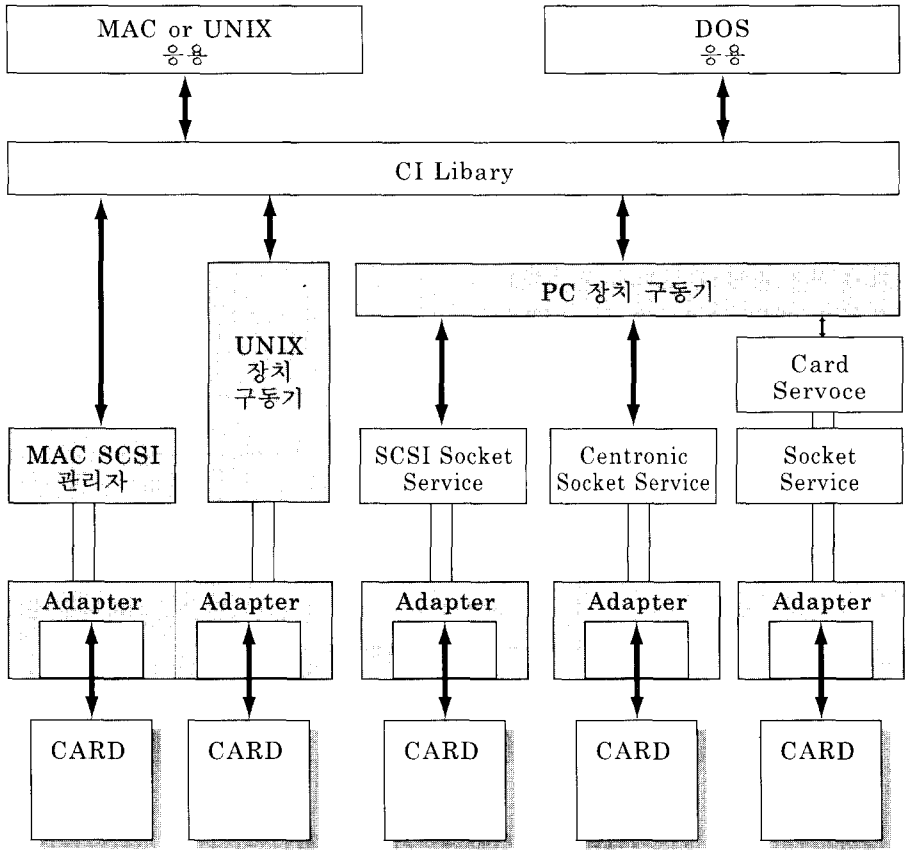


[그림 2-4] CryptoAPI 서비스 구조

2.2.5 Fortezza

NSA(National Security Agency)의 Fortezza 프로그램은 PCMCIA(Personal Computer Memory Card Implementation Association)카드에 암호 기술을 구현한 것으로 사용자는 전문 암호 지식이 없더라도 암호 서비스를 제공 받을 수 있다.

1994년 NSA는 Fortezza에 대한 규격과 표준을 제공하여 미국방부 메시지 시스템(Defence Message System, DMS)에 적용되어 전국방부의 전자 우편을 안전하게 송수신할 수 있도록 계발되었다. 카드는 메시지를 전자적으로 서명하고 암호화 하여 송신하고 수신측이 이를 검증하여 복호화 하는 전자지불, 기록의 기밀성, 기타 컴퓨터 시스템 제어 등에 활용한다.[11]



[그림 2-5] Fertezza 시스템

2.3 상용 암호 서비스 기술 평가 기준[18]

개발자들은 응용에 의해 사용되는 CAPI를 암호 모듈에 대응시키는 소프트웨어를 만들어야 한다. 이 때 반드시 표준 CAPI를 선택해야 하며, 또한 응용 분야에 가장 적합한 CAPI를 선택하는 것이 중요하다.

가. **알고리즘 독립성** : 알고리즘 독립성은 암호 서비스를 제공하기 위해 특정한 알고리즘을 사용하도록 명시하지 않는 CAPI들의 특징이다. CAPI 들은 반드시 현재와 미래의 암호 알고리즘에 대한 광범위하게 선택할 수 있어야 한다. 이 특징은 응용이 기초적인 암호 모듈에 의해 지원되는 어떠한 암호 알고리즘에도 접근할 수 있도록 하며, 호환성을 증대 시킨다.

나. **응용 독립성** : 응용 독립성은 응용을 설계하는데 있어 모든 CAPI가 동등하게 적합하다는 특징이다. CAPI들은 반드시 미래의 응용과 마찬가지로 현재 작성되고 있는 다양한 응용에 암호 서비스를 제공하여야 한다. 응용 독립성의 특징을 갖는 CAPI들은 프로그래머가 다양한 응용을 작성하도록 할 것이다. 이러한 특성은 CAPI들이 폭 넓고 오랫동안 사용되도록 할 것이다. 잘 정의된 CAPI들은 파일 전달과 같은 연결 지향적인 응용과 마찬가지로 전자

우편과 같은 저장과 재전송을 하는 모든 응용을 지원할 수 있어야 한다.

다. **암호 모듈 독립성** : 암호모듈 독립성은 암호 서비스를 제공하는데 있어서 CAPI들이 다른 어떤 것과 마찬가지로 특정한 암호 모듈을 사용할 수 있는 특징이다. 응용은 기초적인 암호 구현에 대해서 상세하게 알 필요가 없다. 예를 들면, 응용은 암호가 하드웨어로 제공되든지, 소프트웨어로 제공되든 지에 대해 알 필요가 없다. 암호 모듈 독립성은 복수 개의 암호 모듈 구현의 사용에 대한 견고한 기초를 제공한다.

라. **암호학적 지식의 정도** : 암호 지식은 응용 개발자의 암호학적인 지식의 양을 의미한다. 목표는 대부분의 응용이 개발자로부터 최소한의 암호학적인 지식을 요구하는 것이다. 예를 들면 키 관리와 같은 몇몇의 응용은 개발자로부터 고도의 지식을 요구한다. CAPI 명세서는 암호학에 대한 지식을 거의 요구하지 않는 것으로부터 상당한 수준의 숙련도를 요구하는 것에 이르기 까지 다양하다. 그래서 CAPI들은 반드시 암호학적인 지식이 있는 응용과 암호학적인 지식이 없는 응용 모두를 지원할 수 있어야 한다.

마. **모듈화 설계 및 보조 서비스** : 모듈화 설계는 CAPI를 특정한 목적이 있는 완전한 보안 서비스를 제공하기 위해 함께 사용되는 단위로 분리하는 것이다. 보조 서비스는 목표 암호 서비스에 의해 사용되는 지원 서비스이다. 이것들은 키의 사용 주기 관리(Key life-cycle management), 암호 모듈 증명, 사용자 인증, 보증서 관리, 인출 능력 그리고 세션의 설치 및 종료 능력 등을 포함한다. CAPI 구조는 반드시 기능적으로 완벽해야 한다. 단순한 구조는 오직 2개의 모듈이 있을 수 있는데, 암호 서비스와 키 관리가 그것이다.

바. **안전한 프로그래밍** : 안전한 프로그래밍은 프로그래머에 의해 초래되는 보안 사고를 방지할 수 있는 것을 말한다. 안전한 프로그래밍을 위한 것으로 일관성 있는 이름 부여 습관, 정보 은익 그리고 쉬운 사용을 들 수 있다. 일관성 있는 이름 사용 습관을 가짐으로써 프로그래머가 각 절차와 파라메터의 목적과 사용을 잘 못 이해할 수 있는 가능성은 감소할 것이다. 정보 은익은 모호한 핸들과 포인터의 사용에 의해 제공되며, 부주의에 의해 기밀성이 있는 정보의 유출을 방지한다. 쉬운 사용은 CAPI 메커니즘이 암호 작용을 매개하고 순서화 하는 상세한 임무를 수 행할 수 있게 하며, 프로그래머의 실수를 감소 시킨다. 이것은 CAPI 상세서가 암호학적인 지식이 없는 프로그래머를 목표로 하는데 있어 매우 중요하다.

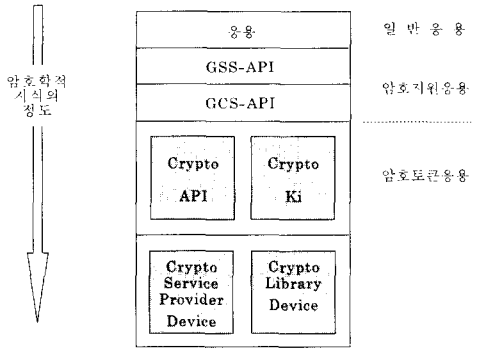
사. **보안 범주** : 보안 범주는 보안에 관계된 기밀성이 있는 정보가 신뢰할 수 있는 전산 기반(trusted computing base) 신뢰할 수 없는 응용에 누출되는 것을 방지하는 경계를 말한다. 신뢰할 수 있는 시스템에서 응용이 밖에 있는 반면에 보안 범주를 갖는 구조는 그 범주 안에 암호을 포함할 것이다. CAPI들은 보안 범주를 위반하는 입구로 사용되어서는 안된다. 그것은 기밀성이 있는 암호 데이터에 접근을 제한하여야 하며, CAPI 인터페이스를 벗어나는 그와 같은 정보를 전파해서는 안된다.

2.4 상용 암호 서비스 기술 비교

앞에서 살펴본 각CAPI는 암호 모듈 독립성을 제공하며, CAPI들간의 중요한 차이점은 응용 개발자의 요구되는 암호 지식의 양과 CAPI를 통해서 복구될 수 있는 기밀성이 있는 정보의 양의 영역이다. GSS/IDUP는 가장 안전한 인터페이스를 제공한다. 그렇지만 암호학을 다루는 가장 제한된 능력을 제공한다. GCS-API, CryptoAPI 그리

고 CryptoKi는 응용에 암호학을 다루는 보다 많은 능력을 제공하며, 응용이 인터페이스를 오용하는 위험성을 증가 시킨다. 응용의 대부분이 암호학적인 지식이 없으므로, 응용 개발자가 GSS/IDUP를 사용하는 것이 권고된다. 응용이 보안 지원 또는 암호 토큰을 다루기 위해서 직접 접근을 요구하는 경우에 GCS-API, CryptoAPI 그리고 CryptoKi는 고려되어야 한다.

일반적으로 암호 지식이 없는 응용 개발자는 GSS/IDUP를 사용하는 것이 좋으며, GSS-API, CryptoAPI 그리고 CryptoKi는 암호학적 지식이 필요한 응용을 개발할 때 사용되어야 한다. 암호학적 지식이 필요한 응용에 의해 직접 호출되지는 않지만 3개의 하위 수준의 API를 의미한다. GCS-API는 GSS/IDUP에 보안 지원 함수를 만들어야 한다. 이와 같이 CryptoKi와 CryptoAPI를 사용해서 GCS-API에 대한 정교한 암호함수를 제공하여야 한다.



[그림2-6] 상용 암호 서비스 계층 구조

3. 마이크로소프트 CryptoAPI

정보화의 진전에 따라 인터넷을 통해 수 많은 가치있는 메시지/문서/전자화폐가 교환되어 처리되고 있다. 이러한 정보의 교환은 안전하다고 말할 수 없는 인터넷상에서 이루어지고 있기에 별도의 보호 방법이 필요하다. 암호기술(cryptography)은 인터넷 상을 오가는 정보의 기밀성(privacy), 무결성(integrity), 인증(authentication)을 제공할 수 있는 보호방법이다.

마이크로소프트사에서는Windows 환경하에서 보호할 수 있는 프로그램 정합 모델로써 CryptoAPI를 개발하였다. CryptoAPI는 1998. 7 현재 버전2.0이 나왔으며, 1996. 7 버전 1.0이 발표되었다.

CryptoAPI는 개발자에게 핵심 암호 기법 및 인증 기능을 제공한다. CryptoAPI 1.0은 키 생성, 키 관리, 키 교환, 암/복호화, 해쉬, 디지털 서명, 서명 확인 등과 같은 공개 키 및 대칭 키 연산을 지원한다. CryptoAPI 2.0은 인증 기반 기능들뿐만 아니라 이런 핵심 암호 기법을 포함하며, 이러한 기법들은 공개 키 연산으로 인증을 사용하고 필요한 캡슐화를 수행하며 응용 프로그램 안에 인증을 적용하는 인코딩을 원하는 개발자에게 필요하다.

CryptoAPI는 CSP가 암호 서비스를 제공하는 서비스 제공자 모델을 사용한다. 이 모델은 개발자가 그들의 응용 프로그램을 발전하고 있는 암호 기술 및 정부 수출 정책에 쉽게 순응하게 한다. CryptoAPI 2.0은 캡슐화에 X.509v.3 인증포멧, ASN.1 인코딩, 그리고 PKCS#7 및 #10 같은 기존 표준을 지원한다. 이것은 CryptoAPI를 사용하는 응용 프로그램이 이들 표준을 따르는 다른 인증 기반 시스템과 함께 운용될 수 있게 한다.[8][9]

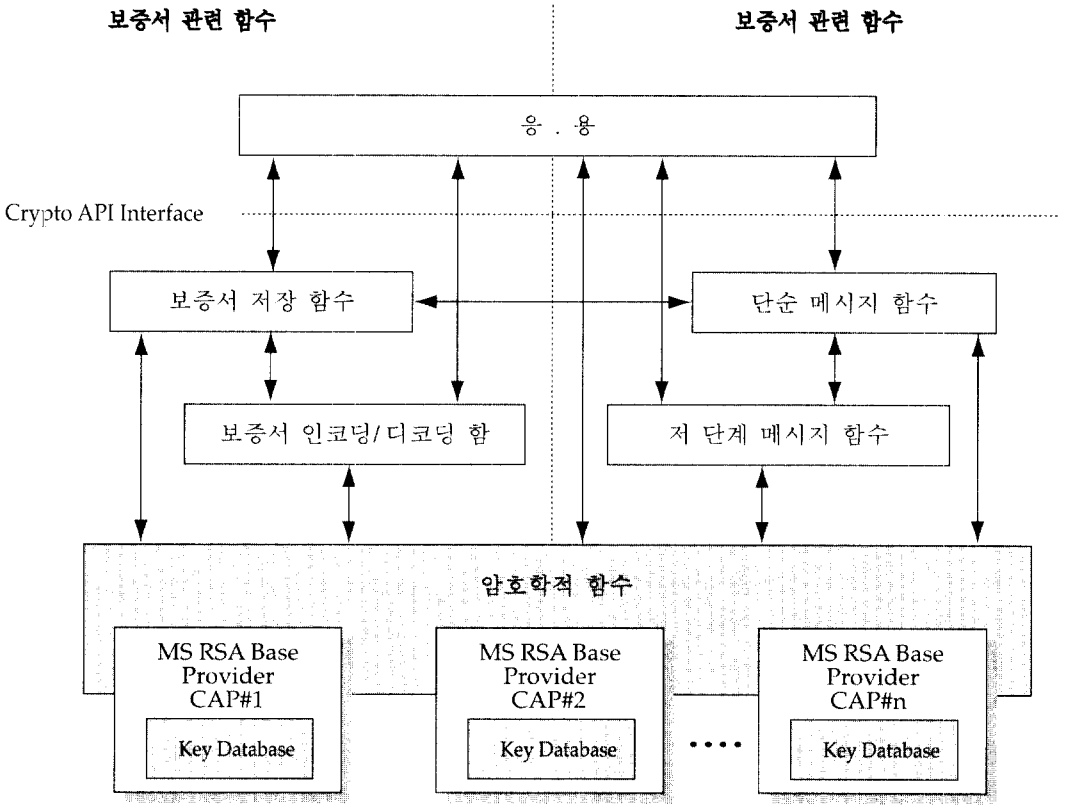
3.1 CryptoAPI 프로그래밍 모델

CryptoAPI는 단순 암호 함수(simplified cryptographic function)와 기본 암호 함수(base cryptographic function)를 통해서 사용자에게 기밀성을 제공한다. 이들 함수는 사용자의 개인키(private key)를 보호하면서 여러 방식의 디지털 서명 및 암호화를 가능하게 한다.

그리고 보증서를 통한 인증(authentication)은

보증서 암/복호화 함수(certificate encode/decode function)와 보증서 저장 함수(certificate store function)를 이용함으로써 가능하게 한다. 보증서는 개체(entity)에 대한 인증이 끝난 후 보증 기관(CA:Certification Authority)에 의해 발행된다.

CryptoAPI 프로그래밍 모델은 그래픽 프로그래밍 모델과 유사한데 암호 서비스 제공자(CSP)들은 그래픽 디바이스 드라이브에, 하드웨어는 그래픽 하드웨어에 비유된다. 그래픽 응용 프로그램이 그래픽 하드웨어를 직접 액세스 하지 못하는 것처럼 암호 응용 프로그램도 CSP나 하드웨어를 직접 액세스하지 못한다. CryptoAPI는 [그림3-1]과 같이 4개의 주요 기능적인 영역으로 구성되어 있다.



[그림 3-1] CryptoAPI 구조

응용 프로그램은 CryptoAPI가 제공한 호출 방식에 따라 서비스를 제공 받는다. 서비스 제공의 가장 기본 모듈은 CSP이다. CSP는 필요한 알고리즘과 암호 세션 상태 및 공개/개인 키를 안전하게 보관하는 장소를 제공한다. 응용 프로그램은 [그림3.1]에 있는 상위 4가지 함수를 직접 호출하나, CSP는 기본 암호학적 함수를 통해서만 통신할 수 있다.

- 보증서 인코딩/디코딩 함수 : OSI 구조를 통하여 보증서를 송.수신할 때 사용하는 함수이다. 보증서, CRL, 보증 요구서, 보증 확장서를 OSI표준에서 요구

하는 ASN.1(Abstract Syntax Notation One)형태로 인코딩/디코딩 방법을 제공한다.(총4개)

- 보증서 저장 함수 : 보증서의 저장, 추출, 활용에 사용하는 함수들로 구성된다. 보증 체인으로 연결되어 최종 루트 CA까지 거슬러 올라갈 수 있는 보증서 증명 서비스 제공이 가능하다.(총65개)
- 기본 암호 함수 : 응용 프로그램이 암호 기술을 직접 사용할 수 있는 기본 함수들로 구성된다. 6가지의 CSP 모듈이 있으며, 강력한 암호 알고리즘과 스마트 카드와 같은 하드웨어를 제공하기도 한다.(총34개)
- 단순 암호 함수 : 타스크 중심의 여러 기본 암호 함수들과 상위 레벨 연결 함수들로 구성된다.(총13개)
- CSPs : 실제 암호화가 수행되는 독립적 모듈이다. 어떤 응용 프로그램은 사용자 고유의 CSP를 요구할 수도 있다. 실제 구현되는 CSP는 적어도 DLL(Dynamic Link Library)과 서명 파일로 구성되는데 이 서명 파일은 CryptoAPI가 주기적으로 CSP의 무결성을 검사하는데 사용된다. 또한 어떤 CSP는 지역(local) RPC를 통한 주소(address) 분리 서비스 방식에 의해 구현되기도 하고, 시스템 디바이스를 통해 하드웨어로 구현되기도 한다. 키 상태 변이와 동작을 하드웨어 내에서 분리시키거나, 응용 프로그램과 이 둘을 단절시킴으로써 보호한다.
- 키 데이터베이스(Key DBs) : 각 CSP는 키 데이터베이스를 지니고 있으며, 키 데이터베이스에는 하나 이상의 키 보관함(key container)를 포함하고 있는데, 특정 사용자들에 속한 키 쌍을 보관하고 있다. 각 키 보관함은 이름을 가지고 있다. CSP는 세션 단위로 키 보관함을 저장하나, 세션 사용키는 보관하지 않는다. 사용자에 대한 기본(default) 키 보관함이 생성되는데, logon 이름으로 부여된다. 또한,각 응용 프로그램은 자신의 키 보관함(키 쌍 포함)을 생성하여 이름을 부여 할수 있다.

함수들은 위에서 설명된 것 외에도 보증 데이터를 비교, 사인, 해쉬, 검증하는 보증서 운용 함수(certification helper function) 43개, 메시지 송.수신시 사용하는 하위 레벨 메시지 함수(low level message function) 14개 그리고 객체들을 설치, 등록, 인코딩하는 OID 지원 함수(OID support function) 18개가 정의 되어 있다.

3.2 CSP(Cryptographic Service Provider)

CSP는 여러 응용 프로그램이 암호와 서명 서비스를 액세스할 수 있는 안전한 방법을 제공한다. 각 CSP들은 필요한 인증 수단을 요구하기도 하는데, PIN(Personal Identification Number)이나 스마트 카드 등이 그 한 예로 사용되기도 한다. CryptoAPI에서 사용되는 CSP는 다음의 3가지 규칙을 바탕으로 설계되었다

- 응용 프로그램은 키 매체를 직접 액세스할 수 없다. 그 이유는 모든 키는 CSP 내에서 생성되며, 응용 프로그램이나 관련된 DLL은 유일한 핸들을 이용해야만 호출할 수 있다.
- 응용 프로그램은 주어진 API를 통해서만 서비스 받을 수 있다.
- 응용 프로그램은 사용자의 인증 데이터를 처리할 수 없다. 인증은 CSP만이 수행할 수 있는데, bio-metric 입력이나 데이터와 같이 사용자 비밀 정보를 다룰 수도 있다.

3.2.1 CSP의 종류

CryptoAPI에 6가지 CSP 종류가 정의되어 있다. 각 CSP는 키 교환 알고리즘, 디지털 서명 알고리즘이 하나씩 정의되어 있으며, CSP 외부로 수출(export)되는 키 변형체(key blob)형식 및 디지털 서명 형식, 키를 추출하는데 사용되는 해쉬 함수, 사용되는 공개/개인 키 길이, 기본(default) 운용 모드 등이 정의되고 있다. [표3-1]와 [표3-2]는 CSP의 종류와 사용 알고리즘를 나타내고 있다.

[표3-1] CSP의 종류

공급자 유형	설 명
PROV_RSA_FULL	디지털 서명과 데이터 암호화 서비스 제공. RSA의 PKCS를 구현
PROV_RSA_SIG	RSA_FULL의 subset이며, hash와 signature 함수와 알고리즘만을 분리 구현
PROV_DSS	RSA_SIG와 같이 해쉬와 서명 함수 그리고 알고리즘을 구현. 단, RSA대신 DSS을 서명 알고리즘으로 사용.
PROV_ NSA	NSA의 Fortezza 카드용 서비스 제공 함수 구현.
PROV_MS_MAIL	Microsoft의 메일이나 호환성 있는 응용 프로그램을 위한 서비스 구현.
PROV_SSL	Netscape의 SSL과 동일한 서비스 구현

[표3-2] CSP 별 사용 알고리즘

공급자 유형	키교환	서 명	암호화	해싱
PROV_RSA_FULL	RSA	RSA	RC2, RC4	MD5, SHA
PROV_RSA_SIG	n/a	RSA	N/a	MD5, SHA
PROV_DSS	n/a	DSS	N/a	SHA
PROV_FORTEZZA	KEA	DSS	Skipjack	SHA
PROV_MS_EXCHANGE	RSA	RSA	CAST	MD5
PROV_SSL	RSA	RSA	Varies	Varies

3.2.2 CSP 연결

응용 프로그램이 매번 수행될 때, 맨 처음 호출되는 CryptoAPI함수는 CryptoAcuireContext 함수이다. 특정 CSP에 대한 핸들을 응용 프로그램에게 반환하며, CSP내의 키 보관함을 가리킨다. CSP형태와 이름을 지정하여 호출하면, 지정된 CSP가 메모리로 로드되어 핸들이 반환된다. 아래 함수들은 CSP와 연결/단절 기능을 제공한다.

[표3-3] CSP 연결 함수

함 수	설 명
CryptAcquireContext	각 CSP의 개별 키 보관함에 대한 핸들 획득
CryptGetProvParam	CSP의 동작을 제어했던 인수들을 회수
CryptReaseContext	CSP와 키 보관함에 대한 핸들을 해제
CryptSetProvider	현재 사용자의 기본 CSP를 설정
CryptSetProvParam	기본 CSP를 설정

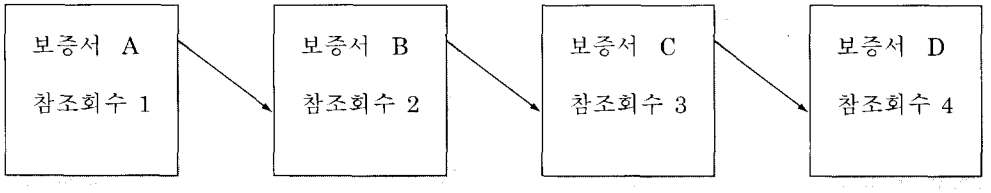
3.3 보증서 기반 인증

네트워크 상의 하나의 개체나 사람을 보증할 수 있는 문서를 디지털 보증(Digital Certification) 또는 간단히 보증이라 한다. 디지털 보증에서는 보증이 수치적으로 연계된 신뢰 구조(Hierarchy Trust)가 구축된다. 모든 사람이 신뢰할 수 있는 기관으로써 보증 기관이 선정되는데 우체국과 같은 정부 조직이나 회사가 될 것이다. 이러한 디지털 보증의 목적 중 하나는 보증서 내에 포함된 피보증인의 공개키 확인이다. CryptoAPI에서는 ITUT-T 권고 X.500(또한, ISO/ICE 9594-8) 계열의 X.509 보증서를 추출, 저장, 삭제, 리스트, 검증할 수 있는 툴을 제공한다.

3.3.1 보증서 관리

보증서 관리에는 보증서 저장과 보증서를 메시지로 사용하는 기능이 있다.

가. 보증서 저장 : 디스크와 같이 저장성이 있는 장치에 [그림 3-2]와 같이 보증서나 보증서 해지 목록(Certification Revocation List, CRL)들을 저장하는 기능을 제공한다.



[그림3-2] 보증서 저장

참조 번호는 초기에는 모든 보증서에서 1 이지만 복사될 때마다 하나씩 증가된다. 예를들어 피보증인에 대한 보증서 certB를 요구할 때 CertGetIssuerCert를 호출하여 참조 번호를 1에서 2로 증가하며, 다 사용한 후에는 CertFree CertContext를 호출함으로써 참조 번호를 1개 감소시킨다. 보증서 저장을 close하려고 할 때 참조 번호가 1 이상이라면 수정되지 않는다.

나. 보증서와 CRL의 추가와 추출 : 보증서와 CRL의 추가와 추출은 [표3-4]와 [표3-5]의 함수를 이용하여 수행한다.

[표3-4] 보증서 추가

보증서 추가 추출함수	설 명
CertAddCertificateContextToStore	보증서를 보증서 보관함에 추가
CertAddCRLContextToStore	CRL 구문을 보증서 보관함에 추가
CertAddCertificateToStore	보증서 보관함을 다른 보증서 보관함에 있는 보증서에 연결
CertAddCRLContextToStore	CRL구문을 보증서 보관함에 추가

처음 2개 Context함수는 구조체 데이터를 아래의 2개 함수는 엔코딩된 보증서를 저장하는데 사용한다.

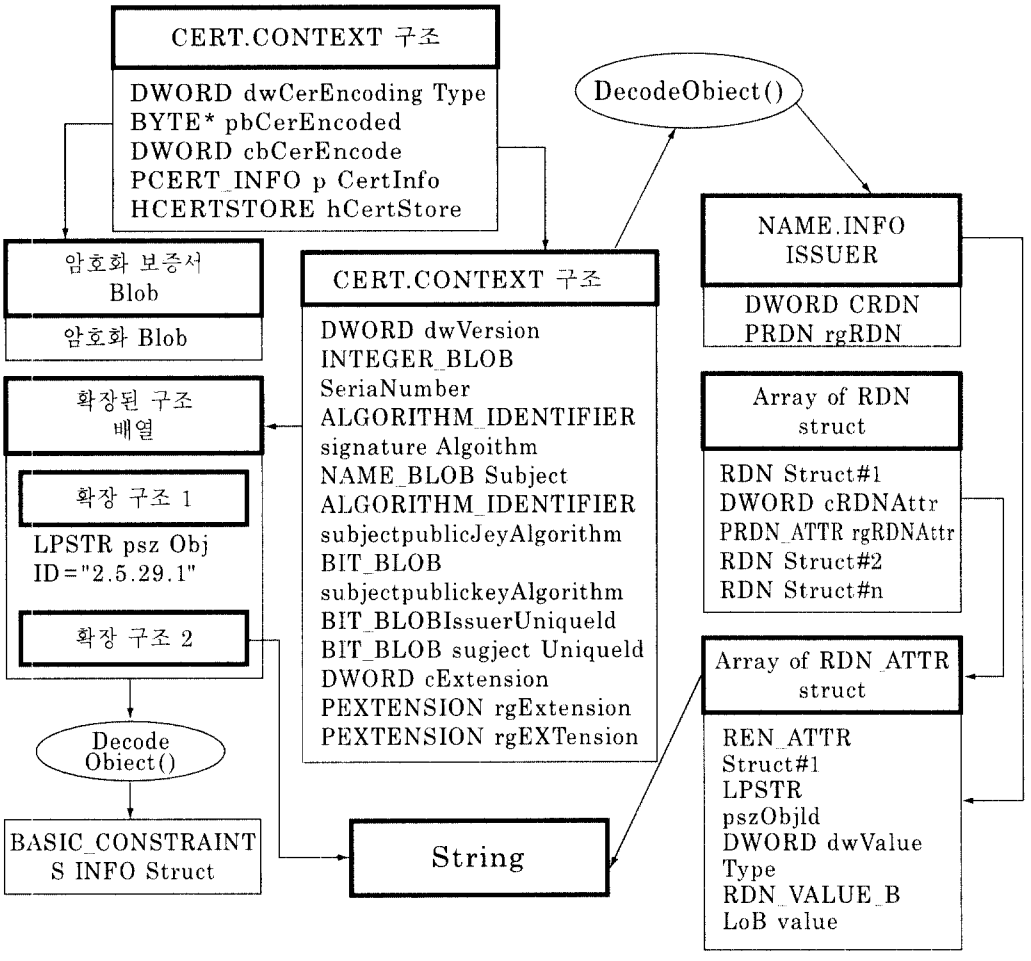
[표3-5] 보증서 추출

보증서 추출 함수	설 명
CertGetIssuerCertificateFromStore	피보증인에 대한 여러 발행자 보증서가 존재할 때 모든 보증서를 출력코자 할 때
CertGetSubjectCertificateFromStore	보증서 보관함 소유권 획득
CertGetCRLFromStore	보증서 보관함에서 CRL 구문 획득

3.3.2 보증서 인코딩/디코딩

CryptoAPI는 보증서의 인코딩/디코딩을 저장한다. 디코딩 절차는 비트들로 구성된 변형체(blob)을 입력으로 받아 미리 저장된 C 구조로 변환한다. DecodeObject가 바로 이 기능을 수행하는 함수이다.

보증서의 내용(Certificate Context, CC)은 단지 C 구조이다. CC는 인코딩 타입, 원래의 인코딩 보증서 blob 포인터, 보증서 정보(CERT_INFO) 포인터, 보증서 저장소에 대한 핸들 내용을 가지고



[그림3-3] CERT_CONTEXT 와 CERT_INFO

있다. CC의 생성은 CertCreateCertificateContext에 의해 수정된다. CERT_CONTEXT 와 CERT_INFO구조의 데이터 타입이 생성되고, 인코딩 보증서 blob가 복제되며, 동일한 포인터가 반환되어 [그림3-3]를 구성한다. [그림3-3]의CERT_INFO C구조의 IssuerUniqueID와 SubjectUniqueID는 X.509 버전2의 구현 부분이며, PEXTENTION부분은 X.509 버전3에서 채택되어 X.509 버전2의 기능을 포함한다.

3.4 암호 키(Cryptographic Keys)

암호 키는 2가지 종류가 있는데 세션 키와 공개키 쌍이 있다. 세션 키는 보통 데이터를 암/복호화하는데 사용되며, 대칭 키 시스템을 사용한다. 따라서, 동일한 키를 공유하여야만 암호 서비스를 제공 받을 수 있다. 공개키 쌍은 서명, 키 교환에 주로 사용되는 비대칭 시스템이다. 공개키 쌍에 의한 암/복호화 및 서명이 바람직하지만, 처리 속도 등에 문제가 있기에 암호화에는 대칭 키를, 서명에는 해시 알고리즘을 사용한다.

3.4.1 CryptoAPI 키 함수

모든 키는 CSP내에 저장된다. CSP는 키를 생성, 파괴, 사용을 관리한다.

가. **세션 키** : 응용 프로그램은 얼마든지 세션 키를 생성하여 세션동안 사용할 수 있으나, 세션이 끝난 후에는 CSP로부터 응용 프로그램 영역으로 키를 수출하여야 한다. 세션 키는 CrypGerKey나 CryptDeriveKey 함수에 의해 생성된다. 이 때, 사용되는 알고리즘을 저장하여야 키가 생성된다.

나. **공개키 쌍** : CryptoAPI는 사용자 당 2개의 공개키 쌍을 제공한다. 하나는 키 교환용 키 쌍(Key exchange key pair)으로 일명 교환키(Exchange Key)라고 불리며, 세션 키를 암/복호화 하는데 사용된다. 다른 하나는 디지털 서명용 키 쌍(Digital signature key pair)으로 일명 서명키(Signature key)라고 불리며, 해쉬에 사용된다. 하나의 키 쌍으로 두 가지 기능을 수행할 수도 있으나 암호학적으로 취약하므로 두 가지 쌍을 사용한다. CryptGenKey함수에 의해 두지 키가 생성되는데 AT_KEYEXCHANGE 또는 AT_SIGNATURE 파라메터에 의해 구분된다.

3.4.2 키 교환

CSP에서 생성되는 키는 응용 프로그램 영역으로 암호화되어 키 변형체(Key blob) 데이터 구조 형태로 수출된다. 키를 얼마의 시간이 지난 후에 사용하기도 하며, 원거리에 있는 사용자에게 전달하기도 한다. CryptExportKey 함수는 CSP내에 키를 응용 프로그램 영역으로 수출하는데 저장되는 키 변형체를 서명하기 위해 공개키가 저장된다. 이렇게 공개키로 서명된 키 변형체는 하드디스크에 저장되었다가 CryptImportKey 함수에 의해 CSP로 수입된다.

가. **세션 키 교환** : 다음에 일반적인 세션 교환 절차를 기술한다.

- CryptGenKey 함수에 대한 random 세션 키를 생성한다. 세션 키로 메시지를 암호화
- 세션 키를 CryptExportKey 함수에 의해 CSP로부터 응용 영역으 이동시킨다. 이 때 상대방 공개 키로써 암호화된 세션 키가 key blob에 위치한다.
- 암호화된 메시지와 key blob을 수신자에 전송
- 수신자가 수신한 key blob으로부터 CSP 내로Import하기 위해 ryptImportKey 함수를 수행한다. 수신자 공개키가 사용되었다면 자동으로 복화가 수행된다.
- 세션 키로써 메시지가 복호화 한다.

나. **키 변형체(key blobs)** : 세가지 종류의 키 변형체가 있다.

- Simple key blob : 수신자의 공개 키로 암호화된 세션 키로써 저장 또는 전송g할 때 사용된다.

[표3-6] 키 교환 함수

함 수 명	설 명
CryptDestoryKey	사용했던 핸들을 파괴
CryptExportKey	암호 키를 CSP 외부로 방출
CryptGenRandom	랜덤 수를 생성
CryptGetKeyParam	키의 동작을 조절했던 인수를 회수
CryptGetUserKey	사용자의 서명 키 쌍 회수
CryptImportKey	키 Blob에서 CSP로 암호 키 전달
CryptSetKeyParam	키 동작을 조절하기 위한 인수 설정

● Public/Private key Blob : 평문 상태의 공개 키 구조

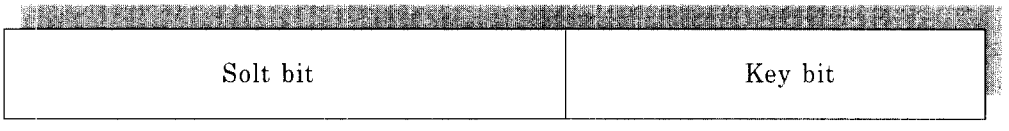
3.5 암/복호화

CryptoAPI는 메시지와 파일을 암호화 하고 복호화 할 수 있는 서비스를 제공한다. 사용알고리즘은 응용 프로그램이 선택하는 CSP에 따라 달라진다. 암호화 방식으로는 스트림과 블록 방식이 있는데, MS의 RSA Base Porvider는 RC4 스트림 암호화 방식과 RC2 블록 암호화 방식을 사용한다. .

블록이 채워지지 않는 경우(64bit) 임의의 수열을 채워서 1 블록을 만드는데 모두 0으로 채우기도 하지만 RSA의 PKCS#5, 6.2. 방식을 이용한다. CryptoAPI를 사용하는 경우 자동으로 패딩되어 송신되고, 수신시 제거된다. 암호 모드로써 ECB(Electronic Code Book), CBC(Ciper Block Changing), CFB(Ciper Feed Back), OFB(Output Feed Back)의 4가지가 있는데 CryptSetKeyParam 함수로써 선택할 수 있다.

3.5.1 Salt bit

CryptoAPI에서는 RC2 40bit 키를 사용하는데, 88bit의 salt를 사용한다. 이는 동일한 내용의 평문이라도 암호문은 많은 변화가 되도록 하기 위한 방법으로 사용된다. CryptGenRandom 함수에 의해 얻어진다.



[그림3-4] Solt bit 구조

3.5.2 수행 속도

[표3-7] CryptoAPI를 bundle 한 120 MHz Pentium에서 수행한 결과이다.

RC2/RC4는 RSA Data security에서 개발되었으나 공개되지 않는다. 원래는 키 크기가 변경되었는데 CryptoAPI RSA Base Provider에서는 40bit, 64bit 블록으로 조정하였다. RSA 공개키 암호 알고리즘은 1970 연대 개발되었고 키 크기가 변경되는데 CryptoAPI RSA Base Provider 에서는 512bit으로 고정하였다.

3.6 해쉬 함수와 디지탈 서명

RSA Base Provider는 PKCS#6를 따른 디지털 서명을 생성한다. 메시지로부터 디지털 서명을 생성하기 위해서는 해쉬 값(메시지 다이제스트)을 만들고, 이를 사인자의 개인키로 사인해야 한다. 해쉬 값은 메시지의 크기에 관계없이 128 또는 160 비트이다. 또한 단 한

3.7 수행속도측정 결과

암호알고리즘	형 태	키 설정(μsec)	암호속도(byte/sec)
DES	64bit	460	1,138,519
RC2	64bit	40	286,888
RC4	stream	151	2,377,723

비트라도 다른 두 개 메시지의 해쉬 값들은 서로 철저히 달라, 해쉬 알고리즘을 깨지 않는 한 해독이 불가능하다.

CryptoAPI에서는 CryptCreatHash 함수에 의해 해쉬 객체가 생성되며, CryptHashData에 의해 데이터가 해쉬 객체에 더하여 진다.

마지막 데이터 블록이 더해진 후 Crypt SignHash 함수에 의해 서명된다. 서명된 후의 해쉬 객체는 CryptDestoryHash함수에 의해 파괴된다. 해쉬는 서명용 개인 키나 키 교환용 개인 키에 의해서 서명되는데, 키 교환 프로토콜시와 같은 경우에는 당연히 후자의 키가 사용된다. 서명 검증시 CryptVerifySignature가 사용되며, 해쉬 객체에 대한 파괴도 이루어져야 한다.

알고리즘으로는 MD2/4/5 해쉬 알고리즘이 순차적으로 개발되었으며 MD5가 권고된다. SHA(Secure Hash Algorithm)은 미국 NIST가 개발하였으며, DSA(Digital Signature Algorithm)이나 DSS(Digital Signature Standard)와 같이 사용할 목적으로 개발되었다.

메시지 인증 코드(Message Authentication Code)는 해쉬 값과 비슷하지만 세션 키에 의해 계산된다. RSA Base Provider에서는 MAC 블록 암호방식을 사용한다. 맨 마지막에 MAC 데이터를 해쉬 값으로 사용한다. 암호방식은 세션 키가 생성될 때 저장된다. 해쉬를 위해서 이용되는 함수는 [표3-8]와 같다.

4. 결 론

본 논문은 정보보호 서비스를 제공하는 암호 서비스 미들웨어로써 국내 보안 API의 개발에 최종 목표를 두고 기존 CAPI를 분석하였다.

CAPI를 분석한 결과 각각의 API는 계층적 중심적용 분야별로 함수의 개발 특징을 보이고 있으나, 암호서비스 기본 기능의 구현은 당연히 공통부분으로 도출된다. CAPI들은 각각의 특징을 간직하며 상호정합을 피하기 위한 개방형 구조로 확장 보완되고 있는 과정에 있다.

CryptoAPI는 자사 독점 운영체제인 Windows' 95와 NT의 하부 플래폼으로써 96년과 97년도에 버전 1.0 및 2.0을 각각 출시하여 인터넷 익스플로러 버전 6.0에 사용중에 있다. 동 API는 RSA사의 PKCS #11을 기본기능으로 채택하여 전체함수(RSA-FULL)와 서명함수(RSA-SIG) 외 기타 CSP들로 구분하여, 암호 알고리즘 스풀과 기본 암호 서비스, 그리고 메시지 응용 보증 서비스 함수의 계층구조로써 구현하였다. 마이크로소

[표3-8] 해쉬 함수의 기능

함 수 명	설 명
CryptCreateHash	해쉬 생성
CryptDestoryHash	사용했던 해쉬 개체를 파괴
CryptGethashParam	해쉬 개체의 동작제어 인수값 회수
CryptHashData	해쉬값 계산
CryptHashSessionKey	키 개체의 암호 해쉬 계산
CryptSetHashParam	해쉬 인수값 설정
CryptSignHash	CSP 동작 설정
CryptVerifySignature	해쉬 개체에 대한 서명 검토

프트사의 기본 전략은 기본 CSP와 핵심 함수만을 정의하고 나머지 접속 부분은 제삼의 ISP (Information Service Provider) 사업자로 하여금 정합하여 개발토록 하고 있다.

GCS-API는 여타 CAPI 중 상용 API로써 최초로 암호 서비스 API를 구현하였다. 이후의 API들은 FORTEZZA의 국방용과 GCS-API의 일반상용을 참조하여 자사의 CAPI를 개발하여 왔다. GCS-API는 암호 알고리즘 의존 함수와 독립 함수의 크게 두 종류의 사용자를 대상으로 계층구조를 이루고 있다. GCS-API 내부 미들웨어는 계층구조로써 암호 알고리즘 스풀(응용프로그래머API, 시스템프로그래머 API)의 하위 계층 함수와 키 관리 기능 그리고 설치 및 구성 서비스 API의 상위 계층 함수로 구성되어 있다.

FORTEZZA CI-LIB는 NIST의 표준 암호 알고리즘들을 채택하여 카드 인터페이스 함수를 부가하였다. 그러나, 공개되지 않는 국방성 전용제품 개발을 주목적으로 디바이스 드라이버 커널 부분의 함수와 키 관리 함수를 각각 하위계층과 관리계층으로 구분하였다. 하드웨어 PCMCIA정합이 포함되는 관계로 하부함수를 극히 세분화하고 DMS적용을 위한 자체 보증구조를 만들어 내부에 X.509 보증 구조를 포함하는 다단계 계층구조를 지니고 있다.

국내에서도 표준 알고리즘 제정과 정보보호 서비스의 필요성이 확대되고 있는 실정에 비추어 한국형 상용 암호 서비스 API의 구현 개발이 절실히 요구된다. 따라서 향후 암호기술 선진 외국의 CAPI 상용화 개발과 국내의 CAPI 요구에 부응하는 연구가 있어야 할 것이다.

참고문헌

[1] Adams, C., "Independent Data Unit Protection Generic security Service Application Program Interface(IDUP-GSS-API)", Internet Draft, February, 1996.

[2] Amy Reiss, "Cryptographic Application Program Interface", Proceedings of 18 NISSC, Baltmore USA, October, 1995.

[3] CCITT Recommentation X509, The Directory : Authentication Framework, 1993.

[4] Emmett Paige, Jr., "Selrction of Migaration System," Assistant Secretary of Defense Memorandum, November, 1993

[5] Kaliski, B., "X/Open Preliminary Specification: Generic Cryptographic Service API Draft 8", April, 1996.

[6] Linn, J., "Generic Security Service Application Program Interface", RFC 1508, November, 1993

[7] Linn, J., "Generic Security Service Application Program Interface, Version 2, Internet draft 5, February, 1996

[8] Microsoft Corporation, "Application Programmer's Guide:Microsoft Crypto API Version 1.0", July, 1996.

[9] Microsoft Corporation, "Application Programmer's Guide:Microsoft CryptoAPI Version 2.0", September, 1996.

[10] NBS, FIPS PUB 81, "Data Encryption Standard", NIST, January, 1988.

[11] NSA, "FORTESSA Application Implementers Guide for The PCMCIA based FORTEZZA Cryptologic Card, V1.0, January, 1995.

[12] NSA, "Security Service API:Cryptographic API Recommendation, NSA cross Organization CAPI Team, June 1995.

[13] RSA Lab, "PKCS#11:Cryptographic Token Interface Standard, April, 1994.

[14] http://premium.microsoft.com/msdn/library/

[15] http://www.dstc.edu.au/BDU/APAP/GSSAPI/GSSAPI.html

[16] http://www.opengroup.org/public/teach/secutity/gcs/ms__comp.htm

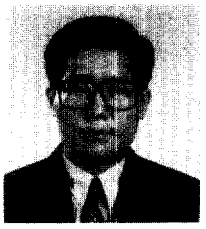
[17] http://www.itu.int/itudoc/itu-t/rec/x.html

[18] http://www.kisa.or.kr/

[19] 김석우, "개방형 보안구조", 한국정보과학회 춘계학술발표회 pp33-42, 1997. 3.

[20] 한국정보보호센터, "'97 정보보호기술 표준화 현황", pp187-207, 1997. 12.

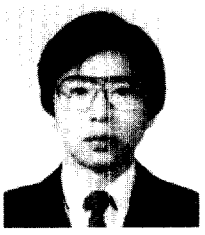
□ 著者紹介



김 점 구

1990 년 광운대학교 이과대학 전자계산학과(이학사)
1994년 광운대학교 전산대학원 전자계산학과 (이학석사)
1997년 - 현재 한남대학교 컴퓨터공학과 박사과정
1990년 - 1994년 7월 ㈜ 제성프로젝트
1994 년 8월 - 현재 시사컴퓨토피아

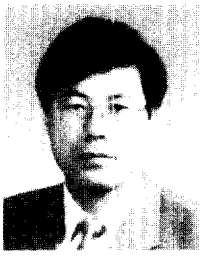
※ 주관심분야 : 컴퓨터 네트워크 정보보호



김 석 우

1979년 한국항공대학교 통신정보공학과 졸업
1989년 뉴저지 공과대학 컴퓨터공학 (석사)
1995년 아주대학교 대학원 컴퓨터공학과 (박사)
1979년 ~ 1980년 삼성전자 특수사업부
1987년 ~ 1989년 AT&T Bell Lab 연구원
1990년 3월 ~ 1997년 3월 한국전자통신연구소 책임연구원
1997년 3월 ~ 현재 한세대학교 정보통신학과 교수

※ 주관심분야 : 정보통신, 정보보호



강 창 구

1975년 한국항공대학교 항공전자공학과 졸업 (공학사)
1986년 충남대학교 대학원 전자공학과 (공학석사)
1993년 충남대학교 대학원 전자공학과 (공학박사)
1979년 ~ 1982년 한국공군 기술장교
1987년 ~ 현재 한국전자통신연구원 부호기술 연구부 실장 책임연구원

□ 著者紹介

이 재 광

1984 년 광운대학교 전자계산과 (학사)
1986 년 광운대학교 대학원 전자계산과 (석사)
1993 년 광운대학교 대학원 전자계산과 (박사)
1986 년 3월 ~ 1993년 8월 군산전문대학 전자계산학과 부교수
1997년 8월 ~ 1998년 7월 미국 University of Alabama 객원교수
1993 년 8월 ~ 현재 한남대학교 컴퓨터공학과 부교수

※ 주관심분야 : 컴퓨터 네트워크, 정보통신, 정보보호