主題

암호화 알고리즘 소개 : SET을 기반으로

한국항공대학교 염동복, 김환국, 박재철, 이병윤, 박종서

차 례

1. 서 론

21세기를 가리켜 정보화 시대라고 해도 과언이 아닐 것이다. 그만큼 정보는 국가나 기업, 개인에게 있어서 가장 중요하게 다루어지고 있다. 그러나, 정보의 절취, 수정, 훼손이나 통신망 등의 불법적인 침입, 바이러스 등으로 인하여 이의 보호가 절실하게 거론되고 있다. 이러한 역기능적인 부분을 암호학적인 방법으로 해결하는 방법 면에 많은 연구가 있어 왔고, 현재도 진행 중에 있다. 정보보호 기술이 민간 부문에서 본격적인 관심을 끌기 시작한 것은 인터넷 전자상거래가 도입, 활성화되면서부터 라고 볼 수 있다. 인터넷상에서의 개인신용정보 등의 유출을 막아야 할 필요성이 커졌기 때문이다.

정보보호를 위한 기술은 크게 암호기술, 전자상거래 정보보호기술, 시스템 보호기술, 네트워크 보호기술 등으로 나눌 수 있다. 여기에서는 암호기술과 전자상거래 정보보호기술에 관한 사항만을 다룰 것이다. 정보 보호를 위한 시스템은 다음과 같은 기본적인 요구사항을 만족시켜야 한다.

(1) 정보누출방지를 위한 기밀성이 보장되어야 한다. (비밀성:Secrecy)
(2) 정보변조 및 파괴를 예방하고 감지할 수 있어야 한다. (무결성:Integrity)
(3) 상대방이 정당한 사용자임을 인식할 수 있어야 한다. (인증:Authentication)
(4) 상대방이 시스템 사용에 대한 부인을 막을 수 있어야 한다. (부인방지:Nonrepudation)
(5) 정당한 사용자의 정보 사용권을 줄 수 있어야 한다. (가용성:Availability)

본 고에서는 불특정 다수를 상대로 위의 요구사항을 만족시킬 수 있도록 고안된 SET(Secure Electronic Transaction)이라 하는 정보보호기

술 내에서 사용되어지는 알고리즘이나 프로토콜들을 이용하여 실제 정보보호 기술들이 어떻게 적용이 되어지는지를 살펴볼 것이다.

2. SET이 생겨난 배경

SET(Secure Electronic Transaction)은 인터넷과 같은 open network에서 안전하게 상거래를 할 수 있도록 보장해주는 지불 프로토콜이다. 다시 말하면 SET은 우리가 일상생활에서 이용하는 신용카드 거래체계를 인터넷을 통한 전자상거래에서도 유사하게 이용할 수 있도록 하고 있다. 1996년 2월 Visa International사와 Master Card International사는 인터넷을 이용한 안전한 신용카드 거래의 기술적 표준을 만들기 위하여 SET 프로토콜의 개발에 합의했으며, SET 프로토콜의 연구 및 개발에는 IBM, 마이크로소프트, 넷스케이프 커뮤니케이션즈(Netscape Communications), 베리 사인(VeriSign), GTE 등 7개 컴퓨터 및 보안 관련 업체가 참여했다. [13]

SET 프로토콜은 메시지의 암호화와 개인을 인증(Authentication) 하는 전자 증명서 등을 통해서 인터넷상에서 안전한 전자상거래가 이루어질 수 있도록 하고 있다. 즉, 메시지 암호화를 통하여 전자상거래에 참여하는 카드소지자의 계좌번호 및 신용카드 번호와 지불 정보 등 민감한 정보의 노출을 방지하며, 전자서명 및 해시(hash) 함수를 이용하여 모든 메시지 내용의 무결성(integrity)을 보장하는 한편, X.509를 기반으로 한 인증서 방식을 이용하여 거래 행위의 실질적인 주체인 카드 소지자와 상인간에 인증을 제공한다. 예를 들면 신용카드를 이용할 경우에는 물건을 사는 사람이 정말로 신용카드 회원인지를 증명할 필요가 있다. SET에서는 이러한 목적으로 인증서를 사용한다. 인증서는 실제로 컴퓨터 상에서 취급하는 데이터이며, 여기에는 본인의 이름, 신용카드의 이름 외에 통신에 필요한 암호키의 정보도 일부 포함된다.

SET을 이용한 상거래 트랜잭션에 참여하는 구성원은 카드소지자(Cardholder), 상인(Merchant), 지불 게이트웨이(Payment Gateway), 그리고 인증기관(CA; Certificate Authority)으로 정의되어 있으며, 신용카드사 또는 제 3자에 의해서 운영되는 지불 게이트웨이는 금융기관 네트워크를 통하여 은행과 연결된다. SET 프로토콜 명세는 다양한 하드웨어 및 소프트웨어 플랫폼간에 동작할 수 있도록 하기 위하여 ASN.1을 이용하여 기술되어 있다.

2.1 SET의 메시지 전달 흐름도

그림 1을 전송자 측면과 수신자 측면으로 나누어 좀더 자세히 설명하면 다음과 같다.

- 전송자 측면

(1) 전송하고자 하는 메시지를 해시 함수를 통해 특정 길이의 메시지 축약(Message Digest)을 생성한다. (해시 알고리즘 사용)

(2) 1에서 생성된 메시지 축약을 전송자의 비밀키로 암호화를 하여 자신의 전자서명을 생성한다. (전자서명 프로토콜 사용)

(3) 메시지와 전자서명, 자신의 인증서를 하나로 합하여 세션 키를 이용하여 암호화를 한다. (대칭키 암호화 알고리즘 사용)

(4) 3에서 사용한 세션 키는 상대방이 아직 모르고 있는 상태이므로, 이를 상대방의 공개키를 이용하여 암호화하여 상대방에게 전송한다. (공개키 암호화 알고리즘 사용)

- 수신자 측면

(5) 전송자가 보내온 세션 키를 자신의 비밀키를 이용하여 복호화 한다. (공개키 암호화 알고

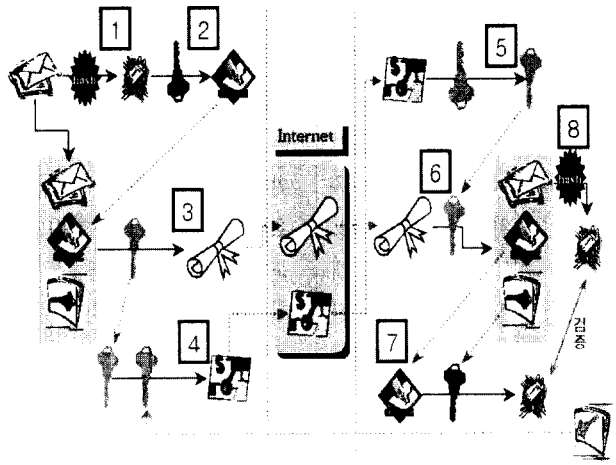


그림 1. SET의 메시지 전달 흐름도

리즘 사용)

(6) 5에서 얻은 세션 키를 이용하여 암호화된 메시지를 복호화 한다. (대칭키 암호화 알고리즘 사용)

(7) 6을 통해 얻은 내용 중 전자 서명 부분을 상대방의 공개키를 이용하여 전송자가 생성한 메시지 축약을 얻어낸다. (전자 서명 프로토콜 사용)

(8) 6을 통해 얻은 내용 중 메시지 부분을 전송자가 사용한 동일한 해시 알고리즘을 이용하여 메시지 축약을 얻어낸다. (해시 알고리즘 사용)

(9) 7과 8에서 얻은 메시지 축약들을 비교해서 서로 같으면 전송도중 메시지가 변조되어지지 않았음을 보장한다. (무결성 보장)

그림 1.은 SET 의 메시지 전달 흐름도를 보이고 있다.

전송자 측면에서 볼 때, 문서를 수신자에게 보내고자 할 때, 인터넷을 통해 전송하는 도중 문서의 변조, 훼손을 막기 위해 메시지 축약(Message Digest)이라는 것을 만들게 된다. 이는 원 메시지의 크기와는 상관없이 항상 특정 길이를 갖게 되며, 패리티(Parity)나 체크섬(Checksum)과 같은 역할을 하게 되는 것이다. 여기에 해시 함수가 사용되어지며, 그 예로 MD5, SHA등이 있다(1). 이렇게 생성된 메시지 축약에 자신의 전자서명 용 비밀키로 암호화를 하여 전자서명(Digital signature)을 생성한다. 전자 서명은 암호화에 사용하는 공개키 알고리즘과 구별되는 것이 자신의 비밀키로 암호화를 하고, 이를 복호화하기 위해 자신의 공개키를 사용한다는 것이다. 그렇게 함으로써, 누가 그 문서를 전송했는가를 확인할 수 있는 것이다(2). 과정 1,2에 의해 생성된 전자서명과 원문서, 그리고 자신의 인증서-인증기관에 의해 발급된-를 하나로 묶어 상대방에게 전송하게 된다. 하지만, 이렇게 보내지는 데이터는 네트워크 분석기 등을 통해 쉽게 그 내용 확인이 가능하므로, 암호화 기술을 사용해 암호문으로 변경을 하게 된다. 이를 위해 대칭키 암호화 알고리즘이 사용되어진다(3). 대칭키 암호화 알고리즘은 공개키 암호화 알고리즘보다 암호화하는데 약 100-1000배정도 빠르기 때문에 메시지 전송을 위한 부분에 주로 이용되어진다. 하지만, 쌍방이 동일한 키를 알아야 한다는 단점이 있다. 정해진 몇몇끼

리의 통신이 아닌 불특정 다수를 상대로 통신을 하고자 하는 경우에는 동일할 키를 보관한다는 것은 무의미한 것이 된다. 이 때문에 전송자는 특정한 연산을 거쳐 매 번 통신이 이루어 질 때마다 세션키(Session key)를 생성한다. 이를 과정 3에서 사용하고, 수신자에게 이를 알려주기 위해 수신자의 공개키로 이 키를 암호화를 하게 된다. 이 과정에 공개키 암호화 알고리즘이 사용되어진다. 이렇게 생성된 세션키의 암호화된 형태를 디지털 인벨롭(Digital envelope)이라 부른다(4).인터넷을 통해 수신자에게 문서가 도달되면, 수신자는 먼저 과정 3을 통해 생성된 암호문을 복호화하기 위해 과정 4의 세션키를 얻어내어야 한다. 이는 자신의 공개키로 암호화되어졌기 때문에 자신의 비밀키로 해독이 가능한 것이다. 만일 인터넷 전송 도중 불법적인 목적으로 이 문서를 해독하려 한다 해도 암호화에 사용된 키를 얻어 낼 수 없기 때문에 문서 복호화가 불가능한 것이다(5). 과정 5에서 얻은 세션키를 사용하여 대칭키 암호화 알고리즘을 사용하여 전송된 암호문을 통해 원문서와 전자서명, 인증서를 생성할 수 있게 된다(6). 과정 6에서 얻은 문서들이 실제 전송자가 보낸 문서임을 보장할 수 있을까? 만일 문서가 변조되어지지 않았다 할지라도, 훼손되어졌을 가능성이 있는 것이다. 과정 1은 이 문서의 변조나 훼손을 막기 위해 사용한다고 했었다. 전송자가 보낸 전자서명을 복호화하게 되면 전송자가 생성한 메시지 축약을 얻게 된다(7). 또한 원 문서를 동일한 해시 함수를 사용하게 되면 과정 1에서 생성된 메시지 축약과 동일한 값을 얻을 수 있을 것이다(8). 이 두개의 메시지 축약을 비교해서 서로 같으면, 문서가 안전하게 도달되어졌음을 보장하는 것이다. 과정 8이 필요한 이유는 해시 함수는 단방향성을 갖기 때문이다. 즉, 문서를 통해 축약을 만들 수는 있지만, 축약을 통해 문서를 유추할 수는 없다는 것이다.

2.2 SET을 이용하여 설명하는 이유

암호화 기술의 세부적인 사항을 먼저 언급하기 전에 암호화 기술의 총아라 칭할 만한 전자 상거래를 위해 고안된 SET의 데이터 전송에 관련된 부분의 흐름을 먼저 살펴보았다. (1)-(9)의 과정 중에 대부분의 정보보호 기술들이 사용되어지고 있음을 알 수 있을 것이다. 즉, 하나의 알고리즘이나 기법만을 사용하여 메시지 전달이 이루어지는 것이 아니라 총체적으로 연동 하여 하나의 시스템을 구성한다는 것이다.

현재 SET의 전체적인 크기의 비대로 인해 실제 인터넷상의 사용여부가 불분명한 상태이긴 하지만, 위의 그림에서 보인 것처럼 정보보호 기술 대부분을 포함하고 있기 때문에 암호화 기술의 전체적인 흐름을 이해하는 데는 가장 적합한 예가 되리라 생각되어진다. 이제 각 부분들의 필요성, 구현 방식 등 세부적인 부분을 살펴보도록 하겠다.

3. 대칭키 암호화 시스템

- 평문(Plain Text) : 보통문장
- 암호문(Cipher Text) : 암호화된 문장
- 암호화(Encipherment) 알고리즘 : E
- 복호화(Decipherment) 알고리즘 : D
- 키(key) : 암호화 매개변수

암호화 알고리즘은 암복호화에 사용되는 키의 특성에 따라 암복호화 키가 같은 대칭키 암호화 알고리즘과 암복호화 키가 서로 다른 비대칭키 암호화 알고리즘으로 크게 구분할 수 있으며, 대칭키 암호 알고리즘은 데이터 처리 형식에 따라 스트림 암호화 알고리즘과 블록 암호화 알고리즘으로 나눌 수 있다. 그림 2는 평문 P를 암호문 C로 만들고, 이를 상

대방에게 전송하여, 다시 평문 P를 얻는 과정을 보이고 있다. 이때, K_e와 K_d가 동일한 것이거나 간단한 연산으로 구해질 수 있는 경우 이를 대칭키 암호화 알고리즘이라 부르며, 서로 다른 경우 이를 비대칭키 암호화 알고리즘이라고 부른다. 암호화를 하는 과정에서 평문 P를 입력 스트림에 따라 한 비트씩 암호화를 수행하는 경우 이를 스트림 암호화 알고리즘이라 부르며, 특정 비트 단위로 암호화를 수행을 하는 경우를 블록 암호화 알고리즘이라고 부른다.

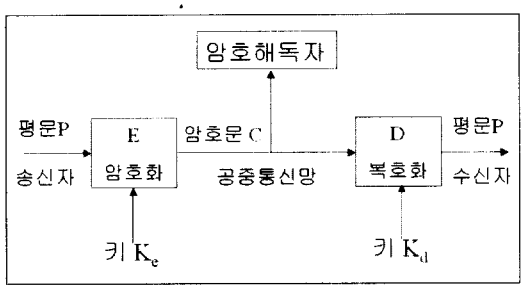


그림 2. 문서의 암 · 복호화 과정

3.1 대칭키 블록 암호화 알고리즘

대칭키 블록 암호 알고리즘은 비밀성을 제공하는데 주로 사용되어진다. N비트 블록 암호 알고리즘이란 고정된 N비트 평문을 같은 길이의 N비트 암호문으로 변경하는 것을 일컫는다. 이때 암복호화 키가 작용한다. 블록 암호화 알고리즘은 대부분 Feistel 암호 구조로 설계되어져 있다. 그림 x는 전체 구조를 보이고 있다. Feistel 암호 구조란 N비트의 블록을 N/2씩 둘로 나누고, R번의 라운드만큼 반복된 연산을 수행하는 구조를 말한다. 이때 전체 알고리즘의 요구되는 비도와 수행 효율성의 상호 절충적 관계에 의해 결정되게 된다. 보통 Feistel 암호구조는 3라운드 이상이며, 짝수 라운드로 구성되어져 있다. 대표적인 대칭키 블록 암호화 알고리즘에는 DES, SEED등이 있다. [11]

DES는 64bit(키는 56bit) 블록 암호화 알고리즘으로 키 길이가 상대적으로 짧아 DES의 안전도 문제가 대두되어 Triple DES나 128bit 암호화 알고리즘들이 제안되고 있는 실정이다. SEED는 국내 KISA, ETRI 암호 전문가들이 만든 128bit 국내형 대칭키 블록 암호화 알고리즘이다.

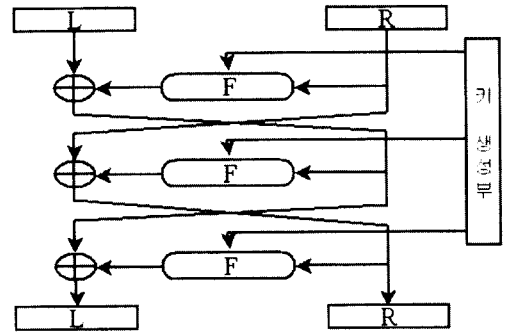


그림 3. Feistel 암호 구조

3.2 대칭키 스트림 암호화 알고리즘

1970년대 유럽을 중심으로 발달되기 시작한 스트림 암호화 알고리즘이다. 이는 키를 키 스트림 생성자를 통해 새로운 비트 스트림을 만들어 내고 이와 평문을 XOR연산하여 전송하게 된다. 이때 키 스트림이 어떠한 주기성을 갖고 반복되어질 때 주기적 스트림 암호화 알고리즘이라고 하며, 반복성을 띄지 않는 방식을 비주기적 스트림 암호화 시스템이라 부른다. 또한 스트림 암호화 시스템은 평문과 키 스트림과의 관계에 따라 동기식 스트림 암호시스템과 자기-동기식 스트림 암호시스템으로 분류할 수 있다.

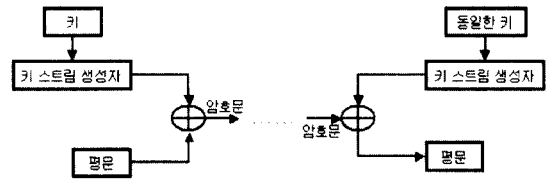


그림 4. 스트림 암호화 시스템의 구조

4. 공개키(Public-Key) 암호화 시스템

기존의 대칭키(Symmetric-Key, Private-Key)에 반하여 암호학의 혁신적인 분수령이라고 할 수 있는 공개키(public) 암호 시스템의 등장은 대칭키가 내재하고 있는 키관리의 문제와 디지털 서명(Digital Signature)상에서의 사용자 인증 문제 등의 해결책을 제시하였다. 이 장에서는 공개키 암호시스템의 개요와 그 필요성, 그리고 대표적인 공개키 암호 시스템인 RSA 암호시스템과 요즘 최대의 관심으로 대두되는 타원곡선(Elliptic Curve Cryptography)암호시스템의 내용과 구현에 대하여 알아보고 성능과 현황에 대하여 고찰한다.

4.1 공개키 암호화 시스템의 개요와 필요성

1976년 Diffie와 Hellman에 의해서 IEEE IT-22에 발표된 논문 "New directions in Cryptography"에서 기존의 암호학 상식을 뛰어넘는 혁신적인 발상으로써 공개키 암호 방식이라는 새로운 암호 방식을 제안하였다. 공개키(Public-Key)암호화 시스템에서는 기존의 비밀키(Private-Key)암호화 시스템에서 하나의 비밀키만을 사용해서 암호화와 복호화를 하는 것과는 달리 암호화와 복호화를 수행할 때 각각 서로 다른 두 개의 키를 사용한다. 암호화 키(공개키 "e")는 개방된 네트워크 상에서의 공개목록(directory)에 등록하고 복호화 키(비밀키 "d")는 개인 각자가 비밀리에 보관한다. 비밀통신 가입자 'A'에게 통신을 하려는 가입자는 가입자 'A'가 공개한 공개키로 전달하려는 평문 'm'을 암호화(Encryption: "Ee(m)")하여 암호문 'c'를 'A'에게 전송하면 'A'는 자신이 보관하고 있던 비밀키로 복호화(Decryption: "Dd(c)")하여 평문을 추출한다. 여기서 중요한 것은 공개키와 비밀키는 당연히 계산상으로 서로 추출이 불가능한 관계이어야 한다. 그림 5는 공개키 시스템의 전반적인 프로토콜의 형태를 설명한다. [2]

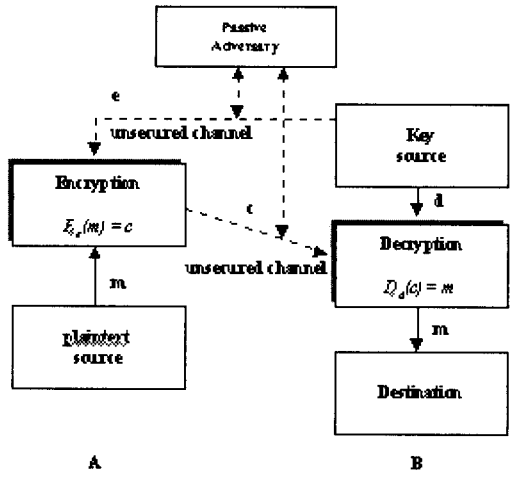


그림 5. 공개키 시스템에서의 암호화와 복호화

그림 5에 보인 바와 같이 공개키 암호 방식은 비밀키 암호 방식에서 필요한 키의 사전 분배가 필요 없는 획기적인 방식이다. 과거의 특정한 분야에서만 사용되어지던 암호 기술이 정보통신기술과 인터넷의 발달로 인하여 현재는 민간분야에서도 수요가 급격히 증가하는 추세이다. 따라서 비밀통신을 하고자 하는 가입자간의 키의 안전한 분배는 중요한 문제가 아닐 수 없다. 이러한 키분배의 문제는 공개키 암호 방식에서 공개키는 공중통신망의 공개목록에 등록하고 자신의 비밀키만 관리하게 함으로써 간단히 해결되었다. 또한, 비밀키 암호 방식의 경우 공중 통신망상에서의 비밀 통신에 필요한 키 수가 가입자 수에 따라 급격하게 증가하는 문제점을 갖고 있다. 그림 6 에서와 같이 공중 통신망의 가입자가 n명이라면 비밀키 암호 방식의 암호 키의 숫자는 ${}_nC_2 = n(n-1)/2$ 개수가 된다. 이와 달리, 공개키 암호 방식의 키 숫자는 $2n$개수가 된다. 그리고 공개키 암호 방식의 구조상 실질적으로 전체 가입자에 대하여 비밀리에 보관해야 하는 키는 하나이다. 그러나 비밀키 암호 방식의 경우가입자 n명에 대하

여 (n-1)개의 가입자 수만큼의 비밀키를 보관해야만 하는 어려움이 있다. [2]

한편, 공중통신망의 암호통신을 위해서는 암호 송신자의 신원을 확인(Authentication)해야 하는 문제가 발생하게 되는데, 비밀키에서는 단일키의 특성상 새로운 정당한 검증기관을 통하여서만이 가능하지만 공개키 암호 방식을 이용하면 간단하게 신원확인을 할 수 있다.

이러한 신원 확인은 공개키 암호 방식을 이용한 디지털 서명(Digital signature)을 사용하여 구현이 가능하다. 즉, 그림 7 과 같이 송신자가 자신의 이름 등을 자신만의 비밀키로 암호화하여 보내고자 하는 평문에 첨부하여 보낸다. 그러면 수신자는 송신자가 공개한 공개키로 그 서명을 해독하여 그 서명이 송신자로부터 온 것 인지의 여부를 확인할 수 있다. 또한 평문의 비밀유지와 그 평문을 보낸 사람의 서명의 확인이 모두 중요한 경우에도 이 공개키 암호 방식은 적용될 수 있다. 평문 및 서명 전체를 송신자 자신의 비밀키로 암호화한 다음 다시 수신자의 공개키로 암호화하는 이중 암호화를 한다. 수신자의 전달된 암호문을 해독하기 위하여, 그에 상응하는 공개키와 비밀키를 역으로 암호문에 적용시킨다. 즉, 먼저 수신자의 비밀키를 사용하여 수신된 암호문을 1차로 해독한 다음, 송신자의 공개된 공개키로 다시 2차 해독을 한다. [2][3]

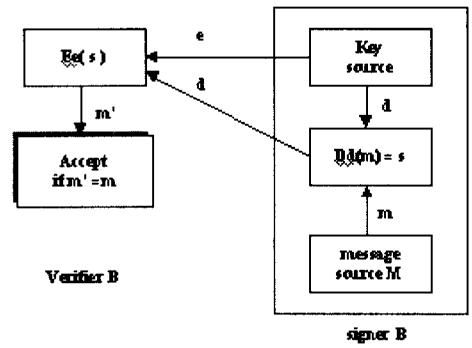


그림 7. 공개키 시스템의 디지털 서명구조

이러한 평문의 비밀유지와 송신자의 신분을 확인해주는 공개키 암호화 방식은 전자 문서 교환(EDI: Electronic Data Interchange)에 활용될 수 있으며 이외에도 원격 로그인 프로토콜, 다자간 제어

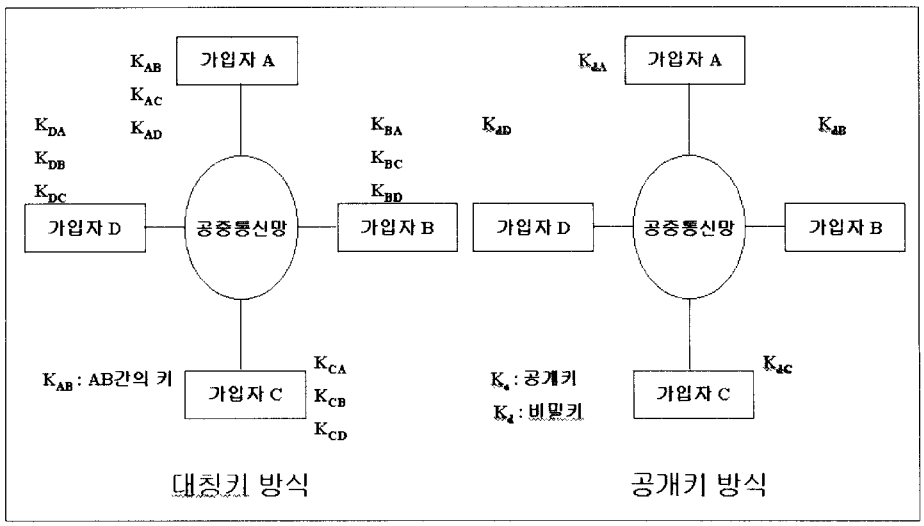


그림 6. 대칭키, 공개키 암호 방식에서 키분배 방법

시스템, 전자화폐, 전자대금결제(EFT: Electronic Funds Transfer), DB분산 운용 등에도 보다 효과적으로 적용될 수 있다. 공개키 암호 방식은 암호학적 비도를 증가시키기 위하여 소인수 분해문제, 이산대수문제, 배낭 문제 등을 이용하여 알고리즘을 작성하고 있으며 실제로 복잡도가 높은 알고리즘의 고속연산을 위하여 소프트웨어 이외에도 암호화 칩(chip)등의 하드웨어적인 방법에 의해서도 구현되고 있다. [3]

4.2 RSA 공개키 암호시스템

1978년 MIT의 Rivest, Shamir와 Adleman이 처음으로 제안한 RSA시스템은 암호화와 전자서명 모두를 제공할 수 있으며 약 200자리 이상의 정수의 소인수분해의 어려움에 그 안정성을 근거로 하고 있다. 즉, 두 큰 소수 p와 q의 곱셈 n은 계산하기가 쉬우나 n으로부터 p와 q를 추출하기는 어렵다는 사실에 그 기반을 두고 있다. 이러한 RSA 시스템의 암호화와 복호화 과정의 핵심연산은 모듈러 지수 연산(Modular Exponentiation, C := AE mod N)으로써 암호화 비도를 증대시키기 위하여 512비트 이상의 큰 수의 모듈러 지수연산을 사용하여 많은 수의 메시지 블록을 연속적으로 처리하게 된다. 모듈러 지수연산은 AB mod N형태의 모듈러 곱셈(Modular Multiplication) 연산을 반복적으로 수행함으로써 구현될 수 있으며, 현재 제안된 알고리즘 중에서 몽고메리(Montgomery) 방법이 가장 빠르고 효과적인 것으로 알려져 있다. 그리고 안전성과 속도향상을 위하여 입출력채널이 고정된 확장 가능한 모듈형태의 칩(chip)으로 제작하려는 연구가 활발히 진행되고 있다. [5][6][7][8]

4.2.1 RSA의 키 생성과 연산

RSA 알고리즘은 위에서 언급했듯이 약 십진수 200자리 정수의 소인수 분해의 어려움에 그 안전성을 근거로 하고 있는 가장 간단한 공개키 암호시스템의 하나이다. 그림 8은 전반적인 RSA시스템 구성을 나타내고 있다.

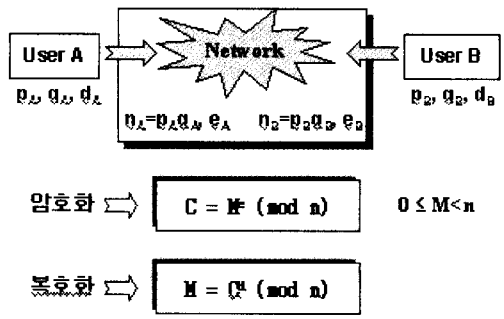


그림 8. RSA 암호시스템 개요

사용되어지는 키들은 n, p 와 q 그리고 e 와 d 이다. Modulus N은 임의의 큰 소수 p, q의 곱 (N := pq)이며 N-1보다 적은 자연수 중에서 N과 서로소의 관계를 가진 원소의 개수를 φ(N)이라고 정의하면 특별히 소수인 p에 대하여 φ(p) := p-1 이다. 공개키에 해당하는 e는 범위가 '1〈e〈φ(N)'이며 식(1)과 같이 φ(N)와 서로 소이고,

$$GCD(\ e,\ \varphi(N)\ ) = 1 \qquad (1)$$

큰 정수 N에 대하여 φ(N)을 구하기 위해서는 N의 소인수 분해가 필수적이며 N의 오일러 함수에 의하여 식(2)와 같이 구할 수 있다.

$$\varphi(N) = (p-1)(q-1) \qquad (2)$$

그리고 식(1)과 e가 φ(N)와 서로 소인 조건을 만족하는 임의의 수 e를 결정하면 비밀키에 해당 하는 멱승 인자 d 는 $e \cdot d \equiv 1 \bmod(\varphi(N))$에 의해서 식(3)으로 결정할 수 있다.

$$d = e^{-1} \bmod (\varphi(N)) \qquad (3)$$

암호화 과정의 수행은 각 키들이 먼저 결정되면

공개키에 해당하는 멱승 인자 e에 의해서 식(4)와 같이 수행된다.

$$C = M^e \pmod N \qquad (4)$$

여기서 M은 $0 \le M \le N$인 평문 블록을 나타내며 C는 RSA 암호화 과정에서 생성되어지는 암호문 블록이다. 복호화 과정은 비밀키에 해당하는 d 에 의하여 식(5)와 같이 암호화의 역과정으로 수행되게 된다. [6][8]

$$M = C^d \pmod N \qquad (5)$$

4.2.2 RSA 시스템의 안전성

RSA 암호 방식의 안전성은 소수 p와 p에 달려 있다. 공개 암호화 키 e와 N을 가지고 간단하게 비밀 복호화키 d를 찾을 수 있다면 RSA 암호 방식은 쉽게 해독된다. 또한 N으로부터 소수 p와 q 찾을 수 있다면 즉, N의 소인수 분해가 가능하면 오일러 함수 φ(N)을 찾게 되어 유클리드 호제법으로 공개 암호화 키 e로부터 비밀 복호화 키 d를 간단하게 찾아낼 수 있다. 현재 N의 소인수 분해를 모르고 오일러 함수 φ(N)을 찾아내는 방법은 알려져 있지 않다. RSA 암호 방식의 안전성을 보장받기 위한 소수 p와 q의 선택 조건과 공개 암호화 키 e와 비밀 복호화 키 d의 조건들이 부가적으로 필요하다. 소수 p와 q는 다음 조건을 만족해야 RSA 암호 방식의 안전성이 증가된다.

- p와 q는 거의 같은 크기의 소수이어야 한다.
- p-1과 q-1은 큰 소수를 인수로 가져야 한다.
- p-1과 q-1의 최대 공약수는 작아야 한다.

위와 같은 조건은 $n = p \cdot q$의 소인수 분해를 어렵게 만들므로 이를 이용한 RSA 암호 방식의 공격을 어렵게 만든다. 현재까지 알려진 소인수 분해 알고리듬 중 가장 효과적인 방법의 복잡도는 $\exp((O(1)+1)\log n \log\log n)^{1/2}$이다. 소수 p와 q의 크기가 100자리이고 N이 200자리인 합성수라면 현재의 소인수 분해 알고리듬과 전자공학 기술로는 이러한 N을 소인수 분해하는 것은 어려운 것으로 알려져 있다. RSA 암호 방식을 상용화한 장비들은 512비트의 N, 약 155자리 수를 사용하거나 664비트, 1024비트의 N을 사용한다. 따라서 RSA 암호 방식에서 N의 크기에 따른 암호화, 복호화 과정의 계산량 증가가 상용화의 장애요인으로 지적되고 있다. [2][7][10]

4.2.3 RSA 암호시스템의 구현과 현황

RSA 암호시스템은 주연산인 모듈러 곱셈에서의 안정성을 위하여 512비트 또는 1024비트 정도의 큰 수의 연산이 필요하다. 그래서 소프트웨어로 구현하기보다는 하드웨어로 구현에 대한 연구와 상용화 주로 이루어지고 있다. 가장 효율적인 모듈러 곱셈 알고리즘으로 알려진 몽고메리(Montgomery) 알고리즘은 기존의 모듈러 곱셈연산이 나눗셈을 수반해야 하는데 반하여 한 스텝에서 덧셈과 모듈러 감소연산을 동시에 수행함으로써 반복적인 모듈러 곱셈연산에 효과적으로 적용될 수 있다. 이러한 몽고메리 알고리즘과 지수연산을 빠르게 할 수 있는 바이너리 방법을 적용하여 2차원 또는 1차원 시스톨릭어레이(systolic array)로 구성하여 입출력 채널이 고정된 확장 가능한 모듈형태의 시스템 구현이 가능하다. 그러나 이러한 RSA 시스템의 수행속도는 DES와 같은 대칭키 암호화 시스템에 비해 통상 아주 느리다. 그러므로 실제 RSA 시스템은 대칭키 암호화 시스템의 암호화키와 같은 작은 데이터의 암호화에 사용되고 있다. RSA 암호시스템의 특허는 미국과 캐나다에서 가지고 있으며, 몇몇 표준에서 RSA 암호시스템을 암호화, 전자서명 그리고 키 설립을 위해 사용하고 있다. 그림 9 는 하드웨어적으로 구성된 모듈러 곱셈기를 모듈러 멱승 코어를 구현하고 이를 적정크기의 VLSI 어레이를 제작한 후

문제크기에 필요한 개수로 연결함으로써 연속적으로 메시지를 고속으로 처리할 수 있도록 구성한 RSA 시스템 구성도를 나타낸다.

최근에는 RSA 암호 시스템의 안전성에 대한 문제가 여러 곳에서 제기되고 있으며 다른 공개키 알고리즘의 안전성에 대한 평가가 계속되고 있다. 타원곡선암호시스템은 더 작은 길이의 키를 사용하여 일반 다른 공개키 알고리즘과 같은 안정도를 제공한다. [5][8][9]

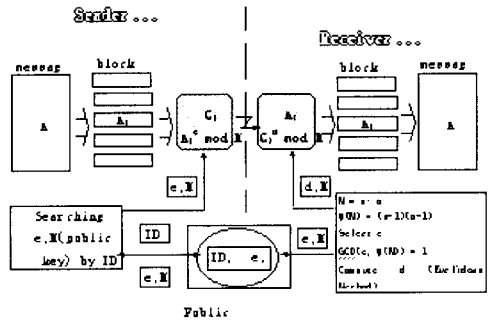


그림 9. RSA 암호시스템 구성도

4.3 타원곡선 암호 시스템(Elliptic Curve Cryptosystem;ECC)

타원곡선(Elliptic Curves)은 약 150년 전부터 수학적으로 광범위하게 연구가 되어왔다. 타원곡선을 이용한 공개키 암호시스템은 유한체(finite fields) 위에서 정의된 타원곡선 군에서의 이산 대수 문제에 기초한다. 타원곡선 암호 시스템은 1985년 N. Koblitz와 V. Miller에 의해서 처음 제시되었다. 타원곡선 암호 시스템은 비트 당 안전도가 타 공개키 시스템보다 효율적이라고 알려져 있다.

4.3.1 타원곡선 암호시스템의 장점

첫째, 같은 유한체 위에서 군을 제공하는 다양한 타원곡선을 사용할 수 있다. 즉 다양한 암호 시스템의 설계가 용이하다. 둘째, 군에서의 Sub-exponential time algorithms가 존재하지 않는다. 따라서 안전한 암호시스템을 설계할 수 있다. 단, 초 특이 타원곡선의 경우에는 이산 대수 문제가 유한체 위에서의 이산대수 문제로 바뀔 수 있는 것이 증명되었다. 이는 Sub-exponential time algorithms가 존재한다는 것이고 완전 지수 복잡도로 타원곡선 암호 시스템이 깨진다는 것을 의미한다. 그러나 연산의 효율성의 높이기 위해서 초 특이 타원 곡선을 사용할 수도 있다.

셋째, 타원곡선 암호시스템은 일반 다른 공개키 알고리즘과 같은 안전도를 제공하는 데에 더 작은 키의 길이를 사용한다. 예를 들어, RSA의 1024비트 키와 ECC 160비트 키를 갖는 암호 시스템은 같은 안전도를 갖는다. ECC의 짧은 Key 길이의 장점은 네트워크 상에서 대역폭과 메모리가 작아지는 것을 의미하므로 메모리 사용과 처리 능력의 제한이 심한 스마트 카드와 휴대폰과 같은 응용에 적용될 수 있다. 넷째, 타원 곡선에서의 덧셈 연산은 유한체에서의 연산으로 표현되어지므로, H/W와 S/W로 구현하기가 쉽다.

4.3.2 타원 곡선의 연산

본 고에서는 타원 곡선 E를 단순화하기 위해 K = Fp = Zp(p: 소수, p개의 원소를 갖는 유한체)로 하고 그 위에서 정의된 타원 곡선 E를

$y^2 = x^3 + ax + b$ 로 한정한다.

$E(Z_p) = \{(x, y) \in (Z_p, Z_p) \mid y^2 = x^3 + ax + b \ \ (a, b \in Z_p)\} \cup \{0\}$

ⓐ $P + 0 = 0 + P \ \ (\forall P \in E(Z_p))$

ⓑ $P = (x, y) \in E(Z_p) \rightarrow (x, y) + (x, -y) = 0 \quad (-P = (x, -y))$

ⓒ $P = (x_1, y_1) \in E(Z_p), \quad Q = (x_2, y_2) \in E(Z_p)$
$(P \neq Q) \rightarrow P + Q = (x_3, y_3)$

$x_3 = \lambda^2 - x_1 - x_2$

$y_3 = \lambda(x_1 - x_3) - y_1$

$\lambda = \frac{y_2 - y_1}{x_2 - x_1}$ if $P \neq Q$

$\frac{3x_1^2 + a}{2y_1}$ if $P = Q$

그림 10. 타원 곡선의 연산

그림 10 은 타원곡선에 사용되는 연산에 대한 정리이다.

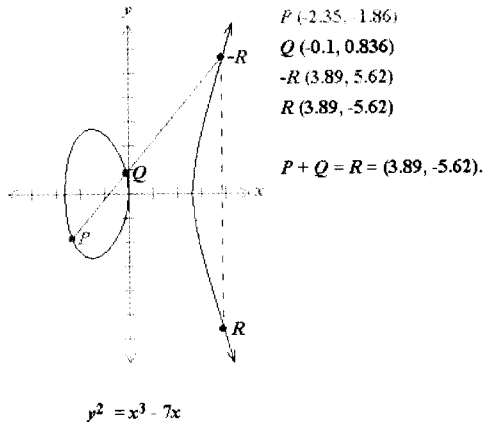


그림 11. 타원 곡선의 그래프

그림 11은 타원 곡선의 덧셈 연산을 좌표 상에 나타낸 것이다. 연산 과정을 단계적으로 살펴보면 다음과 같다.

(1) 타원 곡선 E 위의 임의의 두 점 P, Q를 선택한다.

(2) 두 점 P, Q를 잇는 직선 ㉠을 긋는다.

(3) 타원곡선이 3차 방정식이므로, 직선㉠은 P, Q 이외의 만나는 점 -R(P * Q)이 생긴다

(4) P*G와 무한 원점 (0)을 잇는 직선 ㉡을 긋는다.

(5) (3)과 같은 이유에서 타원곡선과 직선㉡은 만나고 다른 점 R(P + Q)이 생긴다.

타원 곡선 암호 시스템은 타원곡선 위의 점 P를 x번 더하는 계산이 주를 이룬다. 즉, Q = xP를 구하는 덧셈 연산은 모듈러 곱셈을 통해 이루어진다. 주요 공개키 암호 시스템은 효율적인 모듈러 곱셈에 의존하고 있으며, 타원 곡선은 소수 p의 값이 다른 시스템 보다 작기 때문에 그 효율성이 높다. [2][12]

4.3.3 타원곡선을 이용한 키분배

A와 B는 우선 자기 자신의 개인키와 공개키를 만들기 위해 유한체 Fq와 그 위에 정의된 타원 곡선 E를 정하고 그 위의 점 β를 정한다. β는 일반적인 곱셈군의 Diffie-Hellman에서 생성자 g의 역할을 한다. 그러나 β는 반드시 생성자일 필요도 없고 타원 곡선의 점의 개수 N을 정확히 알 필요도 없다. β는 위수가 충분히 큰 E위의 점이면 된다. 그 다음 A와 B는 각각 임의의 정수 난수 a, b를 선택하고 자신의 개인키로 보관한다. 또한 공개키 $a\beta$, $b\beta$를 계산하여 서로 교환한다. 그리고 공통된 비밀키로서 $P = ab\beta$를 사용한다. A는 B의 공개키 $b\beta$에 자신의 개인키 a를 적용시켜 P를 알 수 있고, B도 같은 방법으로 P를 알 수 있다.

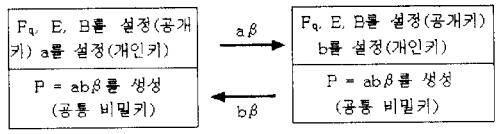


그림 12. 타원곡선 키 분배 과정

4.3.4 타원 곡선을 이용한 ElGamal

E 위의 점 β를 선택한다. 그리고 사용자들은 각각 정수 난수를 a를 개인키로 선택한다. 그리고 $a\beta$를 계산하여 서로 교환하다. 만약 A가 B에게 메시지 Pm을 보내려 한다면 A 정수 난수 k를 선택하여 점들의 순서쌍 $(k\beta,\ P_m + k(a_B\beta))$를 보낸다. 이때, $a_B\beta$ 는 B의 공개키이다. 메시지를 읽기 위해 B는 자신의 공개키 $a_B\beta$를 이용해 다음을 계산한다.

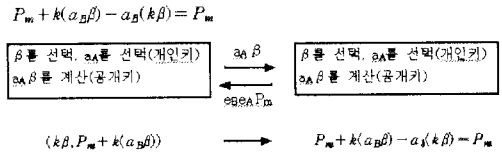


그림 13. ElGamal ECC

4.3.5 ECDSA(Elliptic Curve Digital Signature Algorithm)

ECDSA는 DSA를 타원 곡선으로 변형시킨 것이다. ECDSA와 DSA의 중요 차이점은 r의 생성에 있다. DSA는 r을 임의의 gk mod p를 선택하고 계산한 후, mod q를 계산하여 얻는다. 그러나 ECDSA에서 r은 임의의 점 kF 의 x 좌표를 mod n 하여 얻는다. ECDSA가 160 비트 q와 1024비트 p를 가진 DSA와 비슷한 안전도를 갖기 위해서는 매개변수 n 이 약 160비트이면 된다. 이 경우 DSA와 ECDSA는 같은 서명 길이(320비트)를 갖는다.

4.3.6 타원 곡선 암호시스템과 공개키 암호 시스템의 비교

공개키 암호시스템의 이론적 안전도를 조사하기 위해서 우선적으로 시스템 안전성의 기반이 되고 있는 수학적 문제를 푸는데 얼마만큼의 어려움이 요구되는 가를 분석하는 것이다.

타원 곡선에서의 이산 대수 문제(ECDLP)는 초특이 타원곡선이라고 하는 타원 곡선에서의 작은 부분에 속해 있는 곡선 위에서 정의된 경우, 그 문제 해결은 다른 일반적인 곡선 위에서 이산대수 문제의 해결보다 쉬운 것으로 알려져 있다. 이러한 초 특이 타원 곡선은 쉽게 확인 될 수 있고 따라서 쉽게 피할 수도 있다.

공개키와 비밀키를 계산하는 데에 드는 계산량을 비교하여 보면 RSA의 경우, 서명 검증과 암호화의 속도를 빠르게 하기 위해 짧은 공개키를 사용한다. 그러나 이런 짧은 key의 사용이 RSA의 안전에 영향을 준다. DSA와 ECC는 서명 생성과 암호화에서 사전 계산이 많은 부분을 차지할 수 있다. ECC의 경우 RSA또는 DSA보다 약 10배 정도의 빠른 수행 속도를 보여준다. 공개키와 비밀키와 시스템 매개변수의 저장에 드는 key size를 살펴보면 표1과 같다.

	시스템매개변수(bit)	공개키 (bit)	비공개키 (bit)
RSA	n/a	108	2048
DSA	2208	1024	160
ECC	481	161	160

표 1. 시스템 매개변수와 크기

표 2 는 2000비트의 메시지에 서명하는 것과 100비트의 메시지를 암호화하는 것의 경우를 나타낸 것이다.

	서명 크기(비트)	암호화된 메세지(비트)
RSA	1024	1024
DSA	320	2048
ECC	320	321

표 2. 각 암호 시스템의 서명과 암호화된 메시지의 크기 비교

앞에서 살펴본 바와 같이, ECC는 계산량, key 크기, 대역폭의 관점에서 다른 공개키 암호 시스템보다 더 효율적이다. 이것은 실질적인 사용을 위한 구현에 있어서 높은 속도와 코드 크기의 감소 등의 장점을 가지고 있음을 알 수 있다. [11]

5. 해시 함수

전자상거래에서 네 가지 보안 요소 중 기밀성은 암호화 알고리즘을 통해 얻을 수 있다. 그리고 해시 함수는 인증된 소스에 의해 생성되고 전송되고 저장된 데이터가 불법적인 방법에 의해 바뀌지 않았음을 보장하는 데이터 무결성, 소스로부터의 메시지인지를 인증하는 메시지 인증, 사용자 인증, 디지털 서명 등에 쓰이며 암호알고리즘과 함께 정보 통신 보호의 핵심기술로 사용되고 있다. 암호학상에서 해시 함수 h는 임의의 크기의 메시지 M을 입력으로 받아 고정된 길이의 해시값(compression) h(M)을 쉽게

계산 할 수 있는 함수이다. 그리고 해시 함수는 두 개의 입력 매개변수 메시지와 키를 가지는 MDCs(manipulation detection codes)알고리즘의 keyed 방식과 하나의 입력 매개변수를 가지는 MAC(Message authentication code)알고리즘의 unkeyed방식으로 나눌 수 있다.

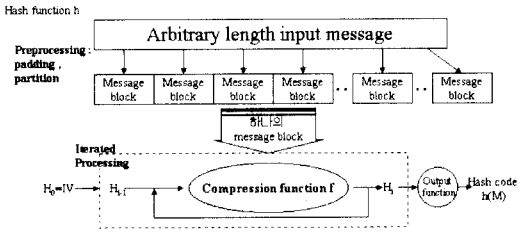


그림 14. 일반적인 해시 함수 구조

그림 14 는 일반적인 해시 함수의 구조이다. 임의의 메시지들을 블록단위로 분할과 패딩의 과정을 거친 후 분할된 하나의 메시지 블록은 해시 함수의 압축함수의 입력으로 들어가는 구조이다.

5.1 Unkeyed 방식 해시 함수(MDCs)

Unkeyed 해시함수는 다음의 3가지 특성을 만족하여야 한다.

(1) preimage resistance : h와 M이 주어졌을 때 H=h(M)을 계산하기 쉬워야 하는 성질로 일방향성이다

(2) second preimage resistance : h가 주어지고 선택된 입력 M에 의하여 같은 출력 값을 갖는 (h(M)=h(M′)를 만족하는 M′를 찾는 것은 계산적으로 불가능한 성질로 weak one-way hash function이다

(3) collision resistance : h가 주어졌을 때 같은 출력을 갖는 두 입력을 찾기가 계산적으로 불가능해야 한다는 조건으로서 strong one-way hash function이다

또한 unkeyed 방식의 MDCs를 두 가지 분류로 다시 나눠본다면 One - way hash funtions (OWHFs)은 pre-image와 second preimage resistance조건이 있어야 하며 Collision resistant hash function(CRHF)는 second preimage와 collision resistance조건을 가져야 한다. 해시 함수는 국제 표준 ISO/IEC 10118에서 총 4부로 구성하며 표준화가 진행 중에 있다.

제1부 : 일반사항
제2부 : n비트 블록 암호알고리즘을 이용한 해시 함수
제3부 : 전용 해시 함수
제4부 : 모듈러 연산을 이용한 해시 함수.

그림 15에 보인 바와 같이 n비트 블록 암호알고리즘을 이용한 해시 함수는 기존의 블록 암호알고리즘을 이용할 수 있어서 추가적인 cost가 적다.

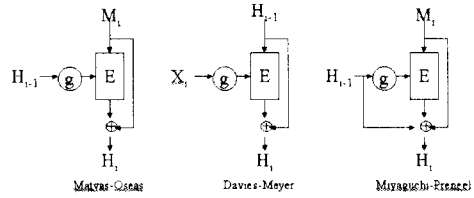


그림 15. 블록 Cipher를 기반으로 한 MDC

n비트 블록 암호알고리즘을 이용한 해시 함수는 해시 입력 메시지 블록 당 출력결과 길이에 따른 single-length hash functions, double-length hash function이 있다. 그림 16은 single-length hash function scheme이며 double-length hash function에는 DES기반의 MDC-2 와 MDC-4가 있다.

대표적인 전용 해시 함수로는 MD(message digest)계열의 MD4, MD5, RIPEM-160, SHA-1 등이 있으며 이중 R.L.Rivest에 의해 제안된 MD계열의 MD5가 널리 사용되고 있다. MD5는 MD4의 제3라운드의 공격에 보완하기 위해 개발된 알고리즘으로 32비트 기반구조를 가진 빠른 알고리즘으로 구성한 효율성, 많은 양의 계산 알고리즘이나, 치환표의 사용이 없는 단순성, 또한 해시 함수의 일반적인 안전성을 보장하는 특징을 가지고 있다. MD5는 512비트 입력메시지 블록을 가지고 128비트의 해시 값을 산출한다.

MD5 512비트 하나의 메시지를 처리하는 해시 함수는 4개의 라운드로 구성되며 각 라운드는 16개의 기본 step 연산의 반복 수행으로 처리된다. 기본 step연산은 다음과 같다.

a〈- b+((a+g(b.c.d)+X[i]+T[i])〈〈s

(a,b,c,d : 각 32비트 연쇄변수, g:비선형 함수, X[i]:16개의 32비트 메시지 블록, T[i]: 상수)

SHA(Secure Hash Algorithm)은 MD4기반이며 미국 NIST에서 개발되었다. MD4와 다른 점은 해시 값이 160비트라는 적과 5개의 32비트 연쇄 변수를 사용한다. 또한 압축 함수의 라운드 개수는 4개이며 MD4가 little-endian인데 반해 big-endian이다.

RIPEMD-160 : MD4기반이며 MD4와 다른 점은 많은 연쇄변수, 많은 라운드수, 라운드 함수를 가지며 little-endian방식이다.

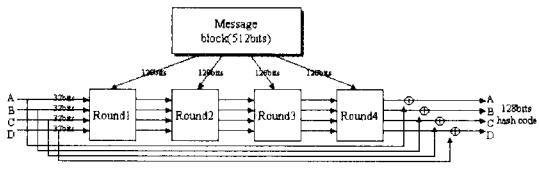


그림 16. 전용 해시 함수 MD5 메시지 처리 구조

5.2 모듈러 연산을 기반으로 한 해시 함수

압축함수에 mod M수학연산을 이용하여 반복 해시 함수를 구현하였다. 모듈러 연산을 위해 이미 존재하는 S/W, H/W의 재사용, 필요한 안전성 강도에 따른 확장성의 특징을 가지나 속도가 느린 단점을 가지고 있으며 대표적인 모듈러 연산 기반 해시 함수로는 MASH가 있다.

5.3 Keyed 방식 해시 함수(MAC)

Keyed hash function은 message authentication code 알고리즘이라는 메시지 소스와 무결성을 보장하기 위한 목적을 가지고 있으며 MAC 알고리즘은 비밀키 K 매개변수를 갖는 해시 함수 hk이다.

MAC이 갖는 특징은 연산의 용이성, 축약 그리고 다음의 성질을 가져야 한다.

computation-resistance : text-MAC 쌍 (x I ,hk(xi))가 주어졌을 때, 새로운 입력 x(x=xi).를 위한 text-MAC쌍(x,hk(x))을 계산하는 것은 불가능해야 한다.

가장 널리 사용되는 MAC알고리즘은 block-chaining을 사용하는 블록암호 알고리즘을 기반으로 한 것으로 DES가 블록암호로 사용될 때, MAC key 는 56비트 DES키이다.

그림 17은 DES-CBC기반 MAC알고리즘이다.

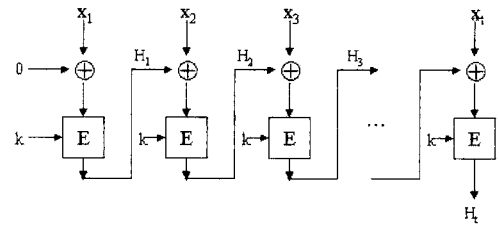


그림 17. CBC기반 MAC알고리즘

이러한 데이터 무결성을 보장하는 해시 함수의 사용 방법은 그림 18과 같다. 또한 디지털 서명에서 해시 함수를 사용함으로써 앞에서 설명한 네 가지 보안 기능 중 인증, 무결성, 부인방지를 한번에 해결해 줄 수 있다. 송신자가 수신자에게 보내려는 메시지를 해시 함수를 통해 생성한 메시지 다이제스트를 송신자의 개인키로 암호화 한 것이 전자서명인데 인증 측면에서 보게되면 송신자의 공개키로 전자서명을 복호화한 내용과 해시 코드 값과 비교함으로써 전자서명을 만든 사람의 사용자 인증을 수행하며, 전자서명 복호문과 해시 값이 같다는 것은 원문이 전송 중 변조되지 않았음을 의미하며, 송신자가 후에 부인을 할 때에 원문과 전자서명을 제시함으로써 부인을 방지 할 수 있다.

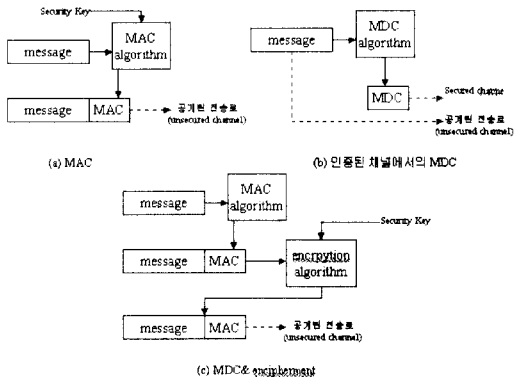


그림 18. 해시 함수를 사용한 데이터 무결성

※ 참 고 문 헌

[1] Schneier, Bruce : "Applied Cryptography" Second Edition , John Wiley & Son, Inc, 1996.

[2] Alfred J. Menezes, Paul C.van Oorschot, Scott A. Vanstone : "Handbook of Applied Cryptography", CRC Press, 1997.

[3] 김 철, "암호학의 이해", 영풍문고, 1996.

[4] 이인수, "RSA 공개키 암호화시스템 현황", 한국정보보호센터, 1998.

[5] C. D. Walter. "Systolic Modular Multiplication." IEEE Trans. on Computer. Vol. 42, No. 3. pp. 376-378.

[6] Ç.K. Koç "High-Speed RSA Implementation." Technical Report TR201, RSA Laboratory, November 1994.

[7] Ç.K. Koç "RSA Hardware Implementation." Technical Report Version 2.0, RSA Laboratory, August 1995.

[8] 하재철, 문상재, "시스톨릭 어레이를 이용한 Montgomery 모듈라 곱셈기 설계." 통신정보보호학회 학술대회 논문집, 제9권 제1호, pp. 135-146, 1999. 3.

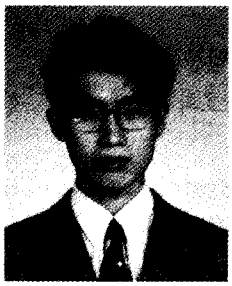
[9] http://security.ece.orst.edu/koc/.

[10] http://www.rsa.com.

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

[12] http://www.certicom.com.

[13] http://www.setco.org.

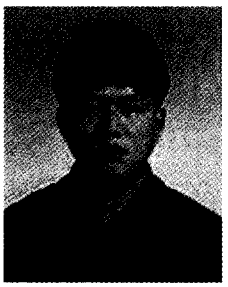


염 동 복

1998. 2. 한국항공대학교 컴퓨터공학과 졸업
1999.11. 한국항공대학교 컴퓨터공학과 대학원 재학 (석사과정)
※관심분야 : 마이크로 프로세서, Network Security, 디지털 논리 회로 설계 등

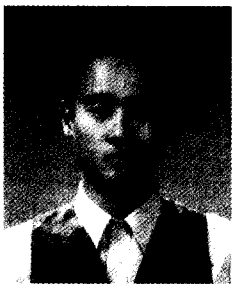
김 환 국

1998. 2. 한국항공대학교 컴퓨터공학과 졸업
1999.11. 한국항공대학교 컴퓨터공학과 대학원 재학 (석사과정)
※관심분야 : Network Security, hash function, 디지털 논리 회로 설계, 전자 상거래 등



박 재 철

1998. 2. 한국항공대학교 컴퓨터공학과 졸업.
1999.11. 現 한국항공대학교 컴퓨터공학과 대학원 재학 (석사과정)
※관심분야 : 마이크로 컨트롤러, Network Security, 암호알고리즘 설계, 디지털 연산 등



이 병 윤

1999. 2. 한국항공대학교 컴퓨터공학과 졸업
1999.11. 現 한국항공대학교 컴퓨터공학과 대학원 재학 (석사과정)
※관심분야 : Network Security, 암호 알고리즘 하드웨어 설계, 디지털 논리 회로 설계, ECC 등

박 종 서

1983. 2. 한국 항공대학교 통신공학과 학사 졸업
1987.12. 미 North Carolina State University 컴퓨터공학 석사
1994. 8. 미 Penn State University 컴퓨터공학 박사
1996. 2. 미 Penn State University 컴퓨터공학과 조교수
1999.11. 現 한국 항공대학교 컴퓨터공학과 조교수
※관심분야 : Network Security, 항공 우주용 제어기 설계, VLSI