제 1 장 암호알고리즘 개요

1.1 암호학 개요

1.1.1 기본 용어

이 절에서는 먼저 암호알고리즘과 관련된 기본적인 용어부터 살펴본다. **암호알고리즘**(cryptographic algorithm)이란 좁은 의미에서 **평문**(plaintext, cleartext)을 **암호문**(ciphertext)으로 변환하고, 암호문을 다시 평문으로 변환할 때 사용되는 알고리즘을 말하며, 넓은 의미에서는 암호기술에서 사용되는 모든 알고리즘을 말한다. 여기서 평문이란 누구나 그 내용을 보면 그 내용을 이해할 수 있는 문서를 말하며, 암호문은 그 자체로는 누구도 그 내용을 이해할 수 없는 문서를 말한다. 암호문의 내용을 이해하기 위해서는 그것을 다시 평문으로 변환해야 가능하다. 이 때 평문을 암호문으로 바꾸는 과정을 **암호화**(encryption, encipherment)라 하고, 암호문을 다시 평문으로 바꾸는 과정을 **복호화**(decryption, decipherment)라 한다. 현대 암호알고리즘에서는 암호화와 복호화하는 과정에 **암호키**(cryptographic key)가 필요하며, 이 키가 없으면 암호문을 다시 평문으로 변환할 수 없다.

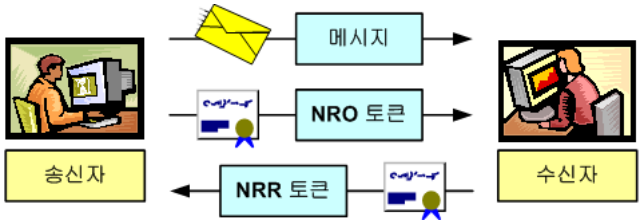
암호학(cryptology)는 이와 같이 메시지를 보안목적으로 변환하는 기술에 관한 학문을 말하며, 크게 **암호기술**(cryptography)와 **암호해독기술**(cryptoanalysis)로 분류할 수 있다. 암호기술은 메시지를 안전하게 유지하는 기술과 학문을 말하며, 암호해독기술은 암호문으로부터 평문을 추정하는 기술과 학문을 말한다.

1.1.2 보안 목적

암호알고리즘을 사용하여 제공되는 보안 서비스 중 가장 중요한 세 가지 서비스는 **비밀성**(confidentiality, secrecy, privacy), **무결성**(integrity), **인증**(authentication)이다. 이 외에 다양한 목적을 위해 암호알고리즘을 사용된다. 비밀성이란 인가된 사용자들만 데이터의 내용을 볼 수 있도록 해주는 서비스를 말하며, 무결성이란 비인가된 데이터의 변경을 발견할 수 있도록 해주는 서비스를 말한다. 여기서 데이터의 변경이란 인가되지 않은 데이터의 삽입, 삭제, 교체를 말한다. 무결성의 경우 비밀성과 달리 데이터가 변경되지 않도록 보장할 수는 없다. 따라서 변경을 발견할 수만 있으면 무결성이 제공된다고 한다. 인증은 식별(identification)과 검증(verification)을 합쳐진 말로서, 주장된 것을 검증하는 것을 말한다. 인증은 크게 메시지 인증, 메시지 원천 인증, **개체 인증**(entity authentication)으로 구분된다. 메시지 인증은 무결성 검증과 같은 말이며, 메시지 원천지 인증은 메시지의 송신된 위치 또는 송신자를 검증하는 것을 말한다. 정보보안에서는 보통 위치는 중요하지 않다. 정보보안에서 가장 중요한 것은 개체 인증이다. 개체 인증이란 주장된 신원을 검증하는 것을 말한다.

세 가지 중요 서비스 외에 가장 많이 요구되는 서비스 중 하나는 **부인방지**(non-repudiation)이다. 부인방지는 개체가 지난 행위나 약속을 부인하지 못하도록 하는 서비스를 말한다. 예를 들어 전자서명을 한 사용자는 나중에 자신이 서명한 사실을 부인할 수 없어야 한다. 메시지의 송신과 수신과 관련하여 국제표준에서는 크게 네 가지 부인방지 서비스를 정의하고 있다. 이들 서비스는 송신자와 수신자가 통신에 참여한 사실을 나중에 부

인할 수 없도록 해준다. 이 중 **송신 부인방지**(NRO, Non-Repudiation of Origin)와 **수신 부인방지**(NRR, Non-Repudiation of Receipt)는 송신자와 수신자 간에 직접 통신하는 경우에 해당하며, **제출 부인방지**(NRS, Non-Repudiation of Submission)와 **전달 부인방지**(NRD, Non-Repudiation of Delivery)는 송신자가 제3의 중계기관을 통해 간접적으로 수신자에게 메시지를 전달하는 경우에 해당한다. 그림 1.1과 1.2는 두 가지 형태의 차이를 보여주고 있다. 부인방지 서비스는 보통 부인하고자 하는 측에서 상대방에게 부인방지 토큰을 만들어 제공한다. 나중에 부인하게 되면 상대방은 이 토큰을 제시하게 되며, 제3자들은 이 토큰을 통해 부인하고 있다는 것을 알게 된다.



<그림 1.1> 송신 부인방지와 수신 부인방지 개념

NRO 서비스에서는 송신자가 수신자에게 토큰을 제공하며, 수신자는 나중에 송신자가 송신 사실을 부인하면 이 토큰을 통해 부인 사실을 입증할 수 있다. NRR 서비스에서는 수신자가 송신자에게 토큰을 제공하며, 송신자는 나중에 수신자가 수신 사실을 부인하면 이 토큰을 통해 부인 사실을 입증할 수 있다.

<그림 1.2> 제출 부인방지와 전달 부인방지

제3의 중계기관을 사용하여 메시지를 전달하는 경우에는 송신자는 수신자와 직접 통신을 하지 않는다. 따라서 부인방지와 관련된 토큰들은 중계기관이 전달해 주어야 한다. NRS 서비스에서는 중계기관이 송신자에게 토큰을 제공하며, 송신자는 중계기간이 나중에 송신자로부터 메시지를 제출받았다는 사실을 부인하면 이 토큰을 통해 부인 사실을 입증할 수 있다. NRD 서비스는 중계기관이 송신자에게 토큰을 제공하며, 송신자는 수신자가 나중에 수신사실을 부인하면 이 토큰을 통해 부인 사실을 입증할 수 있다. 여기서 NRD 토큰은 수신자로부터 NRR 토큰을 받아 송신자에게 주는 형태가 될 수도 있다. 또 송신자는 중계기관에게 NRO 토큰을 전달해 줄 수도 있다.

1.1.3 통신계층과 암호기술

일반적으로 통신구조는 여러 계층으로 구성되어 있으며, 어떤 계층에서 암호기술을 적용

하느냐에 따라 다른 특성을 가지게 된다. 통신 계층에 따른 암호기술 적용은 크게 전송 계층 이상에서 이루어지는 경우와 네트워크 계층 이하에서 이루어지는 경우로 나눌 수 있다. 전송 계층 이상에서 이루어지는 경우를 **단대단**(end-to-end) **방식**이라 하며, 보통 응용 계층에서 암호기술을 적용하는 경우가 많다. 단대단 방식에서는 송신측에서 암호기술이 전체 메시지에 한 번 적용되어 적용된 상태로 목적지까지 전달된다. 반대로 네트워크 계층 이하에서 이루어지는 경우는 패킷 단위로 홉마다 암호기술이 적용되며, **홉-바이-홉**(hop-by-hop) **방식** 또는 **링크 방식**이라 된다. 두 방식의 차이점은 표 1.1과 같다.

<표 1.1> 단대단과 홉바이홉 암호기술 적용의 차이점

	단대단(end-to-end)	홉-바이-홉(hop-by-hop)
장점	● 통신망과 독립적으로 수행가능 ● 기반구조를 신뢰하지 않아도 됨	● 트래픽 분석이 가능하지 않음 ● 패킷 단위로 암호화가 가능하므로 오류 발생시 해당 패킷만 재전송
단점	● 트래픽 분석이 가능함 ● 오류 발생시 전체 메시지를 재전송해야 함	● 통신망의 각 호스트/스위치에 암호기술 능력이 있어야 함 ● 각 호스트마다 반복적으로 암호기술을 적용해야 하므로 효율성이 떨어짐 ● 기반구조를 신뢰해야 함

표1.1에서 알 수 있듯이 두 방식은 상반되는 특징을 가지고 있다. 단대단 방식에서는 상위층에서 전체 메시지에 암호기술이 적용되며, 하위 층에서는 일반 메시지와 동일하게 처리된다. 반면에 홉-바이-홉 방식에서는 하위 층에서 암호기술을 패킷 단위로 직접 적용한다. 만약 암호알고리즘을 통해 메시지를 암호화하여 비밀성을 제공하고자 할 경우 홉-바이-홉 방식에서는 패킷 단위로 각 홉마다 암호화를 반복적으로 적용해야 한다. 그러므로 효율성이 떨어질 뿐만 아니라 중간 호스트들이 패킷의 내용을 볼 수 있게 된다. 반면에 패킷 단위로 암호화를 하기 때문에 오류가 발생하면 해당 패킷만 다시 전송하면 되지만 단대단 방식에서는 전체 메시지를 다시 전송해야 한다.

1.2 암호알고리즘

1.2.1 암호알고리즘의 정의

암호알고리즘은 앞서 언급한 바와 같이 좁은 의미에서는 평문을 암호화하고 암호문을 복호화할 때 사용되는 알고리즘을 말하며, 넓은 의미에서는 암호기술에서 사용되는 모든 알고리즘을 말한다. 이 절에서는 우선 좁은 의미의 암호알고리즘에 대해 살펴본다. 현대 암호알고리즘은 모두 암호키를 사용하며, 알고리즘의 안전성은 이 키에 의존한다. 초기에는 암호알고리즘의 안전성은 암호알고리즘 자체의 비밀성에 의존하는 경우도 있었다. 이와 같은 암호알고리즘을 **제한적 알고리즘**(restricted algorithm)이라 한다. 제한적 알고리즘의 경우에는

각 사용자 쌍마다 다른 알고리즘을 사용해야 하기 때문에 하나의 알고리즘을 모든 사용자들이 공유할 수 없으며, 알고리즘이 노출되면 새로운 알고리즘을 개발해야 하는 문제점을 지니고 있다. 따라서 현재는 이와 같은 형태의 암호알고리즘을 사용하지 않는다. 오늘날에 암호알고리즘은 그것의 자세한 내부 내용까지 공개되어 있으며, 이 공개가 알고리즘의 안전성에 영향을 주지 않는다. 따라서 모든 사용자들은 동일한 암호알고리즘을 공유한다. 다만, 각 사용자 쌍마다 다른 암호키를 사용하며, 암호키가 노출되면 암호알고리즘을 변경하는 것이 아니라 암호키만 변경하면 된다. 암호알고리즘의 내부 내용의 공개가 암호문을 해독하는데 도움이 줄 수 있는 측면도 있지만 반대로 취약점을 빨리 발견할 수 있다는 긍정적인 측면도 있다.

평문을 암호문으로 변환하는 함수를 암호화 함수라 하고, 암호문을 다시 평문으로 복원하는 함수를 복호화 함수라 한다. 암호화 함수는 주로 다음과 같이 표기된다.

$$E(M,\ K) = C,\ E_K(M) = C,\ \{M\}.K$$

반면 복호화 함수는 주로 다음과 같이 표기된다.

$$D(M,\ K) = C,\ D_K(M) = C,\ \{M\}^{-1}.K$$

암호시스템(cryptosystem) 또는 **암호계**는 가능한 평문의 집합, 가능한 암호문의 집합, 가능한 암호키의 집합, 암호화 함수, 복호화 함수, 총 다섯 개 요소로 정의된다. 이 때 암호화와 복호화 함수는 D(E(M,K), K) = M을 만족해야 한다. 이것을 만족하기 위해 암호화 함수는 전단사함수(bijection)이어야 한다. 함수가 전단사함수라는 것은 역함수가 존재한다는 것을 의미하며, 역함수가 있어야 암호문을 다시 평문으로 바꿀 수 있다.

Kerckhoff는 암호시스템의 요구사항을 다음과 같이 정의하고 있다.

- **요구사항 1.** 암호시스템은 이론적으로 안전하지 않다면 최소한 실제적으로 안전해야 한다.
- **요구사항 2.** 시스템의 자세한 내부 사항의 노출이 시스템 안전성에 영향을 주지 않아야 한다.
- **요구사항 3.** 암호키는 기억 가능해야 하며, 쉽게 변경할 수 있어야 한다.
- **요구사항 4.** 암호문을 통신을 통해 전달 가능해야 한다.
- **요구사항 5.** 암호장치는 이동 가능해야 하며, 홀로 사용할 수 있어야 한다.
- **요구사항 6.** 암호시스템을 사용하는 방법이 쉬워야 한다.

대부분의 현재 암호시스템은 무조건적으로 안전하지 않다. 즉, 무한한 컴퓨팅 자원과 시간이 있으면 누구나 암호시스템을 해독할 수 있다. 따라서 실제적으로 안전하다는 것은 그 암호시스템을 해독하는 가장 효율적인 방법을 사용하더라도 현재의 컴퓨팅 능력으로는 너무 많은 시간이 소요되어 의미가 없다는 것을 말한다. 이와 같은 안전성을 계산적 안전성이라 한다. 요구사항 2는 사용하는 암호알고리즘이 제한적 알고리즘이 아니라 암호키에 의존하는 알고리즘이어야 한다는 것을 말하고 있다. Kerckhoff의 요구사항은 아주 오래전에 정의된 것이지만 오늘날에도 거의 그대로 적용되고 있다.

1.2.2 암호알고리즘의 분류

좁은 의미의 암호알고리즘은 평문을 암호화한 결과의 특성에 따라 **결정적**(deterministic)

또는 **확률적**(probabilistic) 암호알고리즘으로 분류할 수 있고, 알고리즘에서 사용하는 암호키에 따라 **대칭**(symmetric)와 **비대칭**(asymmetric) 암호알고리즘으로 분류할 수 있다. 결정적 암호알고리즘은 암호키와 평문이 동일하면 항상 암호문이 동일하지만 확률적 암호알고리즘은 암호키와 평문이 동일하더라도 항상 결과 암호문이 달라지는 암호알고리즘을 말한다. 따라서 확률적 암호알고리즘이 보다 안전하며, 암호알고리즘이 결정적 방식이더라도 랜덤 요소를 포함하여 쉽게 확률적 암호알고리즘으로 바꿀 수 있다. 대칭 암호알고리즘(symmetric key algorithm, conventional algorithm, secret key algorithm)은 암호화할 때 사용하는 암호키와 복호화할 때 사용하는 암호키가 동일하지만 비대칭 암호알고리즘(asymmetric algorithm, public key algorithm)에서는 두 키가 서로 다르다.

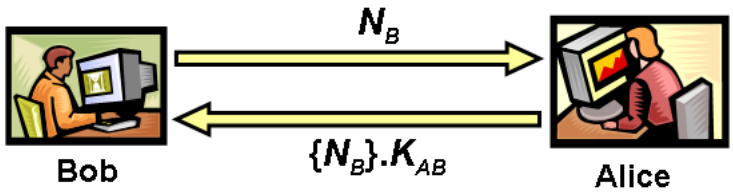
암호키는 사용하는 용도나 특성에 따라 다양한 용어로 표현된다. 암호화키(encryption key)는 암호화할 때 사용하는 암호키를 말하며, 복호화키(decryption key)는 복호화할 때 사용하는 암호키를 말한다. 비밀키(secret key)란 대칭 암호알고리즘에서 사용되는 암호키를 말한다. 공개키(public key)란 비대칭 암호알고리즘에서 모든 사람들에게 공개하는 암호키를 말하며, 개인키(private key)란 비대칭 암호알고리즘에서 각 사용자가 비밀로 유지하는 암호키를 말한다. 공개키는 항상 그것에 대응되는 개인키가 존재하며, 이 두 키를 공개키 쌍이라 한다. 끝으로 세션키(session key)란 보통 비밀키로서 특정 순간에 사용되고 폐기되는 암호키를 말한다.

1.2.3 대칭 암호알고리즘

대칭 암호알고리즘은 암호화키와 복호화키가 같은 암호알고리즘이라 한다. 다른 말로 비밀키 암호알고리즘이라 한다. 암호화키와 복호화키가 같아야 하므로 암호화하는 사용자와 복호화하는 사용자가 같은 암호키를 공유하고 있어야 한다. 이것을 어떻게 안전하게 공유할 것인지가 대칭 암호알고리즘을 사용할 때 가장 큰 이슈이다. 특히, 암호화하여 메시지를 비밀스럽게 전달하고자 하는 사용자와 그것을 받아야 하는 사용자가 원거리에 있을 경우에는 비밀키를 안전하게 공유하기가 쉽지 않다.

대칭 암호알고리즘은 크게 **스트림**(stream) 암호방식과 **블록**(block) 암호방식으로 분류할 수 있다. 스트림 암호방식은 평문 각 비트 또는 바이트를 하나씩 암호화하는 방식이고, 블록 암호방식은 평문을 일정한 블록 크기로 나누어, 각 블록을 암호화하는 방식이다. 대칭 암호알고리즘은 크게 다음과 같은 세 가지 용도로 사용된다.

- **용도 1.** 공개 채널로 전달되는 메시지에 대한 비밀성 보장
  - 송신자와 수신자가 공유하고 있는 비밀키로 메시지를 암호화하여 교환함으로써 제3자가 도청하여도 교환하는 내용을 알 수 없도록 만든다.
- **용도 2.** 기억장치에 저장되는 데이터에 대한 비밀성 보장
  - 다른 사용자가 저장된 데이터를 열람할 수 없도록 데이터를 암호화하여 저장할 수 있다.



<그림 1.3> 대칭 암호알고리즘을 이용한 인증 프로토콜

- **용도 3.** 개체 인증을 위해 사용
  - Alice와 Bob이 비밀키 K_{AB}를 공유하고 있을 경우 그림 1.3과 같은 프로토콜을 통해 인증할 수 있다. Bob은 랜덤한 수 N_B를 하나 생성하여 Alice에게 전달하면 Alice는 이것을 Bob와 공유하고 있는 K_{AB}로 암호화하여 되돌려준다. K_{AB}는 오직 Alice와 Bob만이 알고 있으므로 Bob은 Alice가 회신하였다는 것을 확신할 수 있다. 또한 N_B는 Bob가 이번 세션을 위해 처음 사용한 값이므로 Alice가 보내주신 암호문을 통해 이 메시지의 **최신성**(freshness)을 검증할 수 있다. 이와 같은 용도로 한번 사용하는 랜덤 수를 **난스**(Nonce, Number used just ONCE)라 한다.

1.2.4 비대칭 암호알고리즘

비대칭 암호알고리즘은 다른 말로 공개키 암호알고리즘이라 하며, 암호화키와 복호화키가 서로 다르다. 비대칭 암호알고리즘은 Diffie와 Hellman이 1975년에 처음으로 제안하였으며, 대칭 암호알고리즘에 비해 상대적으로 역사가 짧다. 대칭 암호알고리즘에서는 두 사용자가 어떻게 안전하게 비밀키를 공유할 지가 가장 중요한 이슈이지만 공개키 암호알고리즘에서는 두 사용자가 동일한 암호키를 공유하지 않아도 메시지를 비밀스럽게 교환할 수 있다. 공개키 암호알고리즘을 사용할 경우에 사용자는 해당 사용자의 공개키만 가지고 있으면 메시지를 비밀스럽게 전달할 수 있다. 하지만 공개키 암호알고리즘을 사용할 경우에는 또 다른 중요한 이슈가 있다. 공개키 암호알고리즘에서는 공개키를 상대방에게 비밀스럽게 교환하는 것은 필요하지 않지만 상대방은 공개키를 반드시 인증할 수 있어야 한다. 즉, 주어진 공개키가 누구의 공개키인지 명확하게 검증할 수 있어야 한다. 만약 Bob이 주어진 공개키를 실제로는 Carol 것이지만 Alice것으로 착각하여 중요한 메시지를 이 공개키로 암호화하여 전달하면 Alice는 그 내용을 볼 수 없고, 오히려 Carol만 볼 수 있게 된다.

비대칭 암호알고리즘은 대칭 암호알고리즘과 같은 용도로 사용할 수 있지만 성능 때문에 매우 큰 메시지의 비밀성을 보장하기 위해서는 사용되지 않고, 주로 인증 목적으로 많이 사용된다. 그 이유는 공개키로 메시지를 암호화할 수 있을 뿐만 아니라 개인키로 메시지를 암호화할 수 있기 때문이다. 개인키로 메시지를 암호화하면 누구나 공개키로 복호화할 수 있어 비밀성을 제공할 수 없지만 개인키로 암호화된 메시지는 누가 암호화한 것인지 명확하게 검증할 수 있다. 따라서 개인키로 메시지를 암호화하여 전자서명 역할을 할 수 있다.

1.2.5 대칭 vs. 비대칭 암호알고리즘

대칭 암호알고리즘에서는 두 사용자 간에 비밀키를 공유하는 것이 가장 중요한 이슈이며,

비대칭 암호알고리즘에서는 다른 사용자의 공개키를 인증하는 것이 가장 중요한 이슈이다. 두 종류의 알고리즘 모두 유사한 용도로 사용할 수 있지만 비대칭 암호알고리즘이 대칭 암호알고리즘에 비해 상대적으로 많은 계산 비용이 소요되므로 주로 인증 목적으로 많이 사용된다. 표 1.2는 두 종류의 암호알고리즘을 비교하고 있다.

<표 2.2> 대칭 vs. 비대칭 암호알고리즘

	대칭 암호알고리즘	비대칭 암호알고리즘
키 관계	암호화키 = 복호화키	암호화키 ≠ 복호화키
키 길이	짧다 (보통 128비트)	길다 (RSA의 경우 1024비트)
기본연산	비트 조작 연산	수학 연산 (지수 연산)
성능	상대적으로 우수함	상대적으로 떨어짐
대표 알고리즘	DES, AES	RSA

1.2.6 키 위탁

두 사용자가 안전한 대칭 암호알고리즘을 이용하여 메시지를 암호화하여 교환하면 제3자들은 도청을 하여도 그 내용을 알 수 없다. 만약 범죄 목적으로 메시지를 암호화하여 교환할 경우에는 정부에서는 감청을 할 수 없게 된다. 이 때문에 정부는 비밀 통신을 감청할 능력을 유지하고 싶다. 이를 위해 세 가지 방법이 있다. 첫째, 암호화하여 메시지를 교환하는 것을 금지하는 것이다. 그러나 이 방법은 현실적인 대안이 되지 못한다. 각 사용자들의 행동을 제한하는 것은 한계가 있다. 둘째, 암호알고리즘을 해독할 수 있는 능력을 보유하는 것이다. 이 방법도 현실적인 대안이 되지 못한다. 현재 공개되어 있는 수많은 암호알고리즘은 안전성이 어느 정도 증명되어 있는 것들이다. 따라서 정부가 이들을 해독할 수 있는 능력을 가지는 것은 이와 같은 학문적 증명에 배치되기 때문에 가능한 것이 아니다. 셋째, **키 위탁**(key escrow) 방법을 사용하는 것이다. 키 위탁이란 사용자들이 사용하는 모든 암호키를 강제로 받아 보관하는 것을 말한다. 이 역시 강제로 사용자들로부터 암호키를 받는 것은 쉽지 않다. 강제로 받을 수 있는 방법이 존재하더라도 사용자들이 키 위탁을 신뢰하기 위해서는 정부가 주어진 권한을 남용하지 않을 것을 믿어야 한다. 특히, 오늘날 사용자의 프라이버시는 또 다른 중요한 요구사항이다. 따라서 이를 위해 암호기술에서는 threshold 기법을 사용한다. 이 기법에서는 하나의 정보를 여러 개로 나누어 독립적인 기관이나 사용자에게 분배하며, 이들 중 일부가 동의하면 다시 원 정보를 복원할 수 있다. 따라서 키를 위탁할 때 이것을 여러 개로 나누어 여러 기관이 유지하고 있으며, 특정 기관이 홀로 권한을 남용할 수 없게 된다.

키 위탁은 키를 분실하였을 때 키를 복구하는데 유용하게 사용될 수 있다. **키 복구**(key recovery)를 키 위탁과 혼돈하는 경우가 있는데, 키 복구는 키 위탁을 통해 제공될 수 있는 하나의 서비스이다. 키 복구는 키 위탁 외에 여러 다른 방법을 통해 복구할 수 있다. 예를 들어 키를 여러 곳에 백업하여 문제가 발생하였을 때 복구할 수도 있다. 암호문은 그것을 복호화할 수 있는 키가 없으면 아무런 가치가 없다. 따라서 데이터를 안전하게 암호화하여 파일 서버에 보관하고 있을 때 해당되는 복호화키를 분실하면 이 데이터들은 더 이상 열람

할 수 없게 된다. 따라서 키 복구도 키 관리 측면에서 매우 중요한 요소이다.

1.2.7 전자서명

암호알고리즘은 좁은 의미에서는 암호화와 복호화에 사용되는 알고리즘을 말하지만 넓은 의미에서는 암호기술에서 사용되는 모든 알고리즘을 말한다. 암호기술에서 널리 사용되는 알고리즘에는 앞서 설명한 대칭과 비대칭 암호알고리즘 외에 **전자서명**(digital signature), **해쉬함수**(hash function), **의사난수 비트 생성기**(pseudo-random bit generator) 등이 있다. 이 절에서 전자서명에 대해 알아본다. 전자서명은 기존 서명을 전자적으로 구현한 것으로서 기존 서명이 갖추어야 하는 요구사항뿐만 아니라 전자적으로 구현하였기 때문에 갖추어야 하는 추가 요구사항을 가지고 있다. 따라서 전자서명은 요구사항은 다음과 같다.

- **요구사항 1.** 인증(authentic): 누가 서명하였는지 확인이 가능해야 한다.
- **요구사항 2.** 위조불가(unforgeable): 위조가 불가능해야 한다.
- **요구사항 3.** 재사용불가(not reusable): 서명을 다시 사용할 수 없어야 한다.
- **요구사항 4.** 변경불가(unalterable): 서명된 문서의 내용을 변경할 수 없어야 한다.
- **요구사항 5.** 부인방지(non-repudiation): 나중에 부인할 수 없어야 한다.

요구사항 1을 충족하기 위해서는 각 서명자마다 서명이 독특해야 한다. 요구사항 3에서 다시 사용할 수 없다는 것은 두 가지 측면으로 해석될 수 있다. 첫 번째 측면은 특정 메시지에 대한 서명 블록을 다시 사용할 수 없어야 한다는 것이다. 즉, 특정 메시지의 서명 블록을 다른 메시지의 서명 블록으로 사용할 수 없어야 한다. 여기서 서명 블록이란 메시지와 별도로 존재하는 서명값을 말한다. 보통 전자서명은 일반서명과 달리 서명한 문서 또는 메시지와 별도로 존재한다. 이 요구조건이 충족되기 위해서 전자서명은 메시지에 의존해야 한다. 두 번째 측면은 특정 메시지와 그것의 서명 블록 전체를 다시 사용할 수 없어야 한다는 것이다. 이것을 방어하기 위해서는 서명에 서명 시간을 포함하여 서명을 확인하는 사용자가 이미 확인한 것인지 또는 서명의 유효기간이 지난 것인지 검사해야 한다. 요구사항 4는 요구사항 3의 첫 번째 측면과 동일한 의미이다. 따라서 전자서명이 메시지에 의존하면 요구사항 4를 충족시킬 수 있다.

전자서명은 다음과 같은 측면에서 일반 서명과 다르다. 첫째, 전자서명은 수학적으로 서명자를 검증할 수 있다. 하지만 일반 서명은 보통 원 서명과 눈으로 대조하여 검증하며, 각 서명자의 서명마다 위조의 어려움이 다를 수 있다. 따라서 일반 서명보다 전자서명이 상대적으로 안전하다고 할 수 있다. 둘째, 일반 서명은 문서 위에 하지만 전자서명은 앞서 언급한 바와 같이 문서와 보통 별도로 존재한다. 셋째, 일반 서명은 서명마다 동일한 형태이지만 전자서명은 메시지에 의존해야 하므로 문서마다 다른 형태이어야 한다. 넷째, 전자서명은 원본과 복사본을 구분하기가 어렵다. 따라서 전체 재사용을 방지하기 위해 서명시간을 서명에 포함해야 하며, 검증자는 이미 처리한 서명인지 중복검사를 해야 한다.

전자서명의 요구사항에서 알 수 있듯이 전자서명은 서명자와 메시지에 의존해야 한다. 따라서 전자서명 함수는 메시지와 서명자마다 다른 **서명키**를 입력으로 사용한다. 보통 공개키 암호알고리즘에서 개인키가 서명키의 역할을 하며, 공개키가 **확인키**의 역할을 한다. 전자서명의 구성요소는 서명할 수 있는 메시지의 유한집합 M, 고정된 길이의 서명의 유한집합 S,

서명을 생성하기 위해 사용되는 함수 σ, 서명을 검증하기 위해 사용되는 함수 ν, 가능한 서명키/확인키의 유한집합으로 구성된다. 여기서 서명함수 σ는 서명키 $-K$와 메시지 $m \in M$을 입력받아 전자서명 $s \in S$를 출력하여 주며, 확인함수 ν는 $m \in M$, $s \in S$, 확인키 $+K$를 입력받아 s가 m을 $-K$로 서명한 전자서명이 맞으면 true 아니면 false를 출력하여 준다. 앞서 언급한 위조불가 요구사항을 수학적으로 표현하면 다음과 같다.

- **요구사항 2.** 위조불가(unforgeable): $-K$를 모르는 사람이 $\nu(m,s,+K) = true$인 임의의 $m \in M$과 $s \in S$를 찾는 것은 계산적으로 어려워야 한다.

1.2.8 해쉬함수

해쉬함수는 임의의 길이에 이진 문자열을 고정된 길이의 이진 문자열로 매핑하여 주는 함수를 말하며, 해쉬함수의 결과를 해쉬값, 메시지 다이제스트, 메시지 지문이라 한다. 해쉬함수는 크게 다음과 같은 요구사항을 충족해야 한다.

- **요구사항 1. 압축**: 임의의 길이의 이진 문자열을 일정한 크기의 이진 문자열로 변환해야 한다.
- **요구사항 2. 계산의 용이성**: x가 주어지면 $H(x)$는 계산하기 쉬워야 한다.
- **요구사항 3. 일방향성**(one-wayness): 입력을 모르는 해쉬값 y가 주어졌을 때, $H(x') = y$를 만족하는 x를 찾는 것은 계산적으로 어려워야 한다.
- **요구사항 4. 약한 충돌회피성**(weak collision-resistance): x가 주어졌을 때 $H(x') = H(x)$인 $x'(\neq x)$을 찾는 것은 계산적으로 어려워야 한다.
- **요구사항 5. 강한 충돌회피성**(strong collision-resistance): $H(x') = H(x)$인 서로 다른 임의의 두 입력 x와 x'을 찾는 것은 계산적으로 어려워야 한다.

압축함수는 임의의 메시지를 현재 크기보다 작은 크기로 축소해줄 수 있어야 할 뿐만 아니라 나중에 다시 원래의 크기로 복원할 수 있어야 한다. 하지만 해쉬함수는 압축을 하지만 복원을 할 필요가 없고, 복원이 가능해서도 안 된다. 임의 길이의 메시지를 고정된 길이의 메시지로 변환하므로 함수의 정의역 집합보다 치역 집합이 훨씬 크다. 따라서 서로 다른 메시지가 같은 해쉬값으로 매핑되는 것은 불가피하다. 이처럼 서로 다른 메시지가 같은 해쉬값으로 매핑되면 **충돌**(collision)이 발생하였다고 한다. 안전한 해쉬함수가 되기 위해서는 이와 같은 충돌을 찾는 것이 어려워야 한다. 충돌은 크게 약한 충돌과 강한 충돌로 구분되는데, 약한 충돌은 정해진 어떤 해쉬값으로 매핑되는 또 다른 메시지를 찾는 것이고, 강한 충돌은 같은 해쉬값으로 매핑되는 임의의 메시지 쌍을 찾는 것이다. 따라서 강한 충돌을 찾는 것이 더 쉽다. 그러므로 강한 충돌을 찾는 것이 어려운 해쉬함수가 가장 안전한 해쉬함수이다. 일방향성까지 충족되는 해쉬함수를 일방향 해쉬함수(OWHF, One-Way Hash Function)라 하며, 강한 충돌회피성까지 충족하는 해쉬함수를 충돌회피 해쉬함수(CRHF, Collision-Resistant Hash Function)라 한다.

해쉬함수는 크게 세 가지 용도로 사용된다. 첫째, 전자서명에서 사용된다. 전자서명 암호알고리즘은 보통 공개키 암호알고리즘을 많이 사용한다. 따라서 매우 큰 길이의 메시지 자체를 전자서명하기에는 비용이 많이 소요되므로 전체 메시지 대신에 보통 메시지의 해쉬값

에 서명하는 경우가 많다. 원 메시지 대신에 메시지의 해쉬값을 서명하여 동일한 효과를 얻기 위해서는 기존 전자서명 요구사항이 여전히 충족되어야 한다. 이 때 강한 충돌회피성이 보장되지 않는 해쉬함수를 사용하면 기존 서명 블록을 다른 메시지의 서명 블록으로 활용할 수 있게 된다. 둘째, 메시지의 무결성을 제공하기 위해 사용된다. 셋째, 패스워드를 안전하게 유지하기 위해 사용된다.

해쉬함수는 보통 비밀키를 사용하지 않는다. 따라서 메시지와 그것의 해쉬값을 함께 전송하거나 저장할 경우 그것의 무결성을 안전하게 보장하기가 어렵다. 그 이유는 고의적인 공격자는 메시지 뿐만 아니라 해쉬값도 함께 변경할 수 있기 때문이다. 따라서 보다 안전하게 무결성을 보장하기 위해 해쉬값을 생성할 때 비밀키를 사용하여 해당 비밀키를 모르는 경우에는 올바른 해쉬값을 계산할 수 없도록 하는 기법을 사용한다. 이 처럼 비밀키를 사용하여 계산되는 해쉬값을 **메시지 인증 코드**(MAC, Message Authentication Code)라 하고, 비밀키를 사용하지 않는 일반 해쉬값을 조작 탐지 코드(MDC, Modification/Manipulation Detection Code)라 한다.

1.3 암호해독

1.3.1 암호알고리즘에 대한 공격

좁은 의미의 암호알고리즘에 대한 공격의 목적은 암호문을 생성할 때 사용한 암호키를 발견하거나 암호키를 모르는 상태에서 암호문으로부터 평문을 얻어내는 것이다. 이와 같은 공격을 **암호해독 공격**이라 한다. 암호해독 공격 외에 다른 방법으로 키가 알려지는 것을 노출(compromise)되었다고 한다. 암호기술을 사용하는 암호프로토콜(cryptographic protocol)에 대한 공격은 크게 **수동 공격**(passive attack)과 **능동 공격**(active attack)으로 분류할 수 있다. 수동 공격이란 프로토콜의 진행을 방해하지 않고 공격하는 것을 말하고, 능동 공격이란 프로토콜의 진행에 개입하여 공격하는 것을 말한다. 따라서 암호알고리즘에 대한 공격은 수동 공격의 한 종류로 볼 수 있다.

대칭 암호알고리즘에 대한 가장 쉬운 공격은 **전사공격**(brute-force attack)이다. 전사공격이란 가능한 모든 키를 검사하는 방법을 말하며, 가능한 모든 키를 검사하면 당연히 공격에 성공할 수 있다. 문제는 소요되는 시간이다. 전사공격을 통해 공격에 성공하는 것이 어렵도록 충분히 긴 길이의 암호키를 사용해야 한다. 따라서 매우 큰 길이의 암호키를 사용하는 비대칭 암호알고리즘에 대한 전사공격은 의미가 없다. 또 전사공격은 기계적인 공격이므로 암호해독 공격이라고 하지 않는다.

암호알고리즘에 대한 암호해독 공격은 공격자가 어떤 정보를 가지고 있는지에 따라 크게 **암호문 단독 공격**(ciphertext-only attack), **기지 평문 공격**(known-plaintext attack), **선택 평문 공격**(chosen-plaintext attack), **적응적 선택 평문 공격**(adaptive chosen-plaintext attack), **선택 암호문 공격**(chosen-ciphertext attack)으로 분류할 수 있다. 이와 같은 공격들은 실제 일어날 수 있는 공격이라기보다는 암호알고리즘이 얼마나 안전한지 검사하기 위한 가상의 공격으로 보는 것이 올바르다.

공격자가 가장 적은 정보를 가지고 공격하는 것을 암호문 단독 공격이라 한다. 이 공격에서 공격자는 같은 암호키를 사용하여 같은 알고리즘으로 암호화된 유한 암호문 집합만을 가지고 있다. 따라서 공격자의 목표는 대응되는 평문을 찾거나 사용된 암호키를 찾거나 암호

키를 찾지 못하였지만 암호키 없이 암호문을 복호화할 수 있는 새로운 알고리즘을 발견하는 것이다. 기지 평문 공격에서 공격자는 암호문뿐만 아니라 대응되는 평문도 가지고 있다. 하지만 가지고 있는 암호문/평문 쌍을 공격자가 선택할 수 없다. 이 공격에서 공격자의 목표는 사용된 암호키를 찾거나 암호키를 찾지 못하였지만 암호키 없이 암호문을 복호화할 수 있는 새로운 알고리즘을 발견하는 것이다.

선택 평문 공격에서 공격자는 기지 평문 공격과 마찬가지로 공격자가 평문과 암호문 쌍들의 유한집합을 가지고 있다. 차이점은 선택 평문 공격에서 공격자는 자신이 원하는 평문과 암호문 쌍을 가질 수 있다. 다만, 공격이 시작된 이후에는 새로운 쌍을 얻을 수 없다. 공격 목표는 기지 평문 공격과 동일하다. 적응적 선택 평문 공격에서 공격자는 공격을 하면서 원하는 평문과 암호문 쌍을 계속 얻을 수 있다. 따라서 지금까지 소개한 공격 중 공격자의 능력이 가장 크다. 선택 암호문 공격은 기존 선택 평문 공격과 달리 공격자가 원하는 평문을 선택하면 대응되는 암호문을 얻을 수 있는 것이 아니라 반대로 암호문을 선택하면 그것에 대응되는 평문을 얻을 수 있는 공격을 말한다. 선택 암호문 공격은 주로 공개키 암호알고리즘의 안전성 분석할 때 사용된다.

암호알고리즘에 대한 공격의 결과는 크게 다음과 같은 네 가지로 분류할 수 있다. 첫째, 암호키를 발견하면 완전 성공(total break)을 거두었다고 한다. 둘째, 암호키를 발견하지 못하였지만 복호화할 수 있는 알고리즘을 발견하면 광역 성공(global deduction)을 거두었다고 말한다. 셋째, 어떤 한 암호문으로부터 그것의 평문을 얻어낸 경우 인스턴스 성공(instance deduction)을 거두었다고 한다. 넷째, 암호문으로부터 평문의 일부나 암호키와 관련된 일부 정보를 얻어낸 경우 정보 추출 성공(information deduction)을 거두었다고 한다.

공격의 복잡성을 측정하는 척도로는 데이터 복잡성, 처리 복잡성, 공간 요구사항 등이 사용된다. 데이터 복잡성은 공격이 성공하기 위해 필요한 데이터의 양을 말하며, 처리 복잡성은 공격이 성공하기 위해 필요한 시간을 말하고, 저장공간 요구사항은 공격하기 위해 필요한 메모리 공간을 말한다.

1.3.2 암호알고리즘의 안전성

암호알고리즘의 안전성은 해독하기 위해 요구되는 노력에 의해 측정된다. 보통 대칭 암호알고리즘 같은 경우에는 전사공격을 통해 암호해독이 가능하다. 즉, 현재 실제 사용되는 암호알고리즘들은 무조건적으로 안전하지 않다. 무조건적으로 안전하다는 것은 무한한 컴퓨팅 자원을 가져도 공격에 성공할 수 없다는 것을 말한다. 전사공격을 통해 해독이 가능하더라도 소요되는 시간이나 컴퓨팅 능력이 현실성이 없어 실제 사용하는데 아무런 문제가 없다. 이처럼 어떤 암호알고리즘을 지금까지 알려진 가장 우수한 공격방법으로 공격하여 성공하는데 소요되는 시간이 불합리하게 많은 컴퓨팅 능력이나 시간을 요구할 경우 이 알고리즘은 계산적으로 안전(computationally secure)하다고 말한다. 공개키 암호알고리즘도 대칭 암호알고리즘과 마찬가지로 전사공격 측면에서는 계산적으로 안전하다. 하지만 이들은 보통 어렵다고 알려진 어떤 수학문제에 의존하므로 전사공격을 하기보다는 이 수학문제를 푸는 것이 빠르다. 그러나 공개키 암호알고리즘에 사용되는 수학문제를 푸는 것 자체가 계산적으로 어렵다고 알려져 있다. 다시 말하면 의존하는 수학문제를 푸는 것이 불가능하다고 증명되어 있지는 않지만 지금까지 알려진 어떤 방법을 사용하더라도 불합리하게 많은 시간과 비용이 요구된다는 것이다. 이처럼 알고리즘의 안전성이 어렵다고 알려진 다른 문제와 등가일 경우

에는 이 알고리즘은 증명 가능한 안전성(provably secure)을 가지고 있다고 한다.

지금까지는 공격을 이용하여 해독할 수 있는지에 따른 암호알고리즘의 안전성 분류에 대해 살펴보았다. 암호알고리즘 이와 같은 기준 외에 안전성을 논하기 위해 사용하는 몇 가지 개념이 있다. 어떤 알고리즘이 의미론적으로 안전(semantically secure)하다는 것은 암호문이 주어졌을 때 효율적으로 계산할 수 있는 모든 것은 암호문이 없어도 계산할 수 있다는 것을 말한다. 즉, 암호문이 주어도 공격자는 추가적으로 얻을 수 있는 정보가 아무것도 없어야 한다는 것을 말한다. 어떤 알고리즘이 구별불가 안전성을 가지고 있다는 것은 두 개의 메시지 중 하나를 암호화한 암호문이 주어졌을 때, 공격자가 어떤 평문을 암호화한 것인지 맞출 수 있는 확률이 50%보다 높지 않다는 것이다. 어떤 알고리즘이 NM(Non-Malleability) 특성을 제공한다는 것은 암호문에 대응되는 평문을 알지 못하는 경우 이 암호문을 의미 있는 다른 암호문을 변경하는 것이 가능하지 않다는 것을 말한다.

참고문헌

[1] Jianying Zhou and Dieter Gollmann, “Evidence and Non-repudiation," Journal of Network and Computer Applications, Vol. 20, No. 3, pp. 267−281, Jul. 1997.

연습문제

1. 단대단 암호방식과 홉-바이-홉 암호방식을 비교 설명하시오.
2. 대칭 암호알고리즘을 사용하기 위해 원거리 사용자간에 먼저 선행되어야 하는 것과 비대칭 암호알고리즘을 사용하기 위해 원거리 사용자간에 먼저 선행되어야 하는 것을 설명하시오.
3. 전자서명과 일반서명을 비교 설명하시오.
4. 약한 충돌회피성과 강한 충돌회피성을 예를 들어 설명하시오.