정보보호개론

제2장 암호알고리즘 개요 1부

1. 암호알고리즘

과거 전통적으로 암호기술은 비밀성 유지에 초점을 두었기 때문에 암호알고리즘(cryptographic algorithm)은 원래 암호화와 복호화 과정에서 사용하는 수학 함수를 말하였다. 하지만 오늘날에는 암호기술이 비밀성 보장 목적으로만 사용하는 것이 아니므로 **암호기술로 사용하는 모든 알고리즘**을 **암호알고리즘**이라 한다. 이 장에서는 좁은 의미의 암호알고리즘 위주로 살펴보고, 다음 장에서 넓은 의미에서 해시함수, MAC(Message Authentication Code), 전자서명 알고리즘 등에 대해 살펴본다.

현대 암호화 함수와 복호화 함수는 모두 키(key)를 사용하며, 암호화 과정에서 사용하는 키를 암호키(cryptographic key)라 한다. 가능한 평문의 집합 $\mathcal{M}$, 가능한 암호문의 집합 $\mathcal{C}$, 가능한 암호키의 집합 $\mathcal{K}$, 3개의 집합이 있을 때 암호화 함수는 $M \in \mathcal{M}$, $K \in \mathcal{K}$를 입력받아 $C \in \mathcal{C}$를 출력하는 함수이고, 복호화 함수는 $C \in \mathcal{C}$, $K \in \mathcal{K}$를 입력받아 $M \in \mathcal{M}$을 출력하는 함수이다. 이 교재에서는 다음 표기법을 사용하고자 한다.

$$E.K(M) = C, D.K(C) = M$$

암호프로토콜을 설명할 때에는 $E.K(M)$ 대신에 $\{M\}.K$를 사용한다. 암호화 함수와 복호화 함수는 반드시 다음 정확성(correctness) 요구사항을 기본적으로 충족해야 한다.

$$D.K(E.K(M)) = M$$

위 수식들에서는 암호화와 복호화에 사용하는 암호키가 동일한 경우이고, 암호화와 복호화에 사용하는 암호키가 다를 수 있다.

암호알고리즘은 완전성뿐만 아니라 안전하여야 한다. 더욱이 현대 암호알고리즘의 안전성은 알고리즘의 비밀성(security by obscurity)에 의존하는 것이 아니라 암호키에 의존해야 한다. 이 원리는 1883년 케르크호프스(Kerckhoff)가 제시한 원리이다. **케르크호프스의 원리**는 다른 말로 개방 설계(open design) 또는 Shannon의 격언(maxim)이라고도 한다. 암호알고리즘의 안전성이 키에 의존하는 것이 아니라 알고리즘 비밀성에 의존하면 이 알고리즘을 제한적 알고리즘(restricted algorithm)이라 한다.

제한적 알고리즘은 알고리즘 자체가 비밀이기 때문에 해독하기가 상대적으로 어려울 수 있지만, 오늘날에는 알고리즘이 하드웨어로 제작되거나 소프트웨어로 구현되어 사용되므로 역공학(reverse engineering) 기술 때문에 알고리즘을 비밀로 유지하기 어렵다. 더욱이 제한적 알고리즘은 알고리즘의 내부 구조가 노출되면 알고리즘 자체를 변경해야 할 뿐만 아니라 알고리즘을 공유할 수가 없다. 비밀 통신을 하고자 하는 쌍의 수가 n이면 n개의 서로 다른 알고리즘이 있어야 한다. 이와 달리 키 의존 알고리즘은 n 쌍이 모두 같은 알고리즘을 사용할 수 있다. 다만, 이때 n 쌍은 모두 다른 키를 사용해야 안전하다. 키 의존 알고리즘은 키가 노출되었을 때 알고리즘을 바꾸는 것이 아니라 키만 변경하면 된다. 물론 알고리즘의 내부 내용을 상세하게 공개하기 때문에 알고리즘에 존재하는 허점을 발견하기가 상대적으로 쉽다. 하지만 이것은 부정적인 측면만 있는 것이 아니라 빨리 보완될 수 있다는 긍정적인 측면도

있다. 암호해독기술자들은 알고리즘을 분석하여 존재하는 허점을 발견하여 이를 보완하도록 하는 것이 그들의 주된 연구 목표이다.

암호키는 사용하는 방식, 사용하는 횟수 등에 따라 다음과 같이 여러 가지 용어로 불리워진다.

- 암호키(cryptographic key): 암호알고리즘에서 사용되는 모든 종류의 키
- 암호화키(encryption key): 암호화할 때 사용되는 키
- 복호화키(decryption key): 복호화할 때 사용되는 키
- 비밀키(secret key): 암호화키와 복호화키가 같은 대칭 암호알고리즘(symmetric cryptographic algorithm)에 사용되는 키
- 개인키(private key), 공개키(public key): 암호화키와 복호화키가 다른 비대칭 암호알고리즘(asymmetric cryptographic algorithm)에서 사용하는 키를 말한다. 보통 암호화할 때 사용하는 키가 공개키이고, 복호화할 때 사용하는 키가 개인키이다. 더 자세한 것은 3절에서 설명한다.
- 장기간키(long-term key), 세션키(session key): 사용자가 장기간 동안 유지하고 사용하는 키를 장기간 키라 하며, 단일 통신 세션을 위해 생성하여 사용한 후 버리는 키를 세션키라 한다.

이 외에도 다중사용 키(many-time use key)와 일회용 키(one-time use key) 개념도 있다. 장기간 키처럼 비휘발성 메모리에 유지해야 하면 이 키를 안전하게 보호하기 위한 별도 보안 메커니즘이 필요하다. 이 측면에서 장기간 키의 사용은 최소화하고 필요할 때마다 새 일회용 키를 이용하는 것이 바람직하다. 또 키는 각 용도마다 다른 키를 사용하는 것이 바람직하다.

1.1 암호알고리즘의 분류

일반 알고리즘과 마찬가지로 암호알고리즘도 결정적(deterministic) 알고리즘과 확률적(probabilistic) 알고리즘으로 분류할 수 있다. 결정적 알고리즘은 동일 입력을 이용하면 항상 동일한 과정을 거쳐 같은 결과를 주는 알고리즘이고, 확률적 알고리즘은 동일 입력을 이용하더라도 내부적으로 사용하는 랜덤 요소 때문에 계산 과정이 달라질 수 있는 알고리즘이다. 확률적 알고리즘은 다른 말로 무작위(randomized) 알고리즘이라 한다.

암호알고리즘에서는 같은 입력에 대해 항상 같은 출력을 주는지 다른 출력을 줄 수 있는지 특성이 안전성에 영향을 줄 수 있다. 결정적 알고리즘이면 같은 입력에 대해 다른 출력을 줄 수 없지만, 확률적 알고리즘은 다른 출력을 줄 수 있다. 따라서 암호화 알고리즘은 정보 노출을 줄이기 위해 확률적 알고리즘이어야 바람직하다. 암호화 알고리즘이 결정적 암호알고리즘이면 어떤 암호문에 대한 평문이 노출되었을 때 해당 암호문을 다시 사용하면 공격자는 그것의 대응되는 평문을 알게 된다. 하지만 암호화와 복호화 함수의 완전성 특성 때문에 복호화 함수는 반드시 결정적 알고리즘이어야 한다. 결정적 암호알고리즘도 언제든지 입력에 랜덤 요소를 추가하여 확률적 암호알고리즘으로 변경할 수 있지만, 평문 앞에 랜덤 값을 붙이는 등 단순한 방법으로 변경할 수 있는 것은 아니다. 그와 같은 추가가 안전성에 나쁜 영향을 주지 않아야 한다.

암호화와 복호화 함수를 말하는 좁은 의미의 암호알고리즘은 암호화와 복호화할 때 사용하는 암호키가 같은지 다른지에 따라 대칭, 비대칭 암호알고리즘으로 분류한다. 비대칭 암호알고리즘의 경우 비대칭보다는 공개키 암호알고리즘 용어를 더 많이 사용한다. 두 사용자 간의 메시지를 비밀스럽게 교환하고자 할 때 대칭 암호알고리즘은 두 사용자가 같은 키를 가지고 있어야 하지만 공개키 암호알고리즘은 전송자가 수신자의 공개키를 이용하여 메시지를 암호화하면 수신자는 자신의 개인키로 복호화하는 방식이다. 따라서 대칭 암호알고리즘에서는 대칭키의 비밀성이 중요하지만 공개키 암호알고리즘에서 공개키는 비밀로 유지할 필요가 없다. 하지만 공개키 방식에서는 공개키의 인증이 매우 중요하다. 즉, 사용하는 공개키가 누구의 공개키인지 확인할 수 있어야 한다.

2. 대칭 암호알고리즘

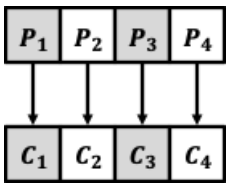
고대 로마 시대는 물론 그 이전에도 군사적 목적으로 이 사용되었다. 따라서 대칭 암호알고리즘을 다른 말로 전통 암호알고리즘(conventional cryptographic algorithm)이라 한다. 이 알고리즘에서 사용하는 키를 비밀키라 하며, 이 때문에 대칭 암호알고리즘을 다른 말로 비밀키 암호알고리즘이라고도 한다. 대칭 암호알고리즘에서는 암호화할 때와 복호화할 때 같은 키를 사용하기 때문에 **원격에 있는 두 사용자가 대칭 암호알고리즘을 이용하기 위해서는 먼저 안전하게 비밀키를 공유**하여야 한다. 대칭 암호알고리즘에서 비밀키는 보통 암호학적으로 안전한 의사난수 생성 알고리즘을 이용하여 생성한다.

대칭 암호알고리즘은 크게 한 바이트씩 암호화하는 스트림(stream) 방식과 일정한 크기만 암호화할 수 있는 블록(block) 방식으로 분류할 수 있다. 스트림 방식은 보통 평문의 한 바이트와 한 바이트의 키 스트림 값을 XOR하여 암호화한다. 이를 위해 주어진 키로부터 키 스트림을 생성하는 알고리즘이 필요하며, 이 알고리즘은 확률적 알고리즘이어야 한다. 현재 널리 사용하고 있는 스트림 방식의 대칭 암호알고리즘은 Salsa20, ChaCha20 등이 있으며, 블록 방식의 대칭 암호알고리즘은 미국 표준인 AES(Advanced Encryption Standard)가 있다.

블록 방식의 대칭 암호알고리즘은 치환(substitution)과 자리바꿈(transposition) 두 가지 연산을 기본 연산으로 사용한다. 치환은 주어진 데이터를 다른 데이터로 바꾸는 것이고, 자리바꿈은 데이터의 위치를 바꾸는 것이다. 치환과 자리바꿈을 보통 둘 다 사용하게 되는데, 이와 같은 방식을 합성 암호(product cipher)라 하며, 합성 암호를 여러 번 해야 원하는 암호 강도를 얻을 수 있어서 정해진 과정을 여러 차례 반복하게 된다. 이때 정해진 과정을 한 번 수행하는 것을 라운드라 한다. 현대 대칭 암호알고리즘은 비트 연산 기반 치환과 자리바꿈 연산을 사용하고 있어 매우 큰 정수 연산을 기반으로 하고 있는 공개키 암호알고리즘에 비해 상대적으로 성능이 우수하다.

2.1 암호화 모드

블록 방식의 대칭 암호알고리즘에서 암호화와 복호화 함수는 항상 입력의 크기가 고정되어 있다. 이 크기를 블록 크기라 하며, 암호화하고자 하는 데이터의 크기가 다양하기 때문에 다양한 크기의 데이터를 암호화하는 방식이 추가로 필요하다. 이를 **암호화 모드**(cryptographic mode)라 한다. 암호화할 평문의 크기가 블록 크기보다 작으면 평문의 크기를 블록 크기로 만들기 위해 **채우기**(padding)를 하게 된다. 채우기는 평문의 크기가 블록 크기보다 작을 때만 사용하는 것이 아니라 클 때에도 필요하다. 평문의 크기가 정확하게 블록 크기의 배수가 아니면 마지막 블록은 채우기가 필요할 수 있다.



<그림 2.1> ECB 모드의 문제점: P_1과 P_3가 같으면 C_1이 C_3와 같아짐

암호화 모드 중 가장 단순한 모드가 ECB(Electronic CodeBook) 모드이다. 이 모드에서는 평문을 블록 크기로 나누어 각 블록을 독립적으로 암호화한다. 이 경우 평문의 크기가 정확하게 블록 크기의 배수가 아닐 수 있기 때문에 마지막 블록은 채우기가 보통 필요하다. 결정적 대칭 암호알고리즘을 사용하여 평문을 ECB 모드로 암호화할 때 평문의 블록 중 값이 같은 블록이 있으면 대응되는 암호문 블록의 값도 같게 된다. 예를 들어 그림 2.1처럼 평문이 4개의 평문 블록으로 구성되어 있고, P_1과 P_3가 같을 때 ECB 모드로 암호화하게 되면 암호문 블록 C_1과 C_3가 같아진다. 이것은 안전성 측면에서 정보가 노출되는 문제점을 갖게 된다. 따라서 ECB 모드를 이용하여 일반 데이터를 암호화하는 경우는 거의 없으며, 이와 같은 문제점이 없는 암호화 모드를 주로 사용한다. 암호화 모드는 8장에서 자세히 설명한다.

2.2 사용 용도

대칭 암호알고리즘은 원격에 있는 두 사용자 간의 공개 채널로 교환하는 데이터의 비밀성을 보장하기 위해 주로 사용한다. 이때 송신자와 수신자는 사전에 안전하게 비밀키를 공유하고 있어야 한다. 사용자가 자신의 데이터를 파일시스템에 저장할 때에도 비밀성을 보장하기 위해 사용할 수 있다. 이 경우 원격에 있는 두 사용자 간의 데이터를 비밀스럽게 교환할 때와 달리 사용할 비밀키를 다른 사용자와 공유할 필요는 없다. 하지만 만약 저장된 암호화된 데이터를 복호화할 수 있는 비밀키를 분실하게 되면 해당 데이터를 다시 얻을 수 없게 된다. 이것에 착안하여 만들어진 악성 소프트웨어(malware)가 랜섬웨어(ransomware)이다. 랜섬웨어는 목표 컴퓨터에 불법적으로 접근하여 파일시스템에 있는 파일들을 암호화하고 돈을 지불하지 않으면 복호화를 해주지 않는 악성 소프트웨어이다.

대칭 암호알고리즘은 개체 인증을 위해 사용할 수 있다. 여기서 개체 인증이란 통신 상대방이 누구인지 확인하는 것을 말한다. 암호기술을 사용한 개체 인증은 상대방만이 할 수 있는 행위의 확인을 통해 이루어진다. 대칭 암호알고리즘을 이용한 통신은 두 사용자가 같은 키를 가지고 있기 때문에 이 키로 암호화된 메시지를 보았을 때 본인이 한 것이 아니면 상대방이 한 것으로 믿을 수 있으며, 이를 통해 개체 인증을 할 수 있다.

예를 들어 A와 B가 K_{AB}라는 비밀키를 공유하고 있다고 가정하자. 이때 다음과 같이 메시지를 주고 받는다고 가정하자.

$$\begin{aligned} &\text{Msg 1.} \quad B \to A: \quad N_B \\ &\text{Msg 2.} \quad A \to B: \quad \{N_B\}.K_{AB} \end{aligned}$$

여기서 N_B는 B가 생성한 랜덤 값으로 이전에 사용한 적이 없는 값이어야 한다. 이와 같은 값을 암호기술에서는 **난스**(nonce, number used just once)라 한다. $\{N_B\}.K_{AB}$는 N_B를 K_{AB}로 암호화한 암호문이다. B는 암호문을 수신하면 이것을 복호화하여 복호화된 값이 N_B와 같다면 이 암호문을 메시지 1에 대한 응답으로 A가 보냈다고 믿을 수 있다. 그 이유는 K_{AB}를 가지고 있는 것은 B와 A뿐이므로 B 스스로가 해당 암호문을 만든 적이 없다면 이 암호문은 A가 만들었을 수밖에 없으며, N_B는 이번에 처음 사용한 값이므로 메시지 1에 대한 응답임을 확신할 수 있다.

2.3 키 위탁

암호알고리즘을 이용하여 비밀성을 보장한 상태로 통신하면 암호알고리즘의 특성 때문에 정부를 포함한 어느 누구도 교환된 데이터의 내용을 얻기 힘들다. 이 때문에 이것이 범죄 등에 악용될 수 있다. 따라서 정부는 모든 통신을 필요하면 감청할 수 있기를 원한다. 이와 관련하여 상용적으로 암호 기술이 도입된 초기에 등장한 개념이 키 위탁(key escrow)이다. 키 위탁이란 사용자가 사용하는 암호키를 정부에 주는 것을 말한다. 이렇게 되면 정부가 해당 권한을 남용할 수 있기 때문에 키 위탁이라는 것이 현실적으로 이루어지기는 힘들다. 물론 1장에서 언급한 임계기반 비밀 공유 방식을 이용하여 여러 기관에 나누어 유지할 수 있더라도 자발적인 키 위탁이나 특정 하드웨어만 사용하여 암호 통신하도록 것은 여전히 현실적이지 못하다. 참고로 키 위탁과 키 복구(key recovery)는 다른 개념이다. 키 위탁을 통해 키 복구를 할 수 있지만, 키 복구는 키 백업과 같은 다른 방법을 통해서도 가능하다. 물론 파일시스템 암호처럼 키 복구 기능이 매우 중요한 요소가 될 수 있다.

3. 공개키 암호알고리즘

공개키 암호알고리즘은 1976년에 Diffie와 Hellman이 처음 개념을 소개하였다. 학술 문헌에 처음 등장한 것이 이때이지만 영국 정부에서 70년대 초에 공개키 기술을 발견한 사실이 영국 정부 비밀문서가 최근 공개되면서 밝혀졌다. 대칭 암호알고리즘은 고대 로마 시대 이전부터 사용된 것과 비교하면 상대적으로 역사가 짧다. 공개키 암호알고리즘에서는 암호화할 때와 복호화할 때 다른 암호키를 사용한다. 따라서 각 사용자는 두 개의 키를 보유하게 된다. 하나는 자신만이 비밀스럽게 사용하는 개인키이고, 다른 하나는 누구에게나 공개할 수 있는 공개키이다. 현재 가장

널리 사용하는 공개키 암호알고리즘은 RSA와 ElGamal 알고리즘이다. 최근에는 이들보다 키 길이가 상대적으로 짧은 타원곡선 기반 공개키 암호알고리즘을 많이 사용하고 있다.

공개키 암호알고리즘에서 키는 보통 개인키를 안전한 의사난수 생성 알고리즘을 이용하여 생성한 후에 이를 이용하여 공개키를 생성한다. 거꾸로 공개키를 먼저 생성한 후에 개인키를 생성하는 경우도 있다. 이 경우에는 키 쌍 중에 어느 하나를 가지고 있어도 특정 비밀 정보를 모르면 대응되는 다른 것을 생성할 수 없다. 공개키 암호알고리즘의 자세한 내부 동작 원리는 9장에서 설명한다.

보통 한 사용자는 다른 사용자의 공개키를 이용하여 메시지를 암호화하여 전달하면 받은 사용자는 자신의 개인키를 이용하여 암호문을 복호화하게 된다. 따라서 대칭 암호알고리즘처럼 원격에 있는 두 사용자가 같은 키를 공유하고 있을 필요가 없다. 하지만 **사용자가 다른 사용자의 공개키를 사용할 때 해당 공개키가 누구의 공개키인지 확인**할 수 있어야 한다. 이것을 **공개키 인증**이라 한다. 예를 들어 A가 $+K_C$를 C의 공개키이지만 B의 공개키로 착각하고 있다고 하자. 그러면 A가 B에게 메시지 M을 비밀스럽게 보내고 싶으면 $+K_C$로 M을 암호화하여 보내게 된다. 이 경우 B는 해당 메시지를 받아도 M을 얻을 수 없으며, 대신 C는 메시지 M을 알게 되는 문제점이 발생한다. 이전 인터넷 뱅킹을 할 때 사용한 **인증서**(certificate)가 공개키를 인증하는 표준 방법이다.

3.1 사용 용도

공개키 암호알고리즘에서는 보통 상대방의 공개키로 메시지를 암호화하고 이것을 받은 사용자가 자신의 개인키로 암호문을 복호화하여 메시지를 얻게 된다. 하지만 공개키 암호알고리즘 중에는 같은 암고리즘을 이용하여 개인키로 메시지를 암호화할 수 있는 것도 있다. 대표적인 것이 RSA 알고리즘이다. 메시지를 개인키로 암호화하면 누구나 공개키를 이용하여 복호화할 수 있다. 따라서 이와 같은 방식으로는 비밀성을 제공할 수 없다. 하지만 개인키는 오직 해당 사용자만 가지고 있는 것이기 때문에 개인키로 어떤 메시지가 암호화되어 있으면 이것은 그 사용자가 암호화했다는 증거가 된다. 따라서 개체 인증을 할 때 이와 같은 방식을 많이 사용하며, 전자서명으로도 사용이 가능하다.

공개키 암호알고리즘은 이처럼 대칭 암호알고리즘과 마찬가지로 비밀성과 인증 서비스를 위해 사용할 수 있다. 하지만 공개키 암호알고리즘의 성능이 대칭 암호알고리즘에 비해 상대적으로 낮기 때문에 비밀성을 목적으로 일반적인 데이터를 공개키로 암호화하는 경우는 드물다. 대신에 다음과 같이 하이브리드 방식을 사용하여 메시지의 비밀성을 보장하는 경우가 많다.

$$\{M\}.K, \{K\}.+K_A$$

여기서 K는 랜덤하게 새롭게 생성한 일회용 대칭키이다. 이 메시지는 오직 A만 복호화할 수 있다. A는 먼저 자신의 개인키를 이용하여 두 번째 암호문을 복호화한 다음 얻은 대칭키 K를 이용하여 첫 번째 암호문을 복호화여 메시지를 얻을 수 있다.

4. 암호알고리즘 마일스톤

대칭 암호알고리즘과 공개키 암호알고리즘 관련 주요 마일스톤을 나누어 살펴보자. 먼저 현대 대칭 암호알고리즘 관련하여 주요 마일스톤은 다음과 같다.

- 1949년: 암호알고리즘 안전성 관련하여 매우 중요한 이론인 완벽 안전성이라는 개념을 C. Shannon이 정립하였다.
- 1977년: DES(키 길이 56비트, 블록 길이 64비트)가 미국 대칭 암호알고리즘 표준으로 채택되었다.
- 2001년: AES(키 길이 128비트, 블록 길이 128비트)가 미국 대칭 암호알고리즘 표준으로 채택되었다.

- 2004년: 유럽에서 eStream 프로젝트가 진행되어 여러 차세대 스트림 암호알고리즘이 개발되었다.
- 2009년: 6개 인증 암호화 모드가 미국 표준으로 채택되었다.

현대 공개키 암호알고리즘 관련하여 주요 마일스톤은 다음과 같다.

- 1976년: W. Diffie와 M. Hellman이 처음으로 학술 문헌에 공개키 개념을 제시하였다.
- 1978년: R. Rivest, A. Shamir, L. Aldeman이 인수분해 기반 RSA 공개키 암호알고리즘을 제안하였다.
- 1984년: ElGamal가 이산대수 기반 ElGamal 공개키 암호알고리즘을 제안하였다.
- 1985년: Shamir가 신원기반 공개키 암호시스템을 제안하였다[1].
- 1985년: N. Koblitz와 V. Miller[2]가 각각 타원곡선 기반 공개키 암호알고리즘을 제안하였다.
- 1993년: 겹선형 사항(bilinear pairing)이라는 개념이 암호기술에 처음으로 사용되었다[3].
- 2001년: Boneh와 Franklin이 겹선형 사상 기반 신원기반 암호시스템을 제안하였다[4].
- 2008년: S. Nakamoto가 비트코인을 제안하였다.

양자 컴퓨팅 관련하여 1995년 P. Shor는 양자 컴퓨팅 알고리즘을 이용하면 인수분해와 이산대수 문제를 다차시간에 해결할 수 있다는 것을 보였고, 1996년 L. Glover는 양자 컴퓨팅 알고리즘을 이용하면 $O(n)$이 요구하는 선형검색을 $O(\sqrt{n})$에 할 수 있다는 것을 보였다. 이 때문에 양자 컴퓨팅이 실제 현실화되면 현재 암호기술이 큰 영향을 주게 된다. 이와 관련한 것은 3장에서 설명한다.

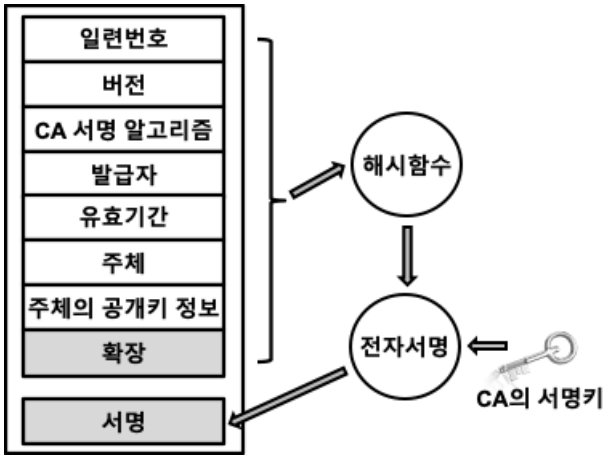
5. 공개키 기반구조

공개키 암호알고리즘에서 가장 중요한 것은 공개키의 인증이다. 이를 위해 가장 널리 사용하는 방법이 인증서이며, 이와 같은 인증서를 사용하기 위해서는 발급, 폐지 등 인증서와 관련된 여러 기능이 필요하다. 인증서를 사용할 수 있도록 해주는 각종 서비스, 정책, 법률 등을 포함하여 관련 모든 요소를 공개키 기반구조(public key infrastructure)라 한다. 일반적으로 우리가 사용하고 있는 공개키 기반구조는 중앙 신뢰 서버를 사용하는 중앙 집중 방식이다.

5.1 인증서

인증서는 1977년 Kohnfelder라는 당시 석사 학생이 제안하였다. 인증서는 공개키와 그것의 주인을 바인딩하여 주는 전자문서로 그림 2.2과 같이 공개키, 공개키 소유자, 발급기관, 유효기간, 사용 용도 등 여러 정보를 신뢰할 수 있는 기관이 전자서명하여 발급하게 된다. 이 기관을 **인증기관**(CA, Certification Authority)이라 한다. 현재 인증서는 국제 표준인 X.509 버전 3에 따라 구성된다. 전자서명 값은 서명한 데이터와 별도로 존재하며, 전자서명의 효율성을 높이기 위해 데이터에 직접 서명하지 않고 그것의 해시값을 구한 후에 그 값에 서명하게 된다. 이에 대해서는 3장에서 자세히 설명한다.

보통 인증서를 발급받으면 인증서를 비휘발성 메모리에 유지한다. 처음에는 하드 디스크에 유지하였지만 여러 기기에서 사용하는 것이 불편하고 PC에 불법 침입한 공격자가 훔칠 수 있는 문제가 있어 USB 메모리에 저장하는 방식으로 바뀌었다. 지금은 USB보다 자신의 스마트폰에 많이 유지하여 사용한다. 공개키 암호알고리즘에서 각 사용자는 2개의 키를 유지해야 한다. 공개키는 인증서 형태로 유지하며, 개인키는 패스워드 기반 대칭 암호알고리즘을 이용하여 암호화된 상태로 유지한다. 패스워드 기반 대칭 암호알고리즘이란 패스워드로부터 대칭키를 생성하여 사용하는 방식을 말한다. 이 때문에 인증서를 사용할 때 패스워드를 요구하는 것이다.



<그림 2.2> 인증서

다른 사용자의 인증서를 사용하기 전에는 해당 인증서가 유효한 것인지 검증하여야 한다. 처음 접하는 인증서의 검증은 다음과 같은 총 4가지 단계로 이루어진다.

- 단계 1. 인증서의 서명 값 확인
- 단계 2. 인증서의 유효기간 확인
- 단계 3. 인증서의 사용 용도 확인
- 단계 4. 인증서의 폐지 여부 확인

위 4가지 단계를 매번 해야 하는 것은 아니다. 보통 자주 사용하는 다른 사용자의 인증서는 사용의 효율성을 위해 보관할 수 있으며, 동일한 용도로 인증서를 다시 사용할 때에는 단계 2와 단계 4만 확인하면 된다.

인증서의 서명 값 확인은 이 인증서를 발급한 발급기관의 인증서를 확보하여 해당 인증서에 있는 공개키를 이용한다. 발급기관의 인증서 역시 확인한 적이 없으면 동일한 4가지 단계를 통해 확인해야 하며, 이것은 최상위 인증기관의 인증서까지 반복적으로 확인해야 할 수 있다. 하나의 인증기관이 전 세계의 모든 인증서를 발급할 수는 없다. 실제로는 다양한 인증기관이 어떤 신뢰 모델을 바탕으로 상하 관계나 동등 관계를 형성하게 된다. 인증기관들이 상하 관계로만 구성되어 있으면 트리 형태로 표시할 수 있으며, 이 트리에 루트에 있는 기관을 최상위 인증기관이라 한다. 상하 관계의 경우 부모 인증기관이 자식 인증기관의 인증서를 발급하게 되며, 안전성 측면에서 비논리적이지만 최상위 인증기관은 자체 서명한(self-signed) 인증서를 사용하게 된다. 동등관계에 있는 인증기관들은 인증서 빠르게 확인할 수 있도록 서로 보증하는 상호 인증서(cross certification)를 발급하여 사용한다. 트리의 단말에 있는 인증기관들이 사용자의 인증서를 발급하게 된다.

국내의 경우 2020년까지는 공인 인증서라는 개념을 사용하였다. 2020년 5월 법이 개정되기 전까지는 국가가 승인한 국가기관, 지방자치단체, 비영리법인만 인증기관 역할을 할 수 있었고, 이 기관이 발급한 인증서를 공인 인증서라 하였다. 미국과 같은 경우에는 공인 인증서를 사용하지 않고 verisign과 같은 일반 사기업이 인증 기관 역할을 하였다. 이와 같은 일반 사기업이 발행한 인증서를 공인 인증서와 구분하기 위해 사설 인증서라 하였다. 하지만 국내의 경우 2020년부터는 법으로 공인 인증서라는 표현을 사용할 수 없도록 하였고, 네이버, 카카오 같은 일반 기업도 인증기관 역할을 할 수 있게 되었다. 법 개정 이후 국가가 승인한 기관이 발급한 인증서를 공동인증서라 한다.

국내와 국외의 이와 같은 차이는 여러 요인이 있지만 인증서를 주로 활용한 분야가 다른 것도 주요 원인이다. 우리 나라는 주로 인터넷 뱅킹 등 사용자를 인증하기 위한 용도로 활용하였지만 국외에서는 개인 인증 용도보다는 웹에서 서버를 인증하기 위한 용도로 주로 사용 및 확산되었다.

인증서는 특정한 용도로만 사용할 수 있도록 제한할 수 있다. 예를 들어 인증서는 인터넷 뱅킹을 할 때도 사용할 수 있고, 주민등록등본와 같은 증명서를 발급받을 때 신원을 증명하기 위해 사용할 수 있으며, 인터넷 쇼핑에서 고액을 지불할 때 안전성을 높이기 위해 사용할 수 있다. 범용 인증서는 모든 용도로 사용할 수 있지만 금융 전용 인증서는 인터넷 뱅킹을 할 때에만 사용할 수 있다. 사용 용도는 범용, 전용으로 국한하지 않고 다양하게 인증서의 사용 용도를 제한하기 위해 사용할 수 있다.

우리가 거래할 때 널리 사용하는 신용카드는 유효기간이 아직 남아 있지만, 재발급 받거나 분실, 도난을 당할 수 있다. 이 경우 우리는 기존 신용카드를 더는 사용할 수 없도록 해야 한다. 인증서도 마찬가지이다. 유효기간이 남아 있는 인증서도 그것을 보관한 USB나 다른 저장장치에 문제가 발생하여 다시는 사용할 수 없게 될 수 있다. 이 경우 인증서를 폐지하고 새 인증서를 발급받아야 한다. 불법적으로 획득한 다른 사용자의 폐지된 인증서(인증서를 가지고 있다고 사용할 수 있는 것은 아니다. 이 인증서에 포함된 공개키에 대응되는 개인키를 확보해야 사용할 수 있음)를 사용하지 못하도록 공개키 기반구조에서는 **인증서 폐지 목록**(**CRL**, Certificate Revocation List)을 유지한다. 이 목록은 무결성이 유지되어야 하므로 주기적으로 인증기관이 서명하여 발급한다. 이 목록에는 유효기간이 남아 있지만 폐지된 인증서의 일련번호와 폐지 사유 등 각 폐지된 인증서마다 최소한의 정보만 유지하며, 유효기간이 지나면 이 목록에서 자동적으로 빠지게 된다. 따라서 CRL의 크기는 무한정 커지는 형태는 아니다. 사용자들은 인증서를 검증할 때 보통 최신 CRL을 다운받아 검증하는 인증서가 이 목록에 포함되어 있는지 확인하게 된다.

CRL을 이용하는 것 외에도 OCSP(Online Certificate Status Protocol)을 이용하여 인증서의 폐지 여부를 온라인으로 문의하여 확인할 수 있다[5]. 보통 CRL은 주기적으로 갱신되기 때문에 OCSP가 더 최신 정보를 제공할 수 있다. 최근에는 인증서를 확인하는 측이 OCSP 서버에 문의하는 형태가 아니라 인증서를 제공할 때 OCSP 확인서를 첨부하는 형태를 많이 사용하며, 이를 OCSP stapling이라 한다. 또한 4가지 단계 모두를 SCVP(Server-based Certificate Validation Protocol)를 이용하여 온라인으로 문의하여 확인할 수 있다[6].

공개키 기반구조에는 크게 인증기관, 등록기관(RA, Registration Authority), 검증기관(Validation Authority), 공개 디렉터리 등이 참여한다. 등록기관은 사용자의 신원을 확인하는 기관으로 인터넷 뱅킹에서는 은행이 그 역할을 하고 있다. 검증기관은 OCSP나 SCVP 등을 서비스하는 기관이며, 공개 디렉터리는 전화부처럼 발급된 모든 사용자의 인증서를 보관하고 사용자들의 요청에 따라 필요한 인증서를 전달해주는 기관이다. 하지만 공개 디렉터리는 현재 현장에서 사용하고 있지는 않다.

5.2 신원기반 공개키 암호시스템

공개키 암호알고리즘에서 공개키의 인증은 매우 중요하며, 이를 위해 인증서를 사용한다. 또한 앞서 살펴본 바와 같이 인증서를 안전하게 활용하기 위해서는 공개키 기반구조가 확립되어 있어야 한다. 하지만 공개키 기반구조를 구축하고 운영하는 것은 비용 등 여러 측면에서 어려움이 많다. 따라서 인증서를 사용하지 않고 공개키를 사용하는 방식에 대한 연구가 있었지만 이 절에서 살펴보는 신원기반 공개키 방식을 제외하고는 특별한 대안이 도출되지 못하였으며, 신원기반도 몇 가지 근본적인 문제가 있어 현장에서는 대안으로 사용하고 있지 못하다. 최근에는 블록체인 기술을 활용한 탈중앙 공개키 기반구조가 대안으로 검토되고 있다. 이것에 대해서는 다음 절에서 살펴본다. 물론 비트코인처럼 공개키 인증 메커니즘 없이 공개키 기술을 활용할 수 있지만 대부분의 응용은 공개키 인증이 중요하고 필요하다.

신원기반 공개키 암호시스템은 1984년 Shamir[1]가 처음 제안하였지만, 전자서명만 가능하였고 암호화가 가능하지 않았다. 2001년에 Boneh와 Franklin[4]은 겹선형 사상(bilinear pairing)을 이용한 신원기반 시스템을 제안하면서 그 당시에 다시 연구가 활발하게 진행되었지만, 지금까지도 근본적인 문제들은 해결을 못하고 있다.

신원기반 공개키 방식의 기본 생각은 전자우편주소나 주민등록번호처럼 사용자의 잘 알려진 독특한 신원정보로부터 그 사용자의 공개키를 계산하여 사용하는 것이다. 사용자들은 다른 사용자의 공개키를 그 사용자의 신원정보로부터 직접 생성할 수 있기 때문에 인증서라는 것을 이용하여 공개키의 소유자를 확인하는 과정이 필요 없게 된다. 즉, 신원기반 공개키 방식에서는 통신하고자 하는 상대방이나 제3의 서버에 문의하지 않고 상대방의 신원 정

보를 알고 있다면 상대방의 공개키를 생성할 수 있는 장점이 있다. 그러나 이 생각은 이 출발부터 근본적인 문제를 내포하고 있다.

일반 공개키 방식에서 사용자는 직접 공개키 쌍을 생성한 다음 개인키는 비밀로 유지하고 공개키를 인증기관에 보내 인증서를 발급받는다. 따라서 해당 사용자 외에 인증기관조차도 사용자의 개인키를 모르게 된다. 이 때문에 부인방지 서비스를 매우 안전하게 제공할 수 있다. 신원기반 공개키 방식은 사용자의 공개키를 그 사용자의 신원 정보만 알면 누구나 생성할 수 있다. 반면에 일반 공개키 방식에서는 보통 개인키를 생성한 다음 그것에 대응되는 공개키를 생성한다. 하지만 신원기반에서는 누구나 공개키를 생성할 수 있기 때문에 대응되는 개인키까지 누구나 생성할 수 있도록 만들 수는 없다. 이 때문에 신원기반 공개키 방식에서는 개인키를 생성하는 별도의 신뢰 기관을 사용한다. 이를 PKG(Private Key Generator)라 한다. PKG는 자신만이 알고 있는 마스터키를 가지고 있으며, 이 마스터키와 사용자의 공개키를 이용하여 해당 사용자의 개인키를 생성하게 된다.

신원기반 공개키 방식에서 PKG는 모든 사용자의 개인키를 생성하고 유지할 수 있기 때문에 막강한 능력을 갖추게 된다. PKG는 사용자의 개인키를 이용하여 해당 사용자 행세를 할 수 있으며, 특정 사용자에게 전달된 암호문을 항상 복호화할 수 있다. 이와 같은 문제는 1장 조건부 프라이버시에서 언급한 임계기반 비밀 공유 기법을 사용하여 해결할 수 있다. 여러 기관에 마스터키를 분산 공유하여 여러 기관이 동의할 때에만 생성할 수 있도록 만들 수 있다.

신원기반 공개키 방식은 이보다 더 근본적인 문제를 가지고 있다. 신원기반 공개키의 핵심은 공개키를 사용자의 신원 정보를 이용하여 계산하기 때문에 인증서가 필요 없다는 것이다. 여기서 신원 정보는 각 사용자마다 독특한 정보이어야 한다. 그렇지 않으면 두 사용자가 같은 공개키를 가질 수 있으며, 이는 두 사용자가 같은 개인키를 사용하게 되는 것을 의미한다. 더욱이 정보가 독특하다고 하여 이들을 모두 공개키를 계산할 때 활용할 수 있는 것은 아니다. 주민등록번호, 여권번호 등은 사용자마다 독특한 정보이지만 타인이 보통 쉽게 얻거나 알 수 있는 정보가 아니다. 이 측면에서 사용자의 메일주소나 휴대전화번호가 신원정보로 사용하기에 가장 적합한 것으로 보인다.

하지만 신원 정보라는 것은 보통 불변 정보이다. 불변 정보란 바꿀 수 없는 정보를 말한다. 물론 메일주소나 휴대전화번호는 엄밀한 의미의 불변 정보는 아니지만 자신이 널리 사용하는 주소나 번호를 공개키 쌍을 바꾸기 위해 변경하는 것은 불편하다. 따라서 불변 정보를 바꾸어 공개키를 변경하는 것은 널리 사용할 수 있는 방법이 아니다. 물론 PKG의 마스터키를 변경할 수 있지만, 이 경우에는 해당 사용자뿐만 아니라 모든 사용자의 공개키를 새롭게 계산하여야 하므로 이 역시 사용할 수 있는 방식이 아니다. 이에 생성 시간과 같은 변경 가능한 정보를 불변정보와 같이 사용하는 방식이 제안되기도 하였지만 사용자들이 다른 사용자의 생성 시간을 알아야 하기 때문에 신원기반 공개키 방식의 원래 장점이 퇴색한다. 물론 고정된 생성 시간을 사용할 수 있지만, 이 경우에는 주기적으로 공개키 쌍을 갱신하여야 하며, 주기 내에서는 변경이 어렵게 되는 문제점이 있다.

신원기반 공개키 방식은 키 갱신 문제와 함께 검토되어야 하는 키의 폐지 문제도 해결할 수 있어야 한다. 키를 폐지한다는 것은 키 갱신을 의미하며, 과거 폐지한 공개키를 다른 사용자들이 사용할 수 없도록 해야 한다. 이것도 신원기반 공개키 방식의 장점을 유지한 상태에서 해결하기 어렵다는 문제점을 가지고 있다. 이와 같은 것들을 해결하지 못하고 있기 때문에 인증서 기반 공개키 방식의 대안으로 사용하고 있지 못하다.

5.3 탈중앙 공개키 기반구조

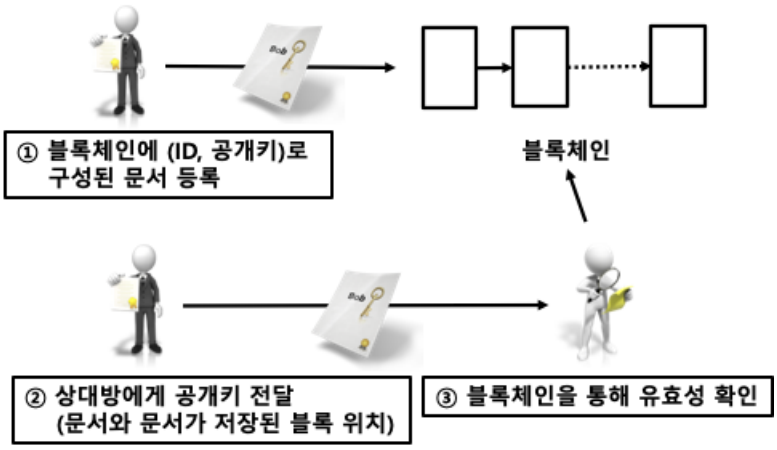
비트코인이 등장으로 비트코인에서 사용하는 **블록체인**(blockchain) 기술이 각광을 받게 되었다. 블록체인에 대해서는 14장과 15장에서 자세히 설명하지만 이 절에서는 탈중앙 공개키 기반구조를 이해할 수 있는 수준에서 간단히 설명하고자 한다. 블록체인은 분산 합의 기술을 이용하여 데이터를 자동으로 분산 저장(참여하는 모든 노드에 같은 데이터를 유지함) 해주며, 블록체인에는 데이터를 추가만 할 수 있고, 기록된 기존 데이터를 수정 및 삭제할 수 없다.

비트코인에서 각 사용자는 공개키 쌍을 생성하여 사용하며, 공개키의 해시값을 자신의 지갑 주소로 활용한다. 이 지갑에 있는 비트코인을 지불에 사용하기 위해서는 대응되는 개인키를 알아야 한다. 이처럼 비트코인은 공개키 기술을 사용하지만 인증서를 사용하고 있지 않다. 비트코인은 어떤 형태의 중앙기관도 사용하지 않는 것이 목표이

기 때문에 기존처럼 신뢰할 수 있는 인증기관이 발급하는 인증서를 사용할 수 없다. 하지만 비트코인에서도 특정 사용자의 지갑 주소를 착각하여 다른 지갑의 주소로 비트코인을 보내면 돈을 잃게 되므로 코인을 양도할 때 목적 주소가 자신이 실제 양도하고 싶은 대상의 주소인지 확신할 수 있어야 한다. 이 확신을 인증서를 이용하지 않고 각 개인에게 맡기는 방식을 사용하고 있다. 거래 당사자들이 오프라인에서 서로의 지갑 주소를 교환하는 등의 방법을 사용하여 신뢰를 구축하는 방법을 사용하고 있다.

이와 같은 형태로 공개키 기술을 사용하는 것이 보편화되면서 공개키 기반구조 자체를 블록체인을 이용하여 탈중앙 방식으로 제공하는 방법을 연구하게 되었다. 그 중에 하나가 현재 DID(Decentralized ID)의 기본 개념에 해당하는 DPKI(Decentralized PKI) 이다 [7]. DPKI는 중앙집중 신뢰기관이 인증서를 발급하지 않고, 각 개체가 스스로 인증서를 발급하고 관리하는 형태이다. 현재 웹은 중앙집중 PKI를 사용하고 있으며, 수 많은 사설 인증기관이 존재한다. 각 브라우저는 신뢰하는 사설 인증기관을 소프트웨어 내에 내장하고 있다. 공격자는 이들 인증기관 중 하나만 공격하는데 성공하면 피싱 사이트를 쉽게 만들어 사용자들을 속일 수 있다. 이 문제를 극복하고자 하는 것이 DPKI의 가장 큰 목표이다.

DPKI에서는 사용자의 실제 신원과 공개키를 바인딩하는 문서를 사용하지 않고, 조회가 가능한 일정한 규격을 갖춘 ID와 공개키를 바인딩한다. 실제 신원과 연결할 수 있지만 프라이버시 측면에서 현재 연구되는 DPKI는 신원과 공개키를 바인딩하지 않는다. 블록체인 기반 DPKI는 ID와 공개키를 바인딩하는 문서를 블록체인에 기록한다. 따라서 블록체인은 기존 PKI에서 공개 디렉토리 역할을 한다.



<그림 2.3> DPKI 동작 원리

블록체인에 저장은 블록체인에서 사용하는 분산 합의 기술에 의해 이루어지지만 DPKI의 이해를 위해서는 아무나 저장할 수 있다고 생각하여도 된다. DPKI에서는 중앙기관의 승인 없이 누구나 공개키 쌍을 생성하고 그것과 바인딩할 ID를 만들어 블록체인에 저장하여 사용할 수 있다. 이렇게 ID와 공개키를 바인딩하여 주는 문서가 블록체인에 저장되면 누구나 쉽게 특정 ID의 공개키를 조회하여 얻을 수 있다. 또 블록체인은 첨삭 전용이므로 저장된 공개키를 조작하거나 수정할 수 없다. DPKI의 기본적 원리는 그림 2.3과 같다.

특정 개체의 공개키를 얻기 위해서는 이 개체가 DPKI에서 사용하는 ID를 알아야 한다. 이 연결은 오프라인에서 서로 교환하거나 기관의 경우에는 기관 홈페이지에 ID를 게시하는 방법 등을 사용할 수 있다. 따라서 누구나 ID를 등록할 수 있지만 해당 ID가 실제 누구의 ID인지 확인하는 것은 DPKI가 아니라 각 응용에서 별도 방법을 제공해야 한다.

PKI 신뢰 모델 중 신뢰 웹(web of trust) 모델이 있다. 이 모델은 개인 간의 신뢰를 통해 신뢰를 확장하는 모델이며, 이메일 보안을 위해 개발된 PGP에서 사용한 모델이다. 신뢰 모델 중 TOFU(Trust On First Use) 모델도 있다. 이 모델에서 첫 신뢰 관계 형성은 무조건적 신뢰에 의존한다. 그다음부터는 첫 번째 형성된 관계를 이용하여 검증하며, 외부적 방법을 통해 언제든지 관계를 점검하고 중단할 수 있다. 보통 메신저나 SSH에서 이와 같은 모델을 사용한다. DPKI도 이와 같은 모델을 사용할 수밖에 없다.

인증서는 발급뿐만 아니라 갱신, 폐지할 수 있어야 하며, 주어진 인증서가 폐지된 인증서인지 알 수 있어야 한다. 블록체인은 첨삭 전용이기 때문에 DPKI에서 갱신, 폐지는 모두 기존 ID와 관려하여 새 문서를 블록체인에 등록하여 이루어진다. ID와 관련하여 가장 최근에 등록된 문서에 있는 공개키가 해당 ID의 유효한 공개키가 되며, 등록된 문서에 공개키 항이 null 값이 등록되어 있다면 해당 ID와 관련된 공개키가 폐지된 것을 의미한다. 이 과정이 안전하기 위해서는 아무나 기존 ID에 대한 문서를 새롭게 블록체인에 등록할 수 없어야 한다. 기존 ID와 연결된 공개키에 대응되는 개인키를 모르면 이 요청을 할 수 없으며, 블록체인에서 사용하는 합의 기술에서는 이 부분을 검증하게 된다. 이 합의 기술에서는 중복 ID의 저장도 걸러준다. 보통 블록체인에서 합의 기술에 참여하는 노드는 특별한 장려책이나 패널티 시스템이 있어, 시스템에서 정한 규칙에 따라 정직하게 행동하지 않으면 불이익을 받게 된다.

각 사용자는 다른 사용자에게 공개키를 전달할 때 블록채인에 등록된 자신의 문서와 해당 문서가 어느 블록에 저장되어 있는지 알려 준다. 이와 같은 정보를 수신한 사용자는 블록체인을 통해 해당 문서가 유효한 문서인지 확인해야 하며, 이 문서가 그 이후 블록에서 갱신 또는 폐지되었는지 확인할 수 있어야 한다. 블록체인 자체를 유지하는 사용자는 이것을 쉽게 할 수 있지만 그렇지 않은 일반 클라이언트에게는 어려운 작업이 될 수 있다. 이 문제를 해결하는 한 가지 방법으로 암호학적 축적기(cryptographic accumulator)[8]의 사용이 제안되기도 하였다. 모든 유효한 ID, 공개키 쌍을 작은 값으로 축적한 최신 값을 매 블록마다 유지하면 사용자는 본인이 유지하고 있는 쌍이 여전히 유효한 것인지 마지막 블록만 다운받아 쉽게 확인할 수 있다.

이와 같은 방식에서 한 가지 문제점은 개인키를 분실했거나 개인키가 타인에게 탈취되었을 때이다. 탈취의 경우에는 먼저 폐지하여 문제를 해결할 수 있지만 분실하였을 경우에는 유효한 갱신 요청을 만들 수 없다. 이 경우에는 새 ID를 만들어 사용하고, 기존 ID는 해당 ID의 등록된 키들의 유효기간이 스스로 지나도록 기다린 방법밖에 없다. 이와 같은 문제 때문에 키를 복구하는 서비스를 보통 함께 제공한다. 사용자의 개인키를 임계 기반 비밀 공유 기법으로 분산 저장하고, 필요하면 이를 이용하여 복구할 수 있도록 해준다.

DPKI에서 블록체인을 사용하는 이유를 요약하면 다음과 같다.

- 이유 1. 자동 분산 저장된다. 단일 실패점이 제거되며, 쉽게 특정 ID의 공개키를 확보할 수 있다.
- 이유 2. 등록된 ID의 공개키를 조작하는 것이 어렵다.

5.4 FIDO

FIDO(Fast Identity Online) 인증은 구글, 마이크로소프트, 모질라와 같은 기업이 참여하는 FIDO 연합에서 개발한 빠르고 간결하며 안전한 인증 방법들을 말한다. FIDO는 웹 서비스에서 주로 사용하는 패스워드를 이용한 사용자 로그인을 대신하기 위해 사용하는 인증 기술이다. FIDO도 공개키 기술을 사용하지만 공개키의 소유주를 증명하기 위한 인증서를 사용하지 않는다. 각 사용자는 특정 서비스에 등록할 때 새로운 공개키 쌍을 생성하며, 공개키 쌍 중 공개키를 서비스 서버에 등록한다. 사용자는 이 서비스에 로그인할 때마다 등록된 공개키에 대응되는 개인키를 소유하고 있음을 증명해야 한다. 이 증명은 사용자가 사용하는 클라이언트 소프트웨어가 자동으로 해준다. 다만, 안전하게 보관된 개인키를 사용자가 활성화해주어야 한다. 이때 FIDO는 다양한 인증 방법을 사용할 수 있도록 해준다. 보통 국내에서는 지문 인증을 많이 사용한다. 최초 서비스 등록할 때 클라이언트 소프트웨어는 사용자의 지문 정보를 받아 등록하며, 사용자 지문이 올바르게 인식된 경우에만 안전하게 보관된 개인키가 활성화된다. 이처럼 FIDO는 기존에 살펴본 PKI와 달리 하나의 공개키 쌍을 여러 서비스에 사용하는 것이 아니라 각 서비스마다 다른 공개키 쌍을 사용한다. 하지만 사용자가 이들 공개키 쌍을 관리하는 것이 아니라 사용하는 클라이언트 소프트웨어가 관리해준다.

참고문헌

[1] Adi Shamir, "Identity-Based Cryptosystems and Signature Schemes," Advances in Cryptology, CRYPTO 1984, LNCS 196, pp. 47–53, Springer, 1985.

[2] V. Miller, "Use of Elliptic Curves in Cryptography," Advances in Cryptology, CRYPTO 1985, LNCS 218, pp. 417–426, Springer, 1986.

[3] A. J. Menezes, T. Okamoto, S. A. Vanstone, "Reducing Elliptic Curve Logarithms to Logarithms in a Finite Field," IEEE Transactions on Information Theory, Vol. 39, No. 5, pp. 1639–1646, Sep. 1993.

[4] Dan Boneh, Matthew Franklin, "Identity-based Encryption from Weir Pairing," Advances in Cryptology, CRYPTO 2001, LNCS 2139, pp. 213—229, Springer, 2001.

[5] S. Santenssion, M. Myers, R. Ankney, A. Malpani, S. Galperin, C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP," IETF RFC 6960, Jun. 2013.

[6] T. Freeman, R. Housley, A. Malpani, D. Cooper, W. Polk, "Server-based Certificates Validation Procotol (SCVP)," IETF RFC 5055, Dec. 2007.

[7] C. Allen, A. Brock, V. Buterin, J. Callas, D. Dorje, C. Lundkvist, P. Kravchenko, J. Nelson, D. Reed, M. Sabadello, G. Slepak, N. Thorp, H.T. Wood, "Decentralized Public Key Infrastructure," `https://danubetech.com/download/dpki.pdf`, Dec. 2015.

[8] David Derler, Christian Hanser, Daniel Slamanig, "Revisiting Cryptographic Accumulators, Additional Properties and Relations to Other Primitives," Topics in Cryptology – CT-RSA 2015, LNCS 9048, pp. 127-144, Springer, 2015.

퀴즈

1. 다음 중 인증서에 포함되지 않는 것은?
   ① 공개키
   ② 공개키의 소유자 정보
   ③ 개인키
   ④ 유효기간
2. 대칭 암호알고리즘 E가 주어졌을 때, 다음 중 가능하지 않은 것은? 여기서 $K_1 \neq K_2$이며, $M_1 \neq M_2$이다.
   ① $E.K_1(M) = E.K_2(M) = C$
   ② $E.K(M_1) = E.K(M_2) = C$
   ③ $E.K_1(M_1) = E.K_2(M_2) = C$
   ④ $E.K_1(M) \neq E.K_2(M)$
3. 공개키 암호알고리즘을 사용할 때 공개키 인증이 매우 중요하다. 공개키의 소유자를 올바르게 확인할 수 없는 경우에 발생할 수 있는 문제점을 모두 선택하시오.
   ① 개인키로 암호화된 것을 다른 사용자가 한 것으로 착각할 수 있음
   ② 공개키로 암호화하였지만 기대하는 상대방이 복호화할 수 있는 것이 아니라 다른 사용자가 복호화할 수 있게 될 수 있음
   ③ 주어진 공개키에 대응되는 개인키를 쉽게 계산할 수 있음
   ④ 사용하는 알고리즘의 성능이 나빠질 수 있음
4. 신원기반 공개키 방식의 가장 큰 문제점은?
   ① PKG의 권한 집중 문제
   ② 사용자가 개인키를 생성할 수 없는 문제
   ③ 키를 갱신하는 문제

④ 공개키를 생성하는 문제

5. 한 번 확인한 인증서를 같은 용도로 다시 사용할 때 항상 다시 확인해야 하는 것을 모두 고루시오.

① 인증서의 서명 값 확인
② 인증서의 사용 용도
③ 인증서의 폐지 여부
④ 인증서의 유효기간
⑤ 인증서의 발급자

6. 탈중앙 공개키 기반구조에서 블록체인을 사용하는 이유가 아닌 것은?

① 주어진 ID에 대한 공개키를 쉽게 조회하기 위해
② 등록된 공개키에 대한 조작이 가능하지 않도록
③ ID와 신원 정보를 쉽게 바인딩하기 위해
④ 인증서의 유효기간
⑤ 자동으로 분산 저장하기 위해

연습문제

1. 대칭 암호알고리즘은 암호화할 때와 복호화할 때 같은 암호키를 사용하는 반면에 비대칭 암호알고리즘은 암호화할 때와 복호화할 때 서로 다른 암호키를 사용한다. 두 방식의 암호알고리즘을 원격에 있는 두 사용자가 활용하고자 한다. 이때 반드시 선행되어야 하는 것을 설명하시오.

2. 공격자가 메시지 M과 그것에 대응되는 암호문 C를 알고 있고, 암호화 알고리즘이 결정적 알고리즘일 때 원격에 있는 두 사용자가 C를 다시 교환하게 되면 공격자가 어떤 메시지를 교환하였는지 알게 된다. 따라서 암호화 알고리즘은 확률적 알고리즘이 되는 것이 안전성에 바람직하다. 그런데 공개키 암호알고리즘에서는 이것이 더욱 더 필요하다. 그 이유를 설명하시오.

3. 공개키 인증서가 비대칭 암호알고리즘을 사용할 때 필요한 이유를 설명하시오.

4. 공개키 인증서의 유효성을 확인할 때 폐지여부를 확인해야 하며, 현재 폐지여부를 확인하기 위해 인증서폐지목록이라는 것을 활용한다. 인증서폐지목록에는 폐지된 인증서의 모든 정보가 저장되는 것은 아니다. 또한 인증서폐지목록이 무한정 커지는 것이 아니다. 그 이유를 간단히 설명하시오.

5. 신원기반 암호시스템은 공개키 방식이지만 인증서의 사용이 필요 없다. 그 이유를 설명하시오. 또 이 시스템이 인증서 기반 공개키 암호시스템을 대체하고 있지 못하는 이유를 간단히 설명하시오.

6. 영문 텍스트를 암호화하는 방법으로 매핑하는 테이블을 이용한다고 가정하자. 즉, 26개의 영문 문자를 각각 다른 영문 문자로 매핑하는 테이블(A $\rightarrow$ C, B $\rightarrow$ N, ..., Z $\rightarrow$ A)을 이용하는 것을 말한다. 다음 각각에 대해 답변하시오.

① 가능한 키의 개수는?
② 이와 같은 방식으로 암호화하였을 때 문제점을 한 가지 설명하시오. (키의 개수나 보관과 관련된 문제는 아니며, 공백, 마침표, 특수문자 등의 암호는 무시하고 암호문을 통해 노출되는 정보가 무엇인지 생각해보시오)

7. HTTPS 프로토콜로 접속이 가능한 웹 사이트에 접속하여 해당 사이트의 인증서 정보를 추출하여 발급한 인증기관, 공개키의 종류, 공개키의 길이와 같은 중요 정보를 제시하시오.