**DES, AES**

텍스트, 영수증, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

암호화 과정에서 key의 개수를 2개로 k1 , k2를 만들어 암호화 할 때, key의 개수를 늘리면 사용하는 key의 길이는 n+m이 될 수 있지만, 암호화 블럭은 (n, m) 길이를 갖는다. 이것은 n bits의 키와 m bits의 키를 2개 푸는 것과 같기 때문에 보안성은 그다지 크게 증가하지 않는다. 반면 m+n은 키의 길이와 암호화 블럭의 길이가 m+n이기 때문에 m+n bits의 키를 모두 해독 해야한다. 이렇게 되면 부루트 포스 공격에 대한 강한 저항력을 갖게 된다.

**MAC (Message Authentication Code)**

텍스트, 폰트, 화이트, 대수학이(가) 표시된 사진

자동 생성된 설명

MAC : 송수신자(A B)가 서로 키를 공유(비밀키)해 둔다. 송신자 A는 평문인 message와 message를 공유해둔 키를 사용해 mac값을 계산한 메시지 인증코드를 보낸다. 수신자 B는 공유된 키를 사용해 수신 받은 message로 mac값을 계산한다. 이때 B가 계산한 mac값과 A가 계산하고 보낸 mac값을 비교해 A에게 온 message인지 인증한다. 비교가 실패하면 A에게 온 것이 아니라고 판단한다.

단점 : 미리 키를 공유해야 해야 하는데 이때 키가 공격자에게 알려진다면, 공격자는 모든 message를 자유롭게 변경 할 수 있다.

텍스트, 폰트, 화이트이(가) 표시된 사진

자동 생성된 설명

RSA : RSA 공개키 암호화 방식은 메시지를 암호화 할 때 사용되는 공개키, 암호화된 메시지를 복호화하기 위한 개인키가 존재하며 두개의 키는 한쌍으로 생성되어 관리된다. (비대층키 암호화 알고리즘) AES 대칭키 암호화 방식에 비해 속도가 느린 단점을 갖지만 보안에 강점을 갖는다.

RSA 단점 : RSA를 이용해 digital signiture을 교환하는 용도로만 사용하고, key exchange를 사용하면 위험하다. key exchange는 Diffie-hellman을 사용하는게 안전하다. 위험한 이유는 PFS를 해야하기 때문이다. 숭수신측은 Shared key를 가지고 세션이 시작할때 계속 같은 Shared key를 사용하기 때문에 이전과 이후에 전달하는 세션이 모두 해킹에 노출된다.

1. RSA는 송신할 A측은 두개의 public key(ku)와 private key(kr)를 만들고 public key를 B에게 보낸다. 수신한 B는 무작위로 shared key(sk)를 만들고, A측의 public key를 사용해 암호화해서 다시 A에게 암호화된 shared key를 보낸다. 이러한 과정으로 다음부터 사용할 shared key를 양측이 공유하게 된다. 만약 A가 B에게 public key로 암호화된 c를 보내는데 중간자가 mitm acttack으로 c를 가로채 중간에 c를 c’으로 바꿔치기를 할수 있다. 이때 c’은 A의 public key와 n(mod n)값을 가지고 변형된 c’을 만든다. 이때 만들어진 c’는 암호문(c)을 복호화 하지 않고도 평문을 추측해 잘못된 평문으로 B에게 전달시킬수 있다.
2. 이 문제가 발생하는 이유는 가)의 RSA방식은 암호화 할 때 난수를 사용하지 않아서 같은 메시지를 암호화하면 항상 같은 암호문이 나오기 때문이다. 때문에 메시지를 암호화 할때마다 난수를 생성하고 이 난수로 메시지를 암호화해 X와 Y를 만들어 사용한다. 이 방법을 사용하면 같은 메시지라도 다른 암호문으로 나오기 때문에 평문을 추측할 수 없다.

텍스트, 폰트, 영수증, 화이트이(가) 표시된 사진

자동 생성된 설명

Digital signature (전자 서명)은 위조될 가능성이 매우 희박하고, 전달하는 메시지가 수정 되었는지 누가 보냈는지 알 수 있다.

1. m(메시지)을 hash 함수(sha2,sha3)를 사용해 message digest로 만든 뒤 수신측의 private key로 암호화해 Digital signature을 만들어낸다. 만들어진 Digital signature는 m와 함께 송신된다.  
    텍스트, 도표, 스크린샷, 원이(가) 표시된 사진

   자동 생성된 설명
2. 수신측은 m(메시지)와 암호화된 Digital signature를 수신한다. m은 hash 함수(sha2,sha3)를 사용해 message digest로 만들고 암호화된 Digital signature를 public key를 사용해 복호화 한다. 복호화된 값과 message digest을 비교해 같다면 데이터가 공격 받지 않고 변경, 위조되지 않았다고 생각할 수 있다.   
   텍스트, 스크린샷, 도표, 폰트이(가) 표시된 사진

   자동 생성된 설명

텍스트, 영수증, 폰트, 대수학이(가) 표시된 사진

자동 생성된 설명

# 2016 중간

1. discuss why secret-key encryption, decryption can not be used for non repudiation

**- 비밀키 암호화 방식(대칭키)이 복호화를 부인 방지에 사용할 수 없는 이유는?**

비밀키(대칭키) 암호화 방식 : 데이터를 암호화및 복호화 하는데 동일한 비밀키 한 개를 사용한다. 비밀키 암호화 방식에서 비밀키의 크기는 [공개키](https://terms.tta.or.kr/dictionary/dictionaryView.do?subject=%EA%B3%B5%EA%B0%9C+%ED%82%A4)에 비해 상대적으로 작고 [암호 알고리즘](https://terms.tta.or.kr/dictionary/dictionaryView.do?subject=%EC%95%94%ED%98%B8%EC%95%8C%EA%B3%A0%EB%A6%AC%EC%A6%98) 내부 구조가 단순하여 암호화와 [복호화](https://terms.tta.or.kr/dictionary/dictionaryView.do?subject=%EB%B3%B5%ED%98%B8%ED%99%94) [처리 속도](https://terms.tta.or.kr/dictionary/dictionaryView.do?subject=%EC%B2%98%EB%A6%AC+%EC%86%8D%EB%8F%84)가 [공개키](https://terms.tta.or.kr/dictionary/dictionaryView.do?subject=%EA%B3%B5%EA%B0%9C+%ED%82%A4) 암호화 방식에 비해 매우 빠르다. 반면 송신자와 수신자가 동일한 비밀키를 공유해야 하는데 비밀키를 분배하는 과정 중 누출이 되면 누구라도 암호문을 [복호화](https://terms.tta.or.kr/dictionary/dictionaryView.do?subject=%EB%B3%B5%ED%98%B8%ED%99%94)할 수 있는 단점이 있다.

부인방지란 송신자나 수신자가 메시지를 주고받은 사실을 부인하지 못하도록 방지하는 것 을 말한다. 비밀키 암호화 방식은 같은 비밀키를 송신자와 수신자가 공유하며 메시지를 복호화하고 암호화한다. 송신하는 측과 수신하는 측은 항상 같은 키로만 복호화 암호화하기 때문에 수신측이 암호화 하였는지 송신측이 암호화 하였는지 알 수 없다. 예를 들어 A(송신자), B(수신자)가 메시지를 주고 받는데 A가 보낸 메시지를 A가 부정하며 B가 만들어낸 메시지라고 주장한다면 B는 A가 만든 메시지라고 증명할 수 없다.

텍스트, 폰트, 화이트, 대수학이(가) 표시된 사진

자동 생성된 설명

**ECC (Eliiptic curve cryptography)**

타원곡선암호 : 타원곡선을 기반으로 한 암호방식으로, 대표적인 [공개키](http://wiki.hash.kr/index.php/%EA%B3%B5%EA%B0%9C%ED%82%A4) 방식 중 하나이다. 타원곡선암호 기술은 기존 [RSA](http://wiki.hash.kr/index.php/RSA) 암호방식에 대한 대안으로 1985년도에 제안된 방식이다. [비트코인](http://wiki.hash.kr/index.php/%EB%B9%84%ED%8A%B8%EC%BD%94%EC%9D%B8)이 타원곡선암호 기술 방식에 속하는 [타원곡선 디지털서명 알고리즘](http://wiki.hash.kr/index.php/%ED%83%80%EC%9B%90%EA%B3%A1%EC%84%A0_%EB%94%94%EC%A7%80%ED%84%B8%EC%84%9C%EB%AA%85_%EC%95%8C%EA%B3%A0%EB%A6%AC%EC%A6%98)(ECDSA; Elliptic Curve Digital Signature Algorithm) 암호 알고리즘을 사용하고 있다. 타원곡선암호는 [타원곡선](http://wiki.hash.kr/index.php/%ED%83%80%EC%9B%90%EA%B3%A1%EC%84%A0) 암호화 알고리즘이라고도 부른다.

암호 원리 🡪 타원곡선 암호는 타원곡선 위의 이산로그 문제가 어렵다는 사실을 이용한 공개키 암호 방식

**ECC와 RSA**

ECC는 RSA보다 짧은 키를 사용하지만, 안전성 면에서는 RSA와 비슷한 수준을 제공한다. 따라서 ECC의 암호화 방식이 RSA 보다 더 작은 비트로도 동일한 안전성을 얻을 수 있다. ECC는 RSA보다 연산이 복잡하다는 단점이 있다.

**ECDLP(타원곡선 이산대수 문제)**

P : 생성자, 타원곡선 상의 임의의 점

k : 개인키보다 작은 소수로, 난수 생성기로 생성

Q : 공개키, 개인키로부터 연산

공개키 Q는 Q = kP = P+P+…+P 으로 P를 k번 덧셈을 한 값임

이때 Q = kP 수식에서 k와 P를 이용해 Q를 구하기는 쉽지만, 알려진 P값과 Q값을 통해 k값을 구하기 어려운 점을 이용했다.(이산대수 문제)

**ECDH (Eliiptic curve Diffie-Hellman)**

A와 B가 키를 교환하려고 한다. 먼저 A와B는 사용할 curves를 정하고 이에 필요한 도메인 파라미터를 사용하기로 한다. A는 kA라는 난수를 생성하고, 사용하기로 한 도메인 파라미터를 사용해 (QA = kA \* G(공개키)) B에게 QA를 전송한다. B도 마찬가지로 kB라는 난수를 생성하고, 사용하기로 한 도메인 파라미터를 사용해 (QB = kB \* G(공개키)) A에게 QB를 전송한다. 그러면 A와 B는 서로 QA와 QB를 공유하게 된다. 이때 A는 자신의 난수 kA와 QB를 가지고 (SAB=KA \* QB) SAB를 만들면 SAB는 (SAB = KA \* KB \* G)가 된다. 마찬가지로 B도 자신의 난수 kB와 QA를 가지고 (SAB=KB \* QA) SAB를 만들면 SAB는 (SAB = KB \* KA \* G)가 된다. 이로써 A와 B는 같은 SAB를 공유하게 된다.

텍스트, 폰트, 라인, 친필이(가) 표시된 사진

자동 생성된 설명

**ECDSA (Eliptic Curve Digiter Signature Algorithm)**

**서명 :** DSA와 마찬가지로 ECDSA 서명은 메시지를 해시하여 메시지 다이제스트를 만든 후 개인키를 사용하여 서명 요소 (r, s)를 만들어 내는 과정이다. ECDSA 개인키를 d, 서명하고자 하는 메시지를 m, 메시지를 해싱하는 해시함수를 H 라고 했을 때, ECDSA 서명은 다음과 같은 과정을 통해 서명 (r, s)를 산출한다.

1. {1, ..., n-1} 사이의 랜덤한 수 k를 선택한다.
2. k\*G 를 계산하여 결과값 (x1, y1)을 구한다.
3. r = x1 mod n 식을 사용하여 r을 계산한다. 매우 드물지만 r=0 이면, 다른 k를 선택해서 반복한다.
4. s = (k-1 (H(m) + dr)) mod n 식을 사용하여 s를 계산한다. 만약 s=0 이면, 다른 k를 선택하여 처음부터 다시 계산한다. 여기서 해시함수는 H(m)을 통해 도메인 파라미터 n과 같은 비트길이를 리턴하는 함수로 가정하는데, 만약 해시함수 H()가 더 긴 비트길이를 리턴한다면, H(m)의 해시값으로부터 좌측에서 n의 비트 길이만큼 만을 취한다.
5. (r, s) 쌍이 디지탈 서명이다.

DSA에서와 마찬가지로, 개인키(d)와 함께 랜덤 수 k는 외부에 노출되지 말아야 한다. 만약 k가 알려진다면, 역으로 계산해서 개인키가 노출될 가능성이 있다. 또한, 랜덤 k는 서명할 때마다 매번 다른 값을 사용하여, 동일한 메시지라도 매번 다른 디지탈 서명을 만들도록 해야 한다.

**검증 :** ECDSA 검증은 메시지(m)와 서명(r, s)를 받아들여 공개키(Q)로 메시지에 대한 서명이 맞는지를 검사하는 과정이다. 메시지를 해싱하는 해시함수를 H 라고 했을 때, ECDSA 서명은 다음과 같은 과정을 통해 검증한다.

1. 상대의 공개키(Q)가 타당한지 체크한다. Q는 무한원점(O)이 아니어야 하고, 타원곡선상에 있는 점이어야 하며, n\*Q = O 이어야 한다.
2. r, s의 범위가 0 < r < n, 0 < s < n 인지 체크한다.
3. w = s-1 mod n 을 계산한다.
4. u1 = H(m)\*w mod n 을 계산한다. 여기서 해시함수는 H(m)을 통해 도메인 파라미터 n과 같은 비트길이를 리턴하는 함수로 가정하는데, 만약 해시함수 H()가 더 긴 비트길이를 리턴한다면, H(m)의 해시값으로부터 좌측에서 n의 비트길이만큼만을 취한다.
5. u2 = r\*w mod n 을 계산한다.
6. (x1, y1) = (u1\*G + u2\*Q) mod n 을 계산한다. 만약 (x1, y1)가 무한원점(O)이면 잘못된 서명이다.
7. v = x1 % n 을 계산한다.
8. v가 r과 동일하면 맞는 서명이다.

ECDSA의 서명과 검증 과정은 타원곡선(EC)를 사용하는 것을 제외하고는 기본적으로 DSA와 동일한 메카니즘을 사용한다.

아래 클래스는 ECDSA에서 해싱된 메시지에 대해 서명을 생성하는 메서드(SignHash)와 그 서명을 검증하는 메서드(VerifyHash)를 예시한 것이다.

텍스트, 폰트, 번호, 스크린샷이(가) 표시된 사진

자동 생성된 설명




1. AES 키 길이(대칭키 알고리즘) : 128bits 이상의 키 길이를 사용한다. 부루트 포스 방식의 공격을 막기 위해 키의 길이가 충분히 커야한다.
2. RSA 키 길이(비대칭키 알고리즘) : 1024, 2048 bit를 가장 많이 사용한다. 키 길이가 길어질 수록 안전해지지만, 암호화 복호화 시간이 길어진다.
3. ECC 키 길이(비대칭키 알고리즘) : 타원곡선을 사용한 암호화 기법으로 이산대수를 활용한 암호화 방식을 사용하기 때문에 복잡한 구조를 갖고있다. RSA 보다 짧은 키를 사용한다.
4. 송수신자가 서로 비밀 key를 공유하고 이 key를 가지고 암호화, 복호화에 사용한다. 그렇기 때문에 처리속도는 비대칭키 알고리즘보다는 빠르다.
5. 암호화하는public key와 복호화하는 private key와 같이 key를 2개 사용한다. 하지만 암호화 복호화 방식이 매우 복잡해지기 때문에 처리속도는 대칭키보다 느리다.
6. RSA와 ECC의 보안 강도가 같다고 할 때 키의 길이는 RSA가 더 길고 ECC의 Key의 길이는 더 짧다. 때문에 처리 속도는 ECC가 RSA 보다 더 빠르다.
7. Message 인증코드로 사용 할 수 있다.
8. 인증과 전자 서명에 이용