정보보안 중간고사 통합본

정보보안

사용자가 서버에 메세지를 보낼 때, 메세지에 대한 내용을 서버와 사용자만 알도록 해야한다(confidentiality). 중간 내용이 변조되지 않도록(integrity). 메세지를 받지 않았다고 우기는 일이 없도록(non-repudiation). 효율성이 있도록 하는것 필요.(efficiency) **네가지 중요.**

보통 세가지 기술 사용.

- 암호기술 중 시메트릭 키 알고리즘(대칭키)
- 공개키(어시메트릭 키/비대칭키)
- 해시함수 cryptographic hash

대칭키

메시지 주어졌을때 알고리즘에 넣어서 키값을 아는 사람만 해독할 수 있도록 하는 것을 대칭키 알고리즘.

> K_s 라는 것은 시메트릭 키 라고 함. 키를 사용해서 암호화 하는 것을 의미.

원본으로 돌아오게 하기 위해서는 K_s 동일 키가 필요. $K_s(K_S(m)) = m$

비 대칭키

개인키 공개키 키 쌍을 가지고 있음. 두개의 키. $K_A^- K_A^+$,(-가 개인키) 공개키로 암호화 하고 이것을 풀 수 있는 방법은 개인키로만 가능하다. 반면에 개인키로 암호화 한것은 서명 했다라고 한다. 개인키로 암호화 한것은 대응되는 공개키로 해독한다.

암호학적 해시

메시지를 암호학적 해시를 적용하면 일정한 용량으로 축소되게 된다. 예를들어 256비트. 이 256비트의 값끼리가 동일할 확률은 매우 적다.

예시

앨리스(사용자)와 밥(서버) 통신. 앨리스가 밥에게 정보를 보낸다고 한다. 밥은 그러면 앨리스에게 공개키를 보낸다. 앨리스는 세션키K_s를 하나 생성한다.(대칭키) 밥이 준 공개키를 이용해서 이 세션키를 암호화$K_B^+(K_s)$ 한다. 그리고 보내고자 하는 메시지를 세션키를 가지고 암호화$K_s(m)$ 한다. 원본 메세지에 해시함수를 적용하고 앨리스의 개인키로 암호화(전자서명)$K_A^-(H(m))$ 한다.

이렇게 하는 이유는 4가지를 만족해야 하기 때문이다.

그래서 총 세가지 $K_B^+(K_s), K_s(m), K_A^-(H(m))$ 를 밥이 받는다. 그러면 밥의 개인키로 K_s 를 얻어내고 메세지를 해독한다. 그리고 해독된 메세지를 가지고 해시를 적용시킨뒤 앨리스가 보낸것과 비교해 서명을 확인한다.

해시함수가 동일한 가치를 지닌다고 할 수 있기 때문에 Non-repudiation만족. 해시값이 동일하기 때문에 integrity 만족. 메세지에 대해 바로 서명하게 하면 매우 느림. 세션키를 가지고 해시로 만든다음에 암호화 하여 효율성도 만족. 세션키를 암호화 하여 효율성을 만족.

여기서 문제가 있는데, 밥이 처음에 내려준 공개키가 신뢰 할수 있는지에 대한 문제이다. 때문에 CA라고 하는 인증 기관을 도입 하게 됨. 인증 기관도 자신의 공개,개인 키가 있다. 밥이 자신의 공개키를 내려줄 때 인증기관의 서명을 받는다.$K_{CA}^{-}(K_{B}^{+}, Bob)$ 이렇게 받은 인증된 키를 내려준다. 앨리스는 이것을 받아 CA의 공개키로 밥이 준 값을 해독한다. 그러면 공개키와 밥의 서명을 받게 된다.

그러면 또 CA의 공개키가 정당한 것인지 검증해야 하는 문제가 발생하는데, 이 문제를 해결 하기 위해 브라우저를 처음 설치 할때 공인된 상위 인증기관의 키는 내장되어 설치하게 된다.

보안 개론

보안의 목표

일반적으로 3가지. Confidentiality(기밀성), Integrity(무결성), Availability(가용성)(CIA)

- 기밀성 : 남들이 모르게. 파일 암호화 등의 방법으로 가능. 허가되지 않은 정보의 유출을 막는 것. 암호화가 주된 요소.
- 무결성 : 내용이 바뀌지 않았음을 증명 할 수 있어야 한다.
- 가용성 : 사용 가능한 상태

Confidentiality

암호화 하여 메세지를 전달하여 기밀성을 유지한다. 그런데 암호화를 풀 키를 전달하는데 문제가 있다. 키를 안전하게 전달할 방법에 관한 문제가 존재하는 것이다. (키 관리의 이슈)

Access Control(접근 제어) : 인가된 사람만 접근 할 수 있도록 하는 것을 접근 제어라고 한다. Ex) 관리자 제어.

달성 도구로써 Authentication,물리적 보안을 이용할 수 있다.

Authentication

- 그사람이 가진 것을 보고 그 사람인 것을 파악한다.(something the person has) 예를들어 자동차의 스마트 키, 공인 인증서, 난수표, 신분증 ID 카드
- 그사람이 아는 것을 가지고 파악한다(something the person knows). 패스워드 같은 것.
- 그사람인 것을 증명한다(something the person is). 생채인증, 지문 등의 방법.

Physical security

물리적 보안. 보안 위험한 곳 등에서는 와이파이를 사용하지 않는다. 금고의 사용 등.

- Faraday cages : 알루미늄 포일 같은 것으로 방을 모두 감싸면 전파가 나가지 못함.

Integrity

데이터에 결함이 발생하면 안된다.

Tools

- 백업: periodic archiving of data(아카이빙)
- 체크섬 : 데이터가 깨졌는지 알 수 있는 것.
- 데이터 복구 코드(Data crrecting codes)

> Raid(Redundant Array of Independent Disks) : 서로다른 디스크를 묶어 데이터 결함을 복구하도록 하는 것.

Availability

내가 원하는 시간에 정보에 접근하여 사용할 수 있어야 한다.

Tools

- Physical protections : 물리적으로 달성
- Computational redundancies : 컴퓨테이션 정보를 중복적으로 관리한다.

또 다른 컨셉

AAA - Authenticity(진위 - 그사람이 했다 까지 볼 수 있음, 서명), Assurance(보증), Anonymity(익명)

- 보증
- 진위 : 전자 서명, nonrepudiation(부인 봉쇄) 나중에 아니라고 말 못하게 함.
- 익명 : 데이터 보안 이슈로 핫함.
    - Aggreagation : 개인 개인의 많은 데이터를 합침
    - Mixing : 섞어서 모르게 함.
    - Proxies : tor 라는 기술이 있음. 접속한 것을 모르게 하는 것이다.
    - Pseudonyms : 별명

Threats and attacks

위협과 공격을 대비해야 함.

Eavesdropping

엳듣는 공격과 위협

Alteration

중간에 내용을 변화하거나 함. **man-in-the-middle attack** 중간자가 변조시켜 공격함.

Denial-of-Service

예를들어 스팸 이메일 테러. (요즘엔 쉽게 막음)

Masquerading

변장. 막기 위해 강력한 인증 필요

Repudiation

부인. 전자 서명 기술로 막음

열가지 보안 원칙 중 몇가지 중요한 원칙

- Economy of mechanism : 보안이라는 것도 경제성을 생각해야 한다.
- Fail-safe default : 장애에 대해 우선순위나 해결에 대함

- Open design : 오픈소스와 비슷한 개념. 암호 알고리즘 등 공개할 것인가 폐쇄적으로 할 것인가.
- Separation of privilege : 권한 분산. 개발자와 운영자를 다른 사람으로 할것. 등
- Least privilege : 딱 권한 만큼만 일을 주는 것.

Role based Access Control

사람에게 권한을 주는 것이 아닌, 역할에 대해서 접근 제어를 주는것.

Caesar Cipher

최초의 암호.

정보보안

One time pad

연속된 비트를 패드라고 함. 비밀 키값을 서로 나누어 가진다. 암호할 메세지에 키값을 끼워넣는다. XOR연산을 수행해서 값을 만든다. 이것을 보낸다. 받은 값에 대해 키값을 XOR연산 수행하면 본래의 값이 나오게 된다.

매번 동일한 유효 키값을 생성하는 것이 어려워 실제로 쓰이기가 다소 어려움. 메세지가 길어진다면 그에 해당하는 긴 키값이 필요함. 그래서 긴 키를 매번 생성하기가 어려움.

대칭키 암호화 알고리즘

Block cipher

64비트로 쪼갬. 쪼갠 것들을 블록단위로 암호화. 마지막 것이 모자라면 패딩을 붙임.

DES, AES

DES

초기 컴퓨터에서 많이 이용했었음. 이후 컴퓨팅 파워가 올라가서 이용이 어려워졌음.(쉽게 뚫림.) 키가 64비트. 그중 8비트는 체크섬용. 실제 사용은 56비트. 56비트로부터 48비트짜리 랜덤 넘버를 생성하여 각각 암호화 하는데 이용한다.

Modes of operation

블록 단위로 자르기 때문에 암호화를 하더라도 원본의 형태를 어느정도 추측이 가능한것이 약점이었다. 여러개의 암호화된 것을 대조해서 힌트가 얻어지는 것. 때문에 블록 암호화를 ECB모드로 하면 안전하지 않음.

- CBC(Cipher block chaining)

  암호화된 블럭을 가지고 체인을 만든것. 앞번 만든 암호화 한것이 뒤에 암호화에 영향 미치도록 한다. XOR암호화에 앞번 암호화 결과를 가지고 추가로 암호화 한다.

- Initialization Vector

  처음 블록의 값이 바뀌면 전체 값이 모두 바뀌기 때문에.

  처음 블럭은 단순 암호화만 하는데 이 첫번것에 대해 시작 벡터를 넣어 암호화 하는것. 그리고 보내는 사람에게 벡터를 같이 알려주는 것.

- Message Authentication Code(MAC)

나한테 메세지가 잘 왔는지 확인하는 것. 맨 마지막에 CBC로 생성된 블럭을 MAC으로 이용하자는 것. CBC residue. 이 맨 마지막 값을 받는측에서 비교하여 동일하다면 값이 정상적으로 왔구나를 확인함.

- integrity 검증 방법
    1. CBC checksum
    2. hash
    3. 전자서명 (개인키로 서명)

Triple DES

DES를 세개 붙인것.

M -> E(k1) -> D(k2) -> E(k1) -> C

하지만 최근에는 이것도 안전하지 않다고 생각함.

Streaming ciper

한바이트 한바이트 랜덤 넘버 생성함. 메세지와 결함하여 XOR처리. 바이트 단위로 처리하는 것을 스트리밍 서비스라고 하기에 비슷한 의미임.

정보보호

AES

키 길이는 DES에 비해 더 길고 시간은 짧음. 인터넷에서 가장 많이 쓰임. 길이는 128/192/256 비트 이용 가능.

pading

Block cipher에서 뒤에 모자란 블록에 대해 패딩을 붙여서 비트수를 맞춘다. 그런데 이 패딩을 어떤 표준화된 값을 부여할 것인가에 대한 문제. 모듈러 연산 8을 한다. 0~7까지 나오는데, 7은 1바이트만 넣으면 된다. 6은 2바이트 5는 3바이트.. 이런식. 넣는 바이트는 1,2,3,4...08순

```
mod 8 == 7 -> 0x01
mod 8 == 6 -> 0x0202
mod 8 == 5 -> 0x030303
...
mod 8 == 0 -> 0x0808080808080808
```

Stream cipher

가장 유명한 것은 RC4. 원본 메세지 i번째 바이트와 i번째 생성한 키값을 XOR해서 바이트를 만든다. 무한한 바이트 단위의 랜덤 넘버를 생성해 적용한다.

반복된 패턴이 나오면 안된다. 통계적으로 완벽하게 랜덤이여야 한다. 다양하고 추측이 되지 않도록 생성해야 한다. 간단하지만 강력한 암호 알고리즘.

AES나 차후 기술들에 비해서 더 빠르다 라고 하기엔 다소 어려움. 스트림도 대칭키 알고리즘인데, 대칭키 알고리즘이 공개키보다 더 빠르다. 대칭키보다는 해시가 더 빠르다.

Attack

- 암호화 된 것을 아는 상황
- plaintext와 암호화된 것을 아는 상황
- 암호 알고리즘의 특정 결과를 아는상황(암호화를 시킬 수 있는)
- 특정 부분의 암호 알고리즘의 결과를 아는 상황(암호문도 정할수 있고 원본문도 정할 수 있는 상황.)

어택에 따라 목표가 다르기도 하고 결과도 다르다.

숙제

AES알고리즘을 사용하여 키를 생성하고 AES를 사용하여 ECB모드를 사용하서 PKCS5 패딩 이용. 암호문을 만들고 다시 그 암호문을 해독하는 프로그램을 만드시오.

어떤 파일을 암호화하고 복호화하는 숙제.

해시

크리토그래피 암호 해시임.

> 여담으로 트리와 해시 둘다 서칭에 이용되는데 해시는 빅오1이지만 트리는 빅오 로그엔. 해시가 아닌 트리를 이용에 고려하는 이유는 범위 연산이 가능하기 때문.

메세지 m과 m'가 있다고 상정. (m'는 m과 1비트만 차이남.) h(m)과 h(m')는 몇 비트 차이날까? 동일 비트 위치에서 50%확률로 같을 수 있다. (?) 완전히 다르게 나와야 한다는건가?

message digest function. 해시가 적용된 값은 digest값이라고 함. 해시는 one-way. 만들어진 해시로 원본을 추측하거나 알수 없어야 한다.

현시점 인터넷어서 가장 많이 쓰이는 크리토스래픽 해시는 SHA 256, SHA 512. SHA1을 개조해서 만든것.

크게 두단계로 적용된다.

1. 메세지를 쪼개고 패딩을 붙인다.
2. 각각을 암호학적 해시를 통해 값을 만든다. (앞의 계산된 값을 반영하여 암호화한다.)

- Birthday attack

  $365!/365^n(365-n)!$ 이다. 이것이 사람이 23명 정도가 모이면 약 50%의 확률로 한쌍 이상이 같은 것이 있다는 것이다.

  해시에서 생각해보면 256비트에서 메세지에 갯수에서 충분하게 많아진다면 동일하게 될 확률이 무시하지 못할 수준이 된다는 것이다.

MAC을 암호학적 해시로 만들 수 있다. 키 K를 먼저 공유한다. m과 h(m)을 공유한다면 되는것이 아닌가라고 생각할 수 있겠지만 중간에서 바꿔치기 할 수 있으므로 아니다. m과 h(K||m)을 보낸다. K와 m을 컨캣한것을 공격자는 모르기 때문에 바꿔치기 하는것이 불가능 한 것이다.

정보보안

개인/공개키

사람이 많아질 수록 많은 수의 대칭 키가 필요함.(n(n-1)/2개) 통신의 참가하는 참여자들이 개인키와 공개키를 갖는 형태. 필요한 키 수가 획기적으로 줄어듦.

공개키 알고리즘은 서명도 가능하도록 만들었음. 개인키로 서명 한 뒤 공개키로 이를 확인. 서명확인을 통해 무결성을 확인 할 수 있다.

해시함수

적용하면 고정된 크기의 결과값이 나옴. 이것은 해쉬를 적용할 때에는 쉽게 할수 있지만 역으로 해석할 수 있어선 안된다(일방향성).

충돌이 나는 것이 불가능해야 한다. SHA256의 경우 해시값이 같을 확률이 매우 극악.

무결성을 확보하는데 이용 할 수 있다.

MAC

무결성 확보를 위해 메세지 M에 대한 해시값을 같이 보낸다. 메세지를 받은 쪽에서 메세지를 해시돌려보고 그것을 받은 해시와 비교한다. 같다면 무결성.

패스워드

패스워드 vs 키

패스워드는 사람이 생성하고 사람이 기억하는것. 키는 기계가 생성함.

만일 패스워드가 패스워드 본래 값 자체로 저장된다고 한다면, 서버가 해킹당했을 때 모든 정보가 다 유출되게 된다. 때문에 해쉬값으로 변경해서 저장함. 로그인 할 때 입력한 패스워드에 해쉬를 적용하여 전송해 확인한다. 암호학적 해쉬는 역으로 해석이 불가능하므로 안전하다.

하지만 추측공격으로 인한 공격은 막기 어려운데, 때문에 패스워드를 복잡하게 해야 하는 것이다.

사회 공학적 공격

기술을 아무리 좋게 만들어도 사람을 속이면 정보를 캐낼 수 있음. 이메일 피싱이나 사칭

대칭키 암호화 알고리즘

Conventional/Symmetric 알고리즘이라고 한다. 센더와 리시버가 공유하는 공통 키를 가진다. 텍스트를 암호화 하면 같은 키로 복호하여 이용한다.

키를 전달하기 위한 별도의 안전한 채널이 있다고 전제한다. 키 보관에 대한 문제가 항상 존재한다.

암호 해석기술

암호학적 공격을 할 수 있다.

정보보안

악성코드 등 변경된 것을 확인 하는데에도 해시가 이용된다. 파일에 대해 해시를 적용한 값들을 가지고 있는다. 이것을 비교해 변조 되었는지 파악. 해시값을 해킹한다는 문제도 있는데 때문에 h(k|f1) 형태로 저장하여 키값을 함께 넣어 변조 위험이 없게 한다.

OTP

세가지 방식

- Time-Sync(시간 동기화)

  OTP장비를 먼저 받음. 고객 정보에 장비 시리얼 넘버를 적용. h(K|serial|time)을 보낸다. 서버가 받으면 h(K|serial|time)을 계산하고 받은값과 비교한다.

> 리플레이 어택. 중요 정보(오티피 키 등) 복제한 뒤 재 요청해서 공격하는 방식. 유효 시간을 줄여서 막을 수 있다.

- Challenge-response

  키값을 사용자와 서버가 공유하고 있음. 사용자가 서버에 접속 요청.서버가 랜덤 넘버 R을 전송. 사용자가 R을 키값 적용하여 전송. 서버는 이 값을 다시 복호화해서 R값을 비교.

- Hash chain

  비밀 키는 사용자만 알고있음. 키에 대해 해시를 지속적으로 적용이 가능하다. h(k), h(h(k)), h(h(h(k)))...fn(k) 이것을 디비에 저장한다. 인증할 때에는 fn-1(k)를 보낸다. 서버는 받은 값에 대해 해시를 한번 적용한다. 이 값과 저장된 값을 비교한다. **해시 함수는 역으로 알아내는 것이 불가능 하기 때문에 안전하다.** 서비스 이용이 끝나면 fn-1(k)는 노출되었기 때문에 해당 값을 저장하고 다음 접근에는 fn-2(k)를 보낸다. 이렇게 가다보면 k를 만나게 되는데, 이때에는 키를 업데이트 하여 다시 등록함. 이 방법의 장점은 키를 사용자만 안다는 것이다.

공개키

공개키 암호화 알고리즘은 두가지 목적.

- 암호 복호

  기밀성 유지를 위해 이용 - confidentiality.

- 전자 서명

  부인 금지 - Non-repudiation

개인키와 공개키를 이용한다. 브라우저 등에서 이용한다. 한가지 단점. 느리고 컴퓨팅 파워를 많이 잡아먹음. 공개키 암호화 알고리즘은 대칭키를 교환하는 수단으로 암호화 복호화 많이 함. 전자서명도 해시함수를 만들고 전자서명을 하는 등의 형태를 취한다.

큰 메세지는 효율이 좋은 대칭키로 암호화 하고 대칭키를 공개키로 암호화 하여 전송.

- 세션키

  한 세션에 대해 유효한 키.

정보보호

RSA

앨리스와 밥이 통신을 공개키 알고리즘으로 수행한다고 할때,

큰 소수 p,q 와 $\phi(n) = (p-1)(q-1)$, 1<e<$\phi(n)$, e*d = 1 mod $\phi(n)$ 를 만족하는 d를 구하고 이것을 키로 이용하는 것.

메세지 m을 바이트 시퀀스로 생각할 때, 이것은 각각 0~127의 숫자로 표현 가능할 것이고, 이것을 쭉 나열하면 하나의 큰 숫자로 표현 가능할 것이다.

이 m을 e번 곱한다. $K_A^+(m) = m^e mod(n)$ 이것이 암호화 한것. 복호화는 개인키로 한번 알고리즘 수행하면 되는데 알고리즘 적으로 보면 $(m^e mod(n))^d mod(n) = m$ 이 성립한다. 이것을 증명하기 위한 몇가지 공식

$m^{ed} mod(n) = m mod(n)$

Fermat's little theorem

서로소 관계의 두수 gcd(a,p) 라면, $a^{p-1} mod(p) = 1$

증명

1xa,2xa,3xa ... (p-1)xa 이것을 모두 p로 나눴을때 나머지를 생각해보자. 1xa mod p ... (p-1) mod p 이렇게 했을때 나머지가 모두 다 다르다.

- 귀류법

  오류를 거짓으로 검증해 참을 얻어내는 것. 여기서는 어떤 2개의 나머지가 같다는 전제로 시작한다.

이후 a x 2a x 3a x 4a .... (p-1)a mod p = a mod p * 2a mod p * 3a mod p ... (p-1)a mod p 이것을 다시 정리하면,

1x2x3x4x...(p-1) mod p.

a x 2a x 3a x 4a .... (p-1)a mod p = 1x2x3x4x...(p-1) mod p. 양번에 (p-1)!로 나눔. 그다음 여차저차 풀면 증명임.

$$(m^2 mod(n))^d mod(n)(m^e)(m^e)(m^e)..d\text{번이므로}$$

$$m^{ed} modn$$

$$m * m^{k\phi(n)} modn$$

$$(mmodn) * (m^{k*\phi(n)} modn) modn(e * d\text{는} 1 + k * \phi(n)\text{이므로})$$

$$(m^{\phi(n)} modn)^k modn//\phi(n) = n = 1$$

$$m^{n-1} modn\text{에서}, m = a, n\text{은} p$$

정보보안

공개키 알고리즘

메세지가 크다는 전제하에 메세지를 대칭키로 암호화하고 대칭키를 공개키로 암호화한다. 메세지를 대칭키로 암호화한 값과 대칭키를 암호화한 값을 보낸다.

메세지에 서명하는 것과 메세지의 해시값에 대해 서명하는 것을 동등한 보안으로 본다. 서명한 h(m)과 받아온 m를 해시한 값을 비교한다. 그 값이 같으면 서명한 것으로 간주해 메세지를 이용한다.

Diffie-Hellman

- discrete logarithms

discrete logarithms문제에서 고안한다. $y = a^x$ 가 주어졌을때 y, x 가 주어진다면 이것이 충분한 수라고 할 때 a를 맞추기가 굉장히 어렵다. 하지만a, x 가 주어지면 y값 찾기가 쉽다.

y가 주어질때 p*q 를 구하는 것은 어렵다. 하지만 p,q를 주고 y를 찾는것은 쉽다. RSA는 이런 소인수 분해에서 도출된 알고리즘.

앨리스와 밥이 대칭키로 통신한다고 한다. 대칭키를 공유하는 것이 어렵다. 앨리스는 큰 소수p와 그것보다 작은수 g를 가진다. 이것이 쌍으로 이루어 공개키이다. 이것을 밥에게 보낸다. 그다음 앨리스가 x라는 넘버를 생성한다. 이것은 알려주면 안되는 값이다. 앨리스는 $g^x mod(p)$를 한 값을 전달한다. 밥은 $g^y mod(p)$ 값을 전달한다. 앨리스는 받은 값에 대해 x승을 곱한다.$(g^y mod(p))^x mod(p)$ 이 값의 결과는 $g^{yx} mod(p)$ 밥도 동일한 과정 하면 $g^{xy} mod(p)$ 두 값을 비교하면 값이 같은 값이다. 키의 길이는 p의 값만큼 나오게 되는것.

공격자는 p,g,$g^x mod(p), g^y mod(p)$ 를 안다. 그런데 이것을 가지고 x와 y를 알아낼 수 없다. 이것이 discrete logarithms의 문제이다.

x,y,p,g는 충분히 큰 값이여야 한다.

RSA

- ((a mod n) + (b mod n)) mod n = (a+b) mod n
- ((a mod n) - (b mod n)) mod n = (a-b) mod n
- ((a mod n) * (b mod n)) mod n =(a*b) mod n
- ((a mod n)^b) mod n = a^b mod n

앨리스가 공개키 알고리즘 이용을 위해 큰 소수 p와 q를 생성한다. n을 계산. n=p*q ϕ(n) = (p-1)(q-1) e값을 구한다. $e, 1 < e < \phi(n), gcd(e, \phi(n)) = 1$을 만족하는 것을 찾는다. 파이n과 공통으로 나누어지는 수가 없는 e를 구하는 것이다. 서로소가 되는 e를 찾는것. gcd는 소인수분해.

e*d = 1 mod $\phi(n)$ 를 만족하는 d를 생성한다. 이게 핵심 e와 곱해졌을때 나머지가 1인 수.

(e*d) mod $\phi(n)$ = 1 인 d를 찾는것임. 중요

k의 개인키 = (d,n) , k의 공개키 = (e,n)

정보보안

RSA

페르마의 소정리 $a^{p-1} mod p = 1$,p는 소수. 오일러 정리 $a^{\phi(n)} mod n = 1$, a와n서로소. $\phi(n)$ 이란 1부터 n이하 중 n과 서로소인 숫자의 갯수.

p가 소수 라면 : $\phi(p)$ = p-1

p,q 는 소수 n=p*q. $\phi(n) = p * q - (pq$와서로소가아닌것의갯수$)$

p*q = p,px2,px3,px4,....px(q-1) , qx1,qx2,qx3 .... ,qx(p-1) 이므로

$p * q - (p - 1 + q - 1 + 1) = pq - p - q + 1 = (p - 1)(q - 1)$

$\phi(n) = (p - 1)(q - 1)$

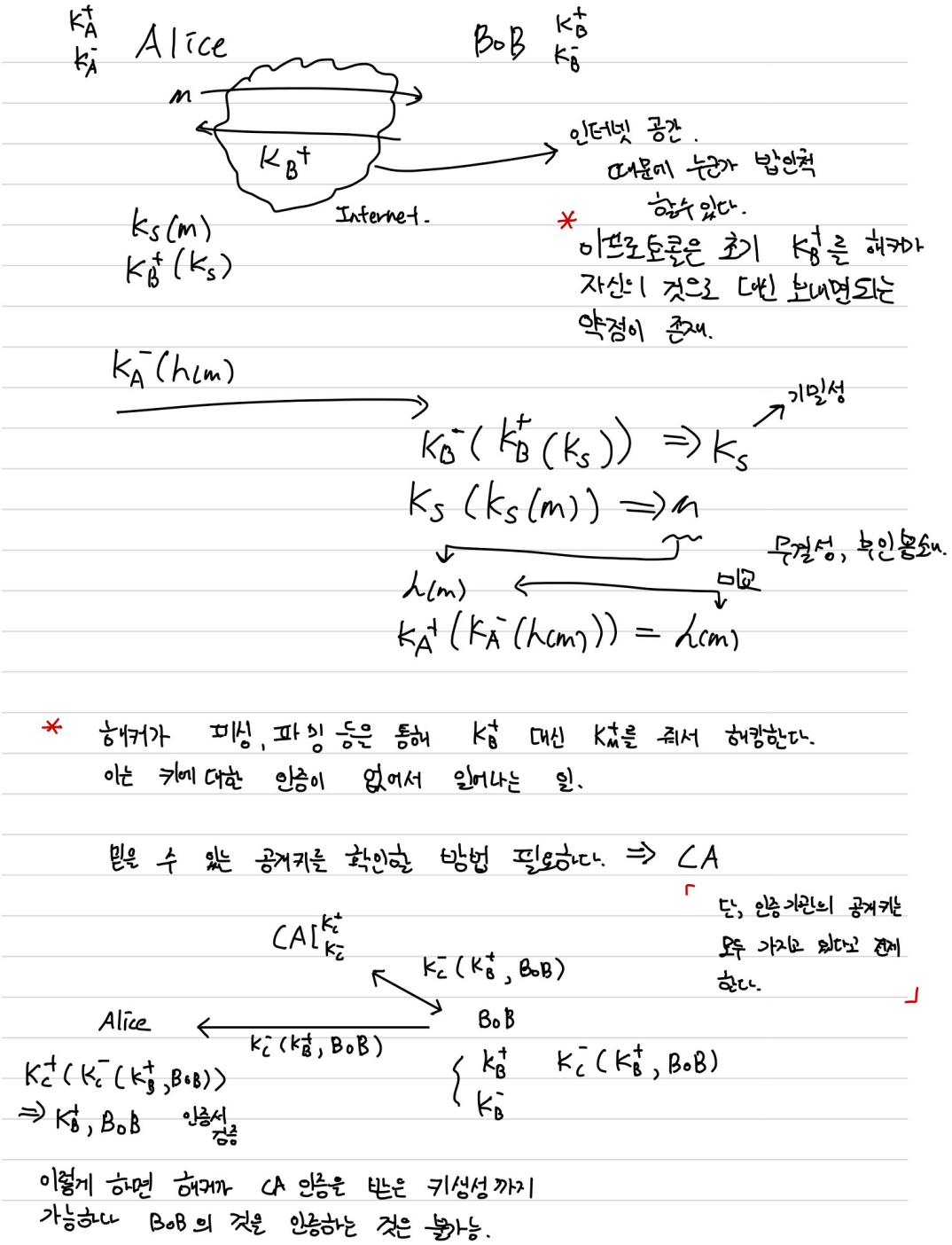
$e, gcd(e, \phi(n)) = 1 => d, ed = 1 mod \phi(n)$

$K_B^+(m) = m^e mod n$

$$K_B^-(m) = (m^e mod n)^d mod n = m^{ed} mod n = m^{k\phi(n)+1} mod n = m^{\phi(n)k} mod n = m mod n => m$$

매우 중요

앨리스와 밥이 공개키로 통신. 앨리스가 큰 메세지 밥에게 보내고 싶다고 할때 밥이 밥의 공개키를 보내준다. 앨리스는 자신의 대칭키를 생성해서 메세지를 암호화한다. 이 대칭키를 공개키로 암호화 한다. 서명을 위해 메세지를 해시한 것을 앨리스 개인키로 암호화 해서 전송한다.



정보보안

X.509

공개키인증서, 인증 알고리즘의 표준 가운데 공개키 기반의 표준. CA가 인증서를 작성함에 있어 나름의 표준이 있어야 다른 곳에서도 이용할 시 이용이 가능하다. 따라서 표준이 필요하다. 버전, 시리얼 넘버, 발급자 서명, 유효 기간등 표준이 있다.

전세계 객체에게 공개키/개인키를 어떻게 표현하게 할 것인가. 본래 X.500이라는 것에 일부. 이것은 전세계 모든 객체에게 보안 프로토콜 제공이 목적이었음. 이것이 실패했다.

LDAP(Lightweight Directory Access Protocol)이라는 것이 살아남았다. X.500에서 일부 실용적인것을 구현한 것 중에 하나. 이것을 응용한 것이 MS directory server. 직원들의 주소록이다. 이것은 권한의 정도를 포함한다.

> 특이하게 국내의 공인인증서 안에는 해당 발급자의 주민등록번호를 해시한 값을 포함하고 있다.

만일, 어떤 해커에 의해 인증서의 완전한 플레인텍스트가 해킹된것 같다고 합리적 의심이 될 때, CA에 알려야 한다. 만료 및 해킹에 의해 인증서를 더이상 신뢰하지 못할때 신고 해야한다. revoking하게 만들어야 한다.

CA는 **Certificate revocation List(CRL)**을 만들어 놓는다. 만일 인증서가 해킹당해 의심당할 경우 블랙리스트인 이 리스트에 올린다. 이용자들은 인증서의 유효함을 CRL에서 찾아보고 이용하게 된다. 블랙리스트에 존재한다면 안전하지 않은 인증서일 가능성이 높으므로.

사실 CA는 이 CRL을 지속하는 것이 부담이 된다. 때문에 두가지 시나리오 취함.

- 24시간중 특정 시간에 한번에 업데이트함.
- 인증서의 상태를 온라인으로 체크하는 프로토콜을 만든다. (OCSP)

OCSP는 부담되기 때문에 일정 이용 수수료를 받는다.

- Delta CRL : CRL 리스트중 바뀐 정보들에 대해서만 전송하자는 생각.

국제 표준

ITU-T/CCITT, ISO

X.400, X.500, X.700 : 하지만 표준이 너무 구현 어려워 일부만 구현해 이용한다.

IETF

SMTP, LDAP, SNMP

SSL

전송계층에서 암호화 하는 것. TCP위에서 가동이 된다. 그 자체의 레코드 포맷이 존재한다. 메세지에 대한 무결성 체크를 한다. 해시 함수를 이용하여 검증.

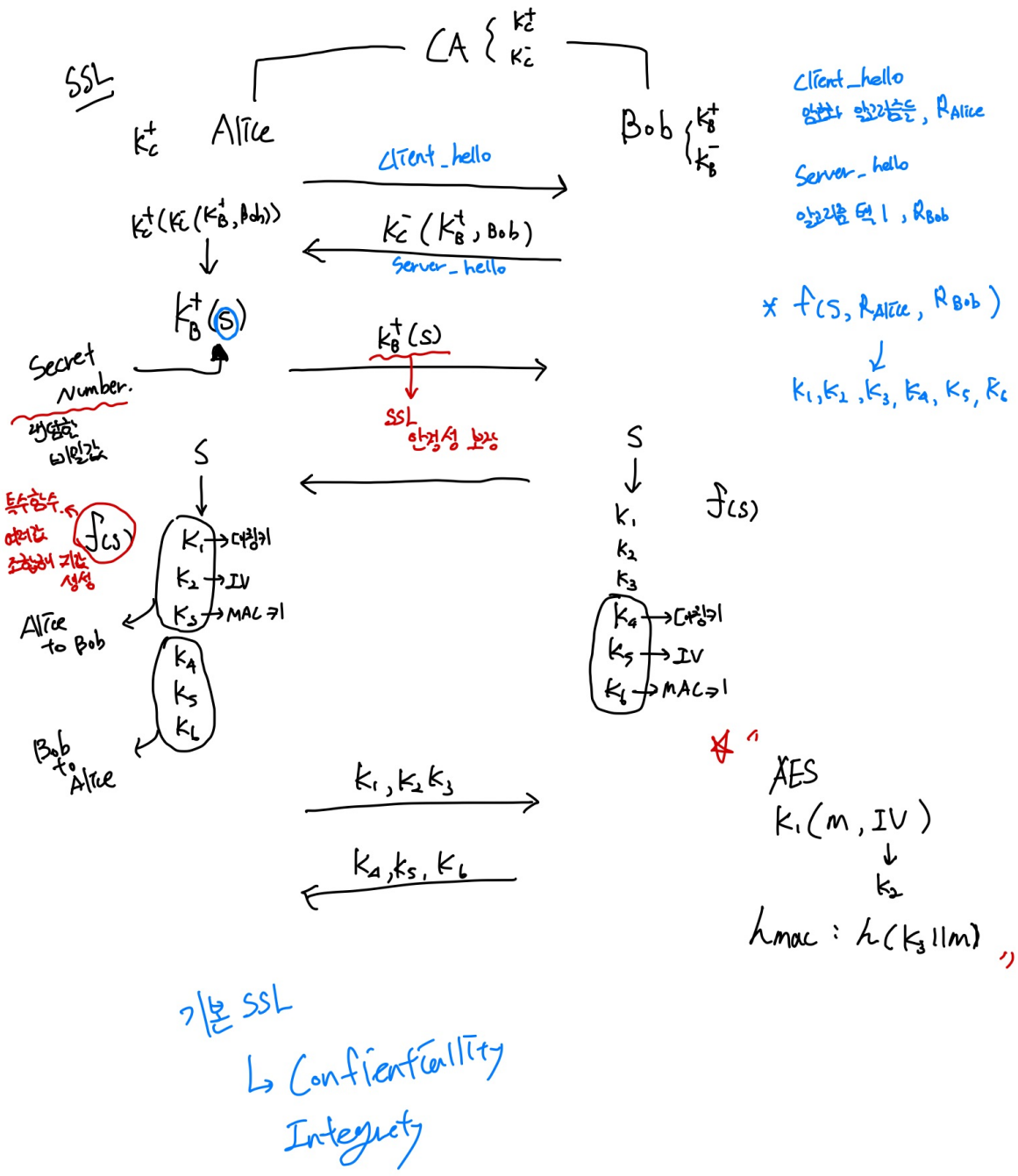
Alert protocol. 문제가 발생했을 때 알려준다. Fatal Error 등

앨리스가 밥에게 통신을 요청한다. 이때 사용할 수 있는 암호화 알고리즘들을 전송한다. 랜덤 넘버를 함께 준다 (Client Hello) 웹서버는 자신의 인증서를다운로드 해준다. 그리고 암호화 알고리즘을 택해 내려준다. 랜덤 넘버를 내려준다. (server hello) 사용자가 내려온 인증서를 통해 암호화 통신.

SSL Session

Handshake protocol

정보보안



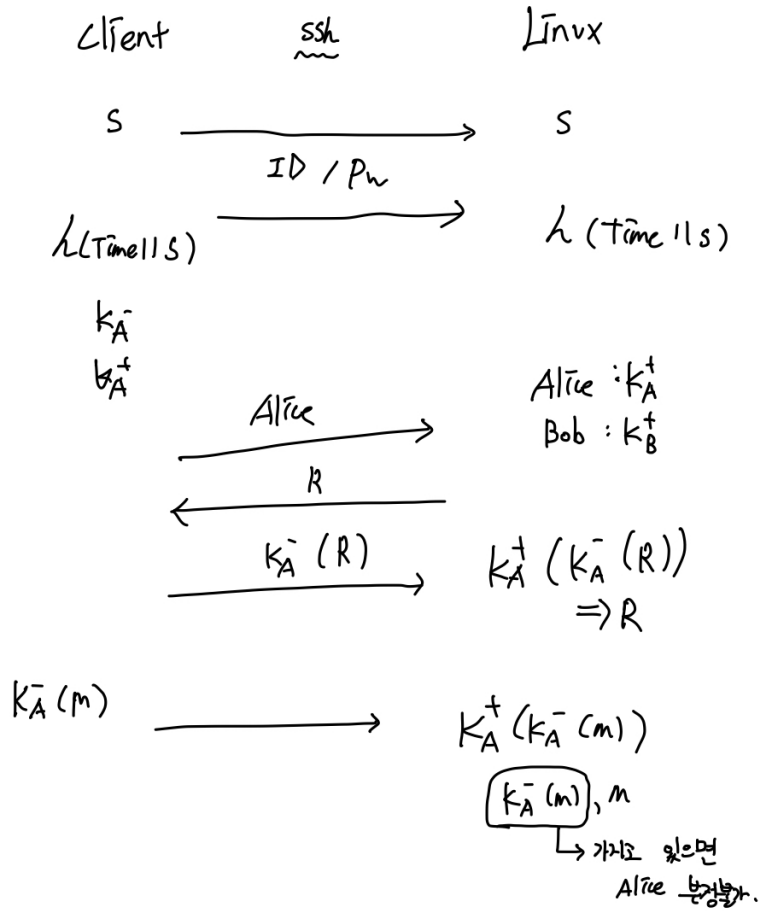
TLS와 SSL은 거의 유사.

과제

Case Study. SSL 간단한 동작. 클라이언트에서 서버 접속해서 메세지 전송하면 서버쪽 에서 클라이언트가 보낸다. 단 주의점이 ppt 데이터 이용시 SSL default port를 443으로 바꿀것. 와이어샤크 이용하기 위함. 서버쪽에서 키 생성이후 공개키 파일화. 클라쪽에서 트러스트스토어

정보보안

SSH



Applied crypto

Secret sharing algorithm

어떤 한명이 모든 비밀을 전부 가지면 위험하기 때문에 비밀을 두사람 이상, 일정 명수 이상 모였을 때만 본래 패스워드가 완벽히 복원되는 알고리즘을 말한다.

1차함수에서 두 점이 주어지면 함수가 확정된다. (한점만 알경우 무수히 많은 함수.) Y절편 기준 함수 $y = ax + b$ 가 있다고 하고 a, b 를 모른다고 하자. 이 함수의 한 점값을 Alice에게 주고 또다른 한 점을 Bob에게 준다. 이 두 식을 합치면 일차 연립방정식으로 나타낼 수 있고 a,b값을 알게 된다.

3사람 이상이서 가지게 하여 이중 2사람만 모여도 해제할 수 있는 형태로 제작 가능하며 한사람에게 2개의 값을 가지고 있도록 하여 권한을 다르게 할 수도있다.

2차함수로 표현하는 것도 가능한데 이 형태의 경우 secret이 3개로 분할 된다.

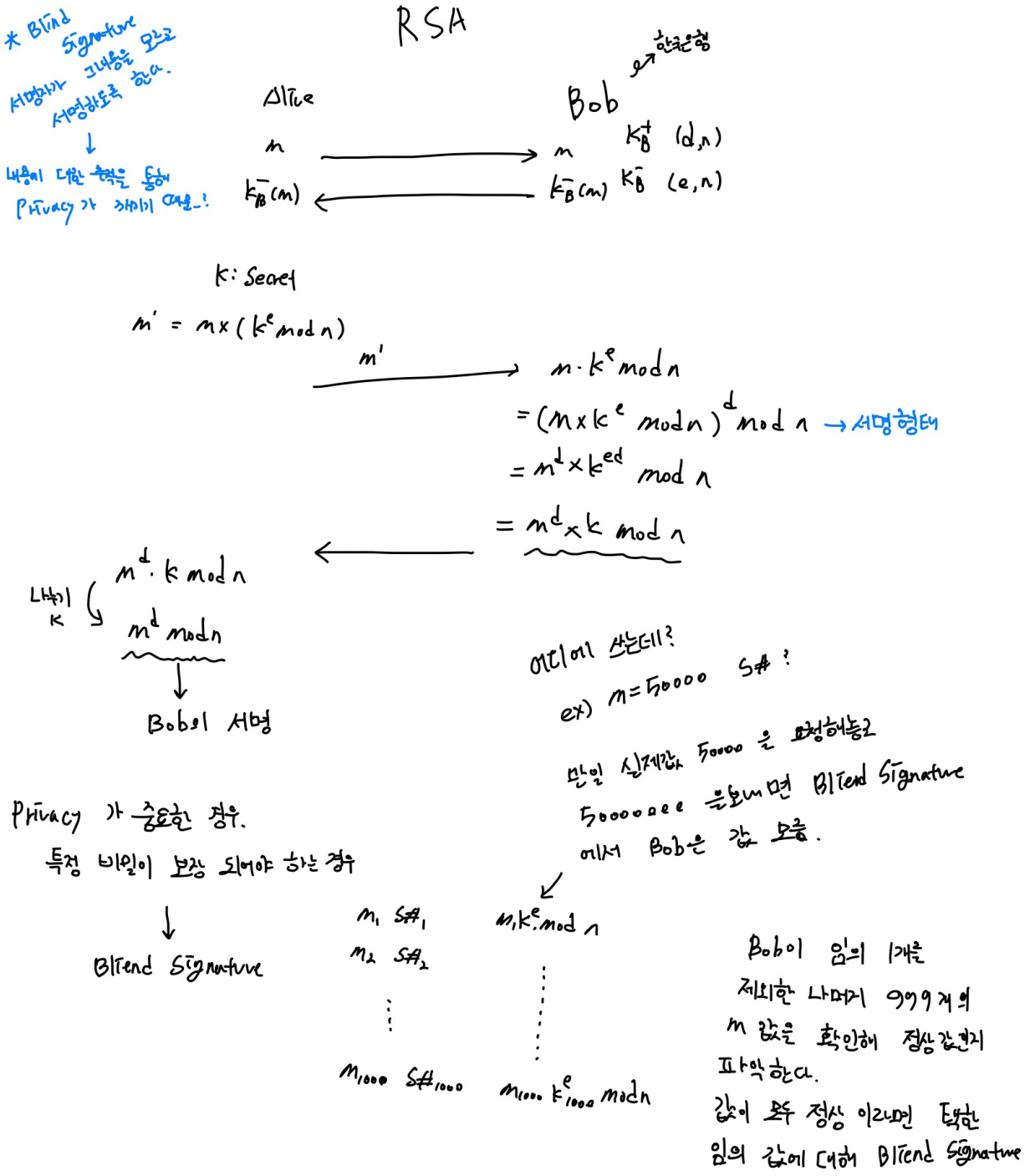
일반화

시크릿을 n개로 분할하고, K명이 모이면 secret이 생성된다고 할 때, k-1차 함수를 선언하면 된다.

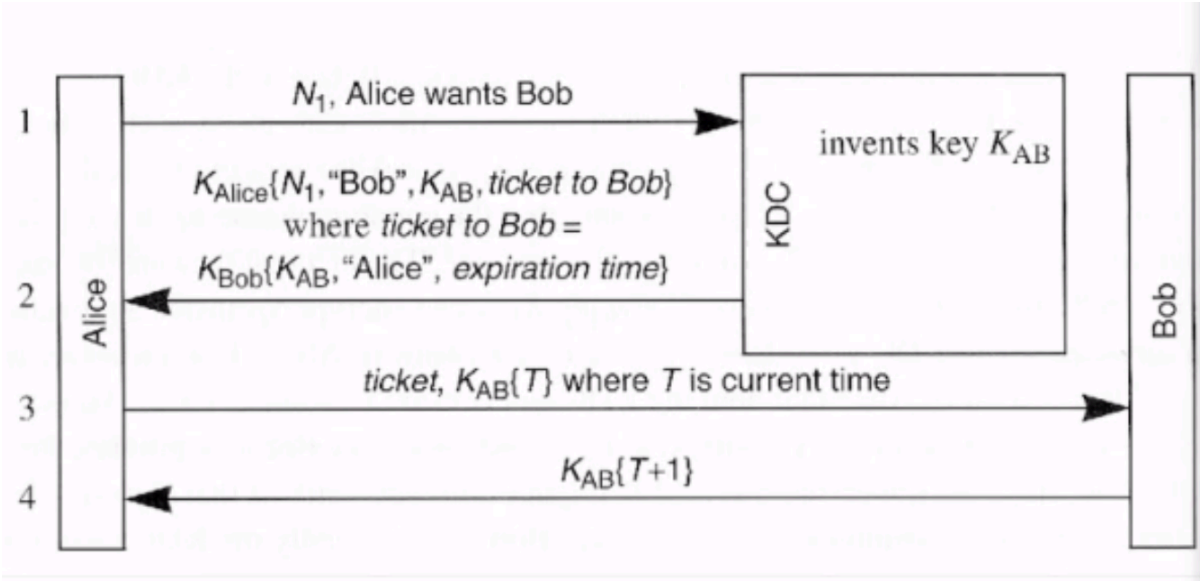
Cut and choose protocol

케이크를 잘라서 먹는다고 할때 한사람을 택해 자르게 하고 나머지가 먼저 고르게 한다. 그러면 공정하게 가능.

Blind signature



- Kerberos : 대칭키 기반 인증 시스템



앨리스가 일반 사용자. 밥이 서버. 앨리스 입장에서 서로다른 여러 서버에 접근해야 하는 경우가 많다. 이럴경우 kerveros를 이용한다. 중앙 센터에 Key distribution center. 서버에 가입을 한다. 서버도 KDC에 등록한다. KDC에는 큰 DB가 존재하여 사용자와 서버의 키를 등록해 놓는다. 그 이후 앨리스가 통신 하고자 할 때, 밥과 통신하고 싶다고 KDC에 연결한다. KDC는 랜덤넘버K_{AB} 를 생성하여 N1,Bob,Kab,티켓을 보내준다. 이 티켓은 밥만 해제할 수 있음. 앨리스는 티켓과 시간을 보내고 밥은 티켓을 열어본 뒤 확인하고 앨리스와 연결한다.