Network Security

<CH 4>

Youn Kyu Lee
Hongik University

Security Models: types of cipher

- Random functions – hash functions

  : accepts an input string of any length and outputs a string of fixed length, say *n* bits long
- Random generators – stream ciphers

  : (reverse of hash function) accepts a short input and produces a long output
- Random permutations – block ciphers

  : keyed-invertible function transfers a fixed-size input to a fixed size output and vice-versa
- Public key encryption(special kind of block cipher):

  : encrypt for anyone, but decrypt for only key owner
- Digital signatures

  : sign by only key owner, but checked by anyone

Asymmetric crypto primitives

- Crypto based on factoring
  - RSA(Rivest, Shamir, Adleman): based on the difficulty of resolving a composite number into its two large random primes (in 2020, 250 decimal digits(829-bit) factorized, in 2019, 240 decimal digits factorized)
- Crypto based on discrete logarithms
  - Diffie-Hellman key establishment
- Elliptic curve cryptography
  - based on “cryptographic” elliptic curves
  - for high performance and shorter variables
- Certification authorities(CAs)

RSA – Crypto based on factoring

- Let p and q be two large prime numbers
- Let $N = pq$ be the **modulus**
- Choose e relatively prime to $(p-1)(q-1)$
- Find d such that ed = 1 mod $(p-1)(q-1)$
- **Public key** is (N,e), **Private key** is d
- Message M is treated as a number ($\boldsymbol{M < N}$)
- To encrypt M, we compute $C = M^e \bmod N$
- To decrypt C, compute $M = C^d \bmod N$
- Recall that e and N are public
- If Trudy can factor $N=pq$, she can use e to easily find d since $ed = 1 \bmod (p-1)(q-1)$
- → **Factoring the modulus breaks RSA**

How RSA Works?

- Given $C = M^e \bmod N$ we must show

  $M = C^d \bmod N = M^{ed} \bmod N$

- We'll use **Euler's Theorem:**

  If x is relatively prime to n then $x^{\varphi(n)} = 1 \bmod n$

- Facts:

  1) $ed = 1 \bmod (p - 1)(q - 1)$
  2) By definition of "mod", $ed = k(p - 1)(q - 1) + 1$
  3) $\varphi(N) = (p - 1)(q - 1)$

- Then $ed - 1 = k(p - 1)(q - 1) = k\varphi(N)$

- Finally,

  $$M^{ed} = M^{(ed - 1) + 1} = M \cdot M^{ed - 1} = M \cdot M^{k\varphi(N)}$$

  $$= M \cdot (M^{\varphi(N)})^k \bmod N = M \cdot 1^k \bmod N = M \bmod N$$

What if M is not relatively prime to N?

Fermat's Little Theorem:

$$x^{p-1} = 1 \bmod p \text{ (for all } x \text{, all prime } p\text{)}$$

- Observation:

$a = b \bmod p\ ,\ a = b \bmod q \rightarrow a = b \bmod pq$

$$M^{ed} = M^{(ed-1)+1} = M^{k(p-1)(q-1)} \cdot M = (M^{(p-1)})^{k(q-1)} \cdot M$$

$$= 1^{k(q-1)} \cdot M = M \bmod p \qquad \text{------ (a)}$$

Similarly, we have $M^{ed} = M \bmod q$ ------ (b)

From (a) and (b), $M^{ed} = M \bmod pq = M \bmod N$

Simple RSA Example

- Pick a large prime number $p = 11$ and $q = 3$
- Then, $N = pq = 33$ and $(p\text{-}1)(q\text{-}1) = 20$
- $e = 3$ (coprime with 20)
- Find d that satisfies $ed = 1 \bmod 20 \Rightarrow d = 7$

- **Public key:** $(N, e) = (33, 3)$
- **Private key:** $d = 7$

Simple RSA Example

- **Public key:** $(N, e) = (33, 3)$
- **Private key:** $d = 7$
- Suppose message $M = 8 \quad (\text{Note } M < N)$
- Ciphertext C is computed as

  $C = M^e \bmod N = 8^3 = 512 \bmod 33 \Rightarrow 17$
- Decrypt C to recover the message M by

  $M = C^d \bmod N = 17^7 = 410{,}338{,}673 \bmod 33$

  $\Rightarrow 12{,}434{,}505 * 33 + 8 \Rightarrow 8$

Caution to RSA Usage

- Use $e = 3$ for all users (but not same N or d)
  - +Public key operations only require 2 multiplies
  - Private key operations remain expensive
  - If $M < N^{1/3}$ then $C = M^e = M^3$ and **cube root attack**
  - For any M, if C_1, C_2, C_3 sent to 3 users, cube root attack works (uses Chinese Remainder Theorem)
- Can prevent cube root attack by padding message with random bits

Modulo Exponential Example

Modulo Exponential Example

- 5^{20} = 95367431640625 = 25 mod 35

Improved method: iteration of squares

- $5^1 = 5 \pmod{35}$
- $5^2 = (5^1)^2 = 5^2 = 25 \pmod{35}$
- $5^5 = (5^2)^2 \cdot 5^1 = 25^2 \cdot 5 = 3125 = 10 \pmod{35}$
- $5^{10} = (5^5)^2 = 10^2 = 100 = 30 \pmod{35}$
- $5^{20} = (5^{10})^2 = 30^2 = 900 = 25 \pmod{35}$

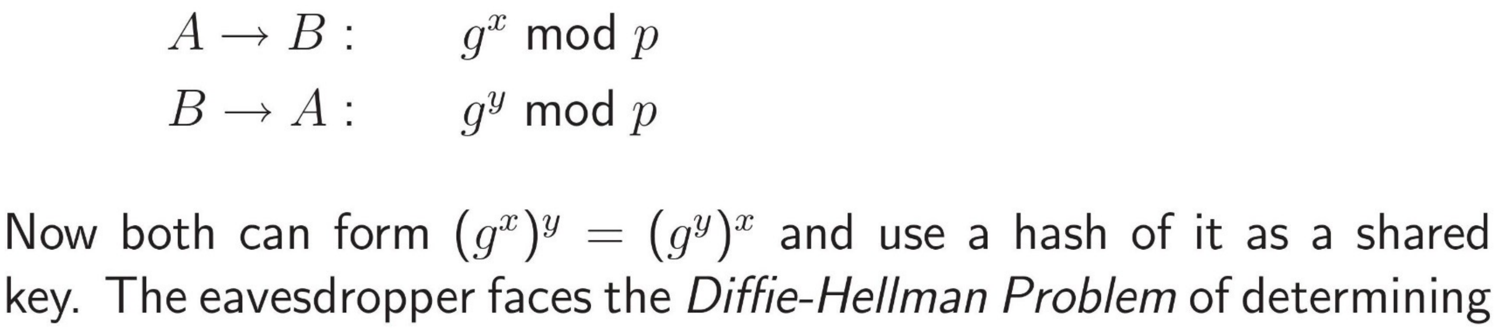
Never deal with huge numbers!

Diffie-Hellman KE: Crypto based on discrete logarithms

- Diffie-Hellman key exchange
  - used to establish a shared symmetric key

    p, g : public

    A's private key: *x*, B's private key: *y*

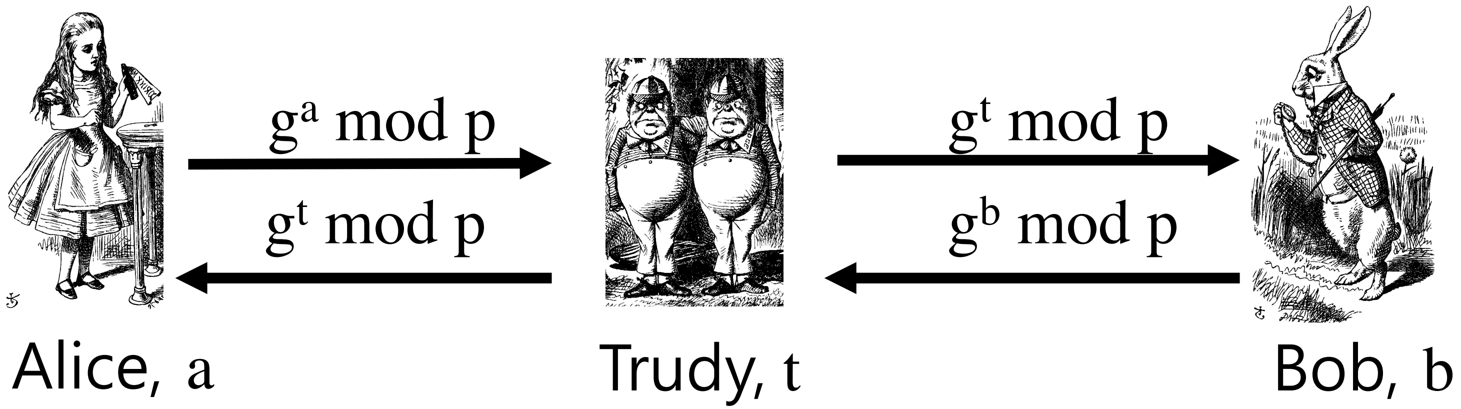


$$A \rightarrow B : \qquad g^x \text{ mod } p$$
$$B \rightarrow A : \qquad g^y \text{ mod } p$$

Now both can form $(g^x)^y = (g^y)^x$ and use a hash of it as a shared key. The eavesdropper faces the *Diffie-Hellman Problem* of determining

  - NOT for encrypting or signing

Diffie-Hellman MiM attack



g^a mod p

g^t mod p

g^t mod p

g^b mod p

Alice, a

Trudy, t

Bob, b

- How to prevent MiM attack?
  - encrypt DH exchange with symmetric key
  - encrypt DH exchange with public(encrypting) key
  - sign DH exchange with private(signing) key

ElGamal digital signature: Crypto based on discrete logarithms

- p, g are public values
- Alice's private signing key X ,
  public signature verification key $Y=g^X \mod p$
- Signing
  - message M (whose message digest is m),
    random key k
  - compute $r= g^k \mod p$
  - compute signature $s=k+mX \mod p\text{-}1$
  - M and (r, s) : given to verifier who has Y

ElGamal digital signature

- Verification: using *M* and *(r, s)*
  - compute *m* from *M* using digest function
  - check if $g^s = r \cdot Y^m$ *mod p*

    (this will be true if signature is valid

    $g^s = g^{k+mX} = g^k \cdot g^{mX} = r \cdot (g^X)^m = r \cdot Y^m$ *mod p*)
- If *M* modified, signature won't match the modified *M*
- Knowing signature won't help divulge *X*
- Without *X*, none can compute valid signature

Elliptic Curve Cryptography

- Elliptic curves are a different way to do the math in public key system
- Why use ECC? (compared to RSA)
  - can be used to sign and encrypt
  - same level of security w/ shorter keys
  - performance: signing and (signing+verifying) are faster
  - requiring less space(key storage) and less bandwidth
- But, it looks RSA will continue to be used for some time

What is an Elliptic Curve?

- An elliptic curve E is the graph of an equation of the form

$$y^2 = x^3 + ax + b$$

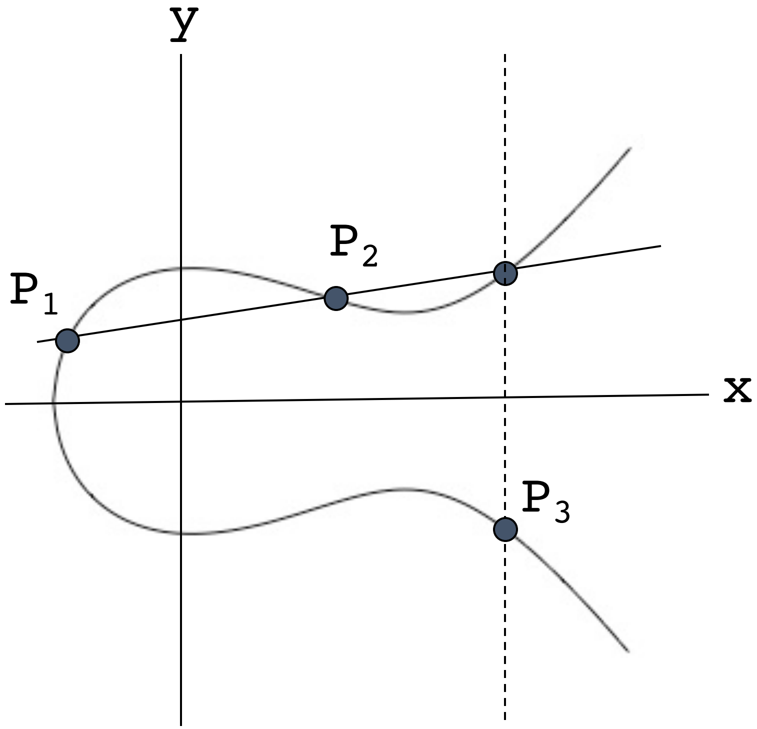
- “cryptographic” elliptic curve

$$y^2 \bmod p = (x^3 + ax + b) \bmod p$$

points allowed: integers from 0 to p-1,

also includes a “point at infinity”

Elliptic Curve Picture



- Consider elliptic curve

  $E: y^2 = x^3 - x + 1$

- If P_1 and P_2 are on E, we can define

  $P_3 = P_1 + P_2$

  as shown in picture

Points on Elliptic Curve

- Consider $y^2 = x^3 + 2x + 3 \pmod{5}$

  $x = 0 \Rightarrow y^2 = 3 \Rightarrow$ no solution (mod 5)

  $x = 1 \Rightarrow y^2 = 6 = 1 \Rightarrow y = 1,4 \pmod{5}$

  $x = 2 \Rightarrow y^2 = 15 = 0 \Rightarrow y = 0 \pmod{5}$

  $x = 3 \Rightarrow y^2 = 36 = 1 \Rightarrow y = 1,4 \pmod{5}$

  $x = 4 \Rightarrow y^2 = 75 = 0 \Rightarrow y = 0 \pmod{5}$

- Then points on the elliptic curve are

  (1,1) (1,4) (2,0) (3,1) (3,4) (4,0) and the point at infinity: ∞

Elliptic Curve Math

- Addition on: $y^2 = x^3 + ax + b \pmod{p}$

  $P_1=(x_1,y_1)$, $P_2=(x_2,y_2)$

  $P_1 + P_2 = P_3 = (x_3,y_3)$ where

  $$x_3 = m^2 - x_1 - x_2 \pmod{p}$$

  $$y_3 = m(x_1 - x_3) - y_1 \pmod{p}$$

  And $m = (y_2-y_1)*(x_2-x_1)^{-1} \bmod p$, if $P_1 \neq P_2$

  $m = (3x_1{}^2+a)*(2y_1)^{-1} \bmod p$, if $P_1 = P_2$

  Special cases: If m is infinite, $P_3 = \infty$, and $\infty + P = P$ for all P

Elliptic Curve Addition

- Consider $y^2 = x^3 + 2x + 3 \pmod 5$.
- Points on the curve are (1,1) (1,4) (2,0) (3,1) (3,4) (4,0) and ∞
- What is $(1,4) + (3,1) = P_3 = (x_3,y_3)$?

$$m = (1-4)*(3-1)^{-1} = -3*2^{-1}$$

$$= 2(3) = 6 = 1 \pmod 5$$

$$x_3 = 1 - 1 - 3 = 2 \pmod 5$$

$$y_3 = 1(1-2) - 4 = 0 \pmod 5$$

- On this curve, (1,4) + (3,1) = (2,0)

ECC Diffie-Hellman

- **Public:** Curve $y^2 = x^3 + 7x + b \pmod{37}$
  and point $(2,5) \Rightarrow b = 3$
- **Alice's private:** $d_A = 4$
- **Bob's private:** $d_B = 7$
- Alice sends Bob: $4(2,5) = (7,32)$
- Bob sends Alice: $7(2,5) = (18,35)$
- Alice computes: $4(18,35) = (22,1)$
- Bob computes: $7(7,32) = (22,1)$

ECC Signature and Verification

- Sign
  1. calculate $z = hash(m)$, m: the message to sign
  2. select a random number k
  3. calculate curve point (x_1,y_1)=$k\ G$,G is elliptic curve base point
  4. calculate $r = x_1 \bmod n$
  5. calculate $s = k^{-1}(z + rd_A) \bmod n,$ where d_A is Alice's private key,
  6. signature (r, s)

- Verification
  1. calculate $z = hash(m)$, m: the message to verity its signature
  2. calculate $u_1 = zs^{-1} \bmod n$ and $u_2 = rs^{-1} \bmod n$
  3. calculate the curve point (x_1,y_1)=$u_1 G +\ u_2 Q_A$, where $Q_A = d_A\ G$
  4. if $r = x_1 \bmod n$, the signature is valid, invalid otherwise

Uses for Public Key Crypto

- Confidentiality

- Authentication

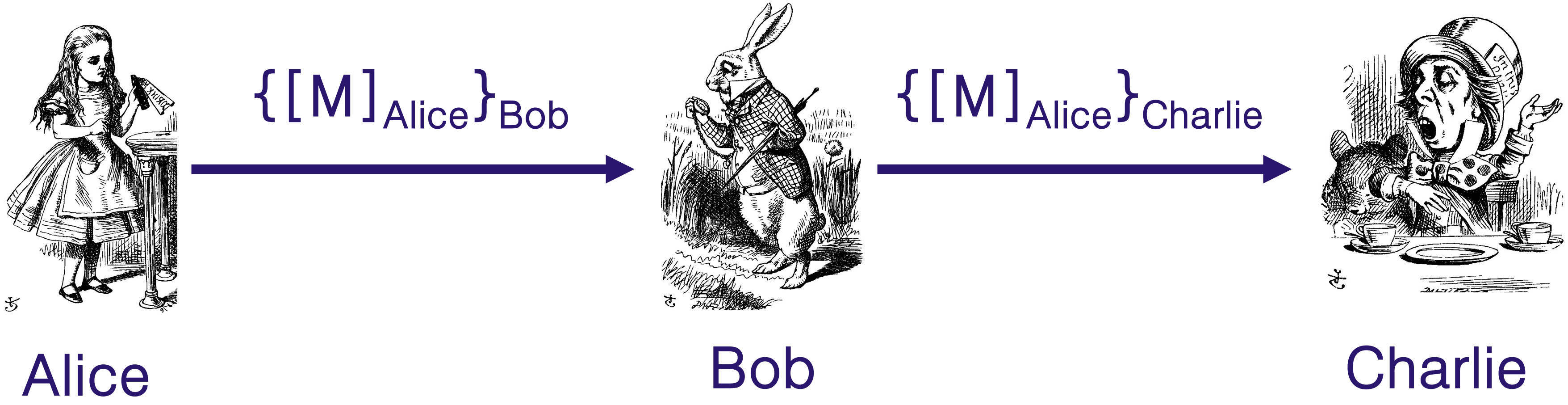
- Digital signature provides integrity and **non-repudiation**

Non-non-repudiation

- Alice orders 100 shares of stock from Bob
- Alice computes **MAC** using symmetric key
- Stock drops, Alice claims she did ***not*** order
- Can Bob prove that Alice placed the order?
- **No!** Since Bob also knows the symmetric key, he could have forged message
- **Problem:** Bob knows Alice placed the order, but he can't prove it
- Now, Alice **signs** order with her private key
- Stock drops, Alice claims she did not order
- Can Bob prove that Alice placed the order?
- **Yes!** Only someone with Alice's private key could have signed the order

Sign and Encrypt

- M = “I love you to death.”



$\{[M]_{Alice}\}_{Bob}$

$\{[M]_{Alice}\}_{Charlie}$

Alice

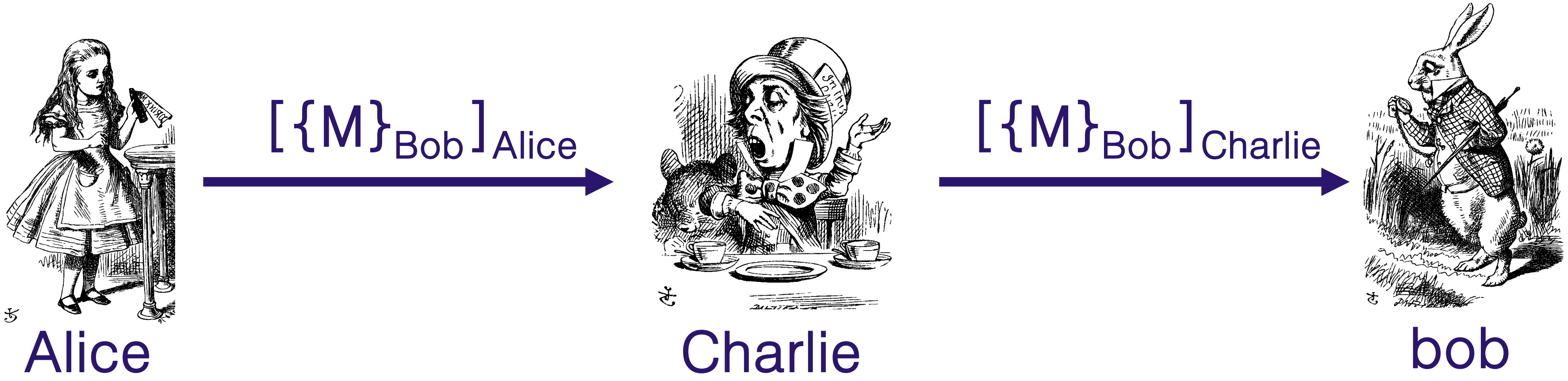
Bob

Charlie

- Problem: What is the problem?
- Answer: Charlie misunderstood the password!

Encrypt and Sign

- M = “Novel research results…”



$[\{M\}_{Bob}]_{Alice}$

$[\{M\}_{Bob}]_{Charlie}$

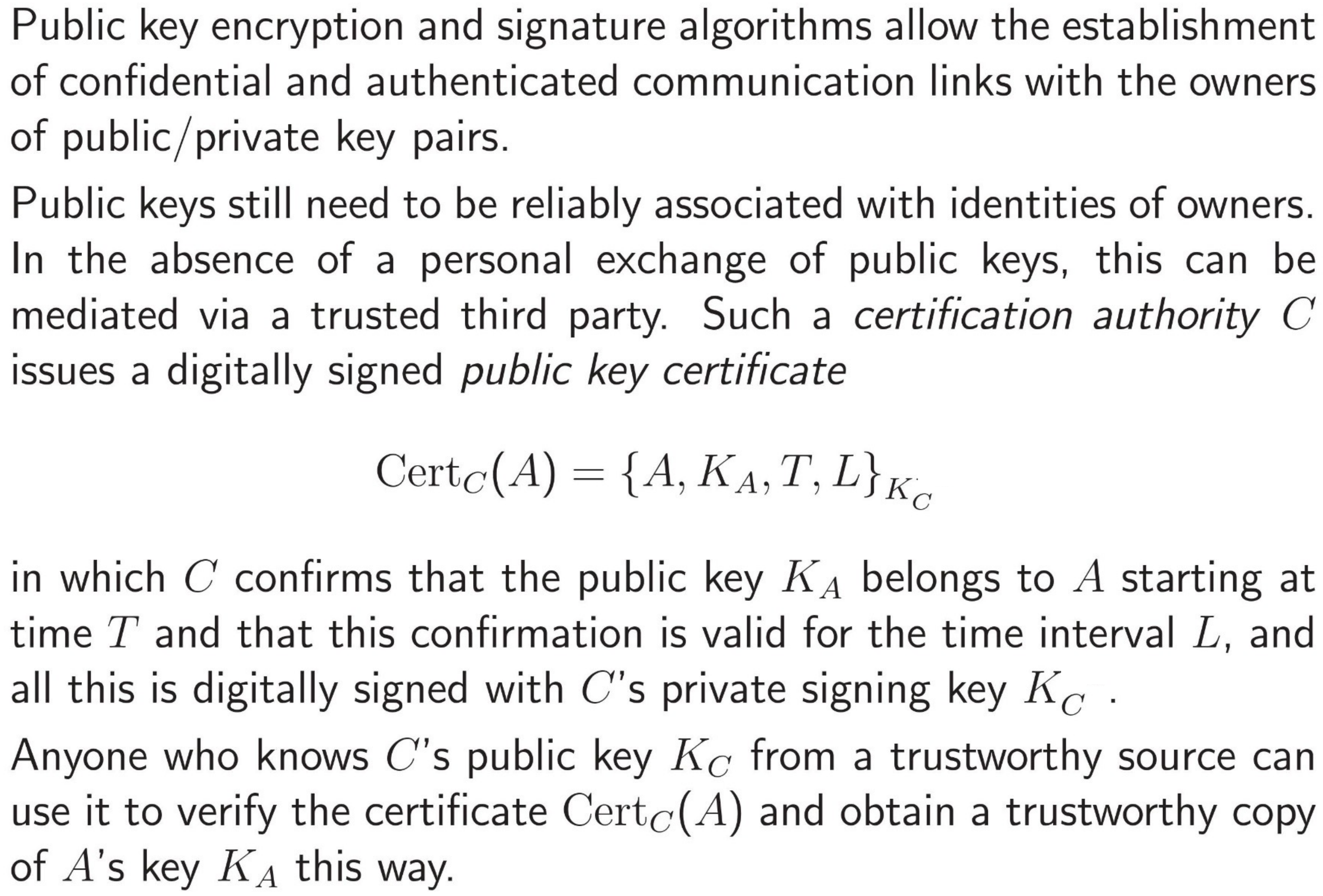
Alice

Charlie

bob

- Note that Charlie cannot decrypt M
- Problem: What is the problem?
- Answer: Bob misunderstood the password!

Public-key Infrastructure



Public key encryption and signature algorithms allow the establishment of confidential and authenticated communication links with the owners of public/private key pairs.

Public keys still need to be reliably associated with identities of owners. In the absence of a personal exchange of public keys, this can be mediated via a trusted third party. Such a *certification authority* C issues a digitally signed *public key certificate*

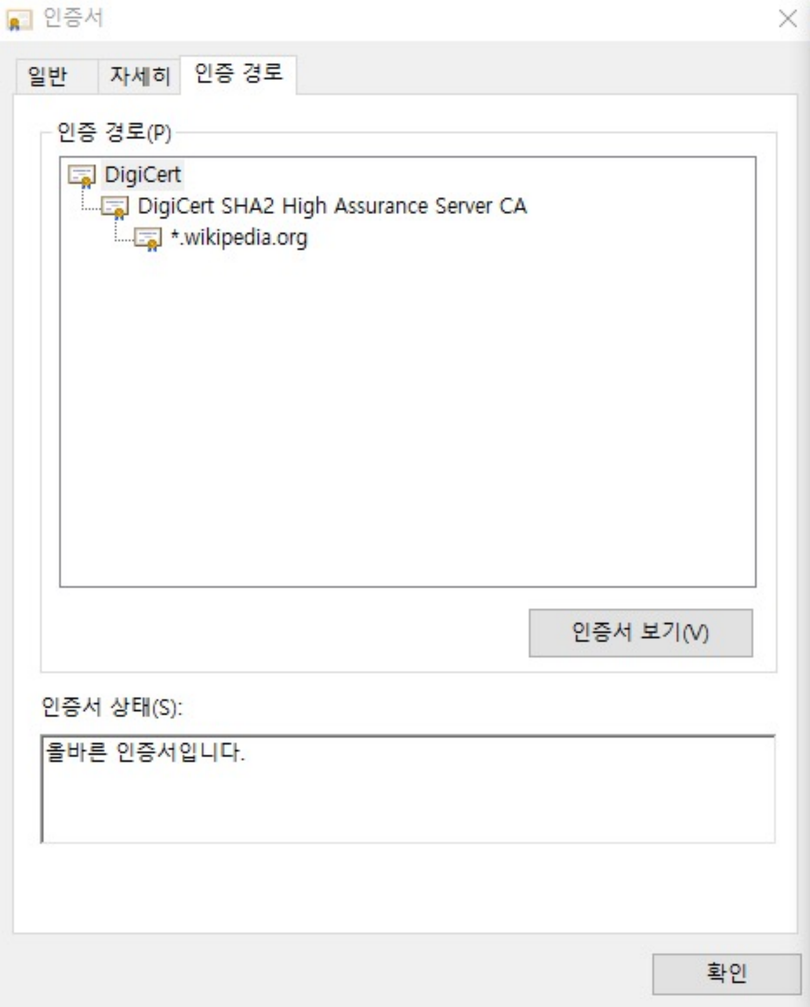
$$\mathrm{Cert}_C(A) = \{A, K_A, T, L\}_{K_C}$$

in which C confirms that the public key K_A belongs to A starting at time T and that this confirmation is valid for the time interval L, and all this is digitally signed with C's private signing key K_C .

Anyone who knows C's public key K_C from a trustworthy source can use it to verify the certificate $\mathrm{Cert}_C(A)$ and obtain a trustworthy copy of A's key K_A this way.

from Introduction to Security by Markus Kuhn
http://www.cl.cam.ac.uk/teaching/0708/IntroSec/

Public-key Infrastructure

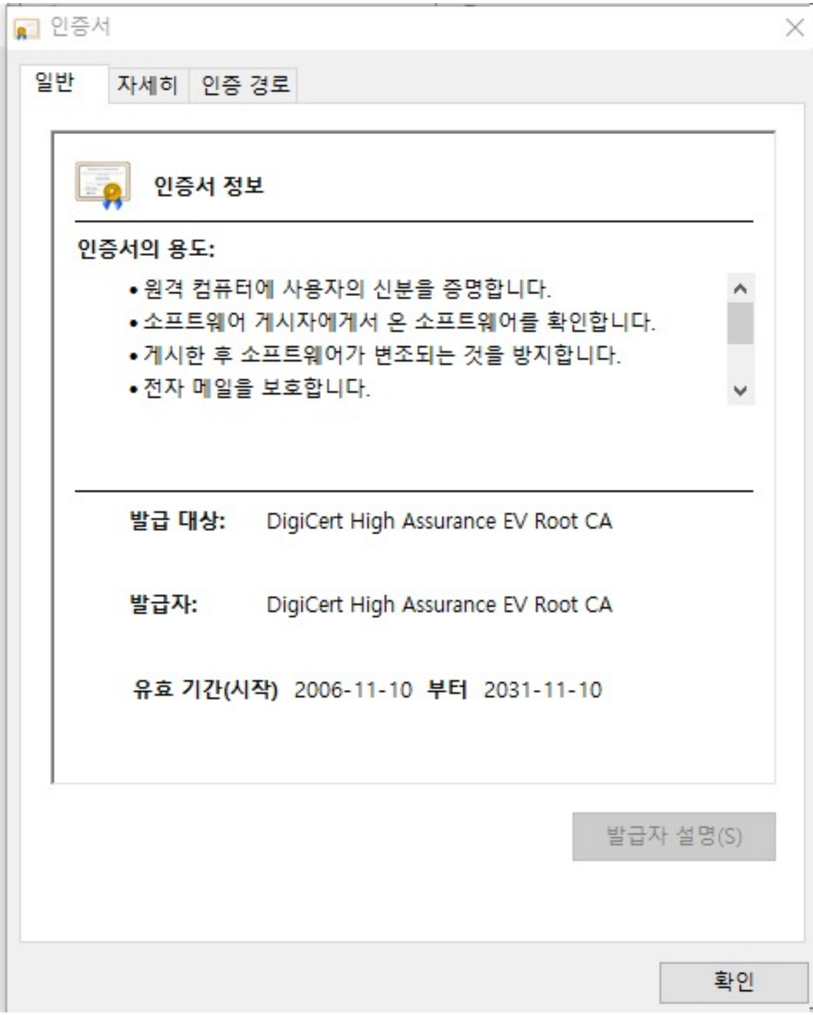


인증 경로(P)

- DigiCert
  - DigiCert SHA2 High Assurance Server CA
    - *.wikipedia.org

인증서 상태(S):

올바른 인증서입니다.



인증서 정보

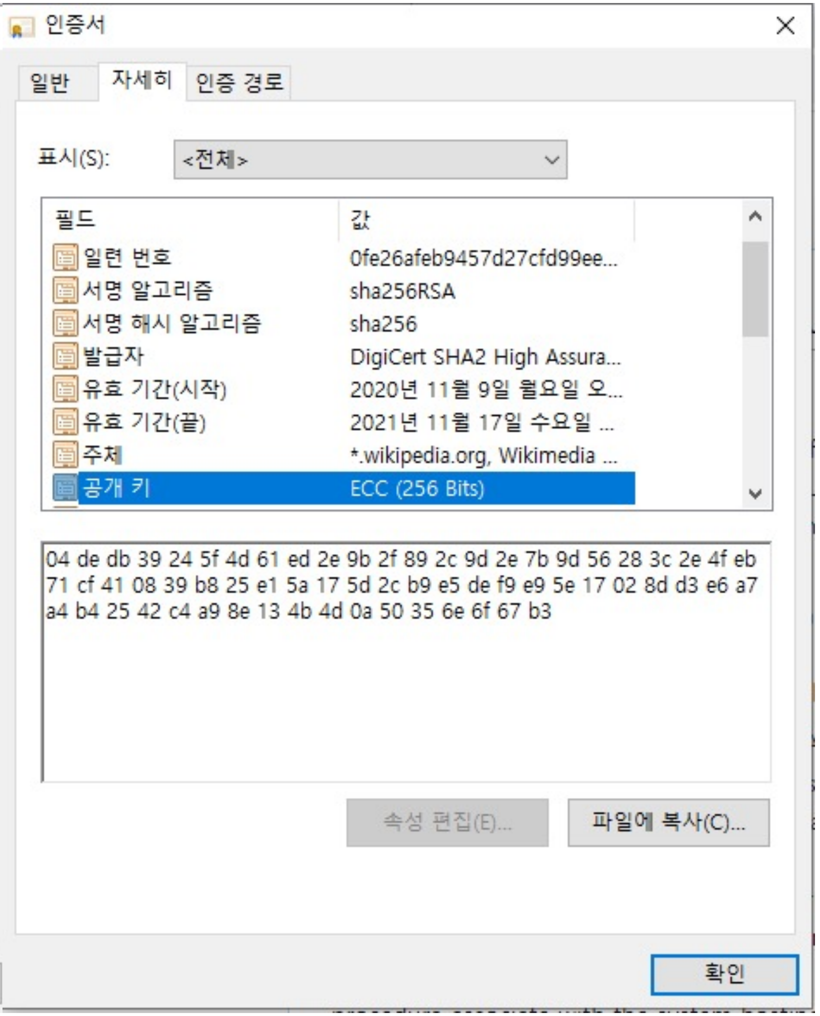
인증서의 용도:

- 원격 컴퓨터에 사용자의 신분을 증명합니다.
- 소프트웨어 게시자에게서 온 소프트웨어를 확인합니다.
- 게시한 후 소프트웨어가 변조되는 것을 방지합니다.
- 전자 메일을 보호합니다.

발급 대상: DigiCert High Assurance EV Root CA

발급자: DigiCert High Assurance EV Root CA

유효 기간(시작) 2006-11-10 **부터** 2031-11-10

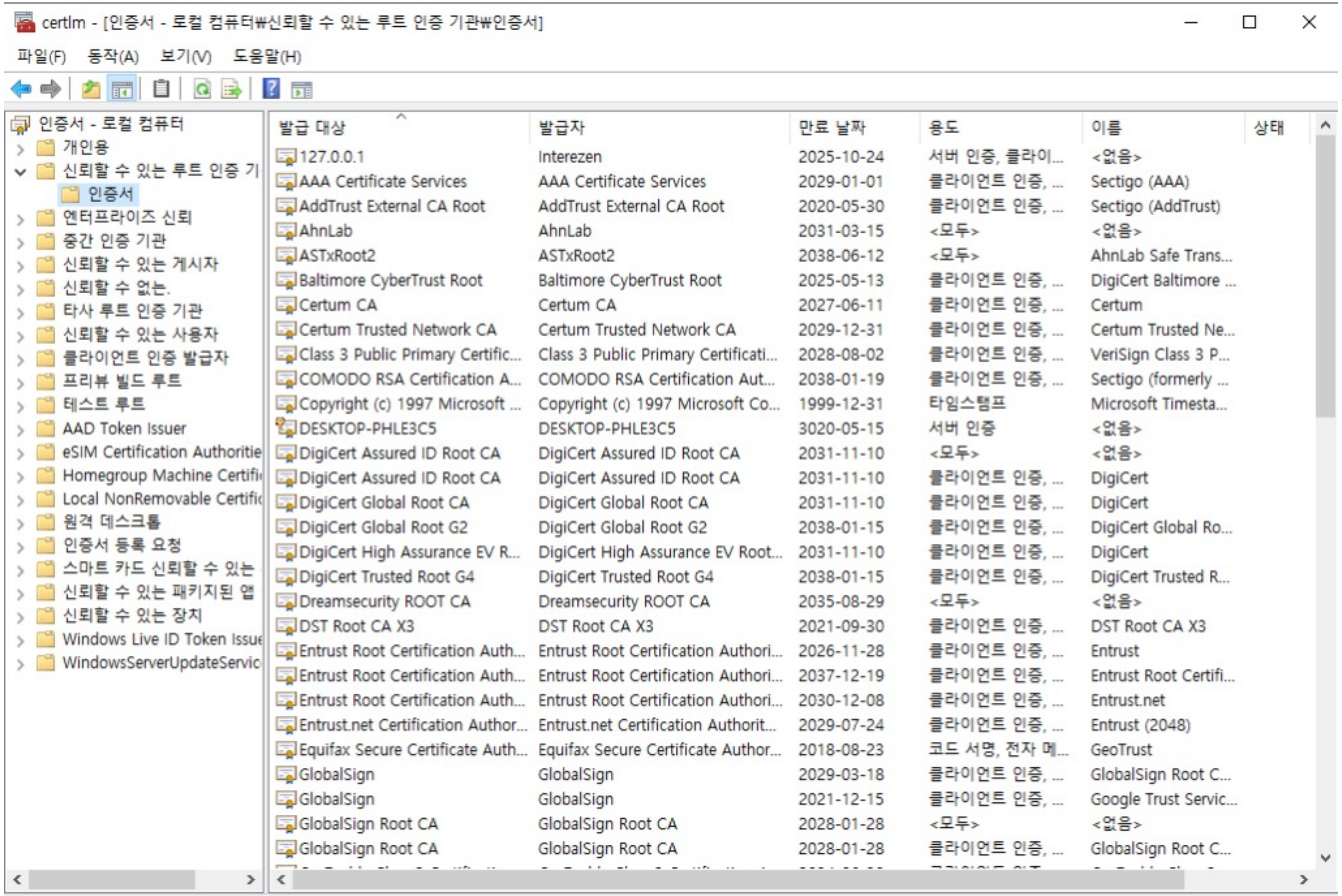


표시(S): <전체>

필드	값
일련 번호	0fe26afeb9457d27cfd99ee...
서명 알고리즘	sha256RSA
서명 해시 알고리즘	sha256
발급자	DigiCert SHA2 High Assura...
유효 기간(시작)	2020년 11월 9일 월요일 오...
유효 기간(끝)	2021년 11월 17일 수요일 ...
주체	*.wikipedia.org, Wikimedia ...
공개 키	ECC (256 Bits)

```
04 de db 39 24 5f 4d 61 ed 2e 9b 2f 89 2c 9d 2e 7b 9d 56 28 3c 2e 4f eb
71 cf 41 08 39 b8 25 e1 5a 17 5d 2c b9 e5 de f9 e9 5e 17 02 8d d3 e6 a7
a4 b4 25 42 c4 a9 8e 13 4b 4d 0a 50 35 6e 6f 67 b3
```

Public-key Infrastructure



발급 대상	발급자	만료 날짜	용도	이름	상태
127.0.0.1	Interezen	2025-10-24	서버 인증, 클라이...	<없음>	
AAA Certificate Services	AAA Certificate Services	2029-01-01	클라이언트 인증, ...	Sectigo (AAA)	
AddTrust External CA Root	AddTrust External CA Root	2020-05-30	클라이언트 인증, ...	Sectigo (AddTrust)	
AhnLab	AhnLab	2031-03-15	<모두>	<없음>	
ASTxRoot2	ASTxRoot2	2038-06-12	<모두>	AhnLab Safe Trans...	
Baltimore CyberTrust Root	Baltimore CyberTrust Root	2025-05-13	클라이언트 인증, ...	DigiCert Baltimore ...	
Certum CA	Certum CA	2027-06-11	클라이언트 인증, ...	Certum	
Certum Trusted Network CA	Certum Trusted Network CA	2029-12-31	클라이언트 인증, ...	Certum Trusted Ne...	
Class 3 Public Primary Certific...	Class 3 Public Primary Certificati...	2028-08-02	클라이언트 인증, ...	VeriSign Class 3 P...	
COMODO RSA Certification A...	COMODO RSA Certification Aut...	2038-01-19	클라이언트 인증, ...	Sectigo (formerly ...	
Copyright (c) 1997 Microsoft ...	Copyright (c) 1997 Microsoft Co...	1999-12-31	타임스탬프	Microsoft Timesta...	
DESKTOP-PHLE3C5	DESKTOP-PHLE3C5	3020-05-15	서버 인증	<없음>	
DigiCert Assured ID Root CA	DigiCert Assured ID Root CA	2031-11-10	<모두>	<없음>	
DigiCert Assured ID Root CA	DigiCert Assured ID Root CA	2031-11-10	클라이언트 인증, ...	DigiCert	
DigiCert Global Root CA	DigiCert Global Root CA	2031-11-10	클라이언트 인증, ...	DigiCert	
DigiCert Global Root G2	DigiCert Global Root G2	2038-01-15	클라이언트 인증, ...	DigiCert Global Ro...	
DigiCert High Assurance EV R...	DigiCert High Assurance EV Root...	2031-11-10	클라이언트 인증, ...	DigiCert	
DigiCert Trusted Root G4	DigiCert Trusted Root G4	2038-01-15	클라이언트 인증, ...	DigiCert Trusted R...	
Dreamsecurity ROOT CA	Dreamsecurity ROOT CA	2035-08-29	<모두>	<없음>	
DST Root CA X3	DST Root CA X3	2021-09-30	클라이언트 인증, ...	DST Root CA X3	
Entrust Root Certification Auth...	Entrust Root Certification Authori...	2026-11-28	클라이언트 인증, ...	Entrust	
Entrust Root Certification Auth...	Entrust Root Certification Authori...	2037-12-19	클라이언트 인증, ...	Entrust Root Certifi...	
Entrust Root Certification Auth...	Entrust Root Certification Authori...	2030-12-08	클라이언트 인증, ...	Entrust.net	
Entrust.net Certification Author...	Entrust.net Certification Authorit...	2029-07-24	클라이언트 인증, ...	Entrust (2048)	
Equifax Secure Certificate Auth...	Equifax Secure Certificate Author...	2018-08-23	코드 서명, 전자 메...	GeoTrust	
GlobalSign	GlobalSign	2029-03-18	클라이언트 인증, ...	GlobalSign Root C...	
GlobalSign	GlobalSign	2021-12-15	클라이언트 인증, ...	Google Trust Servic...	
GlobalSign Root CA	GlobalSign Root CA	2028-01-28	<모두>	<없음>	
GlobalSign Root CA	GlobalSign Root CA	2028-01-28	클라이언트 인증, ...	GlobalSign Root C...	

PKI Trust Models

- Monopoly model
  - One universally trusted organization is the CA for the known universe
  - Big problems if CA is ever compromised
  - Who will act as CA???
    - System is useless if you don’t trust the CA!
- Oligarchy
  - Multiple trusted CAs
  - This is approach used in browsers today
  - Browser may have 100 or more certificates, just to verify certificates!
  - User can decide which CAs to trust

PKI Trust Models

- Anarchy model
  - Everyone is a CA
  - Users must decide who to trust
  - This approach used in PGP: “Web of trust”
  - Why is it anarchy?

    “Suppose a certificate is signed by Frank and you don’t know Frank, but you do trust Bob and Bob says Alice is trustworthy and Alice vouches for Frank. Should you accept the certificate?”

- Many other PKI issues …

Q & A

aiclasshongik@gmail.com