그리드 보안 기술

2016. 2. 3

Contents

01 개요

02 기초암호

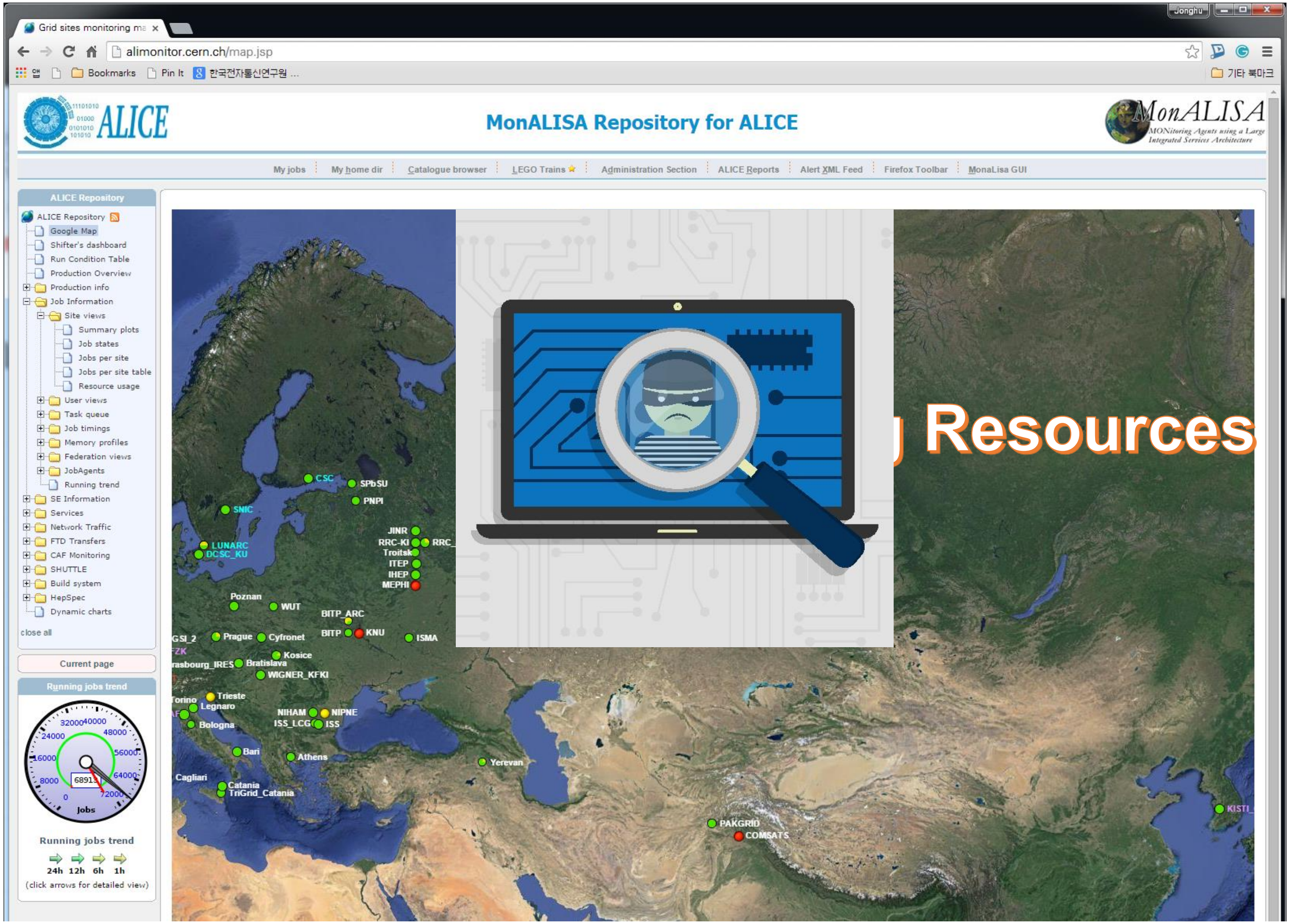
03 X.509 PKI

04 SSL/TLS

05 결론

1. 개요

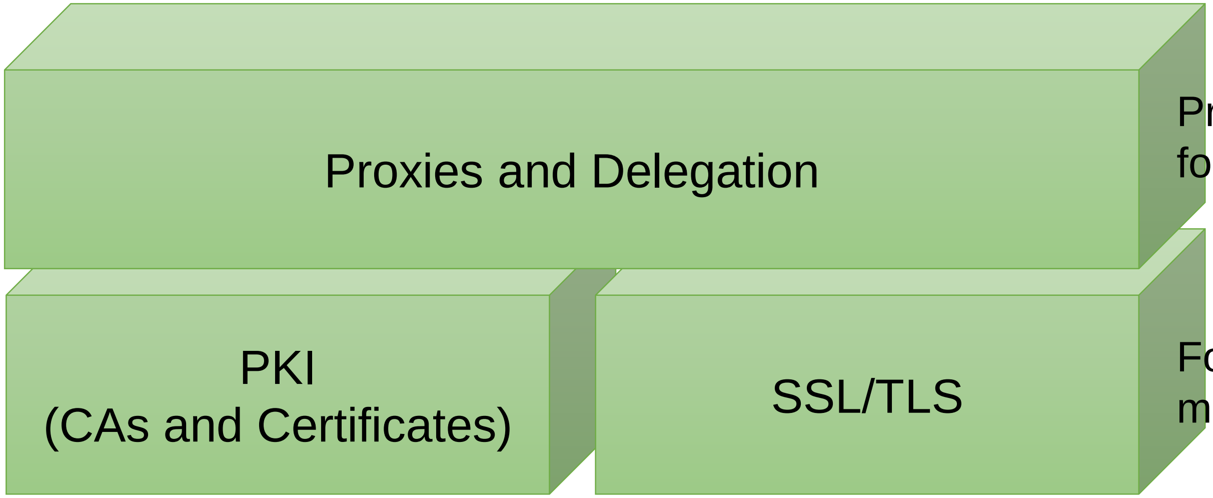
What is Security Threat Grid Computing?



Characteristics of Grid Computing Environment

- Large & dynamic user population and resources pool
- Dynamic resource acquisition and release
- Dynamic creation and destruction of a variety of network connection
- An individual user will be associated with different local name spaces, credentials, or accounts at different sites

e.g. Security Architecture in Globus



Proxies and Delegation for Secure SSO

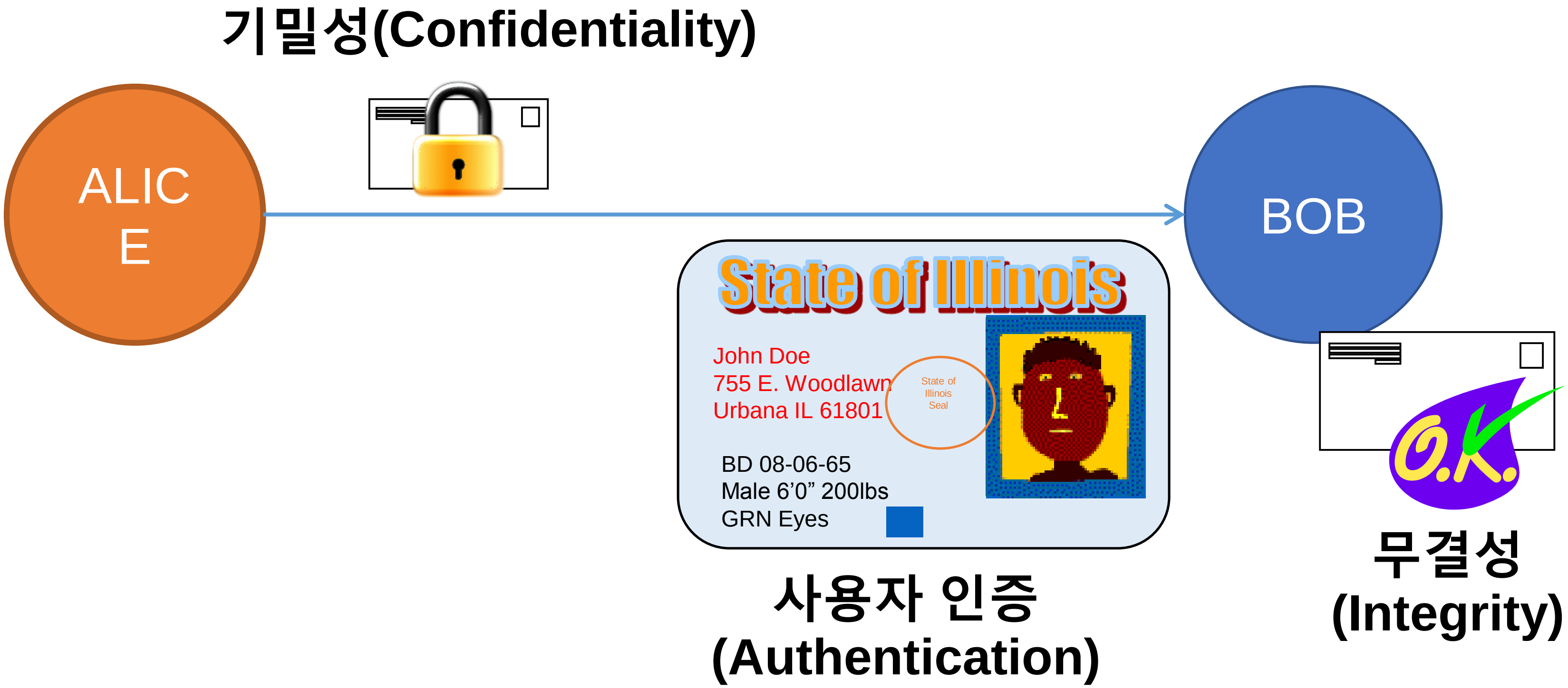
For Authentication and message Protection

- SSO: Single Sign-On
- PKI: Public Key Infrastructure
- CA: Certification Authority
- SSL: Secure Socket Layer
- TLS: Transport Layer Security

In This Class, we are going to..

understand mechanisms behind security process

보안의 목표

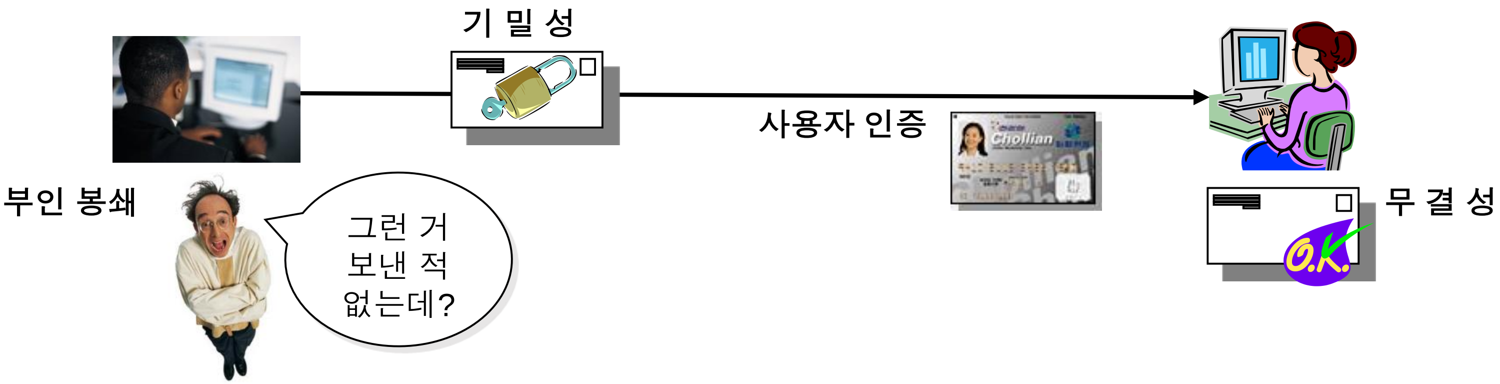


기밀성(Confidentiality)

사용자 인증
(Authentication)

무결성
(Integrity)

보안의 목표



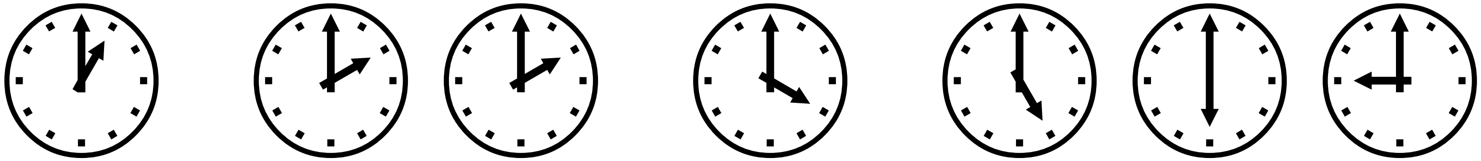
기 밀 성

사용자 인증

무 결 성

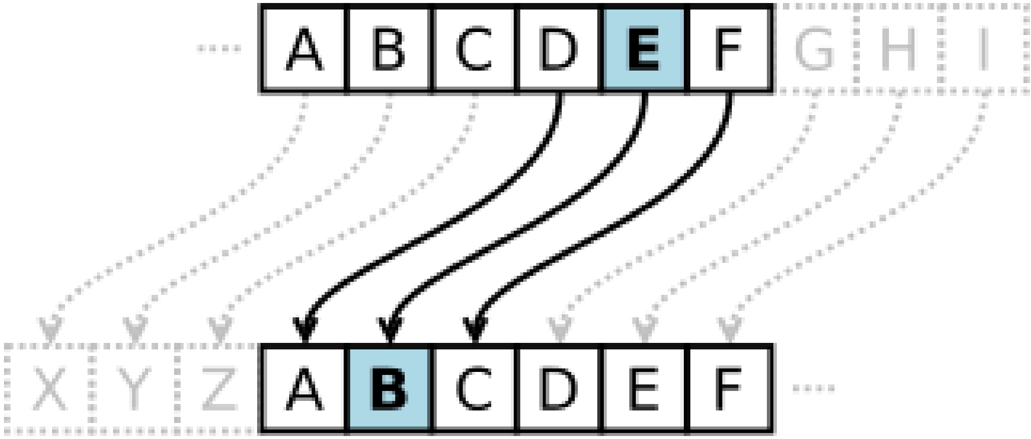
부인 봉쇄

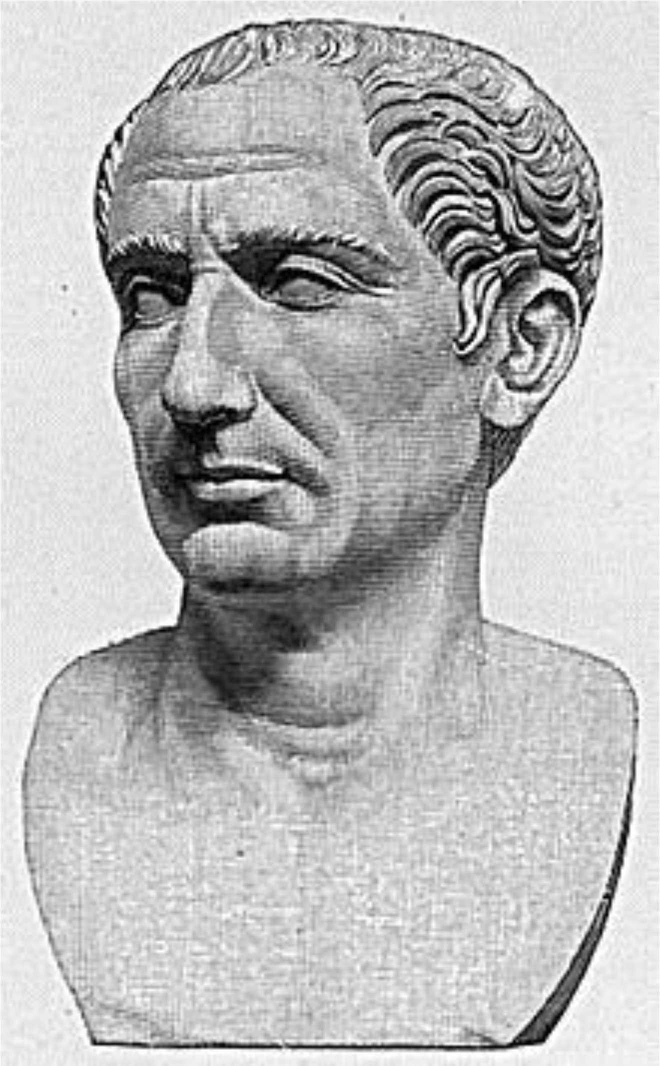
그런 거
보낸 적
없는데?



2. 기초 암호

> *If he had anything confidential to say, he wrote it in cipher, that is, by so changing the order of the letters of the alphabet, that not a word could be made out. If anyone wishes to decipher these, and get at their meaning, he must substitute the fourth letter of the alphabet, namely D, for A, and so with the others.*
>
> —*Suetonius*, *Life of Julius Caesar*



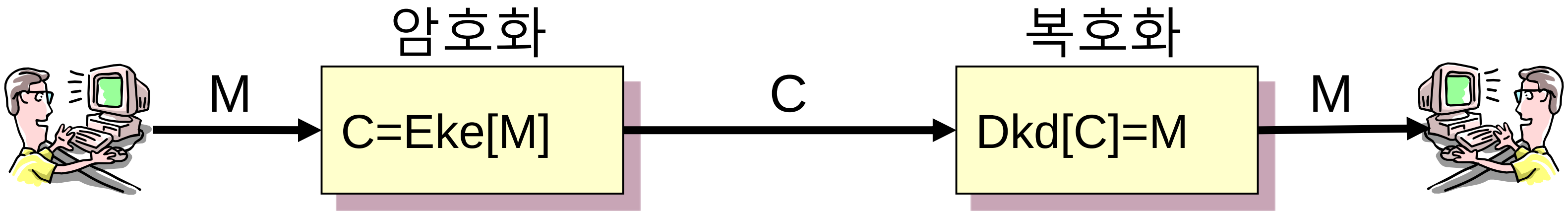


LOVE →(Shift 4) ILSB

평문 암호 알고리즘 암호키 암호문

관용 암호 알고리즘

Conventional Cryptography
Symmetric Cryptography
One-key Cryptography



M: 평문

E: 암호화 알고리즘

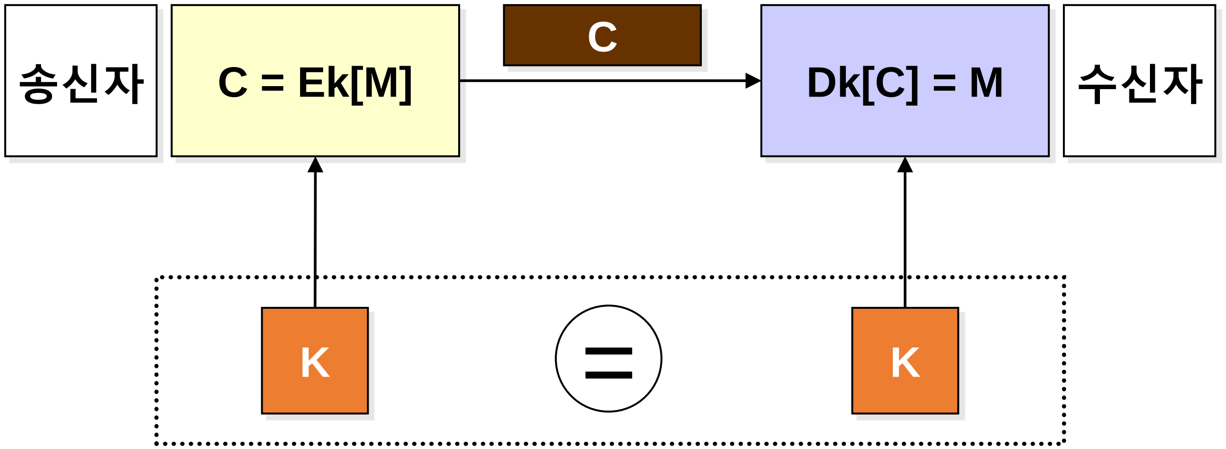
C: 암호문

Ke: 암호화 키

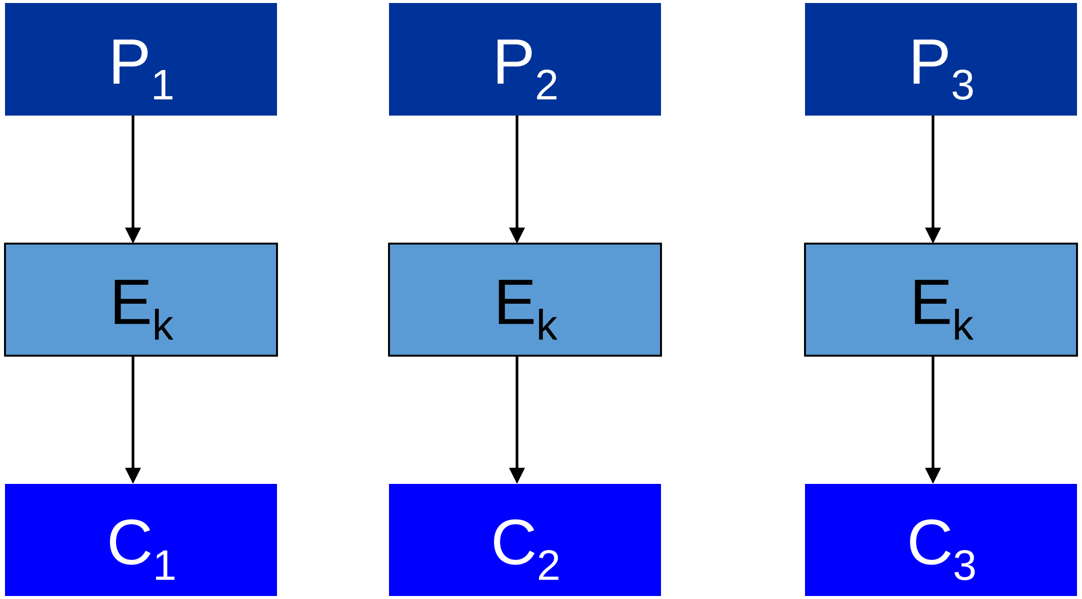
D: 복호화 알고리즘

Kd: 복호화 키

- Ke = kd
- 송신자와 수신자 사이의 공통의 키
- AES, DES, Skipjack, IDEA, FEAL, LOKI, GOST, SEED

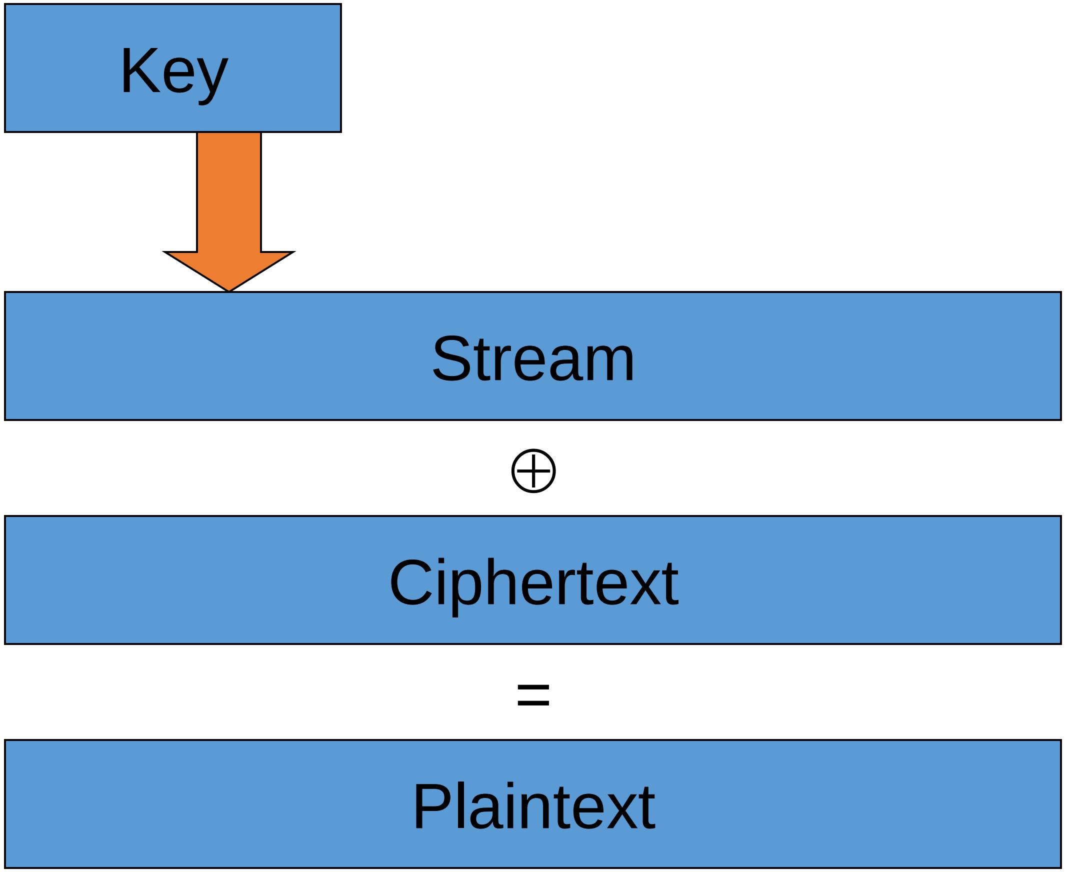


Block Cipher



encrypt each plaintext block separately

Stream Cipher



Key

Stream

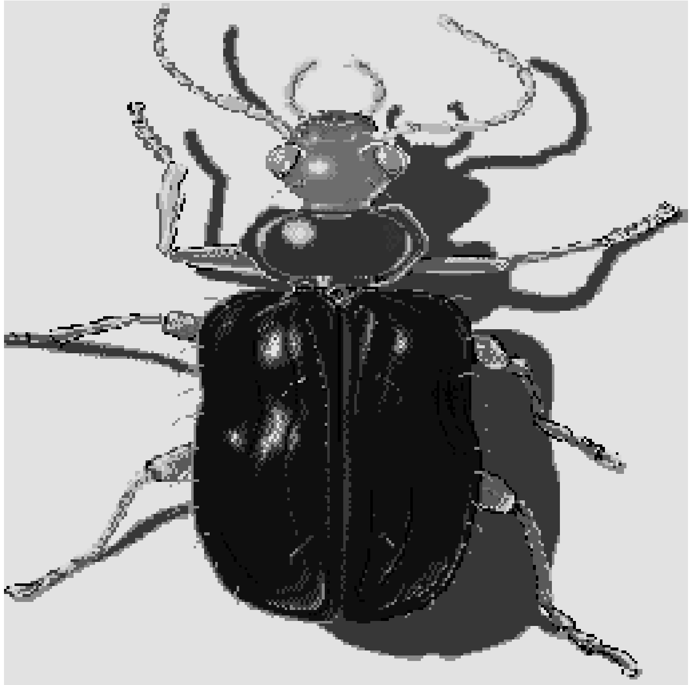
$\oplus$

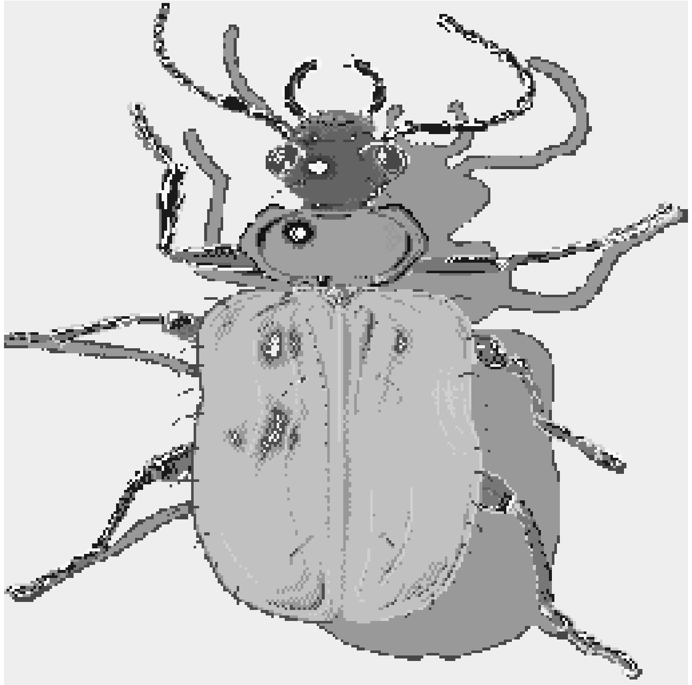
Ciphertext

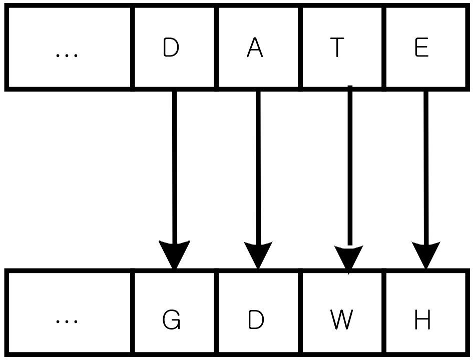
=

Plaintext

블록 암호 예제 – 시저 암호



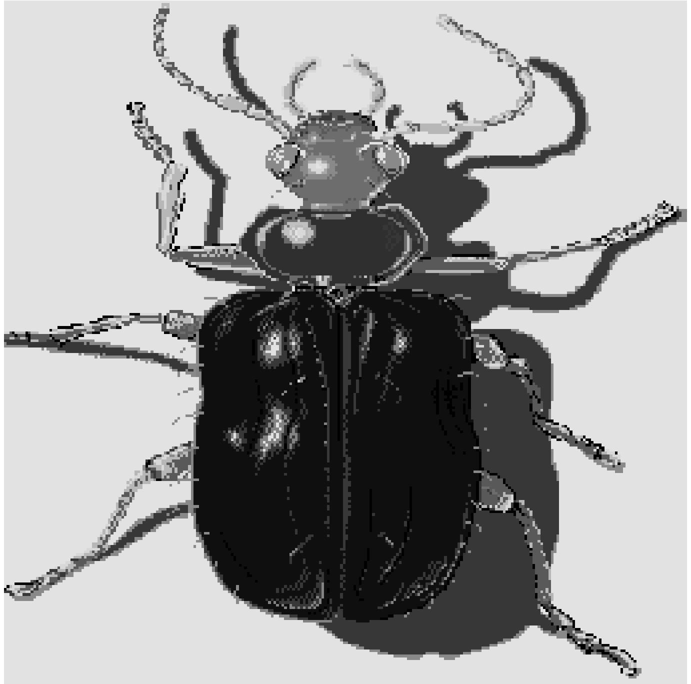


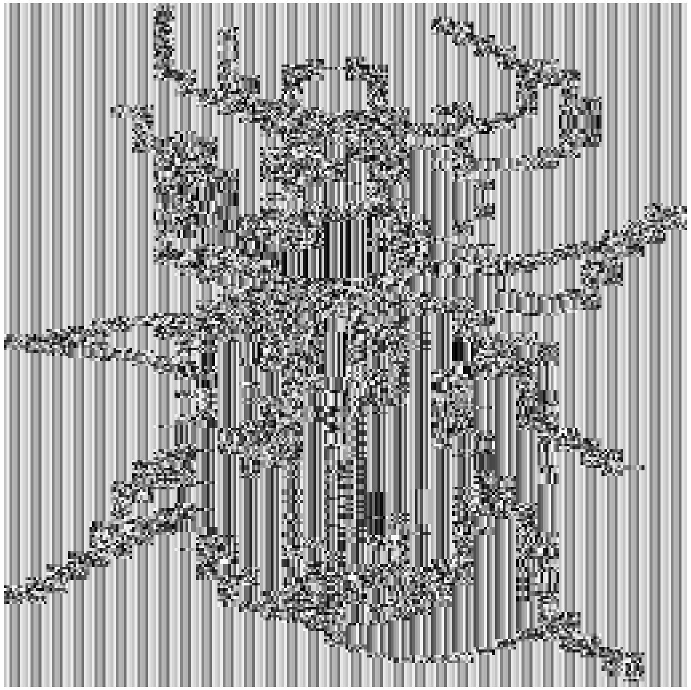


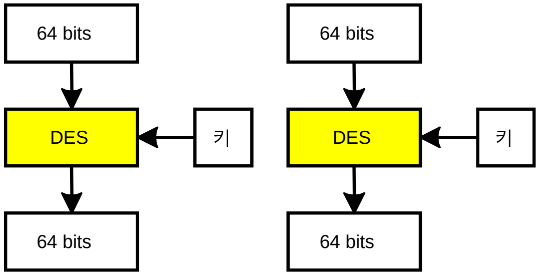
...	D	A	T	E
...	G	D	W	H

각 바이트를 치환표에
의하여 암호화

블록 암호 예제 - Electronic Code Book

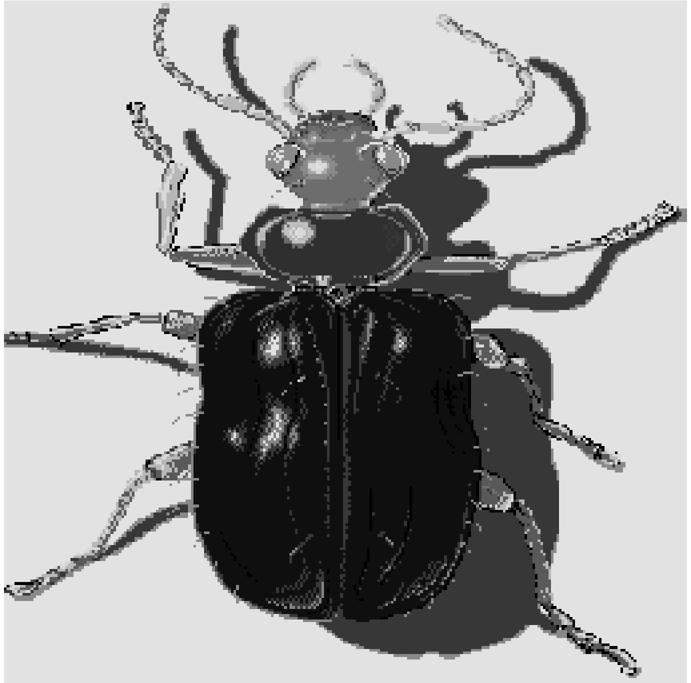


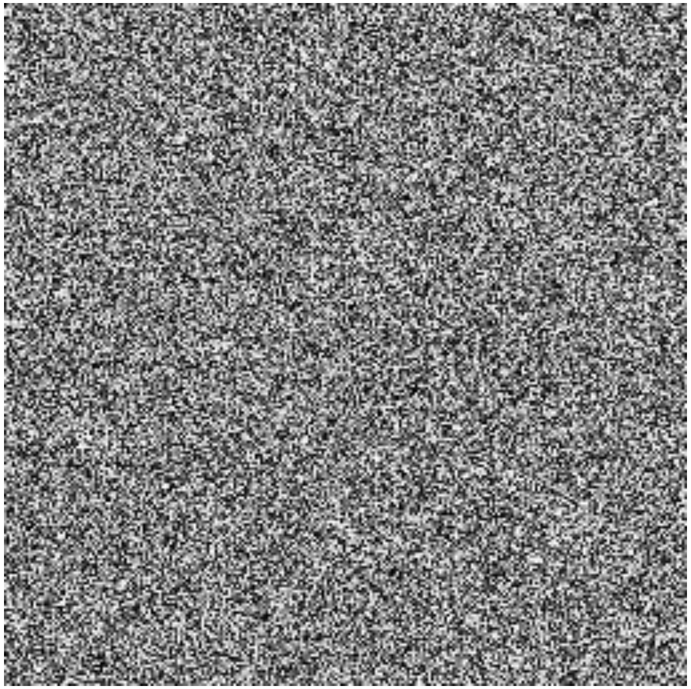


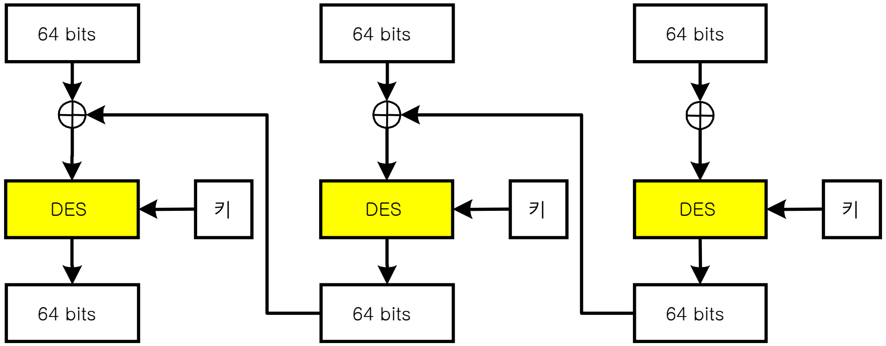


64비트 블록으로 나누어
각각을 암호화

블록 암호 예제 – Cipher Block Chaining

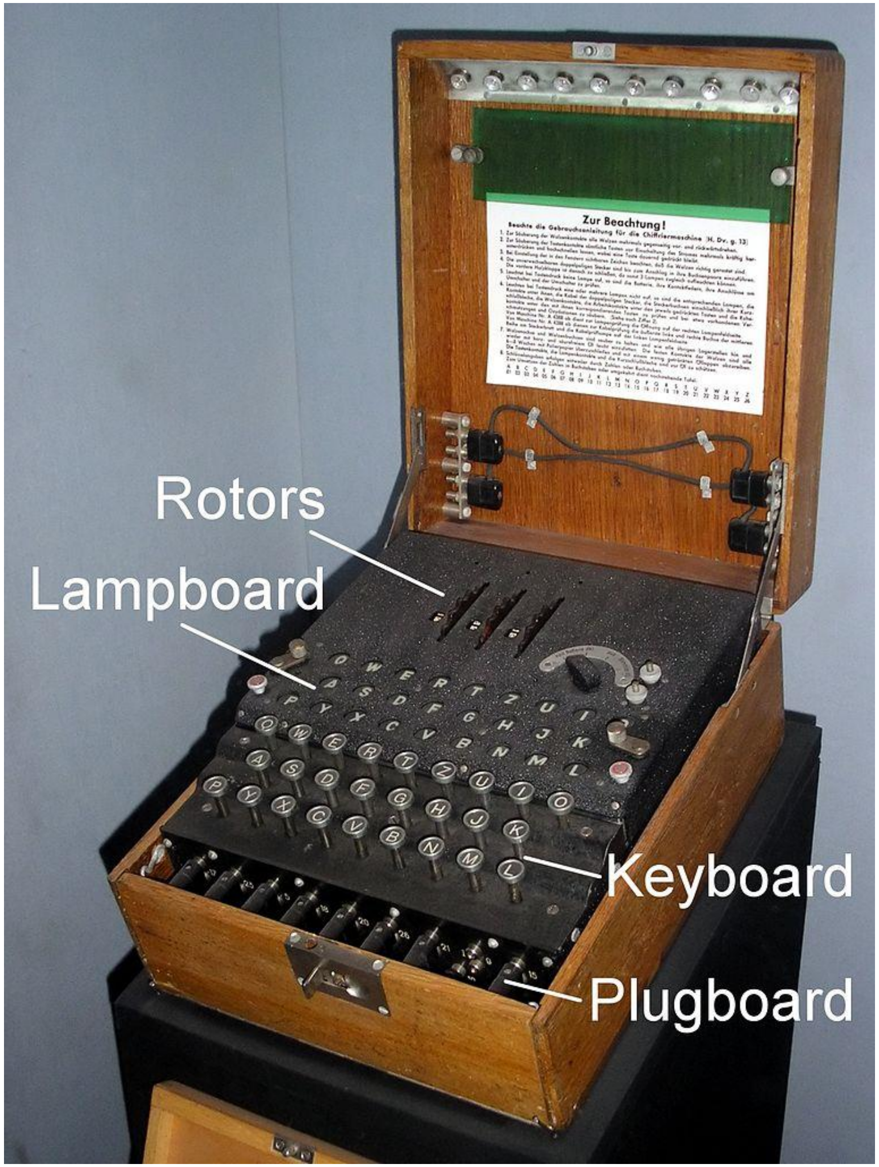






전단계의 암호문 블록이
다음 단계의 입력에 관여함

Enigma (2차 세계대전에서 독일군 사용)



DES – Data Encryption Standard

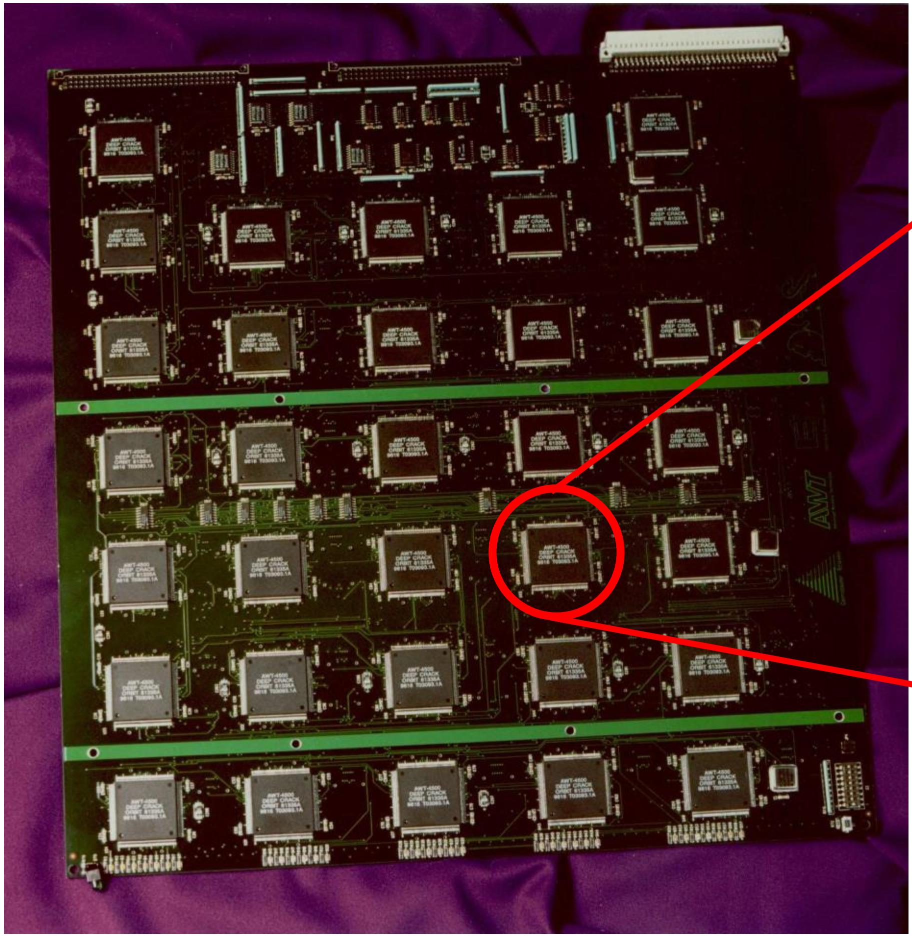
- Designed by IBM, with modifications proposed by the National Security Agency
- US national standard from 1977 to 2001
- De facto standard
- Block size 64 bits;
- Key size 56 bits
- 16-rounds
- Designed mostly for hardware implementations
- Considered insecure now
  - vulnerable to brute-force attacks

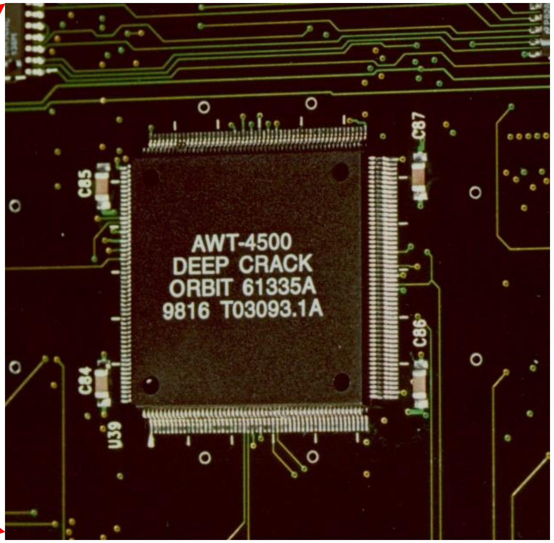
Attacking Block Ciphers

- Types of attacks to consider
  - known plaintext: given several pairs of plaintexts and ciphertexts, recover the key (or decrypt another block encrypted under the same key)
  - how would chosen plaintext and chosen ciphertext work?
- Standard attacks
  - exhaustive key search
  - dictionary attack
  - differential cryptanalysis, linear cryptanalysis
- Side channel attacks.

DES's main vulnerability is short key size.

DES 해독기

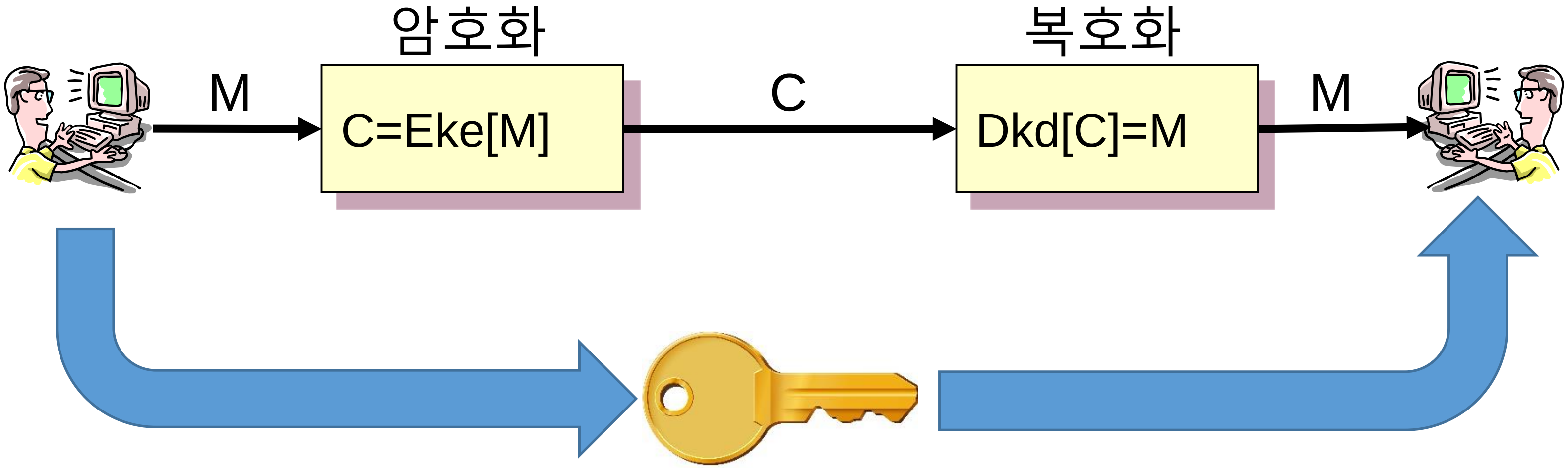

AWT


AWT-4500
DEEP CRACK
ORBIT 61335A
9816 T03093.1A
C87
C85
C86
C84
U39

AES – Advanced Encryption Standard

- In 1997, NIST made a formal call for algorithms stipulating that the AES would specify an unclassified, publicly disclosed encryption algorithm, available royalty-free, worldwide.
- Goal: replace DES for both government and private-sector encryption.
- The algorithm must implement symmetric key cryptography as a block cipher and (at a minimum) support block sizes of 128-bits and key sizes of 128-, 192-, and 256-bits.
- In 1998, NIST selected 15 AES candidate algorithms.
- On October 2, 2000, NIST selected **Rijndael** (invented by Joan Daemen and Vincent Rijmen) to as the AES.

관용키 암호 알고리즘의 문제점



암호화

복호화

M

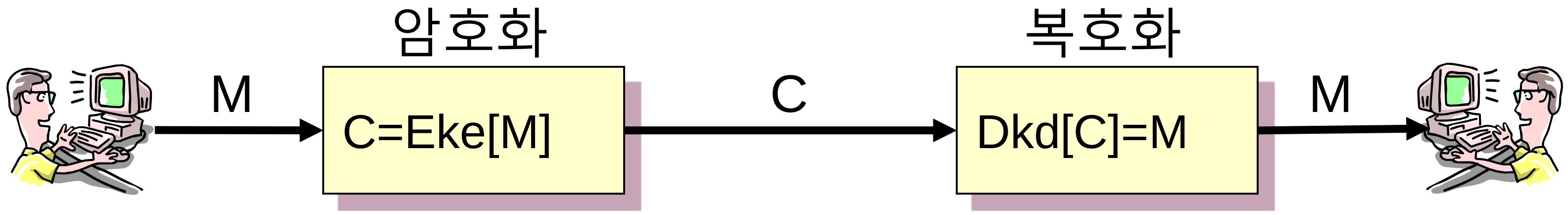
C=Eke[M]

C

Dkd[C]=M

M

공개키 암호 알고리즘(Public Key Cryptography)



암호화

복호화

M

C=Eke[M]

C

Dkd[C]=M

M

A

A의 비밀키

A의 공개키

B의 공개키

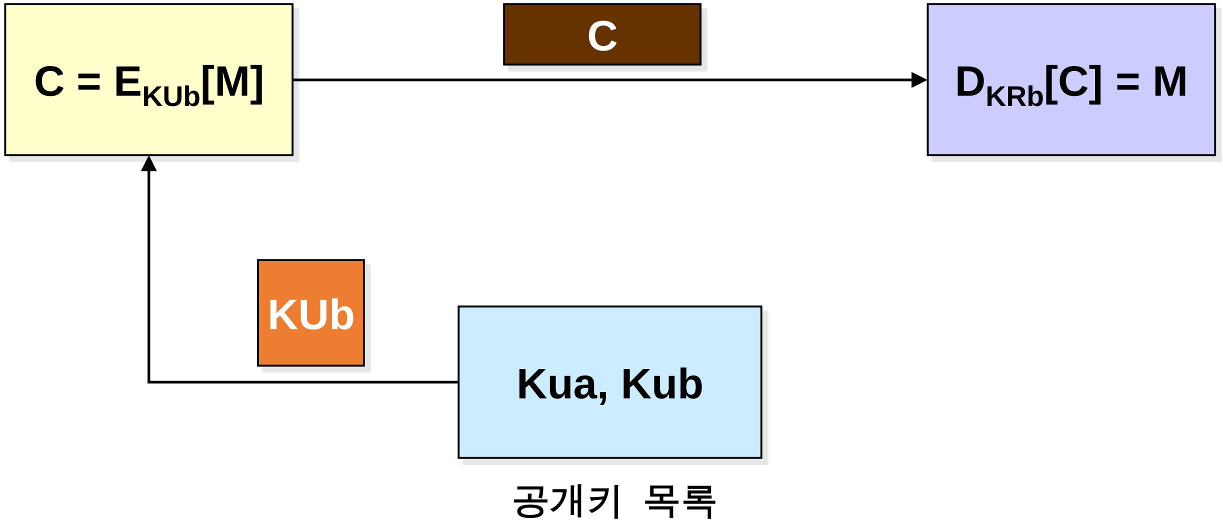
B

B의 비밀키

B의 공개키

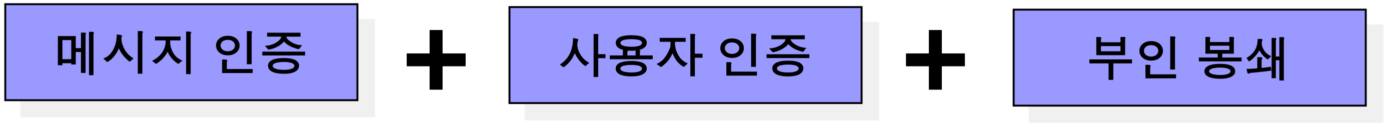
A의 공개키

기밀성



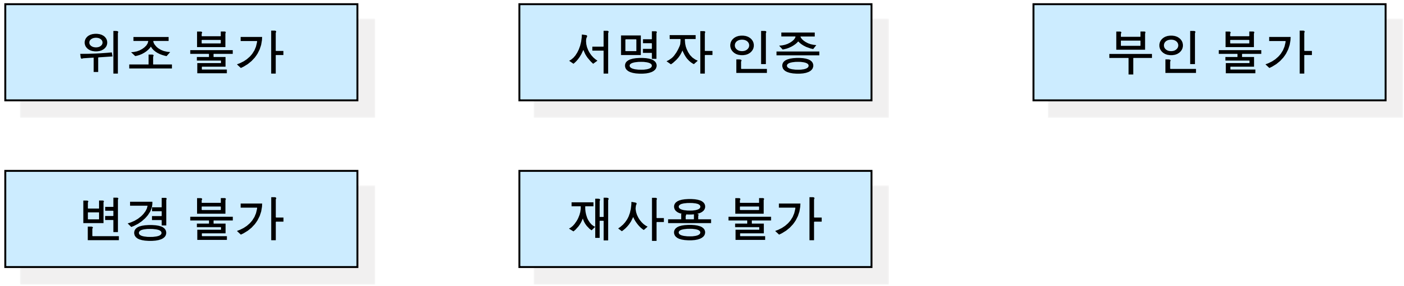
전자서명(Digital Signature)

- 제공 기능



메시지 인증 + 사용자 인증 + 부인 봉쇄

- 요구 조건



위조 불가

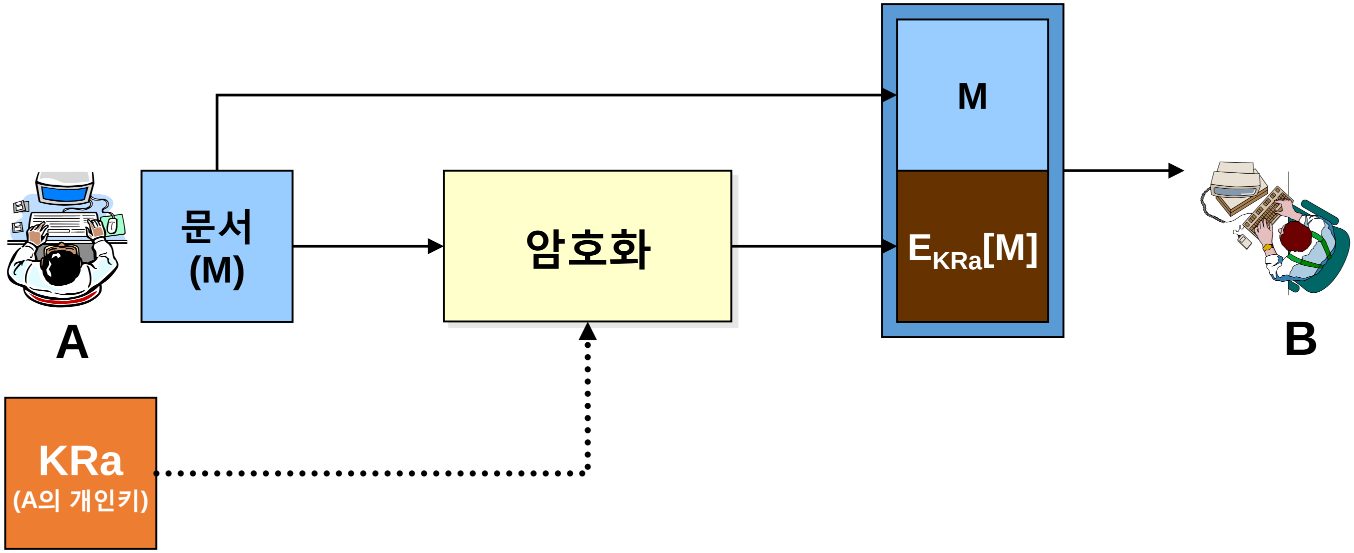
서명자 인증

부인 불가

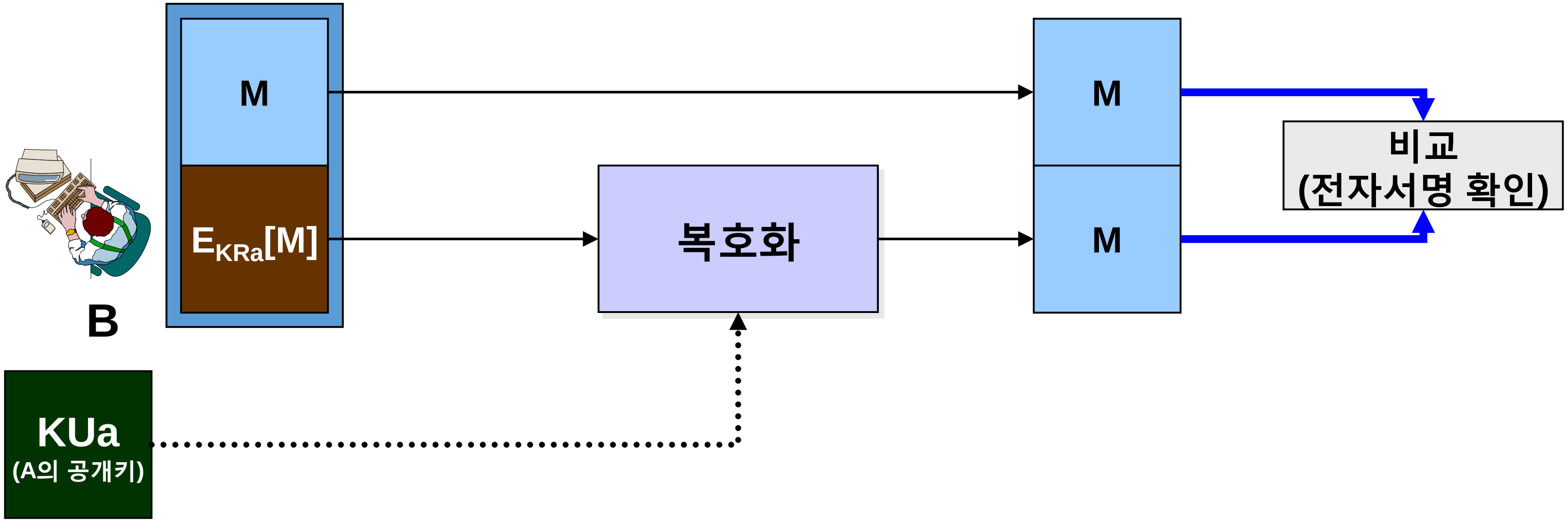
변경 불가

재사용 불가

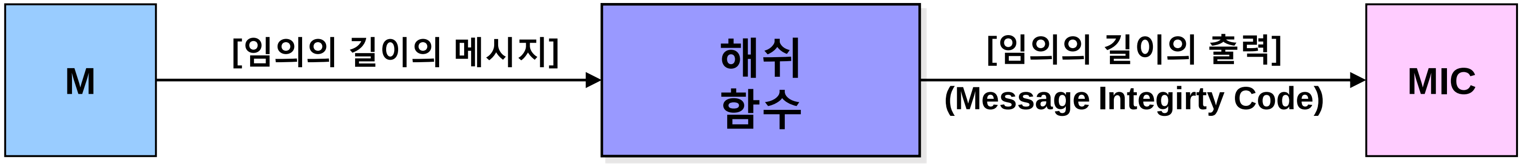
- 전자서명 생성 I



- 전자서명 확인 I



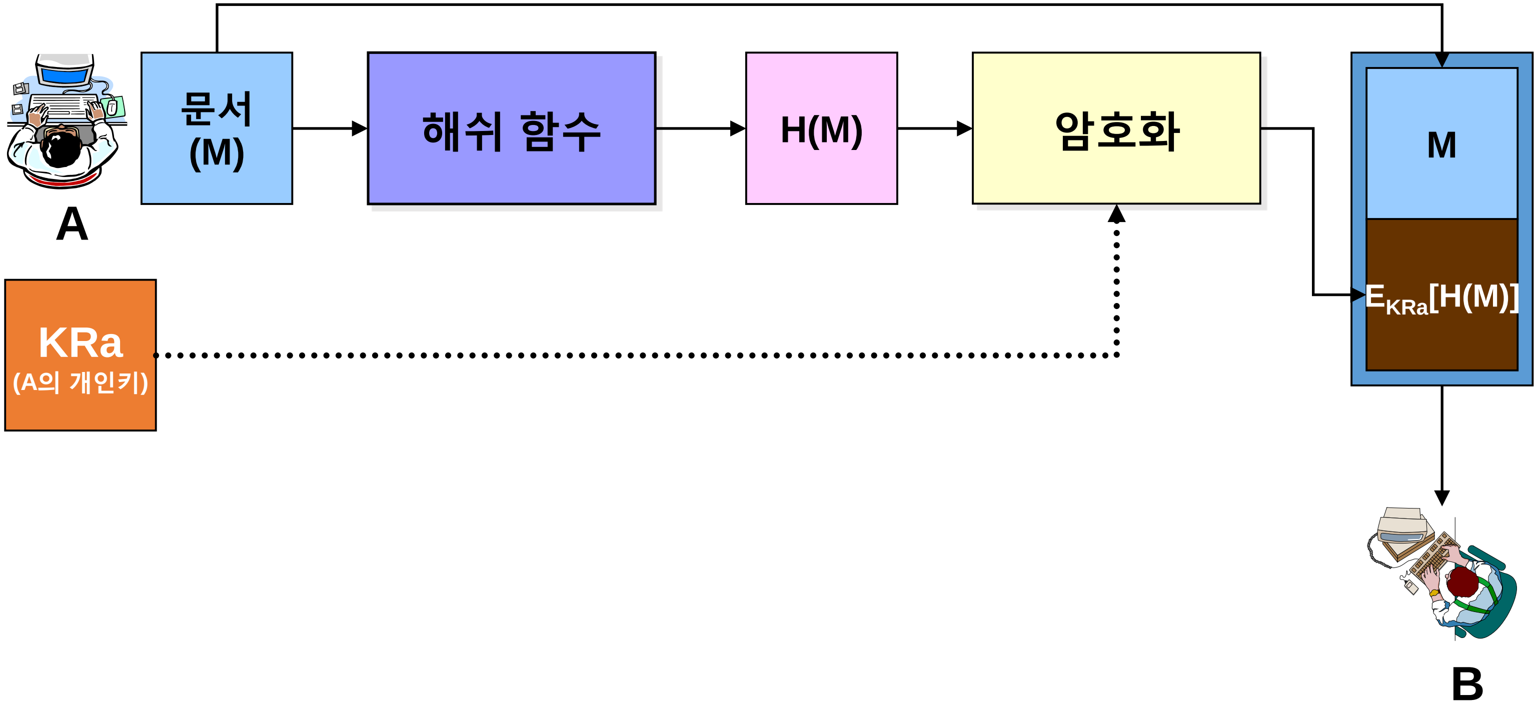
- 해쉬함수

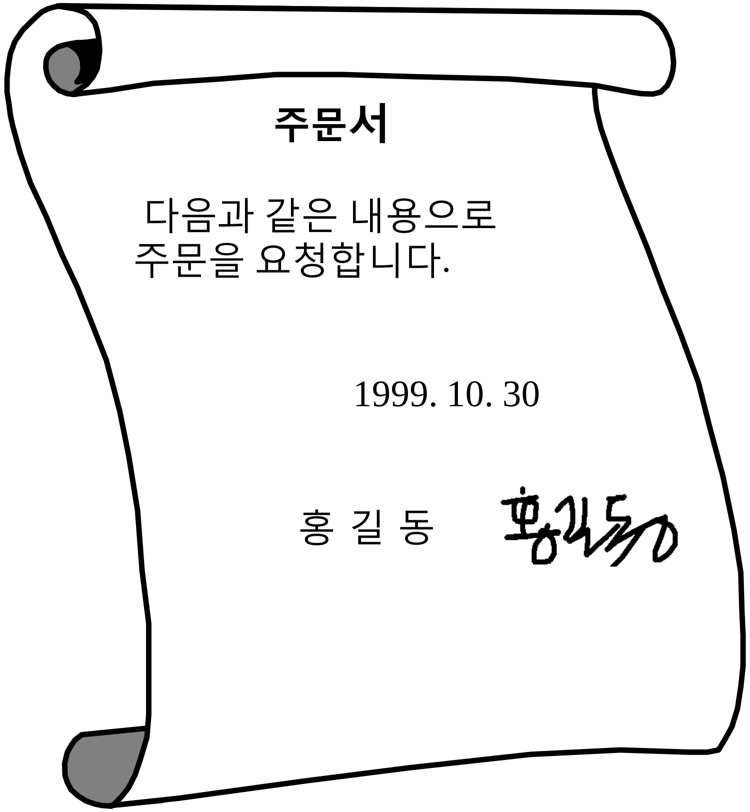


- 메시지로부터 출력을 계산하는 것은 용이
- 출력으로부터 메시지를 계산하는 것은 계산상 불가능
- 서로 같는 해쉬값을 갖는 두 메시지를 찾는 것은 계산상 불가능

➥메시지 인증과 전자서명의 효율을 높이기 위해 사용

- 전자서명 생성 II





주문서

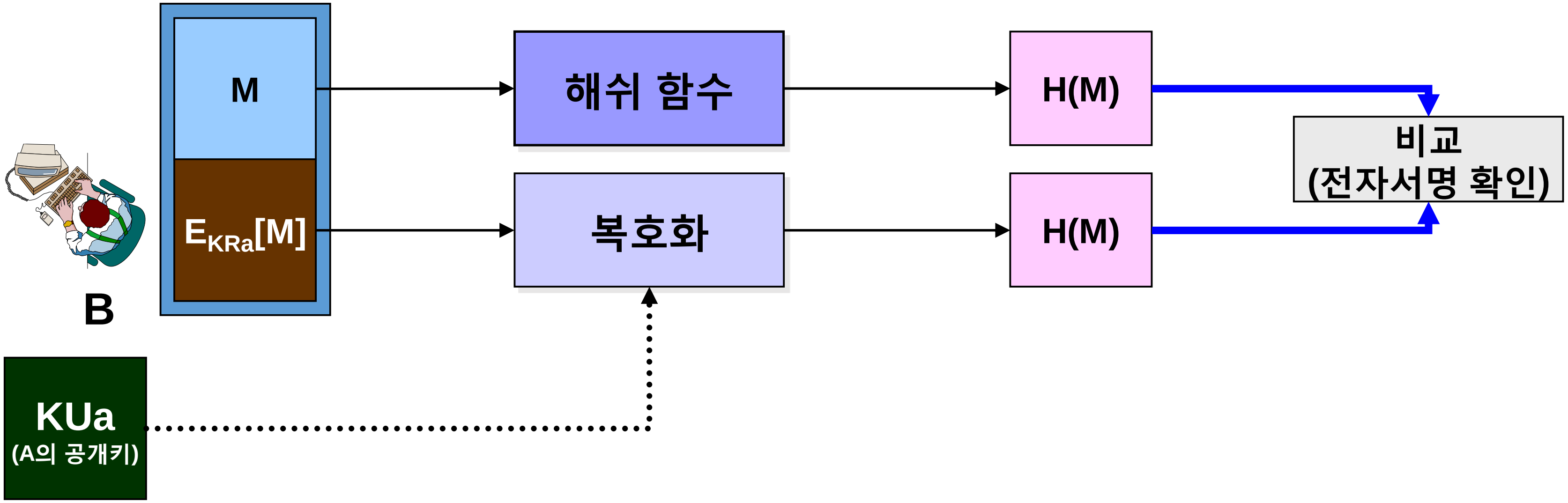
다음과 같은 내용으로
주문을 요청합니다.

1999. 10. 30

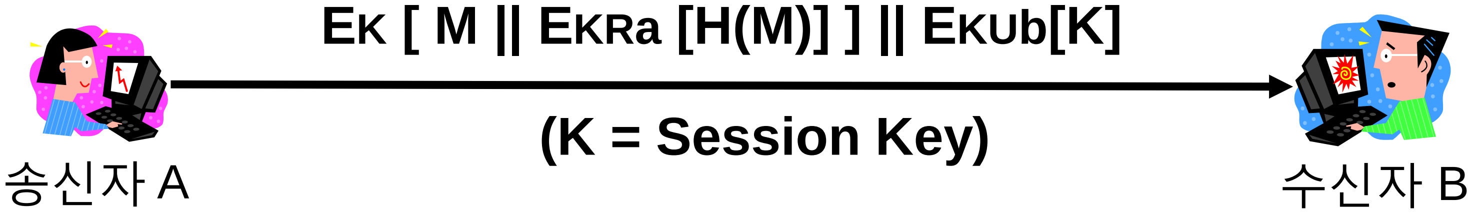
홍 길 동 홍길동

```
주문서

다음과 같은 내용으로 주문을 요청합니다.

1999.10.30

홍길동

------BEGIN Digital Signature ------
Content-Type: application/x-pkcs7
              -signature;
Content-Disposition: attachment;
Content-Transfer-Encoding: base64

ggM7oAMCAQICFEQ41iugpRpD1VzRmFQZQnk
TWVtZSBSb290IENBMQ4wDAYDVQQKEwVTTWV
KoZIhvcNAQkEMRYEFEoe7hq0yyoZEWUp7gA
xI5z0pabAAAAAAAAxI5z0pabAAAAAAAAxI5
------END Digital Signature ------
```

- 전자서명 확인 II



기밀성과 전자서명



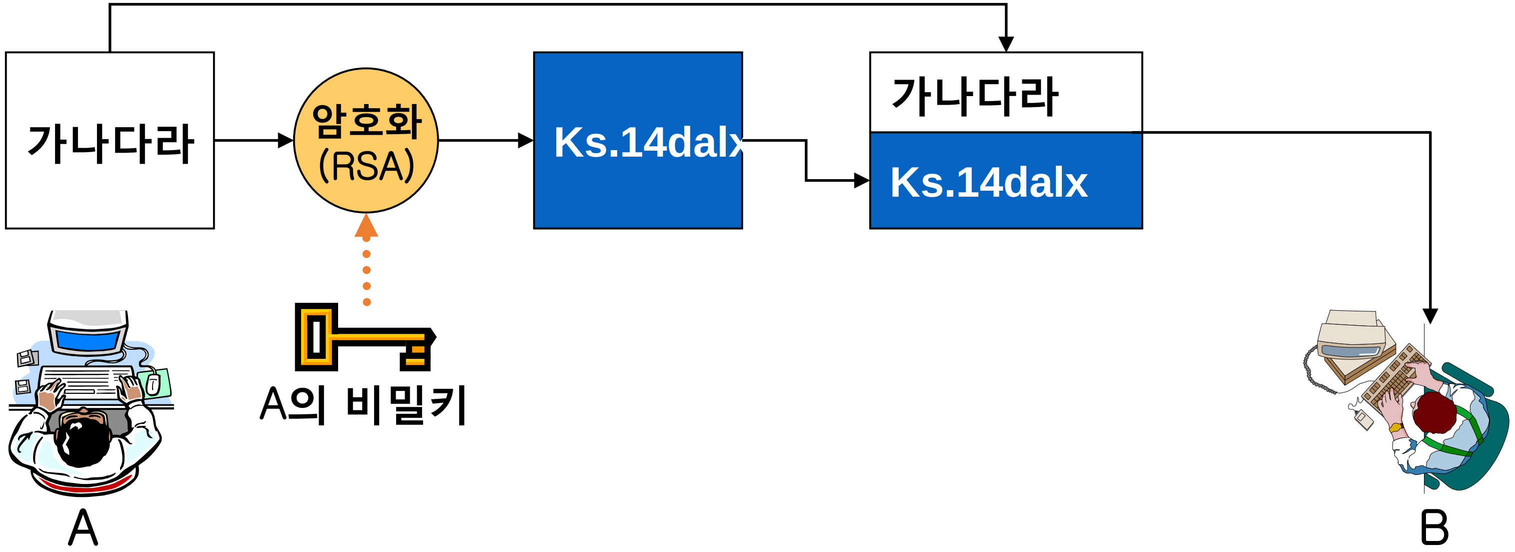
$E_K [\ M \,\|\, E_{KRa} [H(M)]\ ] \,\|\, E_{KUb}[K]$

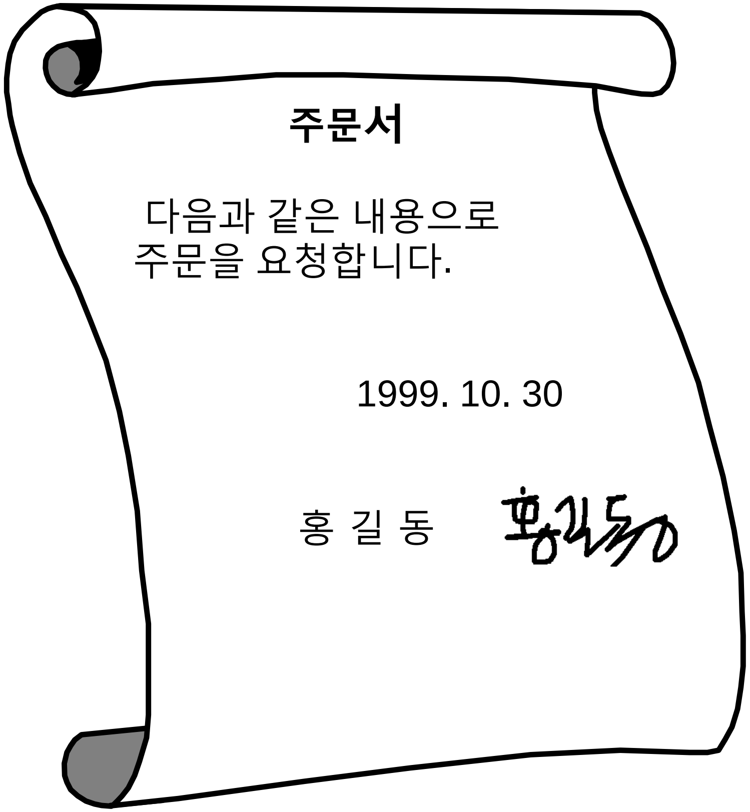
(K = Session Key)

송신자 A

수신자 B

- 전자서명 생성





주문서

다음과 같은 내용으로
주문을 요청합니다.

1999. 10. 30

홍 길 동 홍길동

주문서

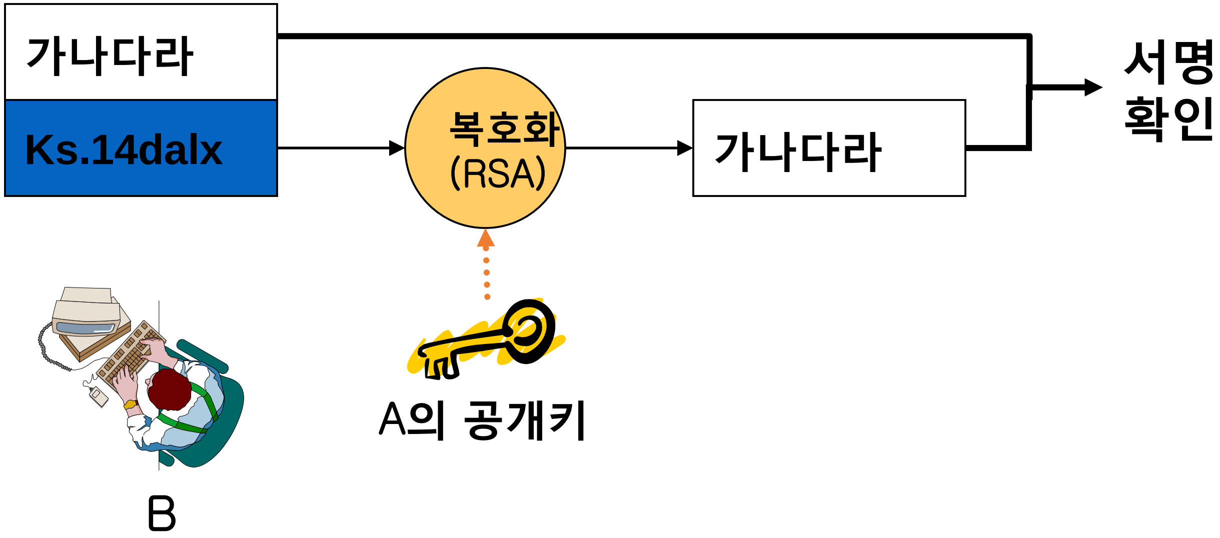
다음과 같은 내용으로 주문을 요청합니다.

1999.10.30

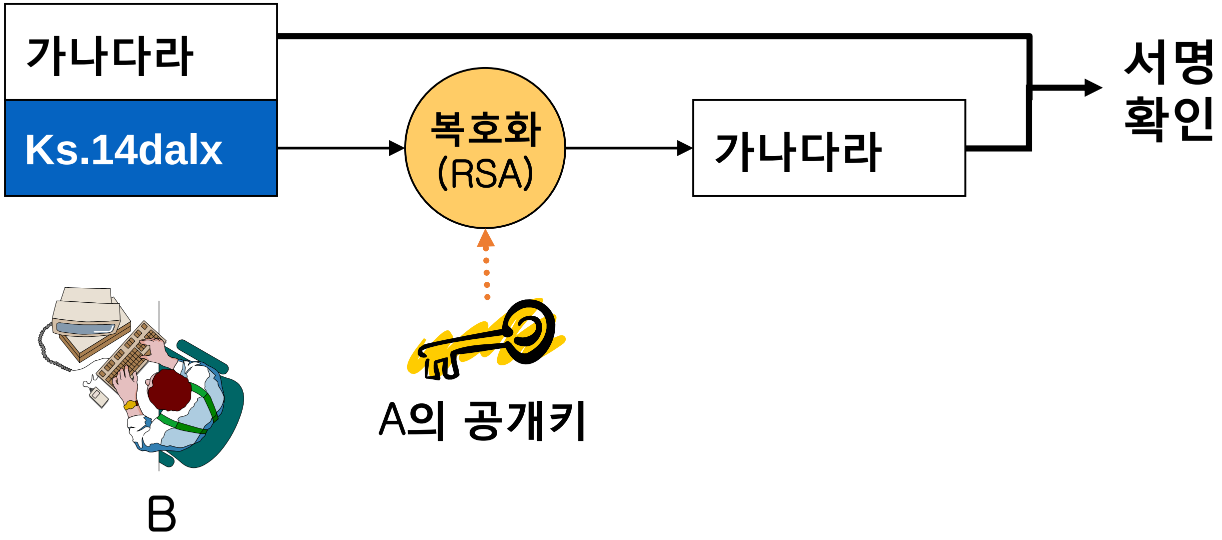
홍길동

```
------BEGIN Digital Signature ------
Content-Type: application/x-pkcs7
              -signature;
Content-Disposition: attachment;
Content-Transfer-Encoding: base64

ggM7oAMCAQICFEQ41iugpRpD1VzRmFQZQnk
TWVtZSBSb290IENBMQ4wDAYDVQQKEwVTTWV
KoZIhvcNAQkEMRYEFEoe7hq0yyoZEWUp7gA
xI5z0pabAAAAAAAAxI5z0pabAAAAAAAAxI5
------END Digital Signature ------
```

- 전자서명 확인



- 전자서명 확인



가나다라

Ks.14dalx

복호화
(RSA)

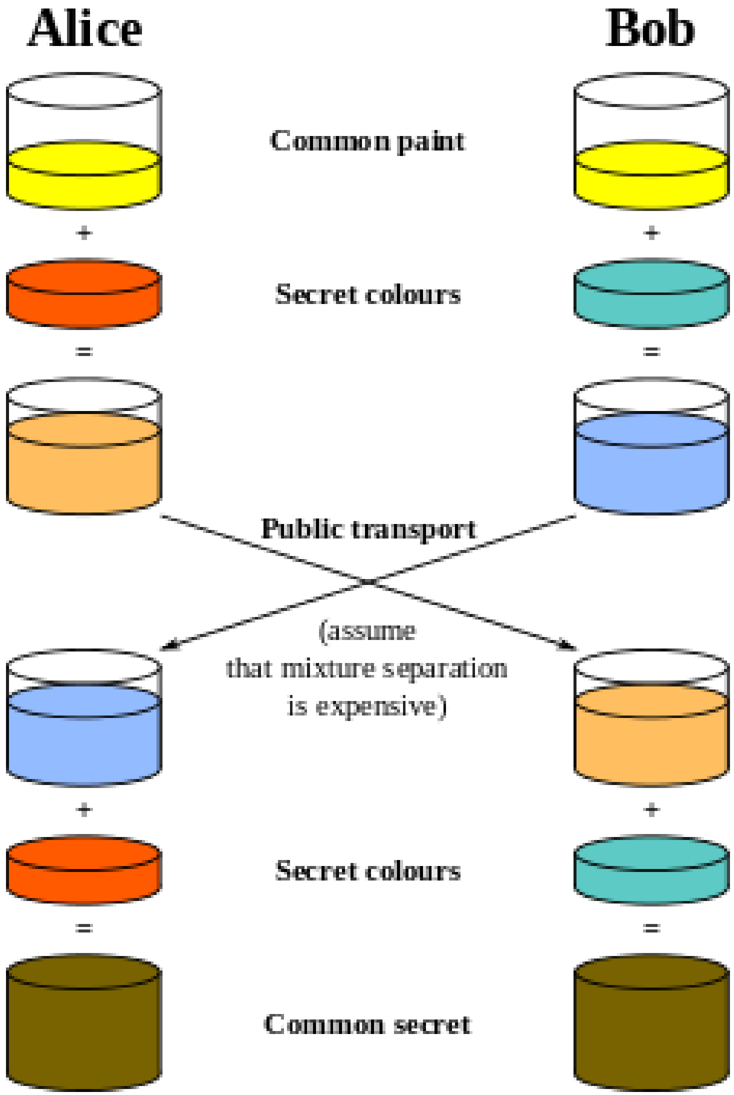
가나다라

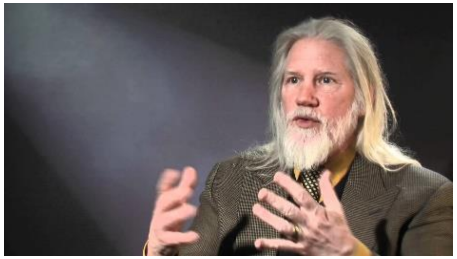
서명
확인

A의 공개키

B

Diffie-Hellman





- Alice and Bob agree to use a prime number p = 23 and base g = 5 (which is a primitive root modulo 23).
- Alice chooses a secret integer $\boldsymbol{a}$ = **6**, then sends Bob $A = g^{\boldsymbol{a}} \bmod p$
  - $A = 5^{\mathbf{6}} \bmod 23 = 8$
- Bob chooses a secret integer $\boldsymbol{b}$ = **15**, then sends Alice $B = g^{\boldsymbol{b}} \bmod p$
  - $B = 5^{\mathbf{15}} \bmod 23 = 19$
- Alice computes $\boldsymbol{s} = B^{\boldsymbol{a}} \bmod p$
  - $\boldsymbol{s} = 19^{\mathbf{6}} \bmod 23 = \mathbf{2}$
- Bob computes $\boldsymbol{s} = A^{\boldsymbol{b}} \bmod p$
  - $\boldsymbol{s} = 8^{\mathbf{15}} \bmod 23 = \mathbf{2}$

RSA

Adi Sharmir Ron Rivest Lin Addleman

BOSTON RED SOX
51 YEARS TOGETHER
DANA-FARBER CANCER INSTITUTE
THE JIMMY FUND
FENWAY PARK
AMERICAN LEAGUE
NATIONAL LEAGUE
NY
BOSTON
APRIL 16
AT BAT
BALL
STRIKE
OUT

1. Select primes: p=17 & q=11
2. Compute $n = pq$ =17×11=187
3. Compute $\varnothing(n)=(p-1)(q-1)$=16×10=160
4. Select e *:* gcd(e,160)=1*;* choose e=7
5. Determine d*:* de=1 mod 160 and d < 160 Value is d=23 since 23×7=161= 10×160+1
6. Publish public key KU={7,187}
7. Keep secret private key KR={23,17,11}

- sample RSA encryption/decryption is:
- given message M = 88 (nb. 88<187)
- encryption:

  $C = 88^{7} \bmod 187 = 11$
- decryption:

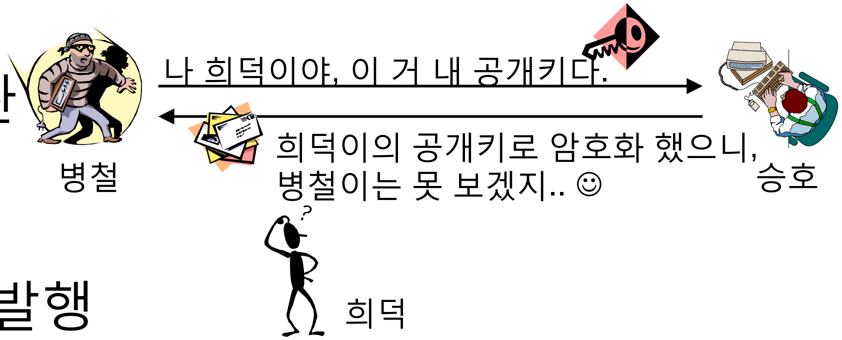
  $M = 11^{23} \bmod 187 = 88$

인증기관(CA: Certification Authority)

- 공개키 암호방식의 문제점
  - ■ 사용자 개인키 보관

    ➔ IC카드 활용

    ➔ 암호화하여 디스켓, HDD 등에 보관



나 희덕이야, 이 거 내 공개키다.

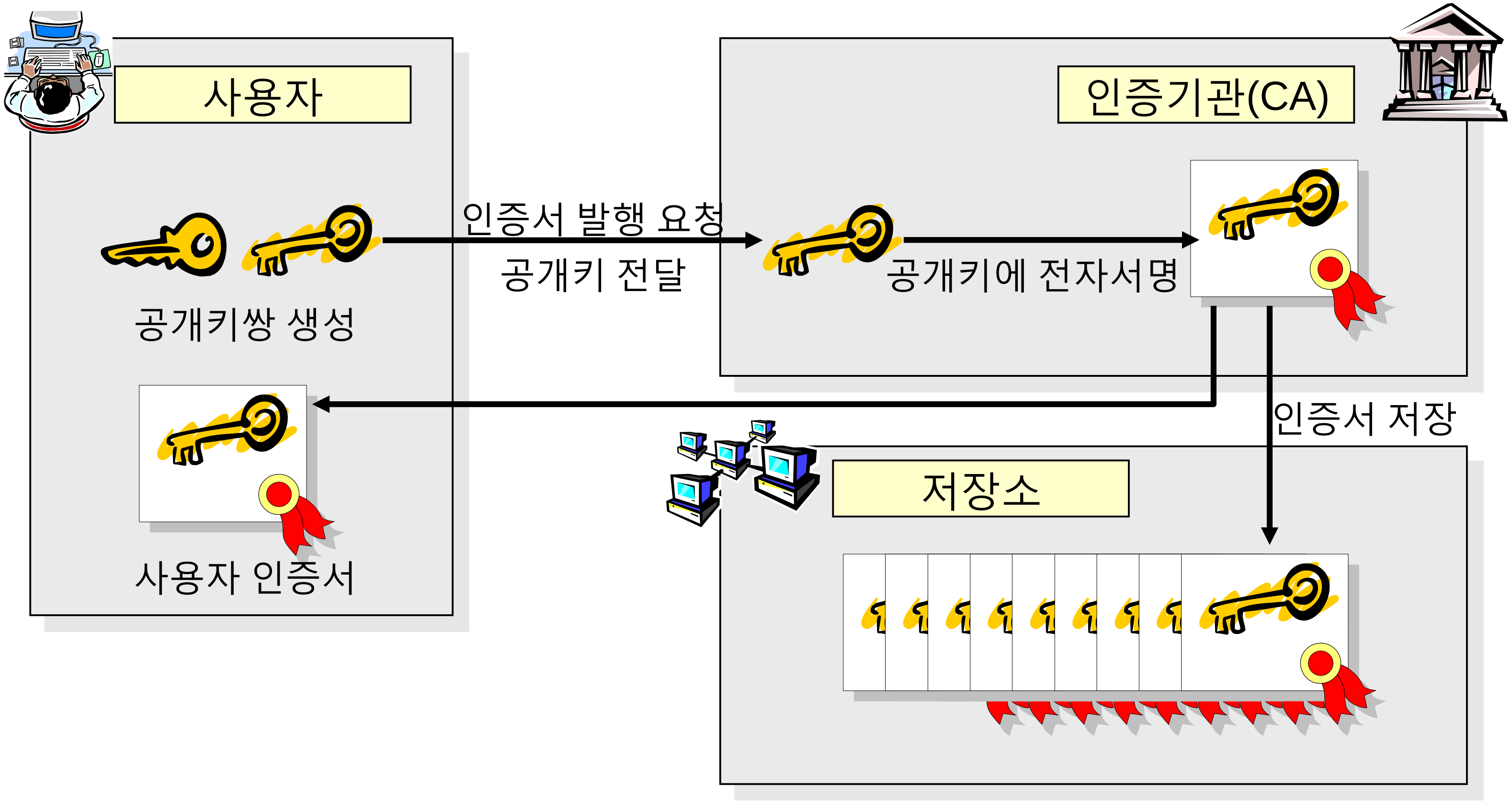
희덕이의 공개키로 암호화 했으니,
병철이는 못 보겠지.. ☺

  - 상대방의 공개키에 대한 진위 여부

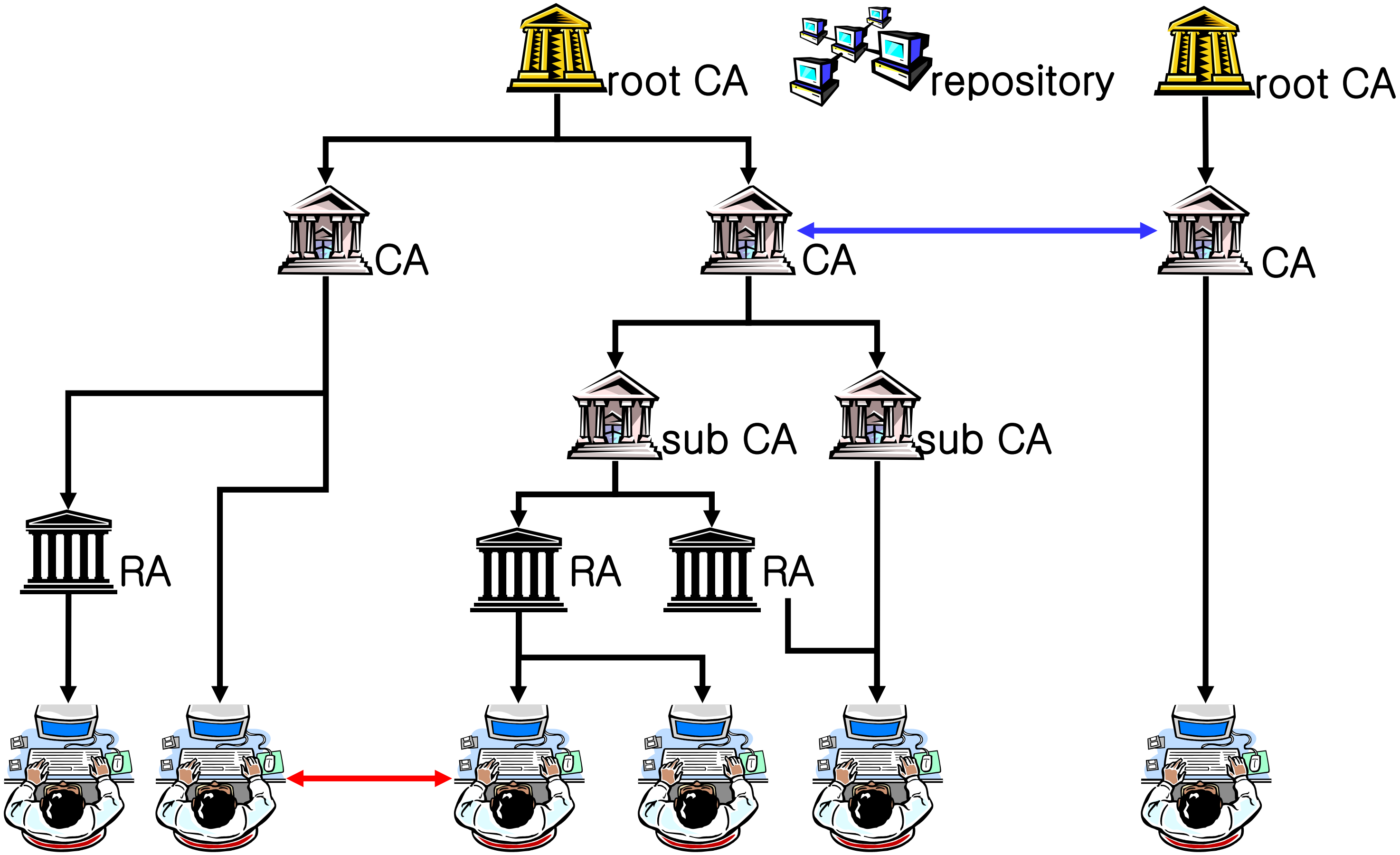
    ➔ 공개키에 대한 인증서(Certificate) 발행

- 인증기관: CA(Certification Authority)
  - 신뢰할 수 있는 제 3기관
  - 사용자의 공개키를 확인하여 인증서 발행

인증기관의 역할



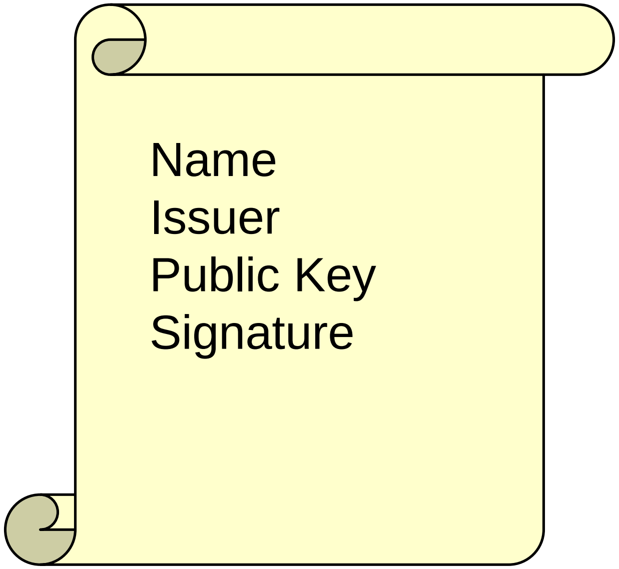
PKI: Public Key Infrastructure



- PKI allows you to know that a given public key belongs to a given user
- PKI builds off of asymmetric encryption:
  - Each entity has two keys: public and private
  - The private key is known only to the entity
- The public key is given to the world encapsulated in a X.509 certificate

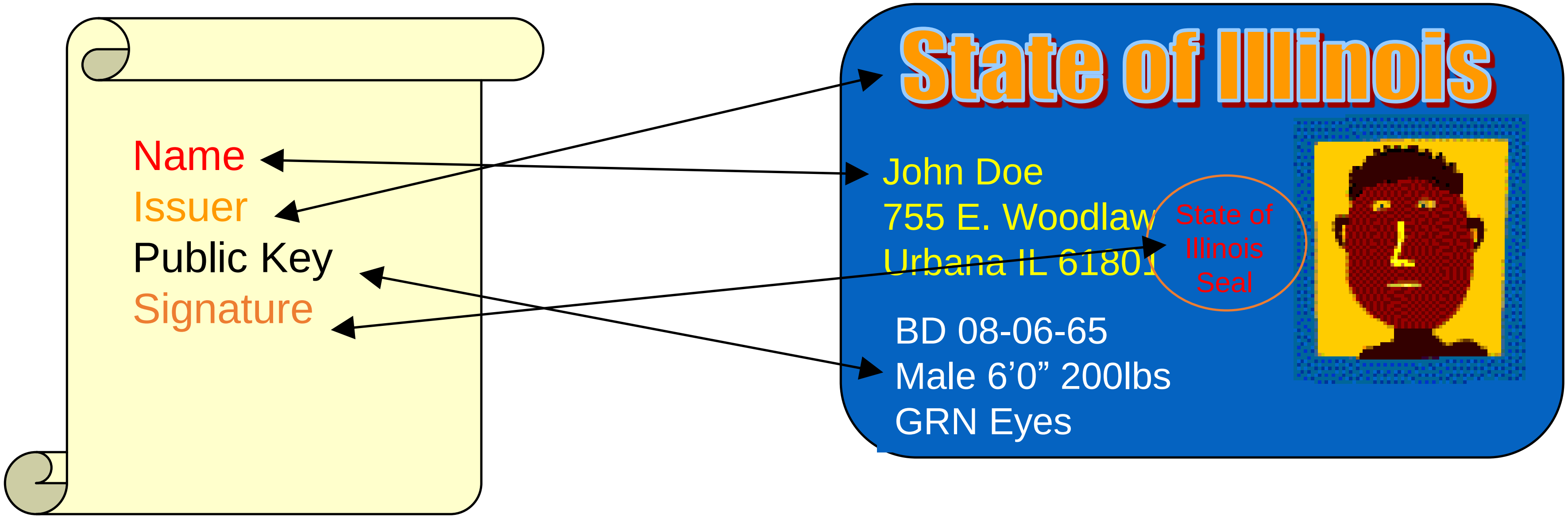
Certificates

- A X.509 certificate binds a public key to a name
- It includes a name and a public key (among other things) bundled together and signed by a trusted party (Issuer)

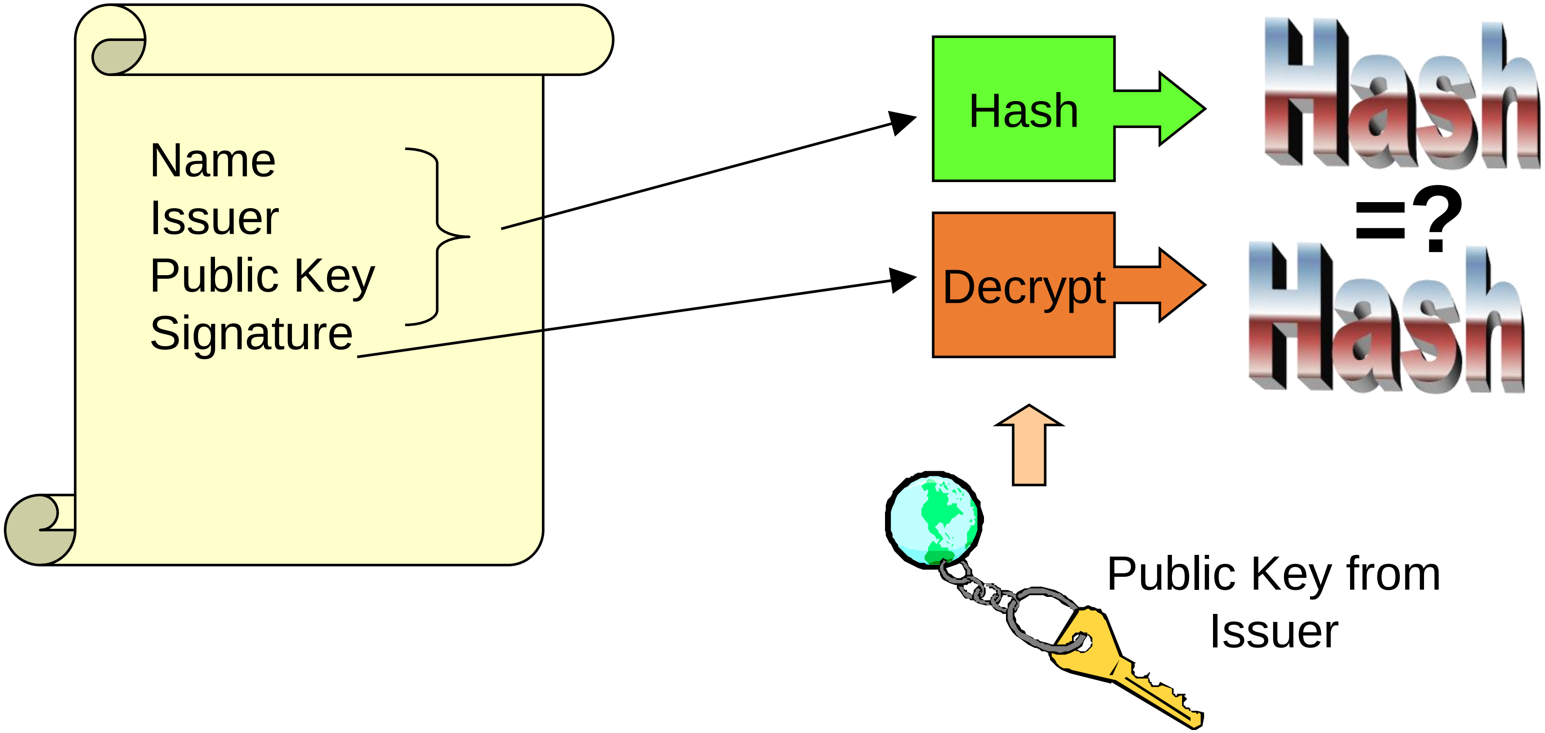


Name
Issuer
Public Key
Signature

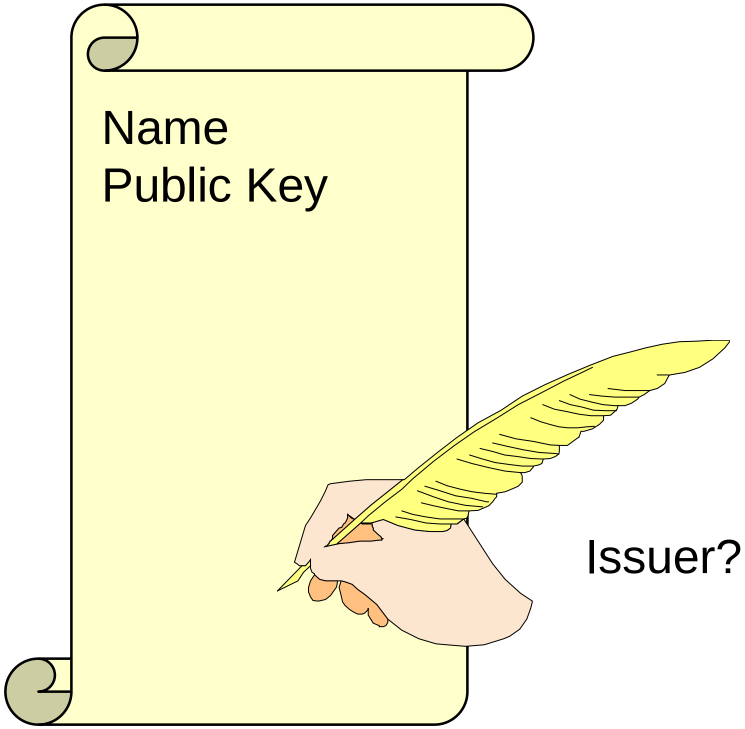
- Similar to passport or driver’s license



- By checking the signature, one can determine that a public key belongs to a given user.

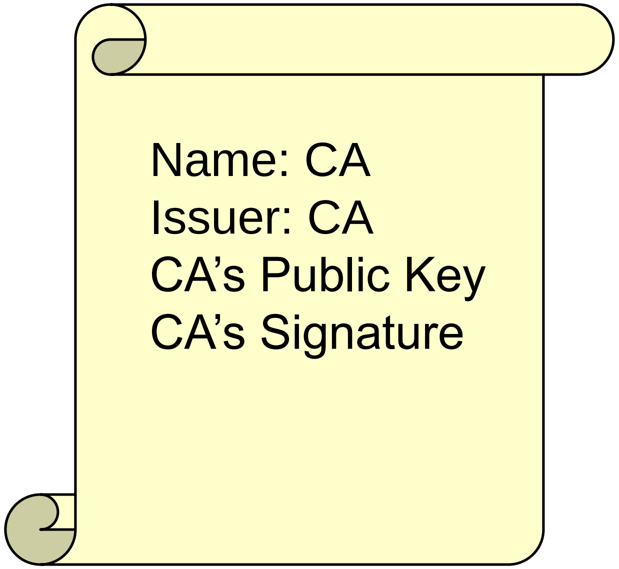


- Question: Who signs certificates?
- Answer: A small set of trusted entities known as Certificate Authorities (CAs)



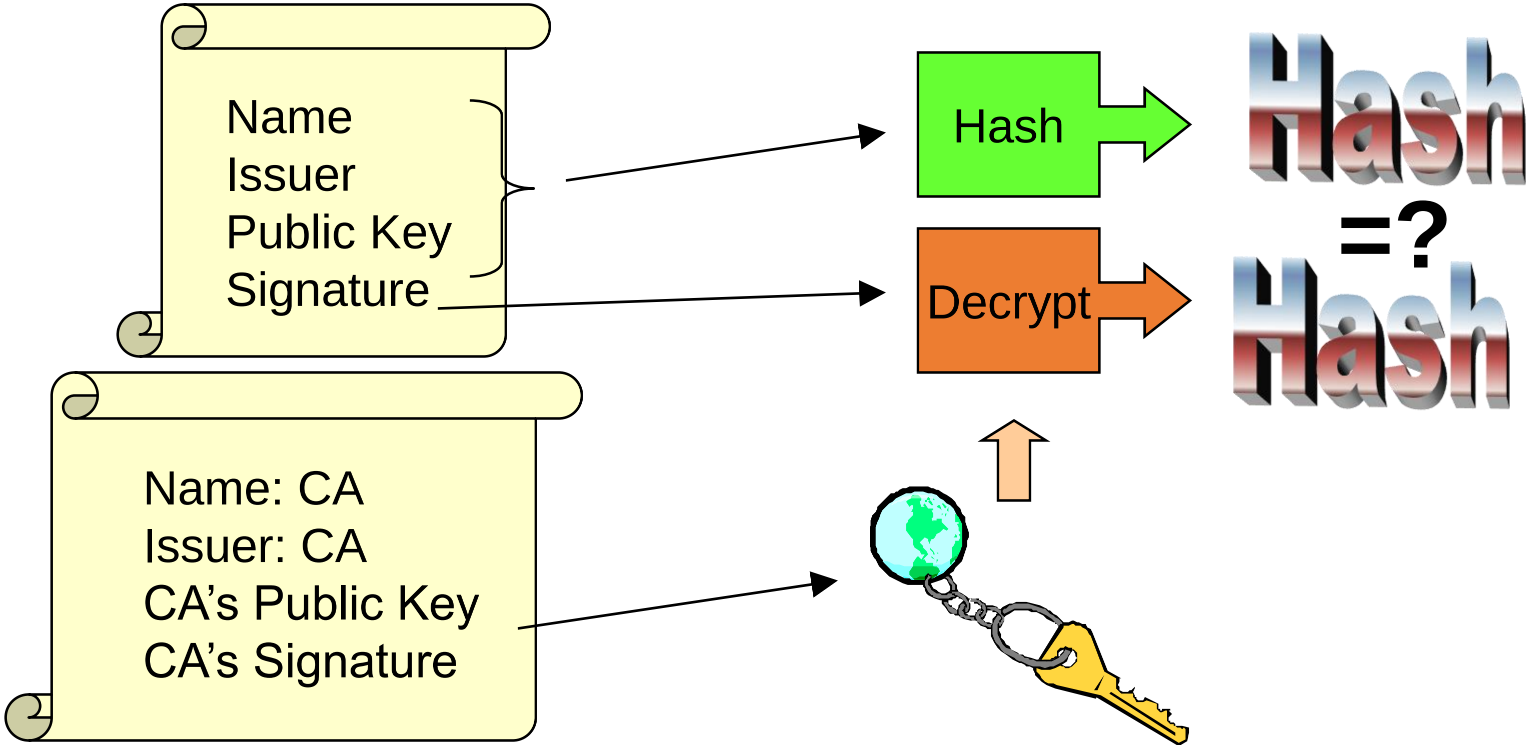
CA

- A Certificate Authority is an entity that exists only to sign user certificates
- The CA signs it’s own certificate which is distributed in a trusted manner



Name: CA
Issuer: CA
CA’s Public Key
CA’s Signature

- The public key from the CA certificate can then be used to verify other certificates



Name
Issuer
Public Key
Signature

Hash

Decrypt

Hash
=?
Hash

Name: CA
Issuer: CA
CA’s Public Key
CA’s Signature

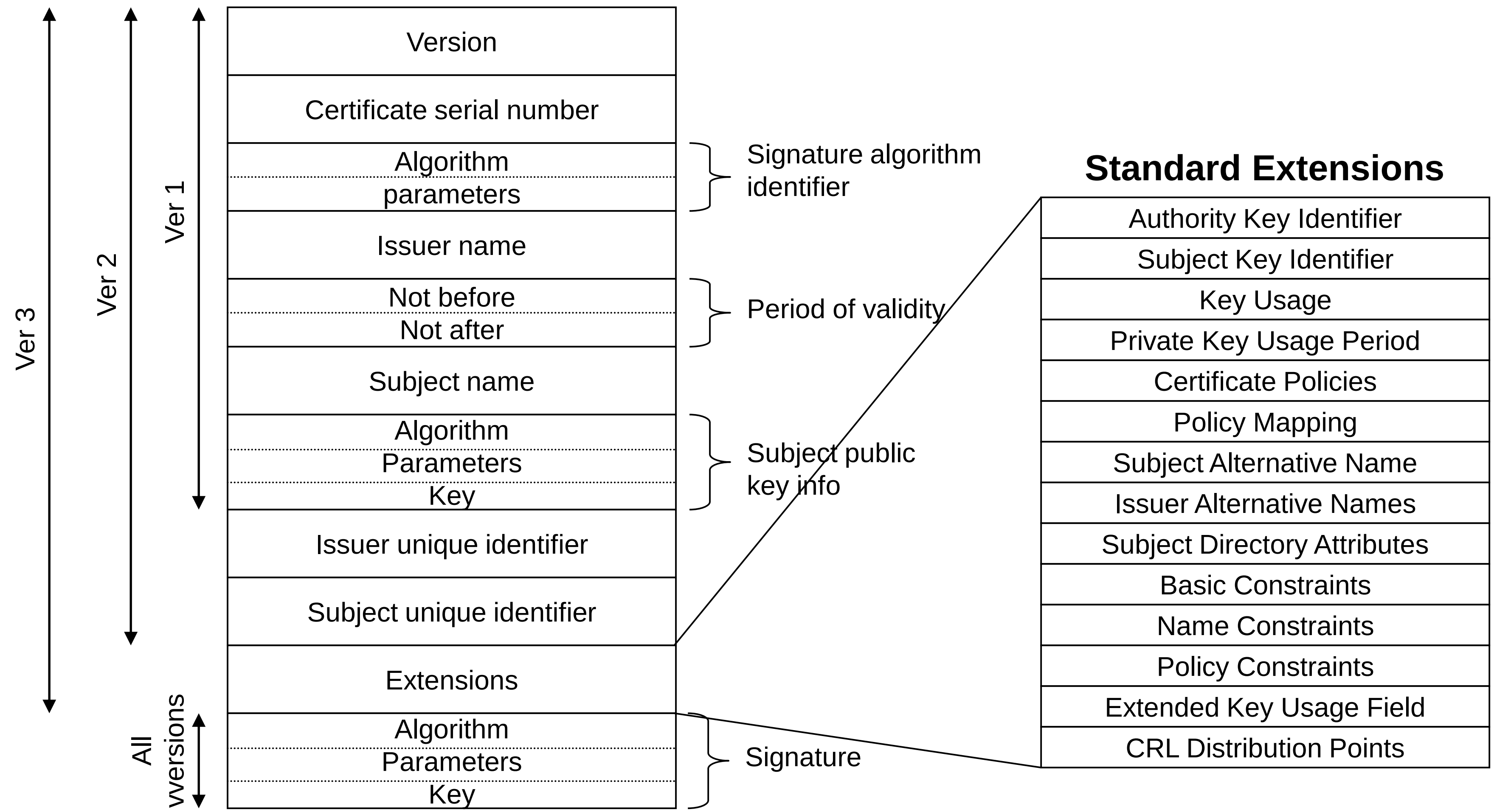
Certificate Policy(CP)

- Each CA has a Certificate Policy (CA) which states when and how a CA issues certificates.
- It states who it will issue certificates for
  - Just like the State of Illinois only issues driver’s licenses’ for residents of the state of Illinois
  - A CA for a grid typically only issues certificates for folks that are already approved to use resources on the grid

Certificate Policy (CP)

- A CA’s CP states how it identifies the people it issues certificates to
    - Similar to having to show a birth certificate to get a driver’s license
    - Some CA’s are very stringent and require similar proof of identity
    - Others are lenient and only require proof via email

X.509 v3 인증서 is an IETF/ITU-T standard



Version

Certificate serial number

Algorithm
parameters

Signature algorithm identifier

Issuer name

Not before
Not after

Period of validity

Subject name

Algorithm
Parameters
Key

Subject public key info

Issuer unique identifier

Subject unique identifier

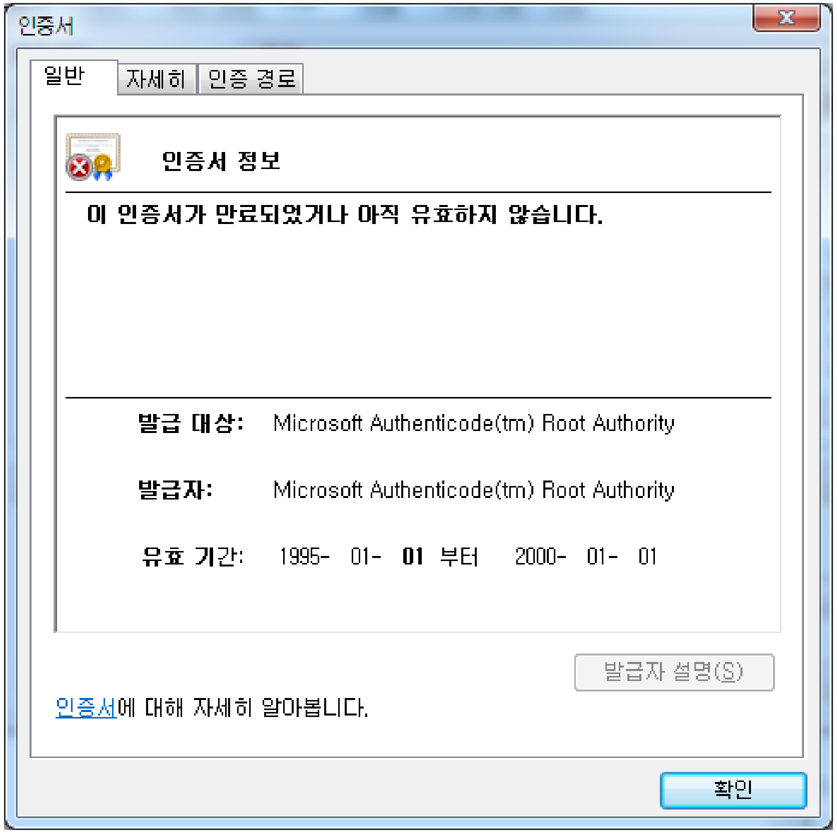
Extensions

Algorithm
Parameters
Key

Signature

Standard Extensions

- Authority Key Identifier
- Subject Key Identifier
- Key Usage
- Private Key Usage Period
- Certificate Policies
- Policy Mapping
- Subject Alternative Name
- Issuer Alternative Names
- Subject Directory Attributes
- Basic Constraints
- Name Constraints
- Policy Constraints
- Extended Key Usage Field
- CRL Distribution Points


인증서

인증서 정보

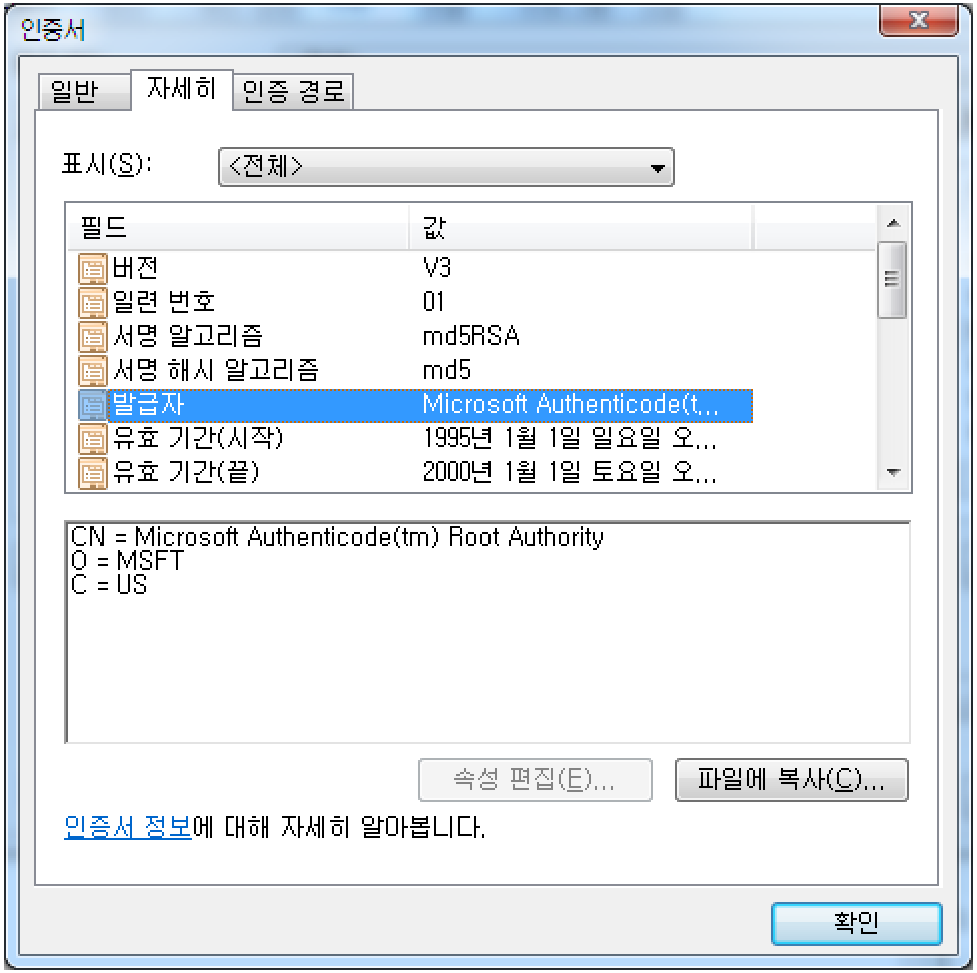
이 인증서가 만료되었거나 아직 유효하지 않습니다.

발급 대상: Microsoft Authenticode(tm) Root Authority

발급자: Microsoft Authenticode(tm) Root Authority

유효 기간: 1995- 01- **01** 부터 2000- 01- 01

인증서에 대해 자세히 알아봅니다.


인증서

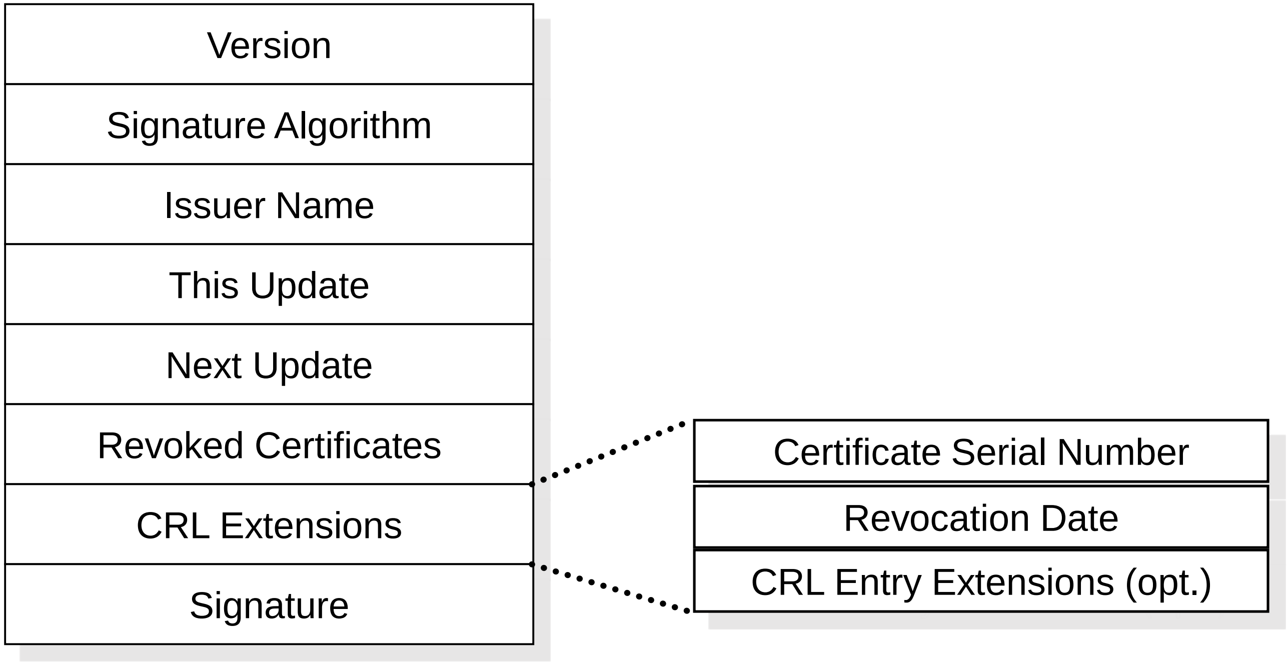
표시(S): <전체>

필드	값
버전	V3
일련 번호	01
서명 알고리즘	md5RSA
서명 해시 알고리즘	md5
발급자	Microsoft Authenticode(t...
유효 기간(시작)	1995년 1월 1일 일요일 오...
유효 기간(끝)	2000년 1월 1일 토요일 오...

```
CN = Microsoft Authenticode(tm) Root Authority
O = MSFT
C = US
```

인증서 정보에 대해 자세히 알아봅니다.

X.509 v2 CRL



Version

Signature Algorithm

Issuer Name

This Update

Next Update

Revoked Certificates

CRL Extensions

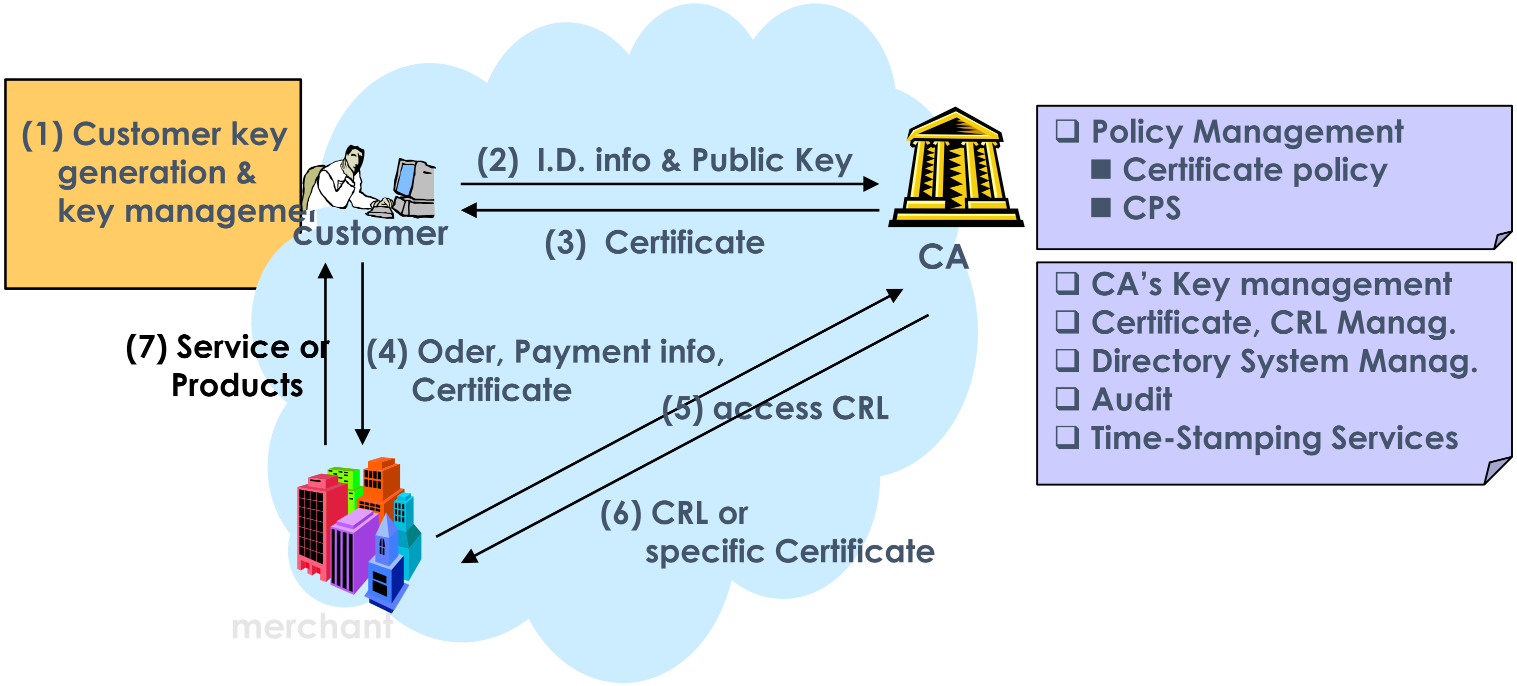
Signature

Certificate Serial Number

Revocation Date

CRL Entry Extensions (opt.)

- 인증기관의 업무
  - 사용자 신원 확인
  - 인증서 발행
  - 인증서 공개
  - 인증서의 효력 상실
  - 비밀키의 생성 및 관리
  - 업무의 중단 및 관리



(1) Customer key generation & key management

customer

(2) I.D. info & Public Key

(3) Certificate

CA

(7) Service or Products

(4) Oder, Payment info, Certificate

(5) access CRL

(6) CRL or specific Certificate

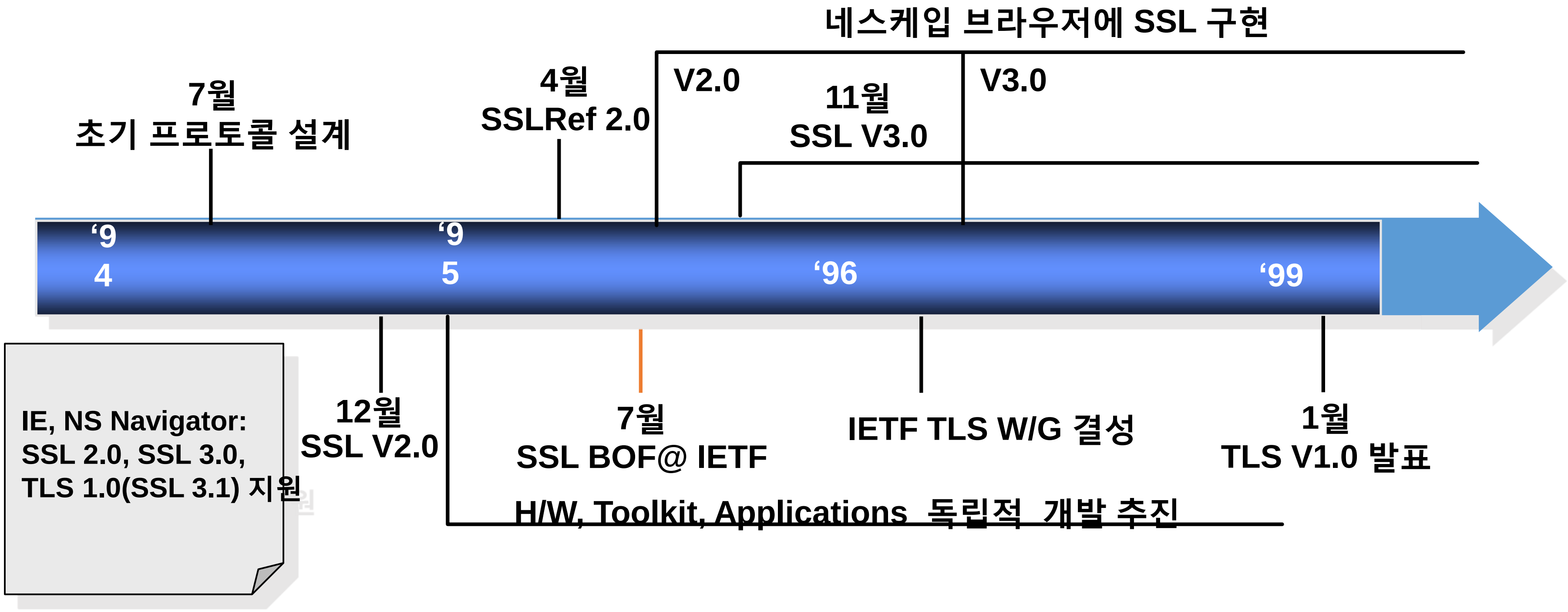
merchant

❑ Policy Management
- ■ Certificate policy
- ■ CPS

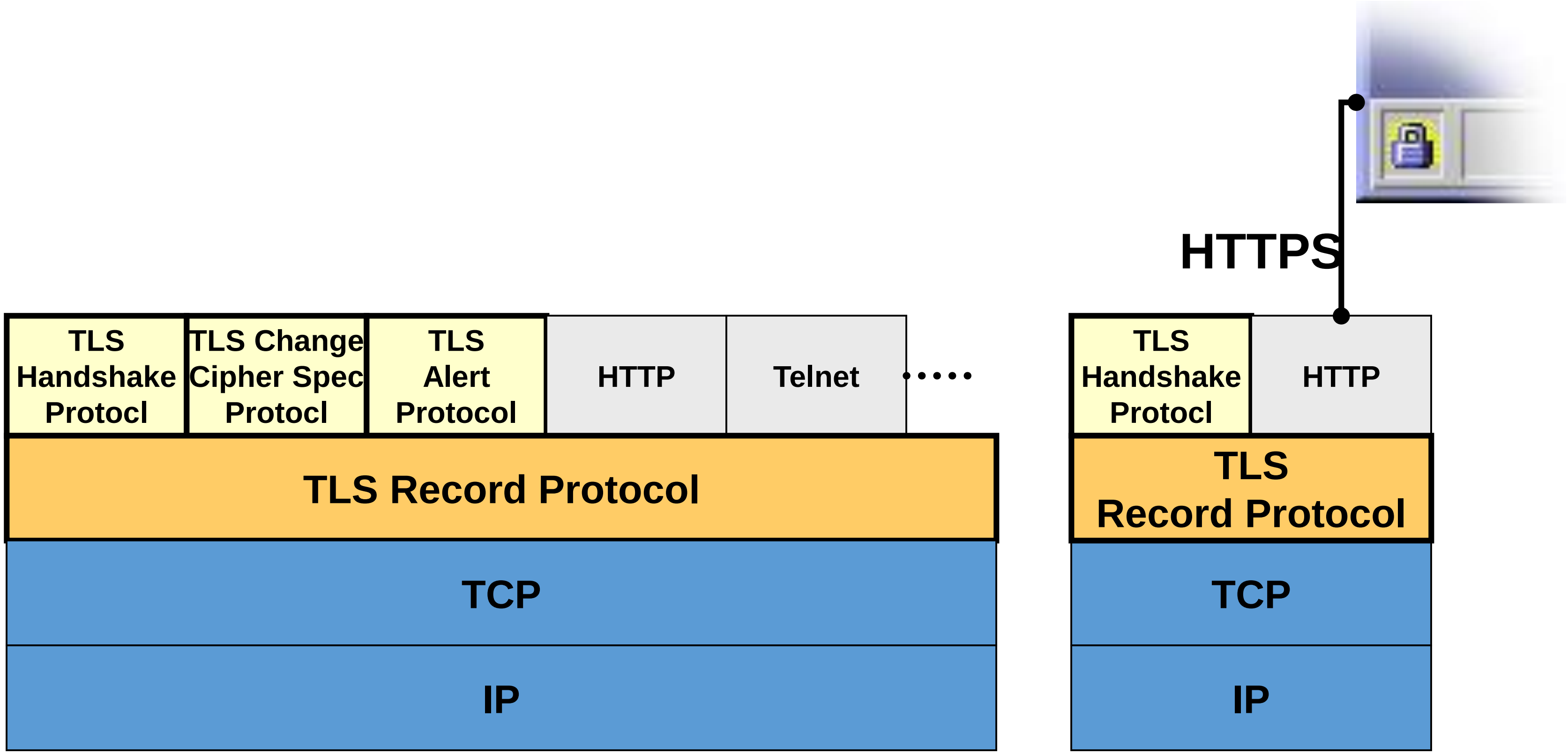
❑ CA's Key management
❑ Certificate, CRL Manag.
❑ Directory System Manag.
❑ Audit
❑ Time-Stamping Services

4. SSL/TLS

- SSL : 1994년 Netscape 사에서 처음으로 제안, 현재 SSL v3.
  - TLS : IETF TLS W/G, SSL을 개정하여 표준화 진행중
    - RFC 2246 : The TLS Protocol Version 1.0



IE, NS Navigator:
SSL 2.0, SSL 3.0,
TLS 1.0(SSL 3.1) 지원



TLS Handshake Protocl

TLS Change Cipher Spec Protocl

TLS Alert Protocol

HTTP

Telnet

•••••

TLS Record Protocol

TCP

IP

HTTPS

TLS Handshake Protocl

HTTP

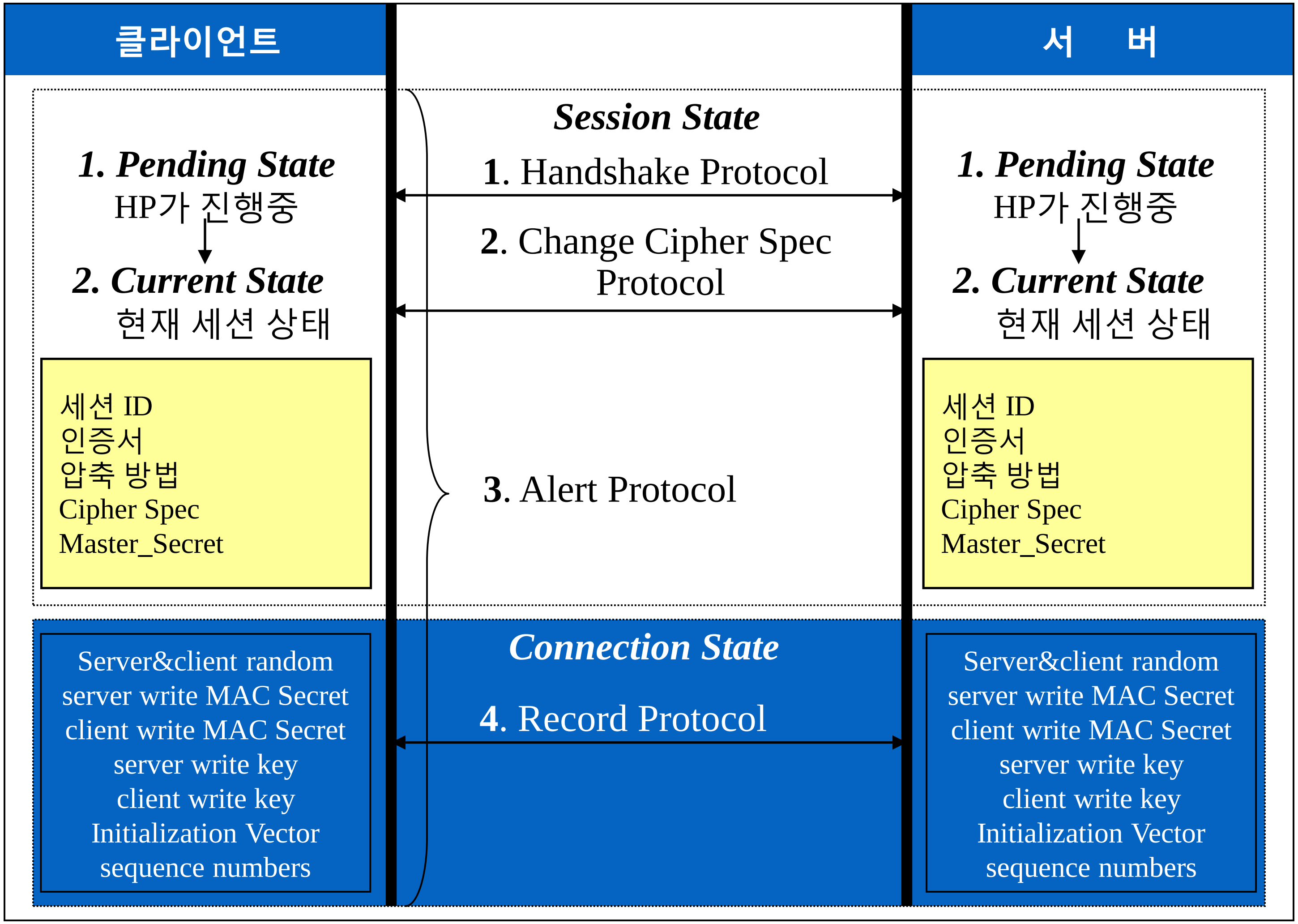
TLS Record Protocol

TCP

IP

The Goals of TLS

보안 서비스 (Security)	• 기밀성 • 무결성 • 사용자 인증
상호호환성 (Interoperability)	• **TLS**를 이용하는 어플리케이션 간의 상호호환성 확보 **(HTTP, Telnet, FTP,…)**
확장성 (Extensibility)	• 새로운 알고리즘의 추가의 용이성
효율성 (Efficiency)	• **Optional session caching scheme to reduce the # of connection**



클라이언트

1. Pending State
HP가 진행중

2. Current State
현재 세션 상태

세션 ID
인증서
압축 방법
Cipher Spec
Master_Secret

Server&client random
server write MAC Secret
client write MAC Secret
server write key
client write key
Initialization Vector
sequence numbers

Session State

1. Handshake Protocol

2. Change Cipher Spec Protocol

3. Alert Protocol

Connection State

4. Record Protocol

서 버

1. Pending State
HP가 진행중

2. Current State
현재 세션 상태

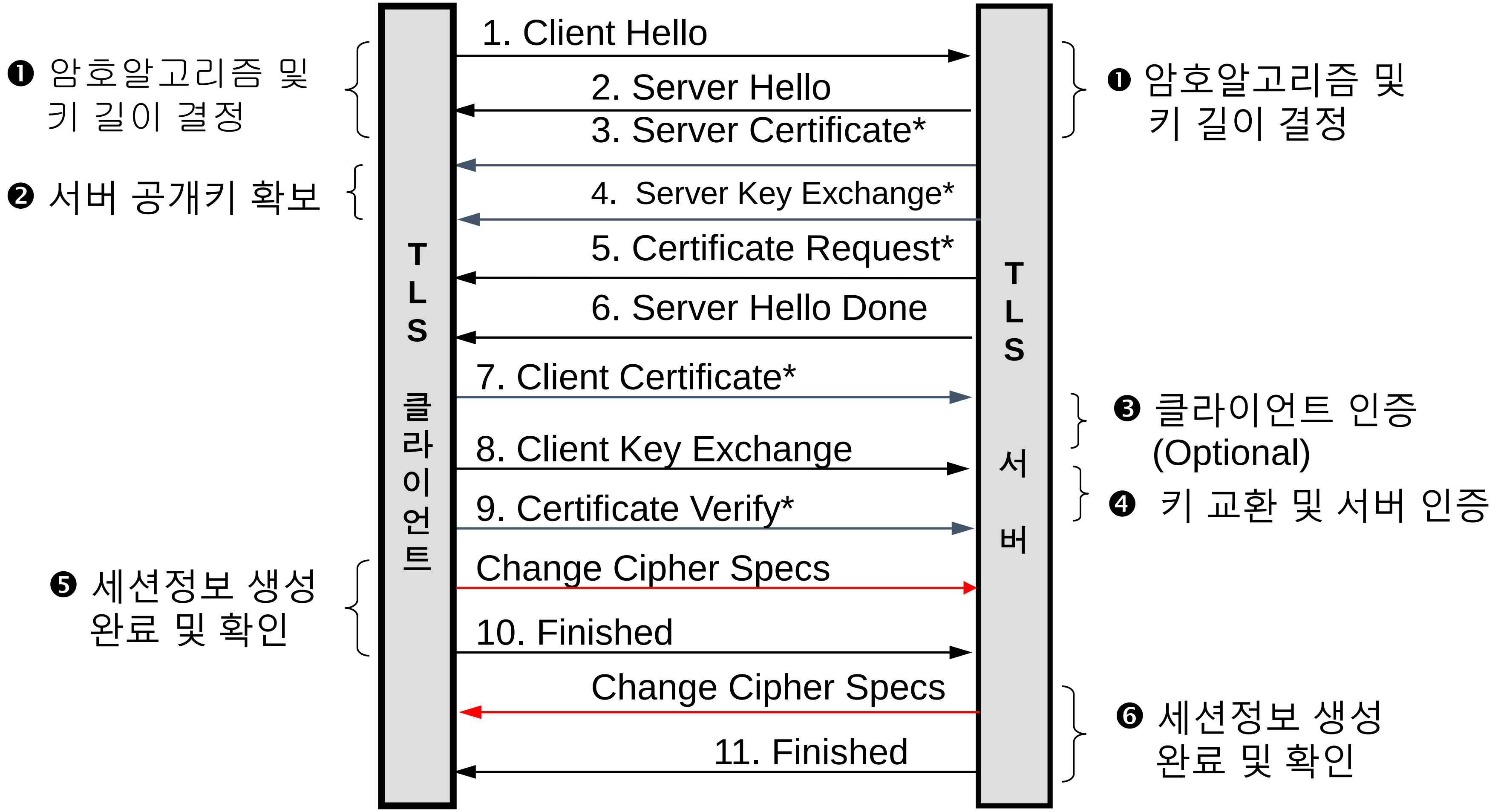
세션 ID
인증서
압축 방법
Cipher Spec
Master_Secret

Server&client random
server write MAC Secret
client write MAC Secret
server write key
client write key
Initialization Vector
sequence numbers

TLS 핸드쉐이크 프로토콜

- 핸드쉐이크의 종류
  - Full Handshake: 새로운 세션을 시작할 때
  - Abbreviated Handshake: 이전 세션을 resume하여 사용할 때
- The Handshake Protocol is responsible for negotiating a session, which consists of the following items:
  - session identifier
  - peer certificate
  - compression method
  - cipher spec
  - master secret
  - is resumable

Full Handshake



TLS 클라이언트

TLS 서버

1. Client Hello

2. Server Hello

3. Server Certificate*

4. Server Key Exchange*

5. Certificate Request*

6. Server Hello Done

7. Client Certificate*

8. Client Key Exchange

9. Certificate Verify*

Change Cipher Specs

10. Finished

Change Cipher Specs

11. Finished

❶ 암호알고리즘 및
키 길이 결정

❷ 서버 공개키 확보

❺ 세션정보 생성
완료 및 확인

❶ 암호알고리즘 및
키 길이 결정

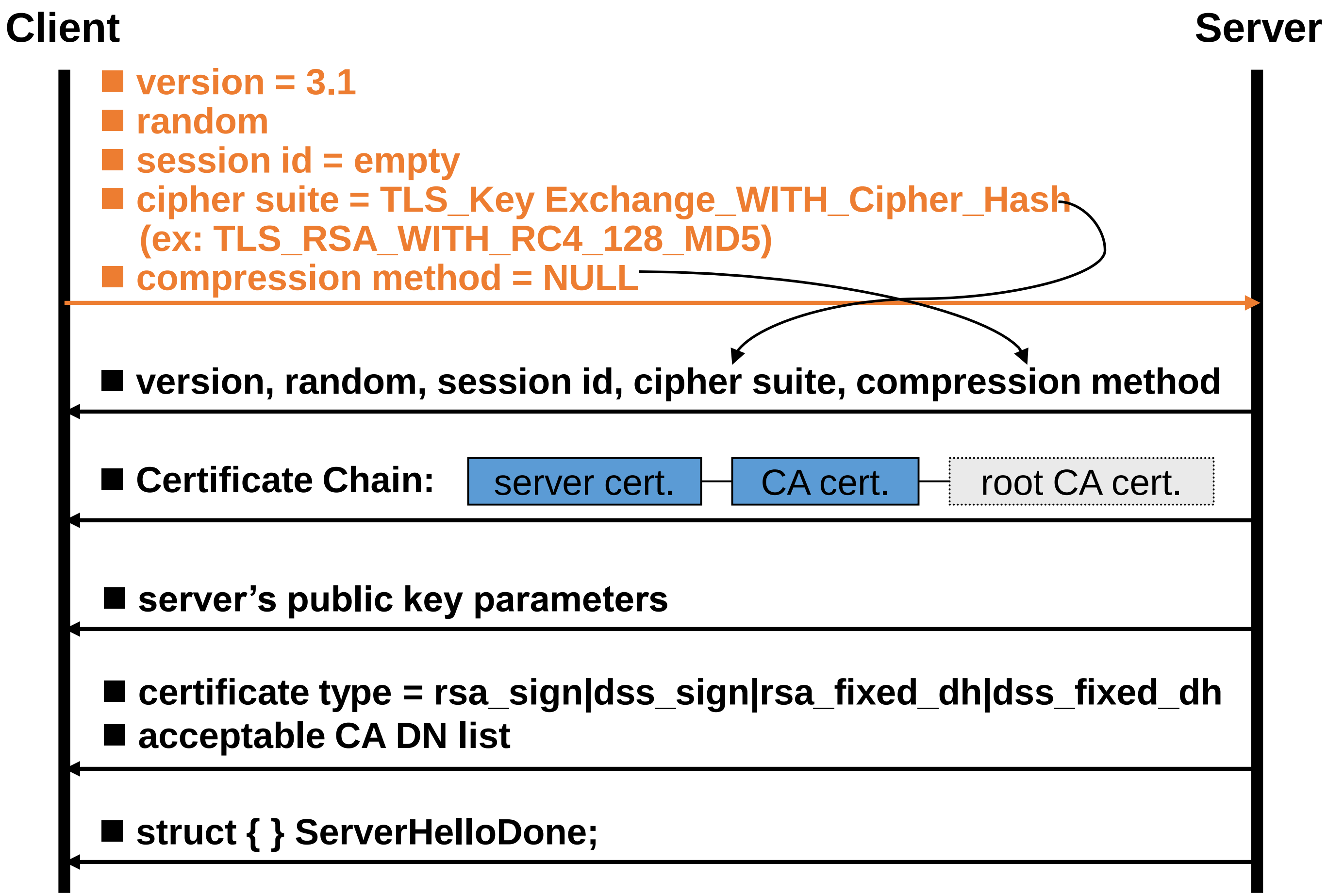
❸ 클라이언트 인증
(Optional)

❹ 키 교환 및 서버 인증

❻ 세션정보 생성
완료 및 확인

- Key Exchange Algorithms

Algorithm	Description	Key Size Limit
DHE_DSS	Ephemeral DH with DSS signature	None
DHE_DSS_EXPORT	Ephemeral DH with DSS signature	DH=512
DHE_RSA	Ephemeral DH with RSA signature	None
DHE_RSA_EXPORT	Ephemeral DH with RSA signature	DH=512, RSA=None
DH_anon	Anonymous DH, no signature	None
DH_anon_EXPORT	Anonymous DH, no signature	DH=512
DH_DSS	DH with DSS-based certificates	None
DH_DSS_EXPORT	DH with DSS-based certificates	DH=512
DH_RSA	DH with RSA-based certificates	None
DH_RSA_EXPORT	DH with RSA-based certificates	DH=512, RSA=None
NULL	No key exchange	N/A
RSA	RSA key exchange	None
RSA_EXPORT	RSA key exchange	RSA=512



Client

Server

- version = 3.1
- random
- session id = empty
- cipher suite = TLS_Key Exchange_WITH_Cipher_Hash (ex: TLS_RSA_WITH_RC4_128_MD5)
- compression method = NULL

- version, random, session id, cipher suite, compression method

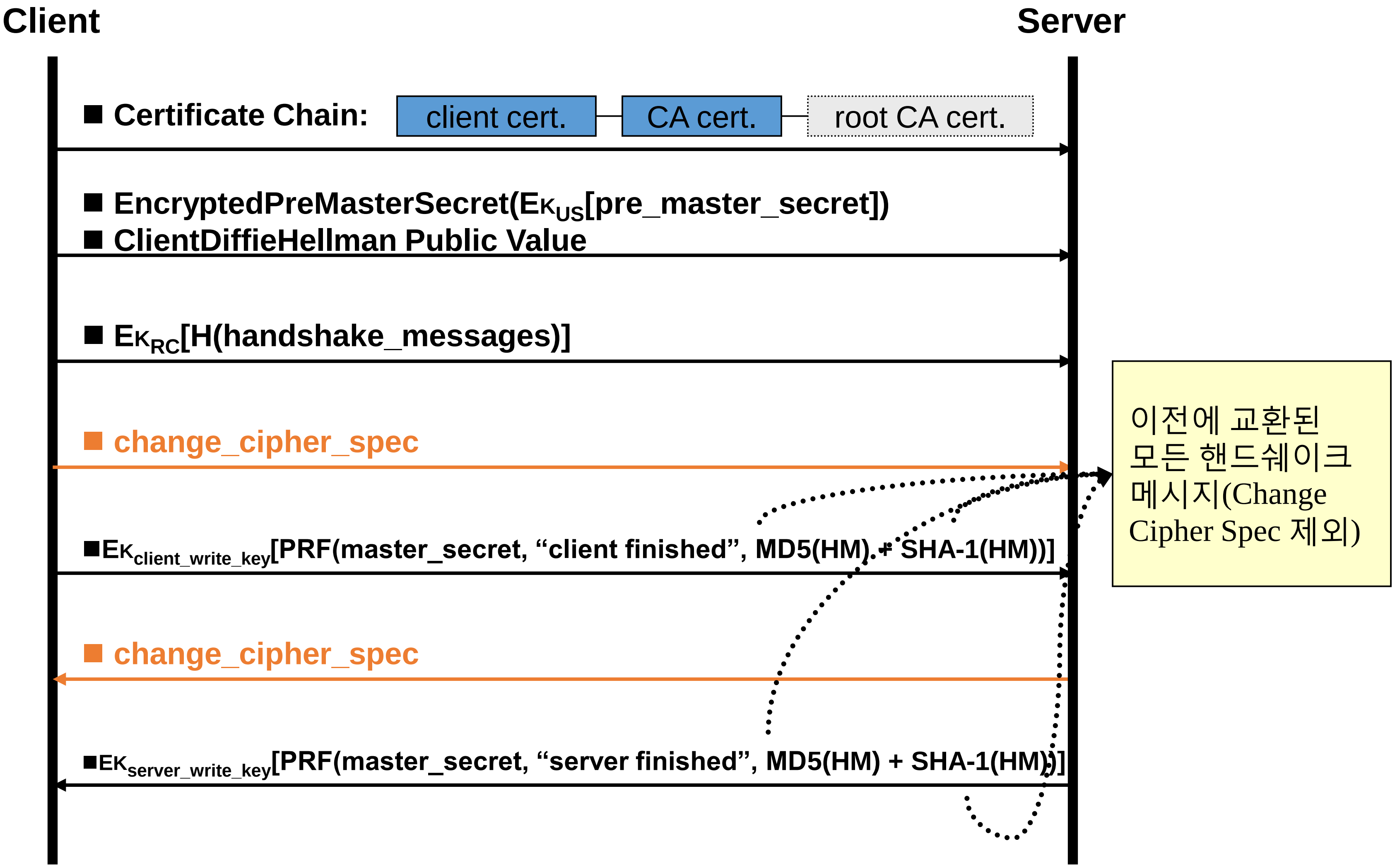
- Certificate Chain: server cert. CA cert. root CA cert.

- server’s public key parameters

- certificate type = rsa_sign|dss_sign|rsa_fixed_dh|dss_fixed_dh
- acceptable CA DN list

- struct { } ServerHelloDone;

- Cipher Suites
  - TLS_NULL_WITH_NULL_NULL*
  - TLS_RSA_WITH_NULL_MD5*
  - TLS_RSA_WITH_NULL_SHA*
  - TLS_RSA_EXPORT_WITH_RC4_40_MD5*
  - TLS_RSA_WITH_RC4_128_MD5
  - TLS_RSA_WITH_RC4_128_SHA
  - TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5*
  - TLS_RSA_WITH_IDEA_CBC_SHA
  - TLS_RSA_EXPORT_WITH_DES40_CBC_SHA*
  - TLS_RSA_WITH_DES_CBC_SHA
  - TLS_RSA_WITH_3DES_EDE_CBC_SHA
  - TLS_DH_DSS_EXPORT_WITH_DES40_CBC_SHA *
  - TLS_DH_DSS_WITH_DES_CBC_SHA
  - TLS_DH_DSS_WITH_3DES_EDE_CBC_SHA
  - TLS_DH_RSA_EXPORT_WITH_DES40_CBC_SHA*
  - TLS_DH_RSA_WITH_DES_CBC_SHA
  - TLS_DH_RSA_WITH_3DES_EDE_CBC_SHA
  - TLS_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA*
  - TLS_DHE_DSS_WITH_DES_CBC_SHA
  - TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA
  - TLS_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA*
  - TLS_DHE_RSA_WITH_DES_CBC_SHA
  - TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA
  - TLS_DH_anon_EXPORT_WITH_RC4_40_MD5*
  - TLS_DH_anon_WITH_RC4_128_MD5
  - TLS_DH_anon_EXPORT_WITH_DES40_CBC_SHA
  - TLS_DH_anon_WITH_DES_CBC_SHA
  - TLS_DH_anon_WITH_3DES_EDE_CBC_SHA



Client

Server

■ Certificate Chain: client cert. CA cert. root CA cert.

■ EncryptedPreMasterSecret(EK_{US}[pre_master_secret])

■ ClientDiffieHellman Public Value

■ EK_{RC}[H(handshake_messages)]

■ change_cipher_spec

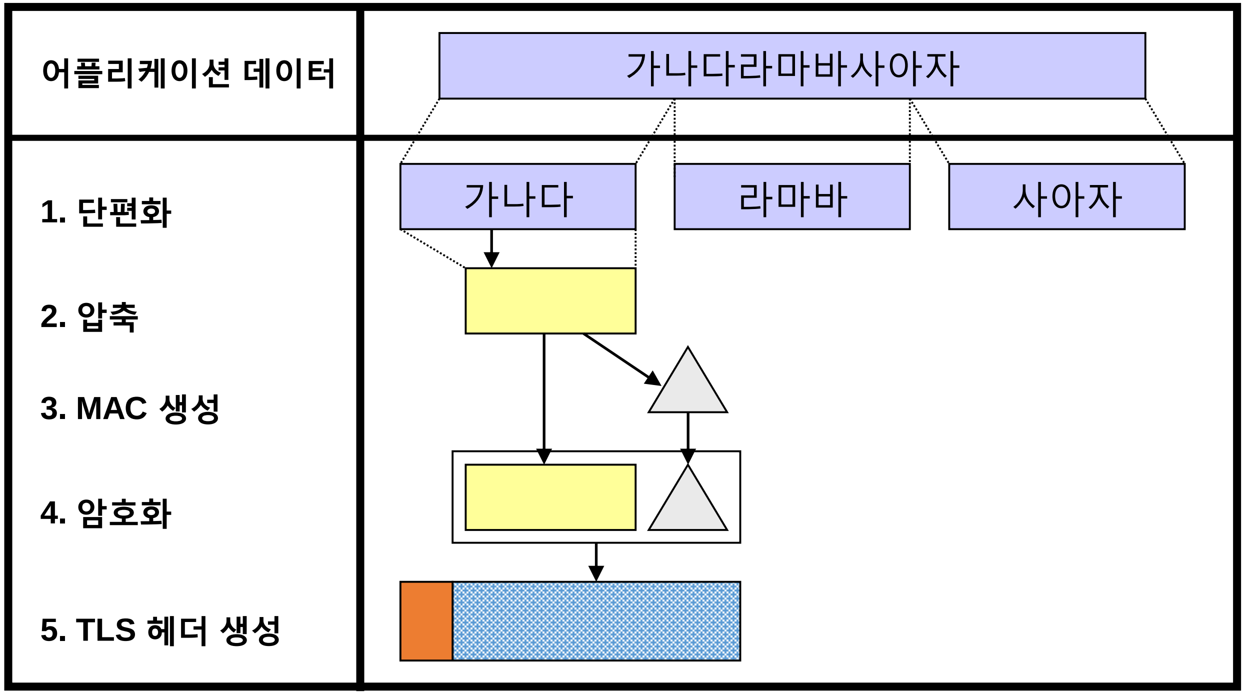
■$\mathrm{EK}_{client\_write\_key}$[PRF(master_secret, “client finished”, MD5(HM) + SHA-1(HM))]

■ change_cipher_spec

■$\mathrm{EK}_{server\_write\_key}$[PRF(master_secret, “server finished”, MD5(HM) + SHA-1(HM))]

이전에 교환된 모든 핸드쉐이크 메시지(Change Cipher Spec 제외)

레코드 프로토콜



어플리케이션 데이터

가나다라마바사아자

1. 단편화

가나다

라마바

사아자

2. 압축

3. MAC 생성

4. 암호화

5. TLS 헤더 생성

- 단편화
  - 2^{14} 바이트로 단편화
- 압축
  - SSL 3.0/TLS 1.0에서는 지원하지 않음
- MAC 생성

```
HMAC_hash(MAC_write_secret,   seq_num +
                                 TLSCompressed.type +
                                 TLSCompressed.version +
                                 TLSCompressed.length +
                                 TLSCompressed.fragment)
```

- 레코드 페이로드 생성

Content Type(1)	Major Version(1)	Minor Version(1)
Compress Length(2 bytes)		
암호화 데이터		

Cryptographic Computation

- Master Secret(48byte) 계산

master_secret = PRF(pre_master_secret, “master secret”,
ClientHello.random + ServerHello.random)

PRF(secret, label, seed) = P_MD5(S1, label + seed) XOR
P_SHA-1(S2, label + seed)

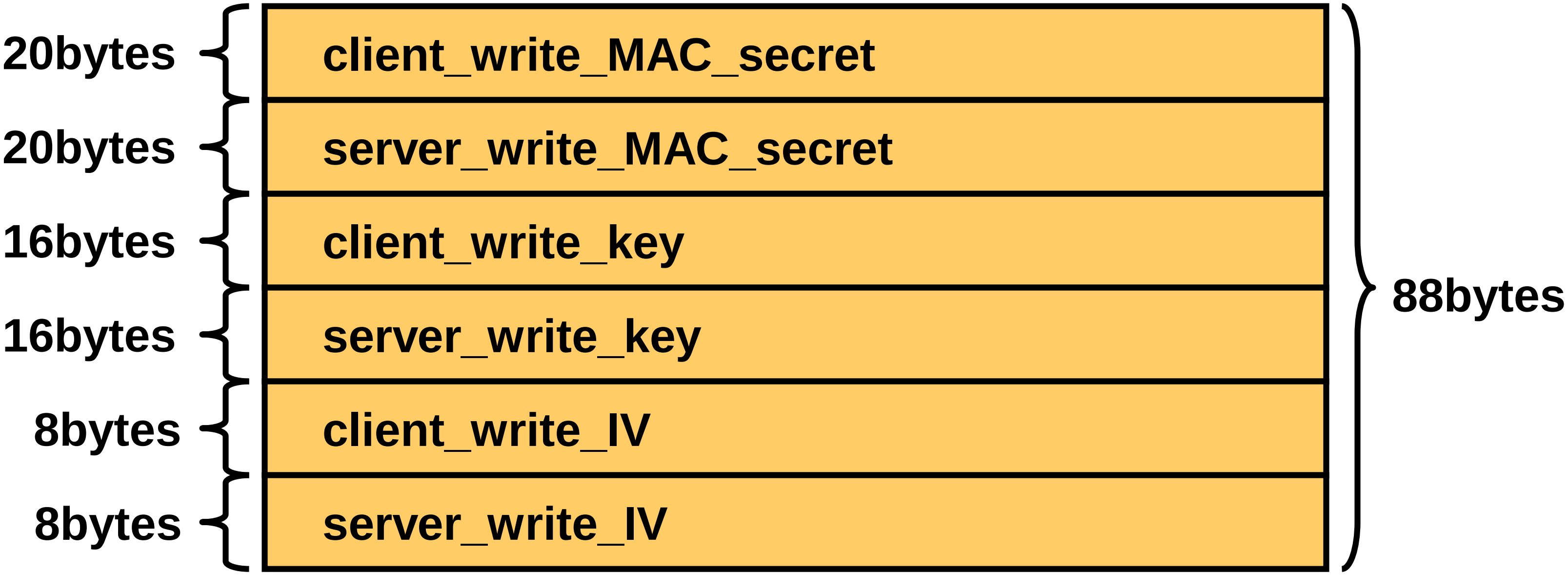
P_hash(Secret, seed) = HMAC_hash(secret, A(1) + seed) +
HMAC_hash(secret, A(2) + seed) +
HMAC_hash(secret, A(3) + seed) + ……

A(0) = seed
A(i) = HMAC_hash(secret, A(i-1))

- Key Block 생성: 필요한 길이만큼 생성

```
key_block = PRF(SecurityParameters.master_secret, “key expansion”,
                SecurityParameters.server_random +
                SecurityParameters.client_random);
```

- Key Block를 다음과 같이 분리해서 키를 생성
  - ex) IDEA_CBC_SHA



20bytes	client_write_MAC_secret
20bytes	server_write_MAC_secret
16bytes	client_write_key
16bytes	server_write_key
8bytes	client_write_IV
8bytes	server_write_IV

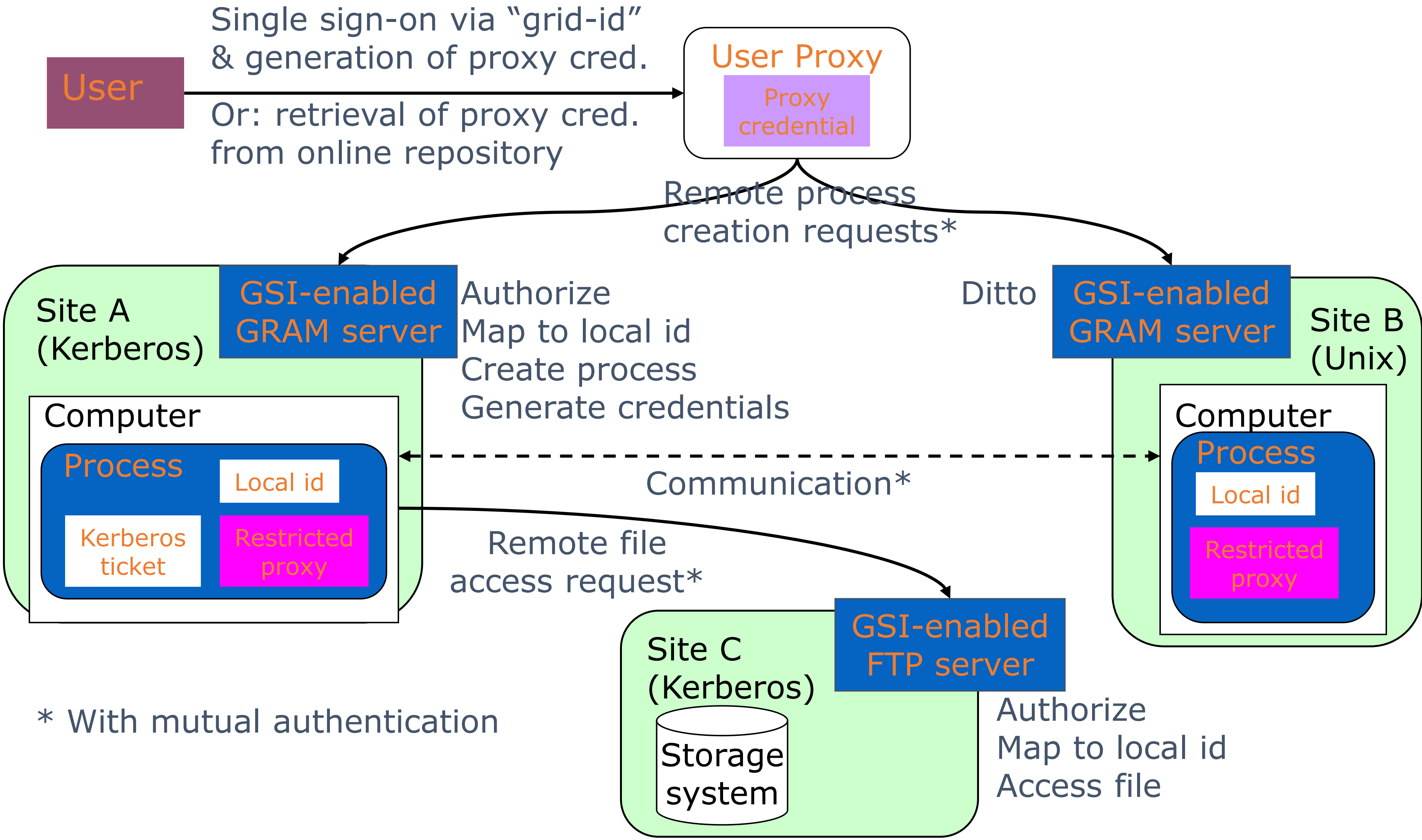
88bytes

5. 결론

- The Grid Security Infrastructure (GSI) is a set of tools, libraries and protocols used in Globus to allow users and applications to securely access resources.
- Based on a public key infrastructure, with certificate authorities and X509 certificates
- Uses SSL for authentication and message protection
- Adds features needed for Single-Sign on
  - Proxy Credentials
  - Delegation
- In the GSI system each user has a set of credentials they use to prove their identity on the grid
  - Consists of a X509 certificate and private key
- Long-term private key is kept encrypted with a pass phrase
  - Good for security, inconvenient for repeated usage
- Single-sign on is important feature for Grid Applications
  - Enables easy coordination of multiple resources
  - User authenticates themselves once, then can perform multiple actions without re-authentication
  - Can allow processes to act on their behalf
- To support single sign-on GSI adds the following functionality to SSL:
  - Proxy credentials
  - Credential delegation

GSI in Action

“Create Processes at A and B that Communicate & Access Files at C”



User

Single sign-on via “grid-id”
& generation of proxy cred.

Or: retrieval of proxy cred.
from online repository

User Proxy

Proxy credential

Remote process
creation requests*

Site A
(Kerberos)

GSI-enabled
GRAM server

Authorize
Map to local id
Create process
Generate credentials

Ditto

GSI-enabled
GRAM server

Site B
(Unix)

Computer

Process

Local id

Kerberos ticket

Restricted proxy

Communication*

Computer

Process

Local id

Restricted proxy

Remote file
access request*

Site C
(Kerberos)

GSI-enabled
FTP server

Storage system

Authorize
Map to local id
Access file

* With mutual authentication

Thank
You
köszönöm
Asante
Shukria
Vinaka
Kiitos
Maake
Dhanyavadagalu
감사합니다
Dankscheen
спасибо
Kam Sah Hammida
ارك شكر
Manana
Dankon
Terima Kasih
Matondo
Dank Je
Mauruuru
Biyan
Diolch i Chi
Taiku
Tack
Blagodaram
Dziekuje
Ngiyabonga
Chokrane
Grazie
Juspaxar
Arigato
Mochchakkeram
Gracias
Khap
Paldies
Tingki
நன்றி
Bedankt
Dakujem
धन्यवाद
cảm ơn bạn
Ua Tsaug Rau Koj
Grazas
Mèsi
Gratias Tibi
Obrigado
Kia Ora
Kop Khun
Děkuji
Nuwun
Nirringrazzjak
Di Ou
ありがとう
Suksama
Rahmat
Hvala
Matur
谢谢
хвала
Welalin
Djiere Dieuf
Misaotra
Danke
Merci
Go Raibh Maith Agat
Tuke
Eskerrik Asko
Salamat
ขอขอบคุณคุณ
Najjis