Lecture 4: Network Security Principles

-COMP 6712 Advanced Security and Privacy

Haiyang Xue

haiyang.xue@polyu.edu.hk

2024/2/5

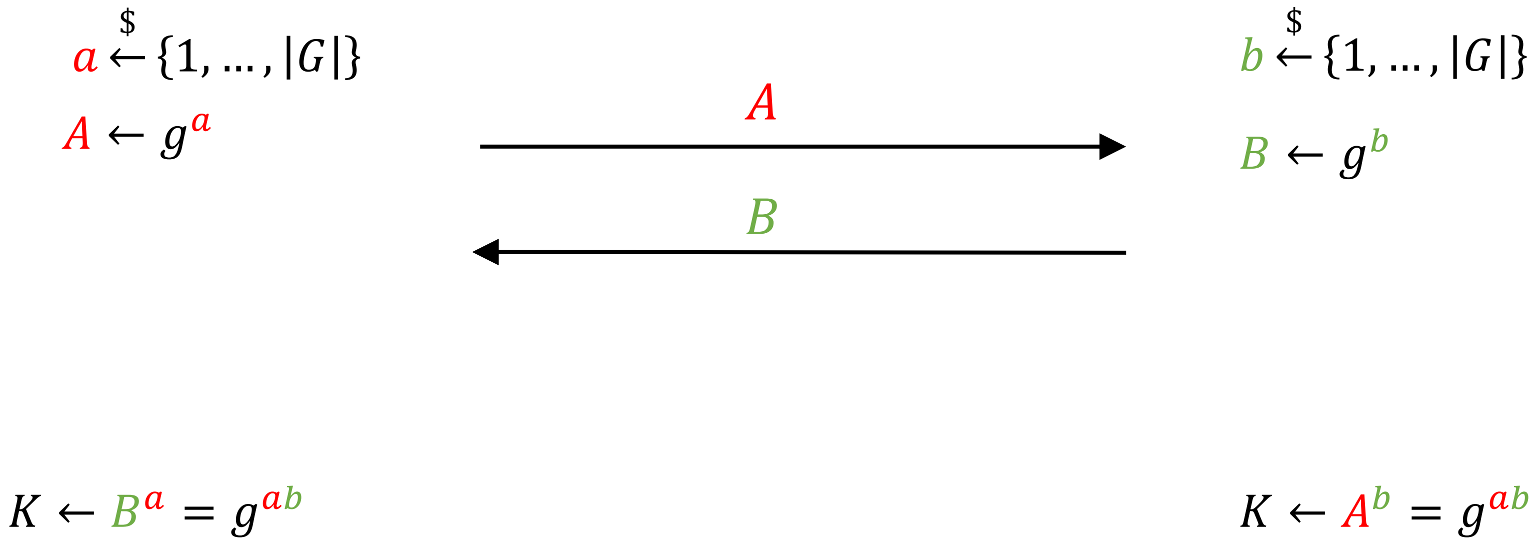
Network Security Principles

- Recall RSA and Digital Signature
- Authenticated Key Exchange
- Public Key Infrastructure(PKI)
- and Certification Authorities

Public key encryption

Diffie-Hellman

$$G = \langle g \rangle$$



$a \overset{\$}{\leftarrow} \{1, \dots, |G|\}$

$A \leftarrow g^{a}$

A

B

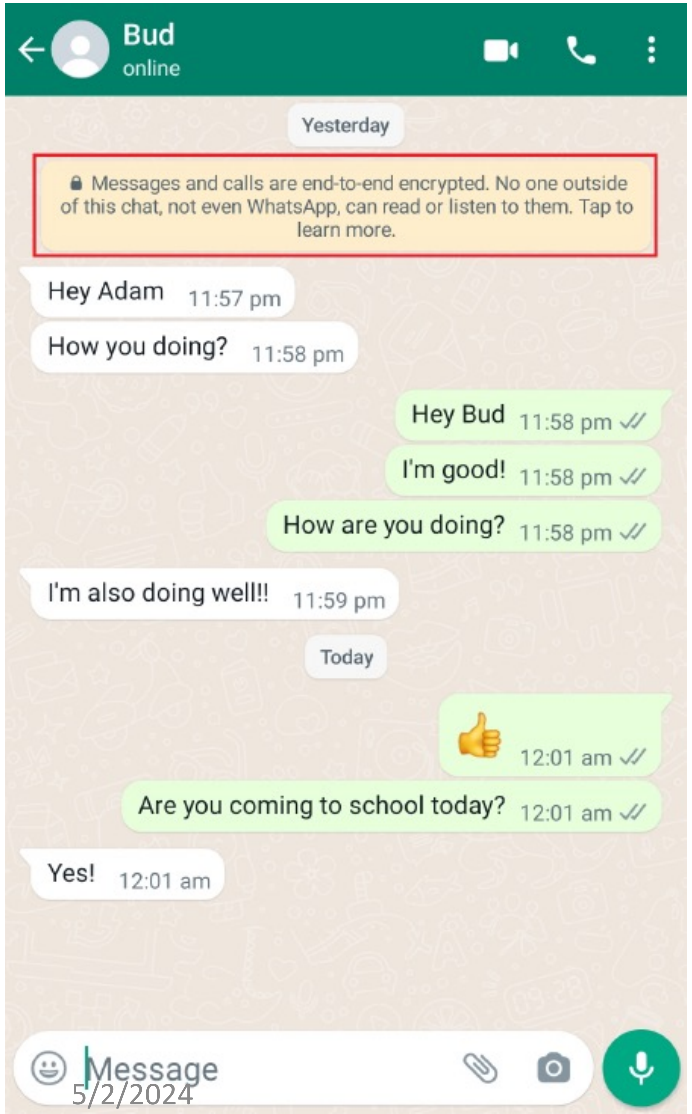
$b \overset{\$}{\leftarrow} \{1, \dots, |G|\}$

$B \leftarrow g^{b}$

$K \leftarrow B^{a} = g^{ab}$

$K \leftarrow A^{b} = g^{ab}$

Ratchet Diffie-Hellman in WhatsApp and Signal



Bud
online

Yesterday

🔒 Messages and calls are end-to-end encrypted. No one outside of this chat, not even WhatsApp, can read or listen to them. Tap to learn more.

Hey Adam 11:57 pm

How you doing? 11:58 pm

Hey Bud 11:58 pm ✓✓

I'm good! 11:58 pm ✓✓

How are you doing? 11:58 pm ✓✓

I'm also doing well!! 11:59 pm

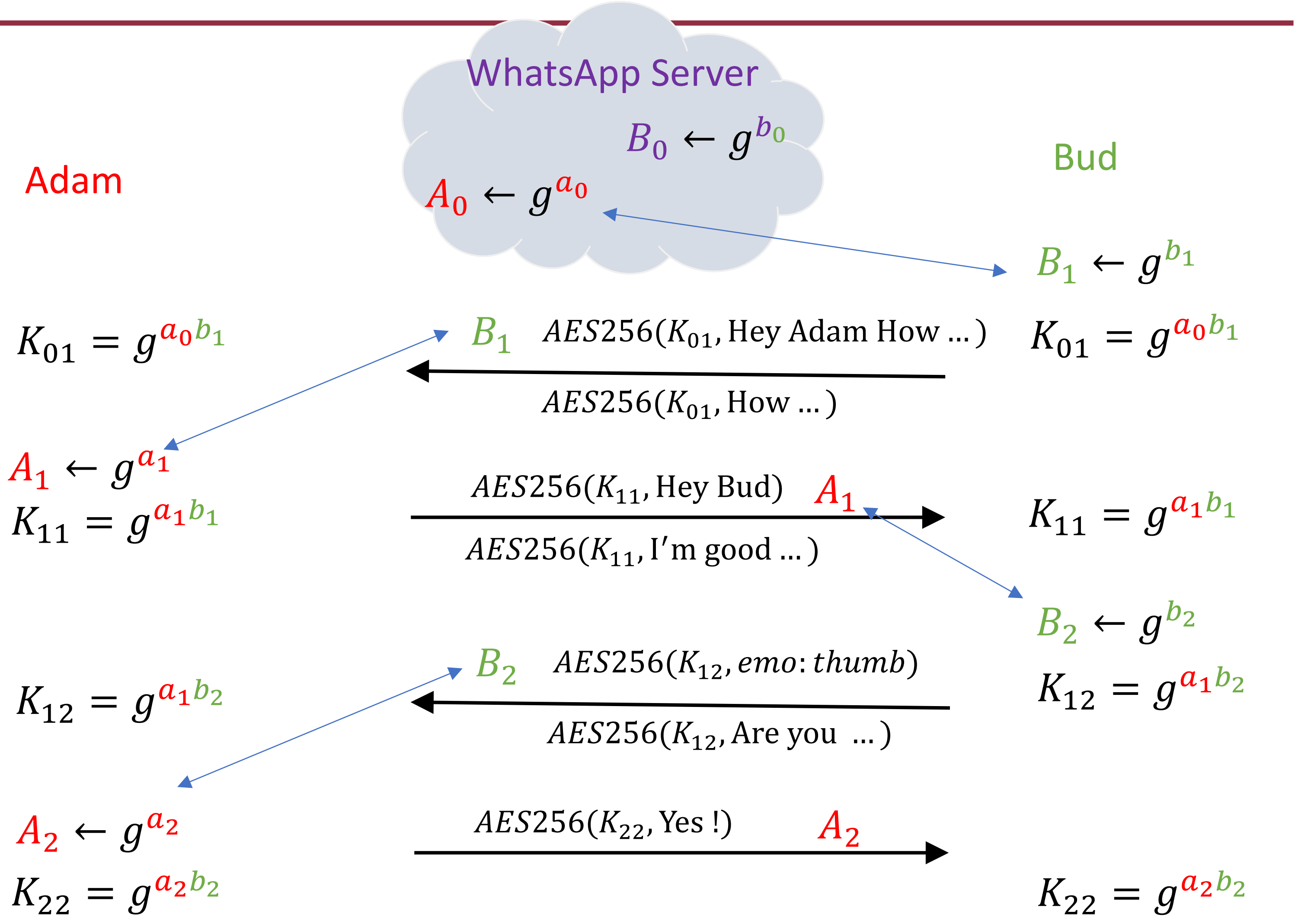
Today

👍 12:01 am ✓✓

Are you coming to school today? 12:01 am ✓✓

Yes! 12:01 am

Message



WhatsApp Server

$B_0 \leftarrow g^{b_0}$

$A_0 \leftarrow g^{a_0}$

Adam

Bud

$B_1 \leftarrow g^{b_1}$

$K_{01} = g^{a_0 b_1}$

B_1 $AES256(K_{01}, \text{Hey Adam How ...})$

$AES256(K_{01}, \text{How ...})$

$K_{01} = g^{a_0 b_1}$

$A_1 \leftarrow g^{a_1}$

$K_{11} = g^{a_1 b_1}$

$AES256(K_{11}, \text{Hey Bud})$ A_1

$AES256(K_{11}, \text{I'm good ...})$

$K_{11} = g^{a_1 b_1}$

$B_2 \leftarrow g^{b_2}$

$K_{12} = g^{a_1 b_2}$

B_2 $AES256(K_{12}, emo: thumb)$

$AES256(K_{12}, \text{Are you ...})$

$K_{12} = g^{a_1 b_2}$

$A_2 \leftarrow g^{a_2}$

$K_{22} = g^{a_2 b_2}$

$AES256(K_{22}, \text{Yes !})$ A_2

$K_{22} = g^{a_2 b_2}$

ElGamal

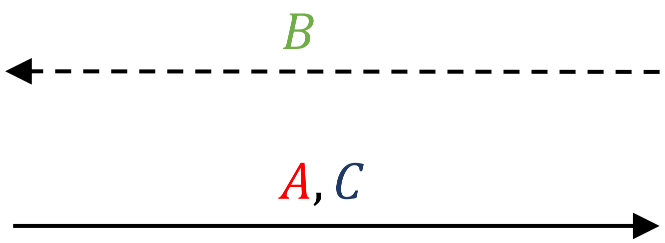
$\mathrm{ElGamal.Enc}: G \times G \to \quad G \times \mathcal{C}$

$\mathrm{ElGamal.Dec}: \boldsymbol{Z}_p \times G \times G \to \quad G$

$$G = \langle g \rangle$$

KeyGen

```
$sk = b \overset{\$}{\leftarrow} \{1, \dots, |G|\}$
$pk = B \leftarrow g^b$
return $(sk, pk)$
```



Enc(pk, M)

```
$a \overset{\$}{\leftarrow} \{1, \dots, |G|\}$
$A \leftarrow g^a$
$K \leftarrow B^a = g^{ab}$
$C \leftarrow K \cdot M$
return $(A, C)$
```

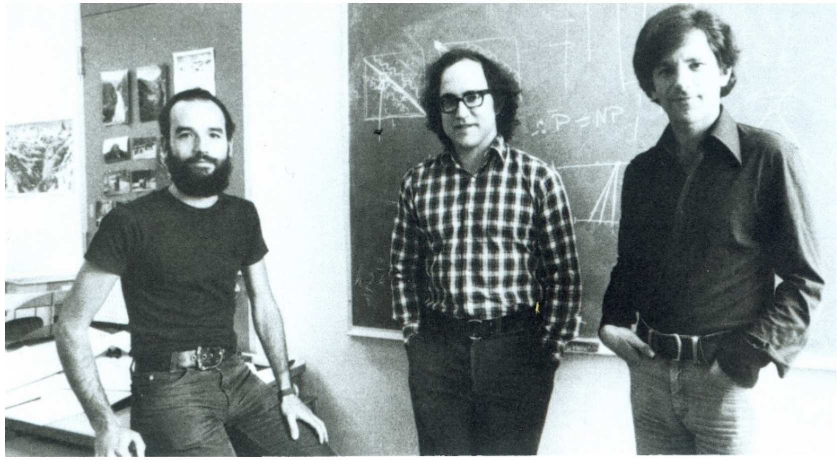
Dec(sk, C)

```
$Z \leftarrow A^b = g^{ab}$
$M \leftarrow C/K$
return $M$
```

RSA in 1977

- The RSA encryption scheme

$$c = E\,(m) = m^e \pmod{n}$$



Adi Shamir Ron Rivest Leonard Adleman

Euler's Theorem

Theorem: if $(G,\circ)$ is a finite group, then for all $g \in G$:

$$g^{|G|} = e$$

- $(\boldsymbol{Z}_p^*,\cdot)$: $|\boldsymbol{Z}_p^*| = (p-1) \quad e = 1$

Fermat's theorem: if p is prime, then for all $a \neq 0 \pmod p$:

$$a^{p-1} \equiv 1 \pmod p$$

- $(\boldsymbol{Z}_n^*,\cdot)$: $|\boldsymbol{Z}_n^*| = \phi(n) \quad e = 1$

Euler's theorem: for all positive integers n, if $\gcd(a,n) = 1$ then

$$a^{\phi(n)} \equiv 1 \pmod n$$

Structure for RSA

$n \leftarrow p \cdot q$ $\qquad\qquad \phi(n) = (p-1)(q-1)$

$$\boldsymbol{Z}_n^* = \underbrace{\text{invertible elements in } \boldsymbol{Z}_n}_{(\boldsymbol{Z}_n^*, \cdot) \textit{ is} \text{ a group of order } \phi(n)!} = \{\, a \in \boldsymbol{Z}_n \mid \gcd(a, n) = 1 \,\}$$

$$a^{\phi(n)} \equiv 1 \pmod n \qquad ed = 1 \bmod \phi(n)$$

RSA Enc

$$\begin{tikzcd} M \arrow[r, "\textbf{Enc}"] & C \leftarrow M^e \bmod n \arrow[r, "\textbf{Dec}"] & M \leftarrow C^d \bmod n \end{tikzcd}$$

Textbook RSA

$$\mathrm{RSA.Enc}:\ \overbrace{\mathbb{Z}^{+}\times\mathbb{Z}^{*}_{\phi(n)}}^{\mathcal{PK}}\times\overset{\mathcal{M}}{\mathbb{Z}^{*}_{n}}\rightarrow\overset{\mathcal{C}}{\mathbb{Z}^{*}_{n}}$$

$$\mathrm{RSA.Dec}:\ \underset{\mathcal{SK}}{\mathbb{Z}^{*}_{\phi(n)}}\times\underset{\mathcal{C}}{\mathbb{Z}^{*}_{n}}\rightarrow\underset{\mathcal{M}}{\mathbb{Z}^{*}_{n}}$$

```
Enc($pk = (n, e), M \in \mathbb{Z}^*_n$)
    $C \leftarrow M^e \mod n$
    return $C$
```

$$\xleftarrow{pk}$$

$$\xrightarrow{C}$$

Common choices of e: 3, 17, 65 537

11_2 10001_2 $1\,0000\,0000\,0000\,0001_2$

```
KeyGen
    $p, q \overset{\$}{\leftarrow}$ two random prime numbers
    $n \leftarrow p \cdot q$
    $\phi(n) = (p-1)(q-1)$
    choose $e$ such that $\gcd(e, \phi(n)) = 1$
    $d \leftarrow e^{-1} \mod \phi(n)$
    $sk \leftarrow d$    $pk \leftarrow (n, e)$
    return $(sk, pk)$
```

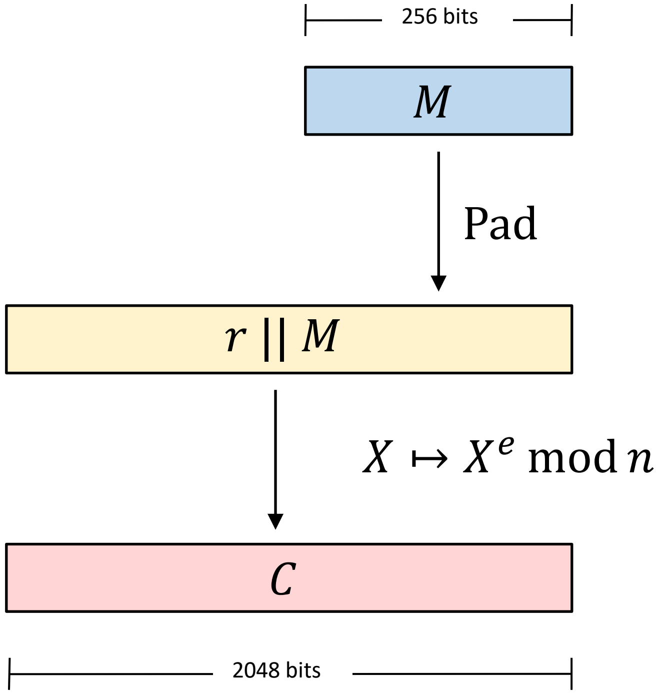
```
Dec($sk = d, C \in \mathbb{Z}^*_n$)
    $M \leftarrow C^d \mod n$
    return $M$
```

RSA in practice

- Textbook RSA is deterministic ⟹ cannot be IND-CPA secure

- How to achieve IND-CPA, IND-CCA?
  - *pad* message with random data before applying RSA function
  - PKCS#1v1.5 (RFC 2313)
  - RSA-OAEP (RFC 8017)

- Do not use Textbook RSA
- RSA encryption not used much in practice anymore



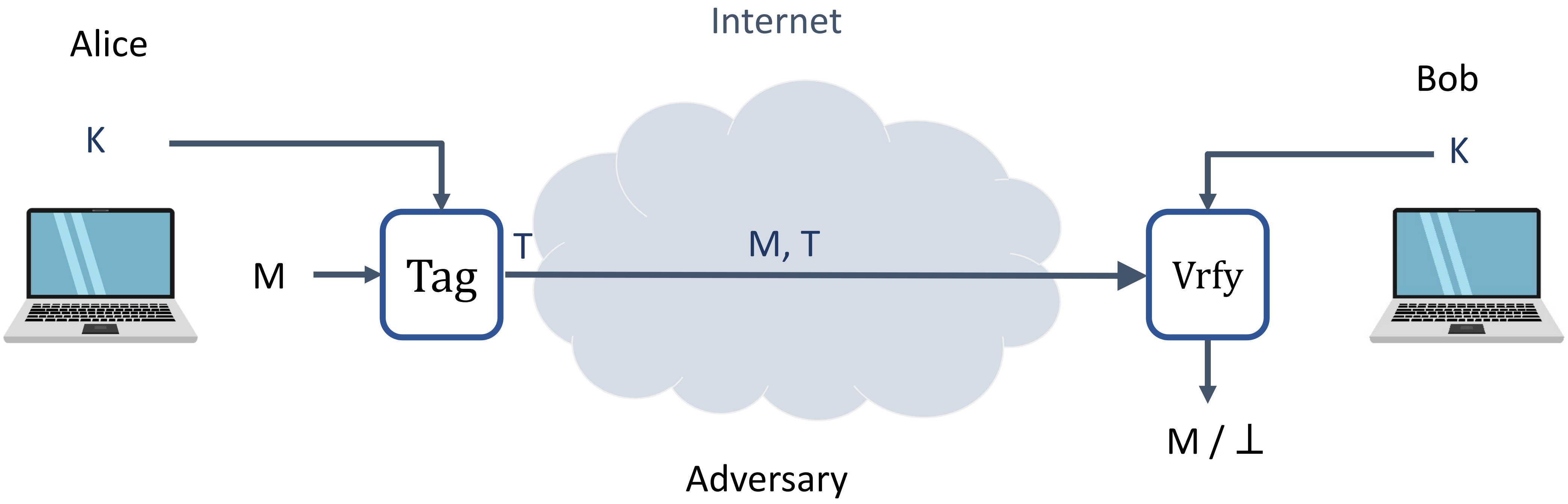
A short summary

- We can build IND-CPA secure ElGamal scheme based on DDH assumption

- Padding with randomness, we can transfer Textbook RSA to IND-CPA scheme

Digital Signature

Achieving integrity: MACs

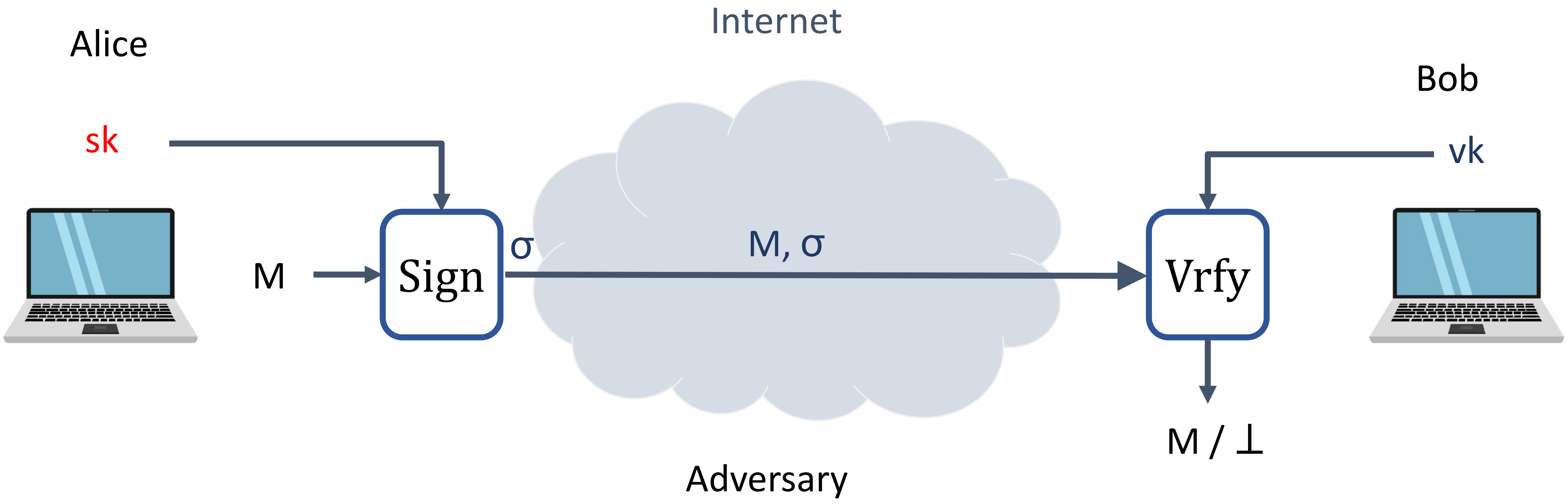


Tag : tagging algorithm (public)

K : tagging / verification key (secret)

Vrfy: verification algorithm (public)

Achieving integrity: digital signatures



Sign : tagging algorithm (public)

sk : signing key (secret)

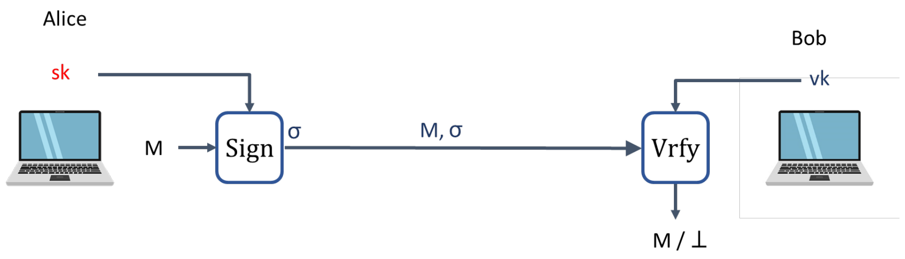
Vrfy : verification algorithm (public)

vk : verification key (public)

Digital signatures vs. MACs

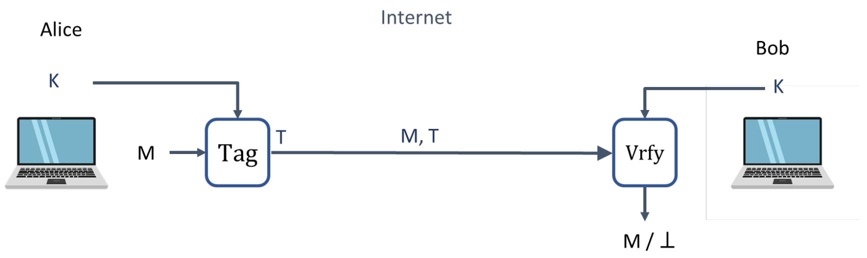
- Digital signatures can be verified by *anyone*

Digital signature



- MACs can only be verified by party sharing the same key

MAC



- **Non-repudiation:** Alice cannot deny having created σ
  - But she can deny having created T (since Bob could have done it)

Digital signatures – syntax

A **digital signature** scheme is a tuple of algorithms $\Sigma = (\mathrm{KeyGen}, \mathrm{Sign}, \mathrm{Vrfy})$

$\mathrm{KeyGen} : (\,) \to \mathcal{SK} \times \mathcal{VK}$

$\mathrm{Sign} : \mathcal{SK} \times \mathcal{M} \to \mathcal{S}$

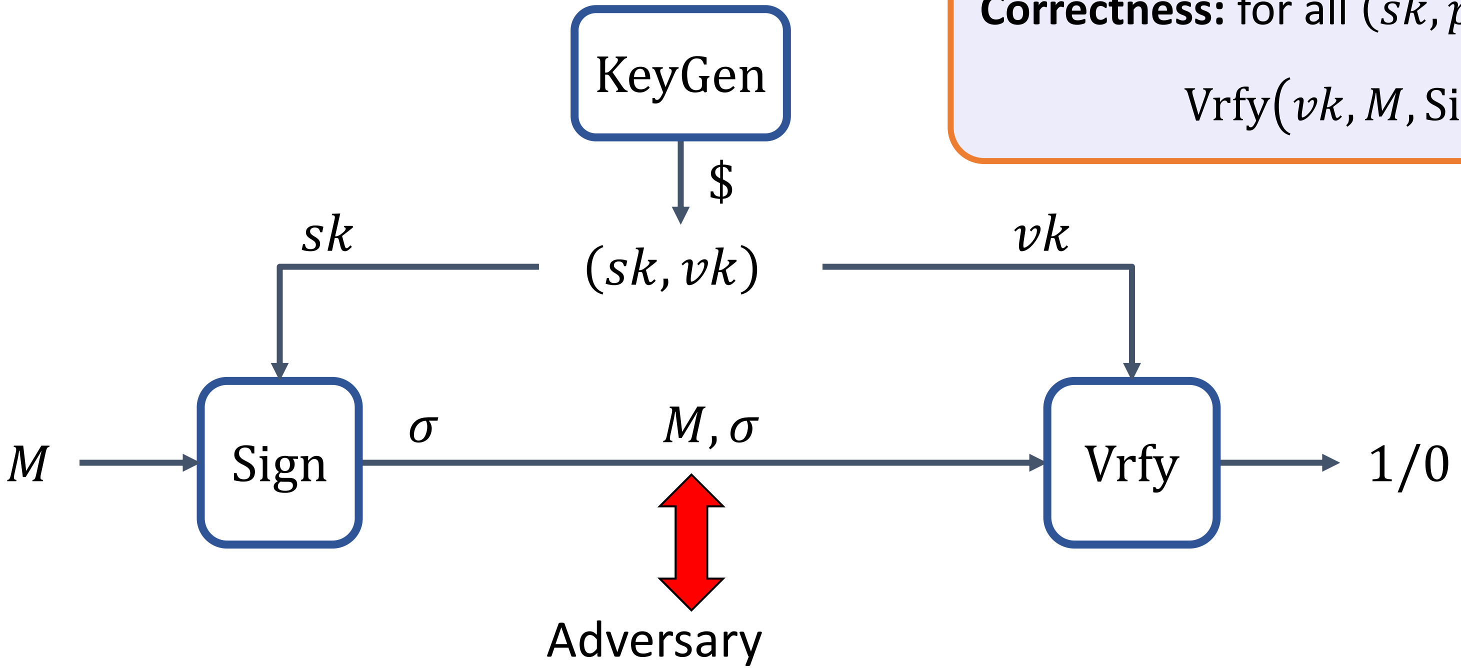
$\mathrm{Sign}(sk, M) = \mathrm{Sign}_{sk}(M) = \sigma$

$\mathrm{Vrfy} : \mathcal{VK} \times \mathcal{M} \times \mathcal{S} \to \{0,1\}$

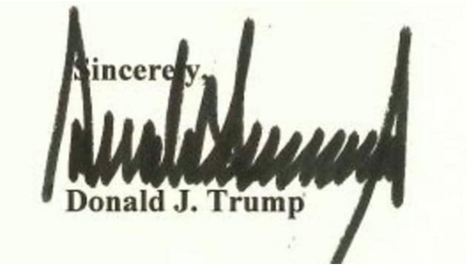
$\mathrm{Vrfy}(vk, M, \sigma) = \mathrm{Vrfy}_{vk}(M, \sigma) = 1/0$

Correctness: for all $(sk, pk) \leftarrow \mathrm{KeyGen}$:

$$\mathrm{Vrfy}\big(vk, M, \mathrm{Sign}(sk, M)\big) = 1$$



Signature: unforgeability


Sincerely,
Donald J. Trump

unforgeable

$\sigma_1, \sigma_2, \ldots$

unforgeable

σ^*

$n \leftarrow p \cdot q$ $\qquad\qquad \phi(n) = (p-1)(q-1)$

$$\boldsymbol{Z}_n^* = \underbrace{\text{invertible elements in } \boldsymbol{Z}_n}_{(\boldsymbol{Z}_n^*, \cdot) \textit{ is} \text{ a group of order } \phi(n)!} = \{\, a \in \boldsymbol{Z}_n \mid \gcd(a, n) = 1 \,\}$$

$$a^{\phi(n)} \equiv 1 \pmod{n} \qquad ed = 1 \bmod \phi(n)$$

$$\begin{tikzcd}
\text{RSA Enc} & M \arrow[r, "\textbf{Enc}"] & C \leftarrow M^e \bmod n \arrow[r, "\textbf{Dec}"] & M \leftarrow C^d \bmod n \\
\text{RSA Sign} & M = \sigma^e & \sigma \leftarrow M^d \bmod n \arrow[l, "\textbf{Vrfy}"'] & M \arrow[l, "\textbf{Sign}"']
\end{tikzcd}$$

Textbook RSA signatures

$$\text{RSA.Sign: } \overbrace{\mathbf{Z}^+ \times \mathbf{Z}^*_{\phi(n)}}^{\mathcal{SK}} \times \overset{\mathcal{M}}{\mathbf{Z}^*_n} \rightarrow \overset{\mathcal{S}}{\mathbf{Z}^*_n}$$

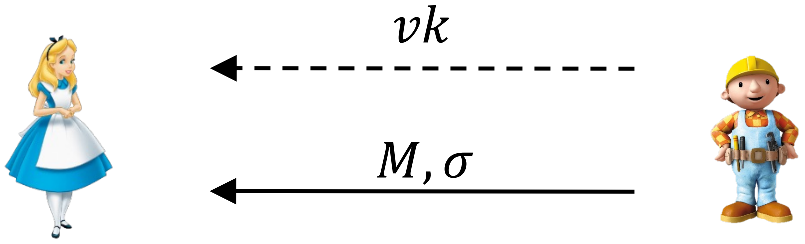
$$\text{RSA.Vrfy: } \underbrace{\mathbf{Z}^+ \times \mathbf{Z}^*_{\phi(n)}}_{\mathcal{PK}} \times \underset{\mathcal{M}}{\mathbf{Z}^*_n} \times \underset{\mathcal{S}}{\mathbf{Z}^*_n} \rightarrow \{1,0\}$$

KeyGen

```
$p, q \overset{\$}{\leftarrow}$ two random prime numbers
$n \leftarrow p \cdot q$
$\phi(n) = (p-1)(q-1)$
**choose** $e$ such that $\gcd(e, \phi(n)) = 1$
$d \leftarrow e^{-1} \bmod \phi(n)$
$sk \leftarrow (n, d)$      $vk \leftarrow (n, e)$
**return** $(sk, vk)$
```

Vrfy$(vk = (n, e), M \in \mathbf{Z}^*_n, \sigma)$

```
if $\sigma^e = M \bmod n$ then
    return 1
else
    return 0
```



Sign$(sk = (n, d), M \in \mathbf{Z}^*_n)$

```
$\sigma \leftarrow M^d \bmod n$
**return** $\sigma$
```

$$d = e^{-1} \bmod \phi(n) \iff ed = 1 \bmod \phi(n)$$

$$\sigma^e = M^{de} = M^{ed \bmod \phi(n)} = M^1 = M \bmod n$$

Insecurity of Textbook RSA signature

Given $\sigma_1 = M_1^d, \sigma_2 = M_2^d$

$\sigma_1 \sigma_2 = (M_1 M_2)^d mod\ n$ is a signature of $M_1 M_2\ mod\ n$

Many other attacks exist

RSA-FDH:Hash-then sign paradigm

$$\text{RSA.Sign}: \overbrace{\boldsymbol{Z}^+ \times \boldsymbol{Z}^*_{\phi(n)}}^{\mathcal{SK}} \times \overset{\mathcal{M}}{\{0,1\}^*} \to \overset{\mathcal{S}}{\boldsymbol{Z}^*_n}$$

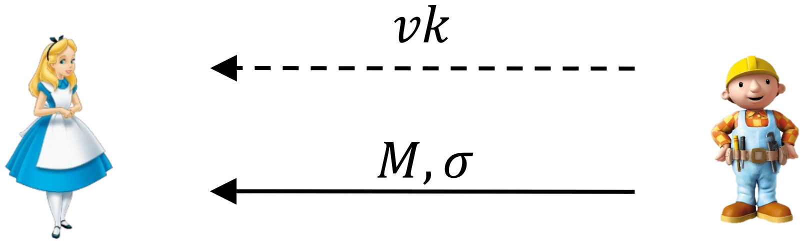
$$\text{RSA.Vrfy}: \underbrace{\boldsymbol{Z}^+ \times \boldsymbol{Z}^*_{\phi(n)}}_{\mathcal{PK}} \times \underset{\mathcal{M}}{\{0,1\}^*} \times \underset{\mathcal{S}}{\boldsymbol{Z}^*_n} \to \{1,0\}$$

KeyGen

```
$p, q \overset{\$}{\leftarrow}$ two random prime numbers
$n \leftarrow p \cdot q$
$\phi(n) = (p-1)(q-1)$
choose $e$ such that $\gcd(e, \phi(n)) = 1$
$d \leftarrow e^{-1} \bmod \phi(n)$
$sk \leftarrow (n, d)$      $vk \leftarrow (n, e)$
return $(sk, vk)$
```

Vrfy$(vk = (n, e), M \in \boldsymbol{Z}^*_n, \sigma)$

```
if $\sigma^e = H(M) \bmod n$ then
    return 1
else
    return 0
```



$$H : \{0,1\}^* \to \boldsymbol{Z}^*_n$$

Sign$(sk = (n, d), M \in \boldsymbol{Z}^*_n)$

```
$\sigma \leftarrow H(M)^d \bmod n$
return $\sigma$
```

RSA-FDH: Hash-then sign paradigm

Theorem: For *any* UF-CMA adversy A against hashed RSA making q $\mathrm{SIGN}_{sk}(\cdot)$ queries, there is an algorithm B solving the RSA-problem:

$$\mathbf{Adv}^{\mathrm{uf\text{-}cma}}_{\mathrm{RSA},\,H}(A) \leq q \cdot \mathbf{Adv}^{\mathrm{RSA}}_{n,e}(B)$$

where H is assumed perfect*

* H is assumed to be random oracle, which is out of the scope of this course. Refer to [KL] Section 12.4 for the formal proof

From the view of attack

Given $\sigma_1 = H(M_1)^d, \sigma_2 = H(M_2)^d$

$\sigma_1\sigma_2 = (H(M_1)H(M_2))^d mod\ n$ is a signature of some m??

Find m such that $\mathrm{H(m) = H(M_1)H(M_2)}$!!!!! One-wayness of H

Digital signature using in practice

- RSA signature
  - RSAwithSHA-256,382,512 (PKCS #1 V2.1, RFC 6594)

- ECDSA signature
  - ECDSA256,384,512 (NIST FIPS 186-4)
  - EdDSA (RFC 6979)

- Schnorr signature

A short summary

- Hash-then sign paradigm of RSA gives a secure signature
- There are Discrete-log-based signatures, ECDSA, and Schnorr

Primitive	Functionality + syntax	Hardness assumption	Security	Examples
Diffie-Hellman	Derive shared value (key) in a cyclic group $A^b = g^{ab} = B^a$	Discrete logarithm (DLOG) Decisional Diffie-Hellman (DDH)		$(\boldsymbol{Z}_p^*, \cdot)$ −DH $(E(\boldsymbol{F}_p), +)$ −DH
RSA function	One-way trapdoor function/permutation	Factoring problem RSA-problem		Textbook RSA
Public-key encryption	Encrypt variable-length input $\mathrm{Enc} : \mathcal{PK} \times \mathcal{M} \rightarrow \mathcal{C}$	Decisional Diffie-Hellman (DDH) Factoring problem RSA-problem	IND-CPA IND-CCA	ElGamal Padded RSA
Digital signatures	$\mathrm{Sign} : \mathcal{SK} \times \mathcal{M} \rightarrow \mathcal{S}$ $\mathrm{Vrfy} : \mathcal{VK} \times \mathcal{M} \times \mathcal{S} \rightarrow \{1,0\}$	RSA-problem Discrete logarithm (DLOG)	UF-CMA	Hashed-RSA ECDSA Schnorr

Assignment 1 (Deadline 10 March)

- Implement the ElGamal Enc algorithm in Sage
  - submit the code
  - Provide “known answer-test” (KAT) values （i.e., example of pk, sk, m and c）

- Implement the Textbook RSA signature in Sage
  - submit the code
  - And show the attack that if $\sigma_1 = M_1^d, \sigma_2 = M_2^d$, then $\sigma_1\ \sigma_2$ is the Textbook RSA signature of $M_1\ M_2$
  - Provide “known answer-test” (KAT) values (i.e., example of vk=(n, e), sk=d, m and σ）

- Write a report about the algorithms and implementation
- Assignment 1 will be available on the blackboard

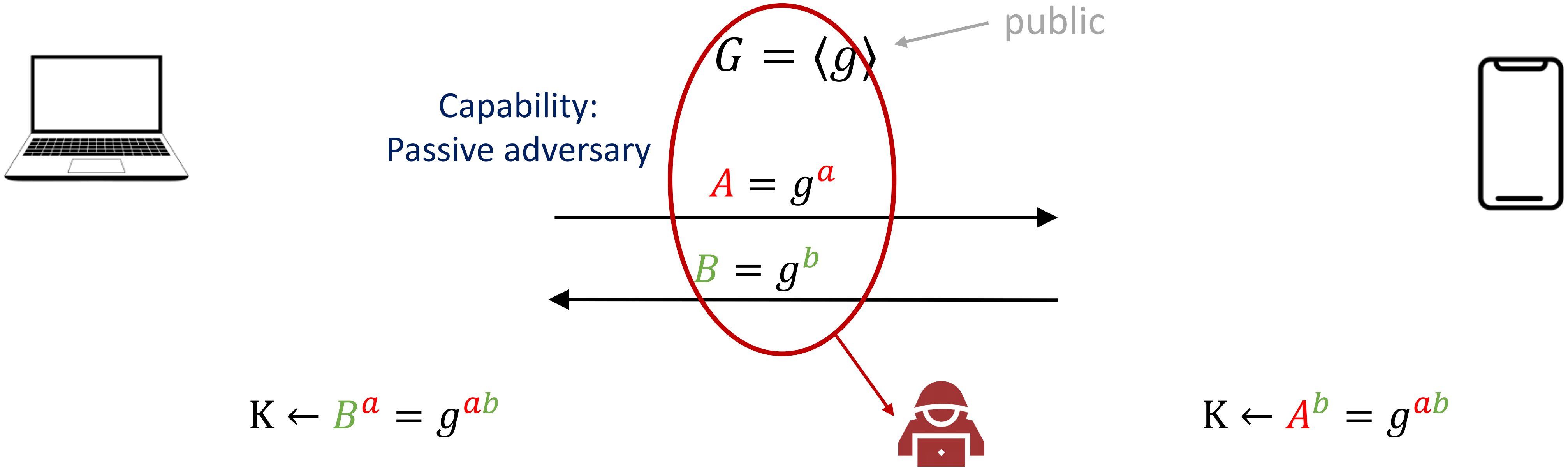
Network security

- authenticated key exchange

- public key infrastructure (PKI)

- and certification authorities

Diffie-Hellman Key Exchange



public

$G = \langle g \rangle$

Capability:
Passive adversary

$A = g^{a}$

$B = g^{b}$

$\mathrm{K} \leftarrow B^{a} = g^{ab}$

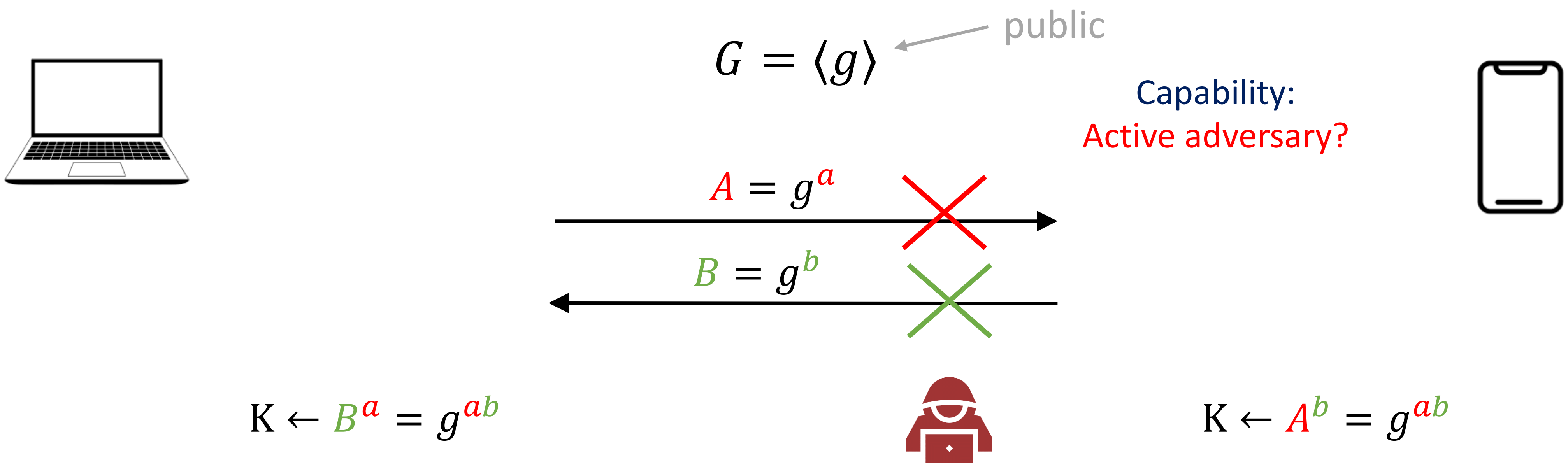
$\mathrm{K} \leftarrow A^{b} = g^{ab}$

Aim: $\mathrm{K} \approx_c$ random key???

Security (given G, g, A, B):

- Must be hard to distinguish $\mathrm{K} \leftarrow g^{ab}$ from random key

Diffie-Hellman Key Exchange



$G = \langle g \rangle$ ← public

Capability:

Active adversary?

$A = g^{a}$

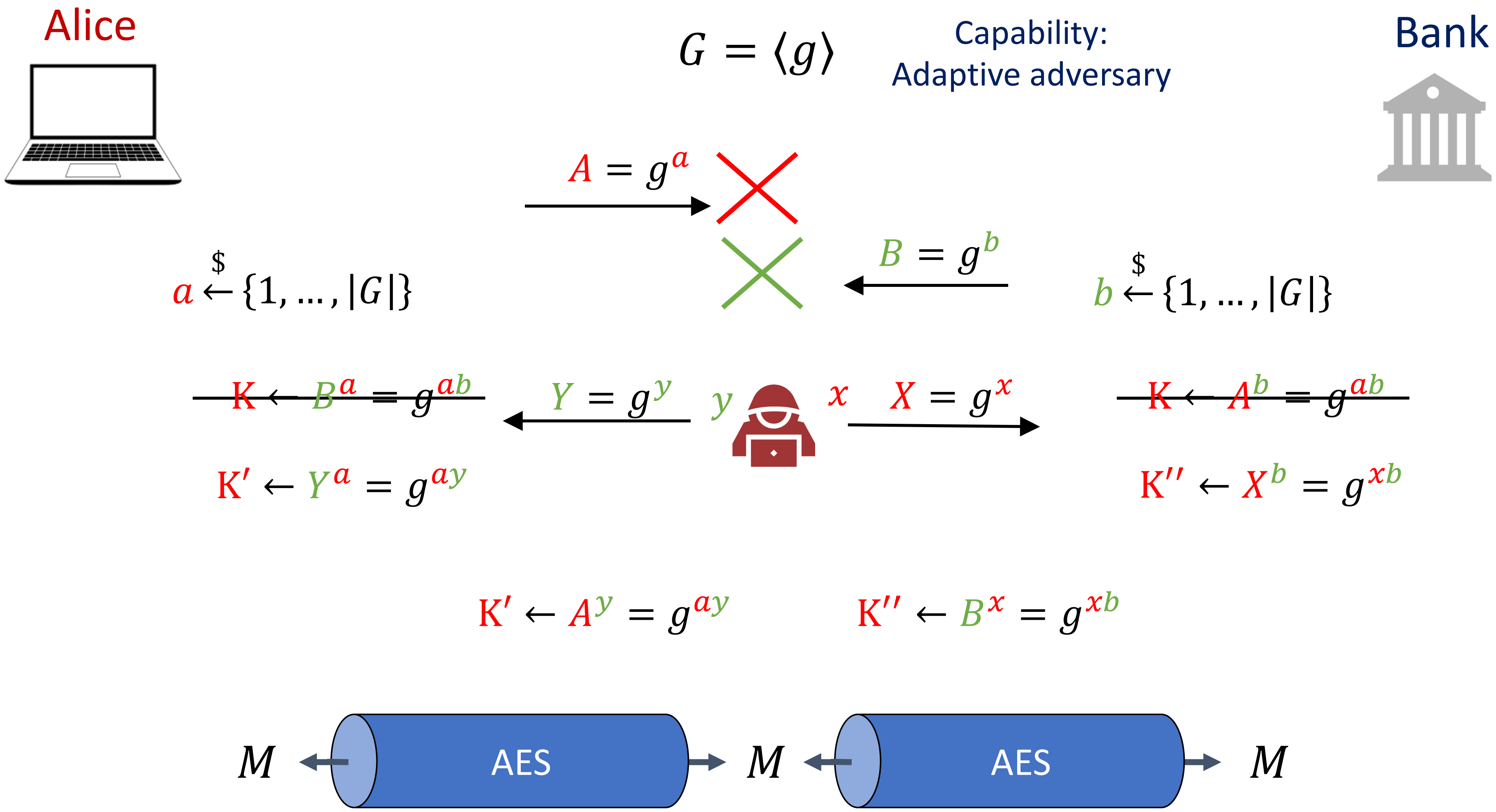
$B = g^{b}$

$\mathrm{K} \leftarrow B^{a} = g^{ab}$

$\mathrm{K} \leftarrow A^{b} = g^{ab}$

Aim: $\mathrm{K} \approx_c$ random key???

Diffie-Hellman: man-in-the-middle attack



Alice

$G = \langle g \rangle$

Capability:
Adaptive adversary

Bank

$A = g^a$

$B = g^b$

$a \overset{\$}{\leftarrow} \{1, \dots, |G|\}$

$b \overset{\$}{\leftarrow} \{1, \dots, |G|\}$

~~$\mathrm{K} \leftarrow B^a = g^{ab}$~~

$Y = g^y$ $\quad y \quad x \quad$ $X = g^x$

~~$\mathrm{K} \leftarrow A^b = g^{ab}$~~

$\mathrm{K}' \leftarrow Y^a = g^{ay}$

$\mathrm{K}'' \leftarrow X^b = g^{xb}$

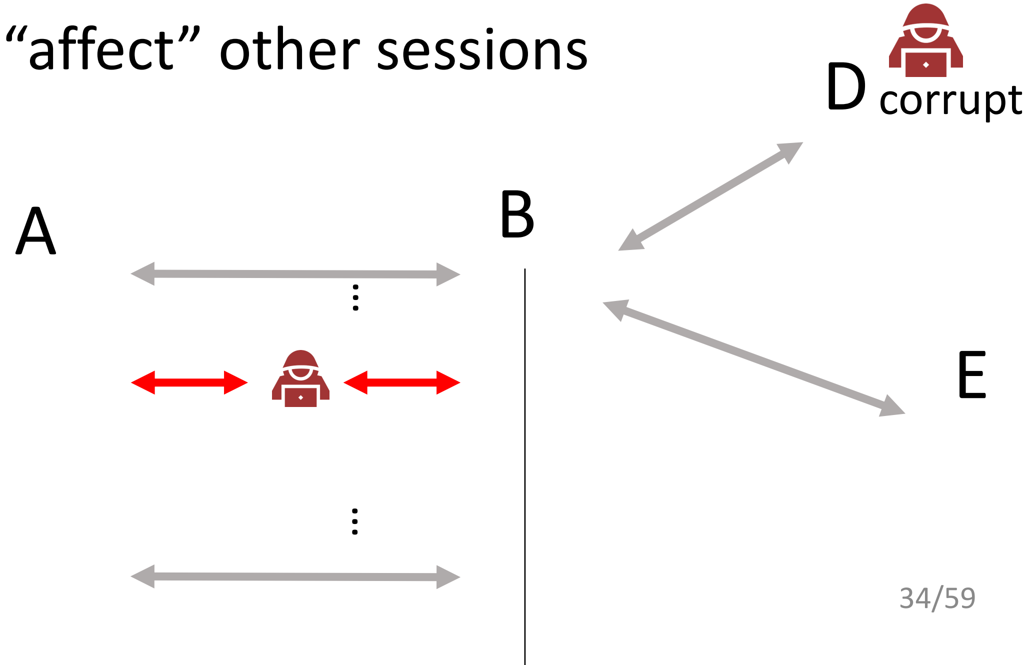
$\mathrm{K}' \leftarrow A^y = g^{ay}$ $\qquad$ $\mathrm{K}'' \leftarrow B^x = g^{xb}$

M AES M AES M

Active Adversary

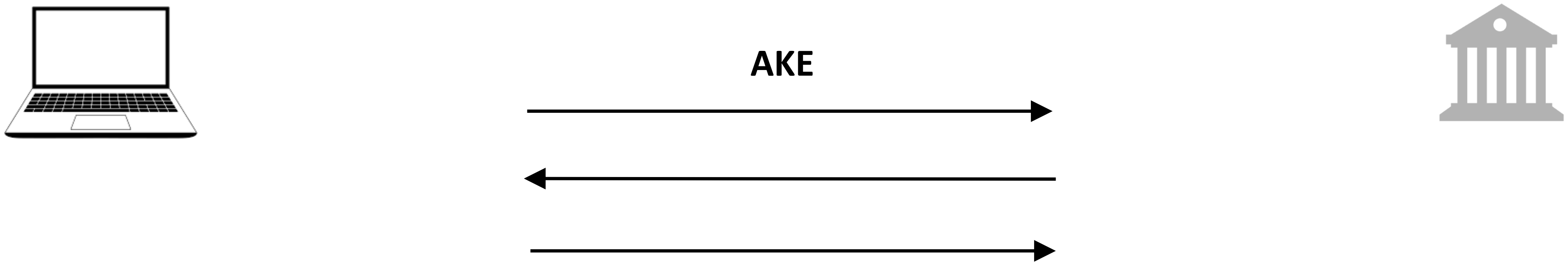
- Adversary has complete control of the network:
  - Can modify, inject and delete packets
  - Like the man-in-the-middle attack

- Moreover, some internet users are honest and others are corrupted
  - Corrupt users are controlled by the adversary
  - Key exchange with corrupt users should not “affect” other sessions



Authenticated Key Exchange (AKE)

- key exchange secure against **active** adversaries
- AKE protocol should allow two users to establish a shared key, and ensure that they are talking with whom they plan to talk with

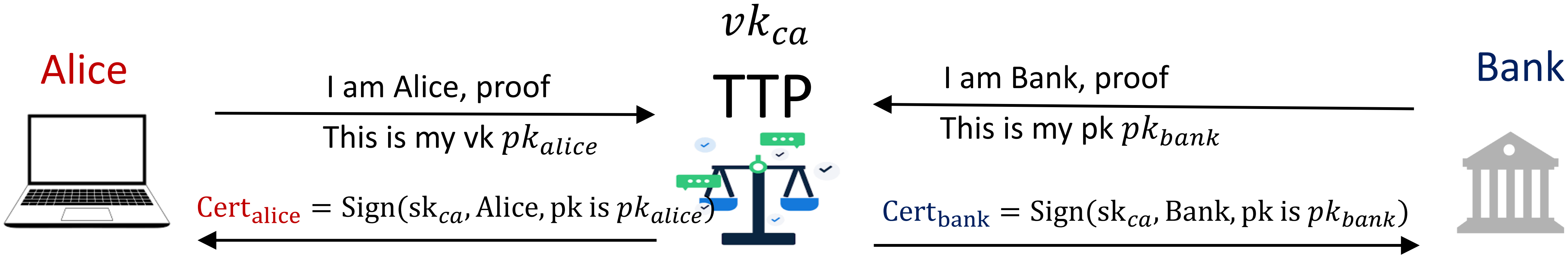


Capability:
Active adversary?

Trusted Third Party

All AKE protocols require a TTP to certify user identities.

Registration process:



Alice

I am Alice, proof

This is my vk pk_{alice}

$\mathrm{Cert}_{\mathrm{alice}} = \mathrm{Sign}(\mathrm{sk}_{ca}, \mathrm{Alice}, \mathrm{pk\ is\ } pk_{alice})$

vk_{ca}

TTP

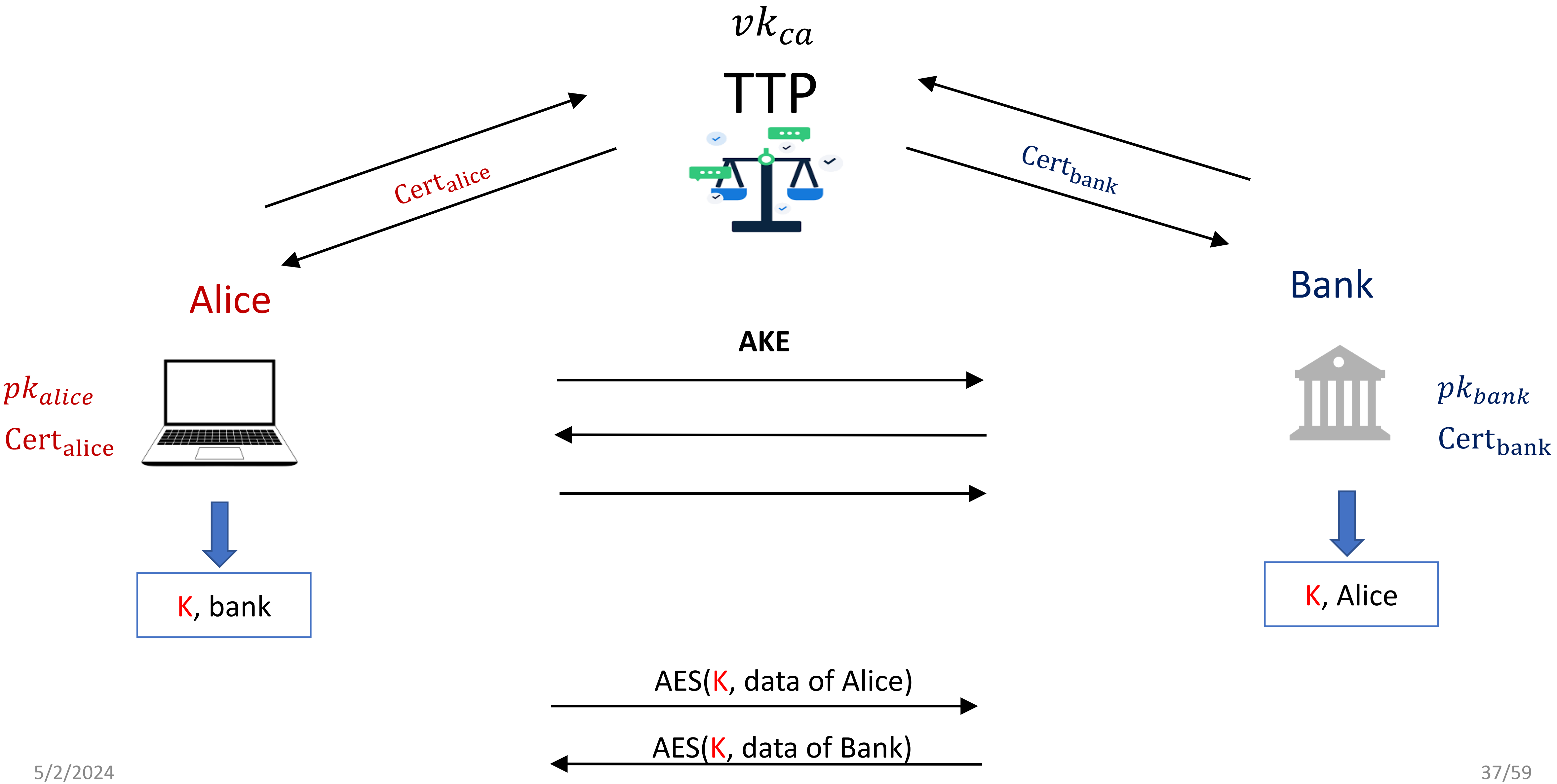
Bank

I am Bank, proof

This is my pk pk_{bank}

$\mathrm{Cert}_{\mathrm{bank}} = \mathrm{Sign}(\mathrm{sk}_{ca}, \mathrm{Bank}, \mathrm{pk\ is\ } pk_{bank})$

AKE-syntax



vk_{ca}

TTP

Cert_{alice}

Cert_{bank}

Alice

Bank

AKE

pk_{alice}

Cert_{alice}

pk_{bank}

Cert_{bank}

K, bank

K, Alice

AES(K, data of Alice)

AES(K, data of Bank)

Basic AKE security (very informal)

- Suppose Alice successfully completes an AKE to obtain **(K, Bank)**

- If Bank is not corrupt then:
  - **Authenticity** for Alice: (similarly for Bank)
  - If Alice’s key K is shared with anyone, it is only shared with Bank

  - **Secrecy** for Alice: (similarly for Bank)
  - To the adversary, Alice’s key K is indistinguishable from random (aim)

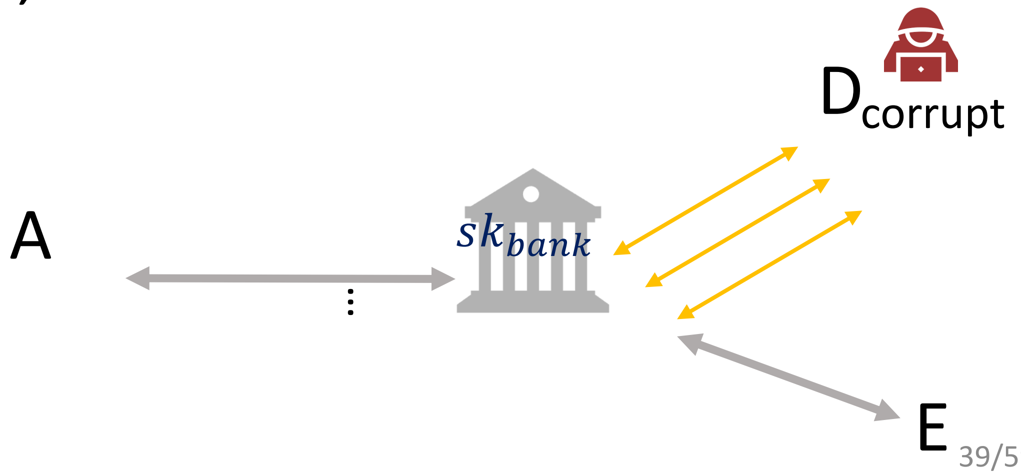
  - **Consistency**: if Bank completes AKE then it obtains **(K, Alice)**

Three levels (core) security of AKE:

- **Static security**: previous slide

- **Forward secrecy**: static security, and if the adversary learns sk_{bank} at time T then all sessions with Bank before T remain secure.

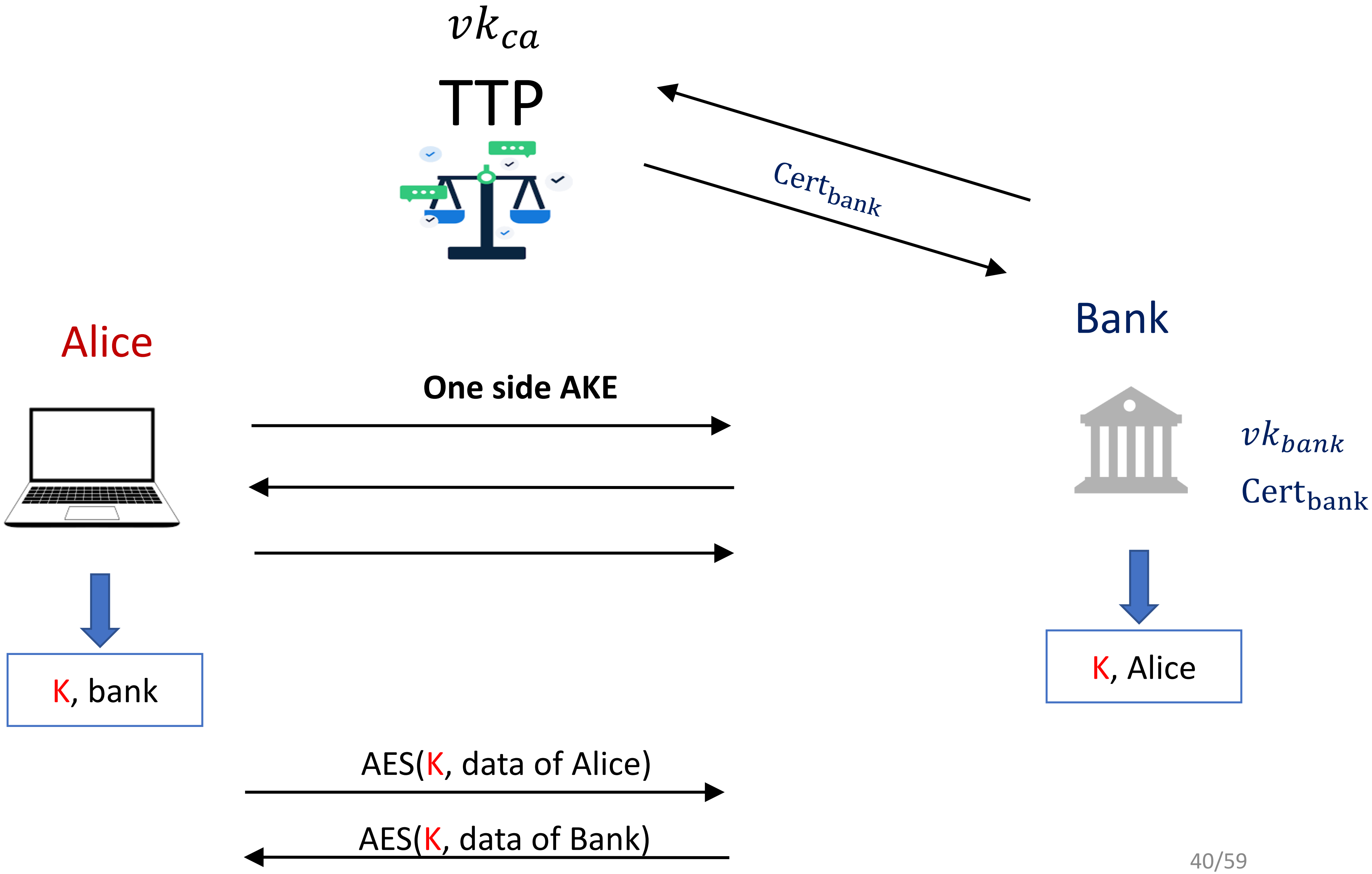
- **Hardware Security Module (HSM)**: Forward secrecy, and if adversary queries an HSM holding sk_{bank} n times, then at most n sessions are compromised.



One-sided AKE: syntax

- only one side has a certificate

- three security levels.



vk_{ca}

TTP

$\mathrm{Cert}_{\mathrm{bank}}$

Bank

vk_{bank}

$\mathrm{Cert}_{\mathrm{bank}}$

Alice

One side AKE

K, bank

K, Alice

AES(K, data of Alice)

AES(K, data of Bank)

Protocol #1 Building blocks

- Bank has $\mathrm{Cert}_{\mathrm{bank}}$ contains pk_{bank}

- Enc_{bank}: IND-CCA secure PKE using Bank's public key

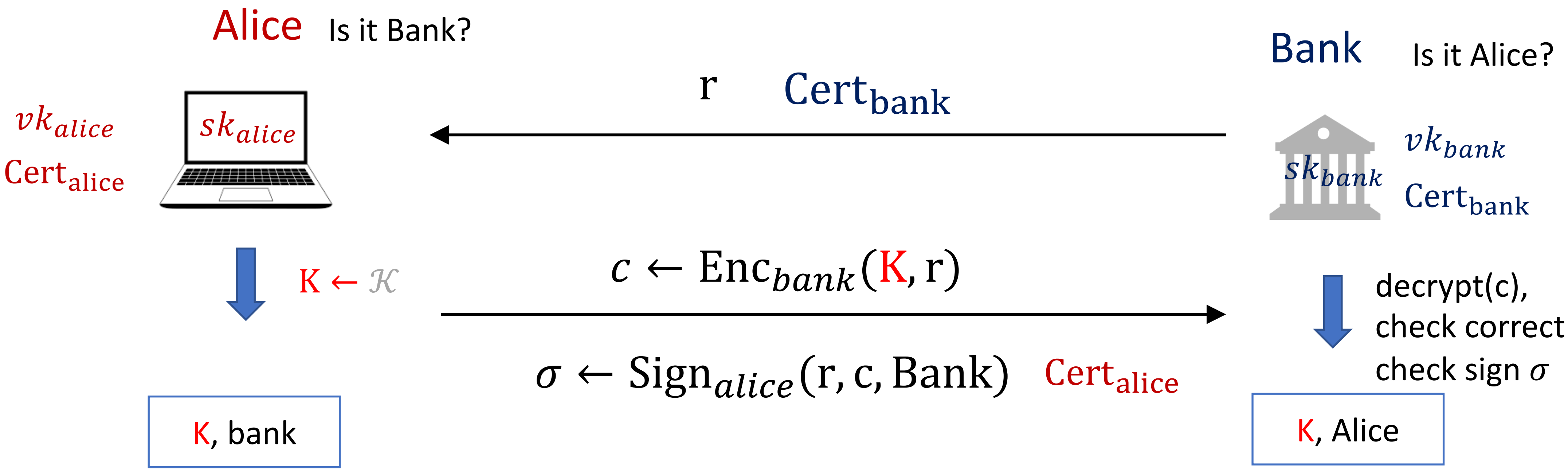
  Bank keeps sk_{bank} as the secret encryption key

- Sign_{alice} / Sign_{bank} : UF-CMA secure signature of Alice/Bank

- AES encryption scheme

Protocol #1

AKE1 of section 21.2 in A Graduate Course in Applied Cryptography



Alice Is it Bank?

vk_{alice}

$\mathrm{Cert}_{\mathrm{alice}}$

sk_{alice}

$\mathrm{K} \leftarrow \mathcal{K}$

K, bank

$\mathrm{r} \quad \mathrm{Cert}_{\mathrm{bank}}$

$c \leftarrow \mathrm{Enc}_{bank}(\mathrm{K}, \mathrm{r})$

$\sigma \leftarrow \mathrm{Sign}_{alice}(\mathrm{r}, \mathrm{c}, \mathrm{Bank}) \quad \mathrm{Cert}_{\mathrm{alice}}$

Bank Is it Alice?

sk_{bank}

vk_{bank}

$\mathrm{Cert}_{\mathrm{bank}}$

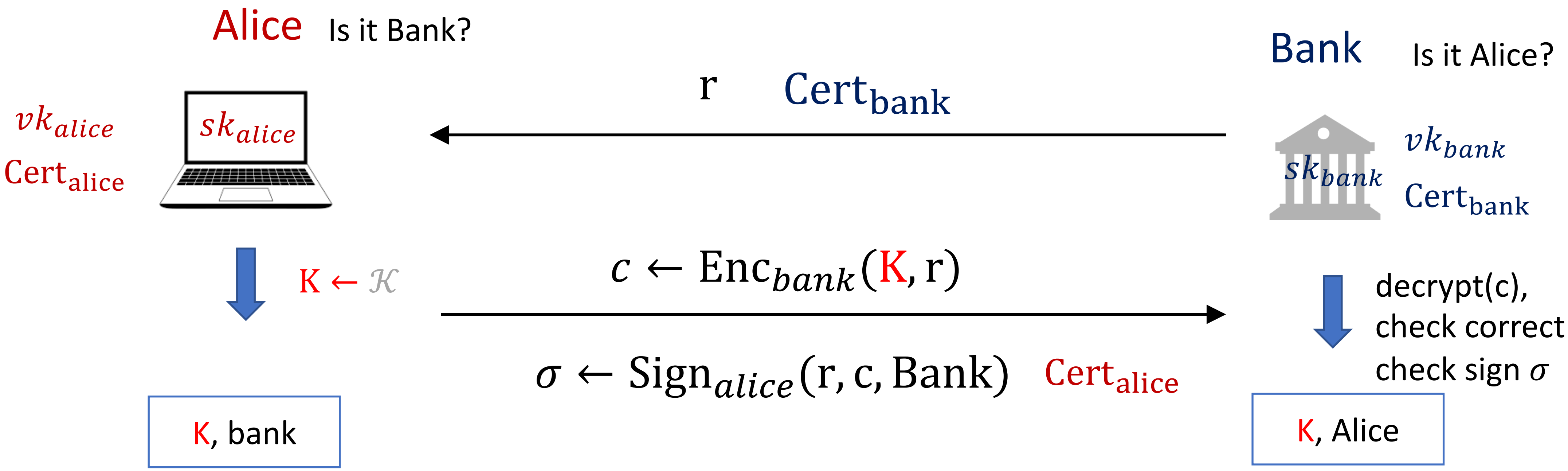
decrypt(c),
check correct
check sign σ

K, Alice

- Theorem: Protocol #1 is a statically secure AKE

- Informally: if Alice and Bank are not corrupt then we have
(1) secrecy for Alice\Bank and (2) authenticity for Alice\Bank

Protocol #1 problem: **no forward secrecy**



Alice Is it Bank?

Bank Is it Alice?

vk_{alice}

$\mathrm{Cert}_{\mathrm{alice}}$

sk_{alice}

r $\mathrm{Cert}_{\mathrm{bank}}$

sk_{bank}

vk_{bank}

$\mathrm{Cert}_{\mathrm{bank}}$

$\mathrm{K} \leftarrow \mathcal{K}$

$c \leftarrow \mathrm{Enc}_{bank}(\mathrm{K}, \mathrm{r})$

$\sigma \leftarrow \mathrm{Sign}_{alice}(\mathrm{r}, \mathrm{c}, \mathrm{Bank})$ $\mathrm{Cert}_{\mathrm{alice}}$

decrypt(c),
check correct
check sign σ

K, bank

K, Alice

Suppose a year later adversary obtains sk_{bank}

⇒ can decrypt all recorded traffic

Protocol #1 is used in TLS 1.2 not TLS 1.3

AKE1 of section 21.2 in A Graduate Course in Applied Cryptography

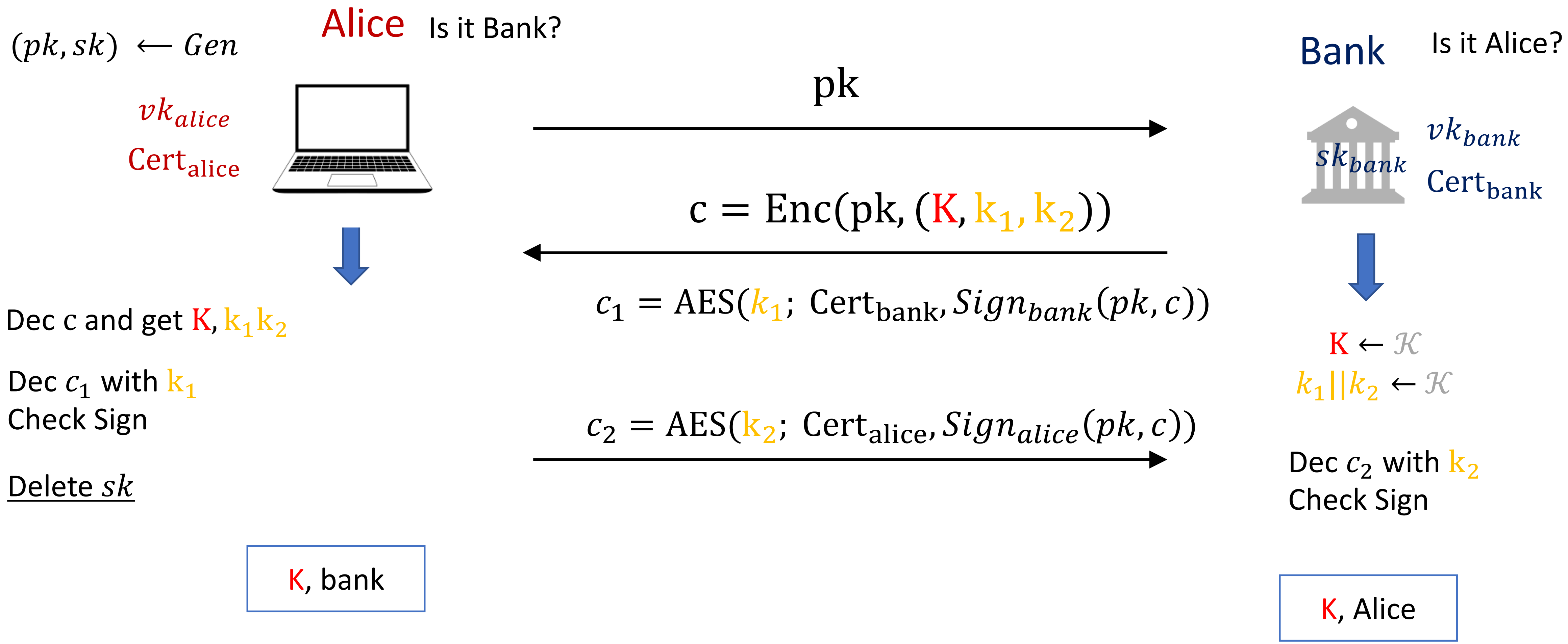
Protocol #2: HSM Security

Forward secrecy, and

n queries to HSM should compromise at most n sessions

AKE4 of section 21.2 in A Graduate Course in Applied Cryptography

Protocol #2



$(pk, sk) \leftarrow Gen$

Alice Is it Bank?

vk_{alice}

$\text{Cert}_{\text{alice}}$

Bank Is it Alice?

sk_{bank}

vk_{bank}

$\text{Cert}_{\text{bank}}$

pk

$\text{c} = \text{Enc}(\text{pk}, (\text{K}, \text{k}_1, \text{k}_2))$

$c_1 = \text{AES}(k_1;\ \text{Cert}_{\text{bank}}, Sign_{bank}(pk, c))$

$c_2 = \text{AES}(\text{k}_2;\ \text{Cert}_{\text{alice}}, Sign_{alice}(pk, c))$

Dec c and get K, $\text{k}_1 \text{k}_2$

Dec c_1 with k_1
Check Sign

<u>Delete sk</u>

K, bank

$\text{K} \leftarrow \mathcal{K}$

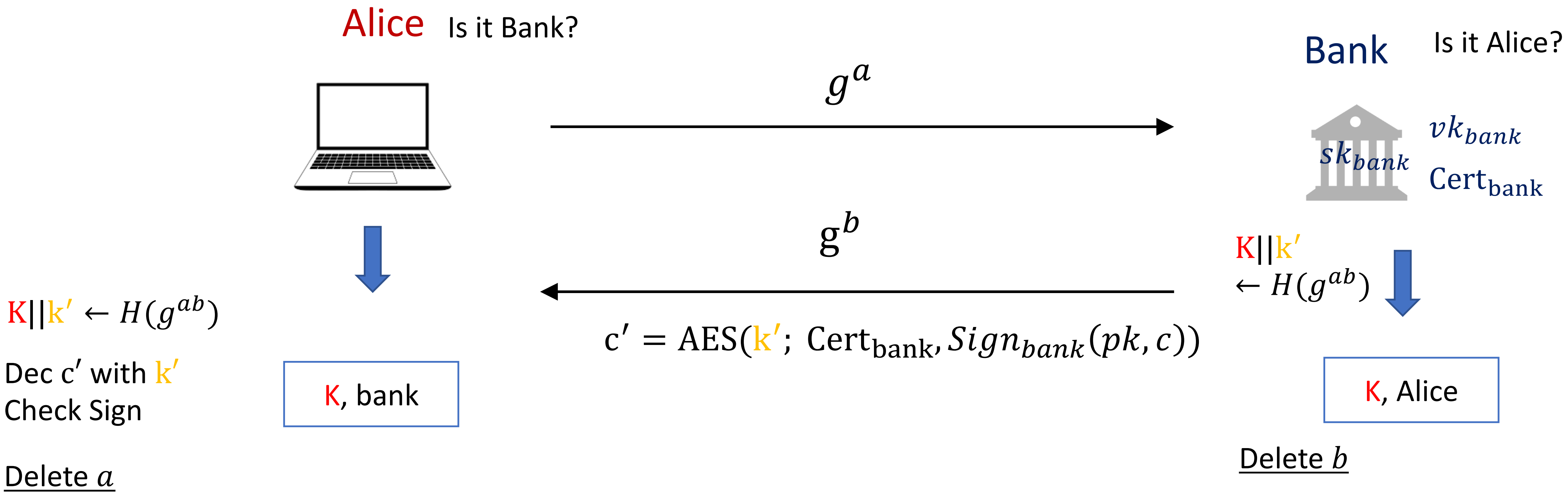
$k_1||k_2 \leftarrow \mathcal{K}$

Dec c_2 with k_2
Check Sign

K, Alice

Main point: need to sign ephemeral pk from client
⇒ past access to HSM will not compromise current session

Protocol #4 one side-use Diffie-Hellman instead of PKE



Alice Is it Bank?

Bank Is it Alice?

vk_{bank}

sk_{bank}

$\mathrm{Cert}_{\mathrm{bank}}$

g^a

g^b

$\mathrm{c}' = \mathrm{AES}(\mathrm{k}';\ \mathrm{Cert}_{\mathrm{bank}}, Sign_{bank}(pk, c))$

$\mathrm{K}||\mathrm{k}' \leftarrow H(g^{ab})$

Dec c' with k'
Check Sign

Delete a

K, bank

$\mathrm{K}||\mathrm{k}'$
$\leftarrow H(g^{ab})$

K, Alice

Delete b

[variant of TLS 1.3]

A short summary

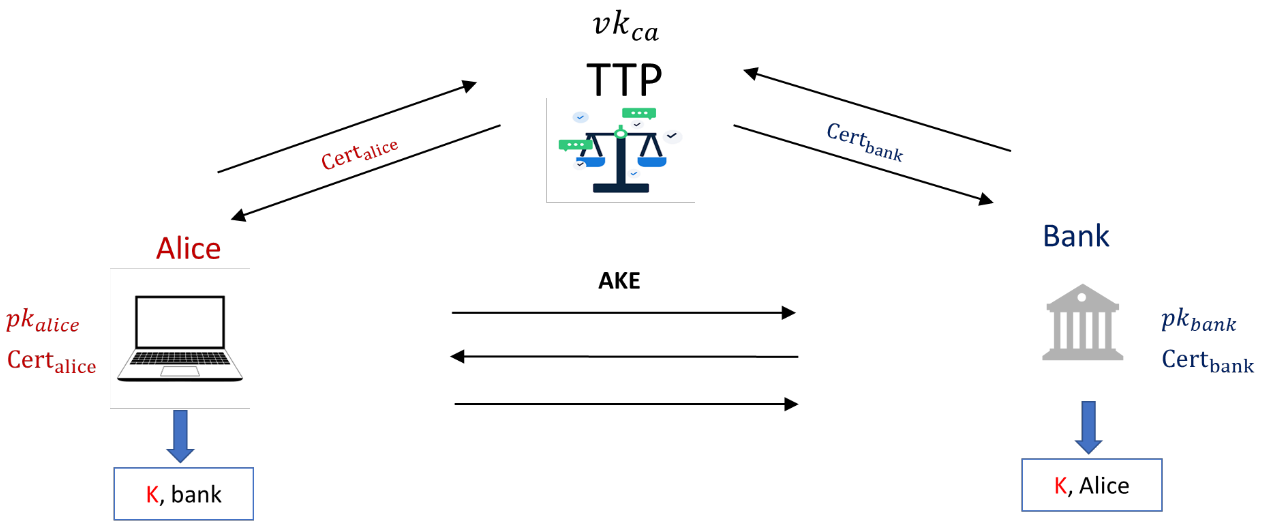
- AKE requires TTP to certify user identities
- Security: static security, Forward secrecy, HSM secrecy
- We can build secure AKE via PKE, signature, and/or, AES

Problem: public key infrastructure (PKI)

- A single TTP

- Single point of failure
  - What if TTP is corrupted?

- How should we deploy the trust of certification?
  - How does Bank communicate with TTP to get Cert_bank?



TTP: Certification Authorities

- Digital Certification

vk_{ca}

TTP

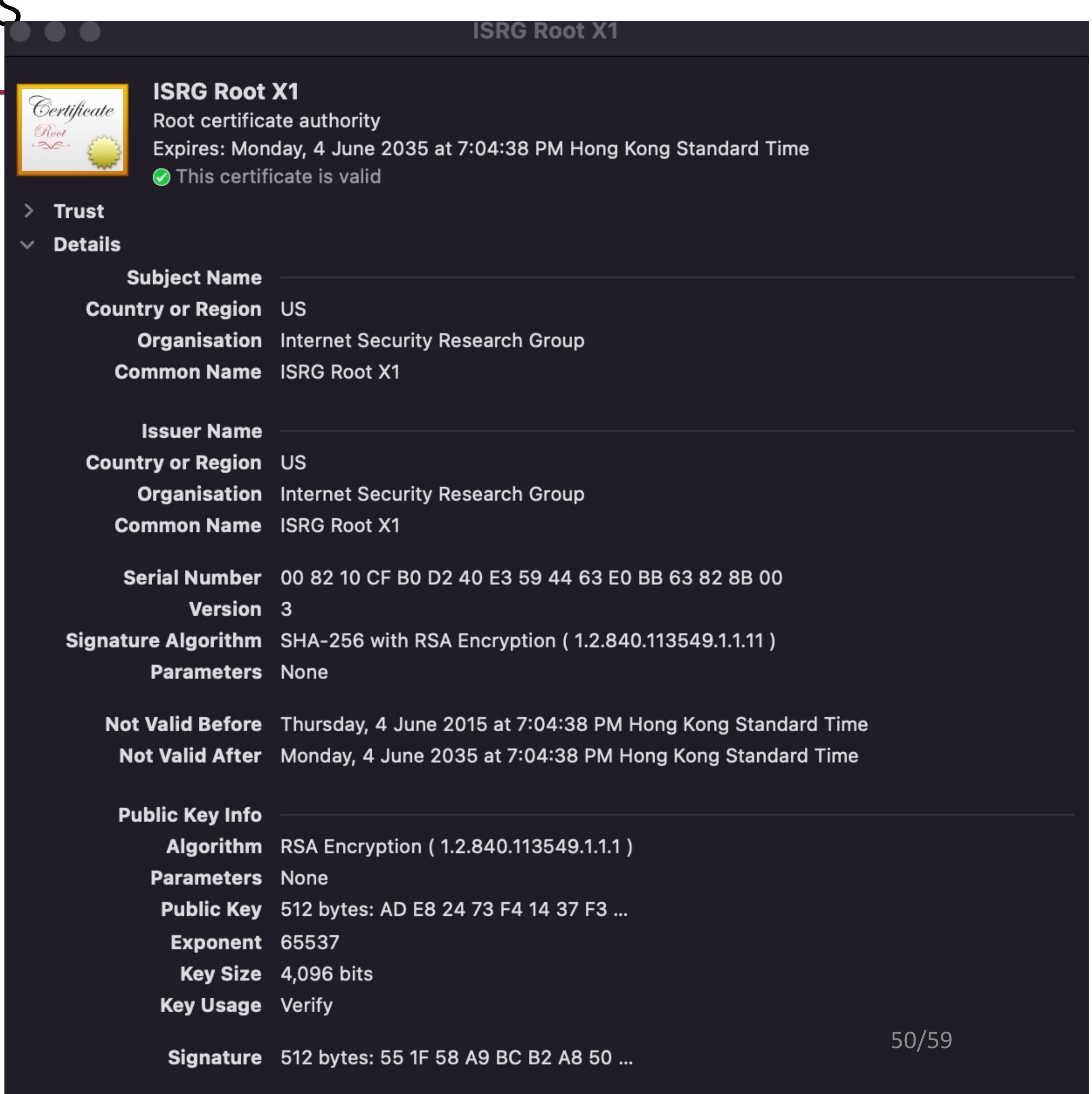
sk_{ca}

$$\mathrm{Cert}_{bank} = \mathrm{Sign}(\mathrm{sk}_{ca}, \mathrm{Bank's\ public\ (sign)\ key\ is\ } vk_{bank};\ \ \mathrm{URL\ ishttps://www.hangseng.com/}\ )$$

Any one with vk_{ca} can verify the Cert_{bank}

TTP: Certification Authorities

- Subject Name
  - Who’s CA
- Issuer Name
  - Who gives this CA
  - Sign name
  - Valid
- PK information
  - pk
  - What is the pk is used
  - Key size



ISRG Root X1
Root certificate authority
Expires: Monday, 4 June 2035 at 7:04:38 PM Hong Kong Standard Time
✓ This certificate is valid

› **Trust**
⌄ **Details**

Subject Name
Country or Region US
Organisation Internet Security Research Group
Common Name ISRG Root X1

Issuer Name
Country or Region US
Organisation Internet Security Research Group
Common Name ISRG Root X1

Serial Number 00 82 10 CF B0 D2 40 E3 59 44 63 E0 BB 63 82 8B 00
Version 3
Signature Algorithm SHA-256 with RSA Encryption (1.2.840.113549.1.1.11)
Parameters None

Not Valid Before Thursday, 4 June 2015 at 7:04:38 PM Hong Kong Standard Time
Not Valid After Monday, 4 June 2035 at 7:04:38 PM Hong Kong Standard Time

Public Key Info
Algorithm RSA Encryption (1.2.840.113549.1.1.1)
Parameters None
Public Key 512 bytes: AD E8 24 73 F4 14 37 F3 ...
Exponent 65537
Key Size 4,096 bits
Key Usage Verify

Signature 512 bytes: 55 1F 58 A9 BC B2 A8 50 ...

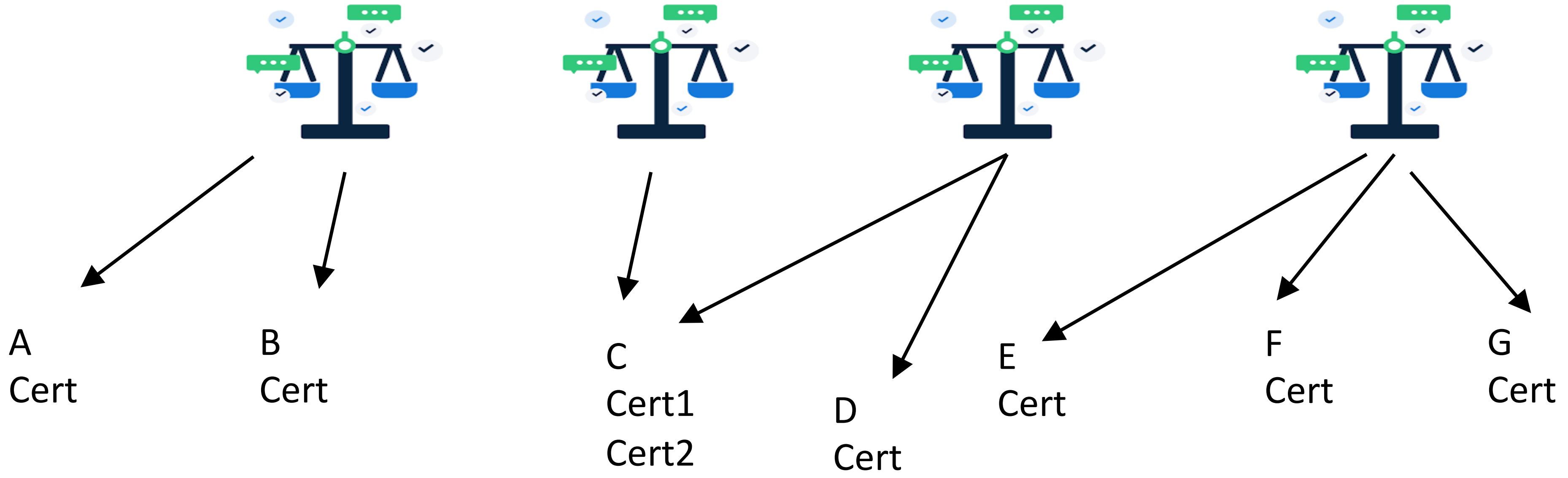
Certification Authorities(CA)

vk_{ca}

TTP

- How should I get the vk_{ca} of TTP?
- a root CA’s public key is provided together with the browser/System

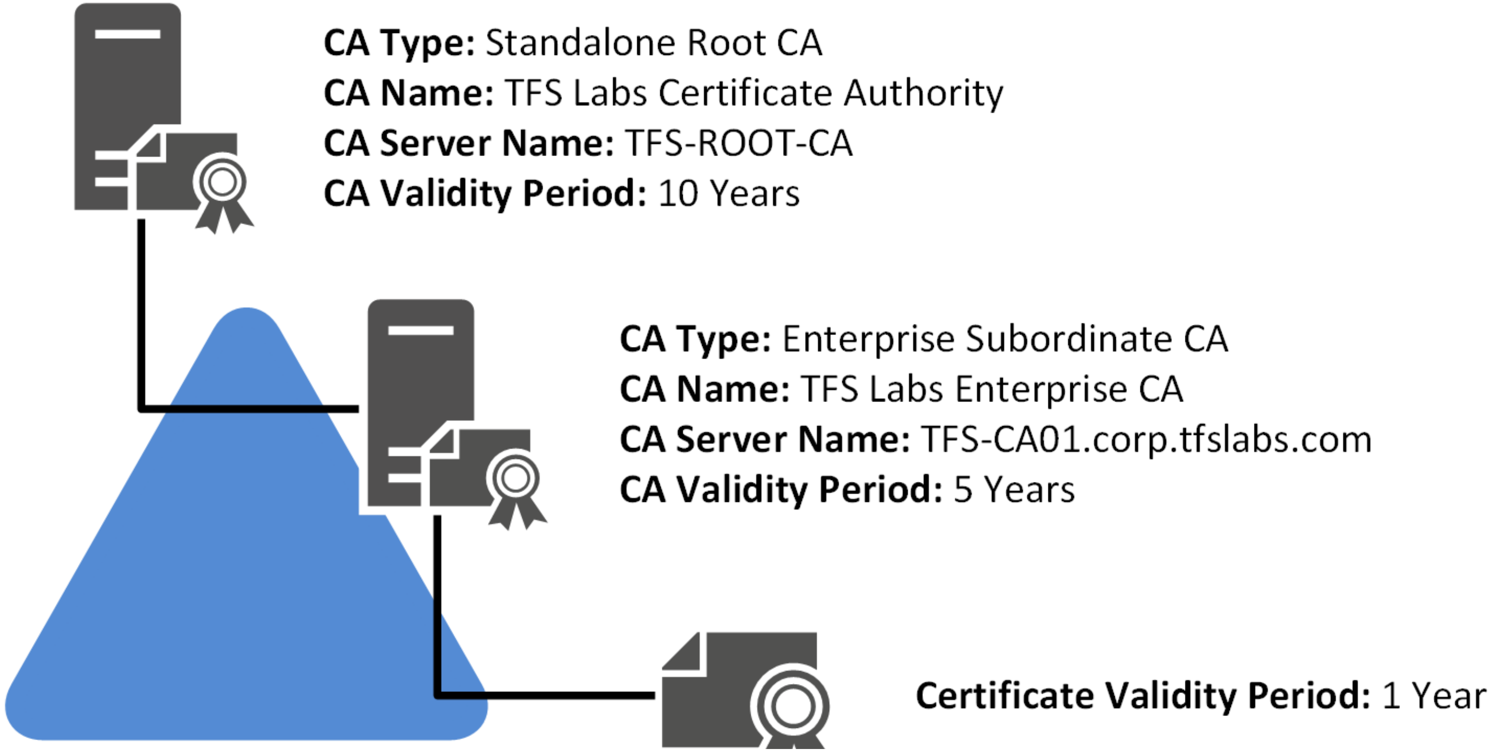
Multiple CAs



- Reduce the risk of single point of failure

Authentication Chain

We could build the trust of certificate chains from a single Root CA

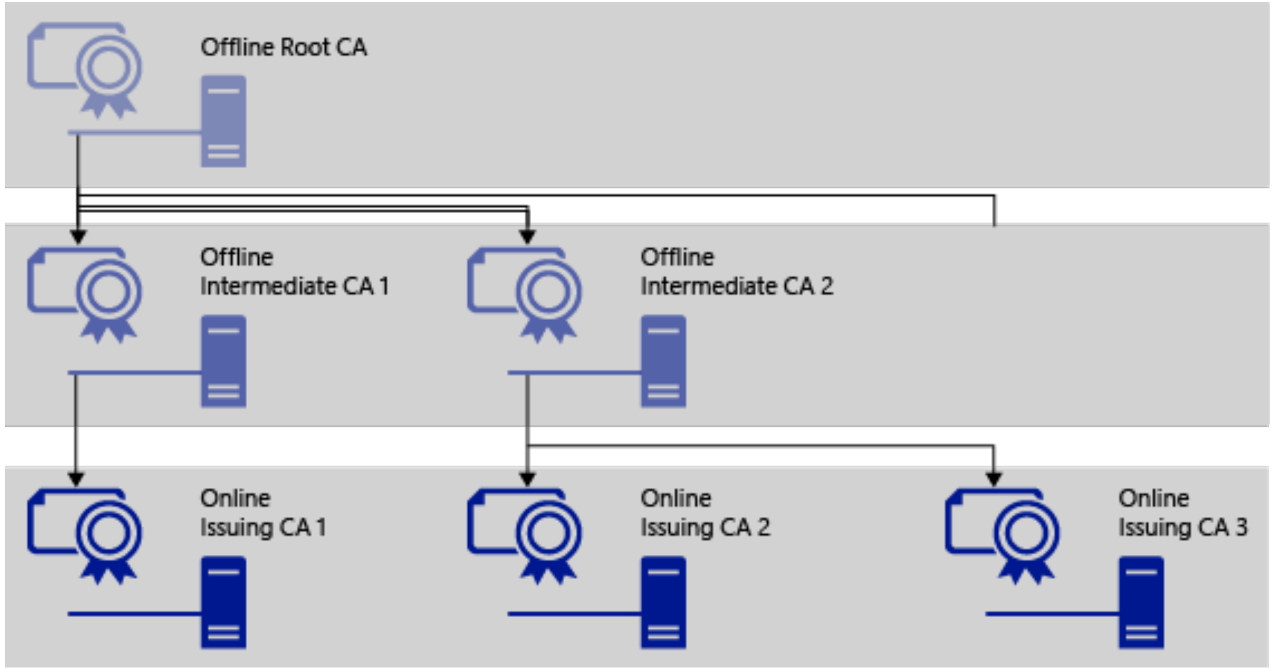


CA Type: Standalone Root CA
CA Name: TFS Labs Certificate Authority
CA Server Name: TFS-ROOT-CA
CA Validity Period: 10 Years

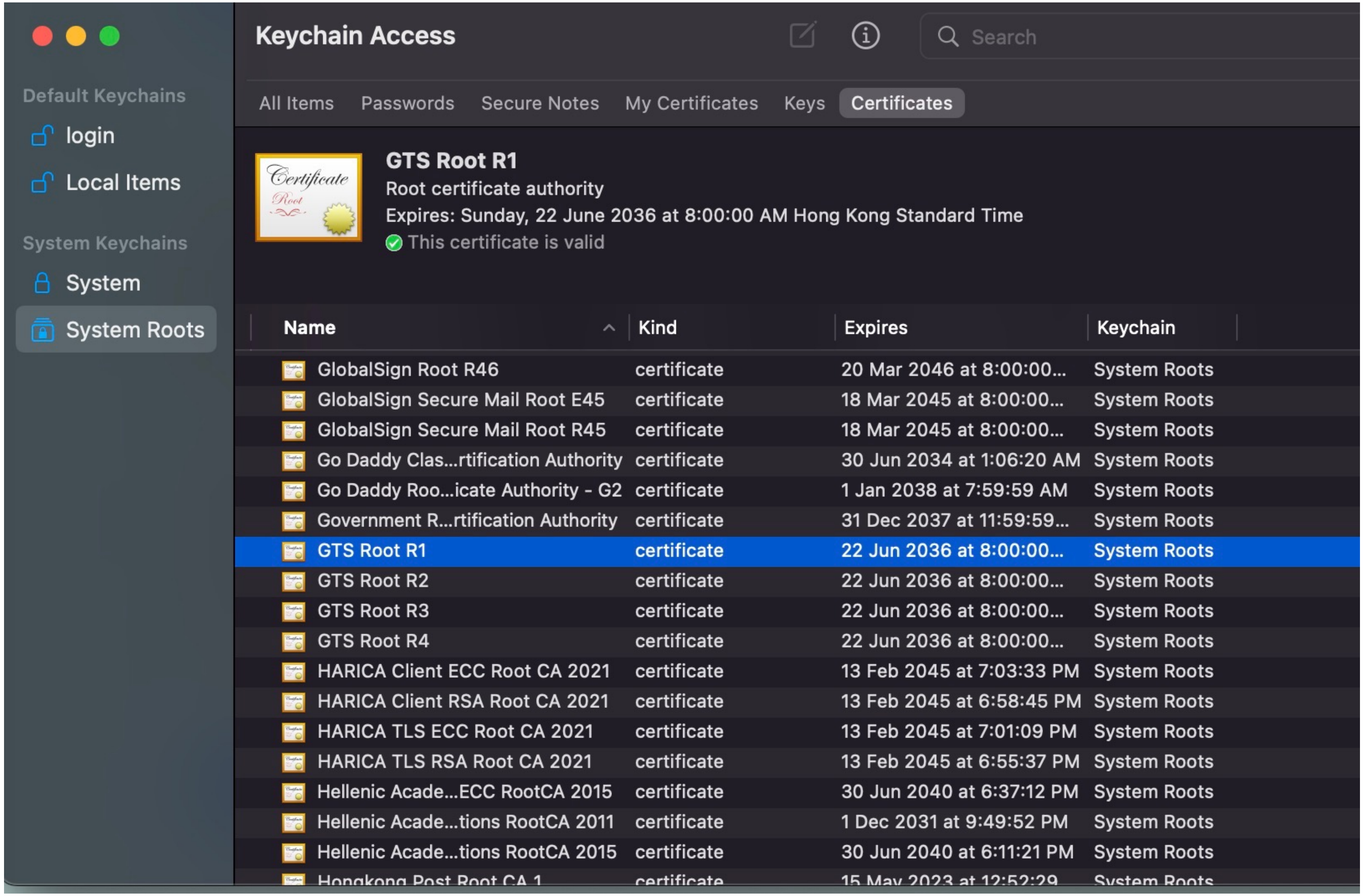
CA Type: Enterprise Subordinate CA
CA Name: TFS Labs Enterprise CA
CA Server Name: TFS-CA01.corp.tfslabs.com
CA Validity Period: 5 Years

Certificate Validity Period: 1 Year

Authentication Chain



Root CA in Mac OS



GTS Root R1

Root certificate authority

Expires: Sunday, 22 June 2036 at 8:00:00 AM Hong Kong Standard Time

✓ This certificate is valid

Name	Kind	Expires	Keychain
GlobalSign Root R46	certificate	20 Mar 2046 at 8:00:00...	System Roots
GlobalSign Secure Mail Root E45	certificate	18 Mar 2045 at 8:00:00...	System Roots
GlobalSign Secure Mail Root R45	certificate	18 Mar 2045 at 8:00:00...	System Roots
Go Daddy Clas...rtification Authority	certificate	30 Jun 2034 at 1:06:20 AM	System Roots
Go Daddy Roo...icate Authority - G2	certificate	1 Jan 2038 at 7:59:59 AM	System Roots
Government R...rtification Authority	certificate	31 Dec 2037 at 11:59:59...	System Roots
GTS Root R1	certificate	22 Jun 2036 at 8:00:00...	System Roots
GTS Root R2	certificate	22 Jun 2036 at 8:00:00...	System Roots
GTS Root R3	certificate	22 Jun 2036 at 8:00:00...	System Roots
GTS Root R4	certificate	22 Jun 2036 at 8:00:00...	System Roots
HARICA Client ECC Root CA 2021	certificate	13 Feb 2045 at 7:03:33 PM	System Roots
HARICA Client RSA Root CA 2021	certificate	13 Feb 2045 at 6:58:45 PM	System Roots
HARICA TLS ECC Root CA 2021	certificate	13 Feb 2045 at 7:01:09 PM	System Roots
HARICA TLS RSA Root CA 2021	certificate	13 Feb 2045 at 6:55:37 PM	System Roots
Hellenic Acade...ECC RootCA 2015	certificate	30 Jun 2040 at 6:37:12 PM	System Roots
Hellenic Acade...tions RootCA 2011	certificate	1 Dec 2031 at 9:49:52 PM	System Roots
Hellenic Acade...tions RootCA 2015	certificate	30 Jun 2040 at 6:11:21 PM	System Roots
Hongkong Post Root CA 1	certificate	15 May 2023 at 12:52:29	System Roots

Root CA in Windows

- Root CA in windows
  - Select Run from the Start menu, and then enter certlm.msc. The Certificate Manager tool for the local device appears.

Root CA in web browser

- chrome://settings/security

- Firefox

Summary

- Recall RSA and Digital Signature

- Authenticated Key Exchange
- Public Key Infrastructure(PKI)
- and Certification Authorities

- For your lecture notes, please refer to
- [KL] Section 12.7
  Dan Boneh and Victor Shoup, A Graduate Course in Applied Cryptography, Section 22
  [Du] Section 24

Thank you

Happy Chinese New Year