Lecture 4: Network Security Principles

-COMP 6712 Advanced Security and Privacy

Haiyang Xue

haiyang.xue@polyu.edu.hk

2023/2/7

Network Security Principles

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

Public key encryption

Diffie-Hellman

$$G = \langle g \rangle$$

$a \xleftarrow{\$} \{1, \dots, |G|\}$

$A \leftarrow g^a$

$A \longrightarrow$

$b \xleftarrow{\$} \{1, \dots, |G|\}$

$B \leftarrow g^b$

$\xleftarrow{\quad B \quad}$

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

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

Ratchet Diffie-Hellman in WhatsApp and Signal

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 $AES256(K_{01}, \text{Hey Adam How ...})$ | $K_{01} = g^{a_0 b_1}$

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

$A_1 \leftarrow g^{a_1}$

$K_{11} = g^{a_1 b_1}$ | $AES256(K_{11}, \text{Hey Bud})$ A_1 | $K_{11} = g^{a_1 b_1}$

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

$B_2 \leftarrow g^{b_2}$

$K_{12} = g^{a_1 b_2}$ | B_2 $AES256(K_{12}, emo: thumb)$ | $K_{12} = g^{a_1 b_2}$

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

$A_2 \leftarrow g^{a_2}$

$K_{22} = g^{a_2 b_2}$ | $AES256(K_{22}, \text{Yes !})$ A_2 | $K_{22} = g^{a_2 b_2}$

ElGamal

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

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

$$G = \langle g \rangle$$

B

A, C

KeyGen

1. $sk = \; b \xleftarrow{\$} \{1, \dots, |G|\}$
2. $pk = \; B \leftarrow g^b$
3. **return** (sk, pk)

Enc(pk, M)

1. $a \xleftarrow{\$} \{1, \dots, |G|\}$
2. $A \leftarrow g^a$
3. $K \leftarrow B^a = g^{ab}$
4. $C \leftarrow K \cdot M$
5. **return** (A, C)

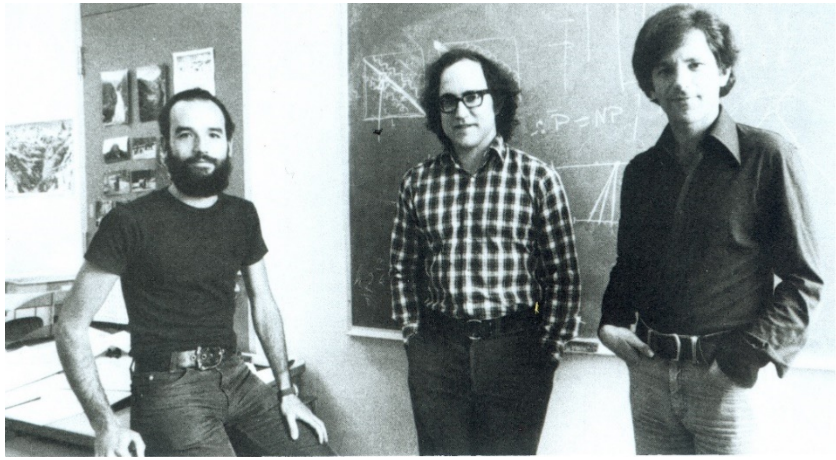
Dec(sk, C)

1. $Z \leftarrow A^b = g^{ab}$
2. $M \leftarrow C/K$
3. **return** M

RSA in 1977

- The RSA encryption scheme

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



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

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

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

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

RSA Enc $\quad M \xrightarrow{\textbf{Enc}} C \leftarrow M^e \bmod n \xrightarrow{\textbf{Dec}} M \leftarrow C^d \bmod n$

Textbook RSA

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

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

KeyGen

1. $p, q \xleftarrow{\$}$ two random prime numbers
2. $n \leftarrow p \cdot q$
3. $\phi(n) = (p-1)(q-1)$
4. **choose** e such that $\gcd(e, \phi(n)) = 1$
5. $d \leftarrow e^{-1} \bmod \phi(n)$
6. $sk \leftarrow d \quad pk \leftarrow (n, e)$
7. **return** (sk, pk)

Enc$(pk = (n, e), M \in \mathbf{Z}^{*}_{n})$

1. $C \leftarrow M^{e} \bmod n$
2. **return** C

$$\xleftarrow{\quad pk \quad}$$

$$\xrightarrow{\quad C \quad}$$

Dec$(sk = d, C \in \mathbf{Z}^{*}_{n})$

1. $M \leftarrow C^{d} \bmod n$
2. **return** M

Common choices of e: 3, 17, 65 537

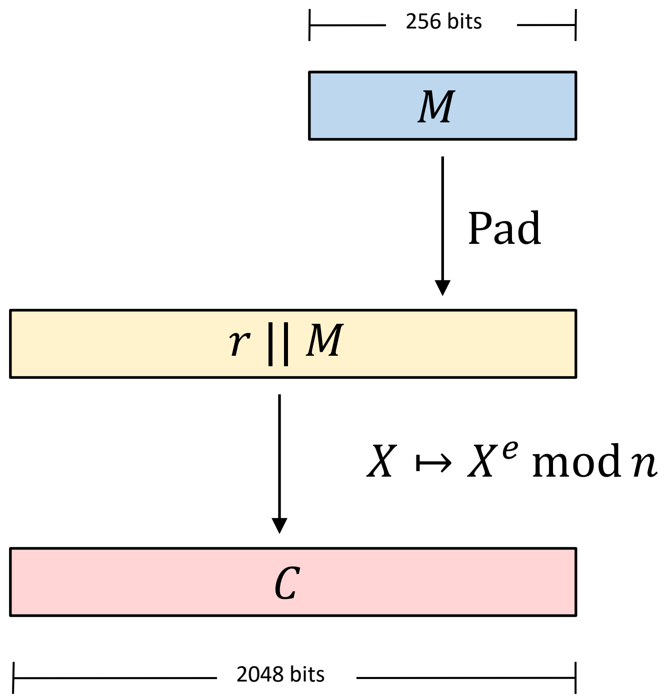
11_2 $\quad 10001_2$ $\quad 1\,0000\,0000\,0000\,0001_2$

RSA in practice

- Textbook RSA is deterministic $\Longrightarrow$ 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



Demo RSA encryption

- Demonstration using SageMath

- https://sagecell.sagemath.org/

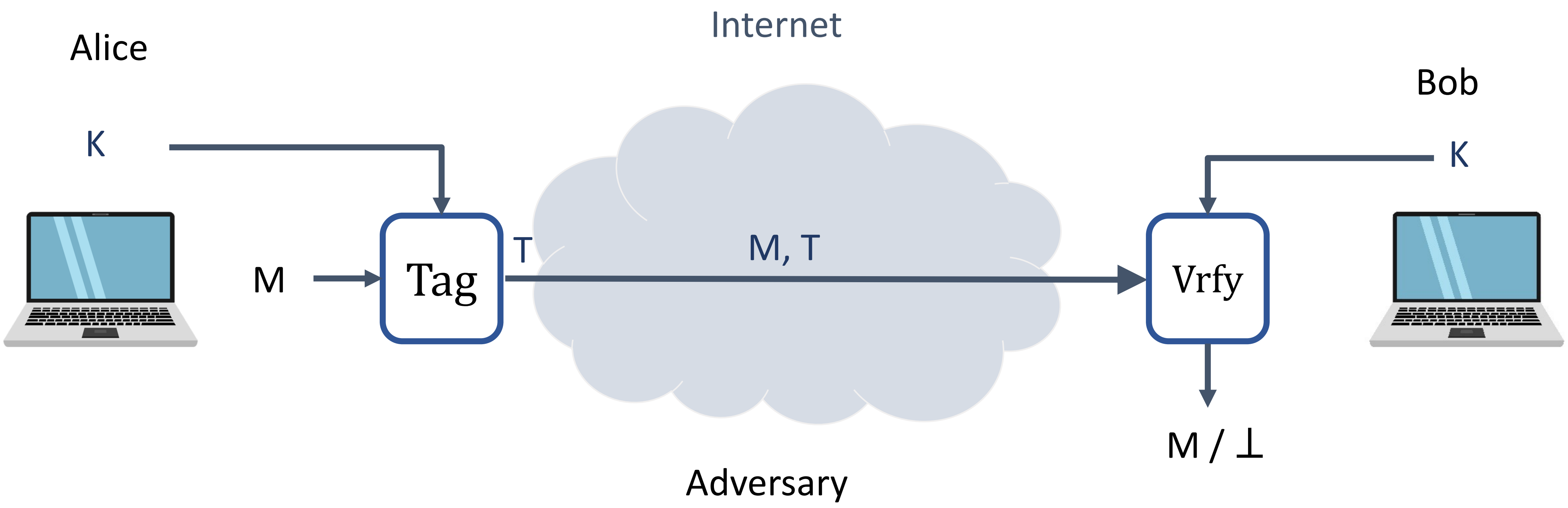
A short summary

- 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

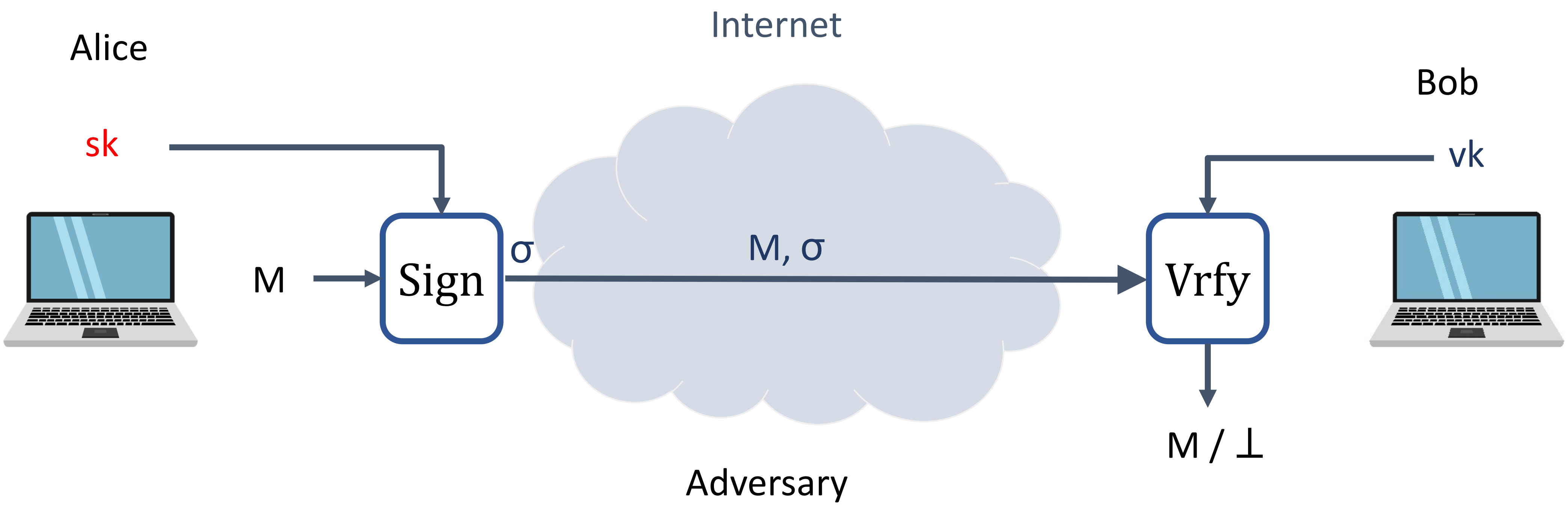


Tag : tagging algorithm (public)

K : tagging / verification key (secret)

Vrfy: verification algorithm (public)

Achieving integrity: digital signatures



Sign : tagging algorithm (public)

Vrfy : verification algorithm (public)

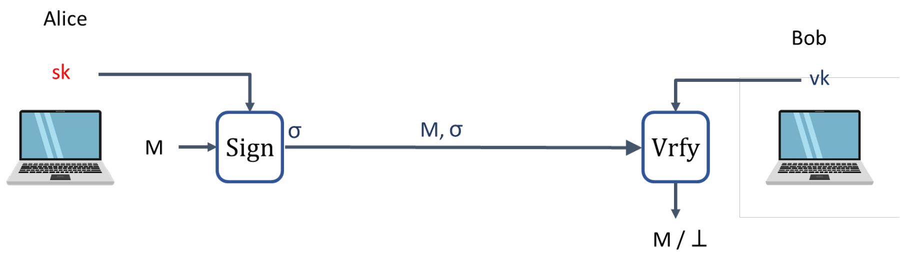
sk : signing key (secret)

vk : verification key (public)

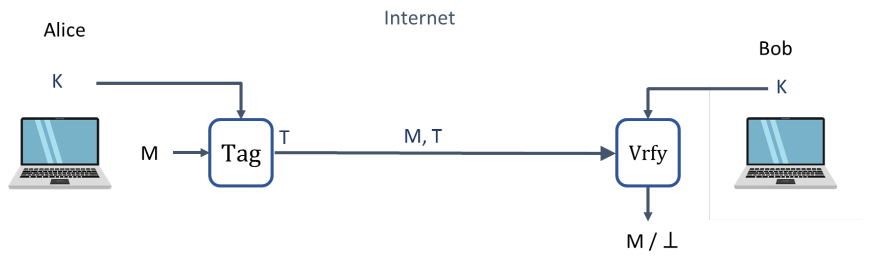
Digital signatures vs. MACs

- Digital signatures can be verified by *anyone*
- MACs can only be verified by party sharing the same key
- **Non-repudiation:** Alice cannot deny having created σ
  - But she can deny having created T (since Bob could have done it)

Digital signature



MAC



Digital signatures – syntax

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

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

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

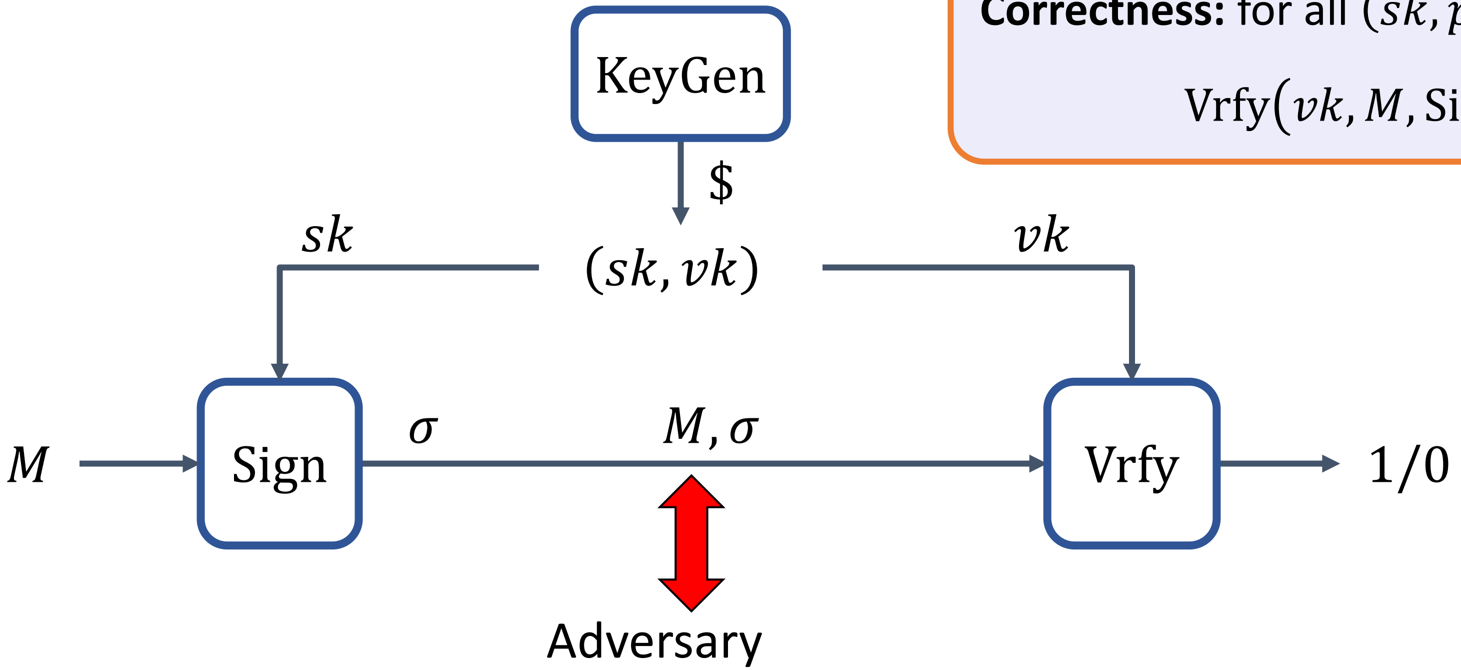
$\mathrm{Sign}(sk, M) = \mathrm{Sign}_{sk}(M) = \sigma$

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

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

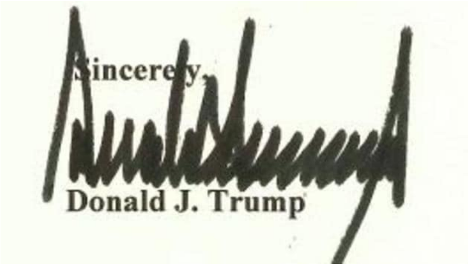
Correctness: for all $(sk, pk) \leftarrow \mathrm{KeyGen}$:

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



18

Signature: unforgeability



unforgeable

$\sigma_1, \sigma_2, \ldots$

unforgeable

σ^*

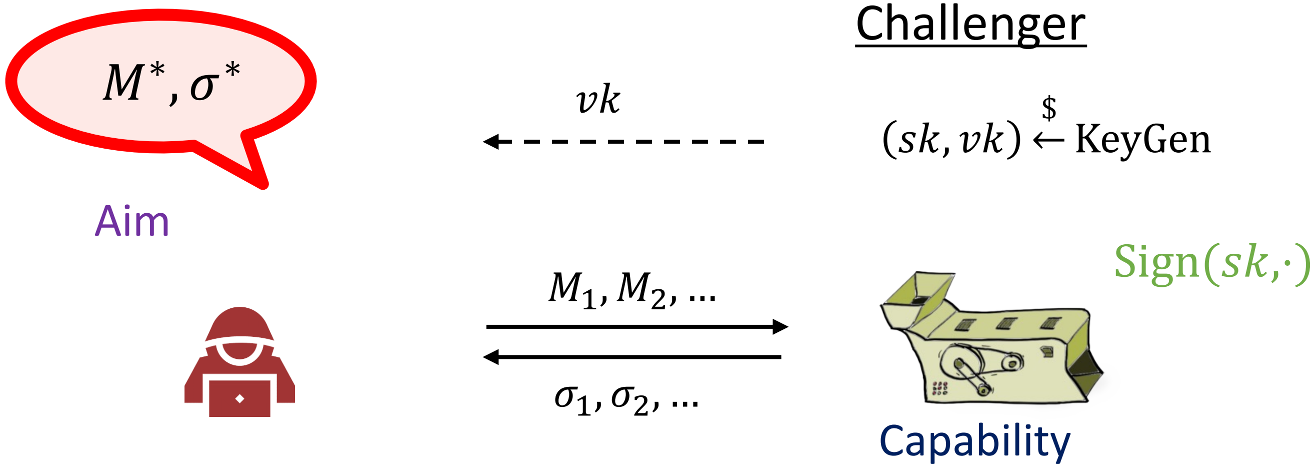
Digital signatures – security: UF-CMA

$\mathbf{Exp}^{\text{uf-cma}}_{\Sigma}(A)$

$(sk, vk) \xleftarrow{\$} \Sigma.\mathrm{KeyGen}$
$S \leftarrow [\,]$
$(M^*, \sigma^*) \leftarrow A^{\mathrm{SIGN}_{sk}(\cdot)}(vk)$
if $\Sigma.\mathrm{Vrfy}(vk, M^*, \sigma^*) = 1$ and $M \notin S$ **then**
  return 1
 else
  return 0

$\mathrm{SIGN}_{sk}(M)$

$\sigma \leftarrow \Sigma.\mathrm{Sign}(sk, M)$
$S.\mathrm{add}(M)$
return σ



If σ^* is a valid signature for M^* (not asked before) then the adversary has **forged** a signature

Definition: The **UF-CMA-advantage** of an adversary A is

$$\mathbf{Adv}^{\text{uf-cma}}_{\Sigma}(A) = \Pr\left[\mathbf{Exp}^{\text{uf-cma}}_{\Sigma}(A) \Rightarrow 1\right]$$

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

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

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

RSA Enc $\quad M \xrightarrow{\textbf{Enc}} C \leftarrow M^e \bmod n \xrightarrow{\textbf{Dec}} M \leftarrow C^d \bmod n$

RSA Sign $\quad M = \sigma^e \xleftarrow{\textbf{Vrfy}} \sigma \leftarrow M^d \bmod n \xleftarrow{\textbf{Sign}} M$

Textbook RSA signatures

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

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

KeyGen

$p, q \xleftarrow{\$}$ two random prime numbers

$n \leftarrow p \cdot q$

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

choose e such that $\gcd(e, \phi(n)) = 1$

$d \leftarrow e^{-1} \bmod \phi(n)$

$sk \leftarrow (n, d) \qquad vk \leftarrow (n, e)$

return (sk, vk)

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

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

vk

M, σ

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

$\sigma \leftarrow M^d \bmod n$

return σ

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

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

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

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

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

KeyGen

1. $p, q \overset{\$}{\leftarrow}$ two random prime numbers
2. $n \leftarrow p \cdot q$
3. $\phi(n) = (p-1)(q-1)$
4. **choose** e such that $\gcd(e, \phi(n)) = 1$
5. $d \leftarrow e^{-1} \bmod \phi(n)$
6. $sk \leftarrow (n, d) \qquad vk \leftarrow (n, e)$
7. **return** (sk, vk)

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

```
if σ^e = H(M) mod n then
    return 1
else
    return 0
```

$\xleftarrow{\quad vk \quad}$

$\xleftarrow{\quad M, \sigma \quad}$

$$H : \{0,1\}^{*} \rightarrow \boldsymbol{Z}^{*}_{n}$$

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

```
σ ← H(M)^d mod n
return σ
```

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:

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

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

Discrete-log-based signatures: (EC)DSA

- Schnorr
  - Elegant design
  - Has formal security proof (based on DLOG problem and H assumed perfect)
  - Patented (expired in February 2008)

- (EC)DSA
  - Non-patented alternative
  - Derived from ElGamal-based signature scheme
  - More complicated design than Schnorr
  - No security proof
  - Standardized by NIST, USA
  - Very widely used

Digital signature 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 28 Feb)

- 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）

- Assignment 1 and instructions are available on the blackboard

Network security

- authenticated key exchange
- public key infrastructure (PKI)
- and certification authorities

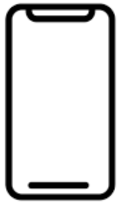
Diffie-Hellman Key Exchange

$G = \langle g \rangle$ public

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???

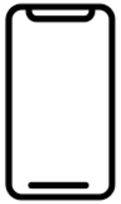
Security (given G, g, A, B):

- 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?



$A = g^a$

$B = g^b$

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

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

Aim: $K \approx_c$ random key???

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

Alice

$G = \langle g \rangle$

Capability:
Adaptive adversary

Bank

$A = g^a$

$B = g^b$

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

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

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

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

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

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

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

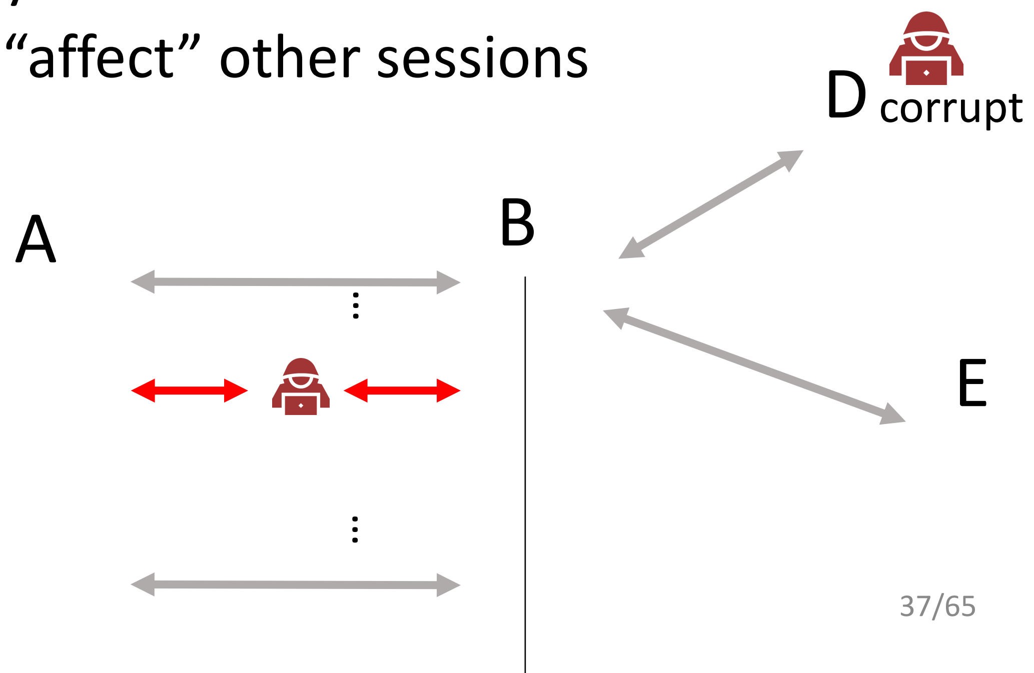
$K' \leftarrow A^y = g^{ay}$

$K'' \leftarrow B^x = g^{xb}$

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 corrupt
  - Corrupt users are controlled by the adversary
  - Key exchange with corrupt users should not “affect” other sessions



Authenticated Key Exchange (AKE)

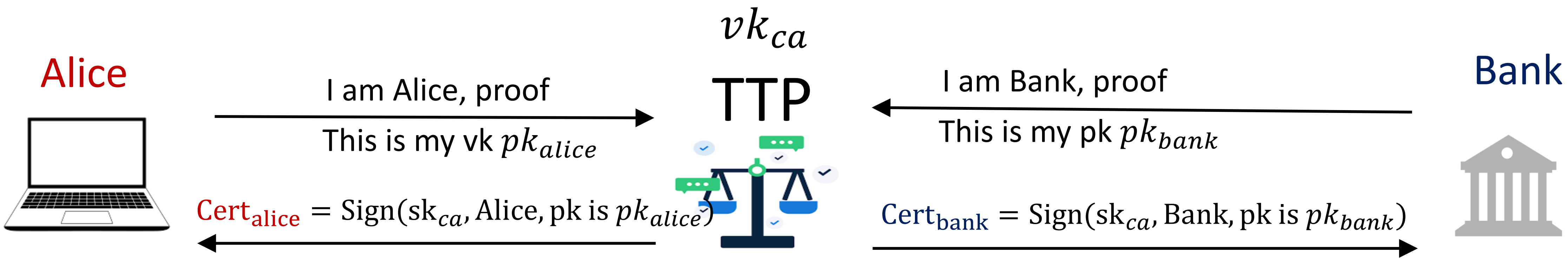
- key exchange secure against **active** adversaries
- AKE protocol should allow two users to establish a shared key, and ensure that they are talking with whom they plan to talk with

Capability:
Active adversary?

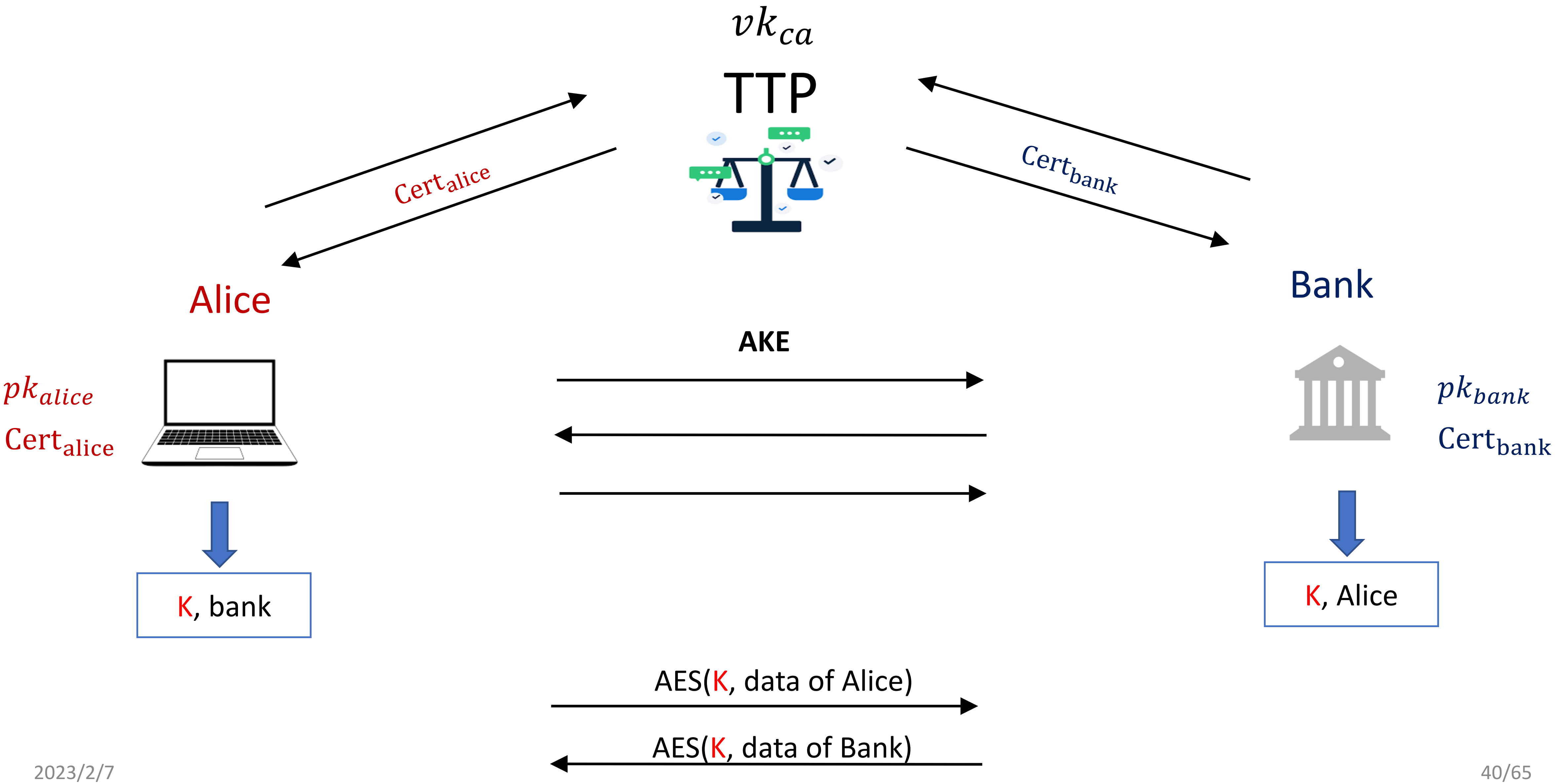
Trusted Third Party

All AKE protocols require a TTP to certify user identities.

Registration process:



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)

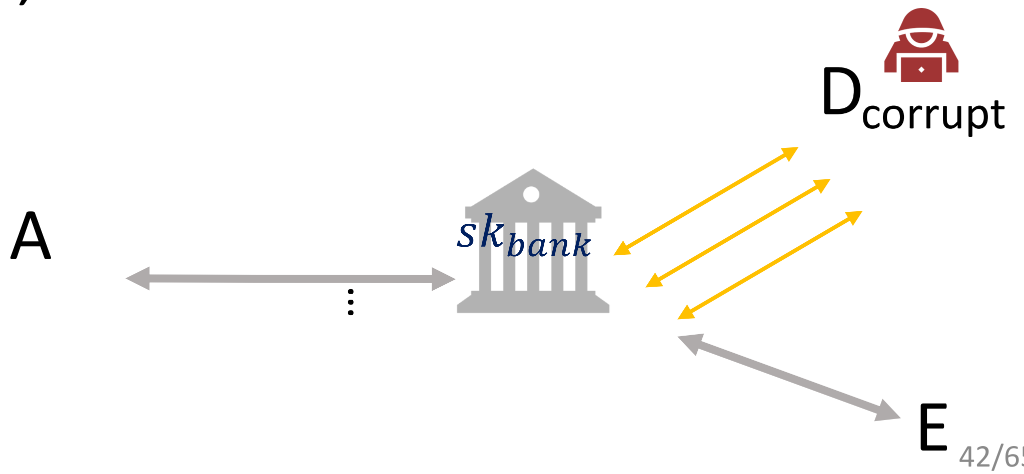
  - **Consistency**: if Bank completes AKE then it obtains **(K, Alice)**

Three levels (core) security of AKE:

- **Static security**: previous slide

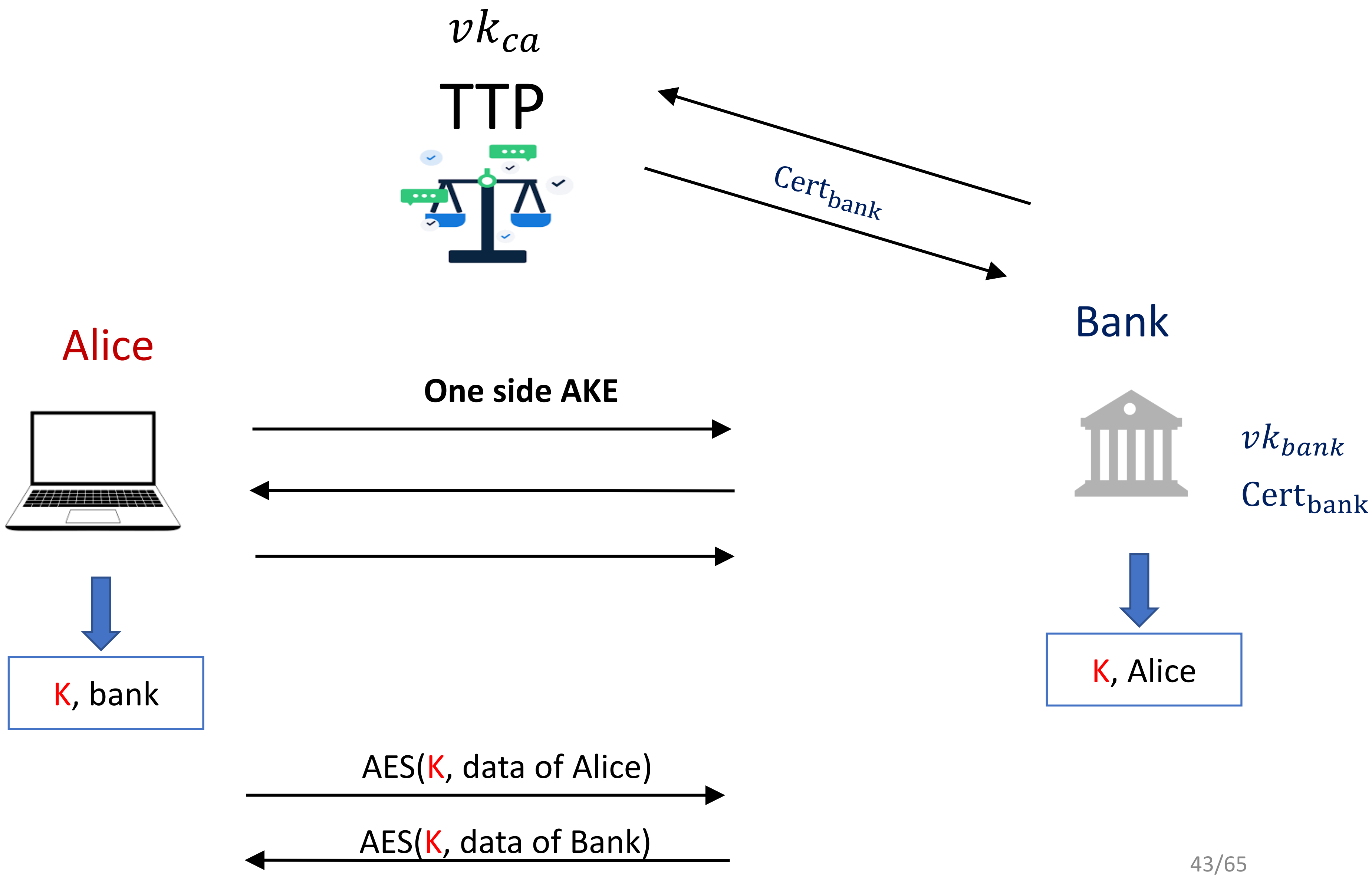
- **Forward secrecy**: static security, and if the adversary learns sk_{bank} at time T then all sessions with Bank before T remain secret.

- **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.



Protocol #1 Building blocks

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

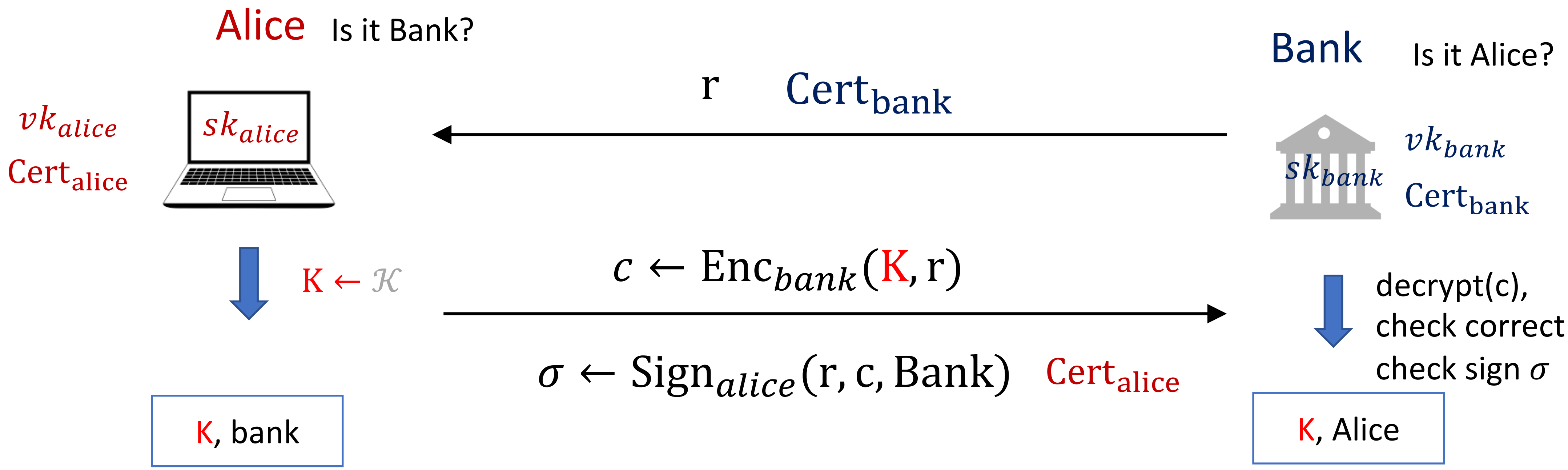
- Enc_{bank}: IND-CCA secure PKE using Bank's public key
  Bank keeps sk_{bank} as the secret encryption key

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

- AES encryption scheme

Protocol #1

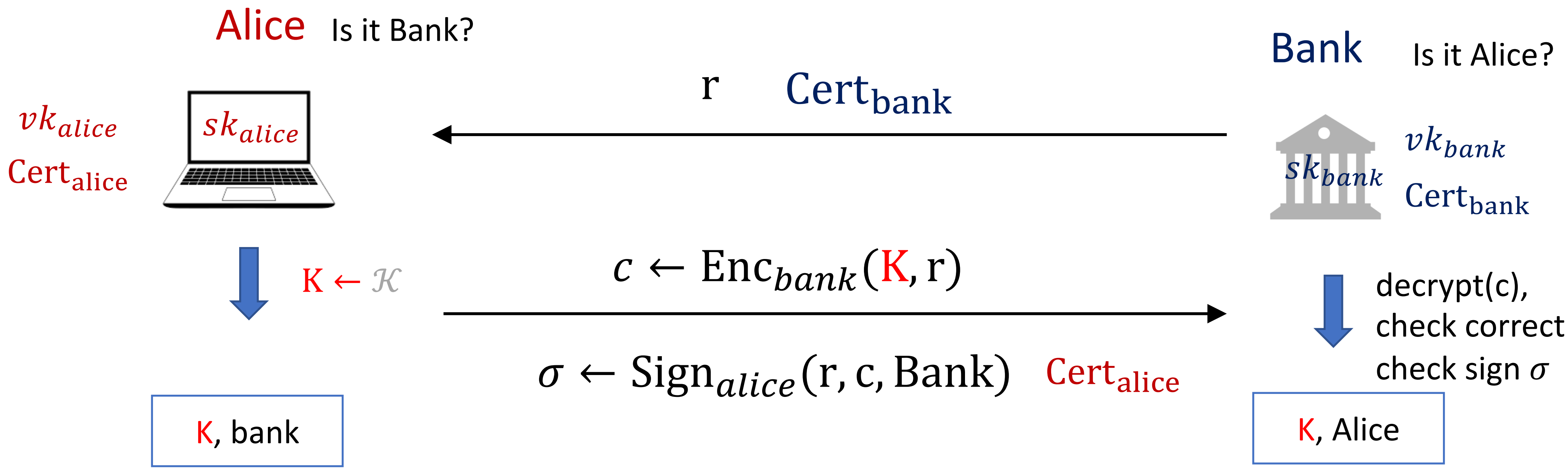
AKE1 of section 21.2 in A Graduate Course in Applied Cryptography



- 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



Suppose a year later adversary obtains sk_{bank}

$\Rightarrow$ can decrypt all recorded traffic

Protocol #1 is used in TLS 1.2 not TLS 1.3

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 →

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

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

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

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

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

Dec c_1 with k_1
Check Sign

Delete sk

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

Dec c_2 with k_2
Check Sign

K, bank

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?

sk_{bank}

vk_{bank}

$\mathrm{Cert_{bank}}$

g^a →

g^b ←

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

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

Dec c′ with k′

Check Sign

K, bank

<u>Delete a</u>

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

K, Alice

<u>Delete b</u>

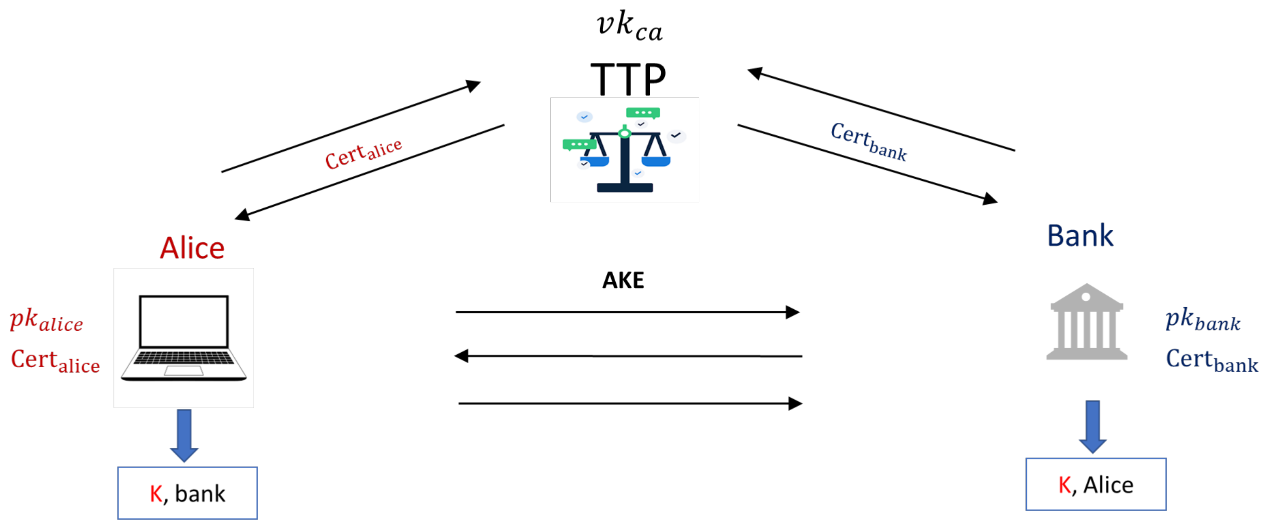
[variant of TLS 1.3]

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

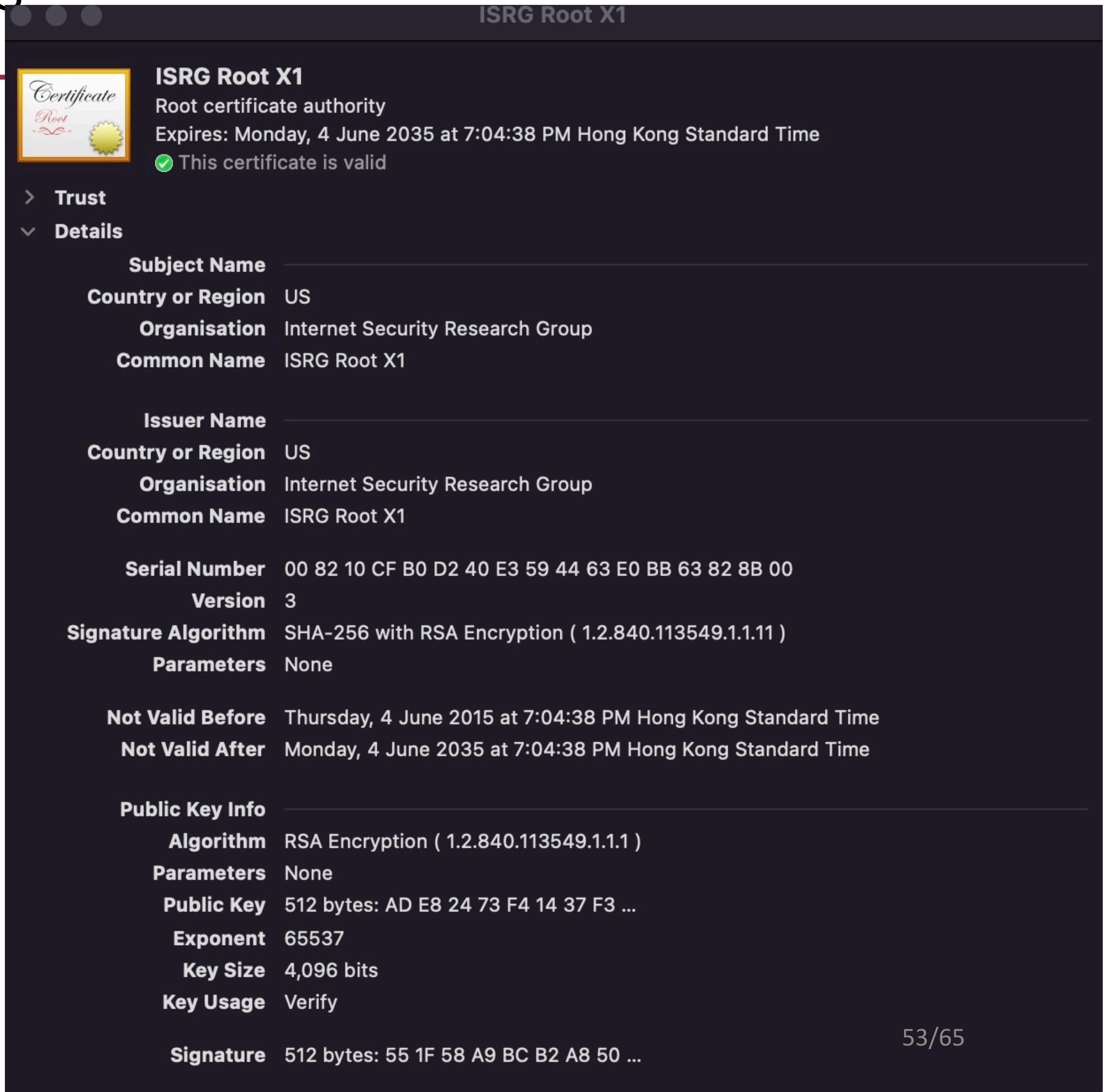
sk_{ca}

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

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



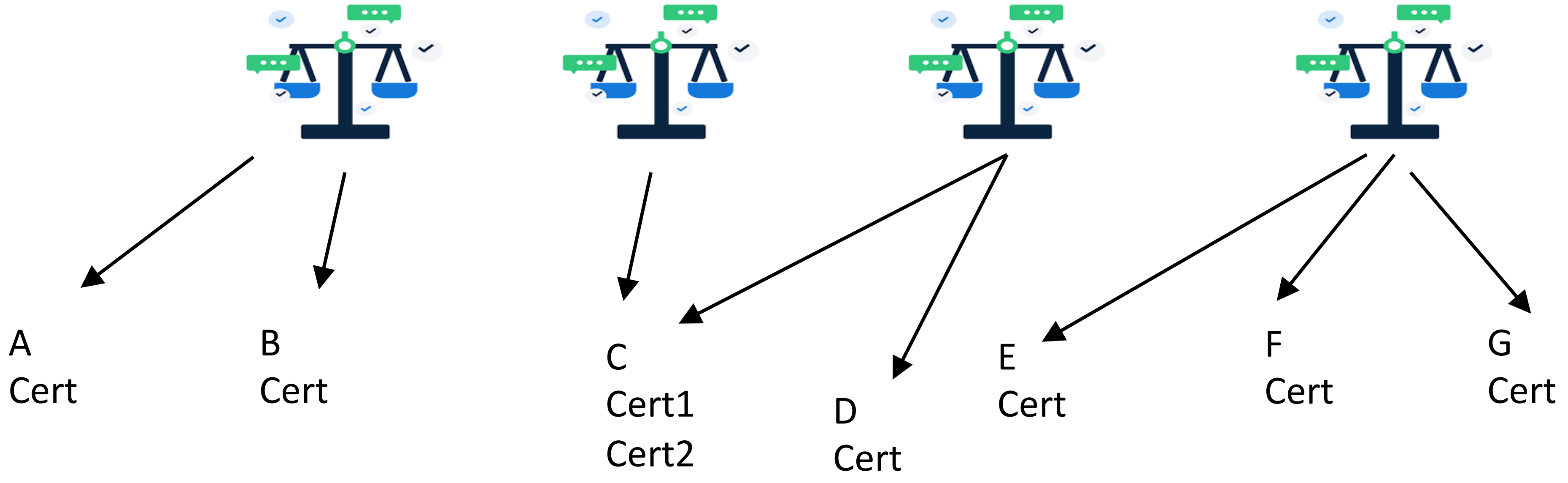
Certification Authorities(CA)

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

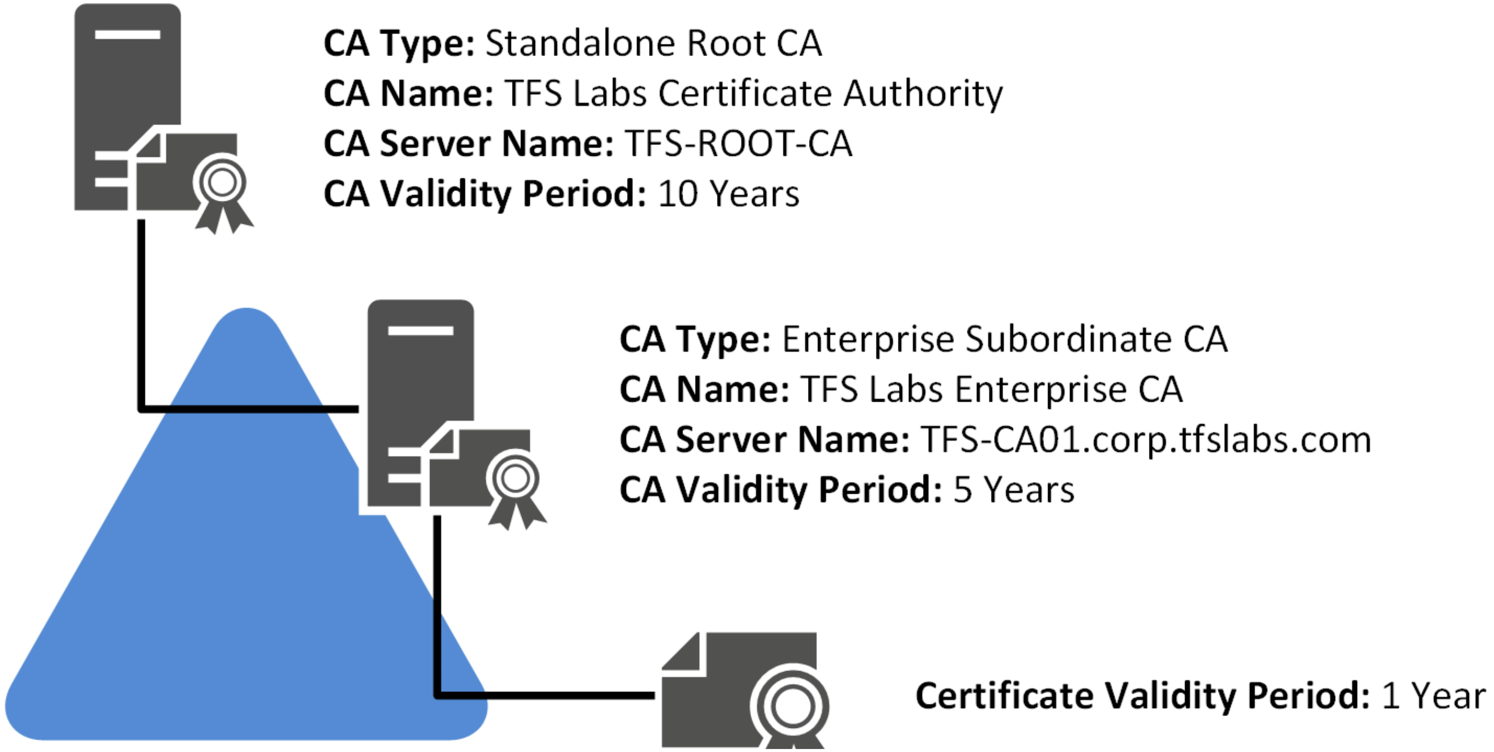
Multiple CAs



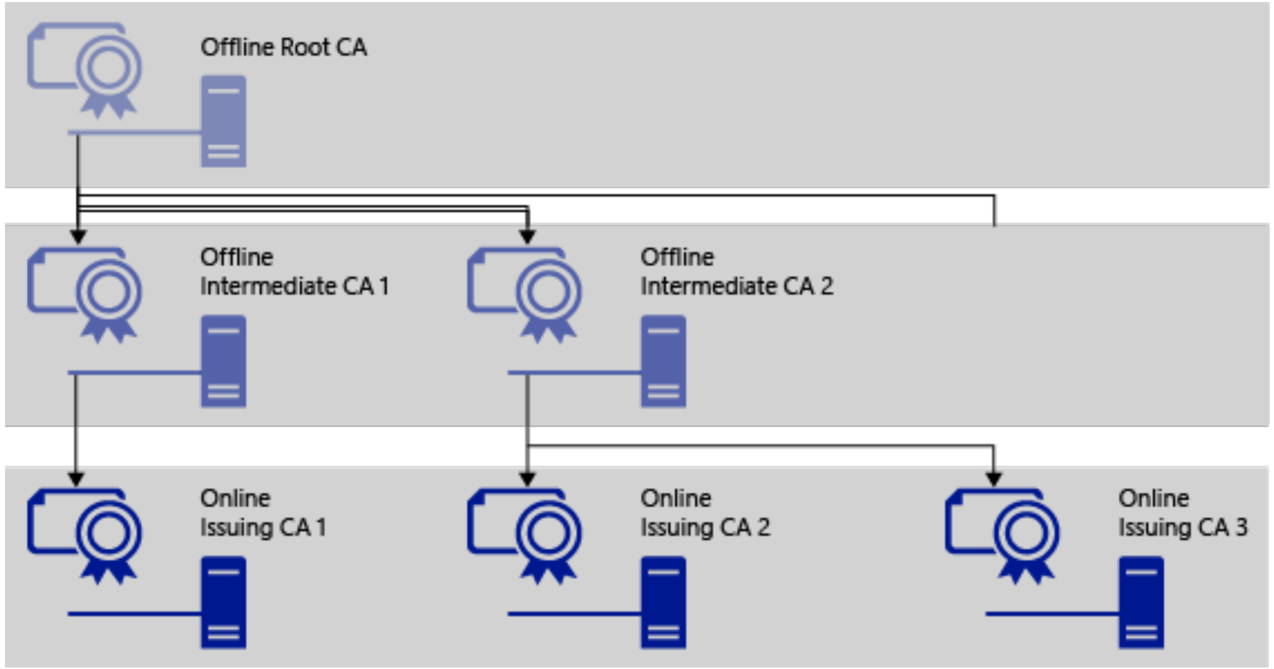
- Reduce the risk of single point of failure

Authentication Chain

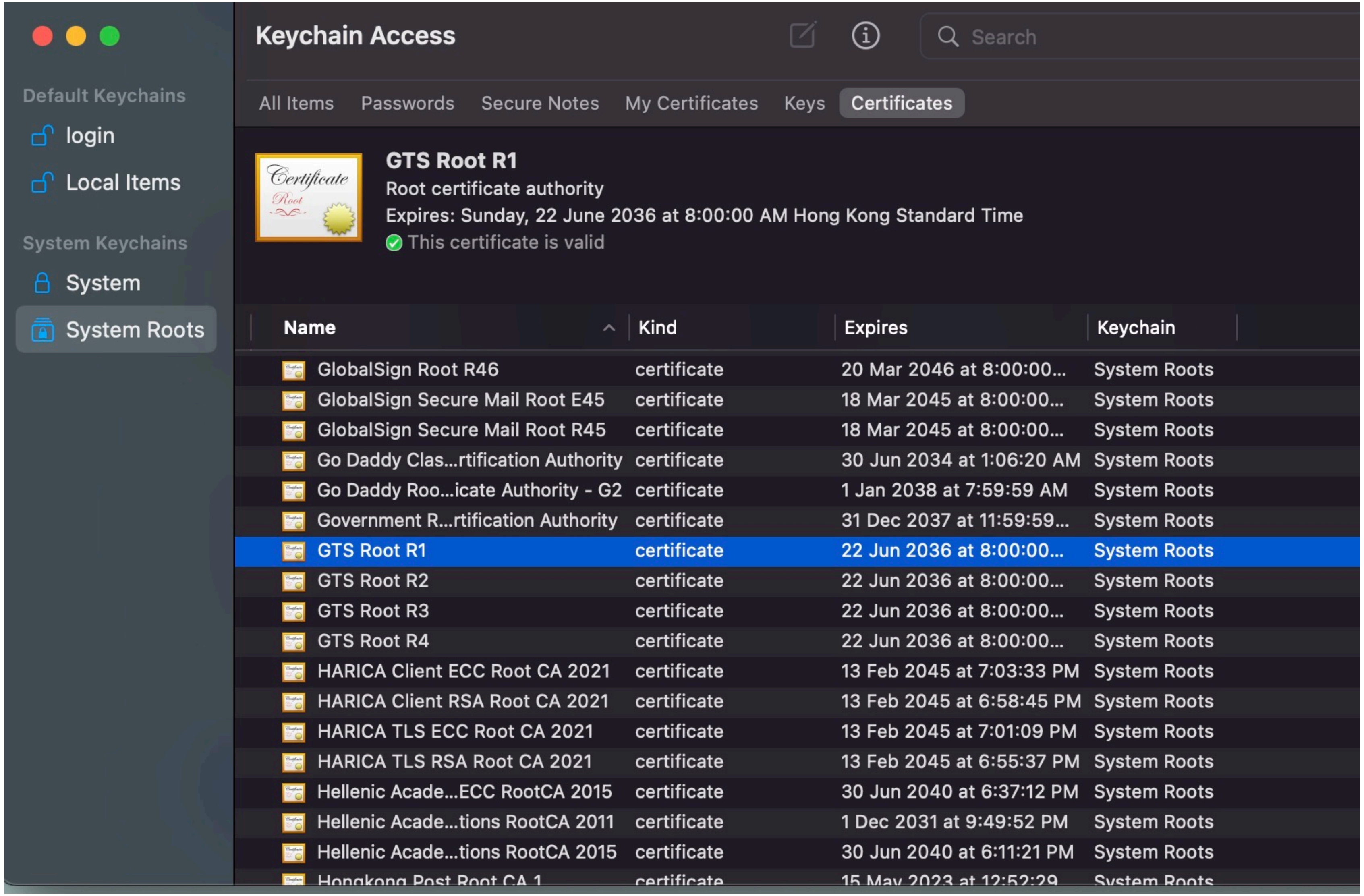
We could build the trust of certificate chains from a single Root CA



Authentication Chain



Root CA in Mac OS



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

- Zhou Jialong, Tan Zusheng
- 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