国立大学法人
電気通信大学
The University of Electro-Communications

Pham Laboratory
Integrated circuit design laboratory

「Course」
RISC-V Computer System Integration

「Lecture 07」 Introduction of Cryptosystem and Cybersecurity

Phạm Công Kha
Hoàng Trọng Thức

Tháng 9/2023

Outline

1. Pillars in Cybersecurity
2. Typical Cryptosystem
3. Current State of Hash
4. Current State of Cipher
5. Current State of Crypto-key

Outline

1. Pillars in Cybersecurity
2. Typical Cryptosystem
3. Current State of Hash
4. Current State of Cipher
5. Current State of Crypto-key

1. Pillars in Cybersecurity (1/5) Introduction

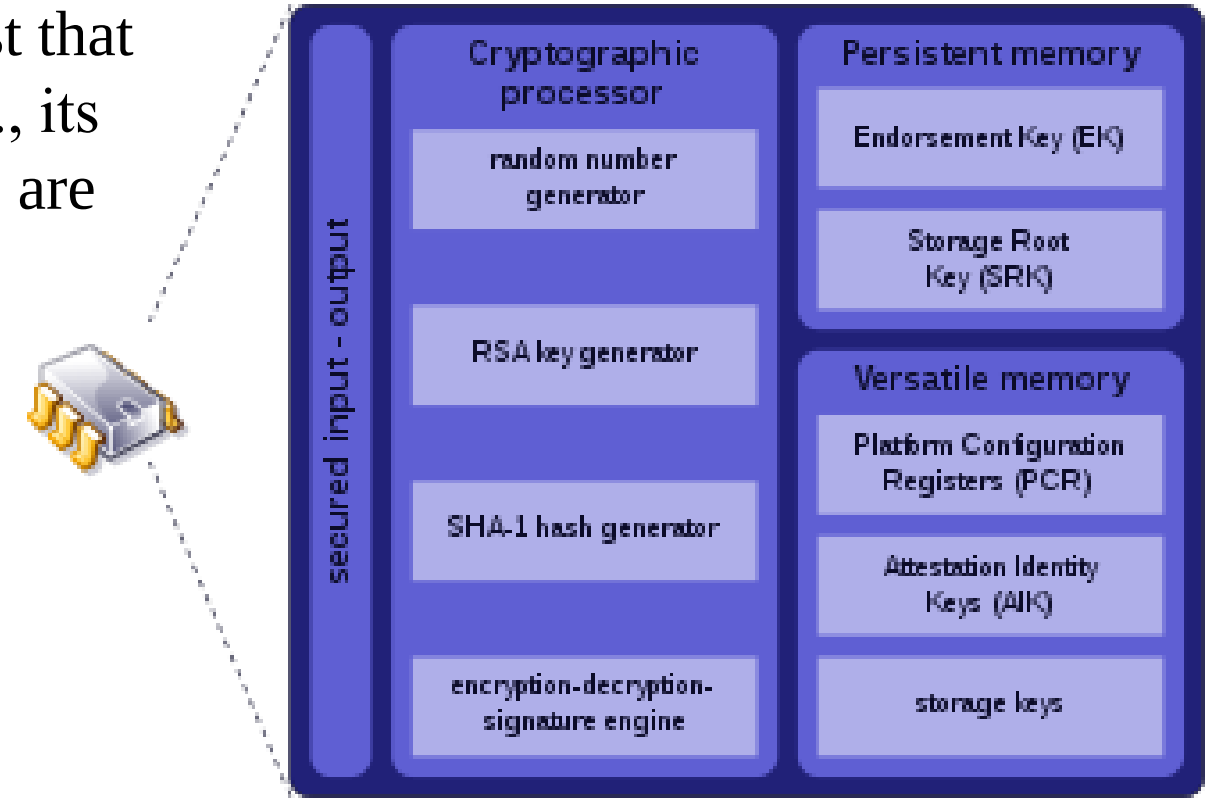
1. **Confidentiality:** the data is ciphered → un-authorized party cannot read the data
2. **Integrity:** the data is original → un-authorized party cannot modify the data
3. **Availability:** authorized parties can access the data anytime without difficulty
4. **Authentication:** verify sender and/or reader identification
   → Reader knows who the sender is, and vice versa
5. **Non-repudiation:** the data is sent only to an authorized party
   → un-authorized party cannot copy the data

**Note:* number 1, 2, and 3 are also called the CIA triad.

1. Pillars in Cybersecurity (2/5) TPM

TPM = Trusted Platform Module

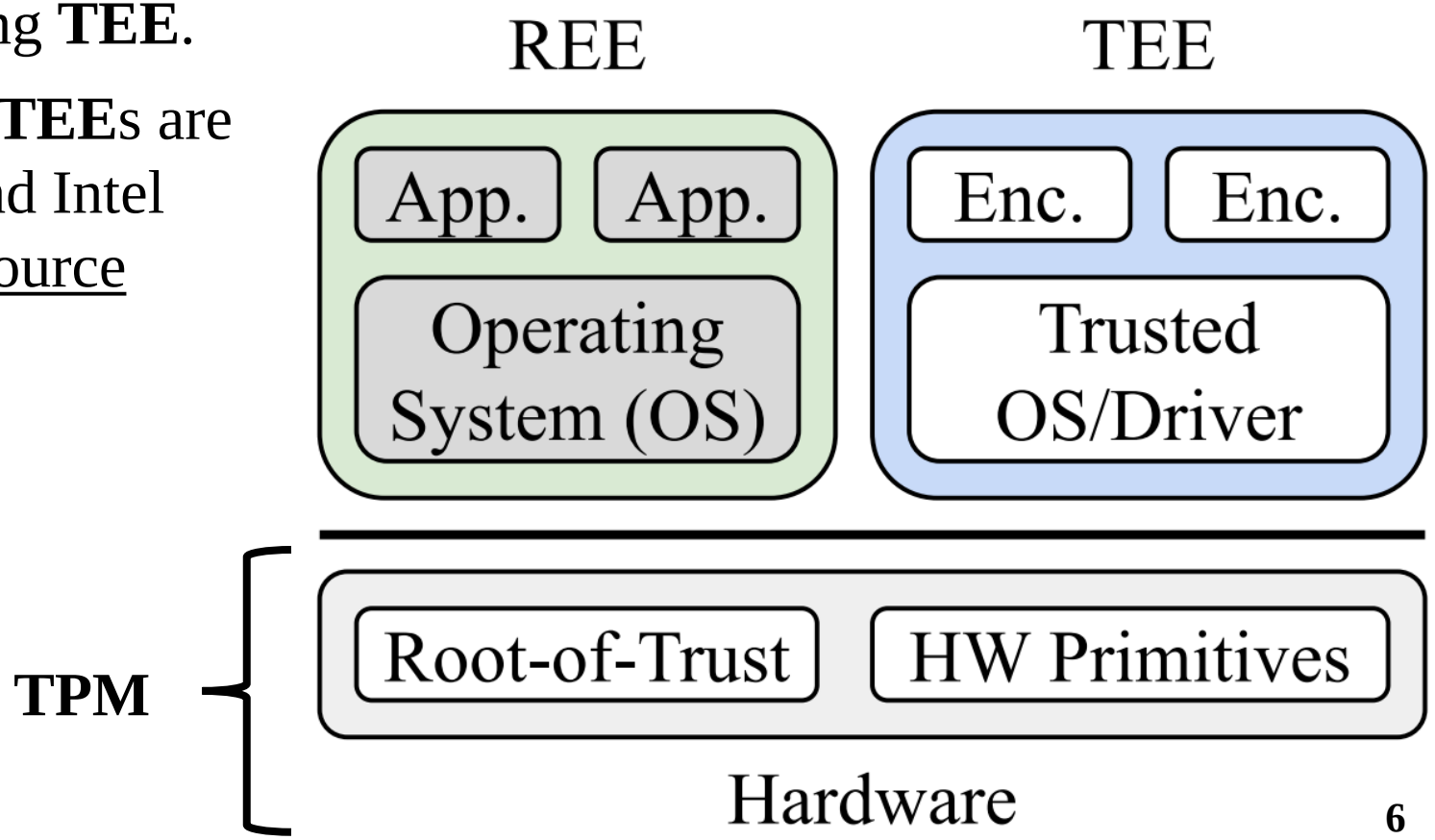
- **TPM** became a standard in 2009: ISO/IEC 11889:2009
- The latest updated was in 2011 with the **TPM** version 1.2
- **TPM** is for solving the **authentication** problem of a computer system.
- The main feature of **TPM** is the remote attestation: Through attestation, the verifier can trust that the platform/module/chip is “clean” (i.e., its vital data is safe and its critical software are not tampered).
- Then, based on the trusted platform/module/chip, we now can develop other applications for **confidentiality**, **integrity**, **availability**, etc.



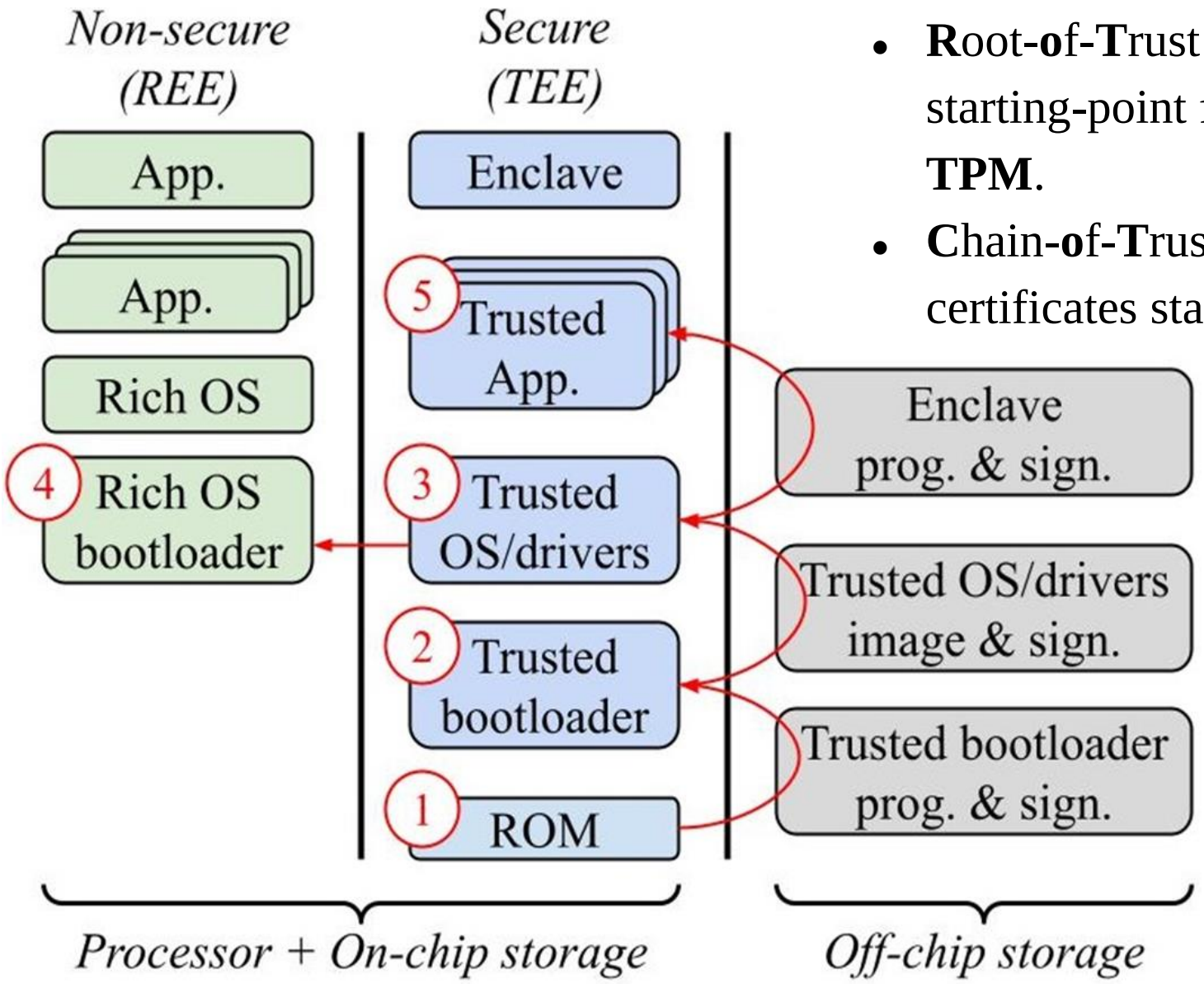
1. Pillars in Cybersecurity (3/5) TEE

TEE = Trusted Execution Environment

- **TEE** is the next step after **TPM**.
- **TPM** is for a *trusted* hardware, **TEE** is for a *trusted* **O**perating **S**ystem (**OS**)
- **TEE** needs **TPM** for the **R**oot-**o**f-**T**rust **(RoT).** Based on the **RoT**, the **C**hain-**o**f-**T**rust (**CoT**) is developed, thus creating **TEE**.
- The most common commercial **TEE**s are ARM TrustZone, AMD SEV, and Intel SGX; the most common open-source **TEE** is Keystone.



1. Pillars in Cybersecurity (4/5) Secure boot



Secure boot in **TEE**:

- **R**oot-**o**f-**T**rust (**RoT**): the first verification at reset, the starting-point for **CoT**. This should be provided by **TPM**.
- **C**hain-**o**f-**T**rust (**CoT**): a series of signatures & certificates started from the **RoT** up to the Rich OS.

Secure boot guarantee:

- All TEE-related assets *(code, trusted OS/drivers, hardware primitives)* are installed and at the initial states *(as expected by designers)*.
- Means: EVERYTHING is signature checked, and EVERY sensitive data are immutable or held in isolation.

1. Pillars in Cybersecurity (5/5) Final thought

A combination of Secure boot and **TEE** / **TPM** solves the problems of *Authentication*, *Non-repudiation*, and *Availability*.

For *Integrity* and *Confidentiality*, Secure boot + **TEE** / **TPM** are not directly solve them, but lay a foundation for other completed solutions.

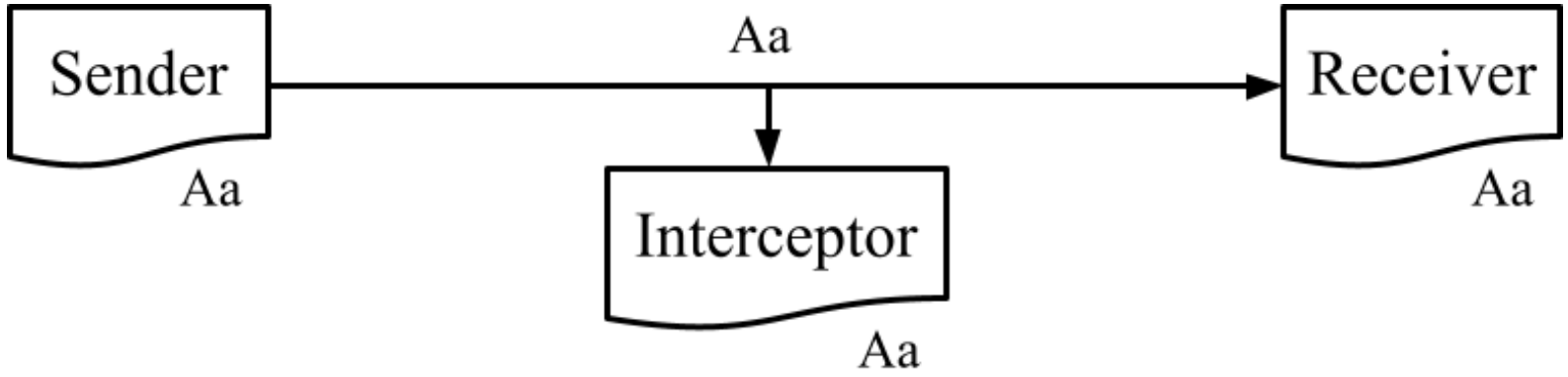
Outline

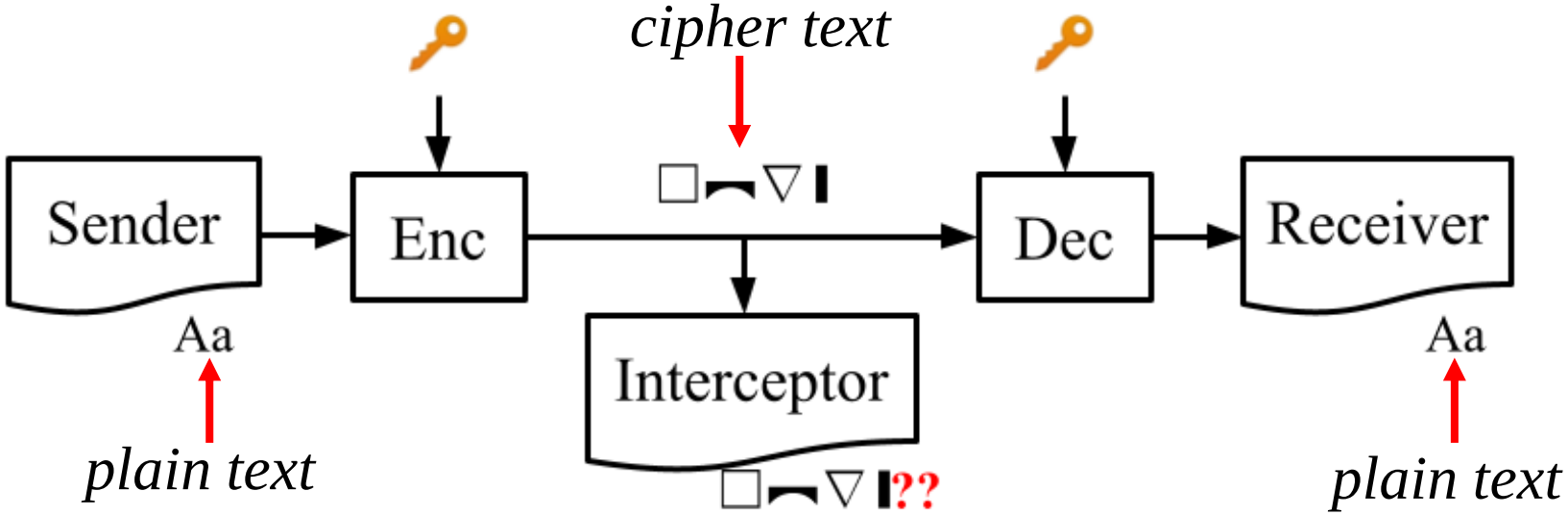
1. Pillars in Cybersecurity
2. Typical Cryptosystem
3. Current State of Hash
4. Current State of Cipher
5. Current State of Crypto-key

2. Typical Cryptosystem (1/16) Cryptosystem introduction

Every cryptosystem begins with the Eavesdropping problem:

- **Eavesdropping:** a hacker intercepts, deletes, or modifies data that is transmitted between two parties.

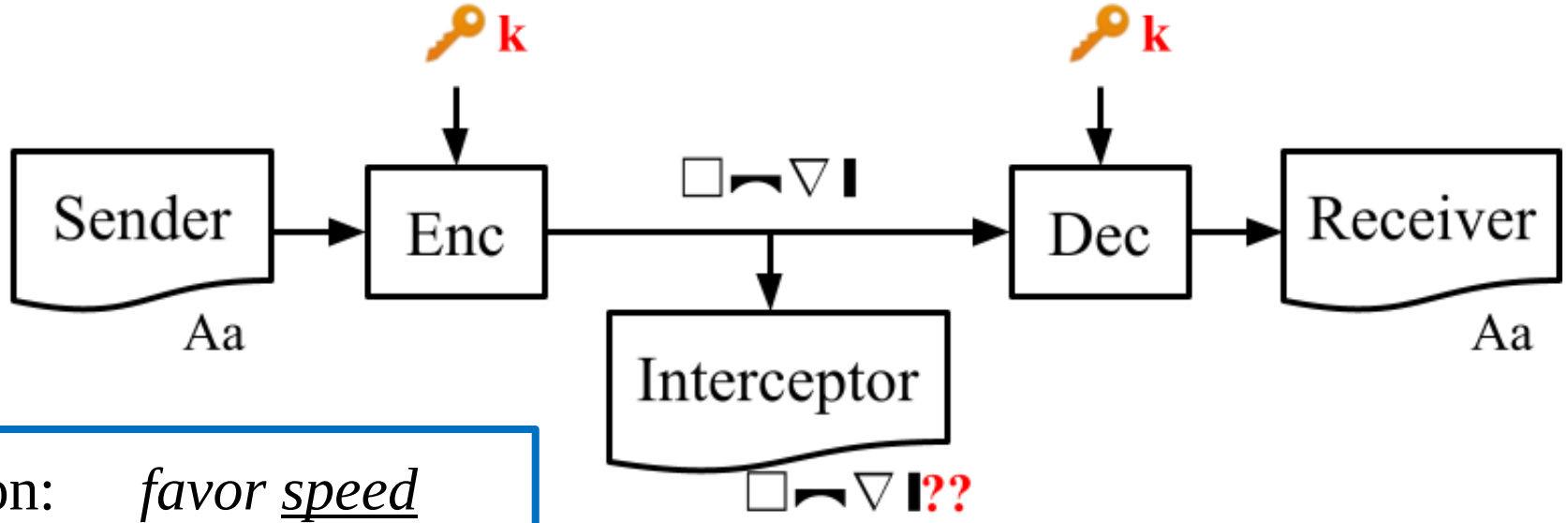




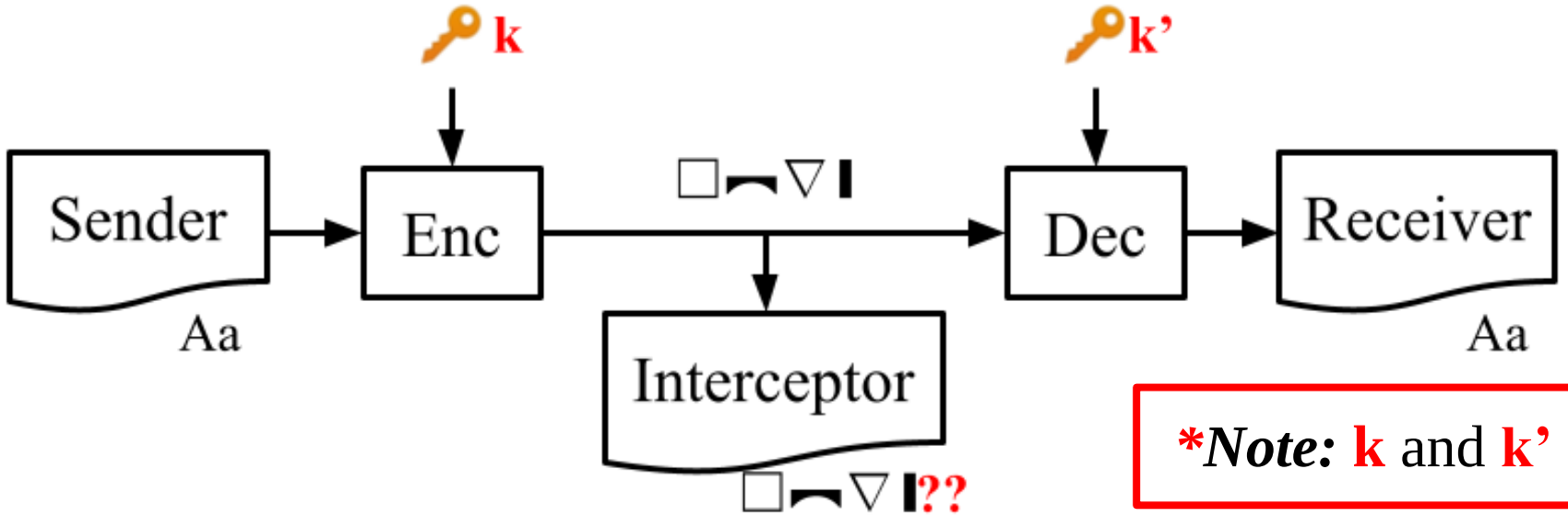
The solution is always an encryption mechanism.

2. Typical Cryptosystem (2/16) Cryptosystem introduction

Symmetric encryption:
Same key for encryption and decryption *(key* **k***)*



- **Symmetric** encryption: *favor <u>speed</u>*
- **Asymmetric** encryption: *favor <u>security</u>*



Asymmetric encryption:
Different keys for encryption and decryption *(pair-key* **k** *and* **k'***)*

***Note:** **k** and **k'** are interchangeable.

2. Typical Cryptosystem (3/16) Cryptosystem introduction

We need to transfer data securely over the *untrusted* transmission line.

⇒ Use a cipher algorithm to encrypt/decrypt the data before/after the transmission.

⇒ The key **k** of a cipher algorithm needs to be agreed upon before the transmission.

⇒ We need to transfer the key **k** securely over the *untrusted* transmission line.

This is a classical

chicken-and-egg problem

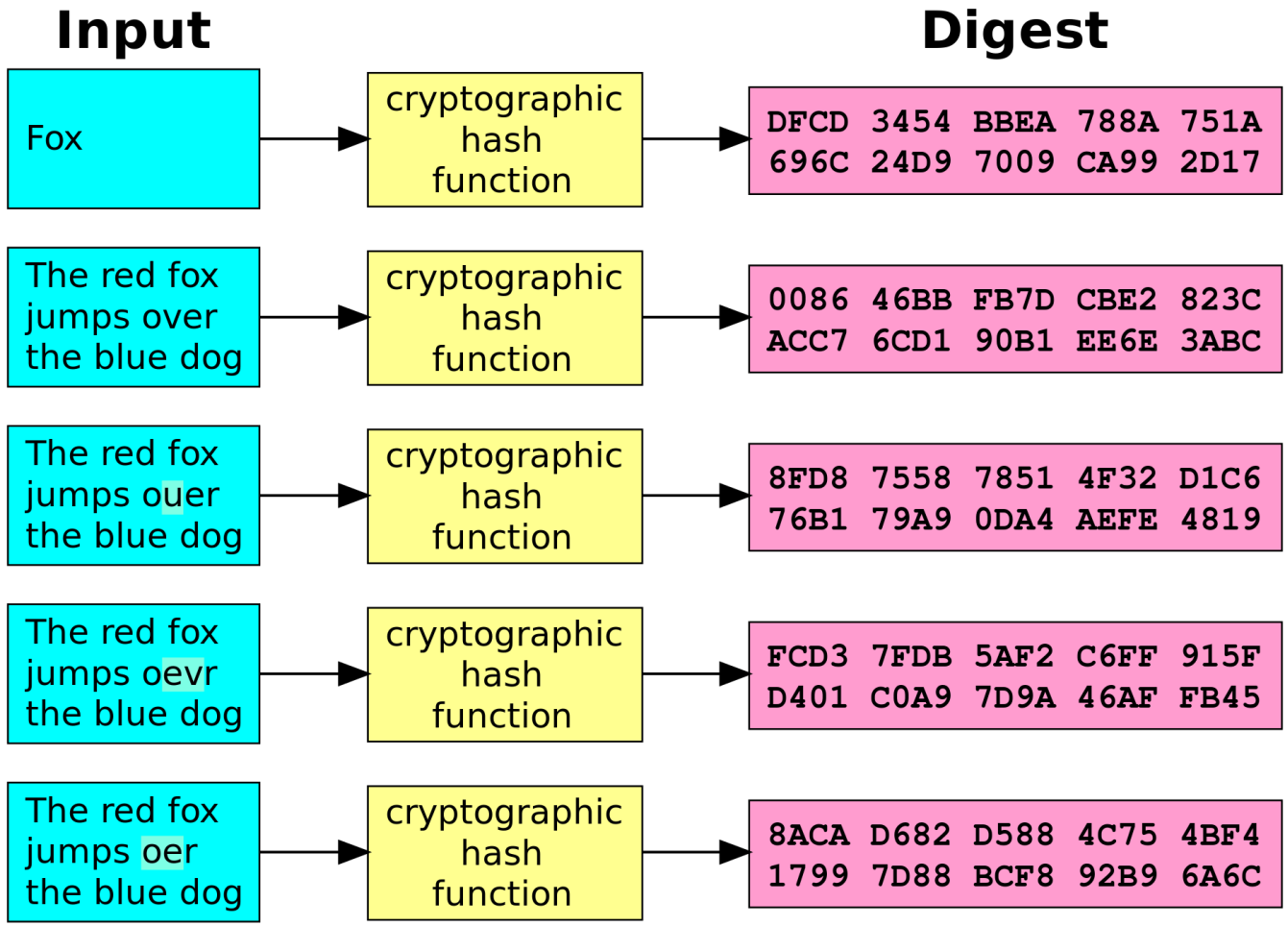
In-a-nut-shell:

Sender and **Receiver** need to share the *key* **k** before transmission: How we can do that?

Solve this problem is the main goal of a cryptosystem

2. Typical Cryptosystem (4/16) The hash concept

Hash function, also called **digest** function, is a function that converts a given string *(or any data alike)* with any length to a fixed-length result.



The input length could be ***10GB*** or even ***null***, the output length is always fixed.

For example:

- CRC-32: always output 32-bit
- SHA-256: always output 256-bit
- MD5-128: always output 128-bit

2. Typical Cryptosystem (5/16) The hash concept

Hash function is a one-way function ⇒ We cannot restore the original data after hashed.

Imagine the *meat grinder*:

The process is irreversible.

The more chaotic the **result** is, the better the **hash** algorithm.

SHA256 online hash function

```
1234567890
```

→

```
c775e7b757ede630cd0aa1113bd102661ab38829ca52a6422ab782862f268646
```

SHA256 online hash function

```
1234567890.
```

→

```
37a82583eeff5905aa8d96e429cb186aacd44466a14a5a2c3795a84b5404381c
```

2. Typical Cryptosystem (6/16) The crypto-key concept

Crypto-key scheme will use an **asymmetric encryption** algorithm and have three functions: `gen-key()`, `sign()`, and `verify()`

`(k,k′) = gen-key(x)`

- Input `x` : called a seed, usually a random number
- Output `k` and `k′` : called a pair-key, interchangeable, have the same fixed-length

`s = sign(m,k)`

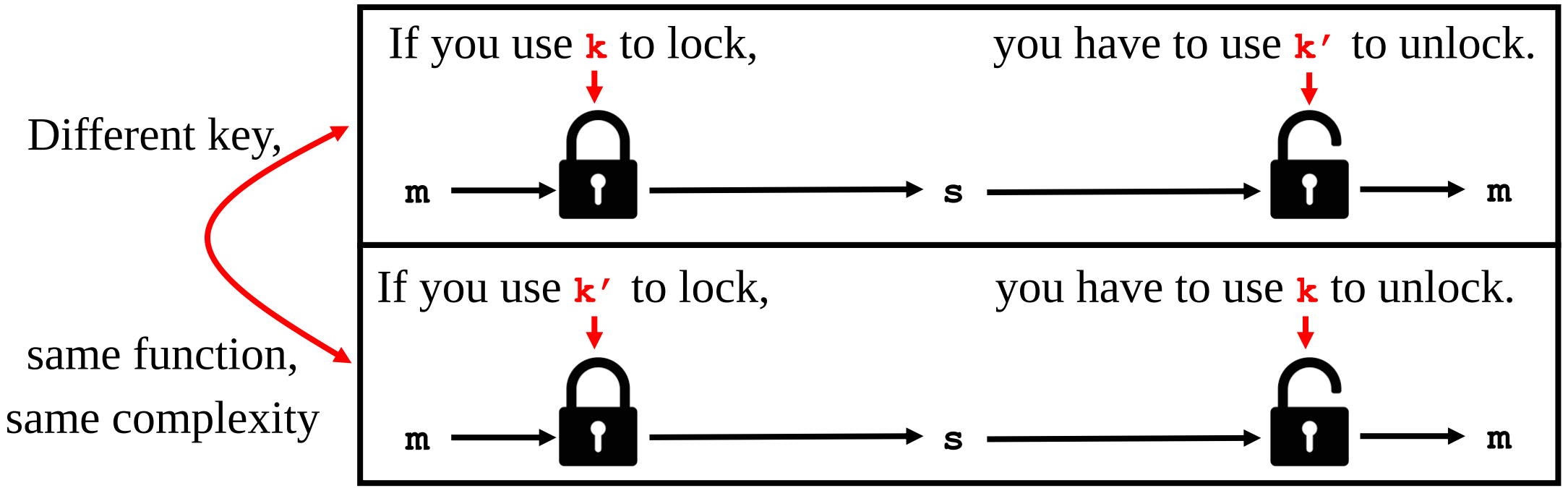
- Input `m` : called a message, could be anything
- Input `k` : called a key, retrieve from `gen-key()`
- Output `s` : called a signature, has a fixed-length

`m′ = verify(s,k′)`

- Input `s` : called a signature, retrieve from `sign()`
- Input `k′` : called a key, retrieve from `gen-key()`
- Output `m′` : called a retrieved-message, suppose to be identical with `m`

2. Typical Cryptosystem (7/16) The crypto-key concept

The **pair-key** is interchangeable



From the pair-key of **k** and **k′**, one will be chosen as public key ***P***, and the other as secret key ***S***.

The only different is, after one has been chosen as public key ***P*** *(publish for everyone to know)*, you have to conceal the secret key ***S*** *(only known to yourself)*.

2. Typical Cryptosystem (8/16) Completed solution

Back to the initial problem…

Let's both parties have their own pair-keys

Side A

$(P_A, S_A) = \texttt{gen-key}(a)$

P_A is know by everyone on the internet.

On the internet, only **A** know its S_A.

Side B

$(P_B, S_B) = \texttt{gen-key}(b)$

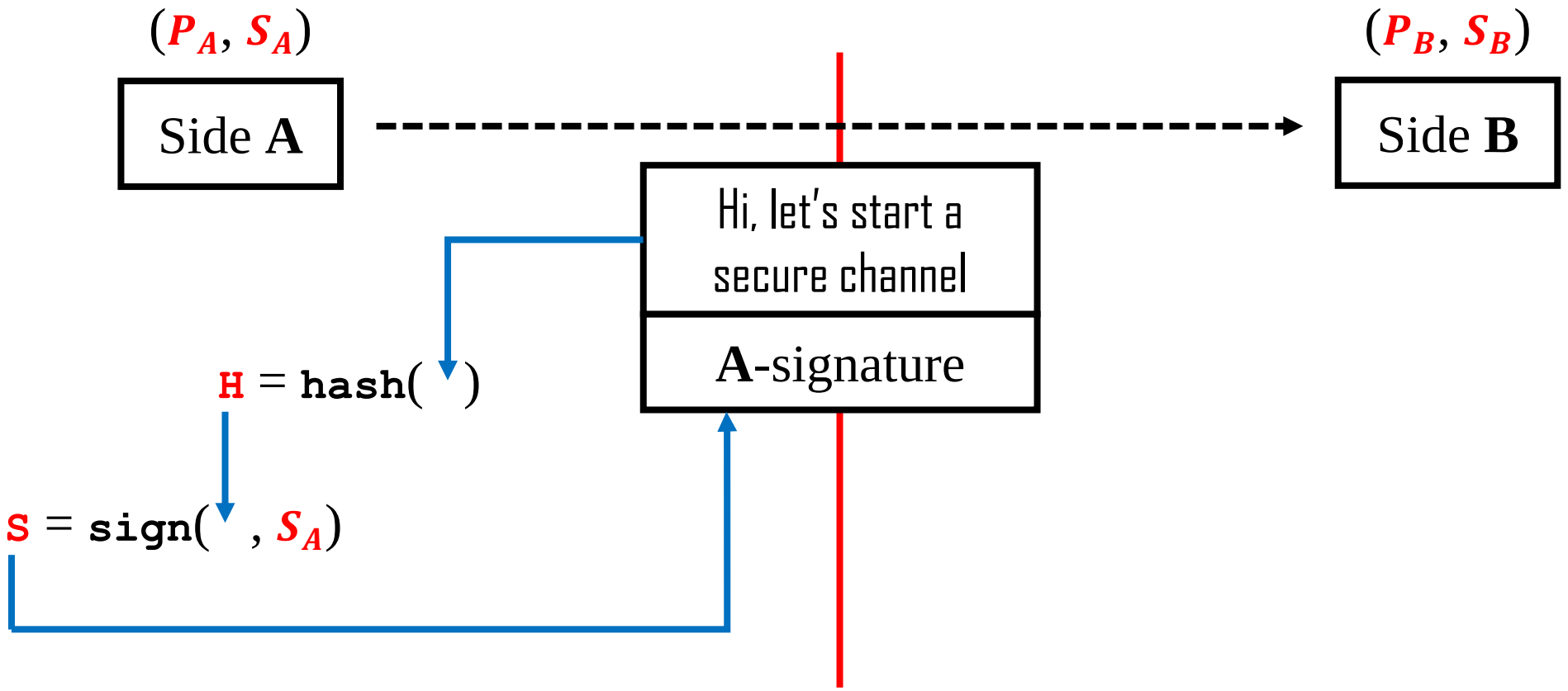
P_B is know by everyone on the internet.

On the internet, only **B** know its S_B.

- This step usually run offline with the **highest** *security settings*.
- The generated pair-keys now become their **identities**.

2. Typical Cryptosystem (9/16) Completed solution

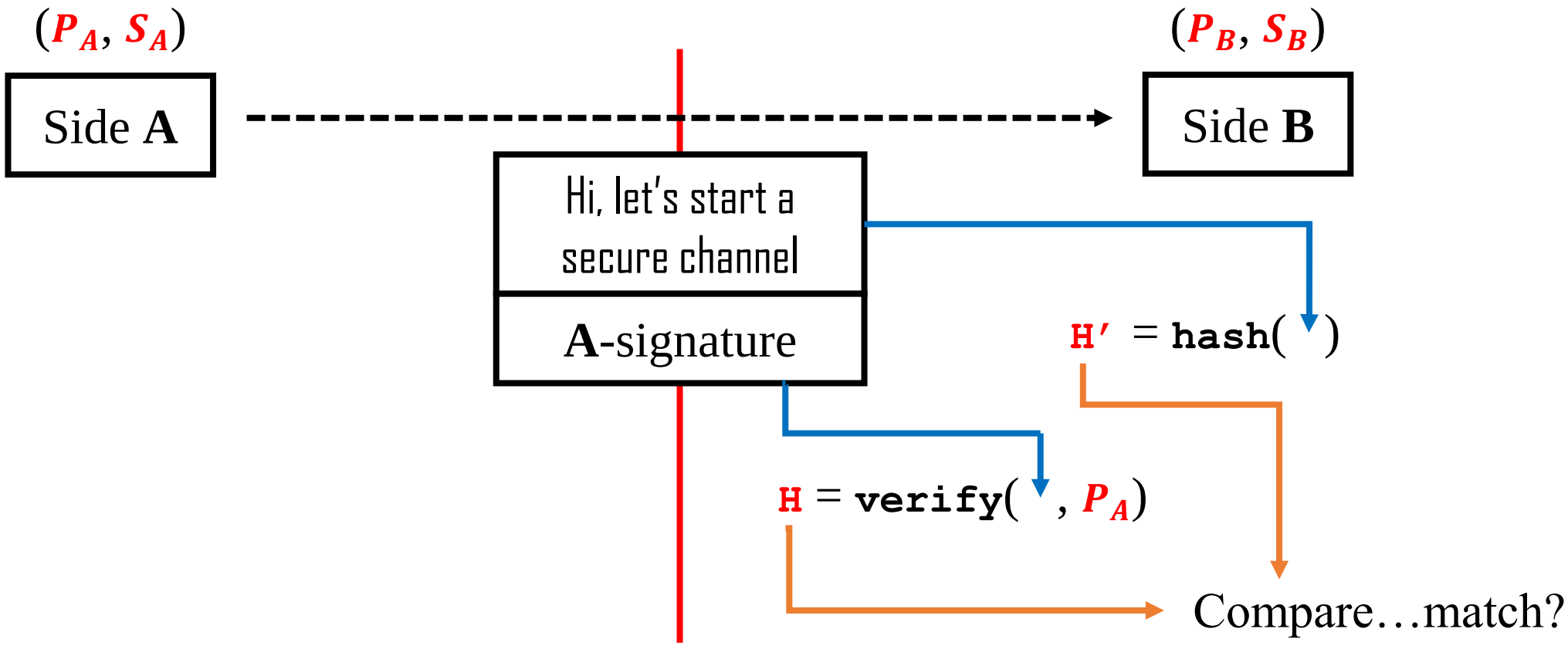
Side **A** wants to start the secure channel.



- The body of message, `m`, is not encrypted *(yet)*.
- The **A**-signature, `S`, is attached to `m` for later verification on the **B** side.

2. Typical Cryptosystem (10/16) Completed solution

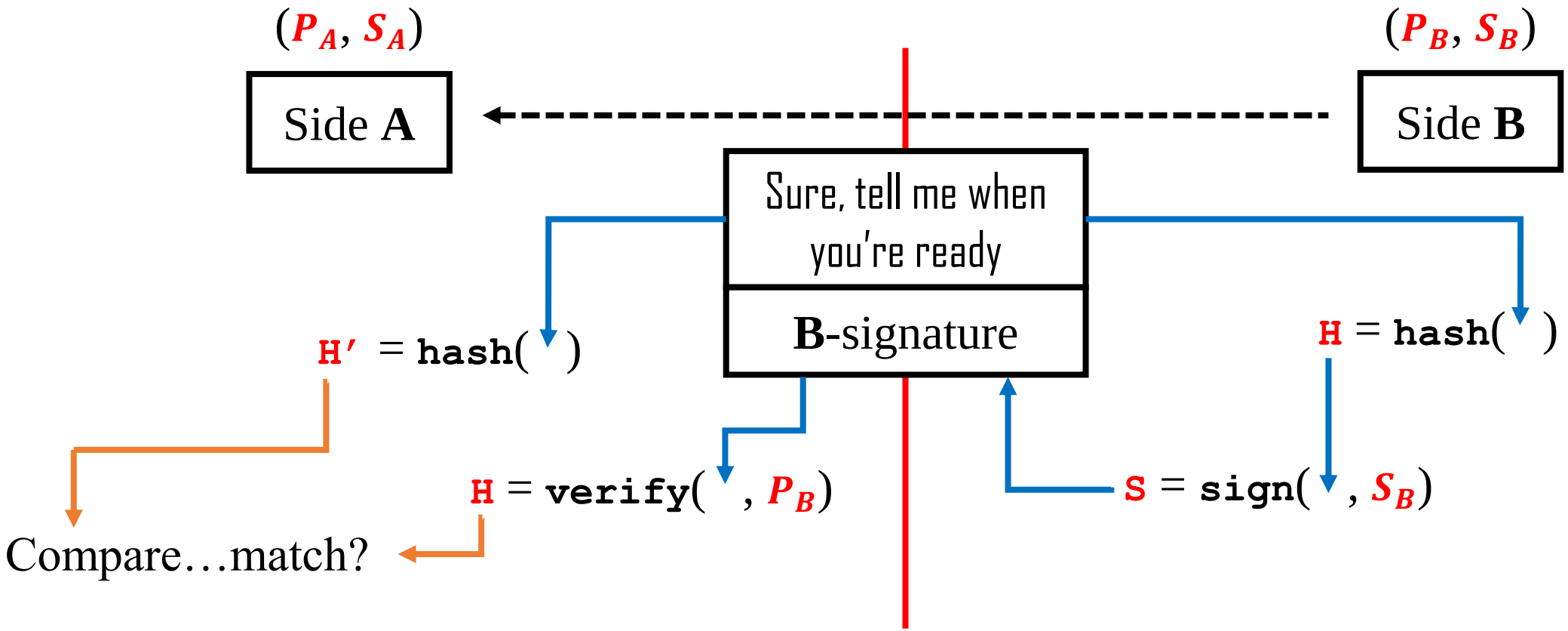
Side **B** receives the message and checks its source.



- **B** will know if *anyone* try to fake **A**'s **identity**.
- **B** will know if *anyone* try to modify the **original message**.

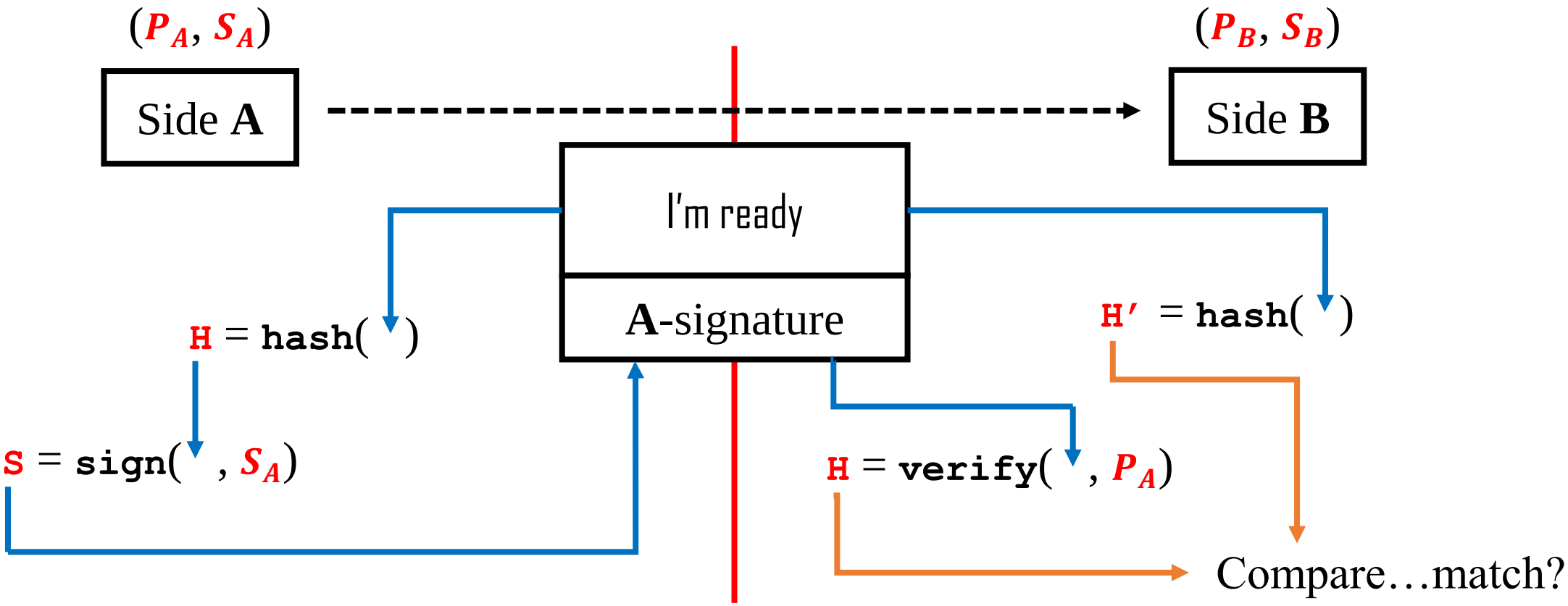
2. Typical Cryptosystem (11/16) Completed solution

Side **B** acknowledges the initial message.



- Using the same mechanism to send and check data.
- Until now, the body of message, `m`, is still not encrypted *(yet)*.

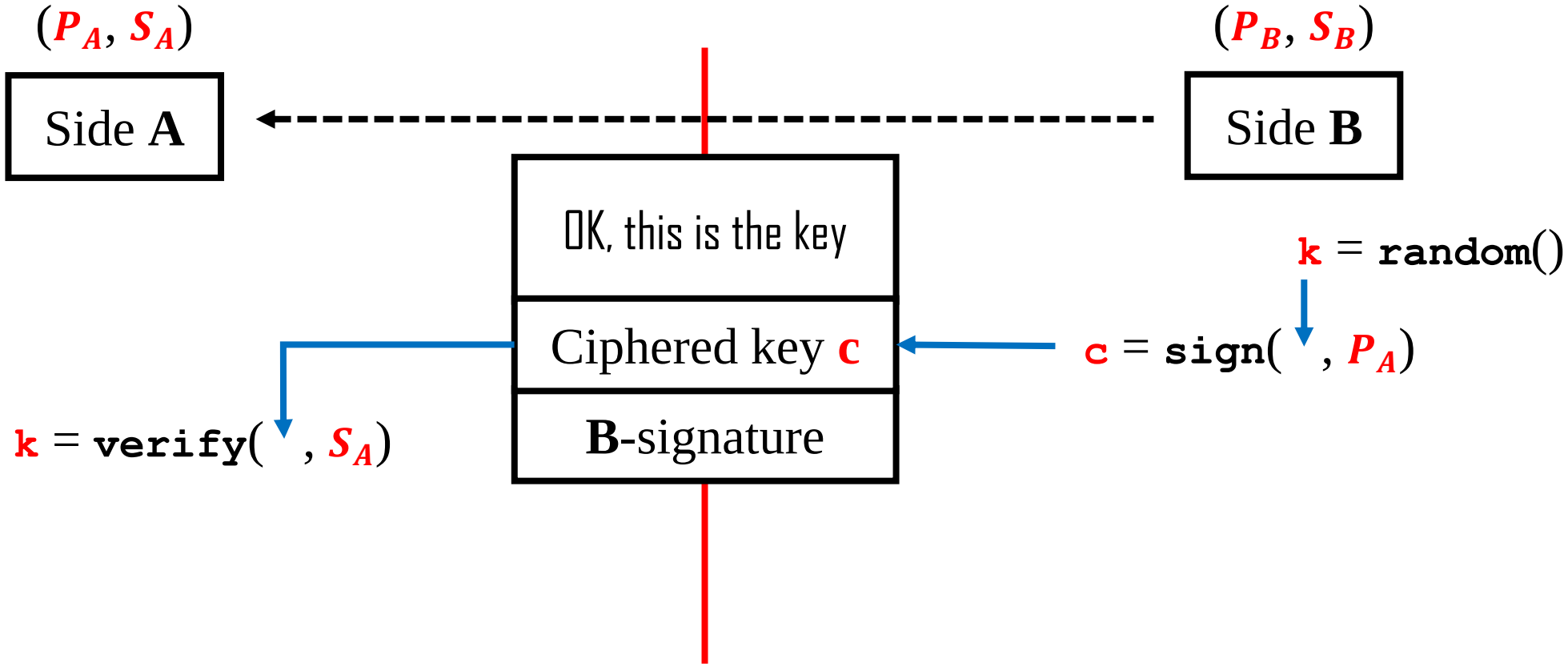
2. Typical Cryptosystem (12/16) Completed solution



- Using the same mechanism to send and check data.
- Until now, the body of message, **m**, is still not encrypted *(yet)*.

2. Typical Cryptosystem (13/16) Completed solution

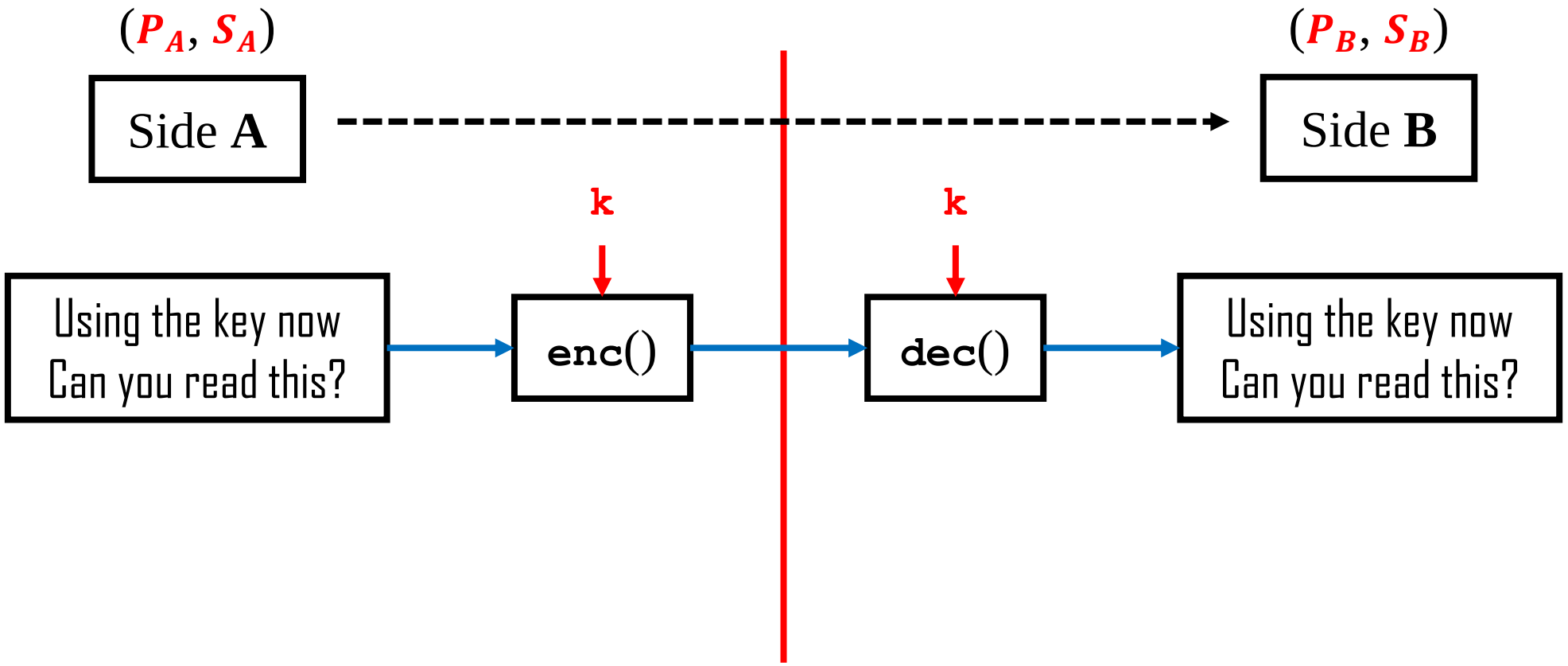
Side **B** sends the key **k** over to **A**.



- From everybody, only **A** can unlock the ciphered key **c** to retrieve the **k**.
- The attached **B**-signature is still needed for checking its source.

2. Typical Cryptosystem (14/16) Completed solution

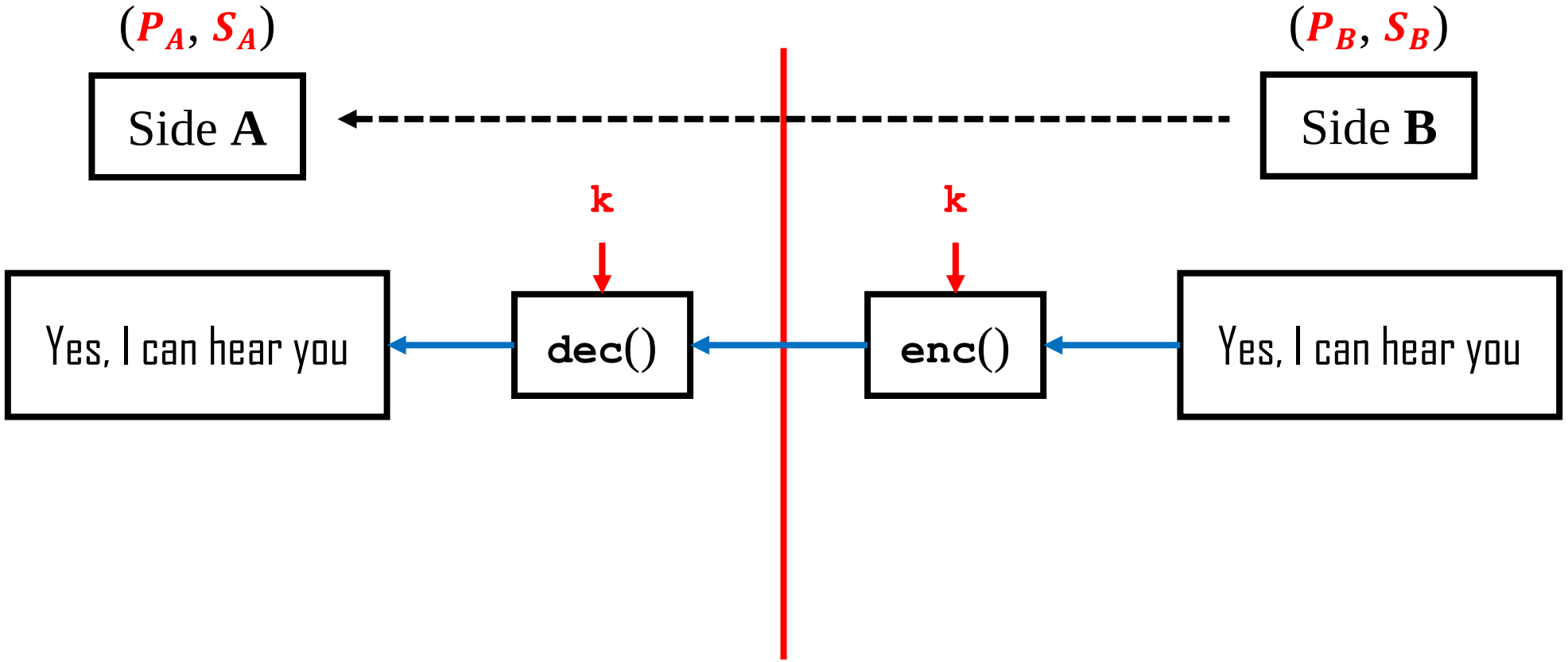
Side **A** tests the key **k**.



- **A** tests the newly received key **k**.
  Now, the body of message, **m**, is encrypted for the very first time.

2. Typical Cryptosystem (15/16) Completed solution

Side **B** acknowledges, thus finishes the handshake.



- From everybody, only **A** and **B** have the key **k**.
- Now, the handshake is over. Typical cipher algorithm is now used.

2. Typical Cryptosystem (16/16) Final thought

- The primary goal of a cryptosystem is about solving the eavesdropping problem.
  ⇒ Thus, solving the *Integrity* and *Confidentiality* problems.
- A cryptosystem usually assumes that the **sender** and **receiver** themselves are trusted *(i.e., the sender and receiver **devices** are not compromised)*.
  ⇒ Therefore, a truly completed cryptosystem needs a **TPM**/**TEE** with *secure boot*.
  Furthermore, **TPM**/**TEE** + *secure boot* also solve the *Authentication*, *Non-repudiation*, and *Availability* problems.
- A typical cryptosystem needs a *hash*, a *cipher*, and a *crypto-key* scheme.
  If we view cryptosystem as a stage, then *hash*, *cipher*, and *crypto-key* are roles to be played, **not** actors. ⇒ Actors can be **changed**, but the roles are **not**.
  For example: *SHA*, *AES*, and *RSA* are actors to play the roles of *hash*, *cipher*, and *crypto-key*, respectively.

Outline

1. Pillars in Cybersecurity
2. Typical Cryptosystem
3. Current State of Hash
4. Current State of Cipher
5. Current State of Crypto-key

3. Current State of Hash (1/2) Popular algorithms

There are many **hash** algorithms were developed.

- **MD5** *(Message Digest 5)* held the top position for a very long time since its both fast and efficient. However, since its security vulnerabilities have been discovered, **MD5** is now discontinued and not recommended for any application.
- **SHA** *(Secure Hash Algorithm)* so far the most popular hash algorithm. **SHA-1** has only the *160*-bit option; both **SHA-2** and **SHA-3** have *224*, *256*, *384*, and *512*-bit options. **SHA-2** is de-facto choice in SSL/TLS protocols, which is widely used in HTTPS. **SHA-3** is the latest update in the **SHA** family since 2012.
- **CRC** *(Cyclic Redundancy Check)* is a widely used hash function, not for cryptographic applications, but for error detection in data transmission. **CRC** provides a very fast and efficient hash solution, just not for security.

Link: https://en.wikipedia.org/wiki/List_of_hash_functions

3. Current State of Hash (2/2) Latest algorithms

NIST

Information Technology Laboratory

COMPUTER SECURITY RESOURCE CENTER

PROJECTS

Hash Functions

So far, the **SHA** family continues to be the latest recommendation.

	Collision Resistance Strength in bits	Preimage Resistance Strength in bits	Second Preimage Resistance Strength in bits
SHA-1	<80	160	160 – L (M)
SHA-224	112	224	min(224, 256 – L (M))
SHA-256	128	256	256 – L (M)
SHA-384	192	384	384
SHA-512	256	512	512 – L (M)
SHA-512/224	112	224	224
SHA-512/256	128	256	256
SHA3-224	112	224	224
SHA3-256	128	256	256
SHA3-384	192	384	384
SHA3-512	256	512	512

Other choices:
the **SHAKE** family.

Link: https://csrc.nist.gov/projects/hash-functions

Outline

1. Pillars in Cybersecurity
2. Typical Cryptosystem
3. Current State of Hash
4. Current State of Cipher
5. Current State of Crypto-key

4. Current State of Cipher (1/6) Popular algorithms

There are even more **cipher** algorithms were developed.

- **AES** *(Advanced Encryption Standard)* is the most popular cipher algorithm. Due to its efficiency and relatively low complexity, it is widely used in various applications from civil to military. Even now, although many security risks of **AES** have been discovered, it is still considered “good-enough” for day-to-day applications.
- **Triple DES** *(Data Encryption Standard)*, also called **3DES** or **TDES**, can be viewed as the successor of the original **DES**, which was discontinued since 2005. By looping three **DES** together with some other modifications, **TDES** brings the security level up to modern-day requirements.
- **GCM** *(Galois/Counter Mode)* is one of the **AEAD** *(Authenticated Encryption with Associated Data)* family. The primary goal of the **GCM** is about high-speed communicating applications.

4. Current State of Cipher (2/6) Popular algorithms

There are even more **cipher** algorithms were developed.

- **MAC** *(Message Authentication Code)* is an idea that using a short piece of information attached to the original data for the authentication of the data itself. There are three main variations of the **MAC,** including **HMAC** *(Hash-MAC)*, **KMAC** *(Keccak-MAC)*, and **CMAC** *(CBC-MAC)*.
- **ChaCha20-Poly1305** is the latest update in the **AEAD** *(Authenticated Encryption with Associated Data)* family. It is the combination between the two formal algorithms of **ChaCha20** *(using as a stream cipher)* and **Poly1305** *(using as a MAC)*.

4. Current State of Cipher (3/6) Latest algorithms

NIST

Information Technology Laboratory

COMPUTER SECURITY RESOURCE CENTER

PROJECTS

Block Cipher Techniques

For block ciphers, **AES** and **Triple DES** with *theirs variations* are still recommended for a typical cipher application.

Advanced Encryption Standard (AES)

AES is specified in FIPS 197, *Advanced Encryption Standard (AES)*, which was a modes of operation designed specifically for use with block cipher algorithm

NIST announced the approval of FIPS 197, *Advanced Encryption Standard* in 2 FIPS-approved symmetric-key algorithm that may be used by U.S. Governme information.

The AES Development details have been archived.

Triple DES

Triple DES is specified in SP 800-67 Revision 2, *Recommendation for the Triple* was approved in November 2017. This revision supersedes SP 800-67 Rev. 1,

Link: https://csrc.nist.gov/Projects/block-cipher-techniques

4. Current State of Cipher (4/6) Latest algorithms

NIST

Information Technology Laboratory

COMPUTER SECURITY RESOURCE CENTER

PROJECTS

Message Authentication Codes MAC

For message authentication, **HMAC**, **KMAC**, and **CMAC** are recommended.

Approved Algorithms

Currently, there are **three (3)** approved* general purpose MAC algorithms: **HMAC, KMAC** and **CMAC**.

Keyed-Hash Message Authentication Code (HMAC)

FIPS 198-1, *The Keyed-Hash Message Authentication Code (HMAC)* (July 2008), specifies a mechanism f using an approved hash function. The approved hash functions are specified in FIPS 180-4, Secure Ha SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. Specific guidelines in c properties are provided in NIST SP 107 Revision 1, *Recommendation for Applications Using Approved H*

KECCAK Message Authentication Code (KMAC)

KMAC is specified in SP 800-185, *SHA-3 Derived Functions: cSHAKE, KMAC, TupleHash and ParallelHash* keyed hash function based on KECCAK, which is specified in FIPS 202, *SHA-3 Standard: Permutation-Ba Output Functions*. There are two variants of KECCAK, KMAC128 and KMAC256.

The CMAC Mode for Authentication

Link: https://csrc.nist.gov/projects/message-authentication-codes

4. Current State of Cipher (5/6) Latest algorithms

The main goal of lightweight cryptography is for small systems such as in low-power IoT applications.

PROJECTS

Lightweight Cryptography

For lightweight cryptography, **Ascon** just has been selected for a standard.
The selection is very recently, at Feb. 2023.

- **Round 1.** In March 2019, NIST received 57 submissions to be cor lightweight cryptography standardization process began with th August 2019. NISTIR 8268 explains the evaluation of the first-rou to the second round of the evaluation process.
- **Round 2.** The second round of the NIST lightweight cryptograph Round 2 in August 2019 and concluded when the finalists were a of the second-round candidates and names 10 finalists.
- **Final Round.** The final round of the process began with the ann the selection of the Ascon family in February 2023. NISTIR 8454 selection of the Ascon family.

Link: https://csrc.nist.gov/Projects/lightweight-cryptography

4. Current State of Cipher (6/6) Latest algorithms

New frontier for cryptography developers: **P**ost-**Q**uantum **C**ryptography *(PQC)*.

Link: https://csrc.nist.gov/projects/post-quantum-cryptography

Outline

1. Pillars in Cybersecurity
2. Typical Cryptosystem
3. Current State of Hash
4. Current State of Cipher
5. Current State of Crypto-key

5. Current State of Crypto-key (1/2) Popular algorithms

In contrast with **cipher** and **hash** algorithms, there are not many **crypto-key** schemes have been standardized.

- **RSA** *(Rivest-Shamir-Adleman)* is the oldest, the most popular, and also the most currently widely used crypto-key scheme. Invented by three cryptography specialists named Ron **R**ivest, Adi **S**hamir, and Leonard **A**dleman, thus the name **RSA**. In **RSA**, increasing the *number of bits* will increase the level of *security* (and *complexity*) exponentially.
- **DSA** *(Digital Signature Algorithm)* can be viewed as the successor of **RSA**. Based on the **RSA** idea, **DSA** modified some procedures to decrease the level *complexity* (thus enhancing *speed*) while trying to maintain the same level of *security*.
- **ECDSA** *(Elliptic Curve DSA)* is the most popular variant of **DSA**. It uses elliptic curves for *gen-key* instead of prime-factor modular as in the conventional **RSA**.

5. Current State of Crypto-key (2/2) Latest algorithms

NIST

Information Technology Laboratory

COMPUTER SECURITY RESOURCE CENTER

PROJECTS

RSA, **DSA**, and **ECDSA** are all recommended. **ECDSA** will completely replace the old **RSA** in the *near future*.

Digital Signatures

As an electronic analogue of a written signature, a digital signature provides assurance that:

1. the claimed signatory signed the information, and
2. the information was not modified after signature generation.

Federal Information Processing Standard (FIPS) 186-4, *Digital Signature Standard (DSS)*, specifies three NIST-approved digital signature algorithms: **DSA**, **RSA**, and **ECDSA**. All three are used to generate and verify digital signatures, in conjunction with an approved hash function specified in FIPS 180-4, *Secure Hash Standard* or FIPS 202, *SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*.

February 3, 2023

NIST published Federal Information Processing Standard (FIPS) 186-5, *Digital Signature Standard (DSS)*, along with NIST Special Publication (SP) 800-186, *Recommendations for Discrete Logarithm-based Cryptography: Elliptic Curve Domain Parameters*.

Link: https://csrc.nist.gov/Projects/digital-signatures

THANK YOU

Tháng 9/2023