



•

5 0 . 0 0 5 C S E

Natalie Agus
Information Systems Technology and Design
SUTD

IMPORTANT SECURITY PROPERTIES



Confidentiality

No one else, except sender and receiver should understand the messages exchanged (encryption)



Authentication

Sender and receiver should confirm the identity of one another



Message Integrity

Sender and receiver want messages exchanged to be unaltered in transit.
Altered messages should be detected.



Availability

Communication link between sender and receiver should be available



Access

Communication link between sender and receiver should be accessible

POSSIBLE SECURITY ATTACKS



Eavesdrop

Intercept messages



Alter messages

Actively insert messages into connection



Impersonate

Fake (**spoof**) source address in packet or any part of the packet



Denial of Service

Prevent service from being used by others



Hijack

Take over ongoing connection by removing sender or receiver unknowingly to either party

CRYPTOGRAPHY

Protects **confidentiality**, hence **preventing** attackers from **eavesdropping**

H O W ?

| | |
|-------------|----------------------------|
| plaintext: | abcdefghijklmnopqrstuvwxyz |
| ciphertext: | mnbvcxzasdfghjklpoiuytrewq |



Naive attempt: Substitution Cipher

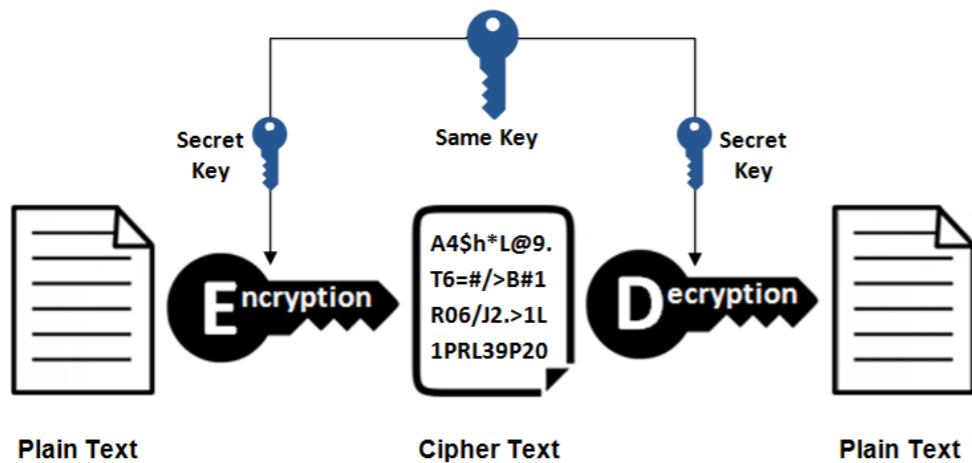
Mapping from a set of 26 letters to another set of 26 letters

Disadvantage: the frequencies of the letters are not masked at all

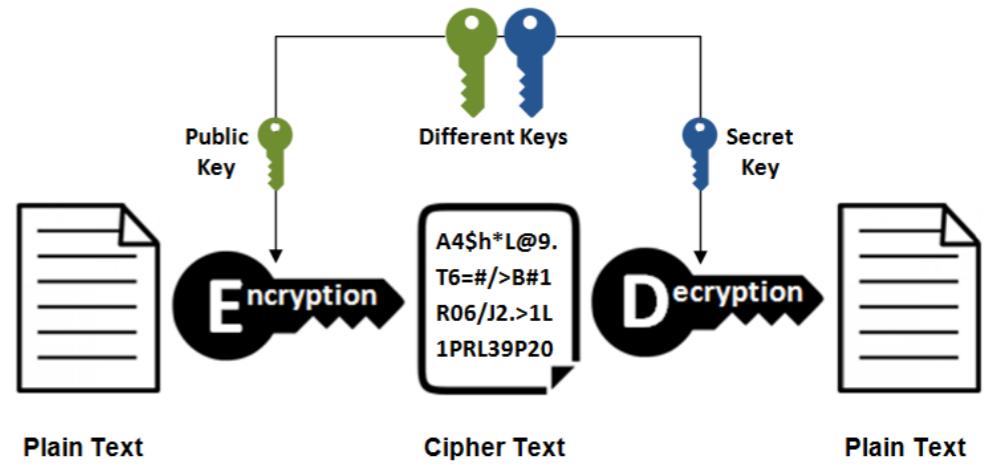
C R Y P T O G R A P H Y

Protects **confidentiality**, hence
preventing attackers from **eavesdropping**

There's two good methods to encrypt and decrypt plaintext



Symmetric Key



Asymmetric Key

• S Y M M E T R I C K E Y C R Y P T O

encryption and decryption both uses the **same** key

Data Encryption Standard (DES):

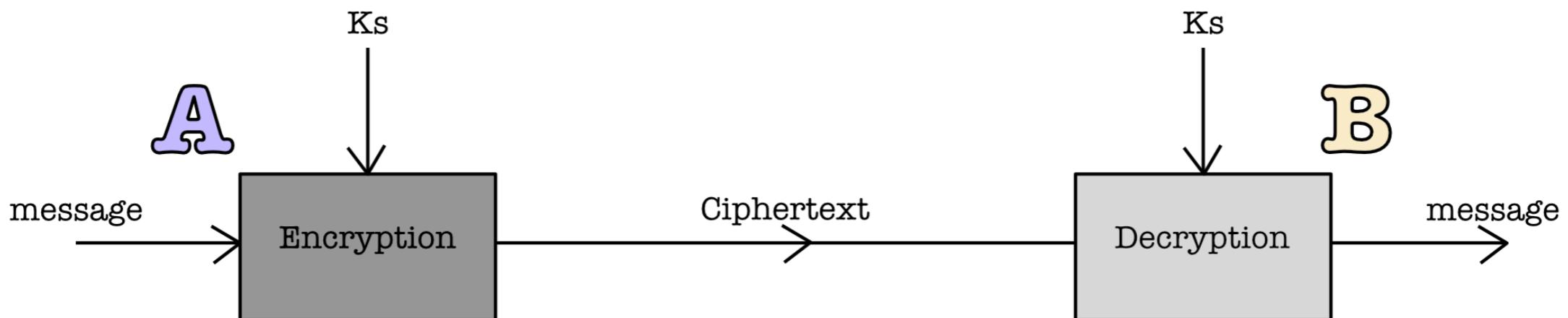
- An algorithm to perform symmetric key cryptography
- 56-bit symmetric key
- 64-bit plaintext input (padding may be needed)
- 16 rounds of encryption to produce 64-bit encrypted output
- It was US encryption standard in 1993
- Can be decrypted using brute force in less than a day
- 3-DES to make it *more* secure

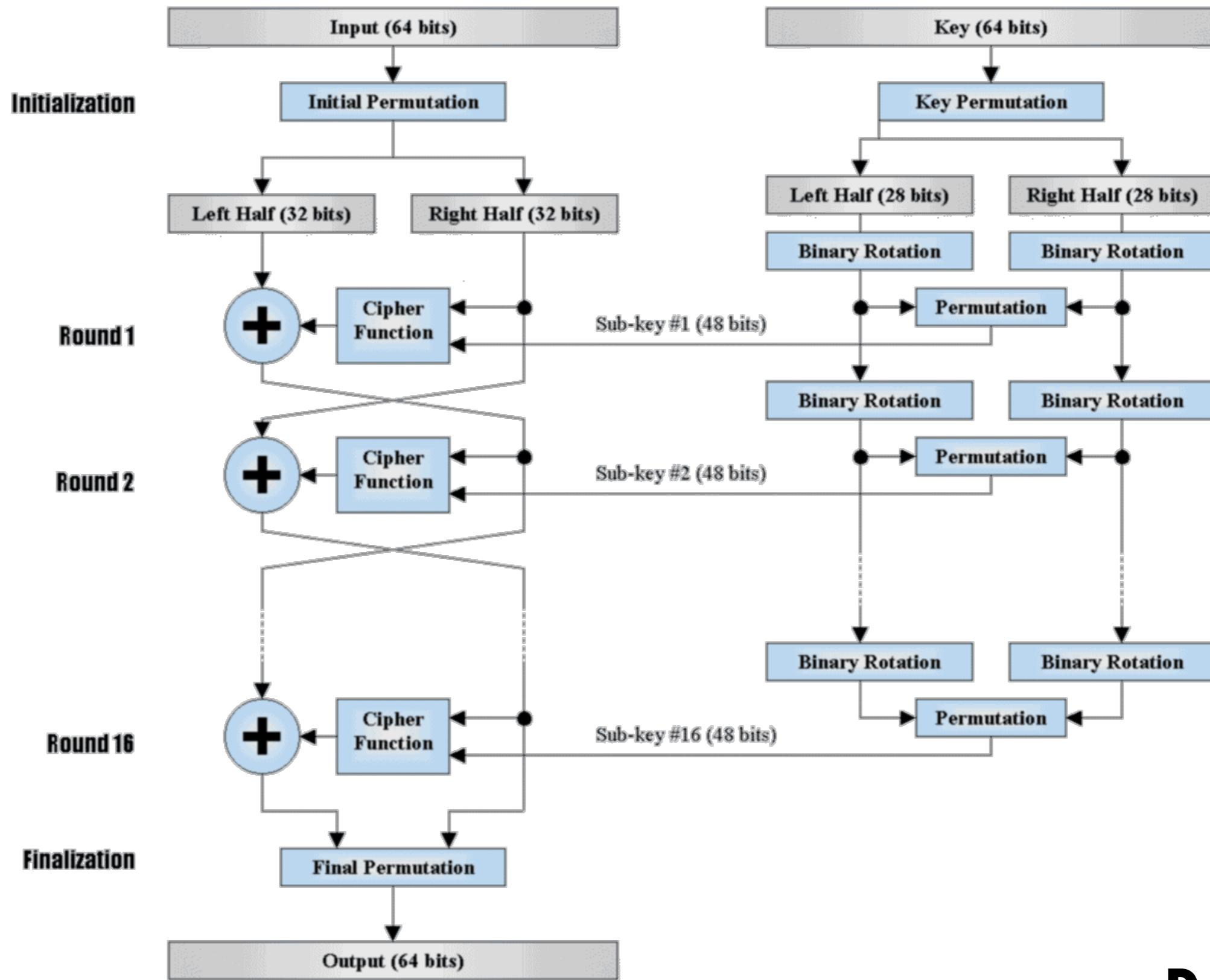
Advanced Encryption Standard (AES):

- A better algorithm to perform symmetric key cryptography
- 128, 192, or 256-bit symmetric key
- 128-bit plaintext input (padding may be needed)
- 10,12,14 rounds of encryption to produce 128-bit encrypted output
- Replaced DES in 2001,
- If 128-bit key is used, brute force decryption will take 1 billion *billion* years (yes, double billions right there), with a supercomputer

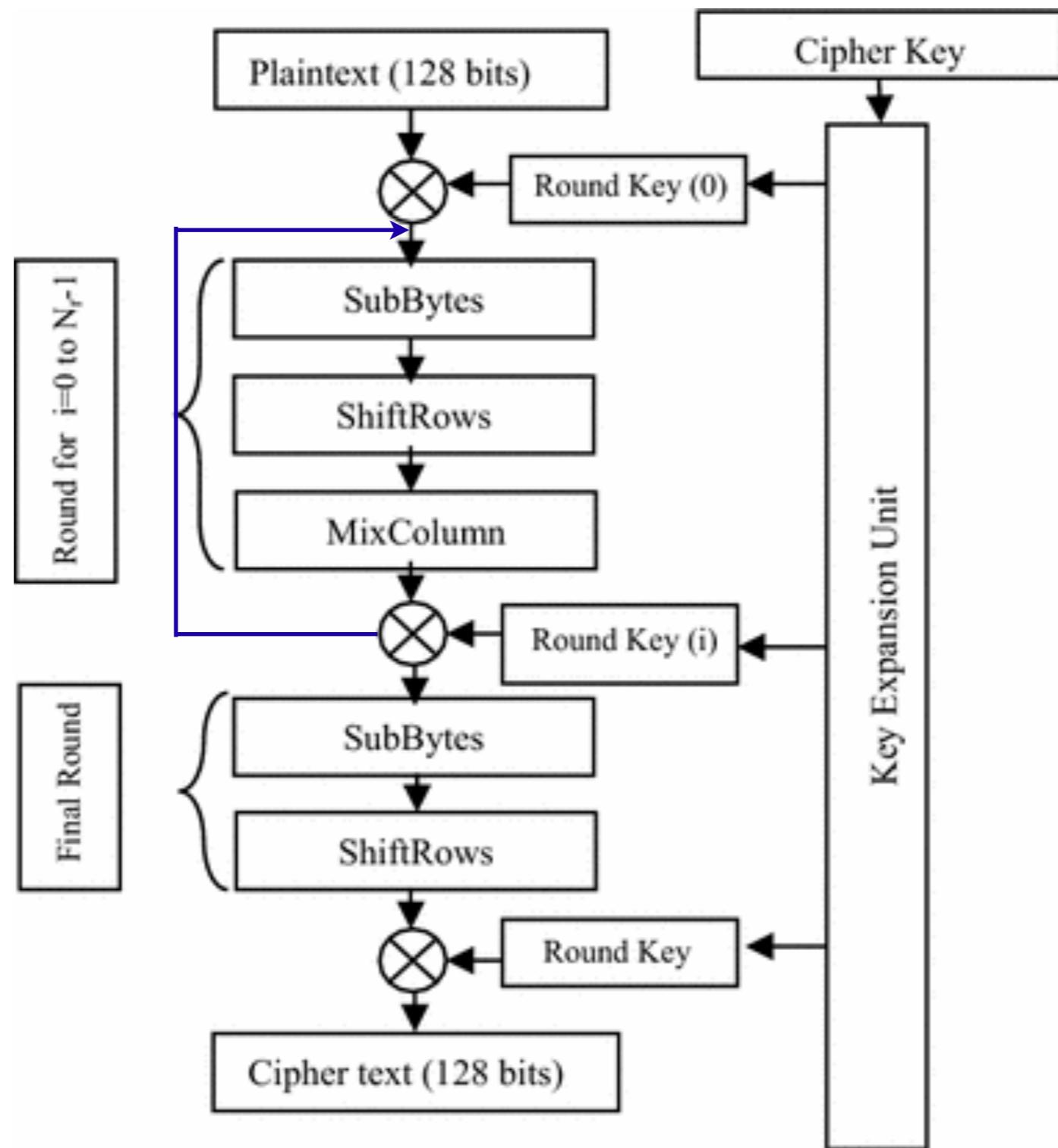
• S Y M M E T R I C K E Y C R Y P T O

encryption and decryption both uses the **same** key





D E S



A E S

P U B L I C K E Y C R Y P T O

a.k.a **asymmetric** key cryptography

Rivest-Shamir-Adleman (RSA) algorithm (1977):

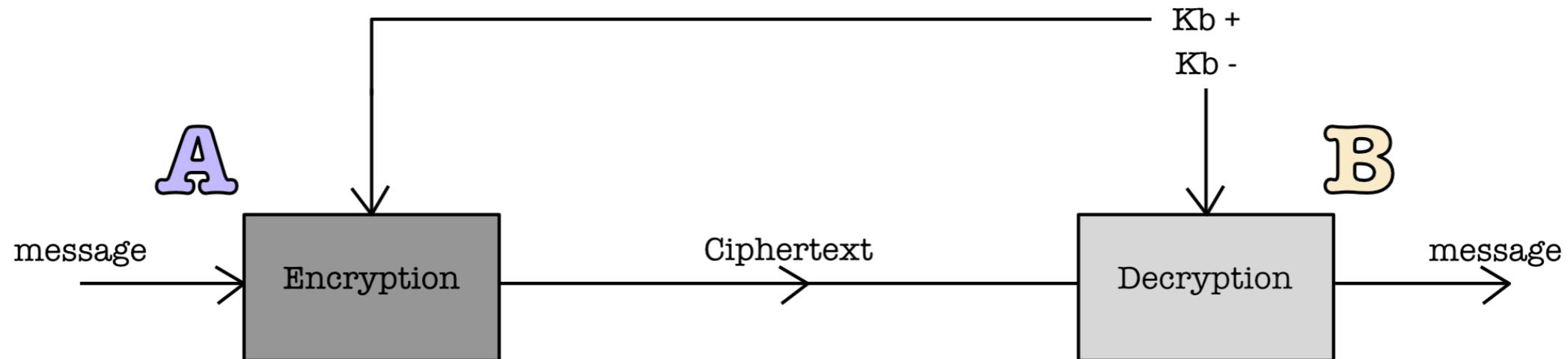
- Sender, receiver, do **not** share secret key
- **Public** encryption key **known to all**
- **Private** decryption key known only to receiver
- 1024 - 4096-bit asymmetric (public-private) key pair
- 1 round of encryption
- Input/output size: depends on key size

Two **important** requirements:

- Given **public** key, it is **impossible** to compute the private key
- Need a public-private key pair such that,
 - $\text{public}(m) = \text{encrypted message}$ ----- **encryption**
 - $\text{private}(\text{encrypted message}) = m$ ----- **decryption**

P U B L I C K E Y C R Y P T O

a.k.a **asymmetric** key cryptography



R S A M A T H

Before that, some quick **modular arithmetic**

$$[(a \bmod n) \pm (b \bmod n)] \bmod n = (a \pm b) \bmod n$$

$$[(a \bmod n) * (b \bmod n)] \bmod n = (a * b) \bmod n$$

$$[(a \bmod n)^d] \bmod n = a^d \bmod n$$

R S A M A T H

- Computer translates between bit patterns to the character you can see on screen, e.g using **UTF-8 encoding**
- You can represent a message by a integer number (simple binary to dec translation)
- Thus **encrypting a message is equivalent to encrypting a number**

...and some quick message-to-number translation:

| Dec | Symbol | Binary | Dec | Symbol | Binary |
|-----|--------|-----------|-----|--------|-----------|
| 65 | A | 0100 0001 | 83 | S | 0101 0011 |
| 66 | B | 0100 0010 | 84 | T | 0101 0100 |
| 67 | C | 0100 0011 | 85 | U | 0101 0101 |
| 68 | D | 0100 0100 | 86 | V | 0101 0110 |
| 69 | E | 0100 0101 | 87 | W | 0101 0111 |
| 70 | F | 0100 0110 | 88 | X | 0101 1000 |
| 71 | G | 0100 0111 | 89 | Y | 0101 1001 |
| 72 | H | 0100 1000 | 90 | Z | 0101 1010 |
| 73 | I | 0100 1001 | 91 | [| 0101 1011 |
| 74 | J | 0100 1010 | 92 | \ | 0101 1100 |
| 75 | K | 0100 1011 | 93 |] | 0101 1101 |
| 76 | L | 0100 1100 | 94 | ^ | 0101 1110 |
| 77 | M | 0100 1101 | 95 | - | 0101 1111 |
| 78 | N | 0100 1110 | 96 | ` | 0110 0000 |
| 79 | O | 0100 1111 | 97 | a | 0110 0001 |
| 80 | P | 0101 0000 | 98 | b | 0110 0010 |
| 81 | Q | 0101 0001 | 99 | c | 0110 0011 |
| 82 | R | 0101 0010 | 100 | d | 0110 0100 |

R S A M A T H

The RSA algorithm

1. Choose 2 large **prime** number : p & q (1024 bits each)

2. Compute:

$$n = pq, \ z = (p - 1)(q - 1)$$

3. Choose e, where $e < n$, and e, z are relatively prime to one another

4. Choose d, where $(ed - 1) \bmod z = 0$

5. Public key, is (n,e):

$$c = m^e \bmod n \longrightarrow$$

m = message
(unencrypted), where **m**
must be < n
c = encrypted message

6. Private key, is (n,d):

$$m = c^d \bmod n$$

7. An important property:

$$\text{if } k = m^d \bmod n,$$

$$k^e \bmod n = m = c^d \bmod n$$

$$x^y \bmod n = x^{(y \bmod z)} \bmod n \text{ for any } x, y$$

R S A M A T H

The RSA algorithm: example

(note this is an example, so we choose small numbers for ease of calculation.

1. Choose 2 large **prime** number : p=5 & q=7
2. Compute:

$$n = 5 * 7 = 35, z = (5 - 1)(7 - 1) = 24$$

3. Choose e = 5, where e<n, and e,z are relatively prime to one another
4. Choose d = 29, where $(ed - 1) \bmod z = 0$

5. Public key, is (n,e):

$$c = 12^5 \bmod 35 = 17$$

m = 00001100 (this is 12)
c = 00010001 (this is 17)

6. Private key, is (n,d):

$$m = 17^{29} \bmod 35 = 12$$

7. An important property:

$$\text{if } k = m^d \bmod n,$$

$$k^e \bmod n = m = c^d \bmod n$$

$$x^y \bmod n = x^{(y \bmod z)} \bmod n \text{ for any } x, y$$

R S A M A T H

The RSA algorithm: proof

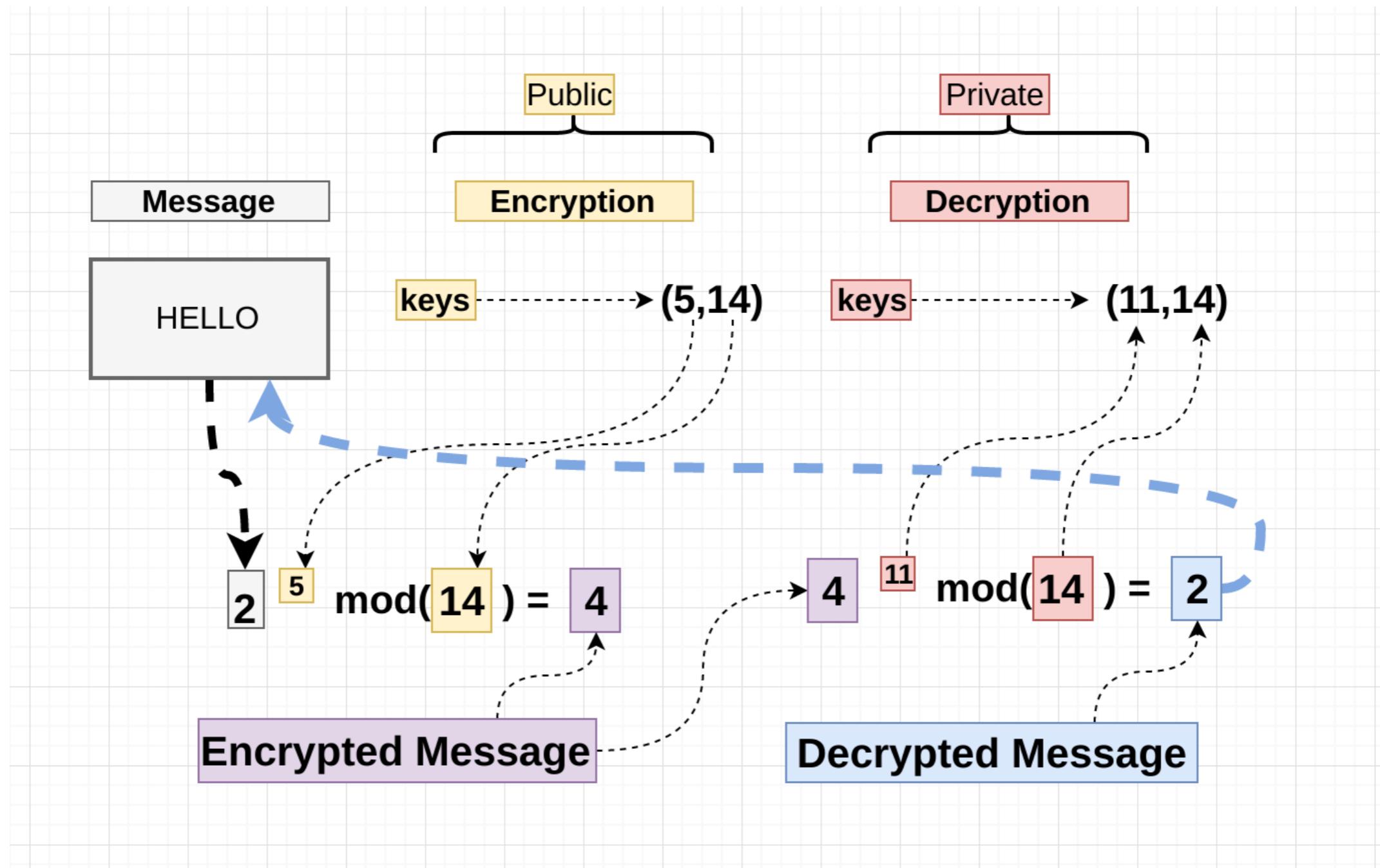
$$c = m^e \pmod{n}$$

$$\begin{aligned} c^d \pmod{n} &= (m^e \pmod{n})^d \pmod{n} \\ &= m^{ed} \pmod{n} \\ &= m^{(ed \pmod{z})} \pmod{n} \\ &= m^1 \pmod{n} \quad \text{----- Since } m < n \end{aligned}$$

$$x^y \pmod{n} = x^{(y \pmod{z})} \pmod{n} \text{ for any } x, y$$

R S A M A T H

The RSA algorithm: another example



• H O W S E C U R E I S R S A ?

Very, very **secure** (1024 bit key above),
unless someone comes up with scalable quantum computers.

- Public key (n, e) is known to everybody. Private key (n, d) is only known to receiver
- **Essentially, need to guess the value of d**
- e is known and d is related to e : $(ed-1) \bmod z = 0$
- To find z , one has to know **p** and **q**
- p and q is related to n : $n = p^*q$
- Hence, one has to be able to **perform prime factorization** of n to know p and q
- If n is sufficiently large, e.g: 1024 bits, it is hard to find the correct prime factors of n (exponential complexity).
- **To crack a simple 128-bit key, a supercomputer requires $\sim 10^{39}$ seconds.** The universe is younger than that.

- The prime factorization problem falls under **NP**
- Note: with *possible scalable quantum computing*, Shor's algorithm can solve prime factorisation in polynomial time.

• HOW PRACTICAL IS RSA?

Exponentiation in RSA is **computationally exhaustive**.

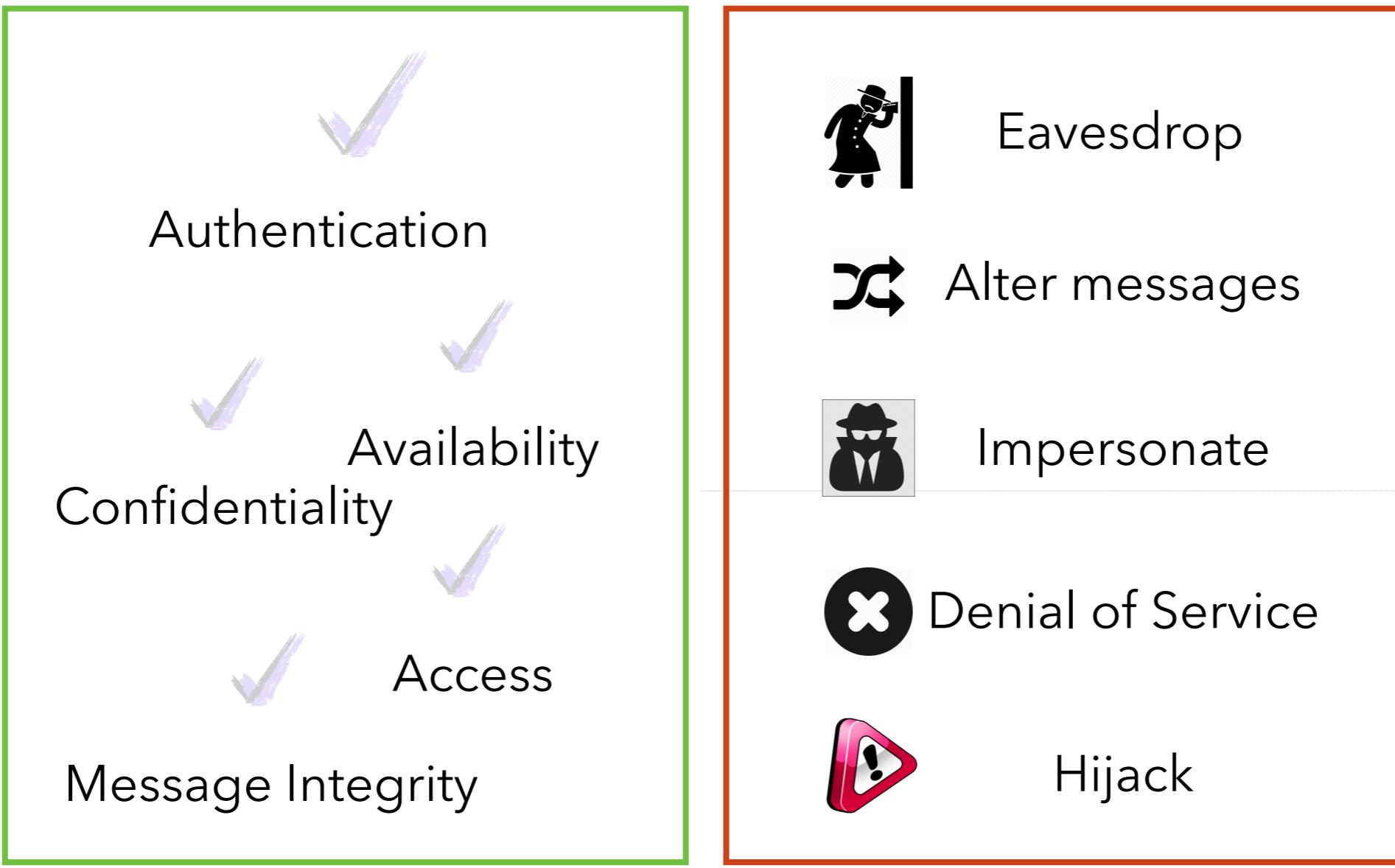
DES is 100x faster than RSA

- **USE SESSION KEY:**

1. Use public-key cryptography to establish a secure connection, meaning that we know for sure who the host at the end of the network line is
2. Generate **symmetric session key** and exchange between sender and receiver
3. **Use this symmetric session key** (instead of the public key) for lots of subsequent communications throughout the session

APPLICATION OF CRYPTOGRAPHY

What kind of security properties can it give? And how does it defend against the attacks?



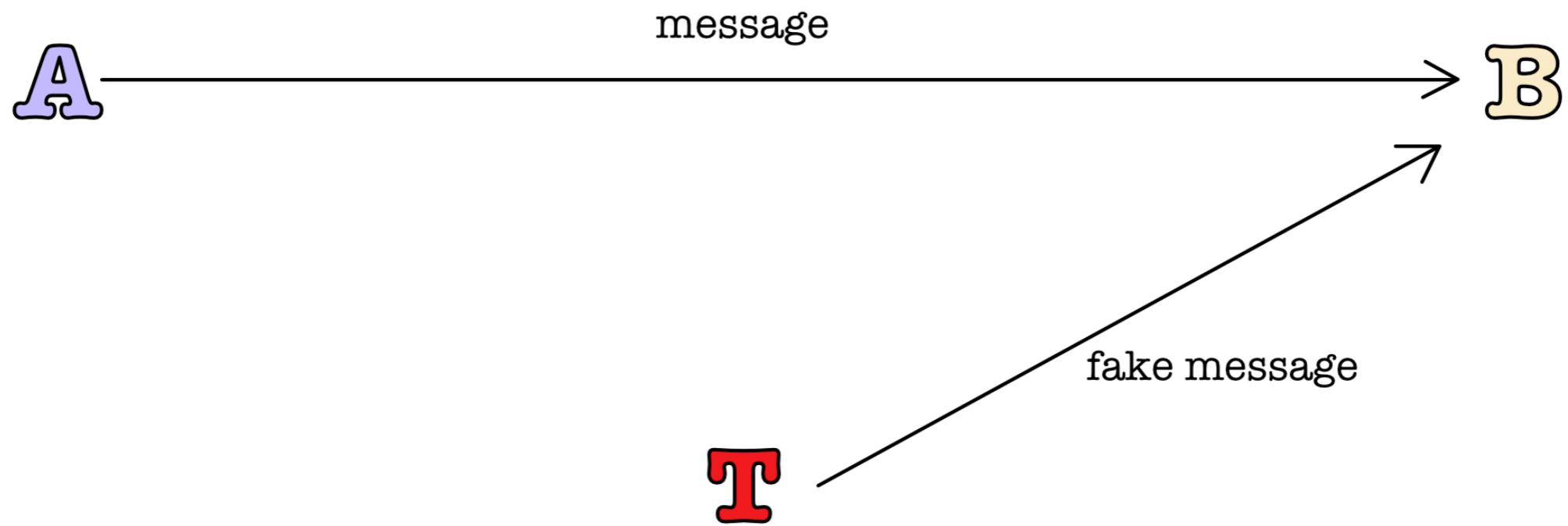
PROVIDES

PREVENT

1. PROVIDING AUTHENTICATION

Need: **prevent impersonation**

Lets consider this case without encryption first:

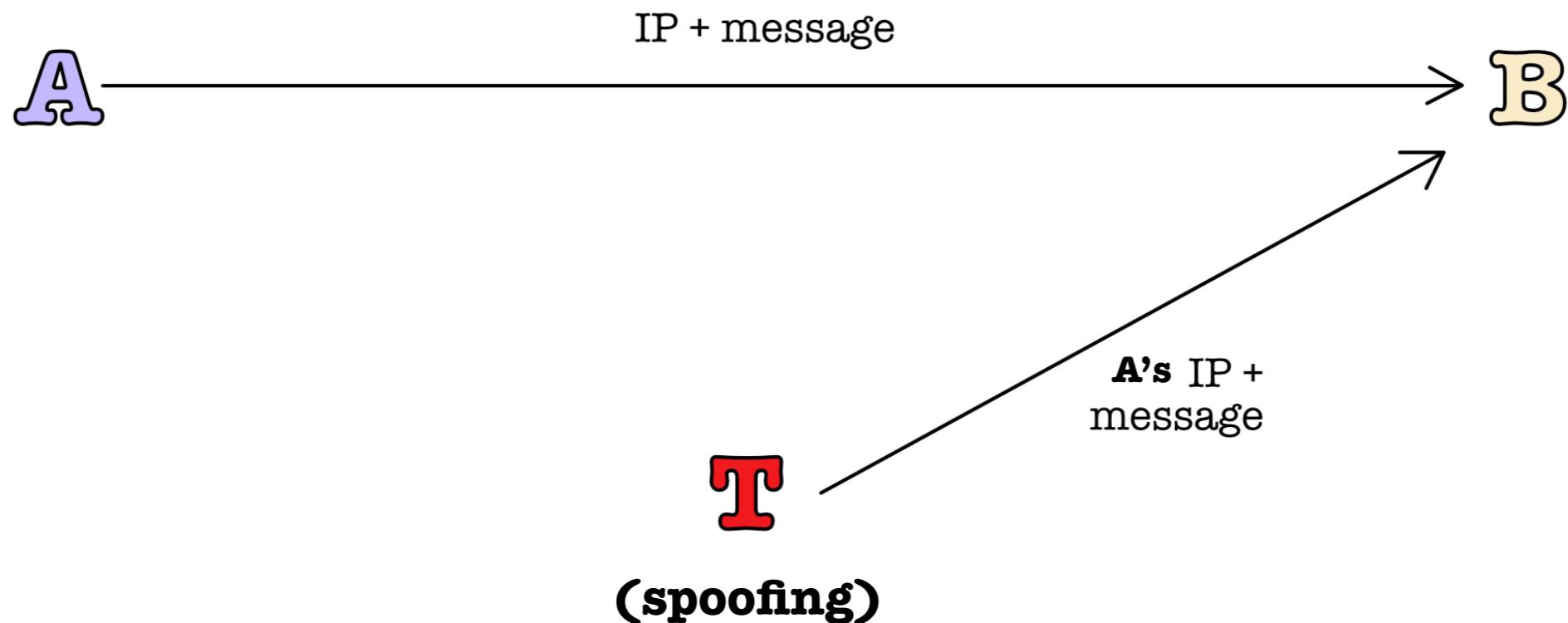


Without any encryption, B doesn't know who is the "real" A

1. PROVIDING AUTHENTICATION

Need: **prevent impersonation and spoofing**

Lets consider this case without encryption first:

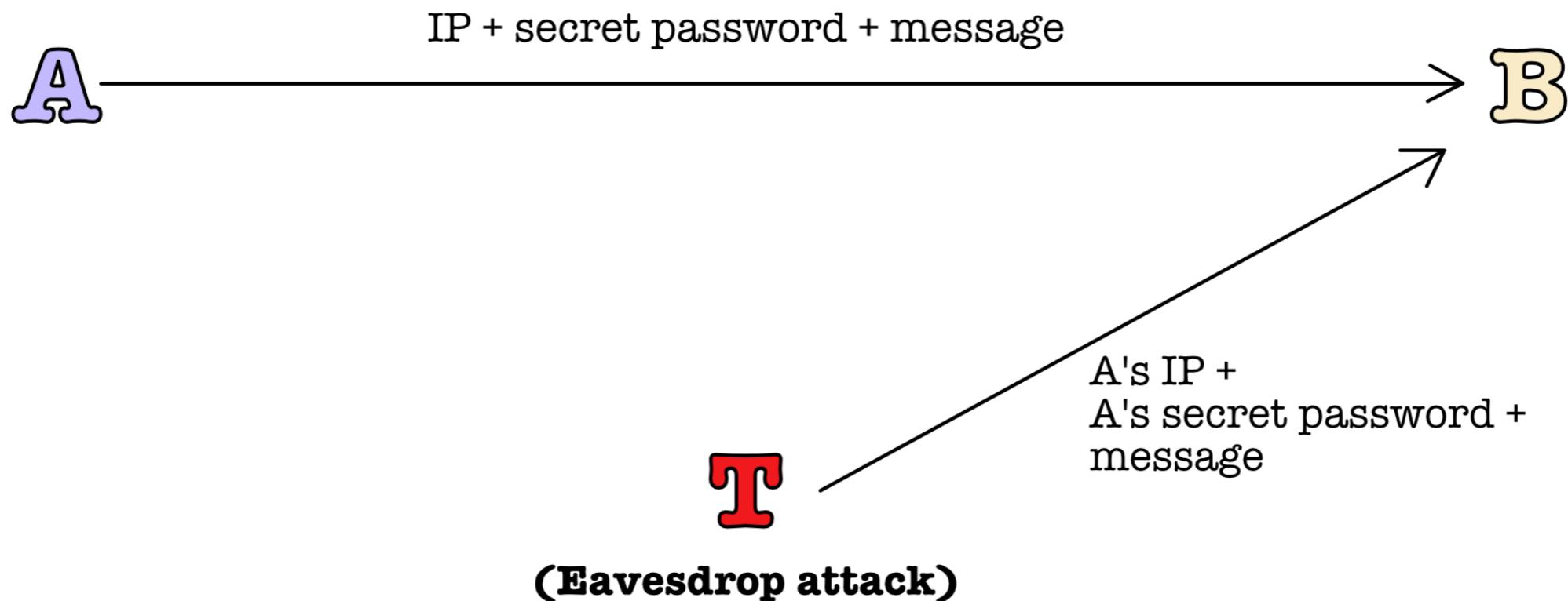


Without any encryption, B doesn't know who is the "real" A, even with IP address (IP is always changing depending on your network)

1. PROVIDING AUTHENTICATION

Need: **prevent impersonation, spoofing, and eavesdropping**

Lets consider this case without encryption first:

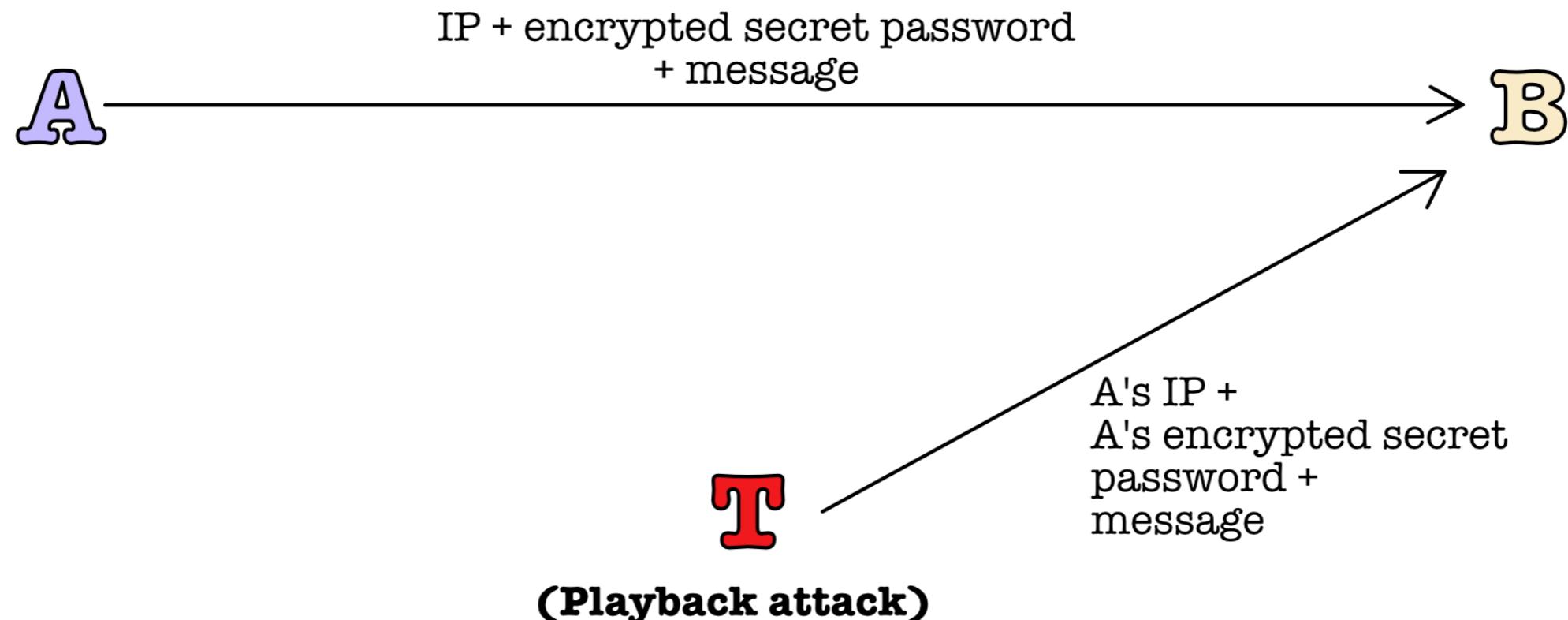


Without any encryption, B doesn't know who is the "real" A, even with IP address (IP is always changing depending on your network), and **nothing is secret in the internet**

1. PROVIDING AUTHENTICATION

Need: **prevent impersonation, spoofing, eavesdropping, and playback**

Some encryption is used here, but not enough



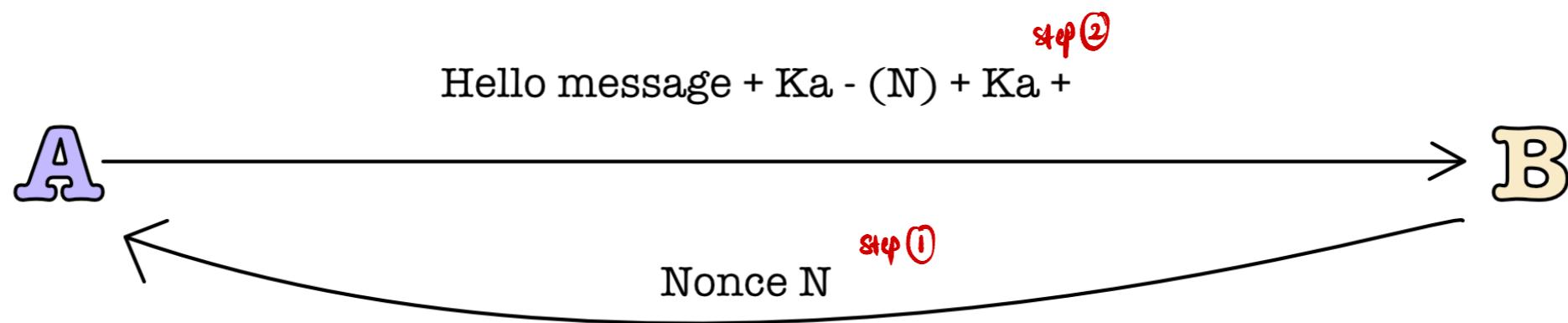
T can still just record and playback the "encrypted secret password". B cannot tell if its really A or T who is sending this.

1. PROVIDING AUTHENTICATION

Need: **prevent impersonation, spoofing, eavesdropping, and playback**

Attempt 1: Lets use encryption and nonce

B sends Nonce
A encrypts it using k_A^{-}



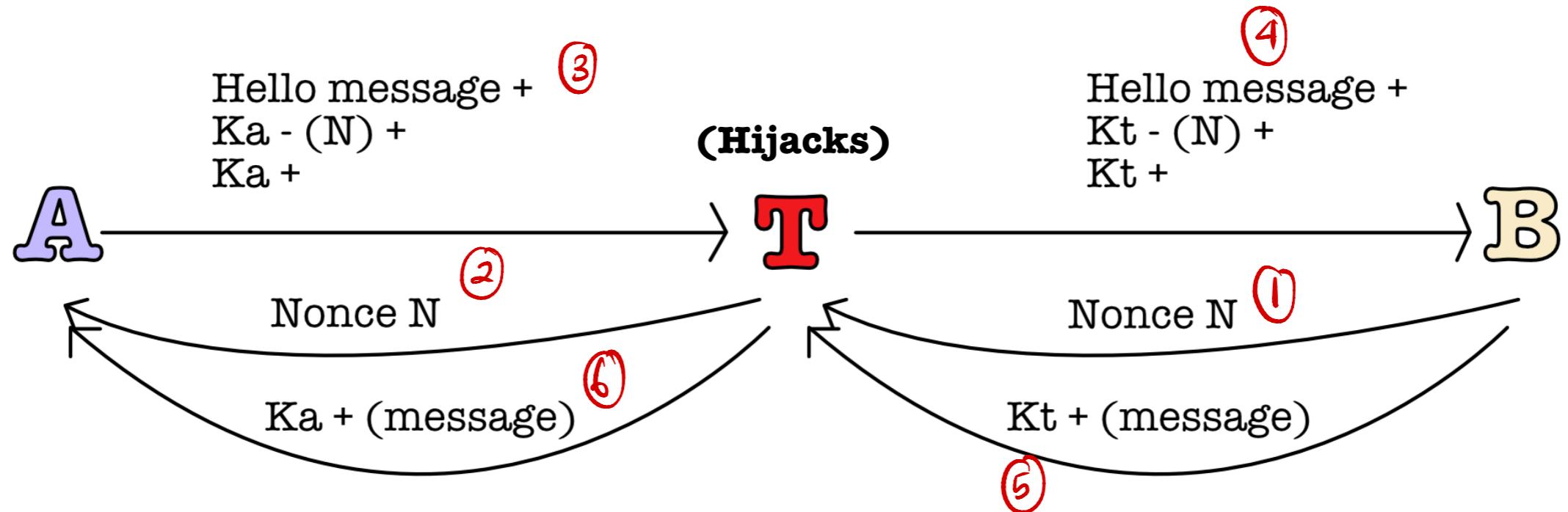
Encrypting message with private key is called **digital signature**

1. PROVIDING AUTHENTICATION

Need: **prevent impersonation, spoofing, eavesdropping, playback, and man in the middle attack**

Attempt 1: Lets use encryption and nonce

Failed by man-in-the-middle attack



1. PROVIDING AUTHENTICATION

Need: **prevent impersonation, spoofing, eavesdropping, playback, and man in the middle attack**

Attempt 1: Lets use encryption and nonce

Result: Failed by man-in-the-middle attack

Nonce is still required to prevent the playback attack

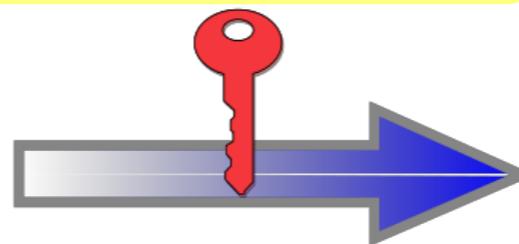
The loophole: B doesn't know whether A's public key he received indeed belongs to A

Solution: Besides Nonce, you also need an **external party (called Certificate Authority: CA)** to verify that A's public key indeed **belongs to A**

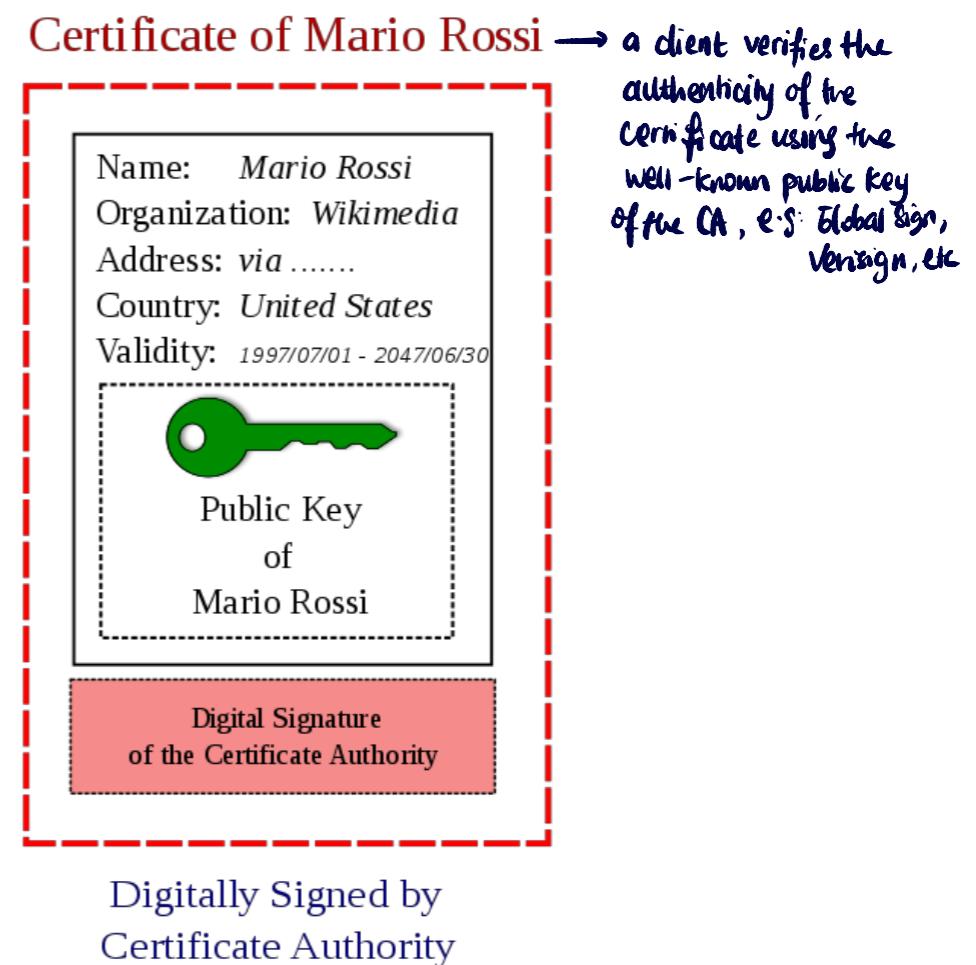
Lets say Mario sends you: Identity Information and Public Key of Mario Rossi



Certificate Authority verifies the identity of Mario Rossi and encrypts with its Private Key



Checking if “Mario” is indeed Mario



2. PROVIDING (AUTHENTICATION AND) MESSAGE INTEGRITY

Practical usage using **message digest**

It is expensive to encrypt long messages using private key (RSA)

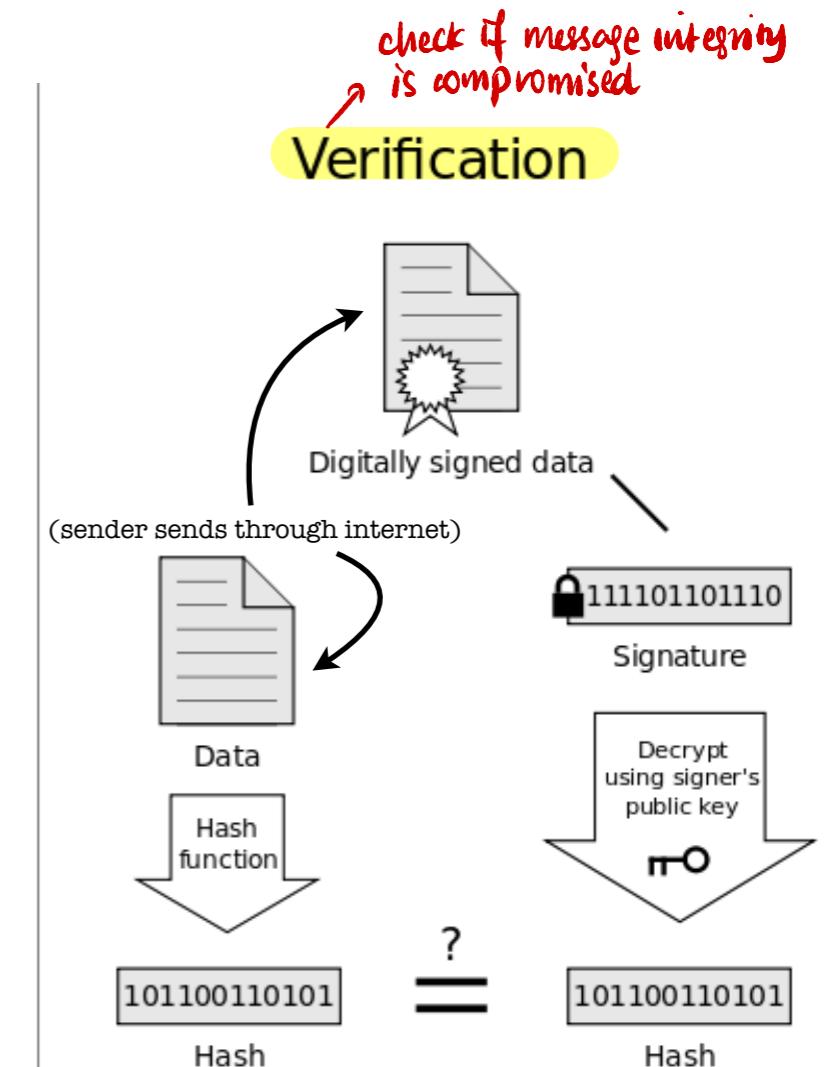
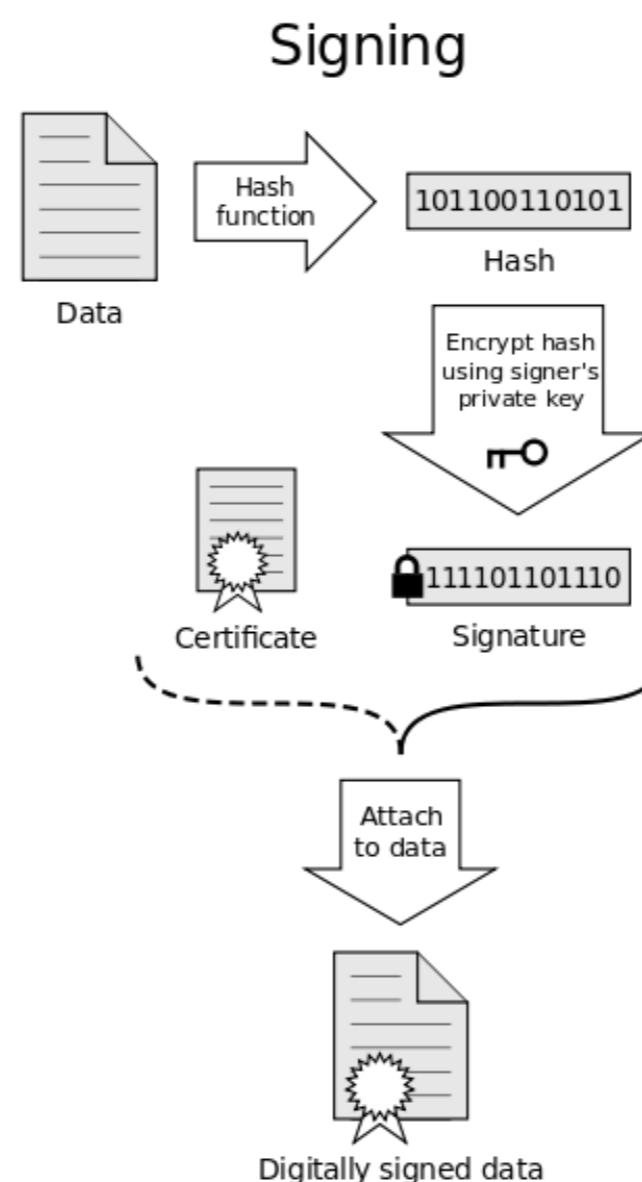
Solution: use message digest

Message digest:

Hash long (arbitrary sized) messages to produce a **fixed size**

message digest

Hash functions: MD5, SHA1



If the hashes are equal, the signature is valid.
and data integrity is confirmed

2 . P R O V I D I N G (A U T H E N T I C A T I O N A N D) M E S S A G E I N T E G R I T Y

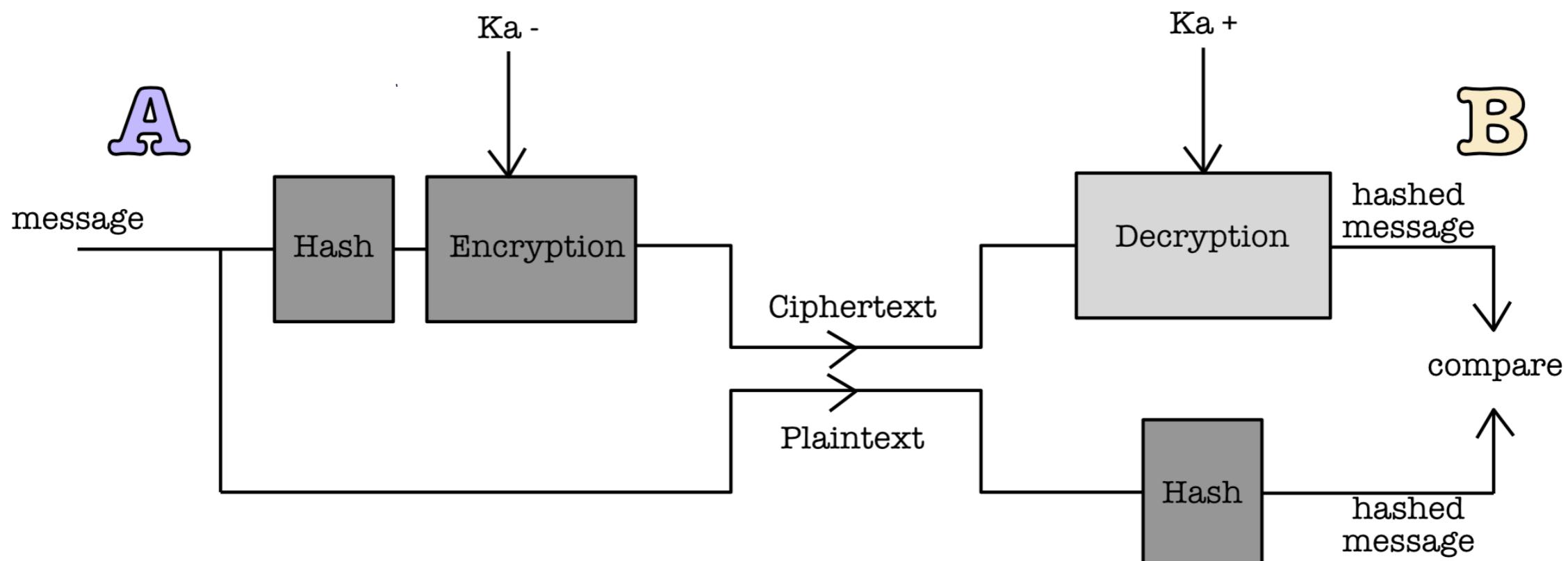
Prevent impersonation, spoofing, eavesdropping, playback, and altering of messages.

Signing (encrypting using private key) + Certificate guaranteed by CA give **nonrepudiation** that data is authorised by A

*digital signature
means this*

*↳ because legally, only
A is supposed to own K_A^-*

Certificate
signed by CA



3 . P R O V I D I N G (A U T H E N T I C A T I O N A N D) C O N F I D E N T I A L I T Y

Eg: **secure e-mail, ssh, https**

Prevents prevent impersonation, spoofing, eavesdropping, playback and man-in-the-middle (hijacking)

