Cryptography

December 2, 2014

- Principles of security

- Encryption algorithms

- Key exchanges

- Hashing algorithms

- Authentication algorithms

Security concepts
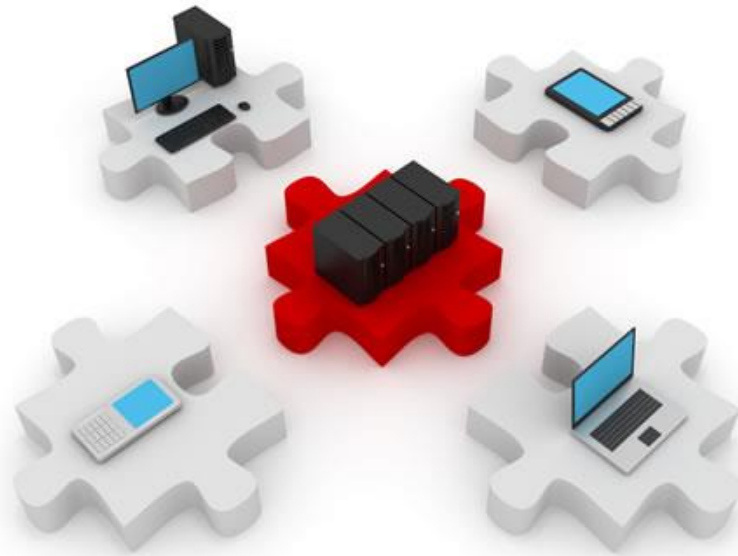
Confidentiality | Keeping data secret

Integrity | Testing whether data has been tampered with

Authentication | Checking whether an entity is who it claims to be

CONFIDENTIALITY

Confidentiality Overview

- Keeping data secret from eavesdroppers
- Data must be retrievable

- Broken when the attacker becomes able to decrypt encrypted content

Caesar	DES
Monoalphabetic Substitution	3DES
Vigenere	AES
OTP	RSA

- Plaintext
  - The text before it is encrypted; the input of the encryption algorithm

- Ciphertext
  - The text after it was encrypted; the output of the encryption algorithm

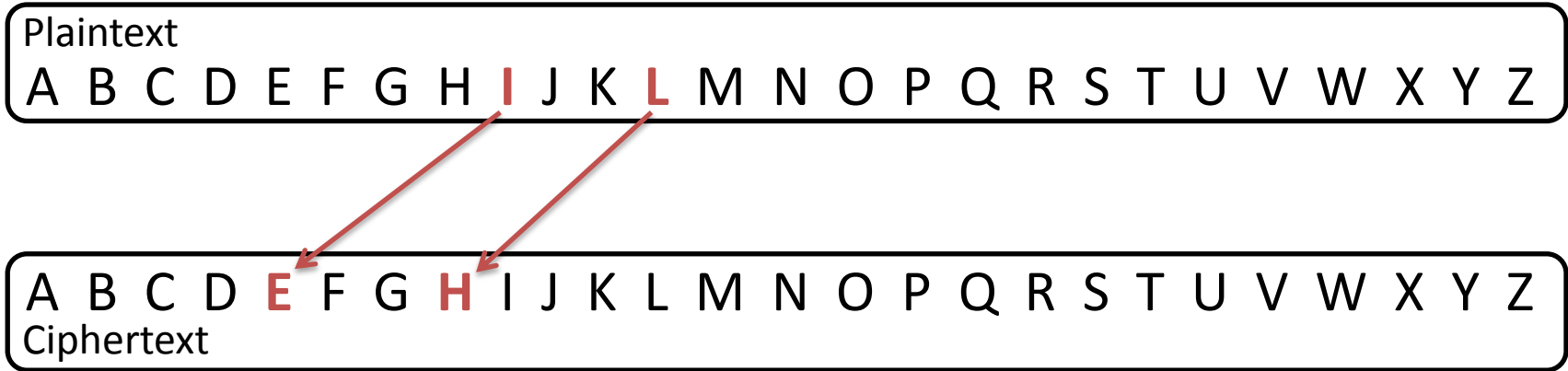| Plaintext *Hello, world!*	→	Algorithm	→	Ciphertext *Uryyb, jbeyq!*

- Key
  - A second input, usually secret, used to customize the encryption algorithm

- Key space
  - The set of data from which keys may be selected
  - A larger set of keys leads to an increase in the duration of brute force attempts

Caesar cipher

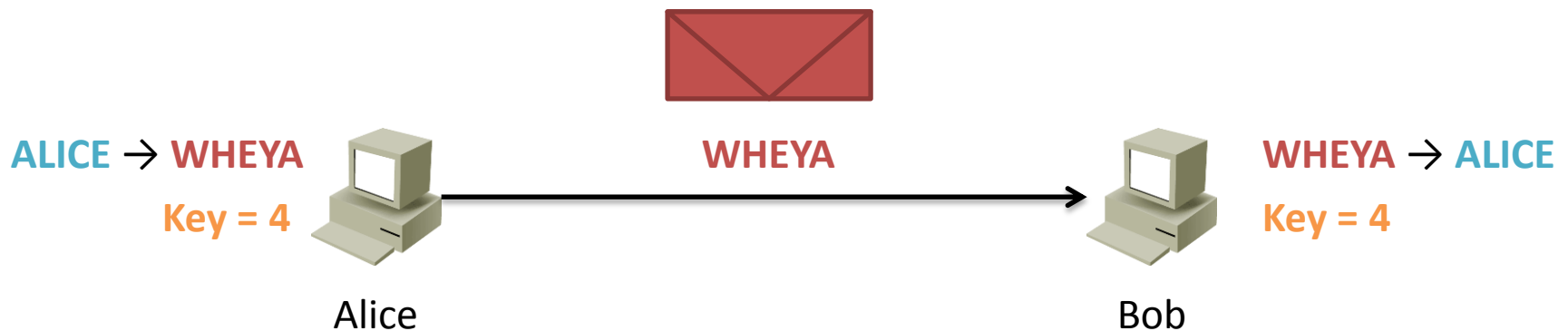| Caesar	Class	Classical Monoalphabetic substitution Symmetrical
	Date invented	1st century BC
	Prerequisites	Both parties must know the secret key

- One of the earliest known uses of encryption
- Used by Julius Caesar during military campaigns

Caesar cipher – Algorithm

Plaintext

A B C D E F G H **I** J K **L** M N O P Q R S T U V W X Y Z

A B C D **E** F G **H** I J K L M N O P Q R S T U V W X Y Z

Ciphertext

Monoalphabetic substitution table:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V

ALICE → WHEYA   **WHEYA**   **WHEYA → ALICE**

Key = 4   **Key = 4**

Alice   Bob

Caesar cipher – Conclusion

The key

Key space: $26 \cong 2^5$

Key format: Number or letter

Weaknesses:

Brute force attacks (low key space)

Frequency analysis

Known plaintext attacks

Keys must be preshared

Verdict: **Do not use**

Substitution cipher

Substitution cipher	Class	Classical Monoalphabetic substitution Symmetric
	Date invented	Specific types in use during 1st century BC
	Prerequisites	Both parties must know the secret key

- One letter (or byte) is substituted for another letter (or byte), according to a permutation

- Caesar cipher is a specific type of substitution cipher

- Many ancient ciphers were variants of the simple substitution cipher
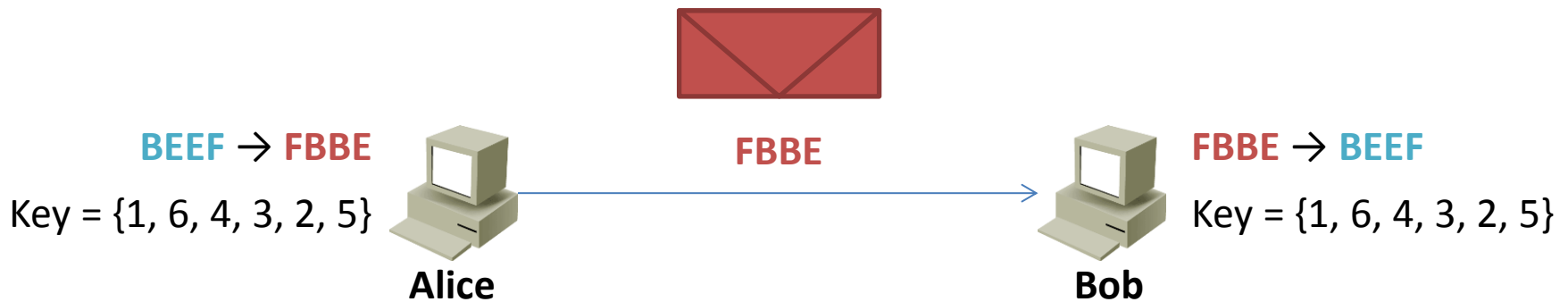
Substitution cipher – Algorithm

- The key is a permutation

- Example #1: Key = {2, 3, 4, 5, 6, 1}

A	B	C	D	E	F
B	C	D	E	F	A

- Example #2: Key = {1, 6, 4, 3, 2, 5}

A	B	C	D	E	F
A	F	D	C	B	E

BEEF → FBBE

Key = {1, 6, 4, 3, 2, 5}

Alice

FBBE

FBBE → BEEF

Key = {1, 6, 4, 3, 2, 5}

Bob

Substitution cipher – Conclusion

The key

Key space: $P_{26} \cong 2^{88}$

Key format: Permutation

Weaknesses:

Frequency analysis

Known plaintext attacks

Keys must be preshared

Verdict: **Do not use**

Vigenere	Class	Classical Polyalphabetic substitution Symmetric
	Date invented	16th century
	Prerequisites	Both parties must know the secret key

- Composed of 26 inverted Caesar ciphers
- Difficulty in breaking it at the time gave it the nickname *The unbreakable cipher*

Vigenere cipher – Algorithm

Key:
SCR

Plaintext:
Hello world

Ciphertext:
???

HELLOWORLD
SCRSCRSCRS
ZECDQNG...

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	E	F	G	H	I
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	E	F	G	H	I	J
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	E	F	G	H	I	J	K
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Vigenere cipher – Conclusion

The key

       Key space: Infinite

       Key format: Letter sequence

Weaknesses:

       Brute force (due to bad key choices)

       Frequency analysis

       Known plaintext attacks

       Keys must be preshared

Verdict: **Do not use**

OTP	Class	Classical Symmetric
	Date invented	1882/1917
	Prerequisites	Both parties must know the secret key

- Shannon proved that the OTP leaks no information about the plaintext message

- The key must be as long as the message

- If the key is used more than once , OTP security is broken

One time pad – Algorithm

$\oplus$	0	1
0	0	1
1	1	0

Modular addition (XOR, $\oplus$)

Hello world

000100011...

111010011...

OTP

$\oplus$

111110000...

$\oplus$

000100011...

Hello world

111010011...

OTP

Alice

Bob

One time pad – Conclusion
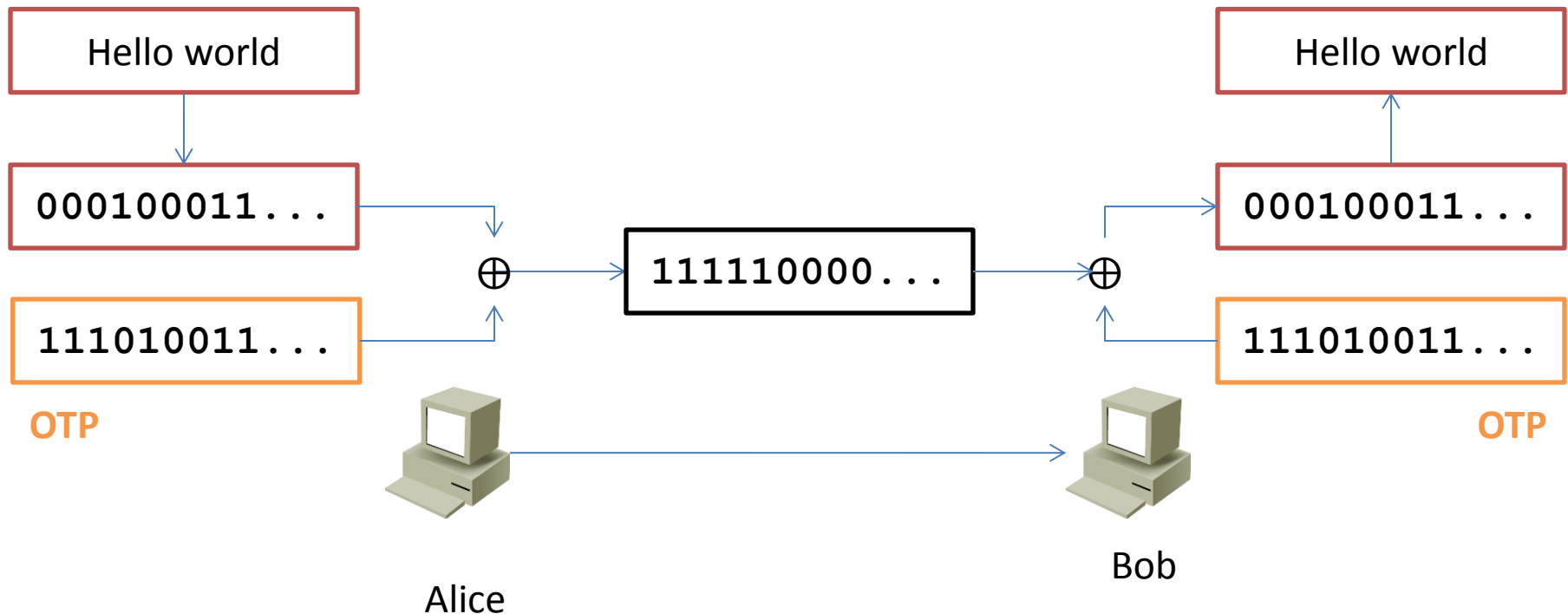
The key

Key space: 2^{length}

Key format: Bit sequence

Weaknesses:

Broken by chosen plaintext attacks

Keys must be preshared

Keys are as long as the message

Keys must only be used once

Verdict: **Use with care**

DES	Class	Modern Symmetric
	Date published	1977
	Prerequisites	Both parties must know the secret key

- Data Encryption Standard
- The first US federal standard for encryption algorithms
- Extensively studied since the 1970s
- Advances in computing power rendered it obsolete

DES – Algorithm

(Naive) DES Encryption

64 bit key

Plaintext Message

Discard excess bits

Block cipher
mode of operation

64 bit block	64 bit block

56 bit key

DES (Feistel Network)	DES (Feistel Network)

64 bit block	64 bit block

Ciphertext Message

DES – Conclusion
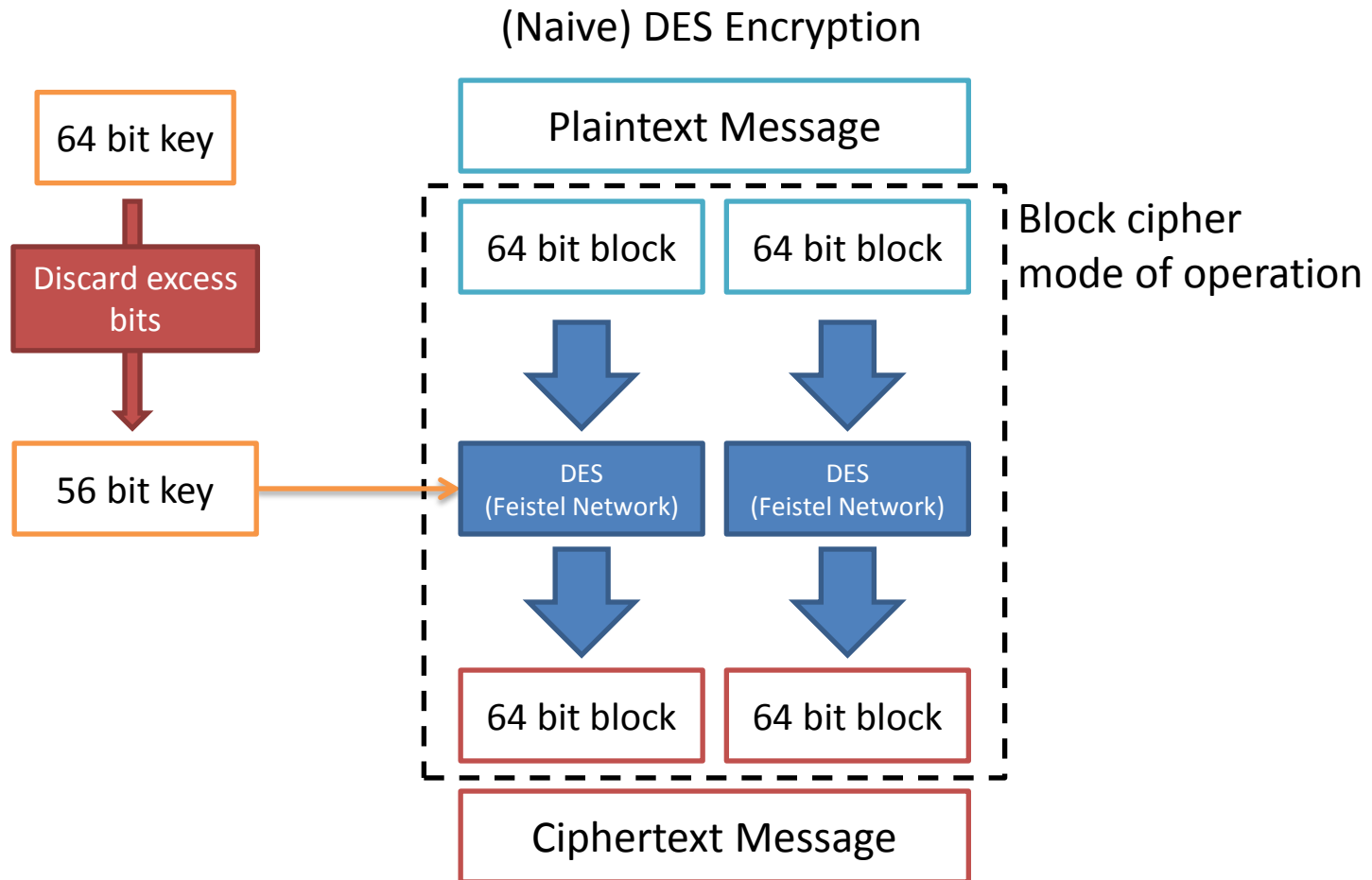
The key

      Key space: 2^{56} (approx.)

      Key format: Bit sequence

Weaknesses:

      Brute force feasible with current processors

      Keys must be preshared

Verdict: **Do not use**

3DES	Class	Modern Symmetric
	Date published	1998
	Prerequisites	Both parties must know the secret key

- Block algorithm, based on three iterations of DES
- Multiple keying options
  - Option 1: all keys are independent
  - Option 2: $K_1 = K_3$; K_1, K_2 independent
  - Option 3: $K_1 = K_2 = K_3$

3DES – Algorithm (keying option 1)

3DES – Conclusion
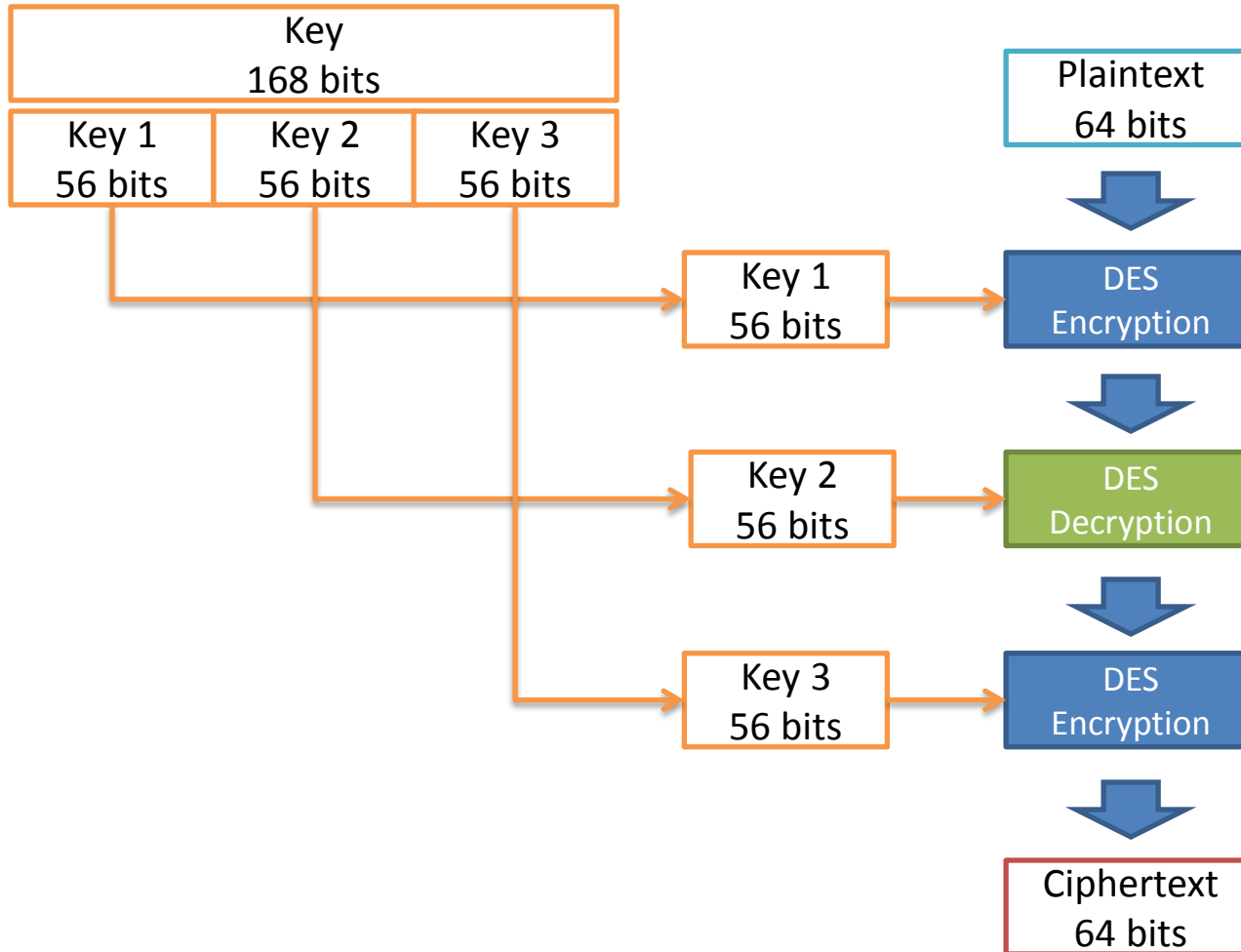
The key

Key space: 2^{168}

Best known attack: 2^{112}

Key format: Bit sequence

Weaknesses:

Keys must be preshared

Slower than other safe options

Verdict: **Safe to use**

AES	Class	Modern Symmetric
	Date published	1998
	Prerequisites	Both parties must know the secret key

- AES is the 128 bit block version of the Rijndael Cipher
- Very fast
- Hardware support
- AES-128, AES-192 and AES-256 refer to key sizes, and not block sizes

```
┌──────────┐          ┌──────────────┐
│ 128 bit  │          │  Plaintext   │
│   key    │          │  128 bits    │
└────┬─────┘          └──────┬───────┘
     │                       ▼
     │            ┌──────────────────────┐         SubBytes
     └──────────▶ │         AES          │
                  │   Substitution and   │         ShiftRows
                  │  permutation network │
                  └──────────┬───────────┘         MixColumns
                             ▼
                  ┌──────────────────────┐         AddRoundKey
                  │      Ciphertext      │
                  │       128 bits       │
                  └──────────────────────┘
```

AES – Conclusion
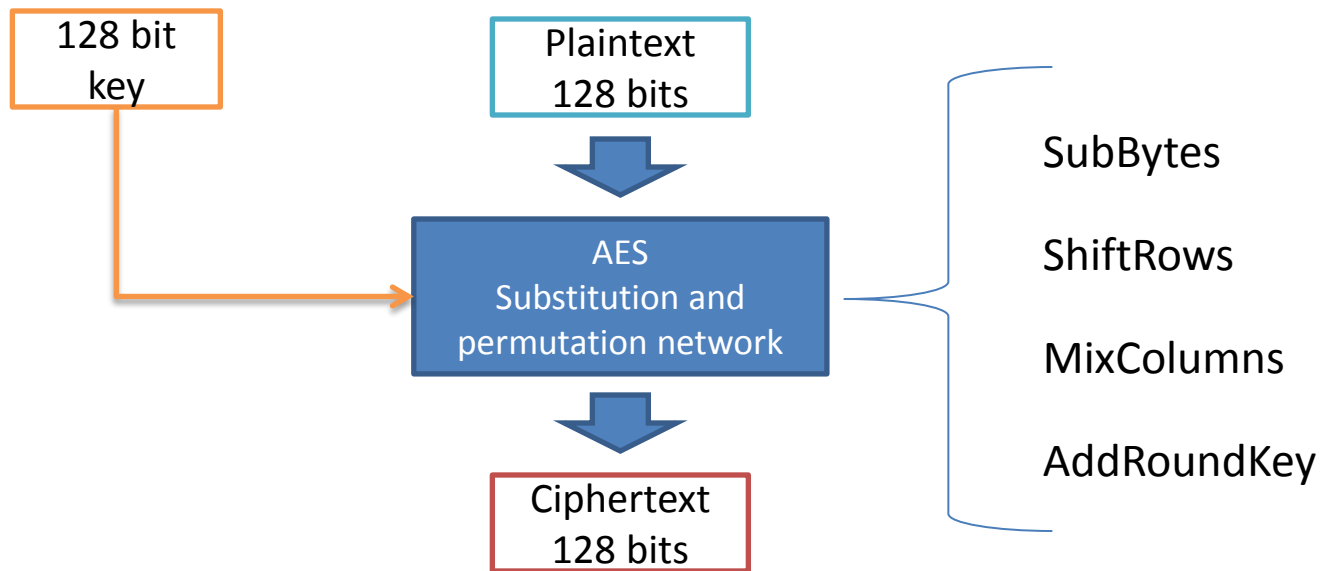
The key

Key space: 2^{128}, 2^{192}, 2^{256}

Best known attacks: $2^{126.1}$, $2^{189.7}$, $2^{254.4}$

Key format: Bit sequence

Weaknesses:

Keys must be preshared

Verdict: **Safe to use**

RSA	Class	Modern Asymmetric
	Date published	1977
	Prerequisites	Receiving party must know the **public** key

- The algorithm uses a key pair:
  - The public key (PubKey, or e); this is free to share
  - The private key (PrivKey, or d); this must be kept secret by the owner

- The important property is that $(x^e)^d = (x^d)^e = x$ (inside an algebraic structure with certain properties)

RSA Encryption – Algorithm

Plaintext

Convert to number(s)

Plaintext x

Public Key e

Compute $c = x^e$

Ciphertext c
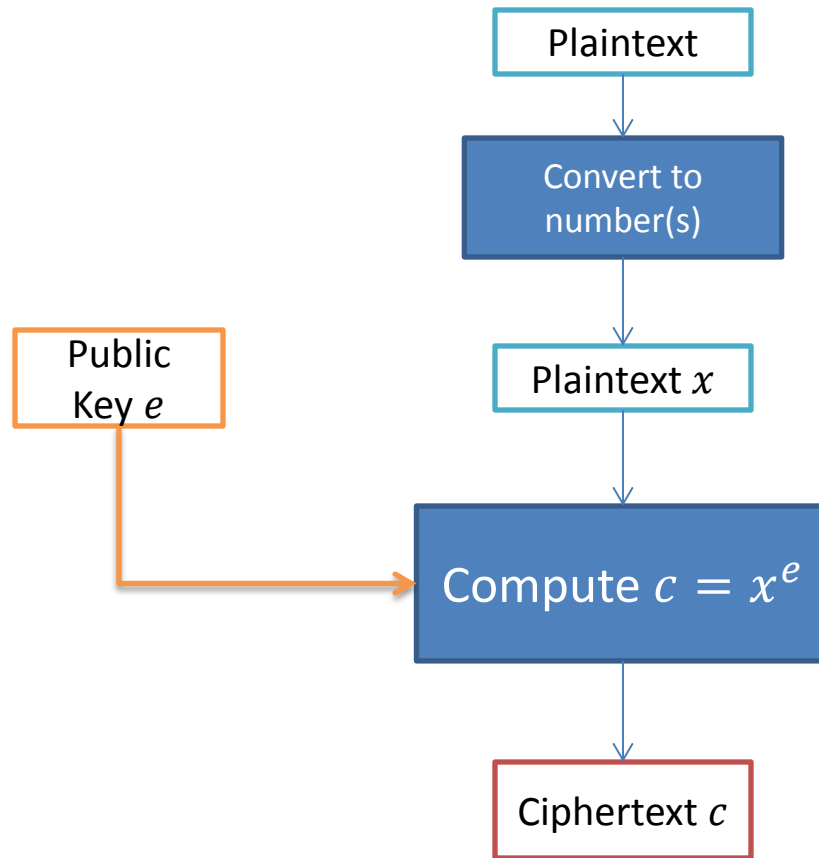
Why isn't the private key used for encryption?

The key

Key **size**: 2^{1024} to 2^{4096}, or larger

Best known brute forced key: 2^{768}

Key format: Large numbers, key pair

Weaknesses:

Very slow

Verdict: **Use sparingly**

Confidentiality – Conclusion

- Problems left to solve:
  - Key distribution
  - Message integrity

- Possible attacks:
  - Brute force
  - Cryptananalysis
    - Frequency analysis
    - Known plaintext/ciphertext cryptanalysis attacks
    - Chosen plaintext/ciphertext cryptanalysis attacks
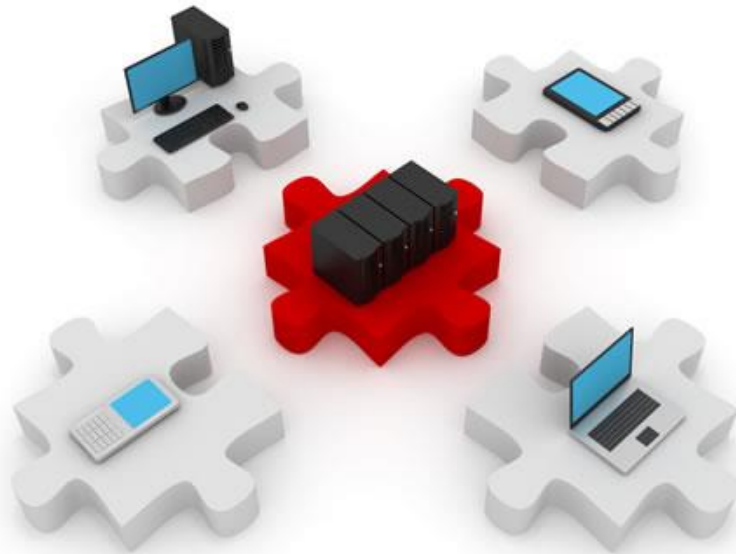
SECURE KEY EXCHANGES

Diffie Hellman

DH	Class	Key exchange algorithm
	Date published	1976
	Prerequisites	Authentication

- The problem:
    - Internet traffic requires encryption
    - Asymmetric encryption algorithms may share public keys freely, but they are too slow during encryption/decryption
    - Symmetric encryption algorithms require preshared keys

Diffie Hellman – Algorithm

Eve

Alice

Bob

g
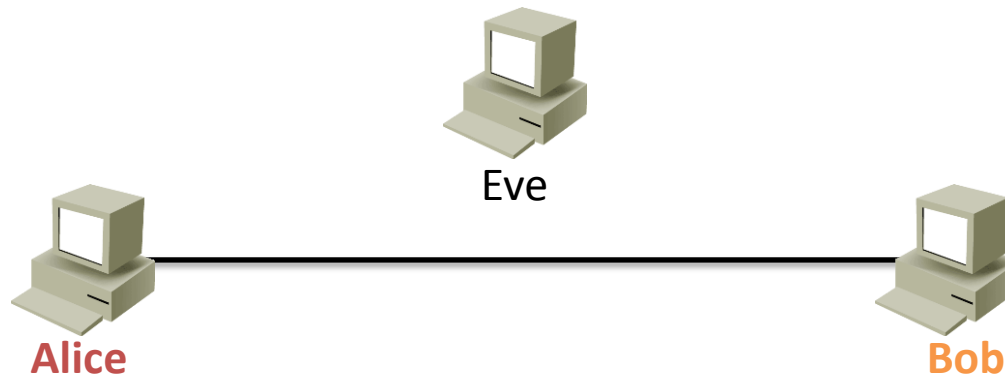
Generate a

Generate b

g^a

g^b

Compute
$k = \left(g^b\right)^a$

Compute
$\mathrm{k} = \left(g^a\right)^b$

Eve
Know $g, g^a, g^b, k = ???$

$$k = (g^a)^e = (g^e)^a$$

Eve

$$k' = \left(g^b\right)^{e'} = (g^{e'})^b$$
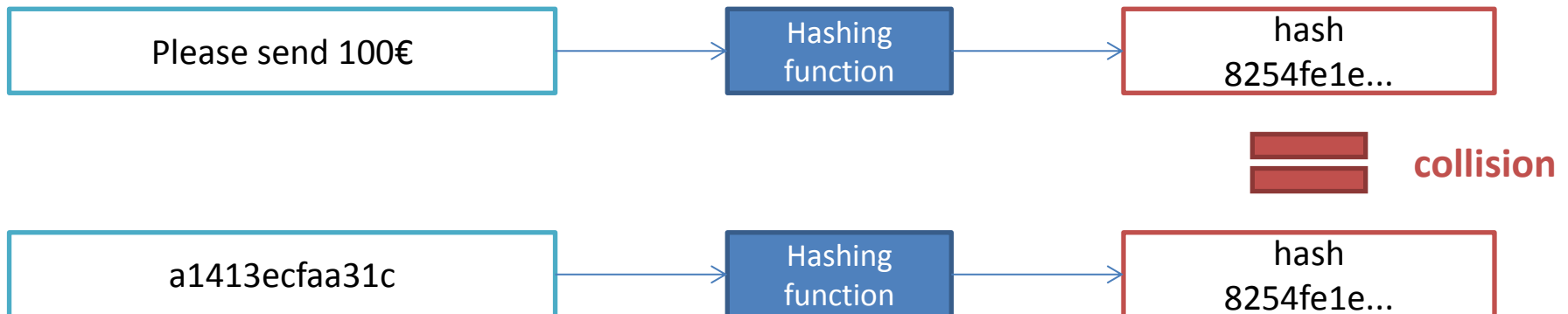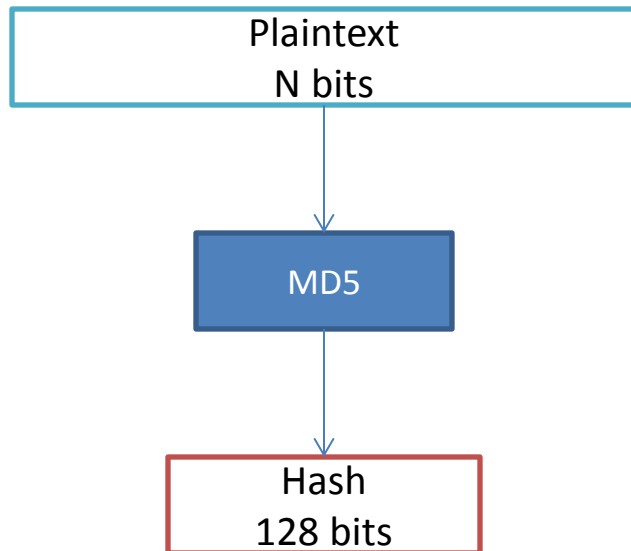
Alice

Bob
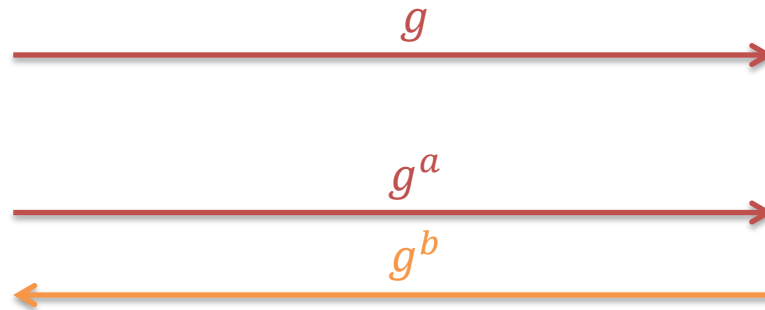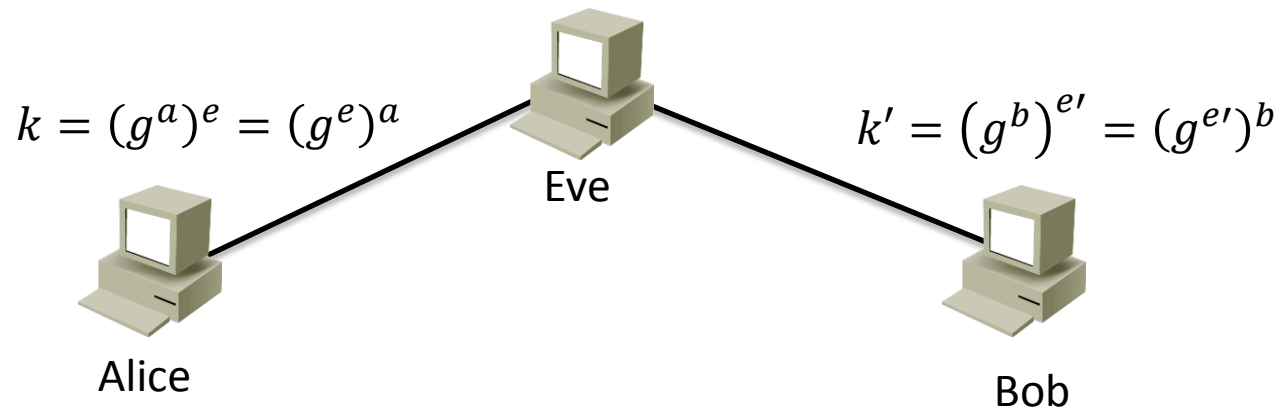
Weakness:

MITM attacks

INTEGRITY

- Integrity algorithms detect whether a message (or file) has been tampered with

- Hash functions output a fixed length summary of the message

- Good hash functions output very different results when small changes are performed on the input message

| Please send 100€ | → | Hashing function | → | hash 8254fe1e... |

≠

| Please send 10000€ | → | Hashing function | → | hash ed06891e... |
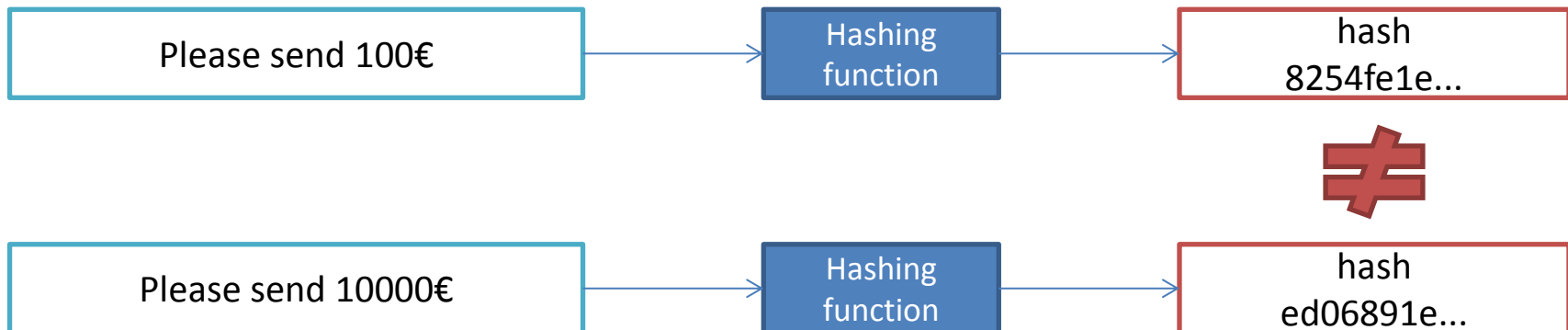
- A hash function is not invertible
  - multiple messages may yield the same hash
  - a hash is considered broken when two such messages are discovered
    - this is called a **hash collision**, and it means the hash has been broken

Please send 100€	→	Hashing function	→	hash 8254fe1e…

collision

a1413ecfaa31c	→	Hashing function	→	hash 8254fe1e…

Message Digest 5

MD5	Class	Hashing algorithm
	Date published	1992
	Hash length	128 bit

Plaintext
N bits

MD5

Hash
128 bits

- MD5 is not collision resistant
- Collisions for file checksums have already been generated

Verdict: **Strongly discouraged**

SHA-1/2/3

SHA	Class	Hashing algorithm
	Date published	1995 (SHA-1), 2001 (SHA-2), 2012 (SHA-3)
	Hash length	128 bit

```
Plaintext
N bits
   │
   ▼
 SHA-2
   │
   ▼
Hash
224/256/384/512/bits
```
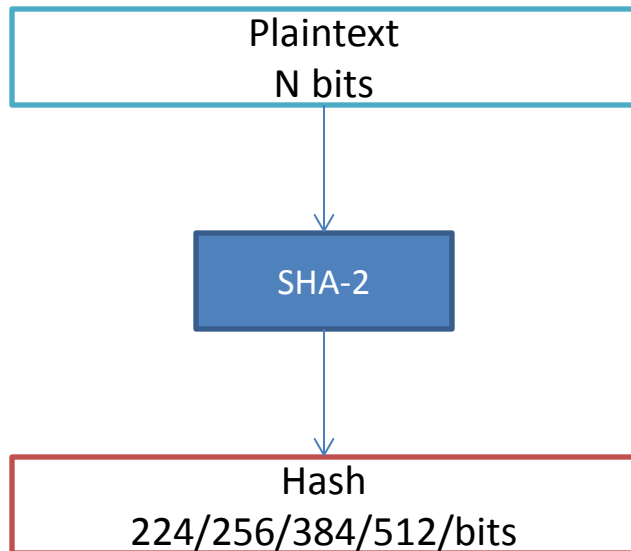
- SHA-1 is now considered broken, but still used by many implementations

- SHA-2 is a federal standard since 2001

- SHA-3 uses a different algorithm to SHA-1 and SHA-2

Verdict for SHA-1: **Strongly discouraged**

Verdict for SHA-2: **Safe for use**
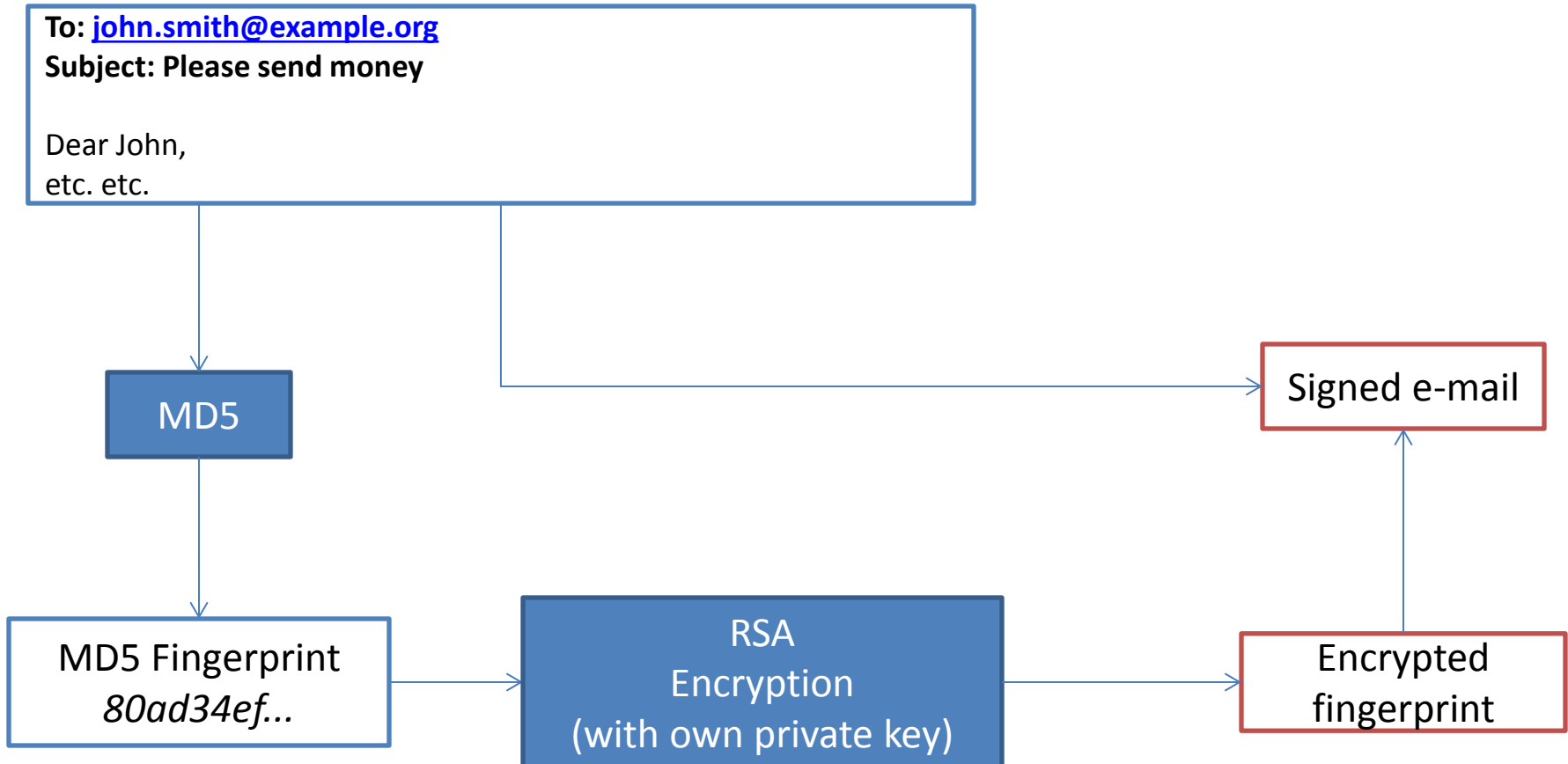
MD5 and SHA use cases

- File checksums
  - For programs, packages, spreadsheets
- Certificate fingerprints (for HTTPS, more on that later)
- Password storage (Linux /etc/shadow)
- Distributed version control (Git, Mercurial)
- Network protocols with protection against message tampering

Message	Hash(Message)

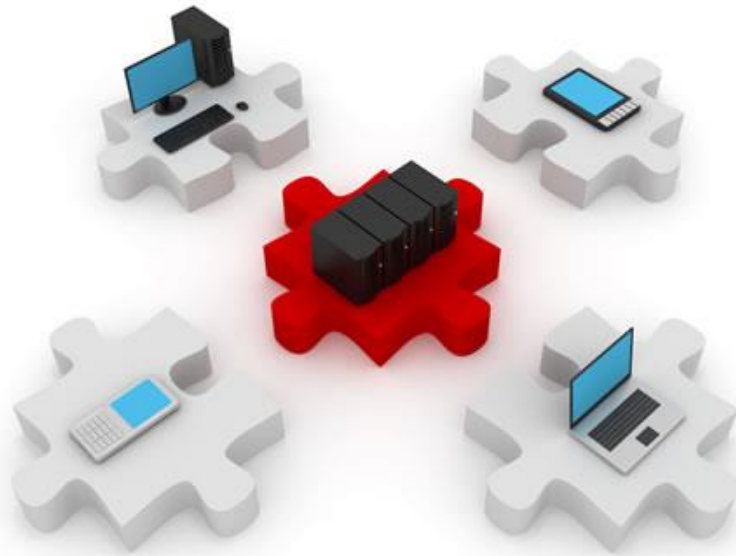Is this message protected against tampering?

Keyed Hashing
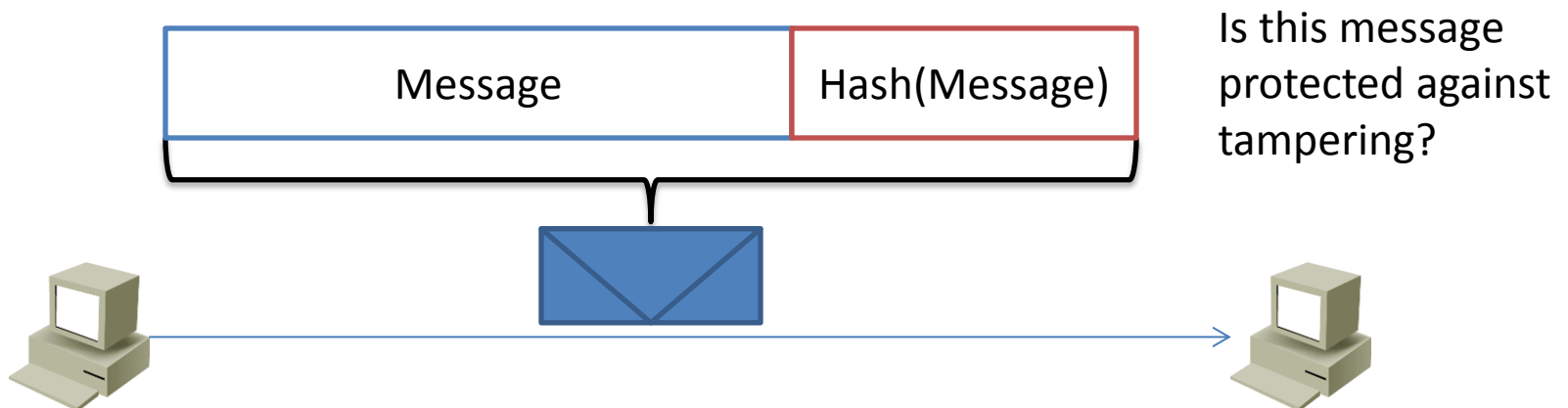
- Same as hashing, but a **pre-shared** key is also hashed along with the message

- Also known as HMAC (Hash-based Message Authentication Code)
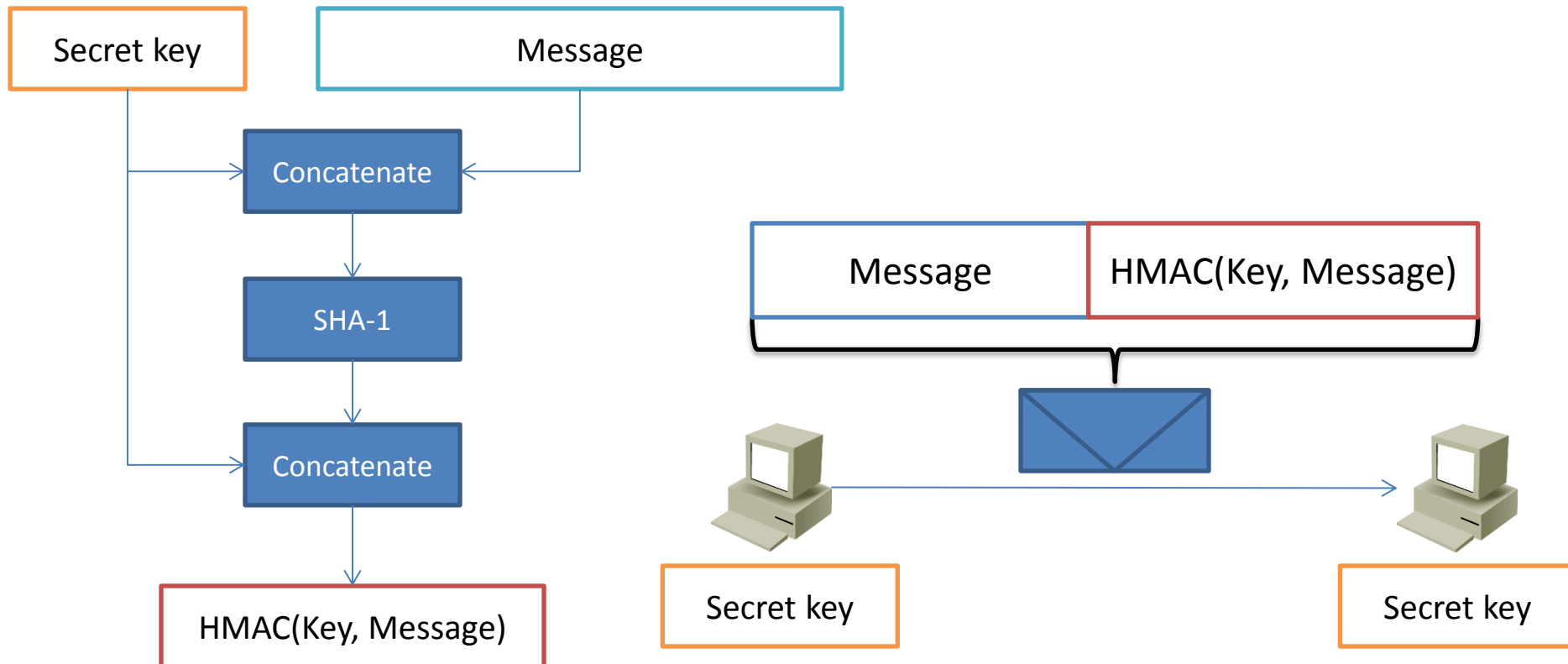
- The RSA private key can also be used for encryption
    - The advantage is that anyone can decrypt, thus proving that the data was actually encrypted by the sender
    - Also provides **non-repudiation**

> **To: john.smith@example.org**
> **Subject: Please send money**
>
> Dear John,
> etc. etc.

MD5

MD5 Fingerprint
80ad34ef...

RSA
Encryption
(with own private key)

Encrypted
fingerprint

Signed e-mail

43

AUTHENTICATION

Password based authentication

- Storing passwords in clear is not recommended
  - Any breach immediately compromises user accounts

Password based authentication
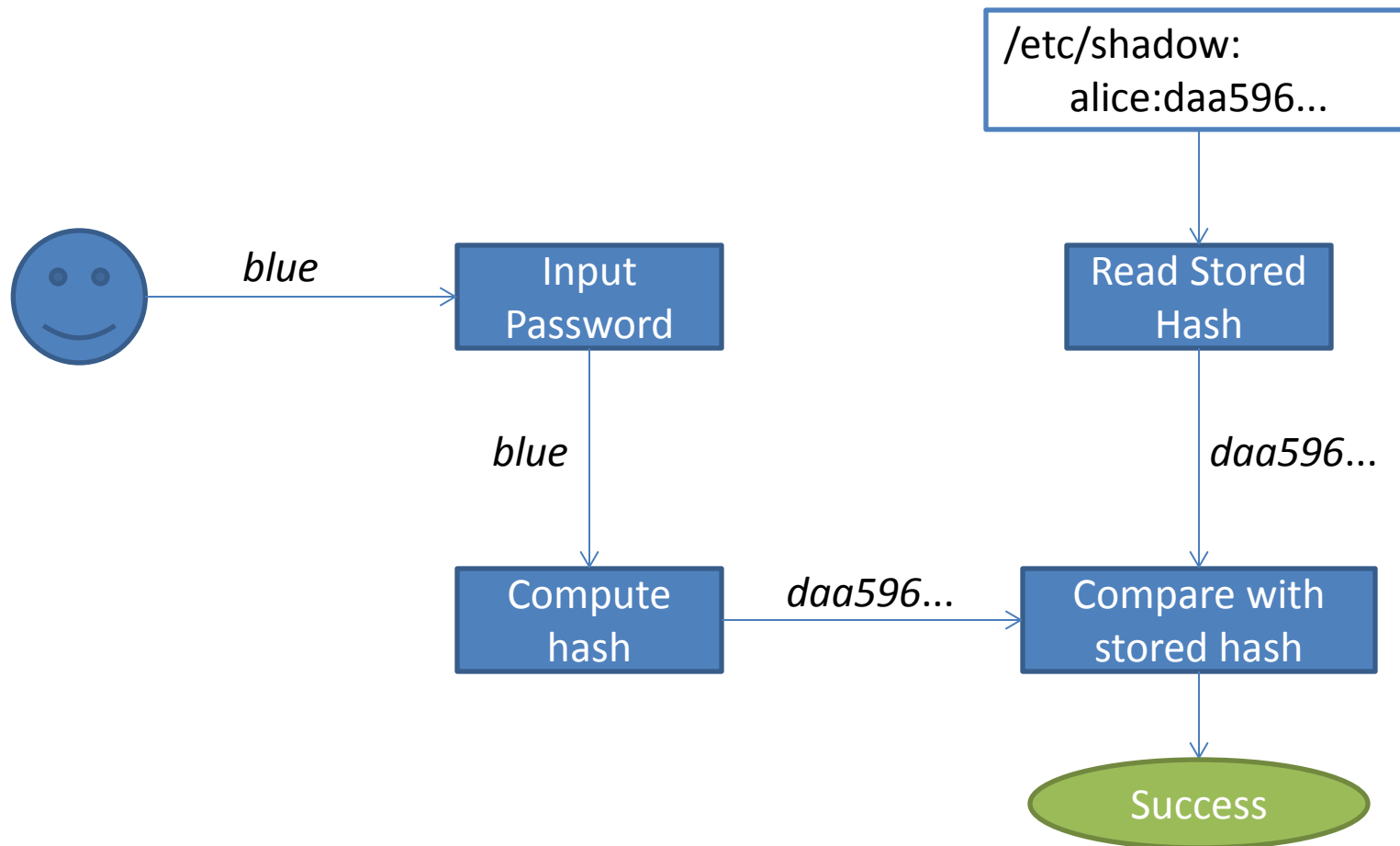
- ## Everything is better with salt
  - The same password will produce different hashes due to different salts
  - Prebuilt hash libraries for common passwords are useless

Anatomy of a Linux password hash:

6eQUjSSnn$E6zx40ad43xpmUxLB...ad

Actual hash

Salt

Hashing algorithm

Salt	Cleartext Password

Concatenate → Hash → Actual hash
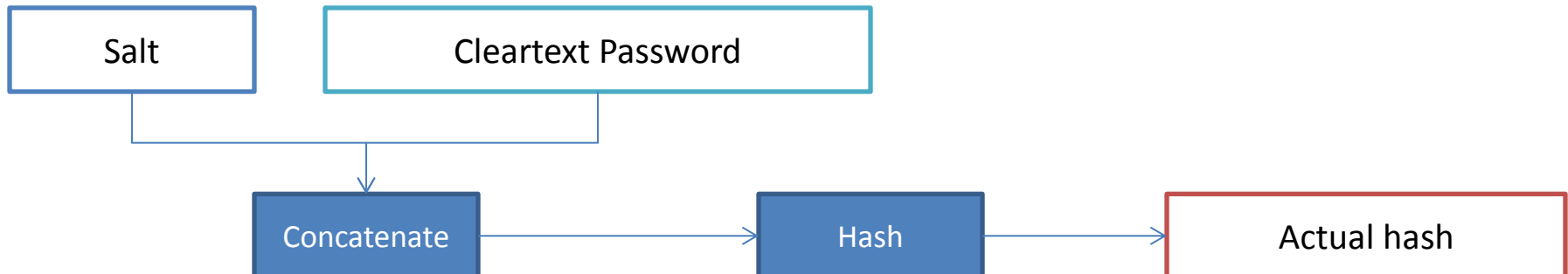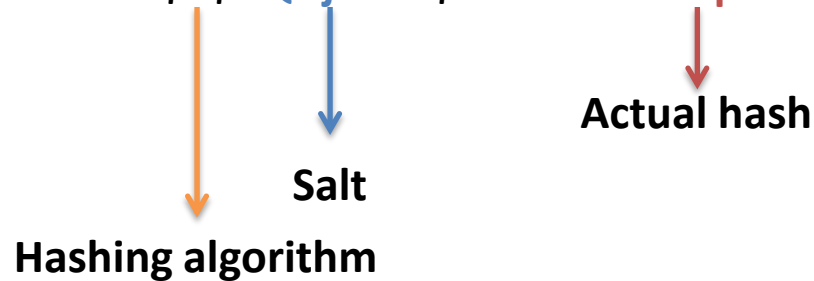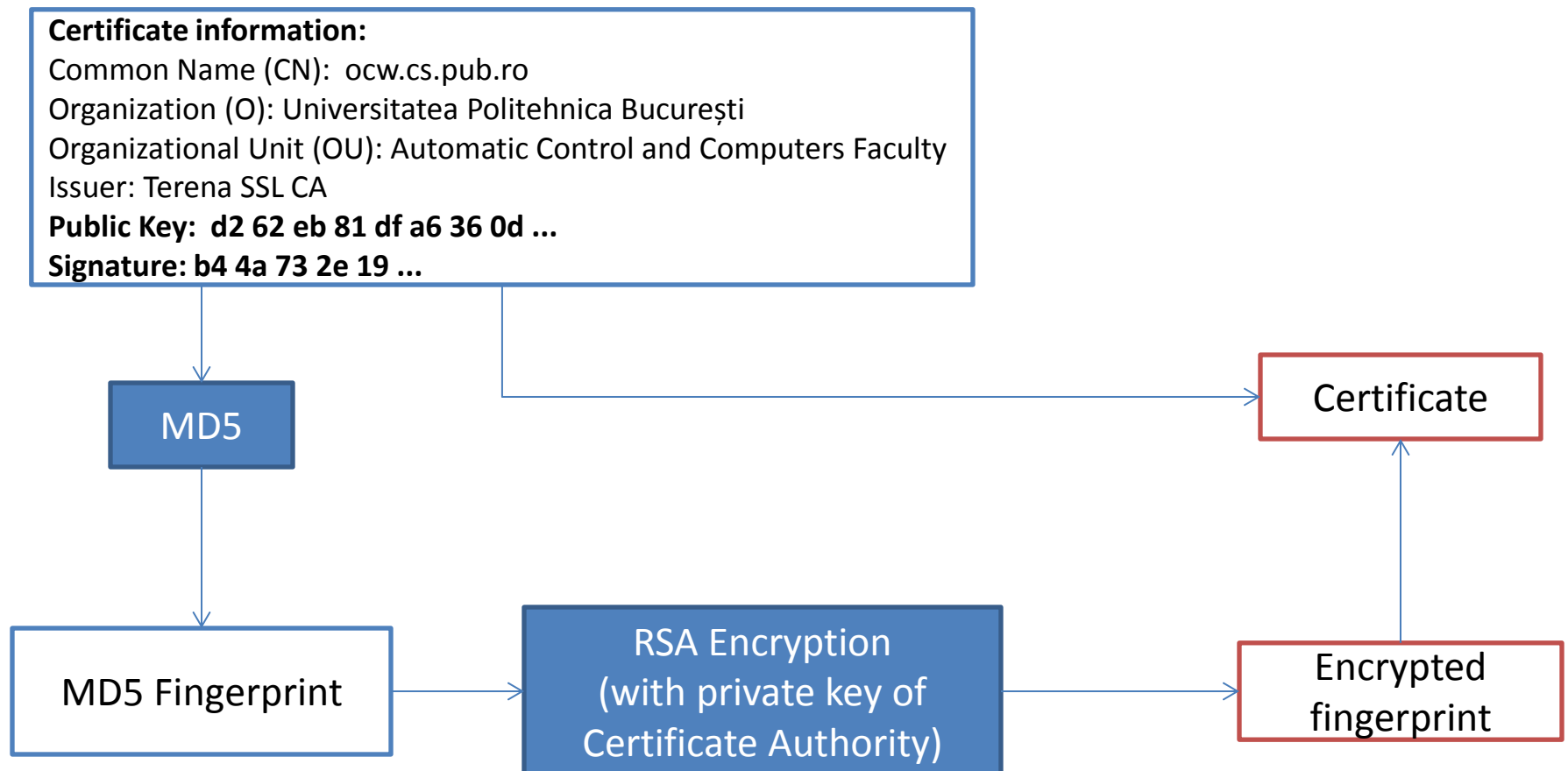
Challenge based authentication

- Solves the problem of authenticating endpoints when a secret key is pre-shared, without transmitting the key over the wire

Secret key K

R1

R2

Secret key K

Generate challenge C

Challenge C

Compute $H = \text{Hash}(\text{K}, C)$

Hash H

Compute $H' = \text{Hash}(\text{K}, C)$

Check $H = H'$

Success / Failure

Certificate based authentication
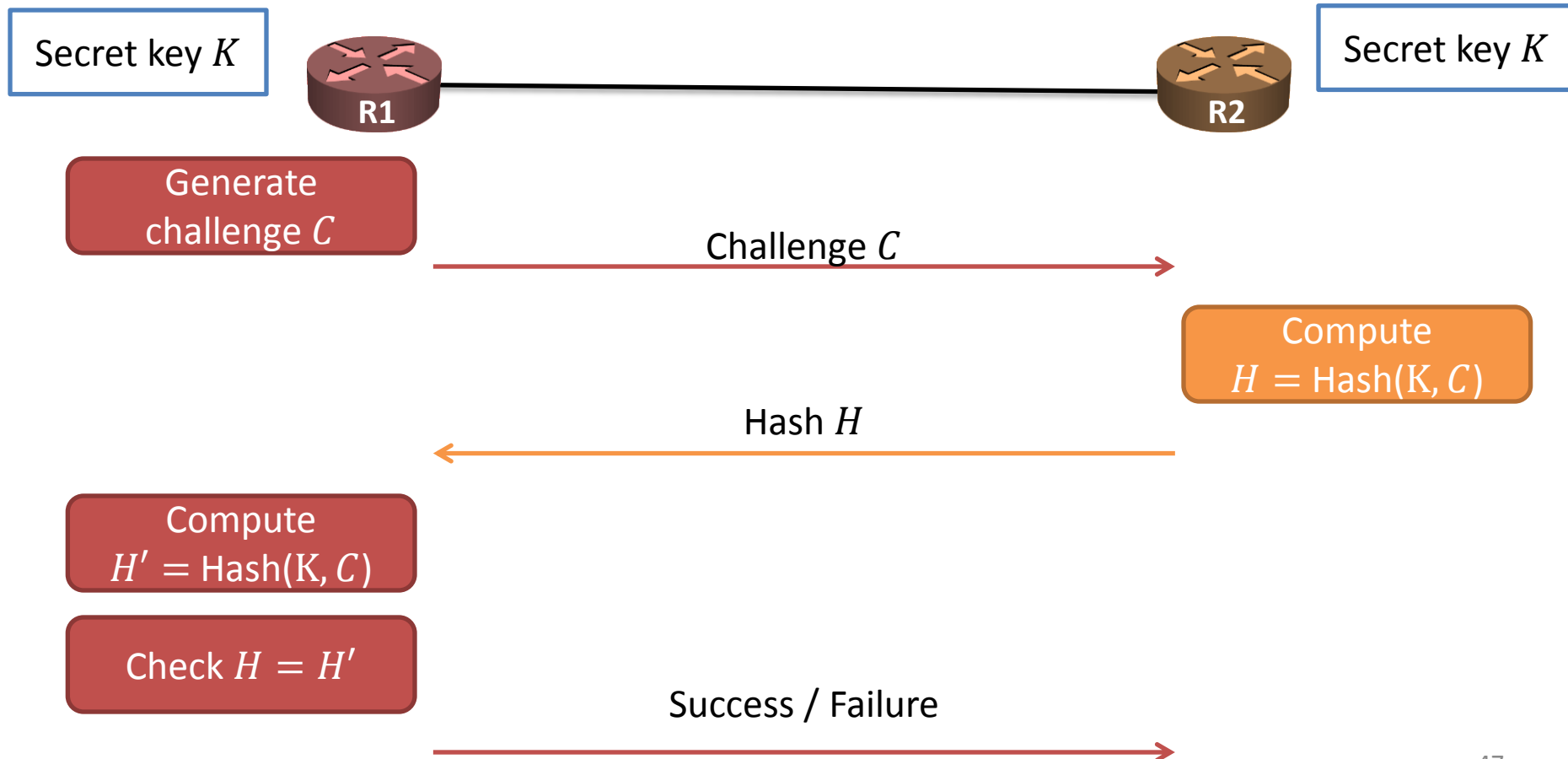
- Certificates contain information about an entity, verified by another trusted entity called a CA (certificate authority)
- Certificates are used to prove that the public key is legit

Certificate information:
Common Name (CN): ocw.cs.pub.ro
Organization (O): Universitatea Politehnica București
Organizational Unit (OU): Automatic Control and Computers Faculty
Issuer: Terena SSL CA
Public Key: d2 62 eb 81 df a6 36 0d …
Signature: b4 4a 73 2e 19 …

```
MD5
```

```
MD5 Fingerprint
```

```
RSA Encryption
(with private key of
Certificate Authority)
```

```
Certificate
```

```
Encrypted
fingerprint
```

HTTPS/TLS

- ## HTTPS = HTTP over TLS

Client — TLS: Server authenticates to client — Server

Client → Server: SSL settings, ciphers, etc.

Server → Client: SSL settings, ciphers, etc. + Certificate

Verify Certificate

Generate master secret S

Client → Server: S, encrypted with PubKey from Certificate

Decrypt master secret S

Generate session key $K = f(S)$

Generate session key $K = f(S)$

Notify further messages are encrypted

Finish setup, encrypted with K

Use K for encryption and integrity

TLS: Certificate verification

Certificate information:
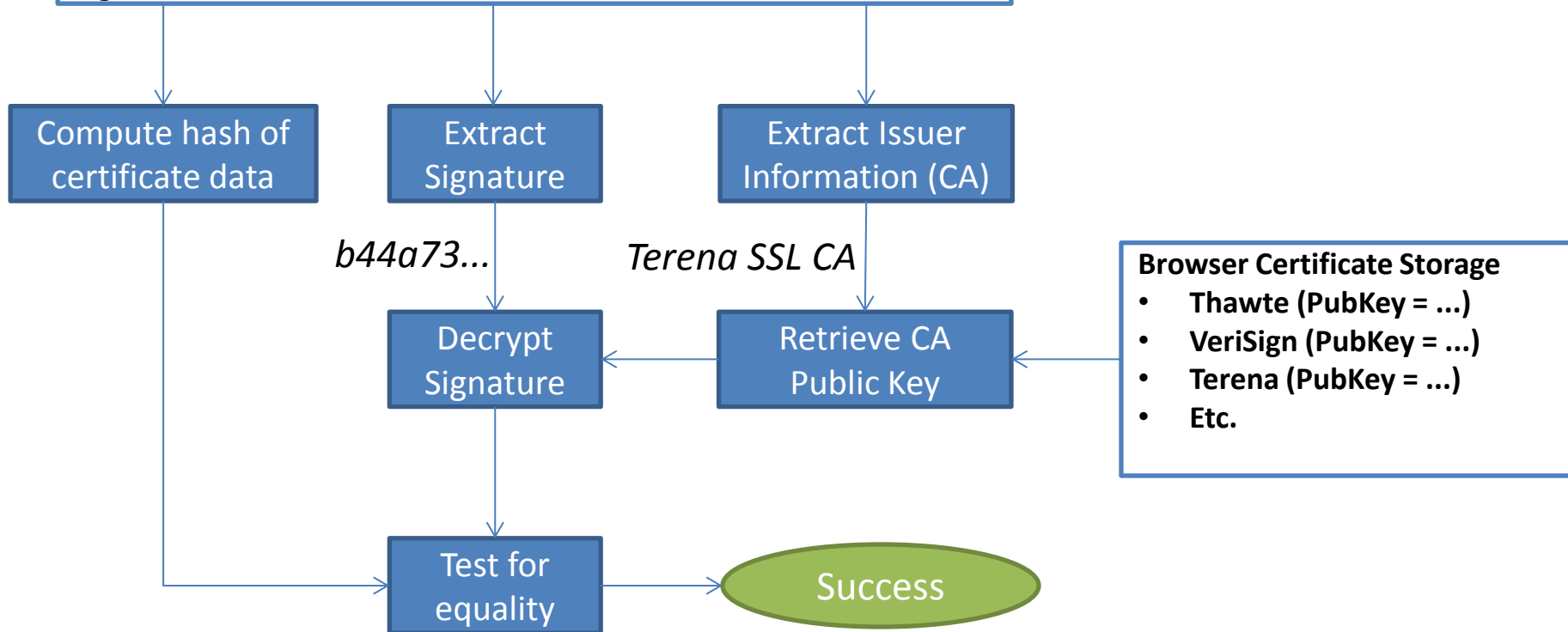Common Name (CN): ocw.cs.pub.ro
Organization (O): Universitatea Politehnica București
Organizational Unit (OU): Automatic Control and Computers Faculty
Issuer: Terena SSL CA
Public Key: d2 62 eb 81 df a6 36 0d ...
Signature: b4 4a 73 2e 19

- Compute hash of certificate data
- Extract Signature
- Extract Issuer Information (CA)

b44a73...

Terena SSL CA

- Decrypt Signature
- Retrieve CA Public Key

Browser Certificate Storage
- **Thawte (PubKey = ...)**
- **VeriSign (PubKey = ...)**
- **Terena (PubKey = ...)**
- **Etc.**

- Test for equality
- Success

- Two paranoid users do not trust **any** third party
- How can they establish a secure channel on the Internet, without exchanging prior knowledge? Secure means:
  - Tampering must be detected
  - No third party is able to retrieve private data (even if said party poses as a one of the users, or performs a MITM attack)
  - Covert information or side channels may not be used

Conclusion

- Cryptographic algorithms for:
  - Data confidentiality
  - Data integrity
  - Authentication

- Further reading:
  - The codebreakers, by David Kahn
  - The code book, by Simon Singh
  - Handbook of applied cryptography, by Alfred Menezes

- Other related topics not discussed now: steganography, covert channels, homomorphic encryption, identity-based encryption, elliptic curve cryptography, pairing-based cryptography, Tor network