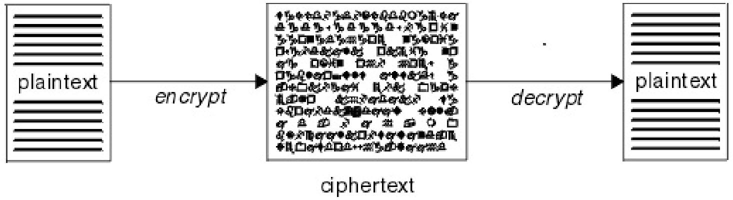
Cryptography

Jordi Cortadella and Jordi Petit

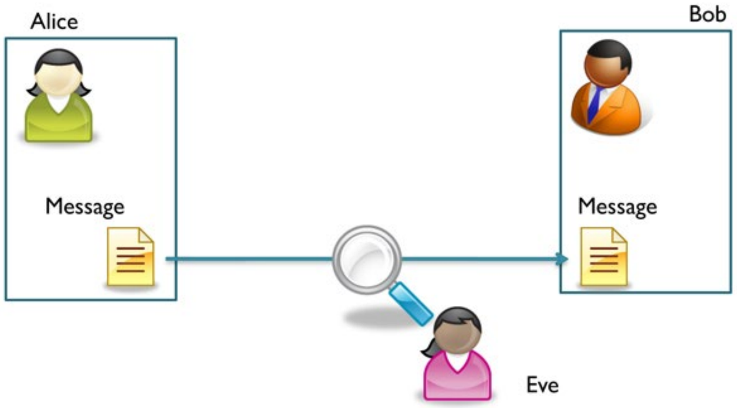
Department of Computer Science

Where do we need cryptography?

- Communication (e.g., sending private emails).
- Digital signatures, i.e., guarantee that digital documents are authentic.
- Network services over unsecure networks (e.g., secure shell (ssh) for remote login, file transfers, remote command execution, etc.).
- HyperText Transfer Protocol Secure (HTTPS): secure communication on Internet.
- Cryptocurrencies (e.g., bitcoin)

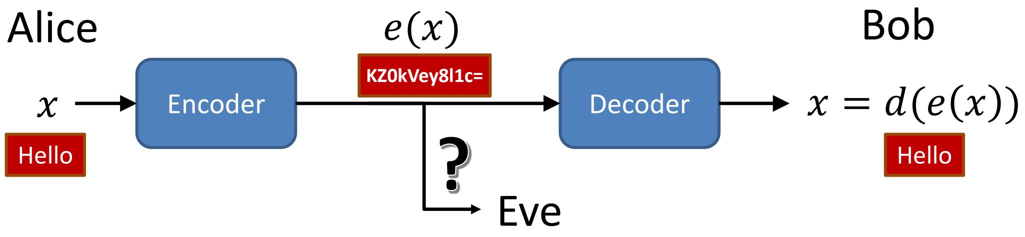


Cryptography



- How can we avoid an eavesdropper (Eve) to overhear a message sent from Alice to Bob?
- Solution: encrypt the message!

Cryptosystem

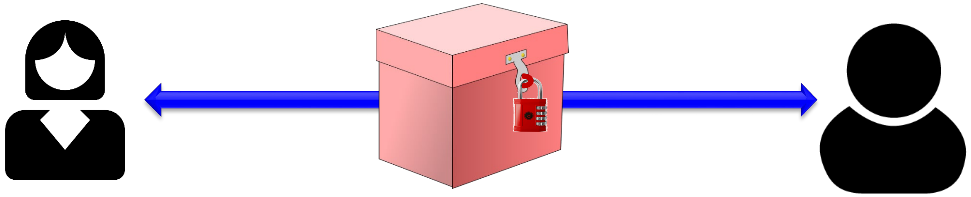


- The encryption function $e(x)$ must be invertible. The inverse is $d(\cdot)$.
- Two schemes:
  - The traditional: Secret-key protocols (symmetric).
  - The modern: Public-key protocols (asymmetric).

Secret-key protocols

- Alice and Bob have to meet privately and chose a secret key.

- They can use the secret key to mutually exchange messages.



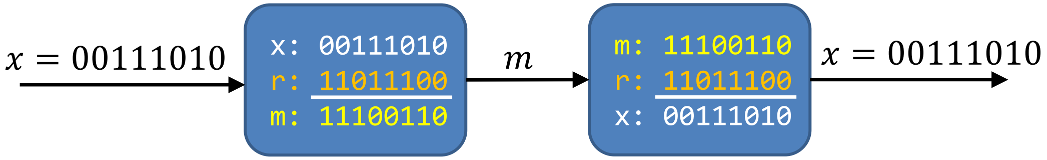
- There are many secret-key protocols. We will explain two of them:
  - XOR encoding.
  - Advanced Encryption Standard (AES).

Secret-key protocol: XOR encoding

- A secret key r is chosen (a binary string).

- The encoding and decoding functions are identical: $e_r(x) = d_r(x) = x \oplus r$.

- Example: $r = 11011100$.



$x = 00111010$

x: 00111010
r: 11011100
m: 11100110

m

m: 11100110
r: 11011100
x: 00111010

$x = 00111010$

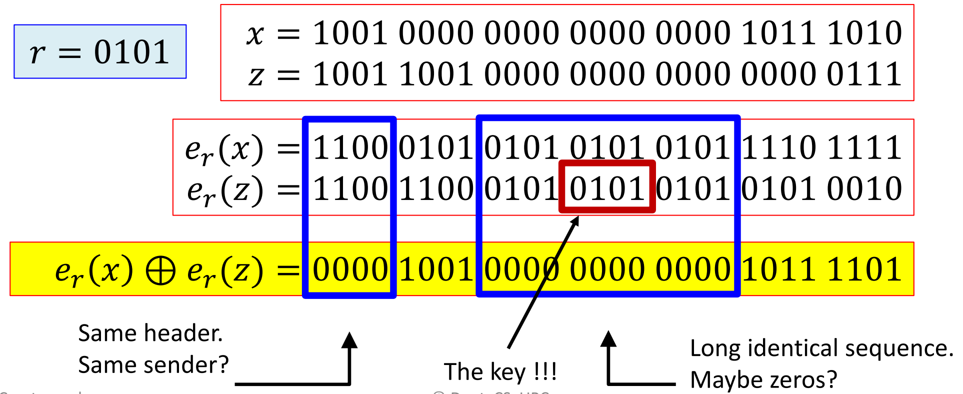
- It is convenient that the bits of r are randomly generated.

- Still, this is not a very robust scheme since Eve can figure out important information by listening several messages.

Secret-key protocol: XOR encoding

- If the key is too short, it needs to be applied many times (once for each block). Messages often show similarities and repeated patterns (same header, same tail, long sequences of zeros, …).

- If we send two messages, $e_r(x)$ and $e_r(z)$, then $e_r(x) \oplus e_r(z)$ may reveal important information. It is convenient to change the key at every message (one-time pad).



$r = 0101$

$x = 1001\ 0000\ 0000\ 0000\ 0000\ 1011\ 1010$
$z = 1001\ 1001\ 0000\ 0000\ 0000\ 0000\ 0111$

$e_r(x) = 1100\ 0101\ 0101\ 0101\ 0101\ 1110\ 1111$
$e_r(z) = 1100\ 1100\ 0101\ 0101\ 0101\ 0101\ 0010$

$e_r(x) \oplus e_r(z) = 0000\ 1001\ 0000\ 0000\ 0000\ 1011\ 1101$

Same header.
Same sender?

The key !!!

Long identical sequence.
Maybe zeros?

Secret-key protocol: AES

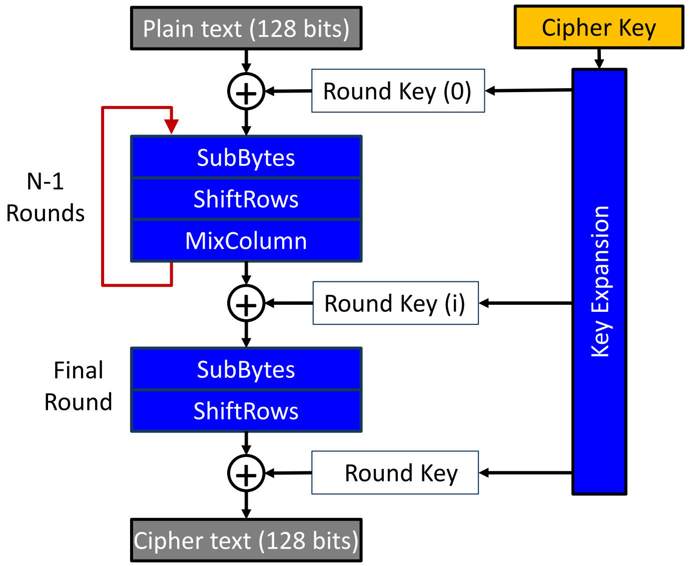
- AES: Advanced Encryption Standard.

- Established as a standard by the U.S. National Institute of Standards and Technology (NIST) in 2001.

- Very robust and used worldwide.

- A family of ciphers with different key and block sizes (key sizes: 128, 196 and 256 bits).

AES scheme



Plain text (128 bits)

Cipher Key

Round Key (0)

N-1 Rounds

SubBytes

ShiftRows

MixColumn

Round Key (i)

Final Round

SubBytes

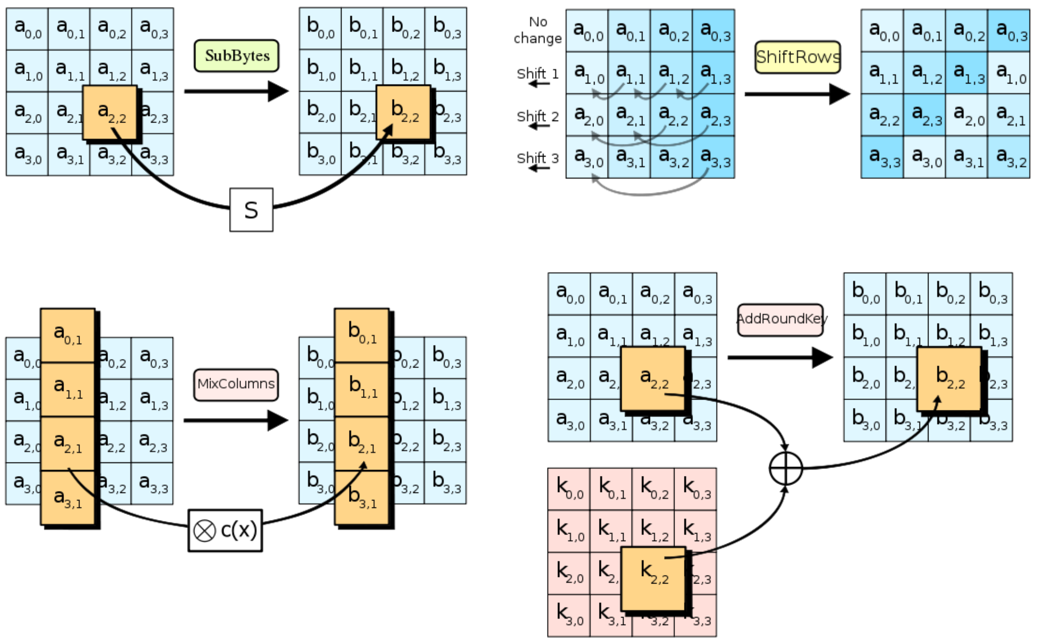
ShiftRows

Round Key

Cipher text (128 bits)

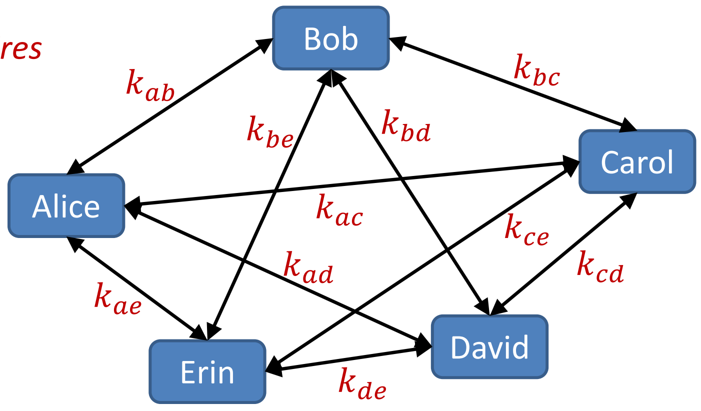
Key Expansion

AES steps



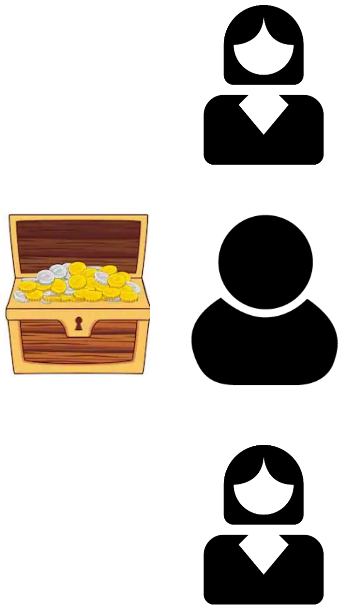
Secret-key protocols: problems

Every channel requires a different key

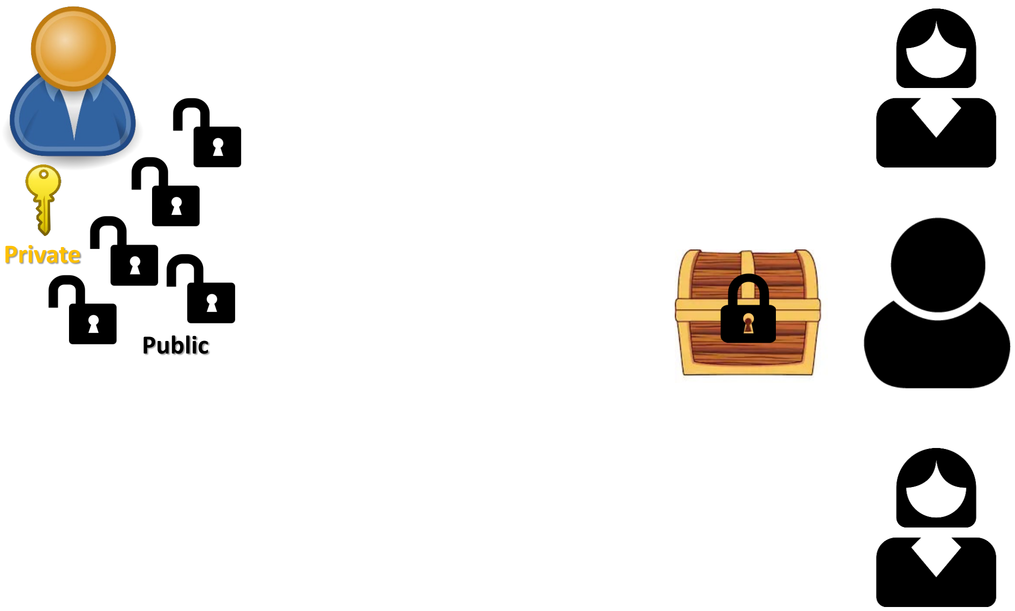


The key cannot be transmitted through the communication channel !

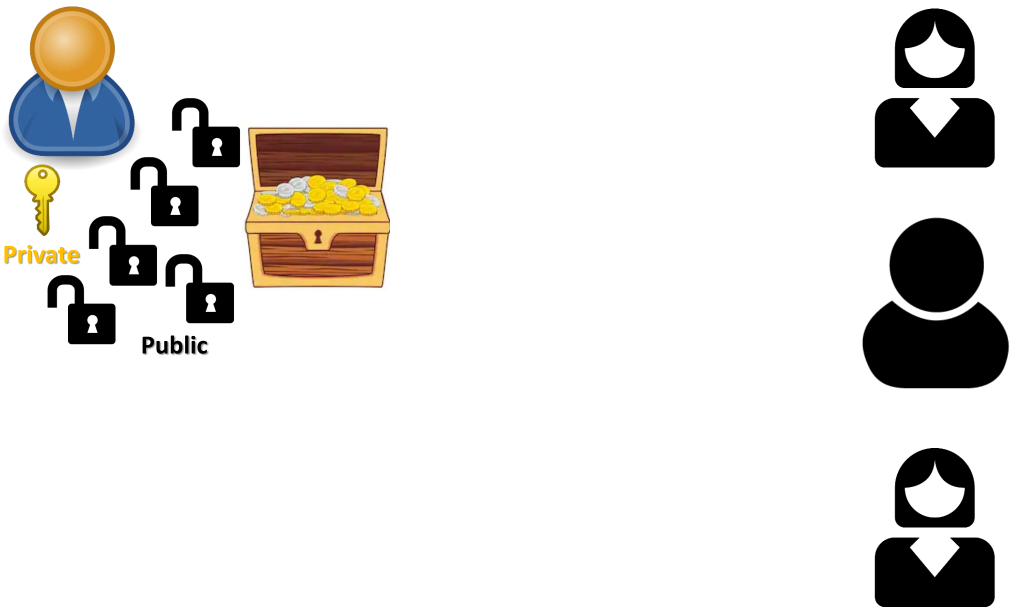
Public-key protocols



Public-key protocols



Public-key protocols



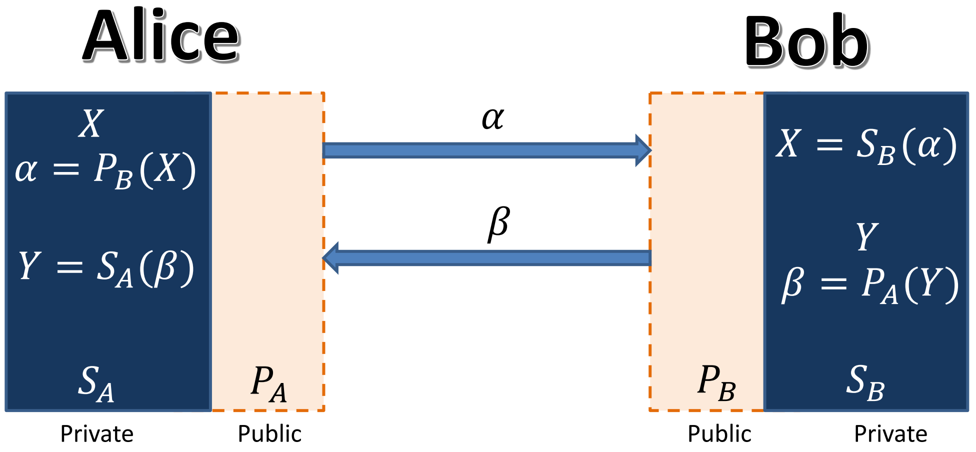
Public-key protocols

- Each participant generates a *public key* (P) and a (private) *secret key* (S). Public keys are revealed to everybody.
- The public/secret keys are a matched pair, i.e.,

$$M = S\big(P(M)\big) = P\big(S(M)\big)$$

- If Alice has the pair (P_A, S_A), anybody can compute $P_A(X)$, but only Alice can compute $S_A(X)$.
- If Bob wants to send a secret message M to Alice, Bob will compute $X = P_A(M)$ and send it to Alice. Only Alice will be able to decipher the message: $M = S_A(X)$.

Public-key protocols



But, how to create a cryptosystem like this?
Using number theory.

RSA cryptosystem

- Public-key cryptosystem (Rivest-Shamir-Adleman, 1977).
- Based upon number theory: modular arithmetic and prime numbers.
- Security: based on the fact that factoring a large number (product of two large primes) is hard.

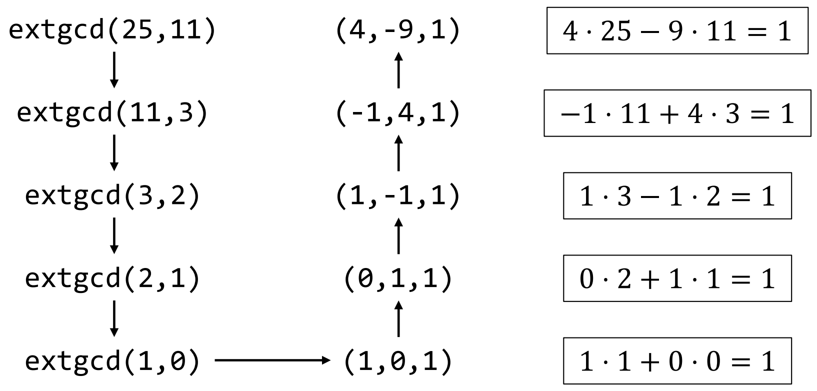
Bézout's identity

- **Lemma:** If d divides both a and b, and $d = ax + by$ for some integers x and y, then necessarily $d = \gcd(a, b)$.
- **Proof:**
  - Clearly, $d \leq \gcd(a,b)$, since d is a divisor of a and b.
  - Since $\gcd(a,b)$ is a divisor of a and b, it must also be a divisor of $ax + by = d$. This implies that $\gcd(a,b) \leq d$.
  - Therefore, $d = \gcd(a,b)$.

Extended Euclid's algorithm

```
function extendedGcd($a$,$b$)
// Input: two positive integers $a$ and $b$, with $a \geq b \geq 0$
// Output: Integers $x$,$y$,$d$ such that $d = \gcd(a,b)$ and $ax + by = d$
  if $b = 0$: return $(1,0,a)$
  $(x', y', d)$ = extendedGcd($b, a \bmod b$)
  return $(y', x' - \lfloor a/b \rfloor y', d)$
```

Proof by induction using the identity
$a \bmod b = a - \lfloor a/b \rfloor b$



extgcd(25,11) (4,-9,1) $4 \cdot 25 - 9 \cdot 11 = 1$

extgcd(11,3) (-1,4,1) $-1 \cdot 11 + 4 \cdot 3 = 1$

extgcd(3,2) (1,-1,1) $1 \cdot 3 - 1 \cdot 2 = 1$

extgcd(2,1) (0,1,1) $0 \cdot 2 + 1 \cdot 1 = 1$

extgcd(1,0) → (1,0,1) $1 \cdot 1 + 0 \cdot 0 = 1$

Modular arithmetic: properties

- $x \bmod N$ is the remainder when x is divided by N.

$$x \equiv y \pmod N \quad \Leftrightarrow \quad N \text{ divides } (x - y).$$

- Addition and multiplication. Let us assume $a_1 \equiv b_1 \pmod N$ and $a_2 \equiv b_2 \pmod N$, then

$$a_1 + a_2 \equiv b_1 + b_2 \pmod N$$
$$a_1 \cdot a_2 \equiv b_1 \cdot b_2 \pmod N$$

- Exponentiation. Let us assume $a \equiv b \pmod N$, then

$$a^k \equiv b^k \pmod N$$

Modular arithmetic: properties

- Given a and N, we say that x is the multiplicative inverse of a (mod N) if

$$ax \equiv 1 \pmod{N}.$$

Example. The multiplicative inverse of 4 (mod 7) is 2:

$$4 \cdot 2 \equiv 1 \pmod{7}$$

- When $\gcd(a, N) = 1$, the multiplicative inverse of a always exists and can be calculated by the extended Euclid's algorithm:

$$ax + Ny = 1 \quad \Rightarrow \quad x \text{ is } a'\text{s inverse (mod N)}$$

Fundamental property

Let p and q be any two primes and $N = pq$. $\phi(N) = (p-1)(q-1)$ is the totient of N, i.e., the number of positive integers smaller than N which are co-prime to N.

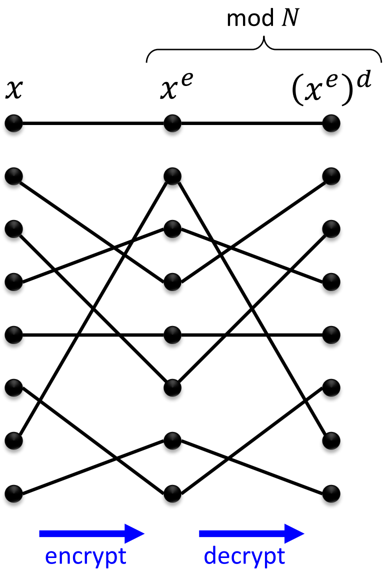
For any e co-prime to $\phi(N)$:

1. The mapping $x \mapsto x^e \bmod N$ is a bijection on $\{0,1, \dots, N-1\}$.

2. The inverse mapping can be obtained as follows. Let d be the inverse of e modulo $\phi(N)$. Then for all $x \in \{0, \dots, N-1\}$,

$$(x^e)^d \equiv x \mod N.$$

The RSA cryptosystem



Bob chooses public and secret keys:

- Bob picks two large random primes, p and q.
- The public key is (N, e), where $N = pq$ and e is a small number co-prime to $(p-1)(q-1)$.
- The secret key is d, the inverse of e modulo $(p-1)(q-1)$, computed using the extended Euclid's algorithm.

Alice sends a message x to Bob:

- Alice takes Bob's public key (N, e) and sends $y = (x^e \bmod N)$.
- Bob decodes the message by computing $y^d \bmod N$.

The RSA cryptosystem: example

- Let $p = 5$ and $q = 17$, thus $N = 85$ and $\phi(N) = 64$.
- Let $e = 3$. It satisfies: $\gcd(e, \phi(N)) = \gcd(3,64) = 1$.
- We calculate $d = 3^{-1} \bmod 64 = 43$ using extended Euclid's algorithm:

$$43 \cdot 3 - 2 \cdot 64 = 1$$

Note: the algorithm gives $1 \cdot 64 - 21 \cdot 3 = 1$, but $-21 = 43 \pmod{64}$

- Let us consider the message $x = 12$.
  - The sender must encrypt x as $y = 12^3 \bmod 85 = 28$.
  - The receiver must decrypt y by computing $x = 28^{43} \bmod 85 = 12$.
- **Remember:** x^k can be efficiently computed with $\log_2 k$ multiplications. **Note:** Multiplication and division of "long" numbers is required (similar to multiplication of polynomials).

Why is RSA secure?

- Typical sizes for p and q are 1024-bit numbers with values larger than $2^{1023.5} \approx 1.8 \times 10^{308}$.
- Eve knows the public key (N, e) and the message y. How can she guess x? There are two options:

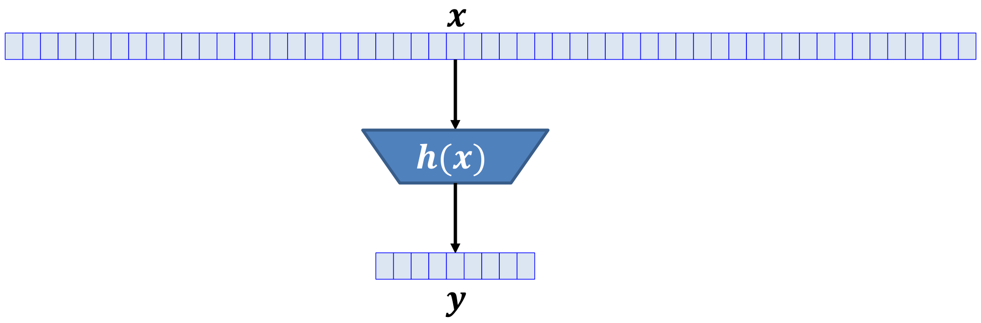
1. Try all possible values of x and check whether $y = x^e \bmod N$. But x is a large n-bit number and checking all values would take exponential time (impractical).
2. Try to guess d and calculate $x^d \bmod N$. This would require to calculate the inverse of e modulo $(p-1)(q-1)$. But p and q are not known unless the factors of N are calculated. Factoring is still a hard problem.

Hybrid cryptosystems

- Public-key cryptosystems (e.g., RSA) are convenient (no need to share keys) but computationally expensive. Secret-key (symmetric) cryptosystems (e.g. AES) are more efficient. Both can be combined.
- Bob wants to send an encrypted message to Alice:
  - Bob generates a new symmetric key k and encrypts the data with this key (using AES).
  - Bob encrypts k using Alice's public key (using RSA).
  - Bob sends both encryptions to Alice.
- Alice wants to decrypt Bob's message:
  - Alice uses her private key to decrypt the encrypted symmetric key k.
  - Alice uses the symmetric key k to decrypt the data (using AES).

Cryptographic hash function (CHF)

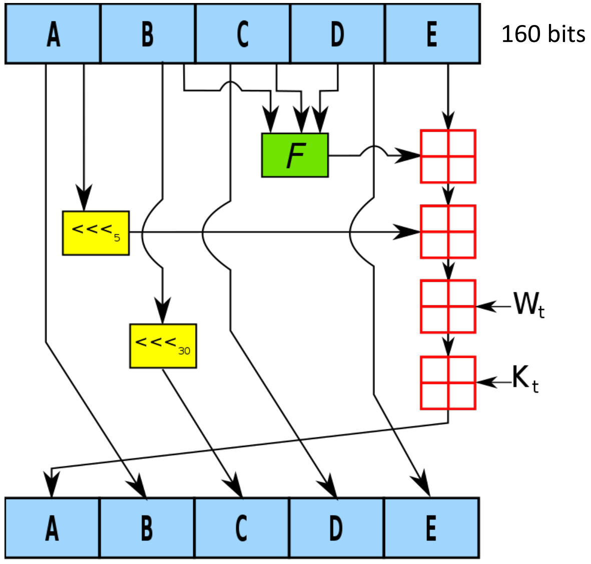
A CHF maps data of arbitrary size to a fixed-size bit string.



Cryptographic hash function (CHF)

- Properties:
  - **Easy** to compute.
  - **Pre-image resistance**: if $y = h(x)$, it is difficult to find x from y.
  - **Collision resistance**: It is difficult to find two inputs, x_1 and x_2, such that $h(x_1) = h(x_2)$.
- Popular CHFs:
  - Message Digest: MD2, MD4, MD5 and MD6. It is a 128-bit hash function.
  - Secure Hash Function: SHA-0, SHA-1, SHA-2, SHA-3. They produce hash values with 160 bits (SHA-1) or 256 bits (SHA-2).
  - And some others …

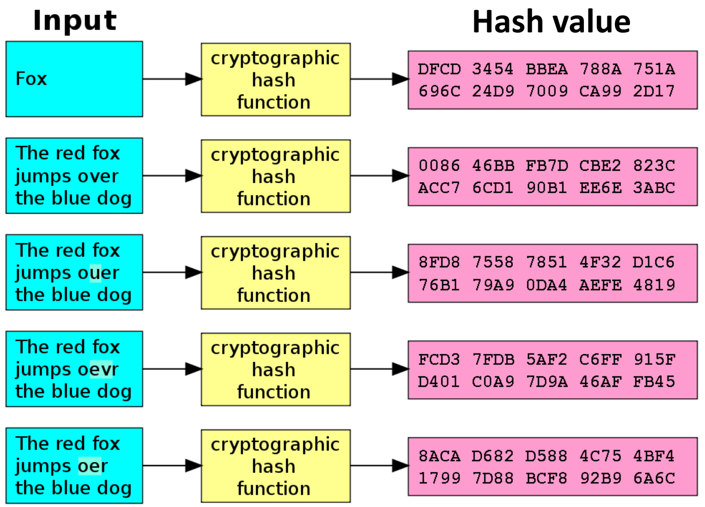
Example: SHA-1



One step of SHA-1

The result is “accumulated”
to the result of previous steps.

Example: SHA-1

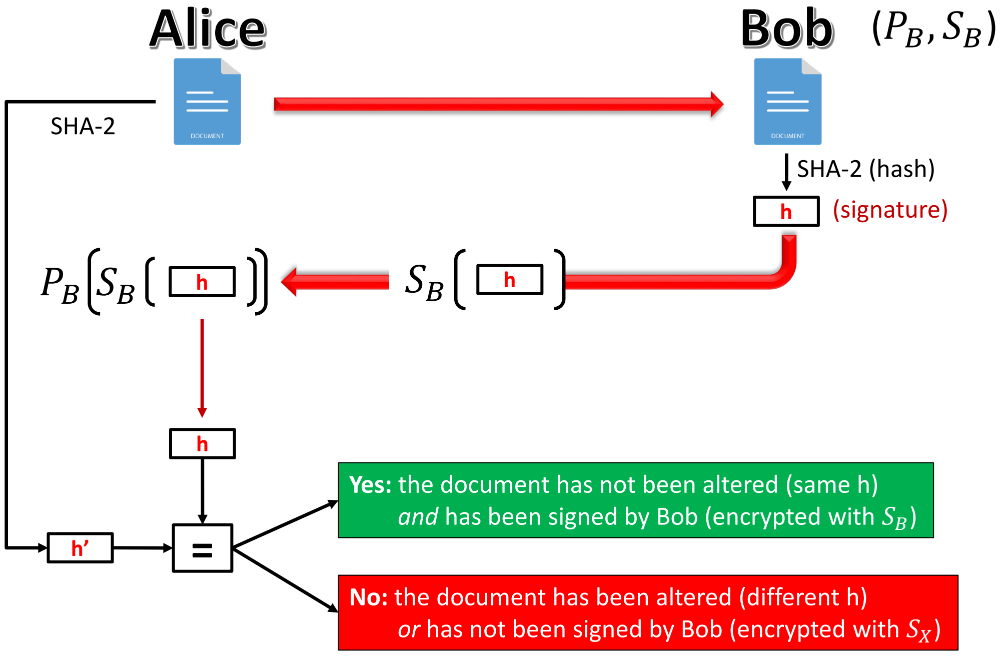


Input		Hash value
Fox	cryptographic hash function	DFCD 3454 BBEA 788A 751A 696C 24D9 7009 CA99 2D17
The red fox jumps over the blue dog	cryptographic hash function	0086 46BB FB7D CBE2 823C ACC7 6CD1 90B1 EE6E 3ABC
The red fox jumps ouer the blue dog	cryptographic hash function	8FD8 7558 7851 4F32 D1C6 76B1 79A9 0DA4 AEFE 4819
The red fox jumps oevr the blue dog	cryptographic hash function	FCD3 7FDB 5AF2 C6FF 915F D401 C0A9 7D9A 46AF FB45
The red fox jumps oer the blue dog	cryptographic hash function	8ACA D682 D588 4C75 4BF4 1799 7D88 BCF8 92B9 6A6C

Digital signatures

- A scheme to guarantee that a message is authentic.
- Consider the following case:
  - Alice sends a document (possibly unencrypted) to Bob and wants Bob to electronically sign the document.
  - Bob “signs” the document and sends it back to Alice.
- Questions:
  - How does Alice know that the document has not been altered? → integrity.
  - How does Alice know that Bob has signed the document (and not somebody else)? → authentication.

Digital signatures


Alice
Bob
(P_B, S_B)
SHA-2
SHA-2 (hash)
h
(signature)
$S_B\left(\boxed{h}\right)$
$P_B\left[S_B\left(\boxed{h}\right)\right]$
h
h’
=

Yes: the document has not been altered (same h) *and* has been signed by Bob (encrypted with S_B)

No: the document has been altered (different h) *or* has not been signed by Bob (encrypted with S_X)

The pending challenge

How to generate large prime numbers?

(not explained in this lecture)

Internet + Privacy → Cryptography

EXERCISES

Simple cryptographic hash

We want to use the XOR operator $\oplus$ for cryptographic hashing as follows. We split every message M into blocks B_i of 5 bits, e.g., $M = 11101 \cdot 00011 \cdot 10100 \cdot 110$. In case the length is not a multiple of 5, additional zeroes are added at the end of the message.

For a message M with k blocks, we define the cryptographic hash h as follows:

$$h(M) = B_1 \oplus B_2 \oplus \cdots \oplus B_k.$$

where $\oplus$ means the bitwise application of XOR. For example, $01110 \oplus 11010 = 10100$.

- What would be the output $h(M)$ for the previous message M?
- If we change one bit of a message, does the output change a lot?
- Assume that we know $h(M)$ and the length of M. Is it easy to find another M' with the same length such that $h(M) = h(M')$? Justify your answer.

Simple RSA

Assume you have $p = 5$ and $q = 7$.

- Which is the smallest value for e?
- What is the corresponding value for d?
- Encrypt the message $M = 3$.
- Find all possible pairs (e, d) valid for this cryptosystem.

Implement an RSA cryptosystem

- Given two primes, p and q, design an RSA cryptosystem (in Python or C++) as follows:
  - Let $N = p \cdot q$. Find the smallest $e \geq 3$, such that (N, e) can be used as public key. Use the extended gcd algorithm.
  - Find d that can be used for secret key.
  - Implement the function encode(x, e, N) that computes $x^e \bmod N$. This function must be efficient. Note: assume that N^2 can be represented as an int.
  - Implement a function to double check, for $0 \leq x < N$, that encode(encode$(x, e, N), d, N) = x$.
- Example: $p = 79$, $q = 491$.

  Public key: $(38789, 11)$, Secret key: 31271.

  $e(2) = 2048$, $e(19) = 23855$, $e(32757) = 4$, $e(38788) = 38788$, $e(10) = 18550$.