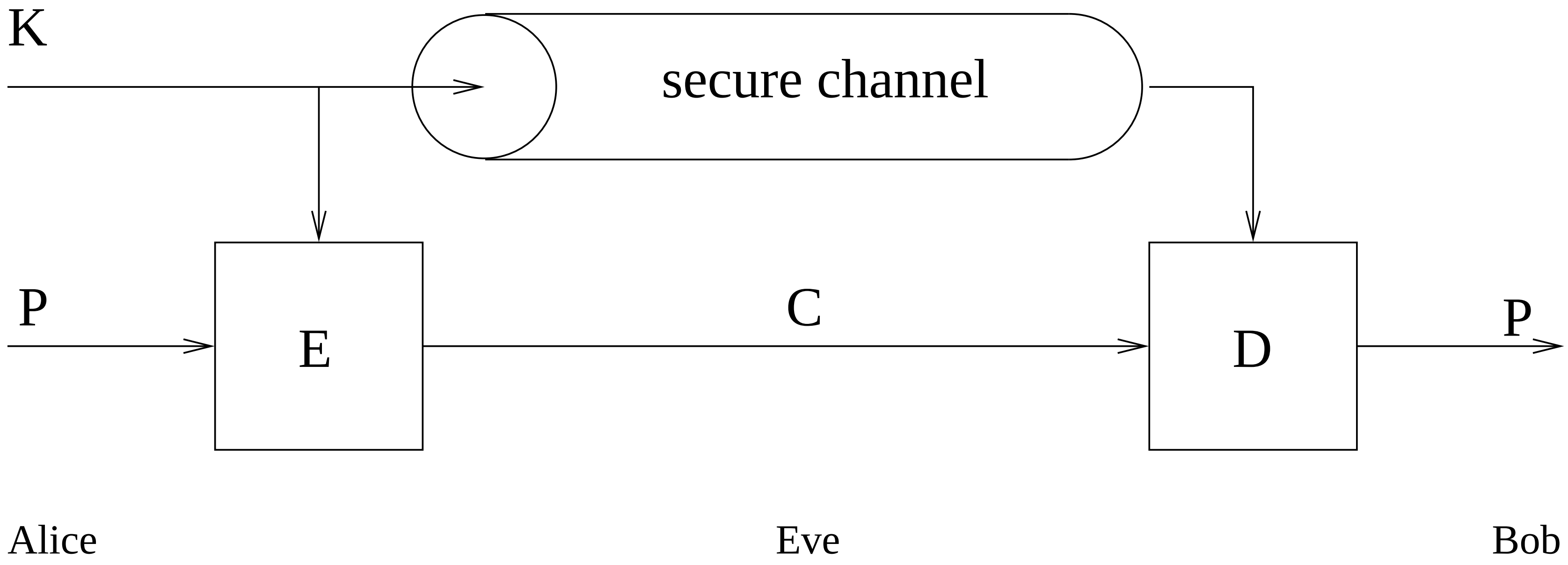
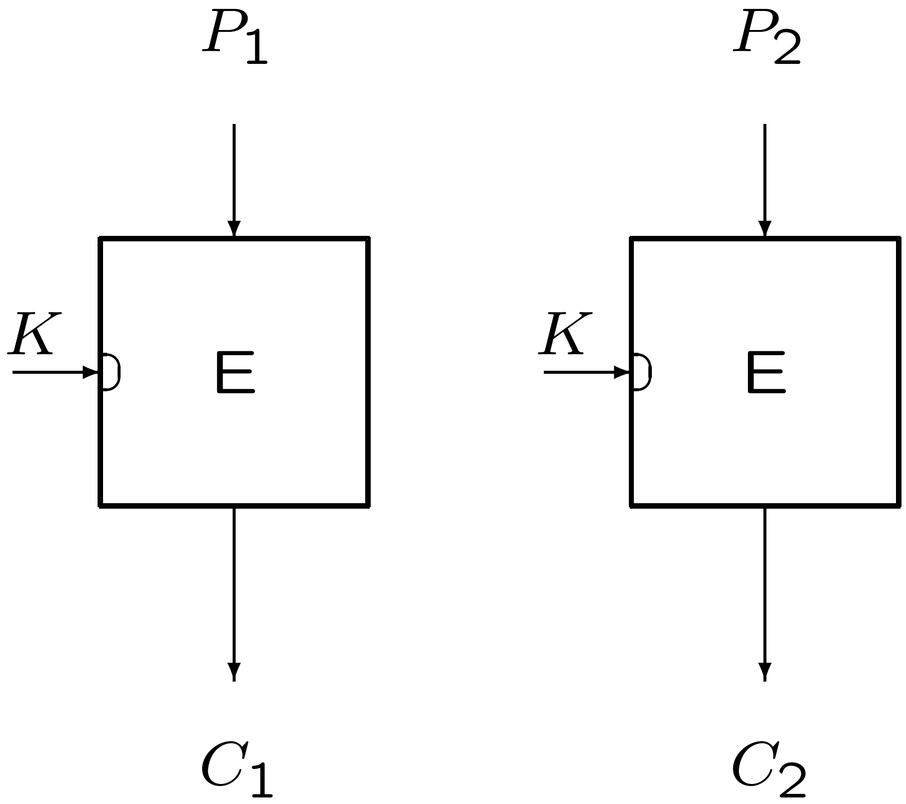
Symmetric cryptography



Modes Of Encryption (1/4, A)

Electronic Code Book (ECB)



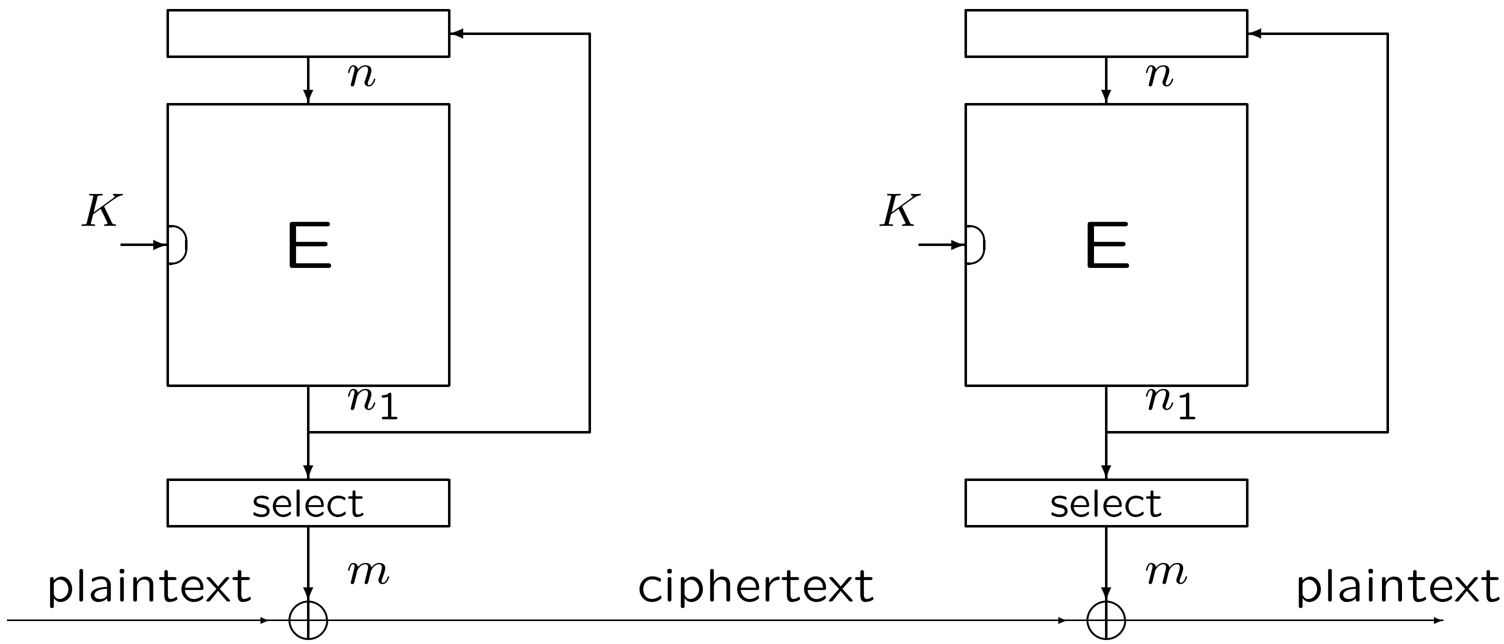
Modes Of Encryption (1/4, B)

ECB: properties

- ciphertext blocks independent
- does not hide patterns and repetitions
- errors: expansion within 1 block
- limited number of applications

Modes Of Encryption (2/4, A)

OFB: Synchronous stream cipher



n = block length, n_1 = # of feedback bits

$n_1 < n$, m = # of selected key stream bits

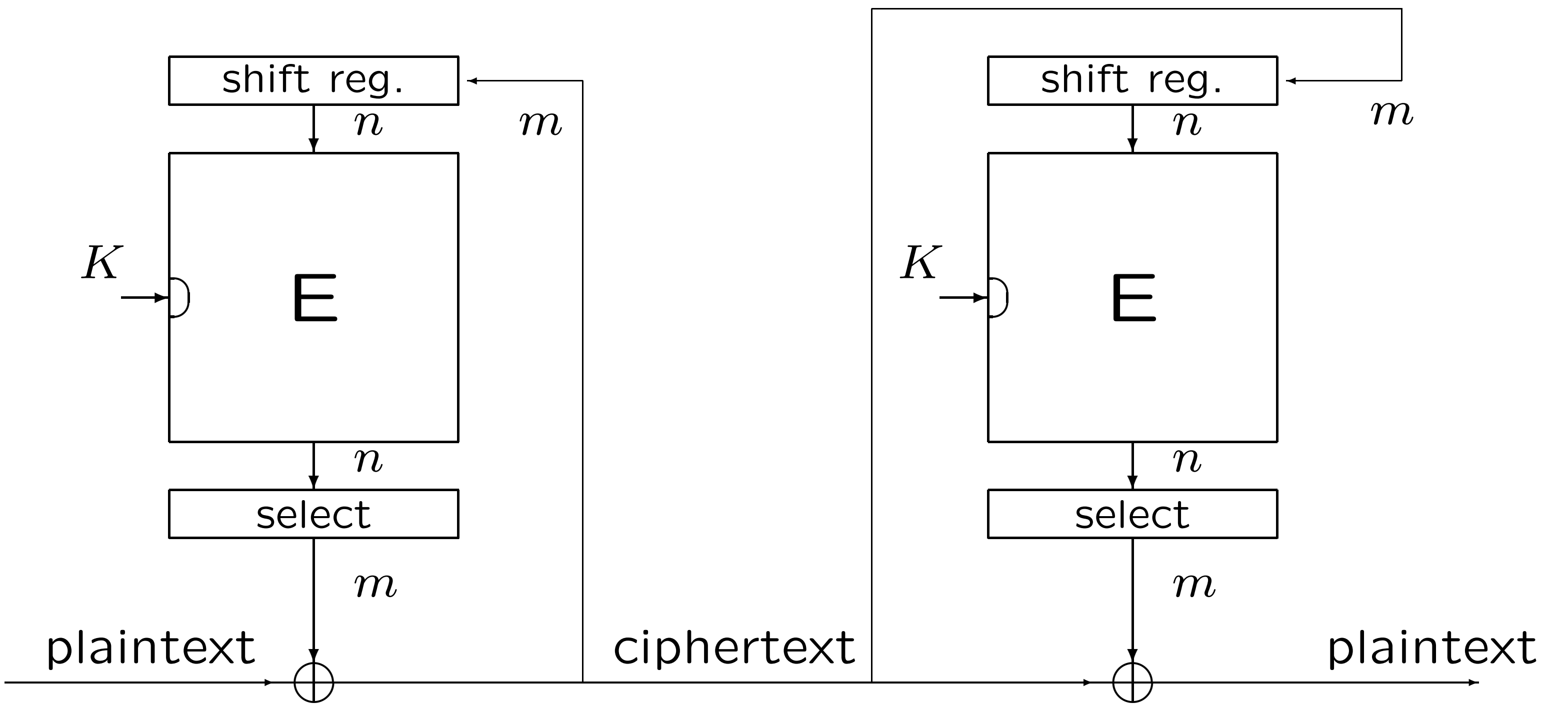
Modes Of Encryption (2/4, B)

Output Feed Back (OFB): properties

- no linking between subsequent blocks
- one-time IV necessary; otherwise insecure
- uses only encryption
- if $m < n$: more effort per bit
- key stream independent of plaintext: can be precomputed
- no error propagation: errors are only copied

Modes Of Encryption (3/4, A)

CFB: Self-synchronising stream cipher



n = block length, m = # of feedback bits

Modes Of Encryption (3/4, B)

Cipher Feed Bak (CFB): properties

- ciphertext depends on all previous plaintext blocks
- one-time IV hides patterns and repetitions
- uses only encryption
- if $m < n$: more effort per bit
- error propagation: error copied and propagated block

Modes Of Encryption (4/4, A)

Modes Of Encryption (4/4, B)

Cipher Block Chaining (CBC): properties

- ciphertext depends on all previous plaintext
- one-time IV hides patterns and repetitions
- error propagation: expansion in 1 block, copied into next block
- decryption of 1 block requires ciphertext of previous and current block

Padding before encryption

- **What?** Extra bits are appended to the plaintext
- **Why?** To assure that the plaintext has a fixed format and to make its length a multiple of the block size
- **How?** A message of n bytes is converted into a message of $n + (b - (n \bmod b))$ bytes:
  - original plaintext consists of n bytes,
  - encryption algorithm's blocklength equals b,
  - the value of the $b - (n \bmod b)$ padding bytes equals $b - (n \bmod b)$
- **Examples:** Given $b = 16$, and $n = 15$, a single byte is padded with the value 1. If $n = 4$, twelve bytes are padded with the value 12, and if $n = 32$, 16 bytes are padded with the value 16.

Secret Key $\leftrightarrow$ Public Key

- key agreement

  How can 2 people who have never met share a key which is only known to these 2 people

- digital signature

  How can one be sure that a message comes from the sender who claims to have produced that message?

Problem 1: Key-Agreement (1/4)

Diffie-Hellman Key Agreement Protocol
($f(X, Z)$: commutative one way function)

$$
\begin{array}{lcr}
\mathit{Alice} & & \mathit{Bob} \\
Y_A = f(X_A, Z) & & \\
& \xrightarrow{Y_A} & \\
& & Y_B = f(X_B, Z) \\
& \xleftarrow{Y_B} & \\
K_{AB} = f(X_A, Y_B) = f(X_A, f(X_B, Z)) & & K_{BA} = f(X_B, f(X_A, Z))
\end{array}
$$

Key-Agreement (2/4)

One Way functions

$$f : X \longrightarrow Y;\, x \mapsto f(x) = y \text{ is a one way function } \Longleftrightarrow$$

- $f(x)$, $\forall x \in X$ is easy to compute
- given $y \in Y$, finding an $x \in X$, with $f(x) = y$ is a hard problem

We will use modular exponentiation as one-way function

Key-Agreement (3/4)

Modular Exponentiation

- given α and a prime p with $\alpha \in [1, p-1]$
- $w = \alpha^x \bmod p$ can be computed efficiently (square and multiply)

Inverse operation (*discrete logarithm*)

- given α, p and w, find x such that

$$\alpha^x \bmod p \equiv w$$

Key-Agreement (4/4)

Diffie-Hellman – example

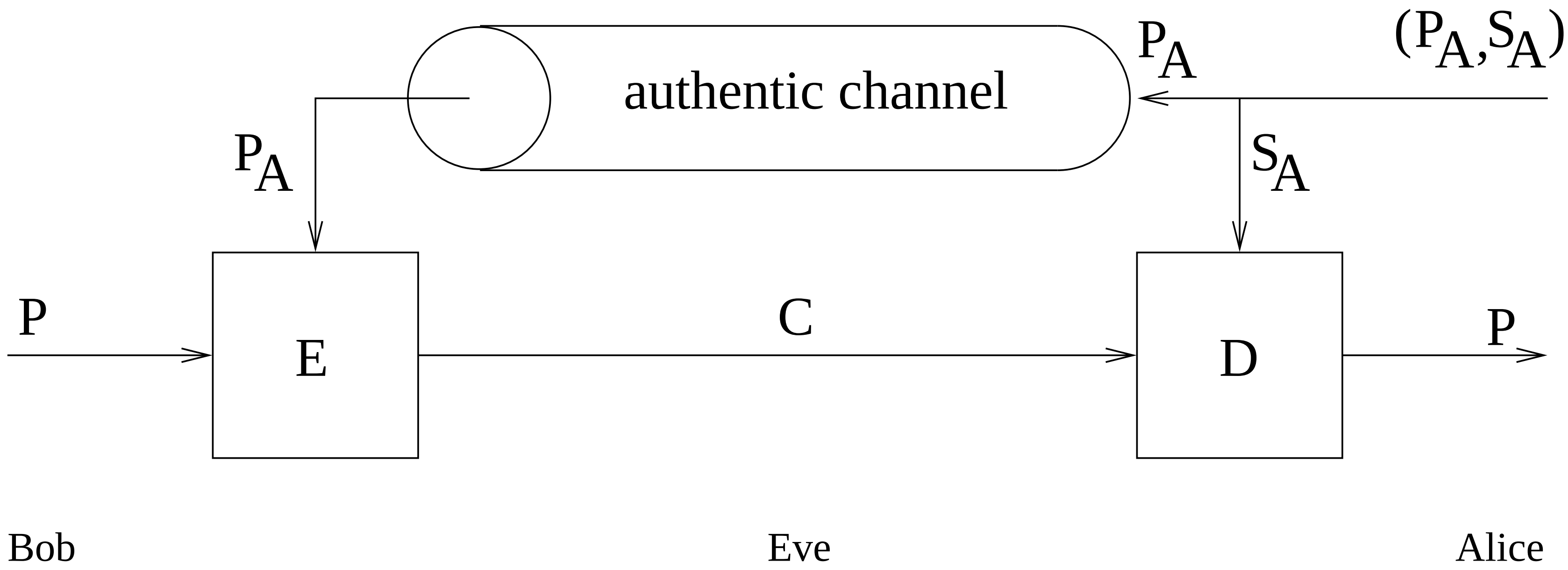
- $p = 37$: the integers from 0 to 36 form a field with $+$ and $\times$ mod 37
- $\alpha = 2$ is a generator of the non-zero elements: powers of 2 generate all non-zero elements: $2^0 = 1$, $2^1 = 2$, $2^3 = 8$, $2^4 = 16$, $2^5 = 32$, $2^6 = 27$, $2^7 = 17$, ..., $2^{36} = 1$
- $X_A = 10 \Rightarrow Y_A = 2^{10} \bmod 37 = 25$
- $X_B = 13 \Rightarrow Y_B = 2^{13} \bmod 37 = 15$
- $K_{AB} = (Y_B)^{X_A} = 15^{10} \bmod 37 = 15^{8+2} \bmod 37 = 7 \times 3 \bmod 37 = 21$
- $K_{BA} = (Y_A)^{X_B} = 25^{13} \bmod 37 = 25^{8+4+1} \bmod 37 = 34 \times 16 \times 25 \bmod 37 = 21$
- $K_{AB} = K_{BA} = 21$

Secret key derivation

- **Why?** Two different keys are necessary: encryption key and MAC key
- **How?** Two steps:
  - First Diffie-Hellman/Station to Station agreement on shared secret
  - Derivation of two different keys, e.g., using a hash function, the shared secret and for each key a different string
- **Advantage?** Ciphertext blocks are different from intermediate results during MAC computation

Problem 2: Public-key cryptography (1/3)

(trapdoor one-way functions)



Public-key cryptography (2/3)

RSA public-key algorithm
trapdoor one-way function:

- given x: "easy" to compute $f(x)$
- given $f(x)$: "hard" to compute x
- given $f(x)$ and the trapdoor information: finding x is "easy"

given two large primes p and q and a public key (e, n)
$n = p \times q$ (factoring n is hard)

$$f(x) = x^e \bmod n \text{ is a trapdoor one-way function}$$

trapdoor information (p, q) allows to find a private key (d, n) such that

$$(x^e)^d = (x^e)^{1/e} = x \bmod n$$

Public-key cryptography (3/3)

RSA public-key algorithm (2): detail
key generation:
choose two primes p and q
$n = p \times q$, $\phi(n) = (p-1)(q-1)$
choose e prime w.r.t. $\phi(n)$
compute $d = e^{-1} \bmod \phi(n)$

$$\text{public key} = (e, n)$$
$$\text{private key} = (d, n) \text{ or } (p, q)$$

encrytion: $c = m^e \bmod n$
decrytion: $m = c^d \bmod n$

Data Authentication (1/2)

What

- data integrity: authentication of content
- data-origin authentication: authentication of sender
- data destination authentication
- time and sequence

Application Areas

- financial transactions
- electronic mail
- computer viruses

Data Authentication (2/2)

two levels

1. **symmetric authentication:**

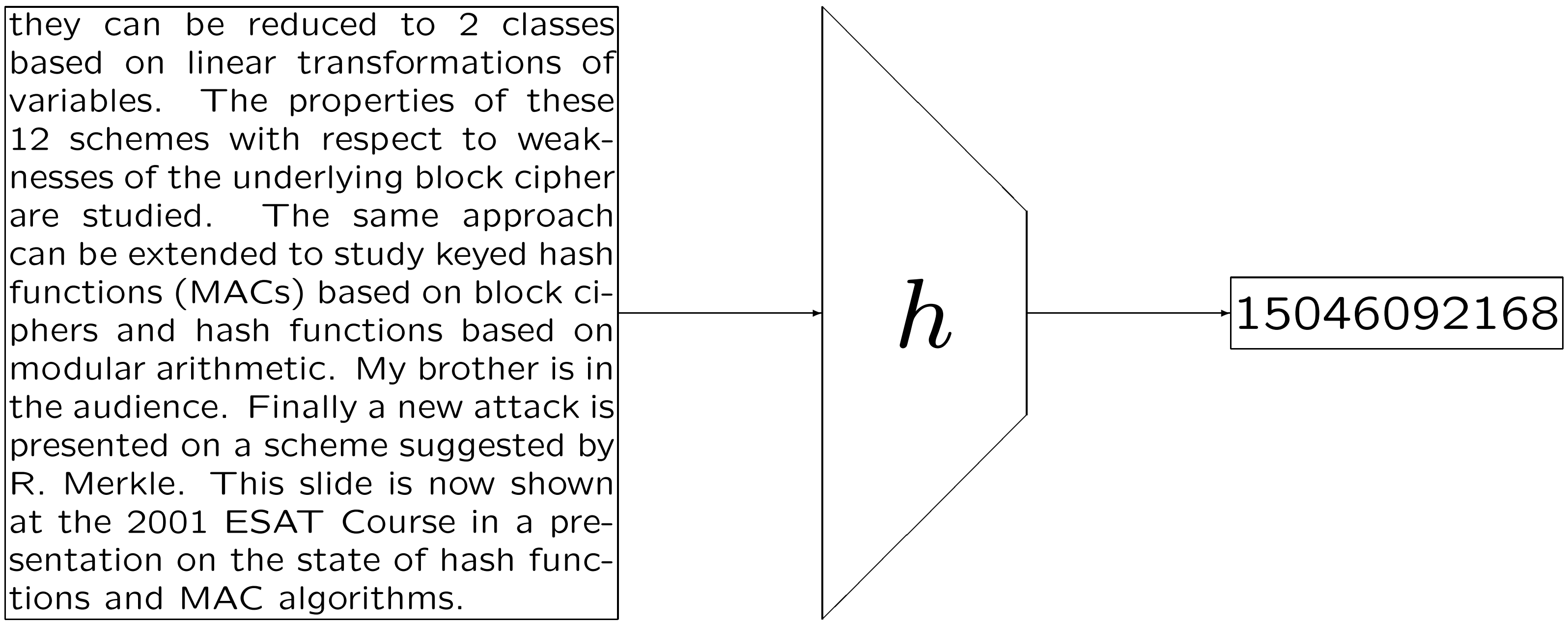
   within group of persons who trust each other

2. **asymmetric authentication** (digital signature):

   - can be verified by third party
   - depends on knowledge of the signer's private key

Hash function(1/2)

Definition - hash function



Hash function(2/2)

MDC (Modification Dedection Code):
Hash function without key

- compression to fixed length
- preimage resistant:
  hard to find an X' with $h(X') = Y$, given $Y = h(X)$
- 2nd preimage resistant:
  hard to find $X' \neq X$ with $h(X') = h(X)$, given X and $h(X)$
- collision resistant:
  hard to find X', $X' \neq X$ with $h(X') = h(X)$

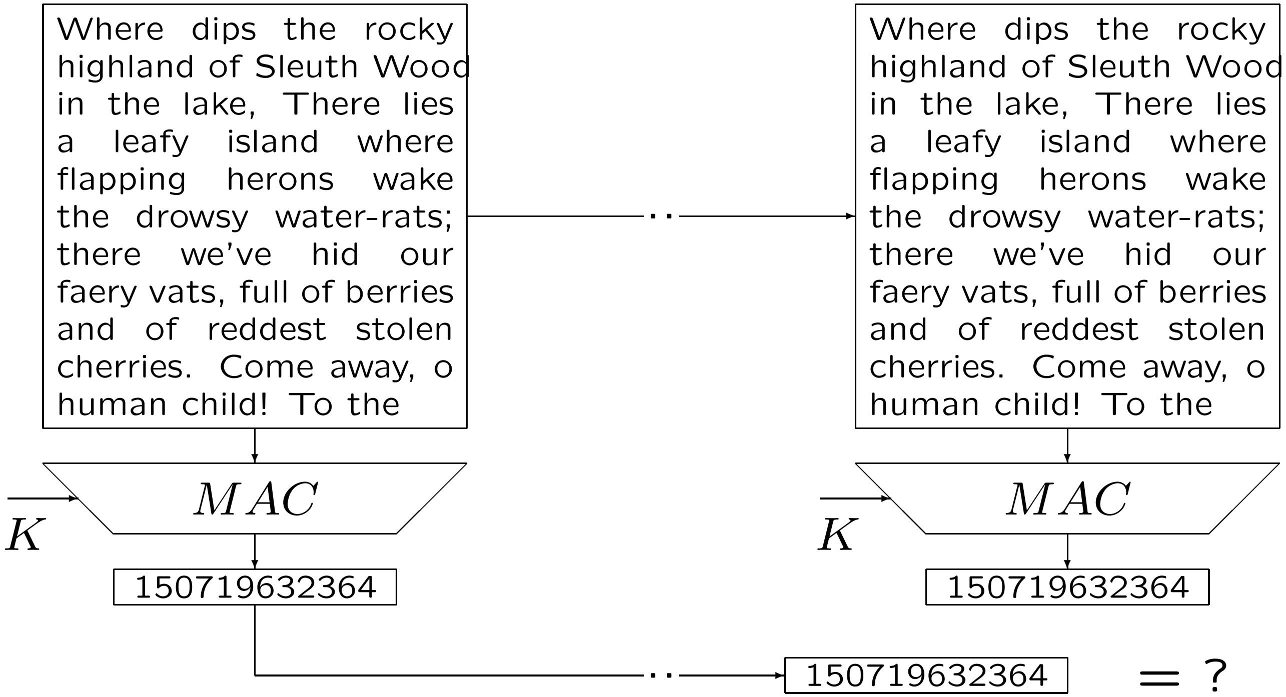
Message Authentication Codes (1/3)

MAC (Message Authentication Code):
Hash function with secret key

- hard to produce a forgery
- can only be generated and verified by someone who secret MAC-key
- do not use the same key for MAC and for encryption

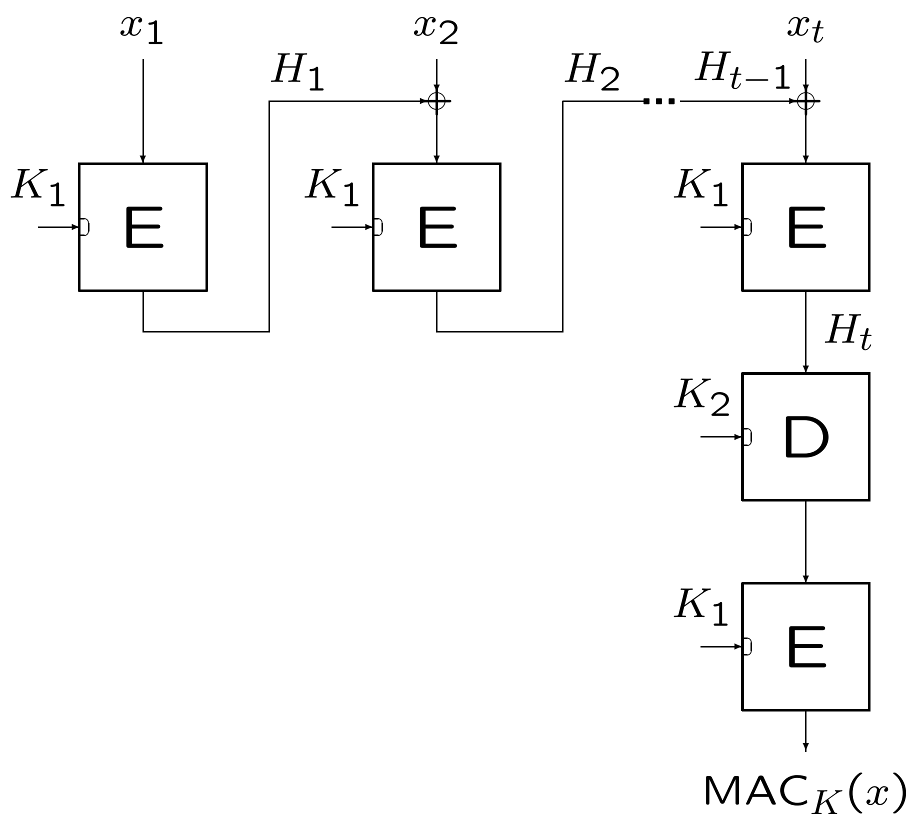
Message Authentication Codes (2/3)

MAC = hash function with secret key

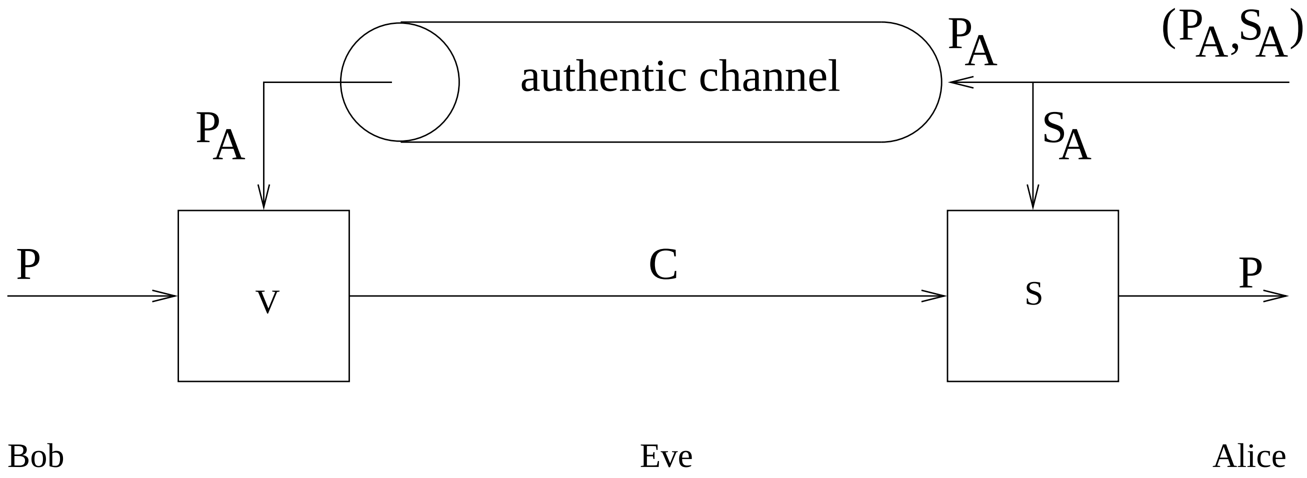


Message Authentication Codes (3/3)

MAC based on block cipher: retail MAC



Digital Signature



signature generation uses signer's private key
signature verification uses signer's public verification key

RSA-signatures with message recovery

For short messages (up to the modulus length - 4 bytes)

- the signer
  - adds redundancy to the message h: $h' = 0x00, 0x01, 0xFF, \cdots, 0xFF, 0x00,$
  - computes $c = (h')^d \bmod n$ using his private key (d, n)
- the verifier
  - receives a digital signature c'
  - computes $h" = (c')^e \bmod n$ using the public key (e, n)
  - checks whether the redundancy in $h"$ equals $0x00, 0x01, 0xFF, \cdots, 0xFF, 0x$
  - strips the redundancy, recovering h
  - believes that h comes from the signer

RSA-signature with appendix

For large messages (longer than the modulus length - 4 bytes)

- the signer
  - computes the hash value h on a message m
  - applies the previous signing method to compute the digital signature c on h
  - the signer sends m together with the signature c to the verifier
- the verifier
  - receives a message m' and a digital signature c'
  - recovers h from the previous verification method, and
  - computes the hash value h''' on the message m', and
  - believes that m' comes from the signer if h equals h'''

Station to Station (STS) Protocol

Alice		*Bob*
generate secret x	$\alpha^x \bmod p$	
	$\longrightarrow$	generate secret y
	$\alpha^y \bmod p, E_k(S_B(\alpha^y, \alpha^x))$	compute secret key k
compute secret key k	$\longleftarrow$	
validate signature	$E_k(S_A(\alpha^x, \alpha^y))$	
	$\longrightarrow$	validate signature

STS Protocol: properties

- both parties know who the other party is
- both parties know that the other party is active (alive)
- both parties are sure that the other party has computed a shared k
- anonymity (if no certificates are sent in the clear)
- perfect forward secrecy: even if the long term signature keys are compromised, one learns nothing about previous session keys
- leakage of k does not compromise long-term keys