T-79.4501
Cryptography and Data Security

Summary

Autumn 2006

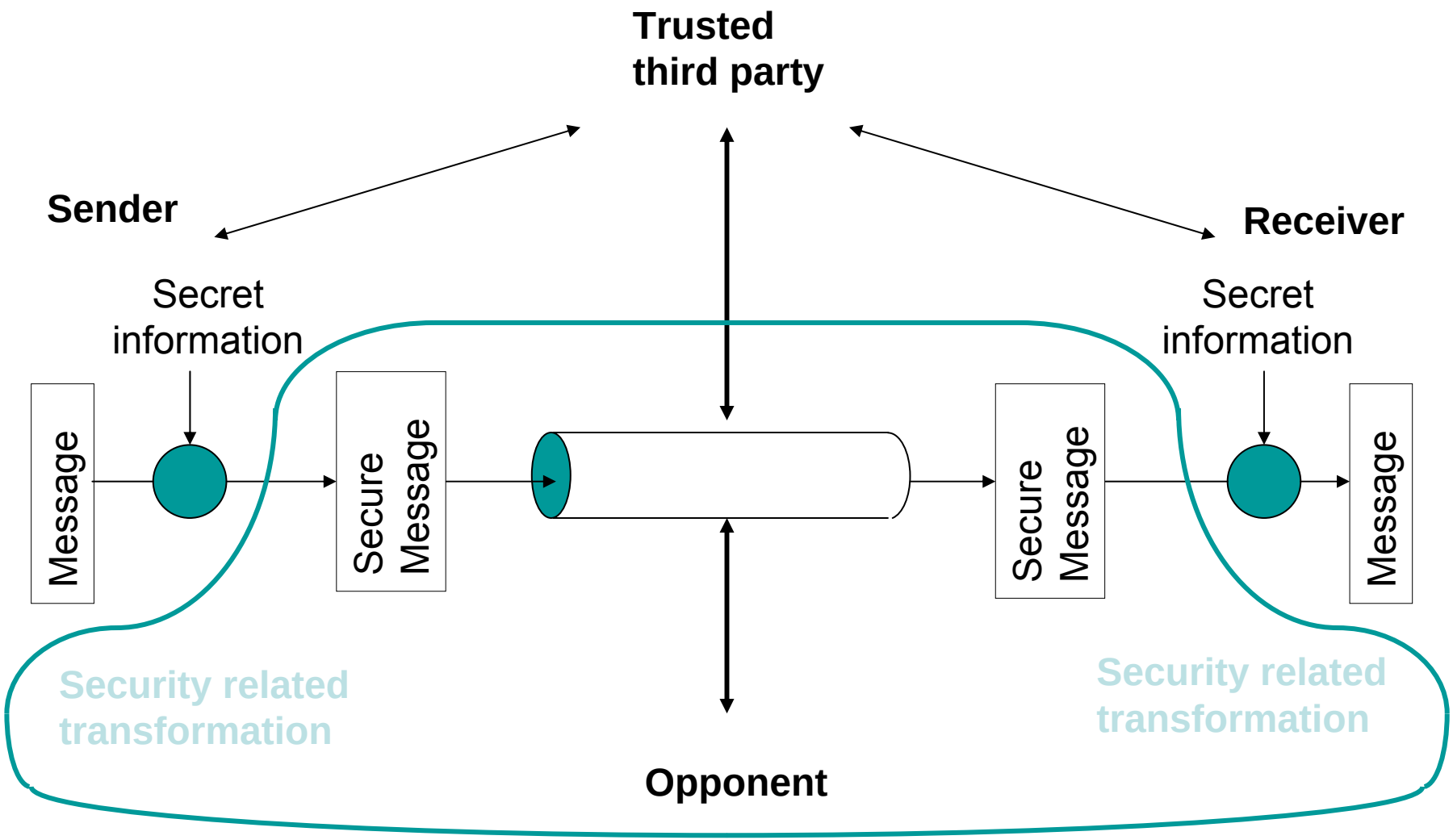
Course Contents (1-6)

- Introduction to data security
- Classical cryptosystems
- Introduction to modern cryptography
- Polynomial arithmetic, Euclidean algorithm; Block ciphers: DES, IDEA, AES
- Stream ciphers: RC4, and other examples
- Block cipher modes of operation
- Hash-functions and MACs
- Mathematical tools: Modular arithmetic, Chinese Remainder Theorem, Euler's totient function, Euler's theorem

Course Contents (7-12)

- Public key cryptosystems: RSA
- Prime number generation
- Public key cryptosystems: Diffie-Hellman, El Gamal, DSS
- Authentication and Digital signatures
- Random number generation and Key management
- Example: Bluetooth security
- Authentication and key agreement protocols in practise: PGP, SSL/TLS, IPSEC, IKEv2 and EAP

Model for network security



Threat model

- How to define security (needs) in practise:
  - First perform threat analysis: cababilities of an attacker, possible attack scenarios
  - Security can then be defined in terms of combatting the perceived threats
  - Not all threats are worth of combatting, absolute security cannot be achieved
- Dolev-Yao attacker model against cryptographic protocols:

  An attacker
  - is a legitimate user of the network, and hence able to correspond with any other user
  - can send messages to another user by impersonating any other user
  - can receive messages intended to any other user

Computer and Communication Layers Security

System level security

"The system is as strong as its weakest link."

Application security

e.g. banking applications over Internet use security mechanisms which are tailored to meet their specific requirements.

Protocol level security

well-defined communication steps in certain well-defined order.

Operating system security

the behaviour of all elements in a network depends on the correct functionality of the operating system that controls them.

Platform security

properties of the computing platform, e.g. protected memory space.

Security primitives

these are the basic building blocks, e.g. cryptographic algorithms.

Example: GSM Security

Main security technical features

- Authentication of the user
  - correct billing
- Encryption of communication over the radio interface
  - confidentiality of user and control data
  - call integrity ($\Rightarrow$ correct billing)
- Use of temporary identities
  - user privacy
  - location privacy

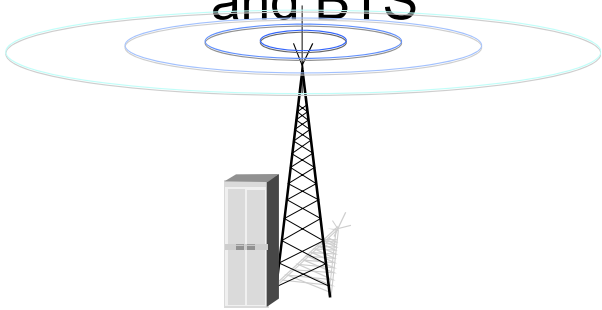
GSM Authentication

MS (SIM)

IMSI, **Ki**

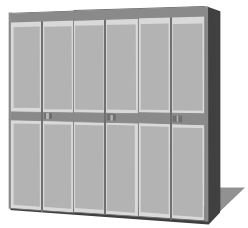
VLR

and BTS



HLR

{{IMSI,**Ki**}}



IMSI / TMSI →

IMSI →

← RAND

← RAND, XRES, **Kc**

Kc

SRES →

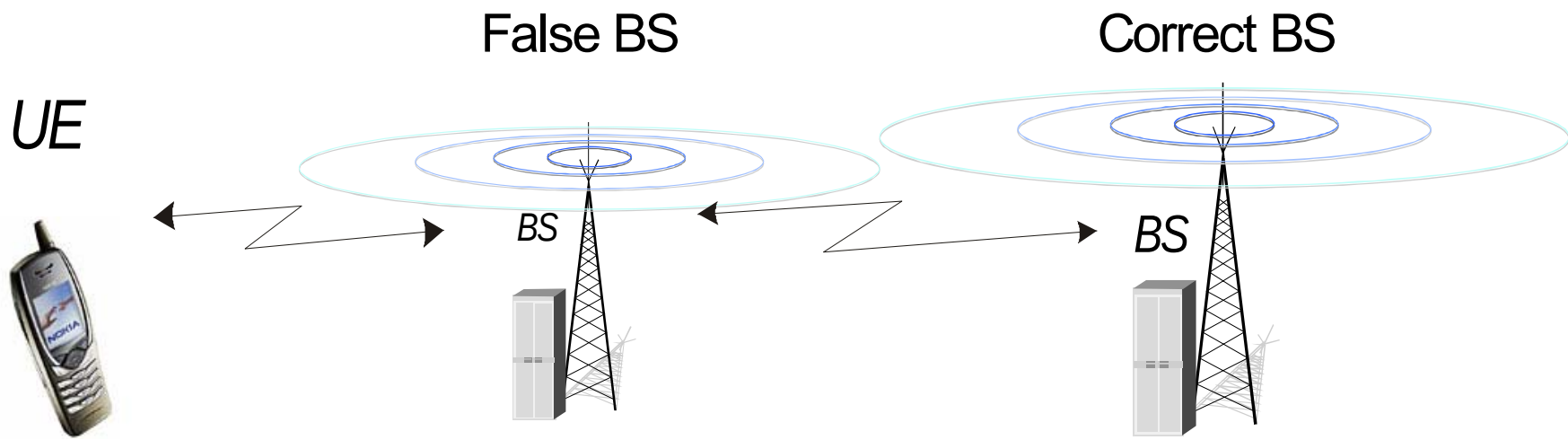
SRES=XRES ?

← encrypted TMSI

Criticism

- Active attacks possible
  - It is possible with suitable equipment to masquerade as a legitimate network element and/or legitimate user terminal
- Missing or weak protection between networks
  - control data, e.g. *keys* used for radio interface ciphering, are sometimes sent unprotected between different networks
- Secret design
  - some essential parts of the security architecture were kept secret, e.g. the cryptographic algorithms

Active Attack



Lessons learnt

- Use independent keys for different algorithms so that a key captured from one broken algorithm cannot be used to compromise security of another algorithm.
- Use strong crypto only
- Active man-in-the-middle attacks in wireless communication must be taken seriously
- Amendments to existing security system extremely difficult to implement:
  - updates to existing devices
  - backwards compatibility
  - version negotiation hard to protect (bidding-down attacks)

Monoalphabetic substitution

Alphabets

Plain: `abcdefghijklmnopqrstuvwxyz`

Cipher: `ABCDEFGHIJKLMNOPQRSTUVWXYZ`

Key = permutation of the 26 characters

Size of key space 26! $\cong 4 \times 10^{26}$

Not possible to do exhaustive search over the key space

Cryptanalysis is based on statistical properties of the plaintext

Relative Frequency of Letters in English

A	8.167
B	1.492
C	2.782
D	4.253
E	12.702
F	2.228
G	2.015
H	6.094
I	6.996
J	0.153
K	0.772
L	4.025
M	2.406

N	6.749
O	7.507
P	1.929
Q	0.095
R	5.987
S	6.327
T	9.056
U	2.758
V	0.978
W	2.360
X	0.150
Y	1.974
Z	0.074

B L U E

· · - · F

F L A G

· - · · L

I S

· - A

D O T

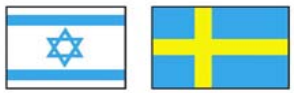
- - · G

R E D

· · · S

F L A G

· · · · H



I S

· · I

D A S H

· - - · P

Playfair Cipher

Key: MONARCHY

is put first in a 5x5 matrix, which is then filled out with the remaining letters of the alphabet (i = j)

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

The encryption rules

Plaintext formatting

`oo -> oxo`

Same row or column

`ar -> RM`

`mu -> CM`

Regular case

`hs -> BP`

`ea -> IM`

Vigénère cipher: Kasiski's method

- Many strings of characters repeat themselves in natural languages.
- Assume the interval between occurence of a string is a multiple of the length of the period.
- Then a repetition of a character string of the same length occurs in the ciphertext.
- By detecting repetitions of strings in the ciphertext one can find the period as the greatest common divisor (GCD) of the repetition intervals
- Their may be false repetitions. The longer the repeating string the more significant it is. Sometimes only strings of length ≥ 3 are considered.

One Time Pad

- Claude Shannon laid (1949) the information theoretic fundamentals of secrecy systems.
- Shannon's pessimistic bound: For perfect secrecy, the length of the key is at least as large as the length of the plaintext.
- An example of a cipher which achieves perfect secrecy is the One Time Pad

$$c_i = (p_i + k_i) \bmod 26$$

where a new fresh random key $k_1\, k_2\, k_3 \ldots k_i \ldots$ is chosen for each new plaintext $p_1\, p_2\, p_3 \ldots p_i \ldots$

- Practical ciphers do not provide perfect secrecy

Primitives and protocols

- Cryptographic primitives and functions are used as building blocks for cryptographic protocols
- For example,
  - A stream cipher primitive is the basic building block for an encryption protocol
  - A message authentication code is the basic building block for an authentication protocol

Different design approaches

- information theoretic
  - security is measured in terms of probabilities
- complexity theoretic
  - security measured in terms of computational and memory requirements
- quantum cryptology
- system based
  - security measured in terms of used cryptanalysis methods

Different assumptions:

- capabilities of an opponent
- cryptanalytic success
- definition of security (e.g., unconditional security, computational security)

Man-made vs. Math-made

Symmetric primitives

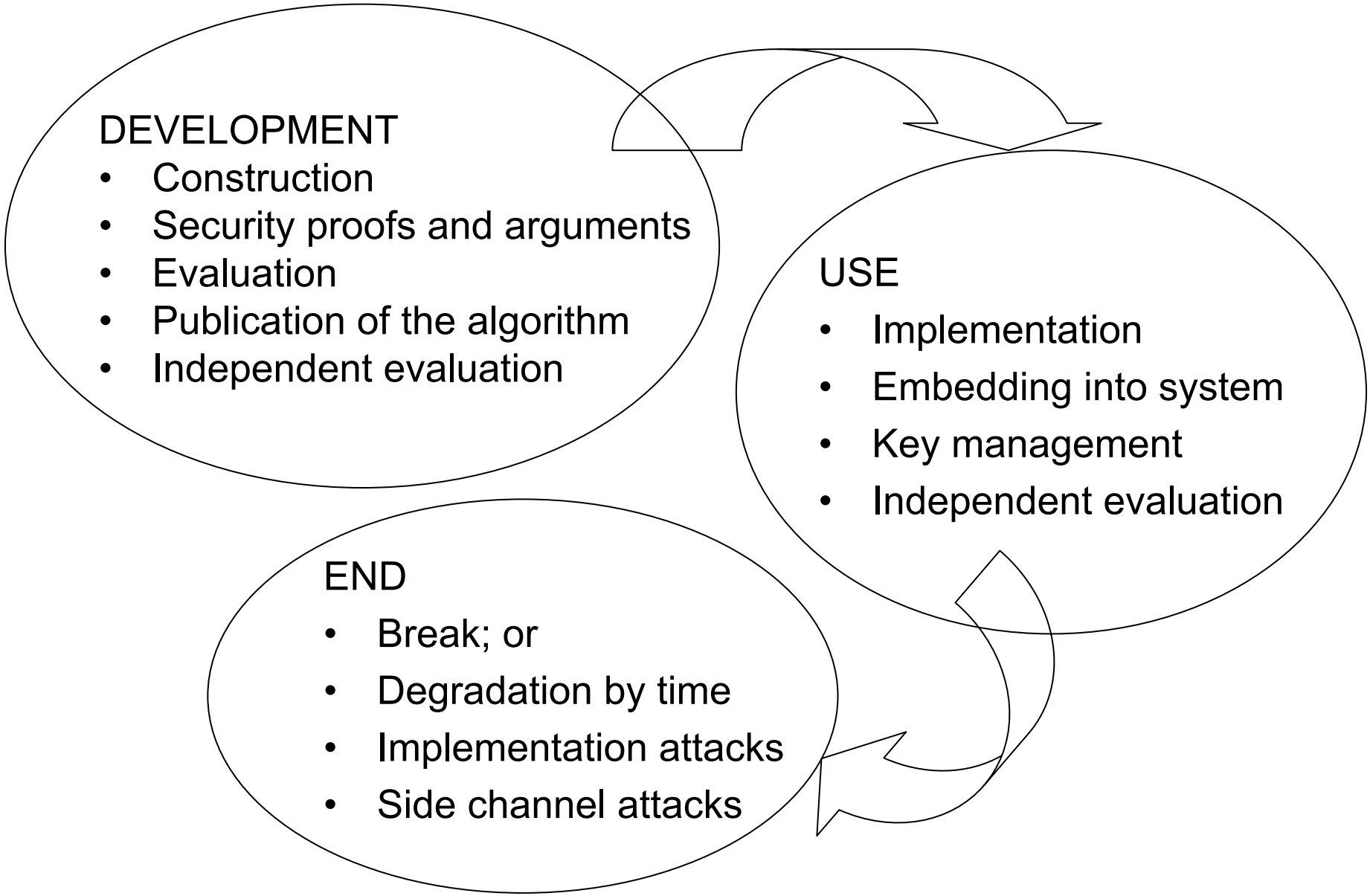
- are based on man-made constructions;
- are fast and easy to implement in software and/or hardware:
- use short keys

Asymmetric (public key primitives)

- are based on mathematical construction and their security is derived from infeasibility of some computationally hard problem.
- are slow and difficult to implement (both in software and hardware)
- have long keys and parameters

It would be possible to construct symmetric primitives based on mathematics, but they are not used in practise because they are not efficient compared to symmetric primitives

Life Cycle of a Cryptographic Algorithm



Block ciphers

Sender: (Message , Secret key) → Ciphertext

Receiver: (Ciphertext, Secret key) → Message

Confidentiality primitive

- Threat: retrieve the plaintext from the ciphertext without the knowledge of the key.
- Security goal: protect against this threat.

Plaintext P: strings of bits of fixed length n

Ciphertext C: strings of bits of the same length n

Key K: string of bits of fixed length k

Encryption transformations: For each fixed key the encryption operation E_K is one-to-one (invertible) function from the set of plaintexts to the set of ciphertext. That is, there exist an inverse transformation, decryption transformation D_K such that for each P and K we have: $D_K(E_K(P)) = P$

Birthday paradox

- No paradox, just somewhat counterintuitive
- Assume that numbers are randomly (with equal probability) picked with replacement from a set of N different numbers.
- Question: How many numbers must be picked until the probability of getting at least one number twice is at least 0.5 ?
- Answer: Approximately $1.17\sqrt{N}$
- See Stallings Appendix 11A, p. 340.

Birthday paradox: Derivation

Let k be the number drawn from the set of N elements with replacement. Then

P = Pr[at least one match] = 1 – Pr[no match] =

$$1-1\cdot\left(1-\frac{1}{N}\right)\cdot\left(1-\frac{2}{N}\right)\cdots\left(1-\frac{k-1}{N}\right)=1-\prod_{i=1}^{k-1}\left(1-\frac{i}{N}\right)$$

Since N is large, we can approximate, for all $i = 1,\ldots, k\text{-}1$

$\left(1-\frac{i}{N}\right)\approx e^{-\frac{i}{N}}$ and get:

$$P\approx 1-\prod_{i=1}^{k-1}e^{-\frac{i}{N}}=1-e^{-\frac{1}{N}\sum_{i=1}^{k}i}=1-e^{-\frac{k(k-1)}{2N}}\approx 1-e^{-\frac{k^2}{2N}}$$

Then $P\approx\frac{1}{2}=e^{-\ln 2}$ if $k^2\approx 2N\ln 2$, that is, $k\approx\sqrt{2N\ln 2}\approx 1.17\sqrt{N}$ as desired.

Generic attack on block ciphers

Let n be the block length in bits. Then if the block cipher encryption operation is used about $2^{n/2}$ times with the same key on any randomly generated data as plaintext, then by Birthday Paradox, the probability of having two equal ciphertexts is about ½. Then one knows that the two corresponding input data are equal.

Stream ciphers

- Stream ciphers are generally faster than block ciphers, especially when implemented in hardware.
- Stream ciphers have less hardware complexity.
- Stream ciphers can be adapted to process the plaintext bit by bit, or word by word, while block ciphers require buffering to accumulate the full plaintext block.
- Synchronous stream ciphers have no error propagation; encryption is done character by character with keys K_i that are independent of the data

$$C_i = E_{Ki}(P_i)$$

- Function E is simple, the function which computes the key sequence is complex
- Example: Vigenère cipher, One Time Pad

$$C_i = (P_i + K_i) \bmod 26$$

Stream cipher encryption

Sender:

Secret key → Key stream

(Key stream , Message) → Ciphertext

Receiver:

Secret key → Key stream

(Ciphertext, Key stream) → Message

Stream ciphers: Security

- Known plaintext gives known key stream. Chosen plaintext gives the same but nothing more.
- Chosen ciphertext attack may be a useful method for analysing a self-synchronising stream cipher.
- The attacker of a stream cipher may try to find one internal state of the stream cipher to obtain a functionally equivalent algorithm without knowing the key.
- Distinguishing a keystream sequence from a truly random sequence allows also the keystream to be predicted with some accuracy. Such attack is also called prediction attack.

Requirements:

- Long period
- A fixed initialisation value the stream cipher generates a different keystream for each key.

Polynomial Arithmetic

- Modular arithmetic with polynomials
- We limit to the case where polynomials have binary coefficients, that is, 1+1 = 0, and + is the same as -.

Example:

$$(x^2+x+1)(x^3+x+1)=$$

$$x^5+x^3+x^2+x^4+x^2+x+x^3+x+1=$$

$$x^5+x=x\cdot(x^4+1)=x\cdot x=x^2(\mathrm{mod}(x^4+x+1))$$

Computation $\mathrm{mod}(x^4+x+1)$ means that everywhere we take $x^4+x+1=0$, for example, we can take $x^4+1=x$.

Galois Field

Given a binary polynomial $f(x)$ of degree n, consider a set of binary polynomials with degree less than n. This set has 2^n polynomials. With polynomial arithmetic modulo $f(x)$ this set is a ring.

Fact: If $f(x)$ is irreducible, then this set with 2-ary (binary) polynomial arithmetic is a field denoted by $\mathrm{GF}(2^n)$.

In particular, every nonzero polynomial has a multiplicative inverse modulo $f(x)$. We can compute a multiplicative inverse of a polynomial using the Extended Euclidean Algorithm.

The next slide presents the Extended Euclidean Algorithm for integers. It works exactly the same way for polynomials.

Extended Euclidean Algorithm for integers and computing a modular inverse

Fact: Given two positive integers a and b there exist integers u and v such that

$$u \cdot a + v \cdot b = \gcd(a,b)$$

In particular, if $\gcd(a,b) = 1$, there exist positive integers u and v such that

$$u \cdot a = 1 \pmod{b}, \text{ and } v \cdot b = 1 \pmod{a}.$$

The integers u and v can be computed using the Extended Euclidean Algorithm, which iteratively finds integers r_i , u_i and v_i such that

$$r_0 = b, \;\; r_1 = a; \;\; u_0 = 0, \;\; u_1 = 1; \; v_0 = 1, \; v_1 = 0$$

and for $i = 2,3,\ldots$ we compute q_i such that

$r_{i-2} = q_i \cdot r_{i-1} + r_i$, where $0 \leq r_i < r_{i-1}$.

We set: $u_i = u_{i-2} - q_i \cdot u_{i-1}$ and $v_i = v_{i-2} - q_i \cdot v_{i-1}$. Then $r_i = u_i \cdot a + v_i \cdot b$.

Let n be the index for which $r_n > 0$ and $r_{n+1} = 0$. Then

$r_n = \gcd(a,b)$ and $u_n = u$ and $v_n = v$.

Extended Euclidean Algorithm: Example

gcd(595,408) = 17 = u×595 + v×408

i	q_i	r_i	u_i	v_i
0	-	595	1	0
1	-	408	0	1
2	1	187	1	-1
3	2	34	-2	3
4	5	17	11	-16

Extended Euclidean Algorithm for polynomials

Example

Example: Compute the multiplicative inverse of x^2 modulo x^4+x+1

i	q_i	r_i	u_i	v_i
0		$x^4 + x + 1$	0	1
1		x^2	1	0
2	x^2	$x + 1$	x^2	1
3	x	x	$x^3 + 1$	x
4	1	1	$x^3 + x^2 + 1$	$x + 1$

Extended Euclidean Algorithm for polynomials

Example cont'd

So we get

$$u_4 \cdot x^2 + v_4 \cdot (x^4 + x + 1) = (x^3 + x^2 + 1)x^2 + (x + 1)(x^4 + x + 1) = 1 = r_4$$

from where the multiplicative inverse of x^2 mod $x^4 + x + 1$ is equal to $x^3 + x^2 + 1$.

Motivation for polynomial arithmetic:

- uses all n-bit numbers (not just those less than some prime p)
- provides uniform distribution of the multiplication result

Example: Modulo 2^3 arithmetic compared to $\mathrm{GF}(2^3)$ arithmetic (multiplication).

In $\mathrm{GF}(2^n)$ arithmetic, we identify polynomials of degree less than n: $a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \cdots + a_2x^2 + a_1x + a_0$

with bit strings of length n : $(a_{n-1}, a_{n-2}, \ldots, a_1, a_0)$

and further with integers less than 2^n:

$$a_{n-1}2^{n-1} + a_{n-2}2^{n-2} + \cdots + a_2 2^2 + a_1 2 + a_0$$

Example: In $\mathrm{GF}(2^3)$ arithmetic with polynomial $x^3 + x + 1$ (see next slide) we get:

$4 \cdot 3 = (100) \cdot (011) = x^2 \cdot (x+1) = x^3 + x^2 = (x+1) + x^2 = x^2 + x + 1 = (111) = 7$

Multiplication tables

modulo 8 arithmetic

	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7
2	2	4	6	0	2	4	6
3	3	6	1	4	7	2	5
4	4	0	4	0	4	0	4
5	5	2	7	4	1	6	3
6	6	4	2	0	6	4	2
7	7	6	5	4	3	2	1

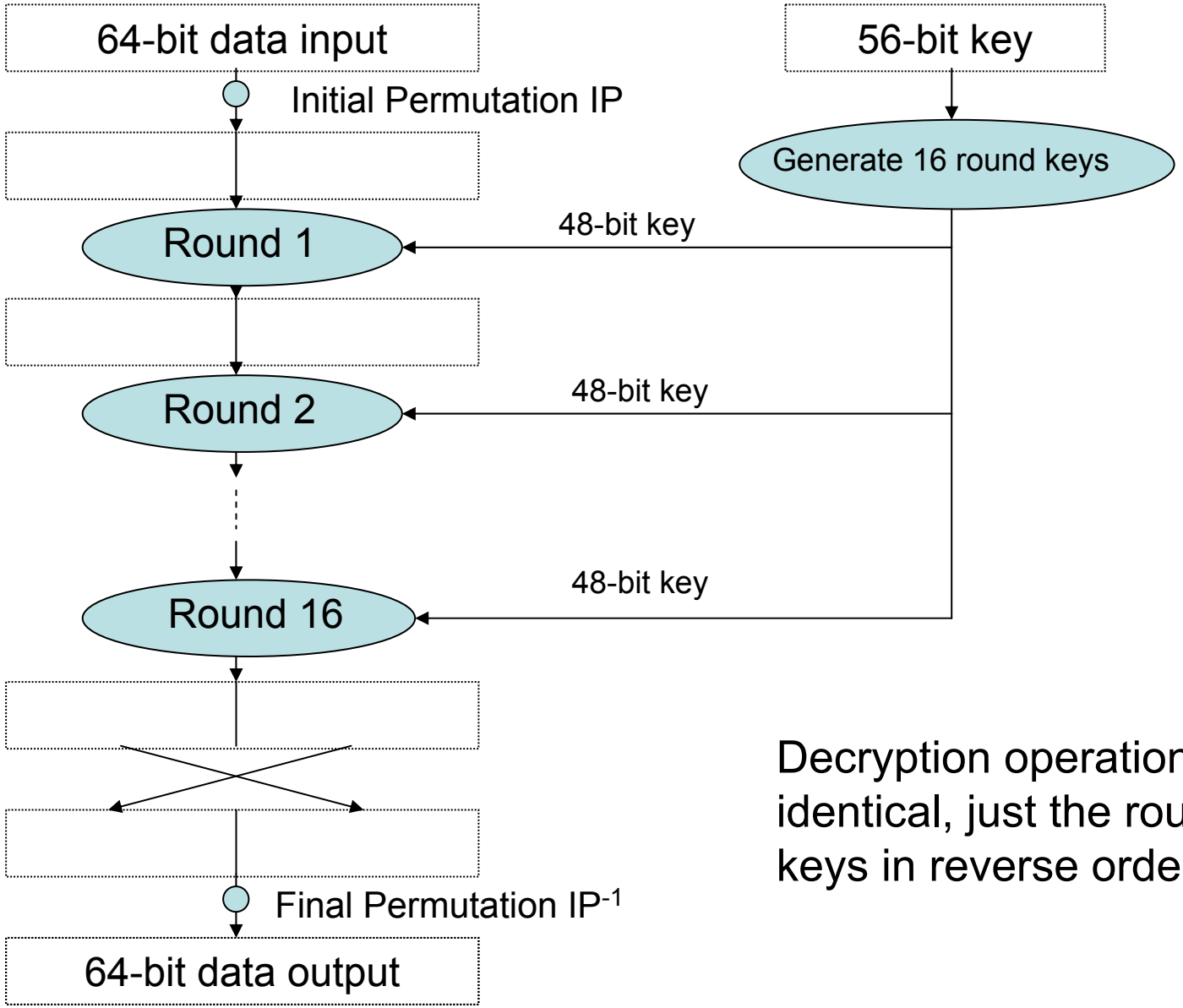
$GF(2^3)$ Polynomial arithmetic

	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7
2	2	4	6	3	1	7	6
3	3	6	5	7	4	1	2
4	4	3	7	6	2	5	1
5	5	1	4	2	7	3	6
6	6	7	1	5	3	2	4
7	7	5	2	1	6	4	3

Block ciphers, design principles

- The ultimate design goals of a block cipher:
  - use the secret key as efficiently as possible
  - no better attack than exhaustive key search
- Confusion and diffusion (Shannon)
- New design criteria are being discovered as response to new attacks.
- A state-of-the-art block cipher is constructed taking into account all known attacks and design principles.
- But no such block cipher can become provably secure, it may remain open to some new, unforeseen attacks.
- Common constructions with iterated round function
  - Substitution permutation network (SPN)
  - Feistel network

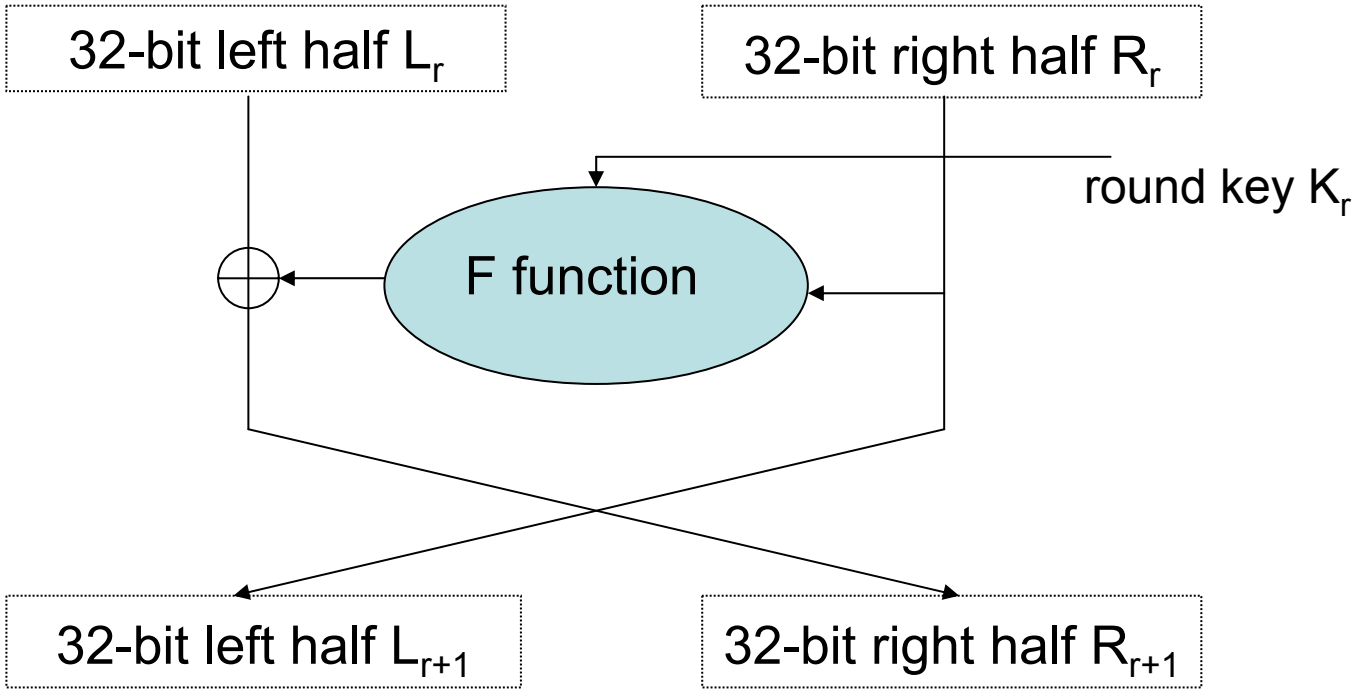
DES encryption operation overview



Decryption operation is identical, just the round keys in reverse order

DES round function

Round function is its own inverse (involution):

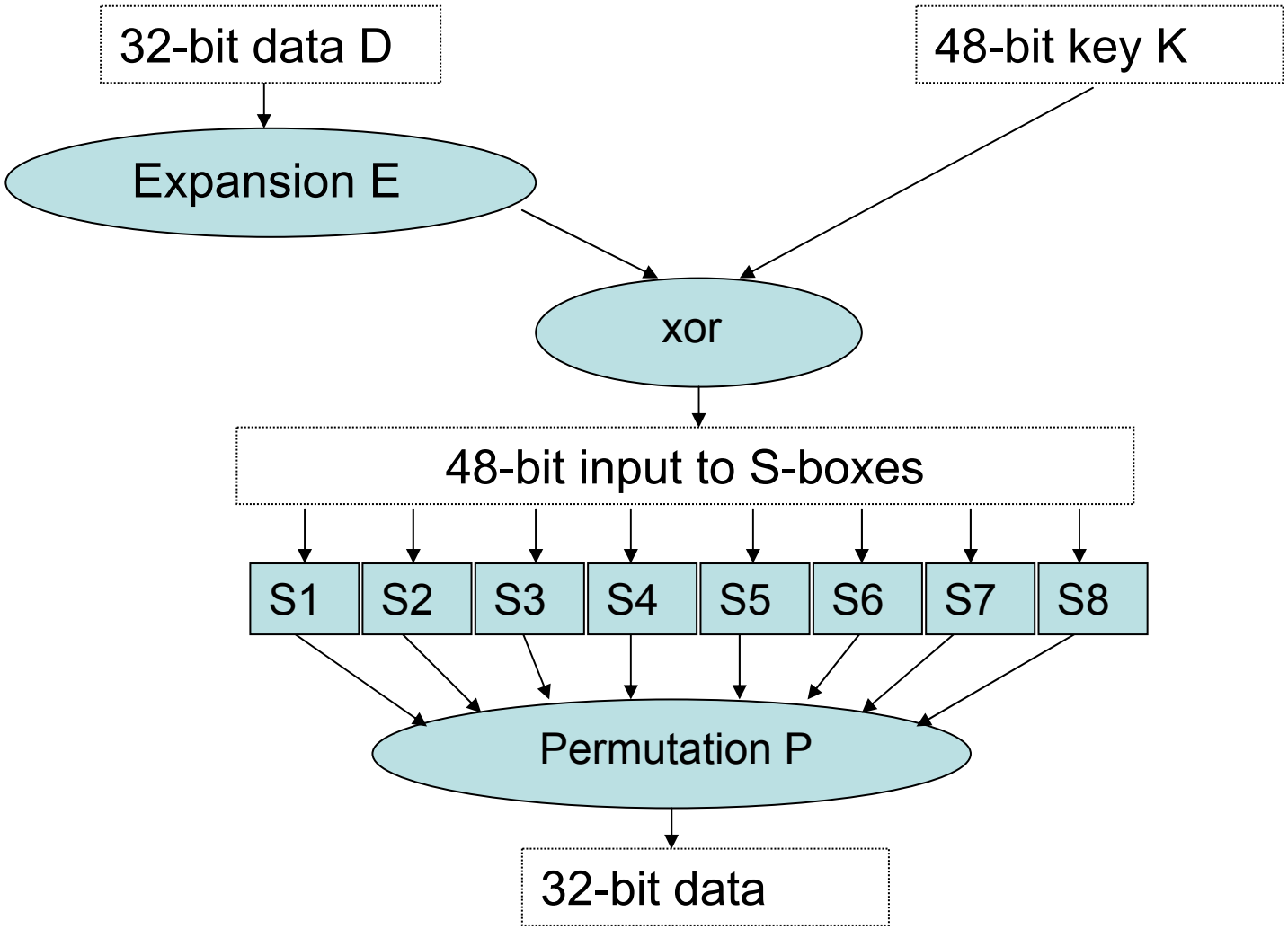


$$L_{r+1} = R_r$$

$$R_{r+1} = L_r \text{ xor } F(R_r, K_r)$$

The F-function of DES

F(D;K) = P(S(E(D) xor K)



The DES S-boxes

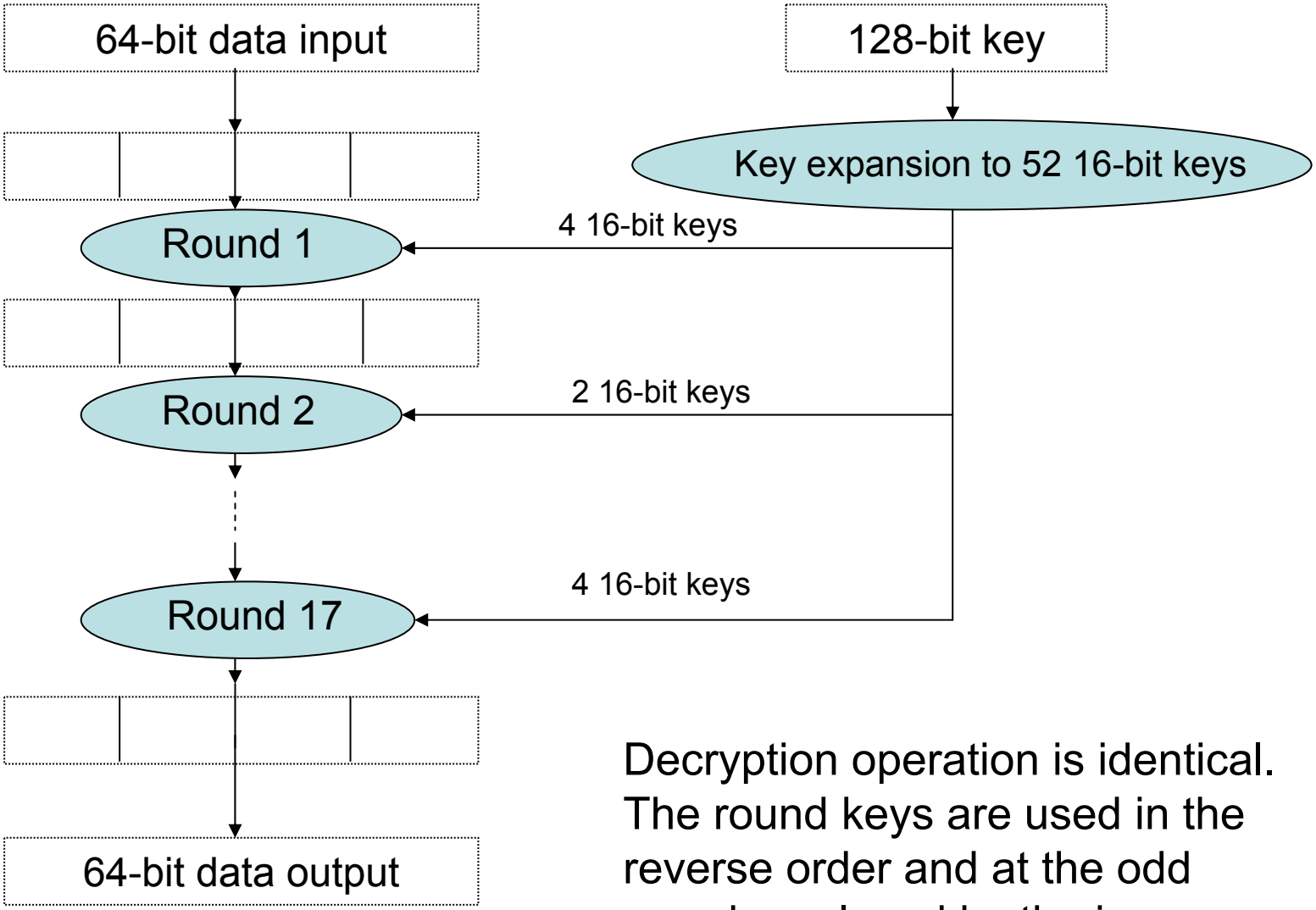
- Small 6-to-4-bit functions
- Given in tables with four rows and 16 columns
- Input data a1,a2,a3,a4,a5,a6
- The pair of bits a1,a6 point to a row in the S-box
- Given the row, the middle four bits point to a position from where the output data is taken.

Example: S-box S_4

7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

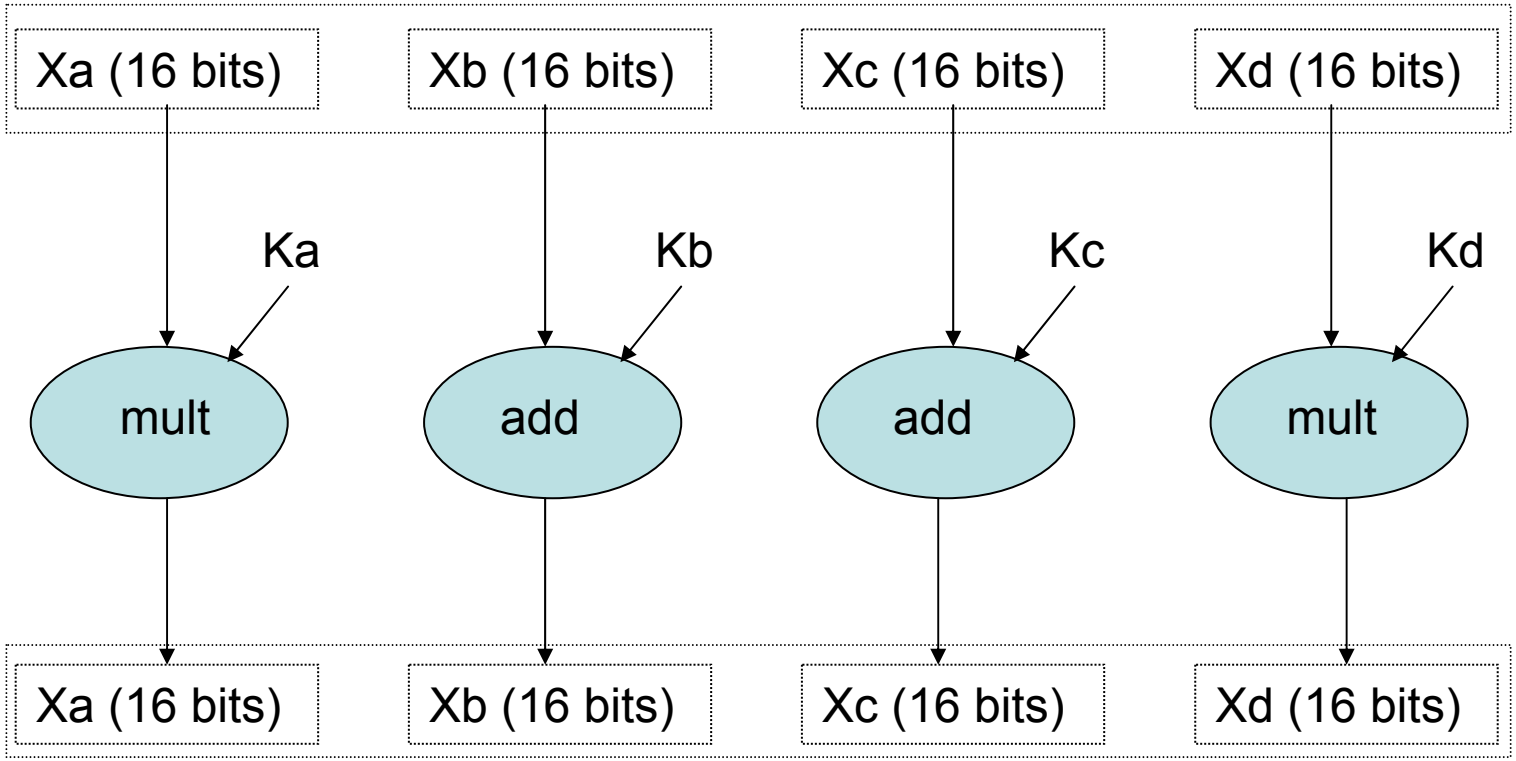
- S-boxes are the only source of nonlinearity in DES. Their nonlinearity properties are extensively studied.

IDEA encryption operation overview



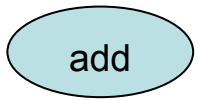
Decryption operation is identical. The round keys are used in the reverse order and at the odd rounds replaced by the inverse values.

One round of IDEA: odd round



Legend:

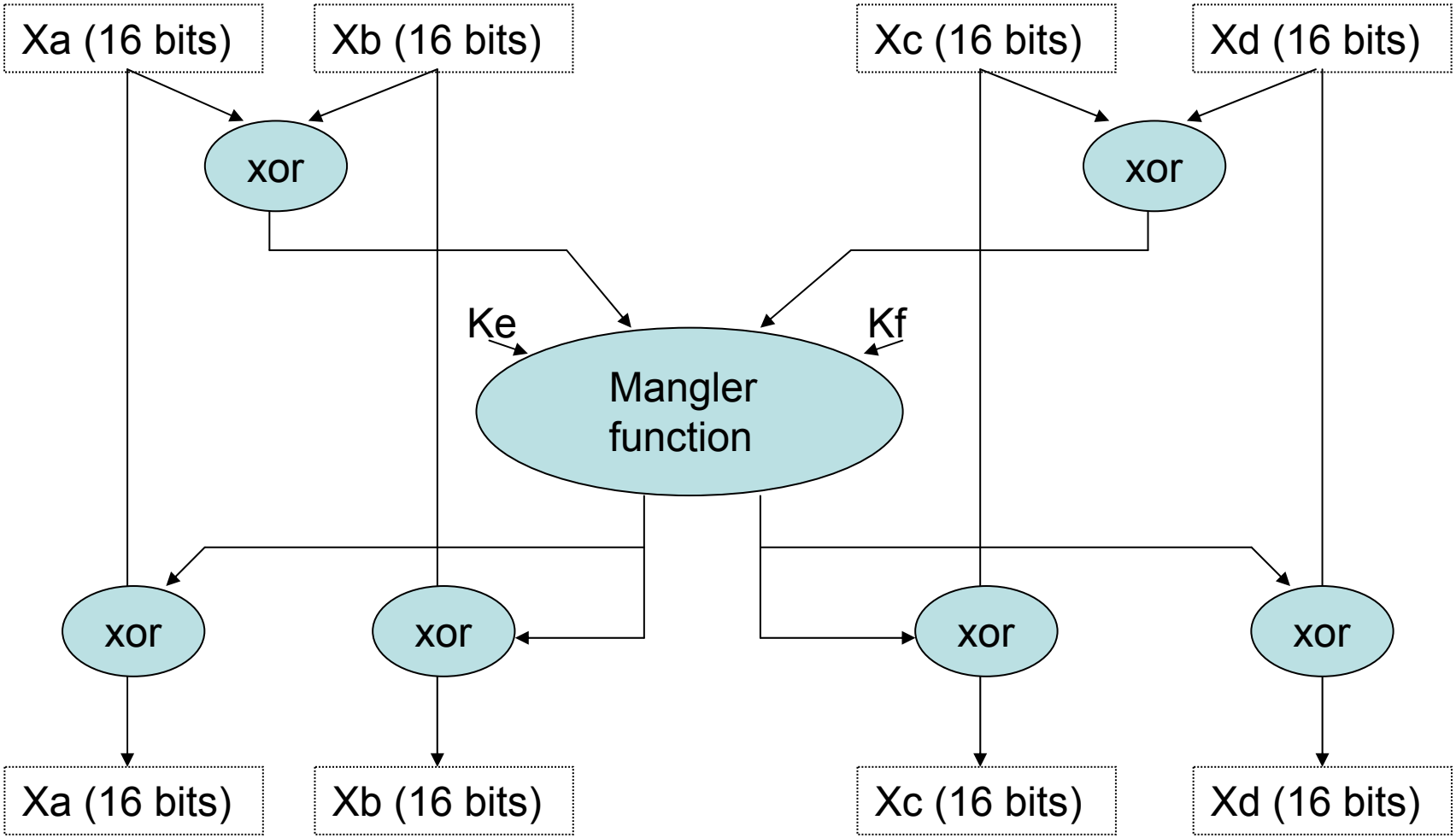
Multiplication modulo $2^{16}+1$, where input 0 is replaced by 2^{16}, and result 2^{16} is encoded as 0



Addition modulo 2^{16}

One round of IDEA: even round

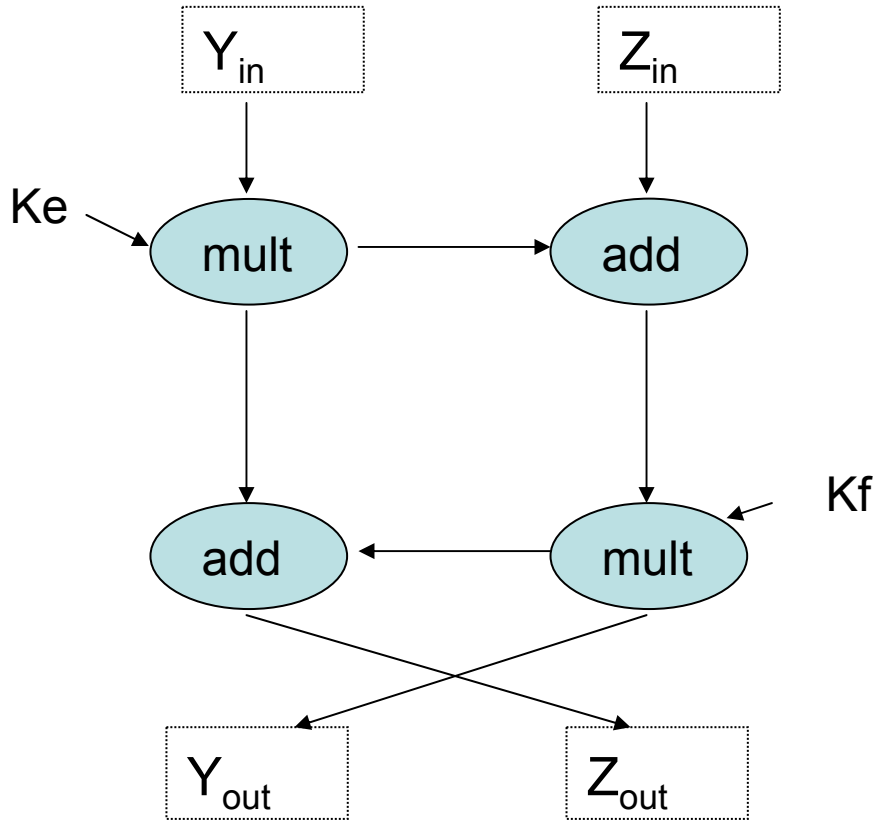
This function is its own inverse!



The mangler function

$Y_{out} = (Ke \text{ mult } Y_{in}) \text{ add } Z_{in}) \text{ mult } Kf$

$Z_{out} = (Ke \text{ mult } Y_{in}) \text{ add } Y_{out}$



The Security of IDEA

- IDEA has been around almost 15 years
- Designed by Xuejia Lai and Jim Massey
- Its only problem so far is its small block size
- Numerous analysis has been published, but nothing substantial
- It is not available in public domain, except for research purposes
- It is available under licence
- It is widely used, e.g in PGP (see Lecture 11)

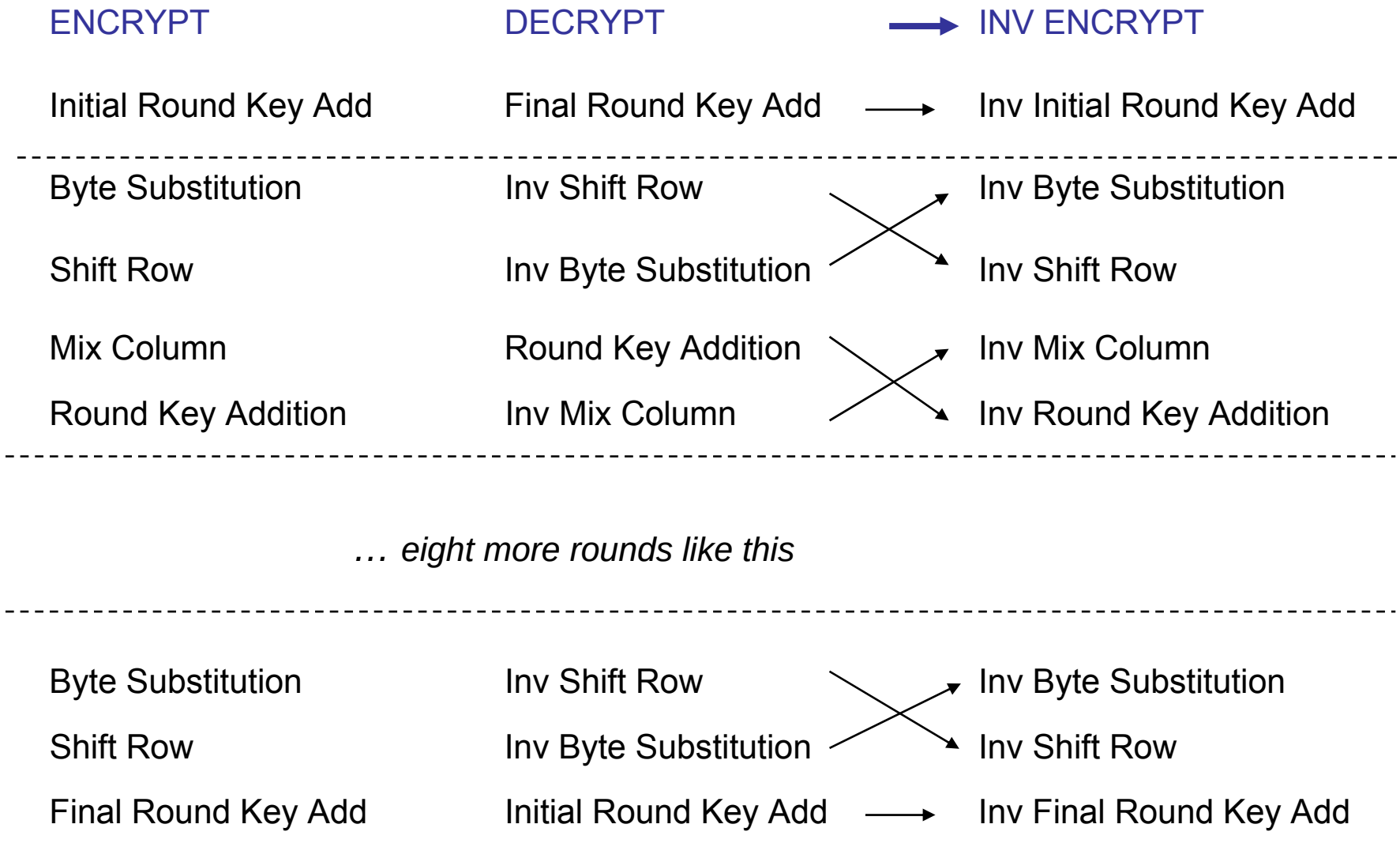
AES

AES

- Candidates due June 15, 1998: 21 submissions, 15 met the criteria
- 5 finalists August 1999: MARS, RC6, Rijndael, Serpent, and Twofish, (along with regrets for E2)
- October 3, 2000, NIST announces the winner: Rijndael
- FIPS 197, November 26, 2001

  Federal Information Processing Standards Publication 197, ADVANCED ENCRYPTION STANDARD (AES)

Rijndael - Inverse Structure



Rijndael S-box Design View

Galois field GF(2^8) with polynomial

$$m(x) = x^8 + x^4 + x^3 + x + 1$$

The Rijndael S-box is the composition $f \circ g$ where

$$g(x) = x^{-1},\ x \in GF(2^8),\ x \neq 0, \text{ and}$$
$$g(0) = 0$$

and f is the affine transformation defined by y = f(x)

Inv ($f \circ g$) = $g \circ$ (Inv f)

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Mix Column - Implemented

The mix column transformation mixes one column of the state at a time.

Column j:

$$b_{0,j} = T_2(a_{0,j}) \oplus T_3(a_{1,j}) \oplus a_{2,j} \oplus a_{3,j}$$
$$b_{1,j} = a_{0,j} \oplus T_2(a_{1,j}) \oplus T_3(a_{2,j}) \oplus a_{3,j}$$
$$b_{2,j} = a_{0,j} \oplus a_{1,j} \oplus T_2(a_{2,j}) \oplus T_3(a_{3,j})$$
$$b_{3,j} = T_3(a_{0,j}) \oplus a_{1,j} \oplus a_{2,j} \oplus T_2(a_{3,j})$$

where:

$$T_2(a) = 2*a \quad \text{if } a < 128$$
$$T_2(a) = (2*a) \oplus 283 \quad \text{if } a \geq 128$$
$$T_3(a) = T_2(a) \oplus a.$$

Mix Column - Design view

The columns of the State are considered as polynomials over GF(2^8).

They are multiplied by a fixed polynomial c(x) given by

$$c(x) = 03 \cdot x^3 + 01 \cdot x^2 + 01 \cdot x + 02$$

The product is reduced modulo $x^4 + 01$.

Matrix form

$$\begin{bmatrix} b_{0,j} \\ b_{1,j} \\ b_{2,j} \\ b_{3,j} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} a_{0,j} \\ a_{1,j} \\ a_{2,j} \\ a_{3,j} \end{bmatrix}$$

The Inverse Mix Column polynomial is $c(x)^{-1} \bmod (x^4 + 01) = d(x)$ given by

$$d(x) = 0B \cdot x^3 + 0D \cdot x^2 + 09 \cdot x + 0E$$

The Security of AES

- Designed to be resistant against differential and linear cryptanalysis
  - S-boxes optimal
  - Wide Trail Strategy
- Has quite an amazing algebraic structure (see the next slide)
- Algebraic cryptanalysis tried but not yet (!) successful
- Algebraic cryptanalysis: given known plaintext – ciphertext pairs construct algebraic systems of equations, and try to solve them.

Differential and linear cryptanalysis

Differential cryptanalysis (Biham-Shamir 1990)

- Chosen plaintext attack
- A large number of pairs of plaintext blocks are generated. Each pair of plaintext has a fixed difference. Corresponding ciphertexts are computed (using the encryption device with a fixed key as black box).
- Main idea: The statistics of the differences of the data blocks before the last round can be predicted.
- Exhaustive search of the last round key are performed by testing if decryptions with the candidate key of the ciphertext pairs gives results that match with the predicted statistics.

Differential and linear cryptanalysis

Linear cryptanalysis (Matsui 1993)

- Known plaintext attack
- A large number of plaintext blocks and their corresponding ciphertexts are known.
- Main idea: The statistics of a fixed linear combination of the data bits before the last round can be predicted by some fixed linear combination of the plaintext bits.
- Exhaustive search of the last round key are performed by testing if decryptions with the candidate key of the ciphertext blocks gives results that match with the predicted statistics.

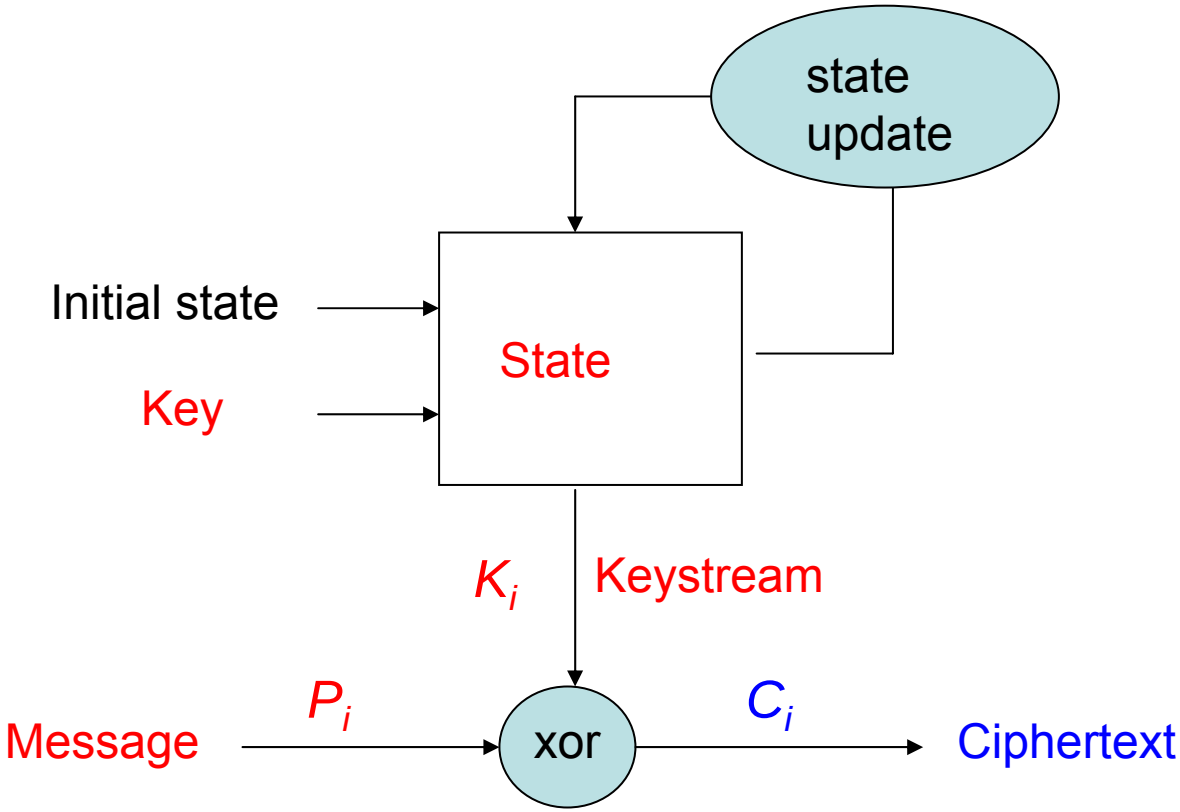
Stream ciphers: Designs

Linear feedback shift register (LFSR). LFSRs are often used as the running engine for a stream cipher.

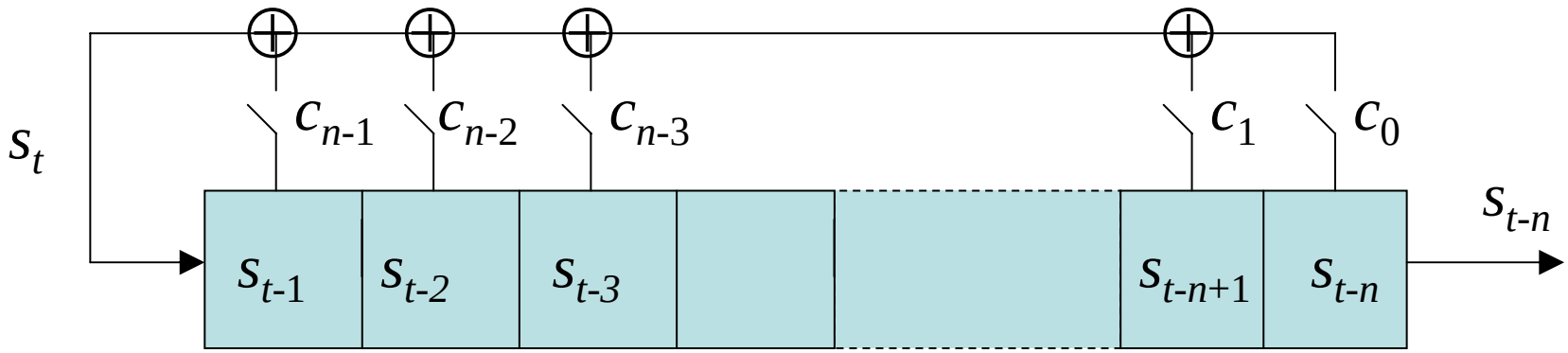
Stream cipher design based on LFSRs uses a number of different LFSRs and nonlinear Boolean functions coupled in different ways. Three common LFSR-based types of stream cipher can be identified:

- *Nonlinear combination generators*: The keystream is generated as a nonlinear function of the outputs of multiple LFSRs
- *Nonlinear filter generators*: The keystream is generated as a nonlinear function of stages of a single LFSR.
- *Clock controlled generators*: In these constructions, the necessary nonlinearity is created by irregular clocking of the LFSRs. The GSM encryption algorithm A5/1 is an example of a stream cipher of this type.

Synchronous stream cipher: encryption



Linear Feedback Shift Register (LFSR)



$$s_t = \sum_{i=1}^{n} c_{n-i} s_{t-i} = c_{n-1} s_{t-1} + c_{n-2} s_{t-2} + \ldots + c_0 s_{t-n} \quad \text{, for all } t \geq n.$$

The taps c_i are defined be giving the *feedback polynomial*

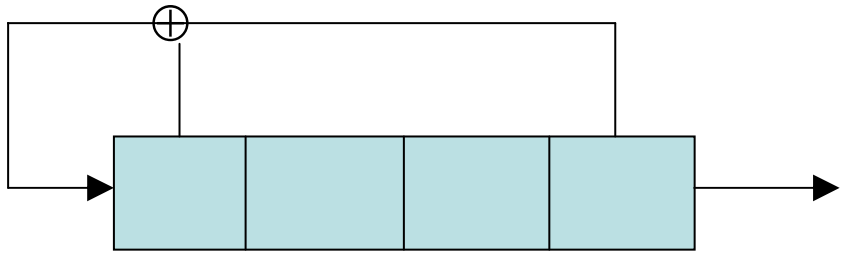
$$f(x) = x^n + c_{n-1} x^{n-1} + c_{n-2} x^{n-2} + \ldots + c_1 x + c_0$$

LFSR: Example

NOTE: Assume now that everything is binary, that is, in bits. Sums are taken mod 2. (Non-binary LFSRs exist.)

$$f(x) = x^4 + x^3 + 1$$

$$\Rightarrow c_0 = c_3 = 1 \text{ and } c_1 = c_2 = 0$$



Let us take this as an initial state:	0	0	1	1
Then the next state is this:	1	0	0	1
And so on:	0	1	0	0
	0	0	1	0
	0	0	0	1
	1	0	0	0
For how long it goes?	...	...	...	...

LFSR statistical properties

The maximum length of the cycle for an LFSR of length n is $2^n - 1$. With maximum cycle the LFSR produces a sequence of length $2^n - 1$.

A maximum length sequence has ideal statistical properties:

- $2^{n-1} - 1$ zeroes and 2^{n-1} ones
- One string of ones of length n; one string of zeroes of length $n-1$
- Also ones and zeroes occur in about equally many pairs, triples ... , and so on.

A maximum length sequence (m-sequence) is achieved using a so-called primitive polynomials. For a source of primitive polynomials see:

http://fchabaud.free.fr/English/default.php?COUNT=1&FILE0=Poly

Autocorrelation function

- The spectral properties of a periodic sequence can be analyzed using a number of transforms related to discrete Fourier transform. One such transform is the Autocorrelation function $C(k)$, $k \in \mathbf{N}$, defined as follows:
- Let N be the length of the cycle (period) of the sequence s_0, s_1 , …, s_i,….Then

$$C(k) = \frac{1}{N} \sum_{i=0}^{N-1} (2s_i - 1)(2s_{i+k} - 1), k \in \mathbf{N}$$

- Clearly, $C(k) = C(N - k)$, for all $k \in \{0,1,\ldots,N\}$.

Golomb's randomness postulates

R1: In the cycle of the sequence the number of 1-bits differs from the number of 0-bits by at most 1.

R2: In the cycle of the sequence, at least ½ of the runs have length 1, at least one ¼ have length 2, at least ⅛ have length 3, etc., as long as the number of runs so indicated exceeds 1. Moreover, for each of these lengths, there are (almost) equally many gaps and blocks.

R3: Let N be the length of the cycle (period) of the sequence (s_i).The *autocorrelation function* is two-valued. That is, for some integer K:

$$C(k) = \begin{cases} 1, \text{if } k = 0, \\ \dfrac{K}{N}, \text{if } 1 \le k \le N-1 \end{cases}$$

Note: In general the autocorrelation function takes more than two values

Definition: A binary sequence which satisfies Golomb's randomness postulates is called a *pseudo-noise* or a *pn-sequence.*

Example

Consider the sequence with cycle length 15:

```
0 1 1 0 0 1 0 0 0 1 1 1 1 0 1
```

R1: The number of 0-bits is 7, the number of 1-bits is 8

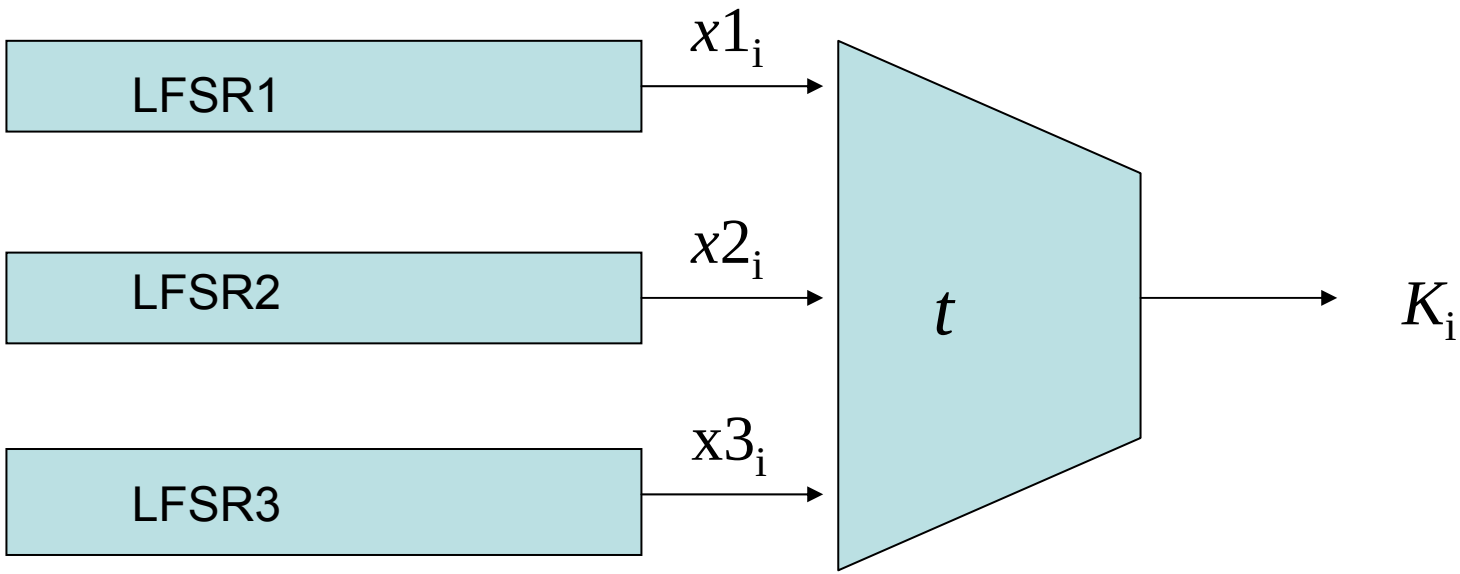
R2: the sequence has eight runs:

- 4 runs of length 1(2 gaps and 2 blocks)
- 2 runs of length 2 (1 gap and 1 block)
- 1 run of length 3 (1 gap)
- 1 run of lenght 4 (1 block)

R3: The autocorrelation function $C(k)$ takes on two values $C(0) = 1$ and $C(k) = -1/15$, for $k \neq 0$

Combination generator

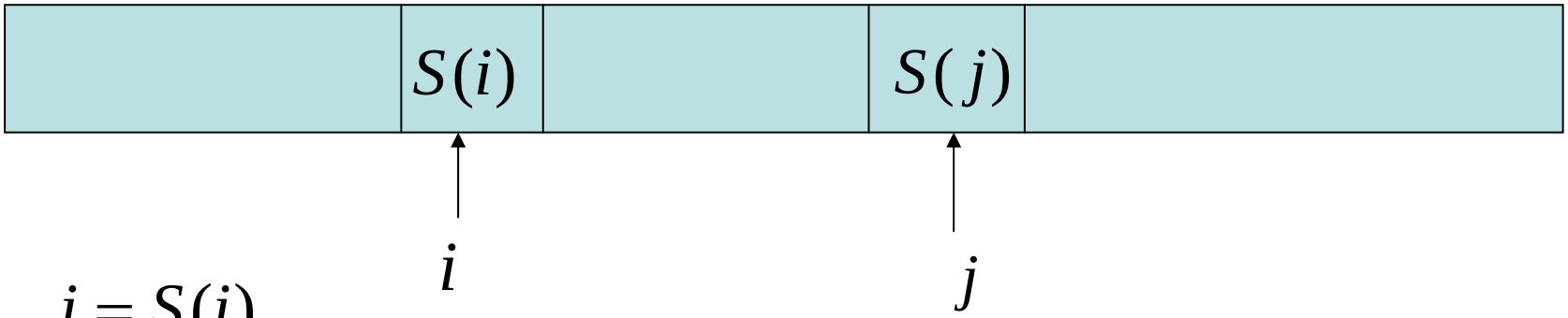
Example: Threshold generator



$$t(x1, x2, x3) = 1, \text{ if at least two of the inputs are equal to } 1$$
$$0, \text{ otherwise}$$

RC4

Register of 256 octets initialised using the key.
Counter i is set to zero. Then:



$$j = S(i)$$

$$S(i) \text{ are } S(j) \text{ swapped}$$

$$k = (j + S(j)) \bmod 256$$

$$\text{ouput} = S(k)$$

$$i = (i + 1) \bmod 256$$

4.2 Block cipher confidentiality modes of operation

Block ciphers are used in different modes of operation.

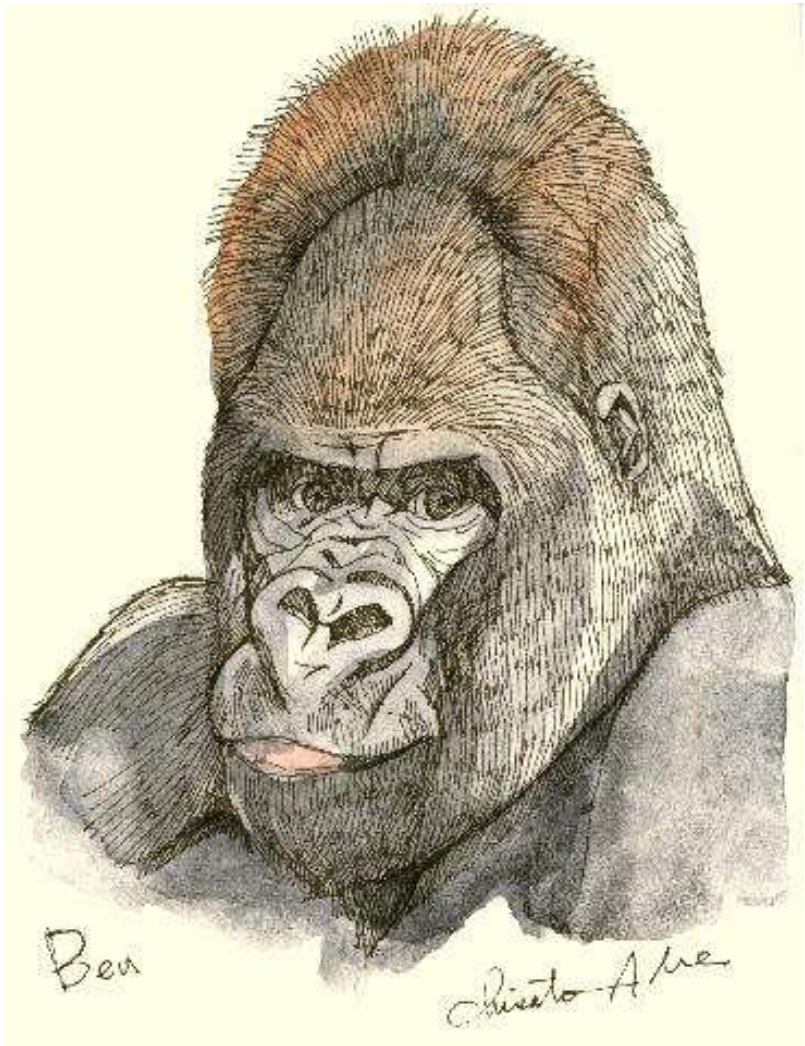
- AES modes of operation:
  - ELECTRONIC CODEBOOK MODE (ECB)
  - CIPHER BLOCK CHAINING (CBC)
  - CIPHER FEEDBACK (CFB)
  - OUTPUT FEEDBACK (OFB)
  - COUNTER MODE (CTR)

standardized by NIST, Special Publication 800-38A , see:
http://csrc.nist.gov/publications/nistpubs/index.html

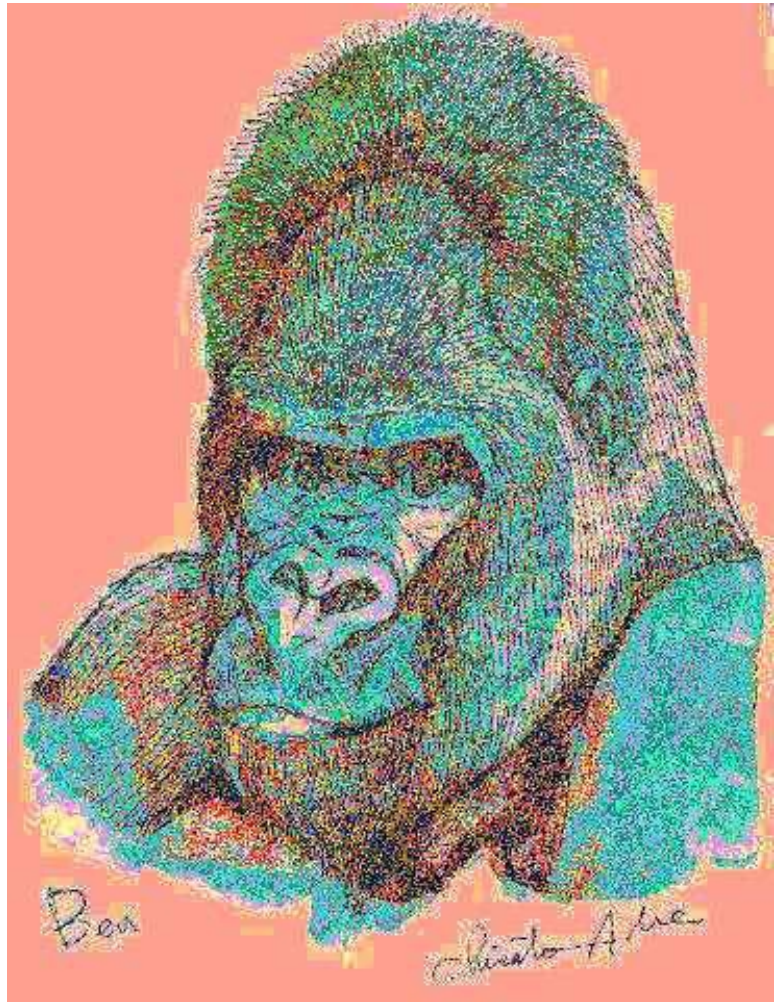
DES algorithm not secure any more (small key size), enhancement

- Triple DES Special Publication 800-67

ECB encryption

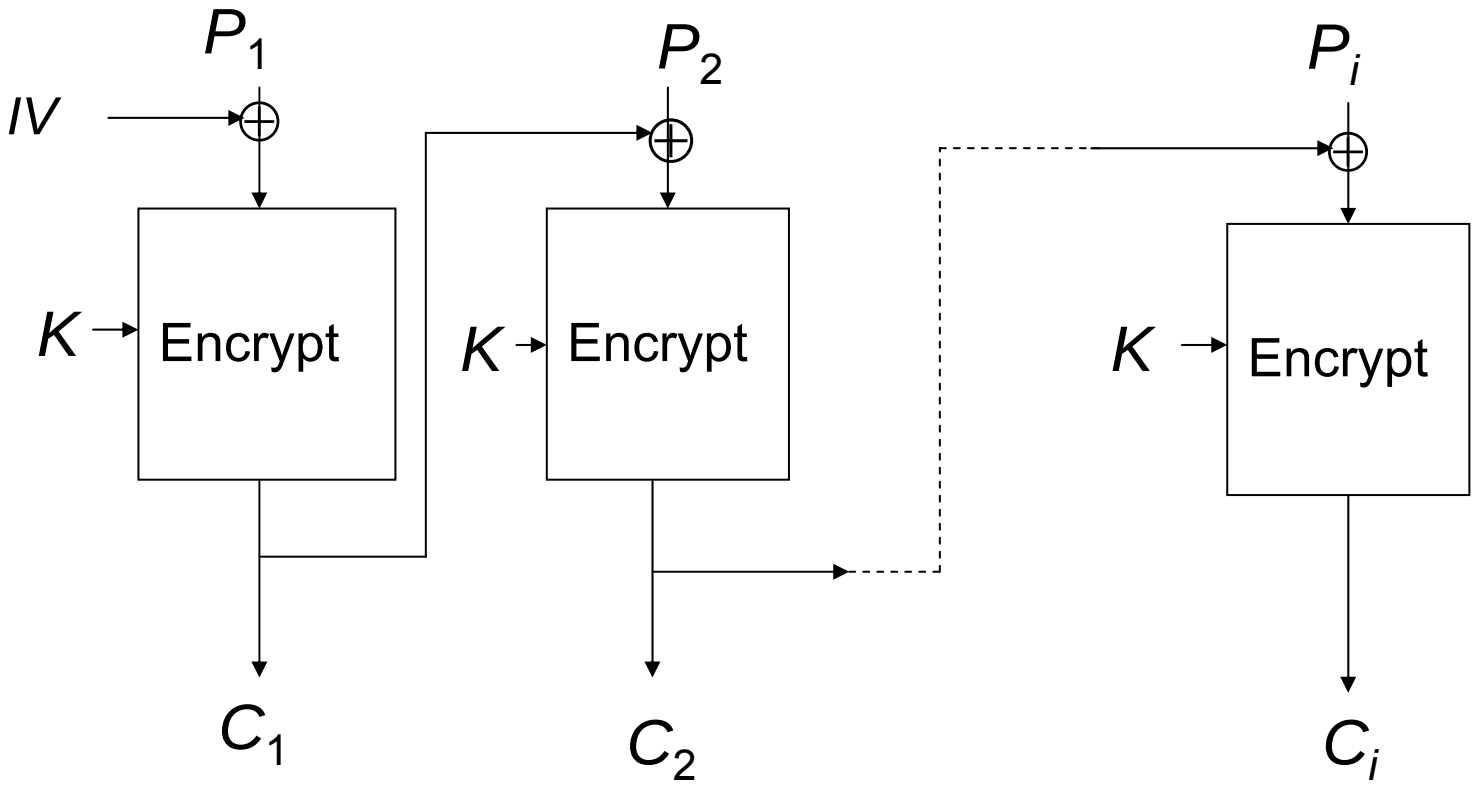


Plaintext



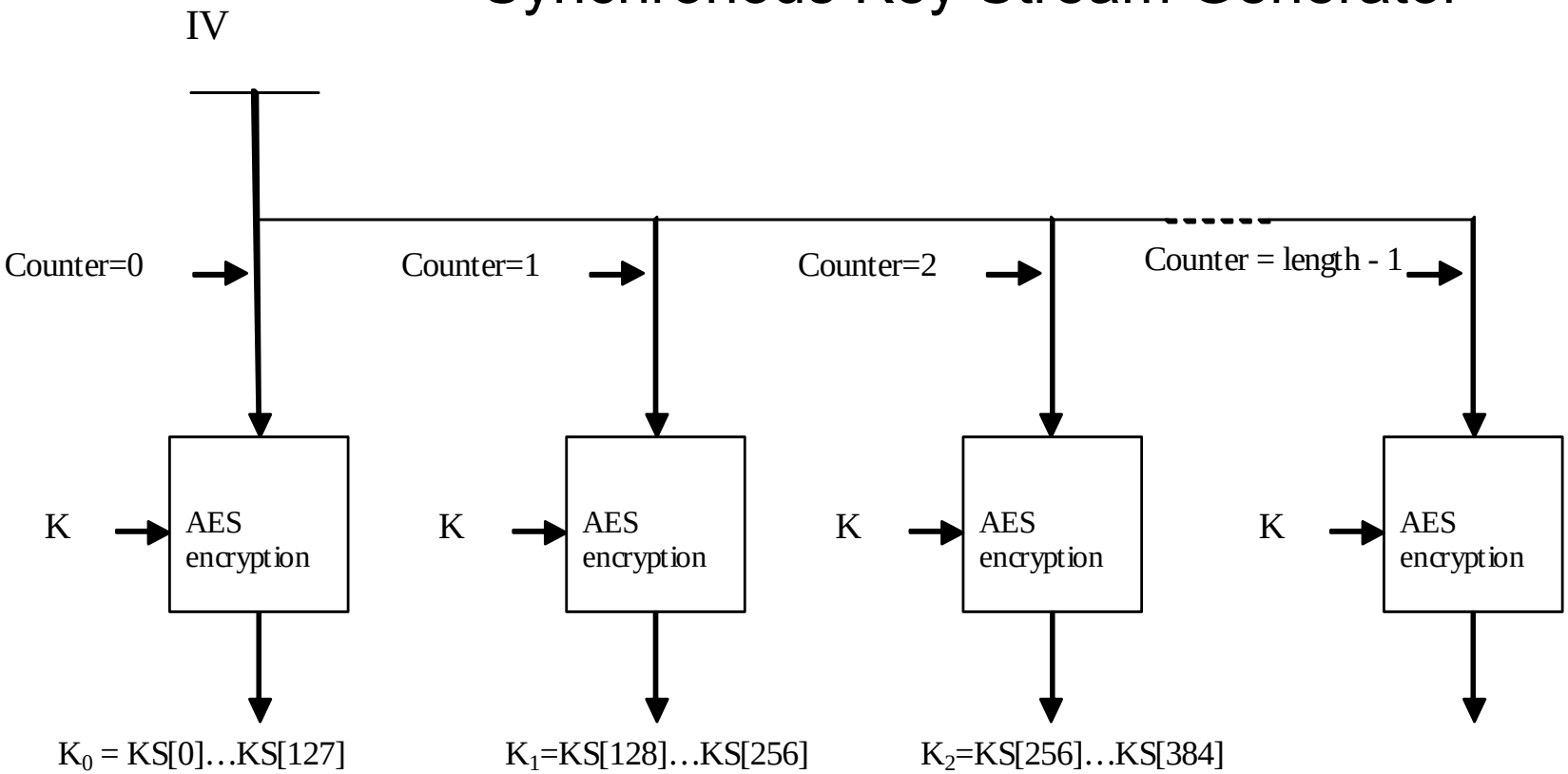
Ciphertext

Cipher Block Chaining (CBC) Mode: Encryption



Counter Mode

Synchronous Key Stream Generator



Triple DES (TDEA, 3DES)

DES algorithm is not strong any more (small key size)

Double DES with two different keys K_1 and K_2 not good either (security not more than single DES) due to the Meet-in-the-Middle Attack (see next slide):

Triple DES Special Publication 800-67, see

http://csrc.nist.gov/publications/nistpubs/index.html

Triple DES with two keys

$$C = E_{K_1}(D_{K_2}(E_{K_1}(P)))$$

reduces to single DES if we select $K_1 = K_2$. In this manner compatibility with legacy applications can be achieved.

Meet in the Middle

Double DES with two different keys K_1 and K_2 is not good: security is not (essentially) more than single DES due to the Meet-in-the-Middle Attack. Such attack can be launched when the attacker has two known plaintext-ciphertext pairs (P,C) and (P',C') where encryption is done with the same keys values K_1 and K_2 . Then the attacker has

$$C = E_{K_2}(E_{K_1}(P)) \text{ and } C' = E_{K_2}(E_{K_1}(P'))$$

or what is the same:

$$D_{K_2}(C) = E_{K_1}(P) \text{ and } D_{K_2}(C') = E_{K_1}(P').$$

Meet in the Middle …

Now we make a table T with a complete listing of all possible pairs $K_2, D_{K_2}(C)$ as K_2 runs through all possible 2^{56} values. The table has 2^{56} rows with 120 bits on each row. We make one more column to this table, and fill it with K_1 values as follows: For each K_1 we compute the value $E_{K_1}(P)$ and search in the table T for a match $D_{K_2}(C) = E_{K_1}(P)$. For each K_2 we expect to find a (almost) unique K_1such that such a match occurs. Now we go through all key pairs K_1 , K_2 suggested by table T, and test against the equation $D_{K_2}(C') = E_{K_1}(P')$ we have based on the second plaintext – ciphertext pair (P’,C’). The solution is expected to be unique. The size of table T is $2^{56}(56 + 64 + \sim 56 \text{ bits}) < 2^{64}$ bits, which is the memory requirement of this attack. The number of encryptions (decryptions) needed is about $4 \cdot 2^{56} = 2^{58}$.

Message authentication codes (MAC)

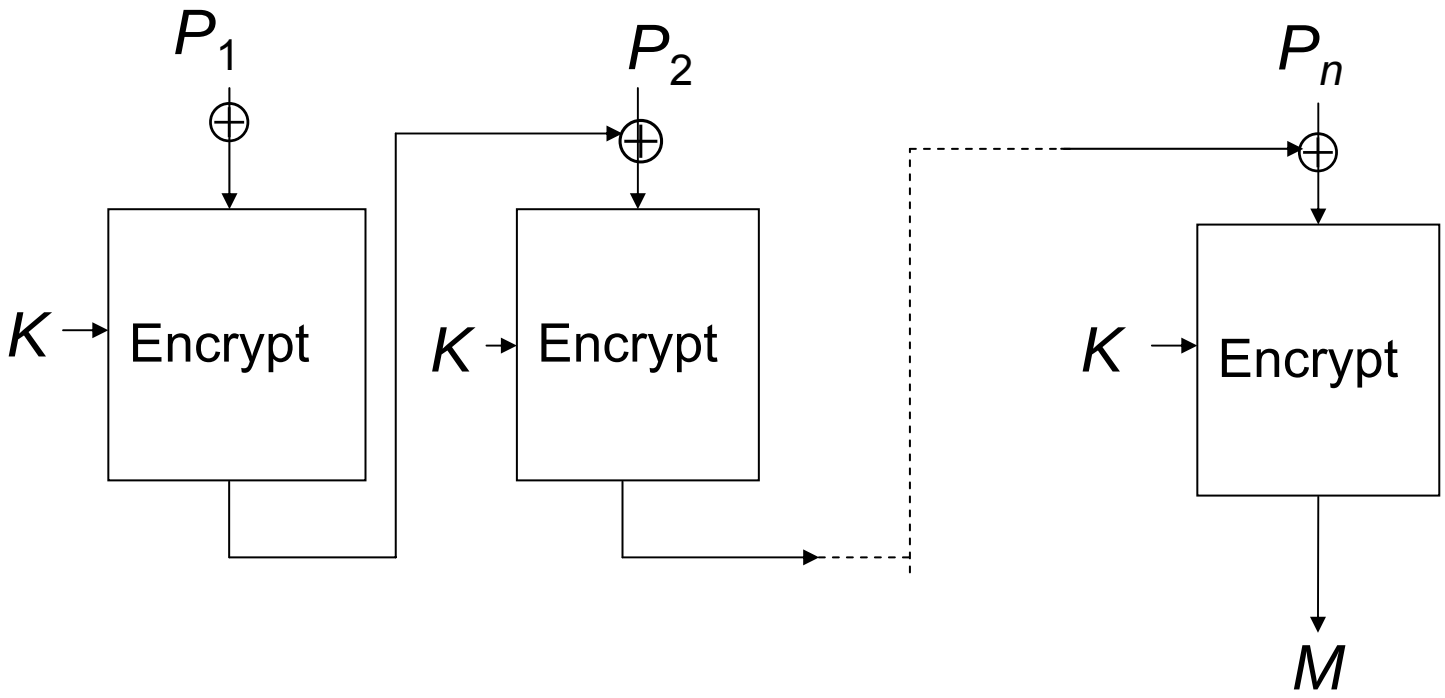
Sender: (Secret key , Message) ⟶ MAC

Receiver: (Secret key , Message) ⟶ MAC

- A MAC of a message P of arbitrary length is computed as a function $H_K(P)$ of P under the control of a secret key K.
- The MAC length m is fixed.
- Security requirement: it must be infeasible, without the knowledge of the secret key, to determine the correct value of $H_K(P)$ with a success probability larger than $1/2^m$. This is the probability of simply guessing the MAC value correctly at random. It should not be possible to increase this probability even if a large number of correct pairs P and $H_K(P)$ is available to the attacker.

CBC MAC

A MAC mode of operation of any block cipher



- CBC encryption with fixed IV = `00…0`. The last ciphertext block (possibly truncated) is taken as the MAC.

Polynomial MAC

- Another MAC suitable for use with stream ciphers
- Idea: An (cryptographically insecure) error detecting code is encrypted using non-repeating keystream (ideally, a one-time pad)
- An n-block message $P = P_0, P_1, \ldots, P_{n-1}$ with block size m bits is associated with the polynomial with m-bit coefficients $P_i \in \mathrm{GF}(2^m)$:

$$P(x) = P_0 + P_1 x + P_2 x^2 + \ldots + P_{n-1} x^{n-1}$$

- The value of the polynomial taken in point $x \in \mathrm{GF}(2^m)$ is taken as an m-bit string $P(x) \in \mathrm{GF}(2^m)$.
- The secret key K consists of a point $x = X$ and an m-bit one-time key stream string $(k_0, k_1, k_2, \ldots, k_{m-1})$.
- First the message polynomial is evaluated at the point x = X. Let us denote $P(x) = (c_0, c_1, c_2, \ldots, c_{m-1})$. The MAC is computed as the xor of the key stream string and the value as

$$H_K(P) = (c_0 \oplus k_0, c_1 \oplus k_1, c_2 \oplus k_2, \ldots, c_{m-1} \oplus k_{m-1})$$

Note: The point X can be reused for different messages

Hash functions

Sender: Message → Hash code

Receiver: Message → Hash code

- A hash code of a message P of arbitrary length is computed as a function $H(P)$ of P . The hash length m is fixed.
- Security requirements:
  1. *Preimage resistance:* Given h it is impossible to find P such that $H(P) = h$
  2. *Second preimage resistance:* Given P it is impossible to find P' such that $H(P') = H(P)$
  3. *Collision resistance: I*t is impossible to find P and P' such that $P \neq P'$ and $H(P') = H(P)$

Hash functions

- Similarly as MAC algorithms, hash functions typically operate on relatively large blocks of data. Most hash functions are iterated constructions. The core function in a hash function is a compression function. At each round the compression function takes a new data block and compresses it together with the compression result from the previous rounds. Hence the length of the message to be authenticated determines how many iteration rounds are required to compute the MAC value.

- Hash function is public: Given a message P anybody can compute the hash code of P.

Collision Attack

- An upperbound to the security of hash functions is due to the collision probability which is estimated using Birthday Paradox.
- Random oracle: Given a message an ideal hash function, with m-bit output, computes the hash value by selecting the value uniformly at random from all possible 2^m values. To find a collision with probability at least 1/2, approximately $1{,}17 \cdot 2^{m/2}$ messages need to be hashed. This gives an estimate to the workload of making the collision attack successful.

Revised SHA Standard

csrc.nist.gov/publications/**fips**/**fips**180-2/**fips**180-2.pdf

	SHA-1	**SHA-256**	**SHA-384**	**SHA-512**
Hash size	160	256	384	512
Message size	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
Block size	512	512	1024	1024
Word size	32	32	64	64
Number of steps	80	80	80	80
Claimed security	2^{80}	2^{128}	2^{192}	2^{256}

HMAC- hash based MAC

- RFC 2104: the MAC for IP security
- To use available hash functions
- To allow hash function to be replaced easily
- To preserve the performance of a hash function
- Easy handling of keys
- Well understood cryptographic security

- Recent collision attacks against hash functions do not effect HMAC constructions

HMAC algorithm

H	hash function
M	message input to HMAC (after hash function specific padding added)
L	number of blocks in M
b	number of bits in a block
n	length of the hash code of H
K	secret key, recommended length ≥ n
K^+	a b-bit string formed by appending zeros to the end of K
$ipad$	= `00110110` repeated b/8 times
$opad$	= `01011100` repeated b/8 times

$$HMAC(K;M) = H[(K^+ \oplus opad) \,||\, H[(K^+ \oplus ipad)||M]]$$

Public key encryption

Sender: (Message , Public key) → Ciphertext

Receiver: (Ciphertext , Private key) → Message

- Slow, usually used to encrypt short messages in more complex protocols than just bulk message encryption: data authentication, key agreement etc.
- Because of the mathematical structures involved, complex message formatting rules (with hash functions) are required.
- Chosen ciphertext attacks maybe an essentially more serious threat than chosen plaintext (for symmetric block ciphers they are about the same). We will see an example later.
- RSA, ElGamal in different groups, Pairing based techniques …

Chinese Remainder Theorem (two moduli)

Problem: Assume m_1 and m_2 are coprime. Given x_1 and x_2, how to find $0 \le x < m_1 m_2$ such that

$$x = x_1 \bmod m_1$$

$$x = x_2 \bmod m_2$$

Solution: Use the Extended Euclidean Algorithm to find u and v such that $u \cdot m_1 + v \cdot m_2 = 1$. Then

$$x = x \cdot (u \cdot m_1 + v \cdot m_2) = x \cdot u \cdot m_1 + x \cdot v \cdot m_2$$

$$= (x_2 + r \cdot m_2) \cdot u \cdot m_1 + (x_1 + s \cdot m_1) \cdot v \cdot m_2.$$

It follows that

$$x = x \bmod (m_1 m_2) = (x_2 \cdot u \cdot m_1 + x_1 \cdot v \cdot m_2) \bmod (m_1 m_2)$$

Chinese Remainder Theorem (general case)

Theorem: Assume $m_1, m_2, \ldots, m_t$ are mutually coprime. Denote $M = m_1 \cdot m_2 \cdot \ldots \cdot m_t$. Given $x_1, x_2, \ldots, x_t$ there exists a unique x, $0 \leq x < M$, such that

$$x = x_1 \bmod m_1$$

$$x = x_2 \bmod m_2$$

$$\ldots$$

$$x = x_t \bmod m_t$$

x can be computed as

$$x = (x_1 \cdot u_1 \cdot M_1 + x_2 \cdot u_2 \cdot M_2 + \ldots + x_t \cdot u_t \cdot M_t) \bmod M,$$

where $M_i = (m_1 \cdot m_2 \cdot \ldots \cdot m_t)/m_i$ and $u_i = M_i^{-1} \pmod{m_i}$.

Proof: For the case $t = 2$ see previous page. The general case is handled in T-79.5501.

Chinese Remainder Theorem: Example

Let $m_1 = 7$, $m_2 = 11$, $m_3 = 13$. Then $M = 1001$.

Problem: Compute x, $0 \le x \le 1000$ such that

$x = 5 \bmod 7$

$x = 3 \bmod 11$

$x = 10 \bmod 13$

Solution:

$M_1 = m_2 m_3 = 143$; $M_2 = m_1 m_3 = 91$; $M_3 = m_1 m_2 = 77$

$u_1 = M_1^{-1} \bmod m_1 = 143^{-1} \bmod 7 = 3^{-1} \bmod 7 = 5$; similarly

$u_2 = M_2^{-1} \bmod m_2 = 3^{-1} \bmod 11 = 4$; $u_3 = (-1)^{-1} \bmod 13 = -1$.

Then

$x = (5 \cdot 5 \cdot 143 + 3 \cdot 4 \cdot 91 + 10 \cdot (-1) \cdot 77) \bmod 1001 = 894$

Euler's Totient Function $\phi(n)$

Definition: Let $n > 1$ be integer. Then we set

$$\phi(n) = \#\{\, a \mid 0 < a < n, \gcd(a,n) = 1\},$$

that is, $\phi(n)$ is the number of positive integers less than n which are coprime with n.

For prime p, $\phi(p) = p - 1$. We define $\phi(1) = 1$.

For a prime power p^e, we have $\phi(p^e) = p^{e-1}(p-1)$.

The multiplicative property holds:

$$\phi(m \times n) = \phi(m) \times \phi(n), \text{ for all } m, n, \gcd(m,n) = 1 .$$

Now Euler's totient function can be computed for any integer using its prime factorisation.

Example: $\phi(18) = \phi(2 \times 3^2) = \phi(2) \times \phi(3^2) = (2-1) \times (3-1)3^1 = 6$, that is, the number of invertible (coprime with 18) numbers modulo 18 is equal to 6. They are: $1, 5, 7, 11, 13, 17$.

Euler's Theorem

$$Z_n^* = \{a \mid 0 < a < n, \gcd(a,n) = 1\}, \text{and } \# Z_n^* = \phi(n)$$

Euler's Theorem: For any integers n and a such that $a \neq 0$ and gcd(a,n) = 1 the following holds:

$$a^{\phi(n)} \equiv 1 (\mathrm{mod}\, n)$$

Fermat's Theorem: For a prime p and any integer a such that $a \neq 0$ and a is not a multiple of p the following holds:

$$a^{p-1} \equiv 1 (\mathrm{mod}\ p)$$

Setting up the RSA

- Generate two different odd primes p and q
  - Pick odd integers of suitable size and test for primality
- Compute $n = pq$ and compute $\phi(n) = (p-1)(q-1)$
- Select a public exponent e such that $\gcd(e, \phi(n)) = 1$
- Using Extended Euclidean Algorithm compute the multiplicative inverse of e modulo $\phi(n)$. Denote $d = e^{-1} \bmod \phi(n)$.

Public key: $K_{\text{pub}} = (n, e)$

Private key: $K_{\text{priv}} = (n, d)$

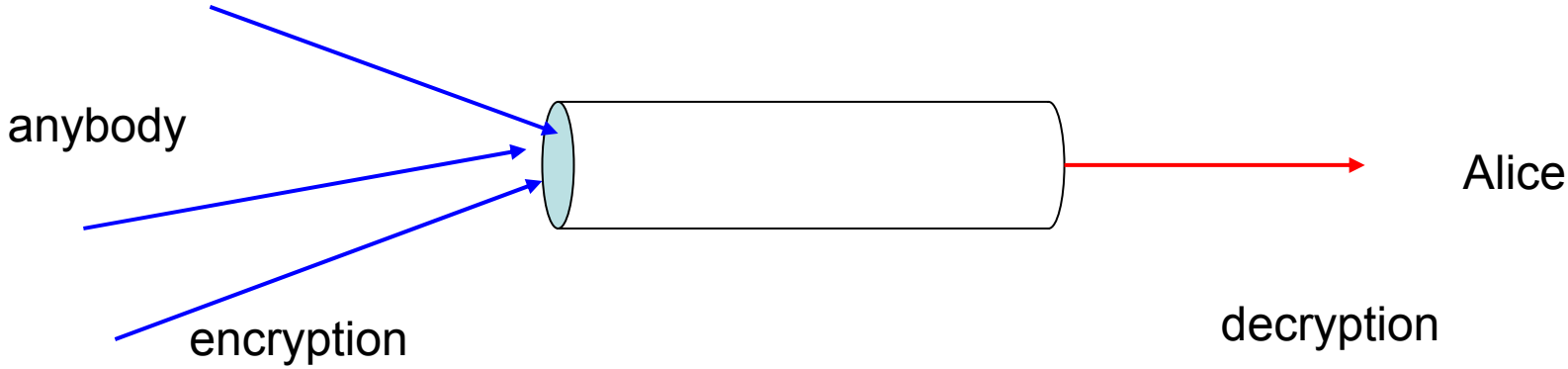
(or $K_{\text{priv}} = (p, q, d)$. This is needed if private computations make use of the CRT.)

n is called the RSA modulus; e is the public encryption exponent; d is the private decryption exponent.

The Principle of Public Key Cryptosystems

Encryption operation is public

Decryption operation is private



Alice's key for a public key cryptosystem is a pair: $(K_{\text{pub}}, K_{\text{priv}})$ where K_{pub} is public and K_{priv} cannot be used by anybody else than Alice.

RSA encryption and decryption

Let M be a message, $0 \le M < n$. Then

encryption of M is $C = M^e \bmod n$

decryption of C is $M = C^d \bmod n$

This works, because $(M^e)^d \bmod n = M$.

Proof. (For $M \in Z_n^*$): By Euler's theorem,

$M^{\phi(n)} \equiv 1 (\bmod n)$. On the other hand, $ed \equiv 1 (\bmod \phi(n))$

It follows:

$$(M^e)^d = M^{ed} = M^{1+k\phi(n)} = M \cdot (M^{\phi(n)})^k = M (\bmod n)$$

Square and multiply

Fast exponentiation algorithms exist. The simplest one is the Square and Multiply Algorithm. It has two versions: left-to-right or right-to-left. Below we show the right-to-left (from the lsb to msb) version.

To compute $a^d \bmod n$

use the binary representation of the exponent d :

$$d = d_0 + d_1 \cdot 2 + d_2 \cdot 2^2 + \cdots + d_{k-1} 2^{k-1}$$

where $d_0, d_1, d_2, \ldots, d_{k-1}$ are bits, i.e. equal to 0 or 1. Now

$$a^d = a^{d_0 + d_1 \cdot 2 + d_2 \cdot 2^2 + \cdots + d_{k-1} 2^{k-1}} = a^{d_0} (a^2)^{d_1} (a^{2^2})^{d_2} \cdots (a^{2^{k-1}})^{d_{k-1}} (\bmod n)$$

Compute the k powers:

$$a = a^{2^0}; a^{2^1}; a^{2^2}; a^{2^3}; \cdots; a^{2^{k-1}} (\bmod n)$$

and from of product (modulo n) of those powers $a^{2^i} (\bmod n)$, for which the corresponding $d_i = 1$.

Security of RSA

- If factoring of n is easy, then $\phi(n)$ is easy to compute. Given $\phi(n)$ and e, it is easy to compute the private exponent d .

- But even if factoring is hard (as it is believed to be) there may be some other ways to break RSA, without factoring the modulus. But no such break is known. All known breaks can be handled by proper selection of parameters, and message formatting.

Generated set of a primitive element

Example: Finite field $\mathbf{Z}_{19}$

$g = 2$

$g^{\,i} \bmod 19,\ i = 0,1,2,\ldots$

Element $g = 2$ generates all nonzero elements in $\mathbf{Z}_{19}$. It is a primitive element.

i	g^i	i	g^i
0	1	10	17
1	2	11	15
2	4	12	11
3	8	13	3
4	16	14	6
5	13	15	12
6	7	16	5
7	14	17	10
8	9	18	1
9	18		

Cyclic subgroups

$\mathbf{F}$ finite field, $g \in \mathbf{F}^*$, let $\langle g \rangle$ denote the set generated by g : $\langle g \rangle = \{1=g^0, g^1, g^2, \ldots, g^{r-1}\}$, where r is the least positive number such that $g^r = 1$ in $\mathbf{F}$. By Fermat's and Euler's theorems $r \leq \#\,\mathbf{F}^*$= number of elements in $\mathbf{F}^*$.

Definition: r is the order of g.

$\langle g \rangle$ is a subgroup of the multiplicative group $\mathbf{F}^*$ in $\mathbf{F}$.

Axiom 1: $g^i \cdot g^j = g^{i+j} \in \langle g \rangle$.

Axiom 2: associativity is inherited from $\mathbf{F}$

Axiom 3: $1 = g^0 \in \langle g \rangle$.

Axiom 4: Given $g^i \in \langle g \rangle$ the multiplicative inverse is g^{r-i} , as $g^i \cdot g^{r-i} = g^{r-i} \cdot g^i = g^r = 1$

$\langle g \rangle$ is called a cyclic group. The entire $\mathbf{F}^*$ is a cyclic group generated by a primitive element, e.g, $\mathbf{Z}_{19}^* = \langle 2 \rangle$.

Generated set of g

Example: Finite field $\mathbf{Z}_{19}$

$g = 7$

$g^{\,i} \bmod 19$

The multiplicative order of 7 is 3 in $\mathbf{Z}_{19}$.

i	g^i
0	1
1	7
2	49=11
3	77=1
4	7
5	11
…	…

Example: Cyclic group in Galois Field

$GF(2^4)$ with polynomial $f(x) = x^4 + x + 1$

$g = 0011 = x + 1$

$g^2 = x^2 + 1 = 0101$

$g^3 = (x+1)(x^2+1) = x^3 + x^2 + x + 1 = 1111$

$g^4 = (x+1)(x^3 + x^2 + x + 1) = x^4 + 1 = x = 0010$

$g^5 = (x+1)(x^4 + 1) = x^5 + x^4 + x + 1 = x^2 + x = 0110$

$g^6 = (x+1)(x^2 + x) = x^3 + x = 1010$

$g^7 = (x+1)(x^3 + x) = x^4 + x^3 + x^2 + x = x^3 + x^2 + 1 = 1101$

$g^8 = (x+1)(x^3 + x^2 + 1) = x^4 + x^2 + x + 1 = x^2 = 0100$

$g^9 = (x+1)x^2 = x^3 + x^2 = 1100$

$g^{10} = (x+1)(x^3 + x^2) = x^2 + x + 1 = 0111$

$g^{11} = (x+1)(x^2 + x + 1) = x^3 + 1 = 1001$

$g^{12} = (x+1)(x^3 + 1) = x^3 = 1000$

$g^{13} = (x+1)x^3 = x^3 + x + 1 = 1011$

$g^{14} = (x+1)(x^3 + x + 1) = x^3 + x^2 + x = 1110$

$g^{15} = (x + 1)(x^3 + x^2 + x) = 1 = 0001$

Discrete logarithm

Given $a \in \langle g \rangle = \{1, g^1, g^2, \ldots, g^{r-1}\}$, there is x, $0 \leq x < r$ such that $a = g^x$. The exponent x is called the discrete logarithm of a to the base g.

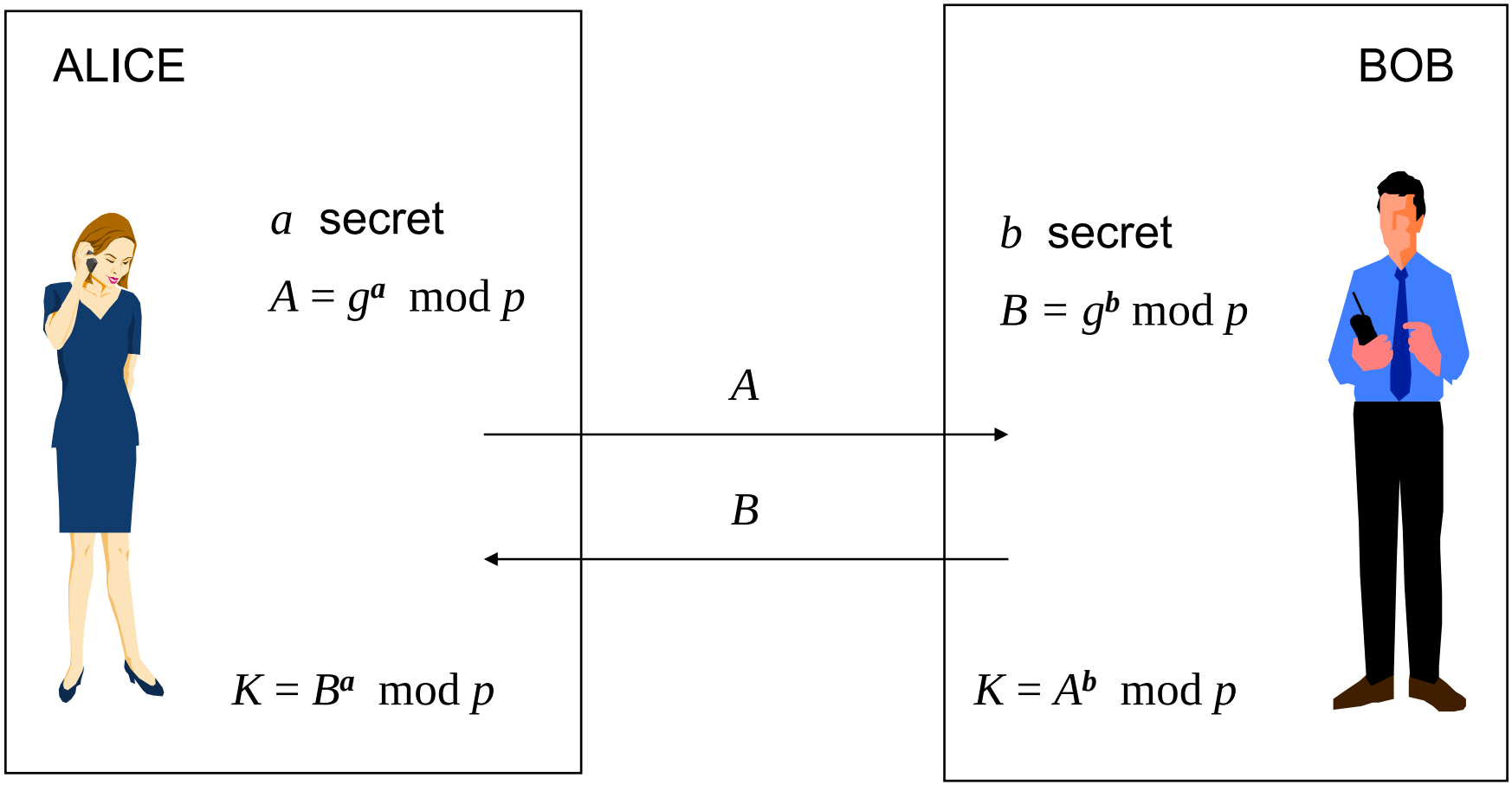
Example: Solve the equation

$$2^x = 14 \bmod 19$$

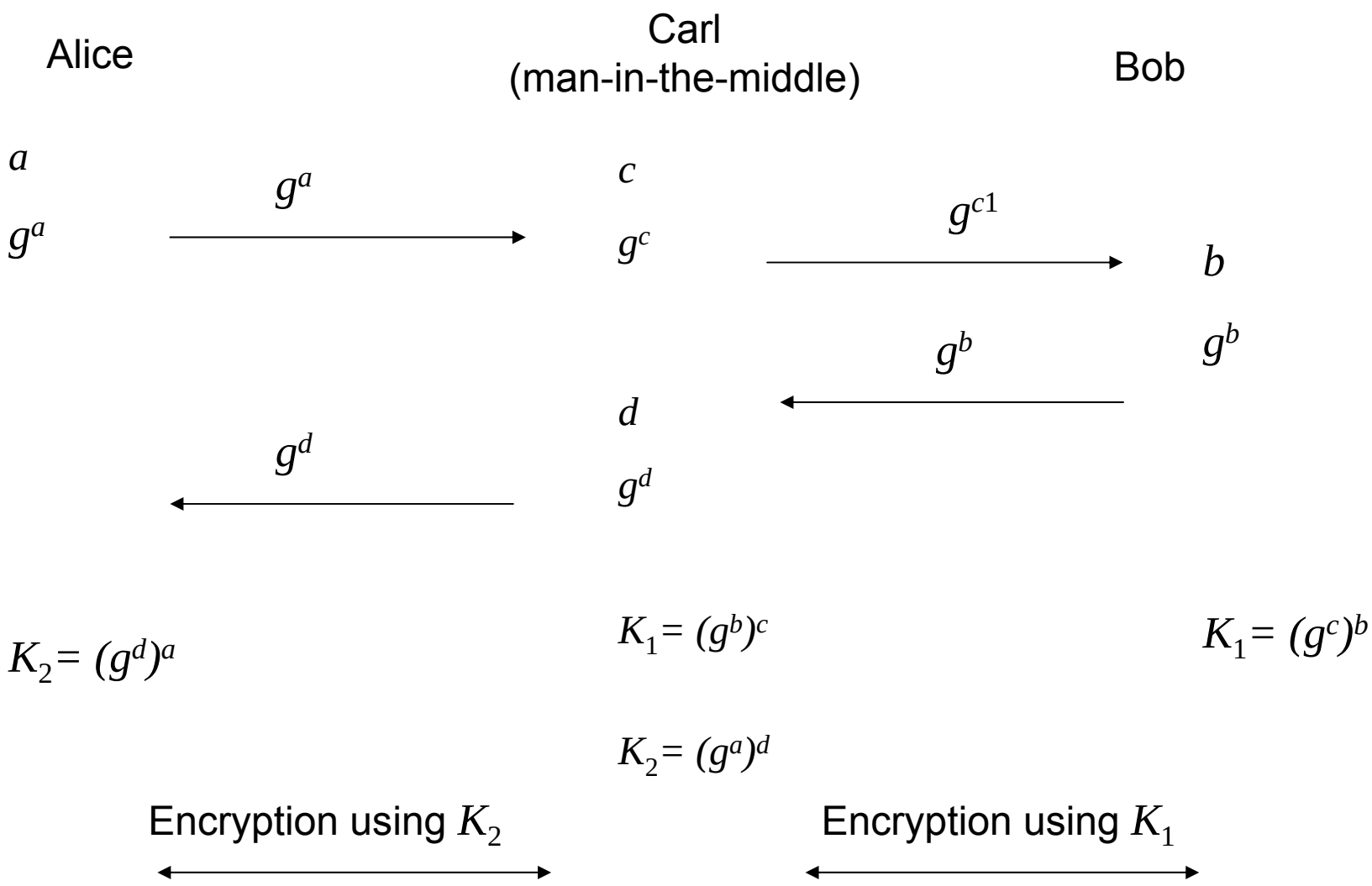
We find the solution using the table (slide 13): x =7.

Without the precomputed table the discrete logarithm is often hard to solve. Cyclic groups, where the discrete logarithm problem is hard, are used in cryptography.

Diffie-Hellman Key Exchange



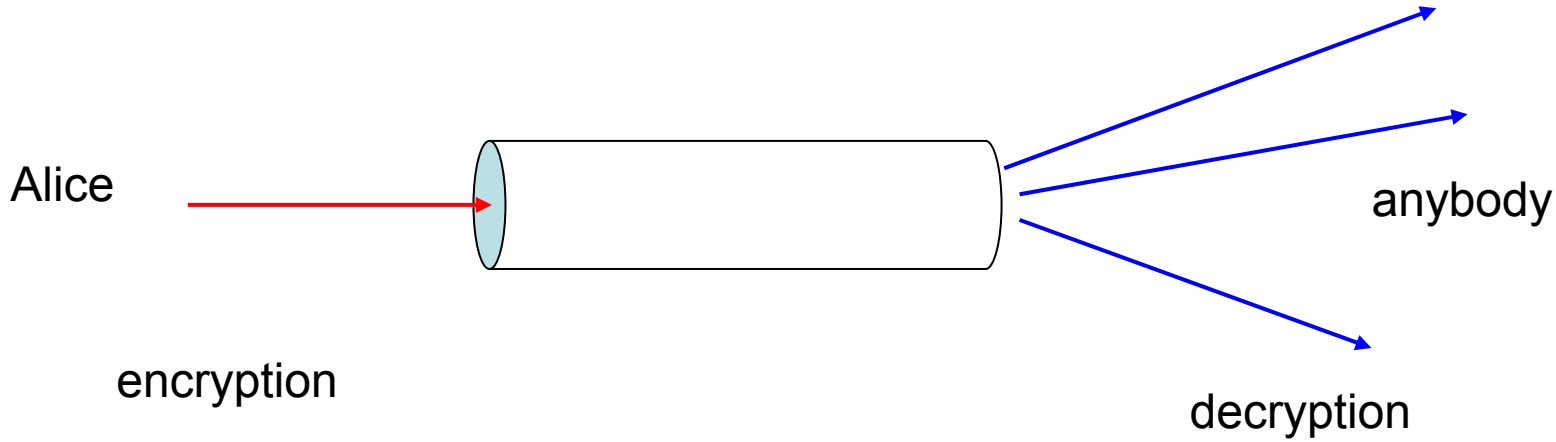
Man-in-the-Middle in the DH KE



Authentication function based on asymmetric cryptography

Encryption operation is private

Decryption is a public operation



Alice's key for a public key cryptosystem is a pair: (K_{pub}, K_{priv}) where K_{pub} is public and K_{priv} cannot be used by anybody else than Alice.

Digital signatures

Sender: (Message , Private key) → Signature

Receiver: (Signature , Public key) → Validity (1 bit)

- Important primitive; the only one to provide non-repudiation.
- Slow, message are signed by applying the digital signature operation on a fixed length hash of the message.
- Used for
  - message authentication protocols
  - non-repudiation protocols
  - authentication and key agreement
  - commitment schemes
  - …
- RSA, ElGamal in different groups, Schnorr, DSA, Pairing based techniques

Authentication functions

- Authentication functions are cryptographic primitives which are used by message authentication protocols between two parties, sender and receiver. Sender attaches to the message an authenticator. Receiver uses the authenticator to verify authenticity of the message.

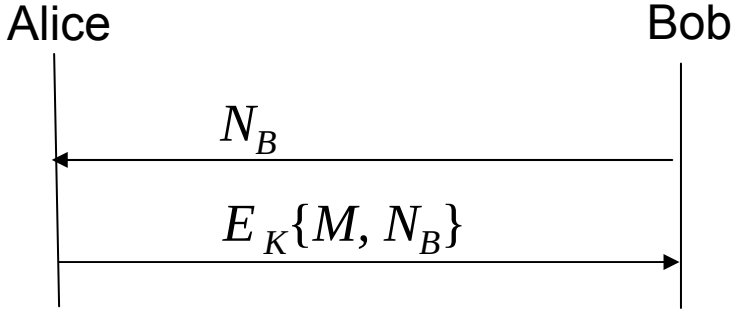
- Authentication functions:
  - Message encryption (with integrity protection)
  - Message authentication code (MAC function)
  - Hash function
  - Digital signature

- Note. Message encryption even with a block cipher does not provide message integrity, see next slide.

Basic Authentication Techniques: Message Freshness and Principal Liveness

Challenge-Response Mechanism

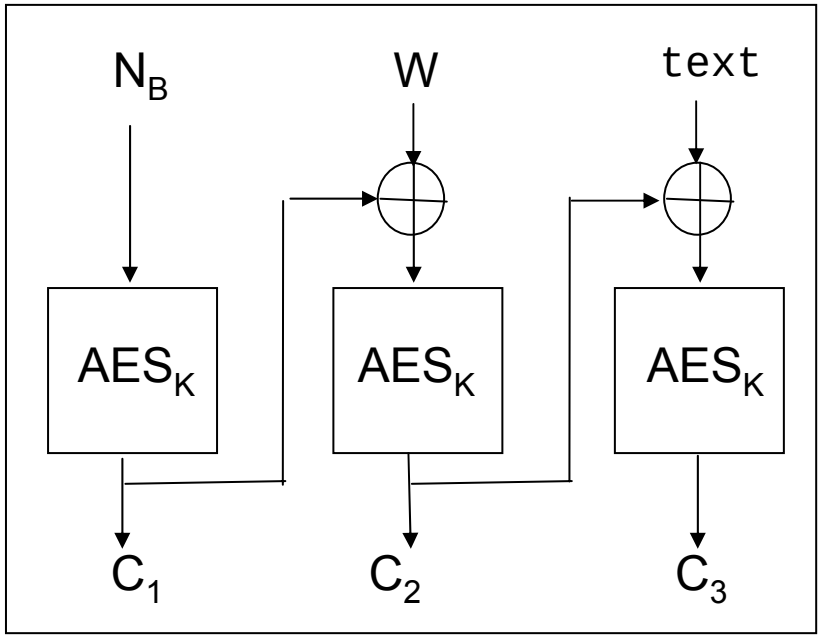
- Alice and Bob share a key K of an encryption algorithm.
- Alice has a message M, she wants to transmit to Bob.
- Bob wants verify the freshness of M and liveness of Alice



- It is necessary that the algorithm E_K offers data-integrity. If confidentiality is not needed then better to use a message authentication algorithm.

Non-integrity of CBC encryption

- Bob wants to verify the liveness of Alice's love and receive a fresh new key
- Alice's message M = W || "I love you" , where W is a128-bit key
- Encryption is CBC with 128-bit block cipher (AES)
- N_B is a 128-bit value; $(C_1, C_2, C_3) = E_K(N_B, M)$




```
text = 49 20 6c 6f 76 65 20 79 6f 75
   Δ = 00 00 04 0e 02 00 00 00 00 00
text'= 49 20 68 61 74 65 20 79 6f 75
```

- Malice changes the second ciphertext block to $C_2' = C_2 \oplus \Delta$
- After decryption Bob reads
  M' = W' || "I hate you"
  where W' is a random 128-bit value

The Use of Random Numbers

- Random numbers are an essential ingredient in most (if not all) cryptographic protocols: there is no security without *apparent randomness* and *unpredictability*; things must look random to an external observer.
- Cryptographic keys
  - symmetric keys
  - (private) keys for asymmetric cryptosystems
- Cryptographic nonces (= **n**umbers used **once**) to guarantee freshness
  - random numbers with some additional properties

Random and pseudorandom numbers

Random numbers are characterized using the following statistical properties:

- Uniformity: Random numbers are uniformly distributed
- Independence: generated random numbers cannot be derived from other generated random numbers
- Generated using physical devices, e.g, quantum random number generator

Pseudorandom numbers are non-random numbers that cannot be distinguished from random numbers:

- Statistical distribution of a sample of certain (large) size cannot be distinguished from the uniform distribution
- Independent-looking: pseudorandom numbers should be unpredictable: given a sequence of previously generated pseudorandom numbers nothing cannot be said about the future terms of the sequence
- Generated using deterministic algorithms from a short truly random or pseudorandom seed.

Counter Mode PRNG

Also known as Cyclic Encryption (Meyers 1982). It consists of a counter with period N and an encryption algorithm with a secret key.

IV Initial value of the counter C

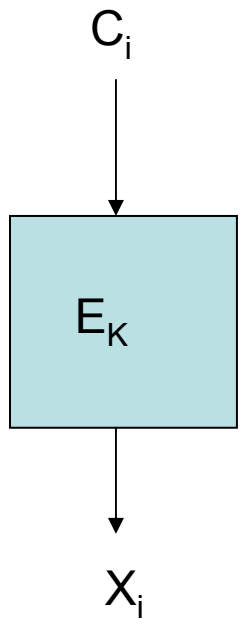
K Key of the block cipher encryption function E_K

X_i i-th pseudorandom number output

$C_0 = IV;$

$C_i = C_{i-1}+1;$

$X_i = E_K(C_i),\ i = 1,2,\ldots$



The period is N. If the length of the counter is less than the block size of E_K then all generated numbers within one period are different.

Blum-Blum-Shub

- Cryptographically provably secure PRNG
- Very slow, output 1 pseudorandom bit per one modular squaring modulo a large integer

p, q two different large primes; $p = q = 3 \pmod 4$

n modulus, $n = pq$

s secret seed; set $x_0 = s^2 \bmod n$

x_i i-th intermediate number

B_i i-th output bit

For i = 1,2,…

$x_i = (x_{i-1})^2 \bmod n$

$B_i = x_i \bmod 2$

Distribution of symmetric keys

Distribution of shared symmetric keys for A and B; using one of the following options:

1. Physically secured

- A selects or generates a key and delivers it to B using some physically secure means
- A third party C selects a key and delivers it to A and B using some physically secure means

2. Key distribution using symmetric techniques

- If A and B have a shared secret key, A can generate a new key and send it to B encrypted using the old key
- If C is already using a shared secret key K_1 with A and a second key K_2 with B, then C can generate a key and send it encrypted to A and B.

3. Key management using asymmetric techniques

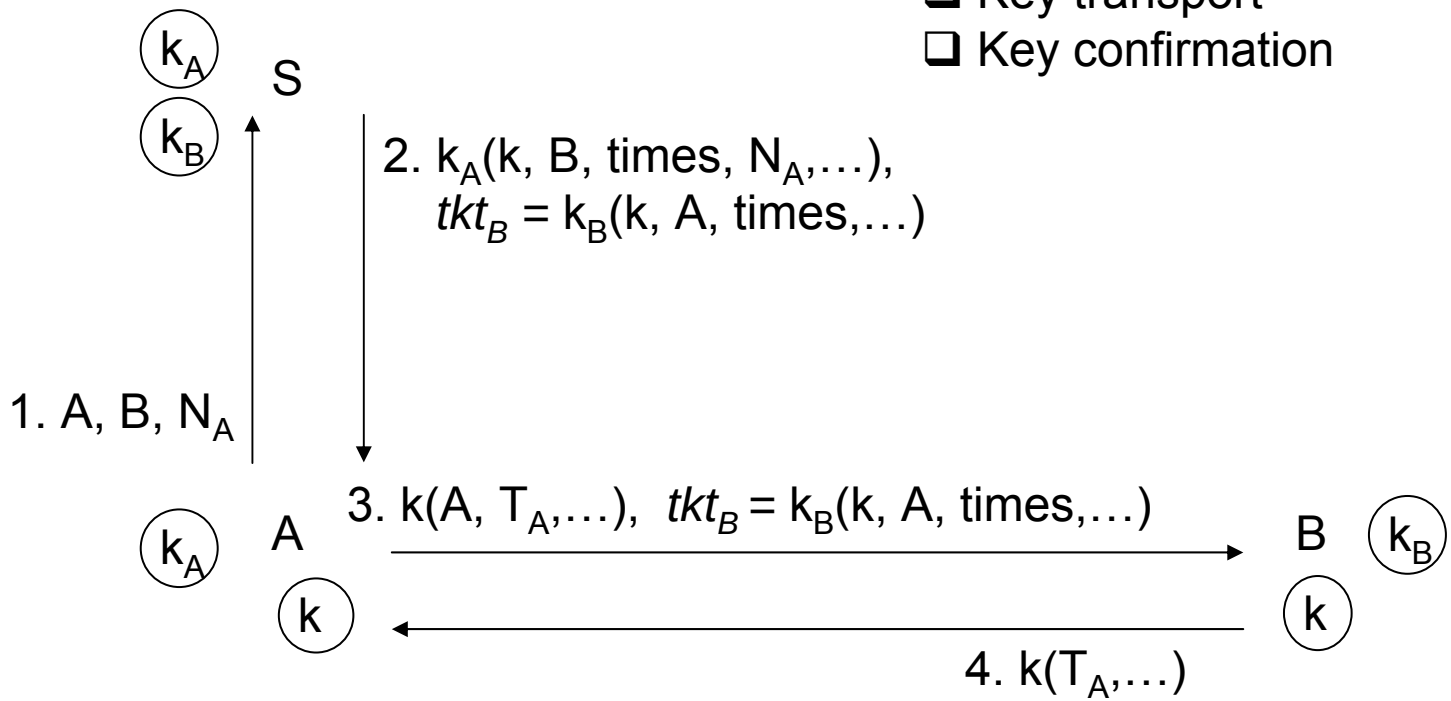
- If Party A has a public key of B, then A can generate a key and send it to B encrypted using a public key
- If party C has the public key of A and the public key of B, it can generate a key and send it to A and B encrypted using their public keys.

Key Hierarchy

1. Master Keys
   - long term secret keys
   - used for authentication and session key set up
   - Distributed using physical security or public key infrastructure
2. Session Keys
   - short term secret keys
   - used for protection of the session data
   - distributed under protection of master keys
3. Separated session keys
   - short term secrets
   - to achieve cryptographic separation: Different cryptographic algorithms should use different keys. Weaknesses in one algorithm should not endanger protection achieved by other algorithms.
   - derived from the main session key

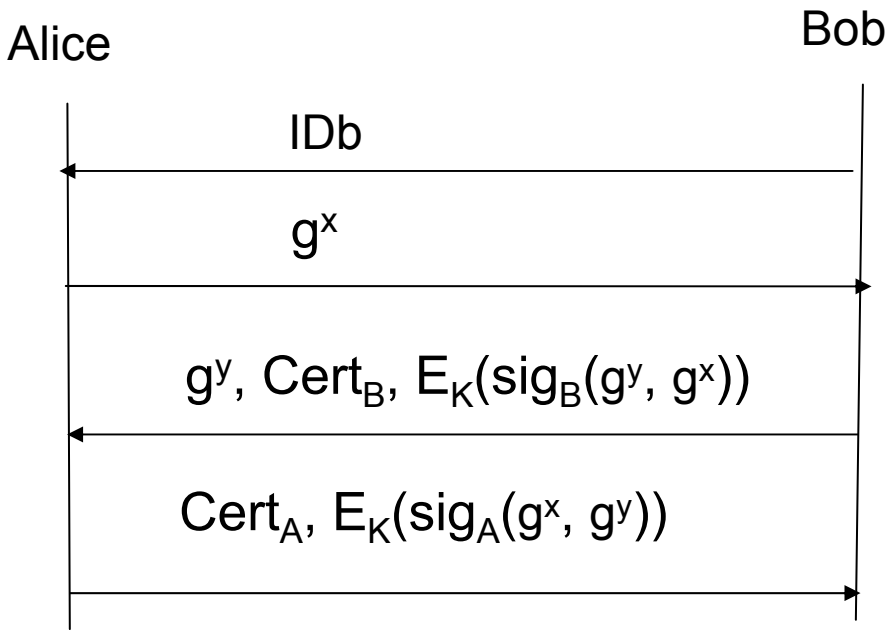
Example: Kerberos

- Prior enrollment with server
- Timestamps to ensure freshness
- Key transport
- Key confirmation



Station-to-Station (STS) Protocol: Authenticated Diffie-Hellman

- Provides *perfect forward secrecy (PFS)*: compromise of long term private keys does not compromise past session keys
- *PFS* requires the use of public key cryptography



Distribution of Public Keys

- Public announcement
  - Just appending one's public key, or the fingerprint (hash) of the public key in one's signed email message is not secure
  - PGP public key fingerprints need to be truly authenticated based on face-to-face or voice contact
- Publicly available directory
  - An authorized directory, similar to phone directory that is published in print
- Public-key Authority
  - Public keys obtained from an online service. Communication needs to be secured.
- Public-key Certificates
  - Public keys bound to user's identities using a certificate signed by a Certification Authority (CA)

CA and Registration Authority

Certification Authority

- E.g. in Finland: Population Register Center
- The certificate is stored in the subject's Electronic Identity Card

Registration Authority

- Identifies the user based on user's true identity and establishes a binding between the public key and the subject's identity

Management of private keys

- Private keys generated by the user
- Private key generated by a trusted authority
- Private key generated inside a smart card from where it is never taken out. The public key is taken out.

Certificate Revocation List

- Black list for lost or stolen private keys
- CRL must be available online for certificates with long validity period

Pretty Good Privacy

- Email encryption program
- Bottom–up approach to the distribution of trust
- Each user acts as his/her own CA and signs the public keys of other users
- User can accept authenticity of a public key based on recommendation by a third trusted user
- RSA public key encryption used for distribution of session keys *)
- Digital signatures produced by RSA or DSA signature algorithms
- Hash functions are MD5 and SHA-1
- Symmetric encryption performed using IDEA in CFB mode (self-synchronising stream cipher)
- Public keys held in ”Key-ring”
- Revocation of public keys is a problem

*) A data encryption protocol, where the data is encrypted using symmetric encryption, and the symmetric encryption key is encrypted using public key encryption, is called as ”hybrid encryption”