Lecture 4a
Symmetric Encryption III

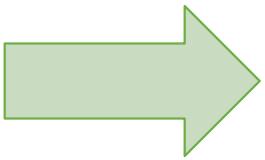
Stefan Dziembowski
www.crypto.edu.pl/Dziembowski

University of Warsaw

24.10.18

version 1.0

Plan



1. Substitution-permutation networks
2. Cascade ciphers
3. Practical considerations

Substitution-permutation networks

Based on the ideas of **Claude Shannon** (1916–2001) from **1949**.

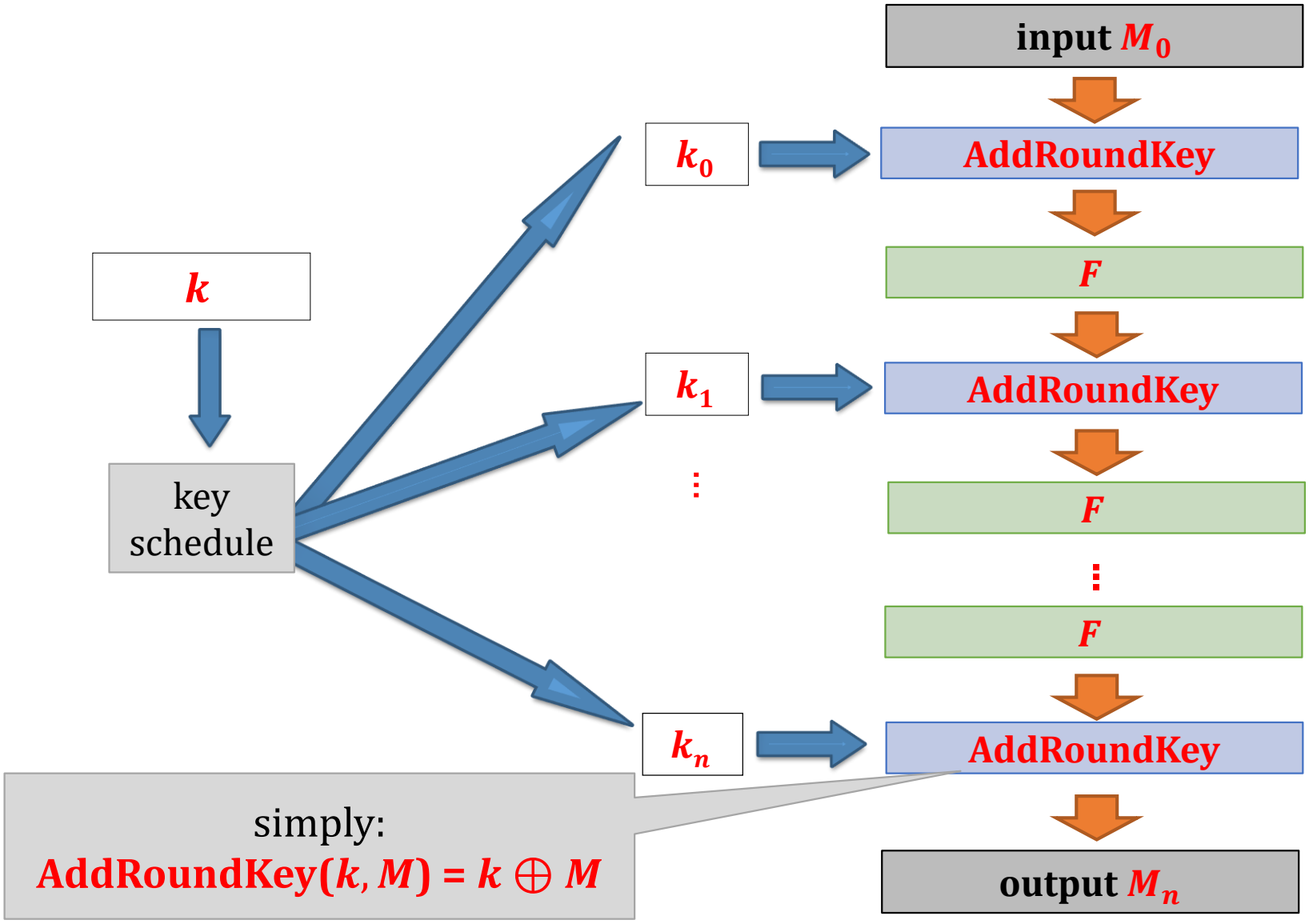
Used in **AES (Rijndael), 3-Way, SAFER, SHARK, Square**...

Advanced Encryption Standard (AES)

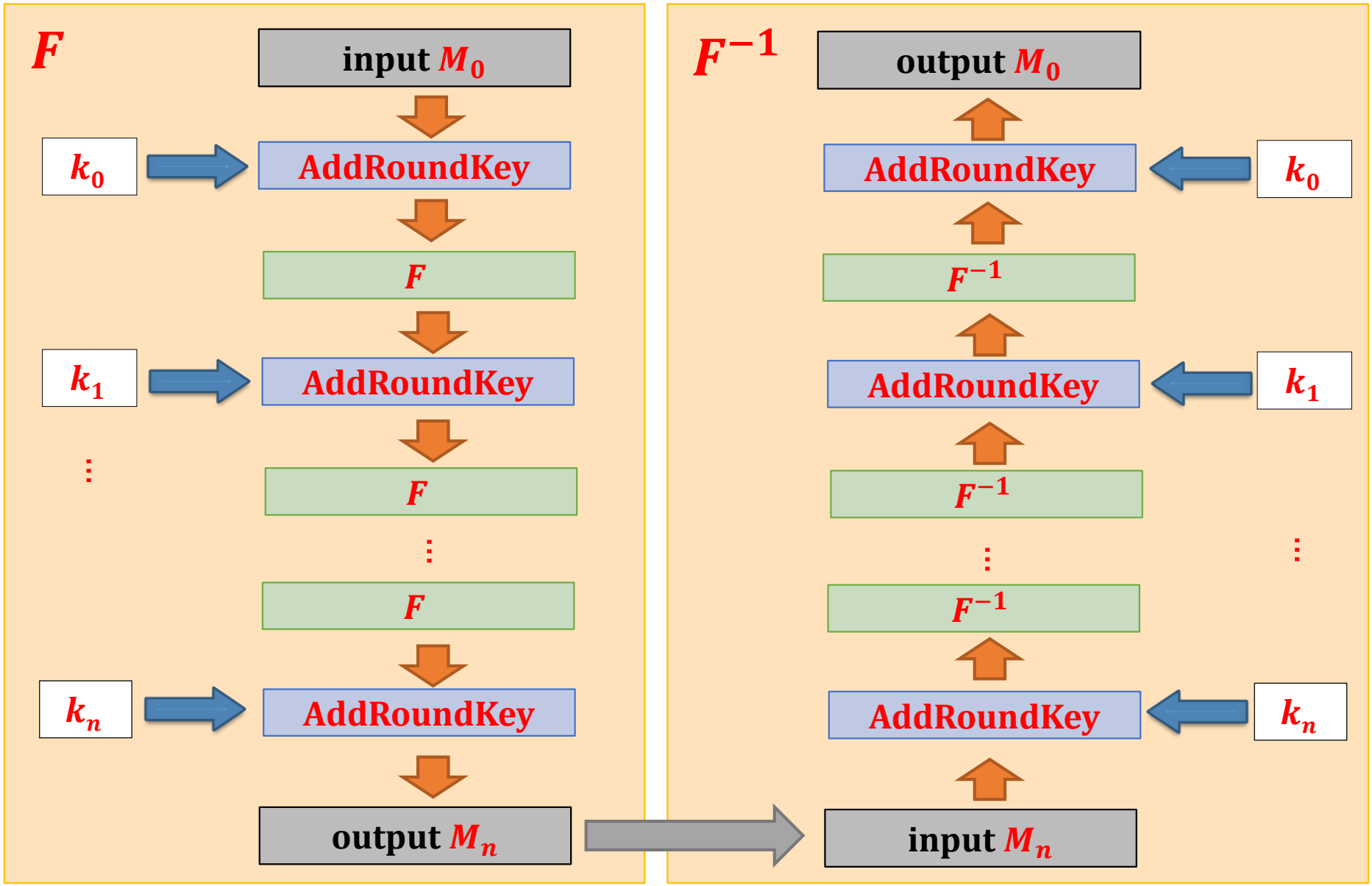
- Competition for **AES** announced in **January 1997** by the US **National Institute of Standards and Technology** (NIST)
- **15** ciphers submitted
- **5** finalists: **MARS, RC6, Rijndael, Serpent**, and **Twofish**
- **October 2, 2000**: **Rijandel** selected as the winner.
- **November 26, 2001**: **AES** becomes an official standard.

- Authors : **Vincent Rijmen**, **Joan Daemen** (from Belgium)
- **Key sizes**: **128**, **192** or **256** bit, **block size**: **128** bits

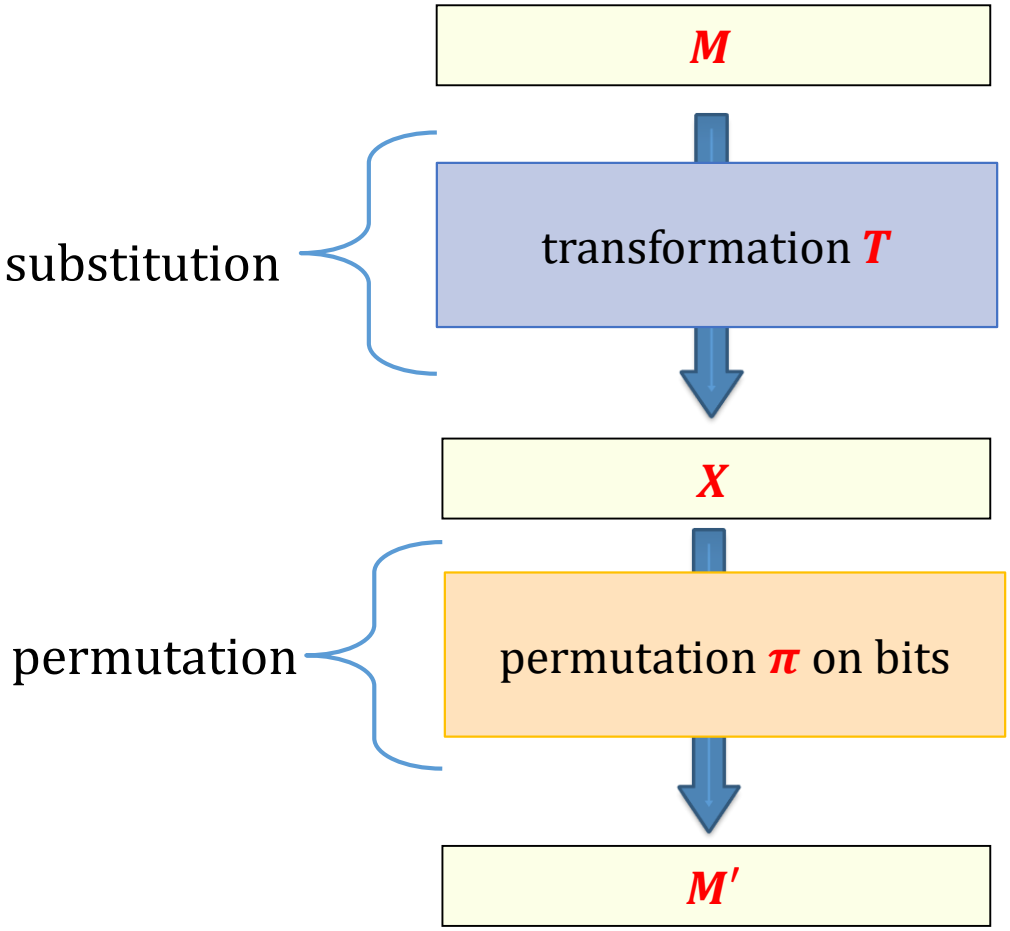
Substitution-permutation networks



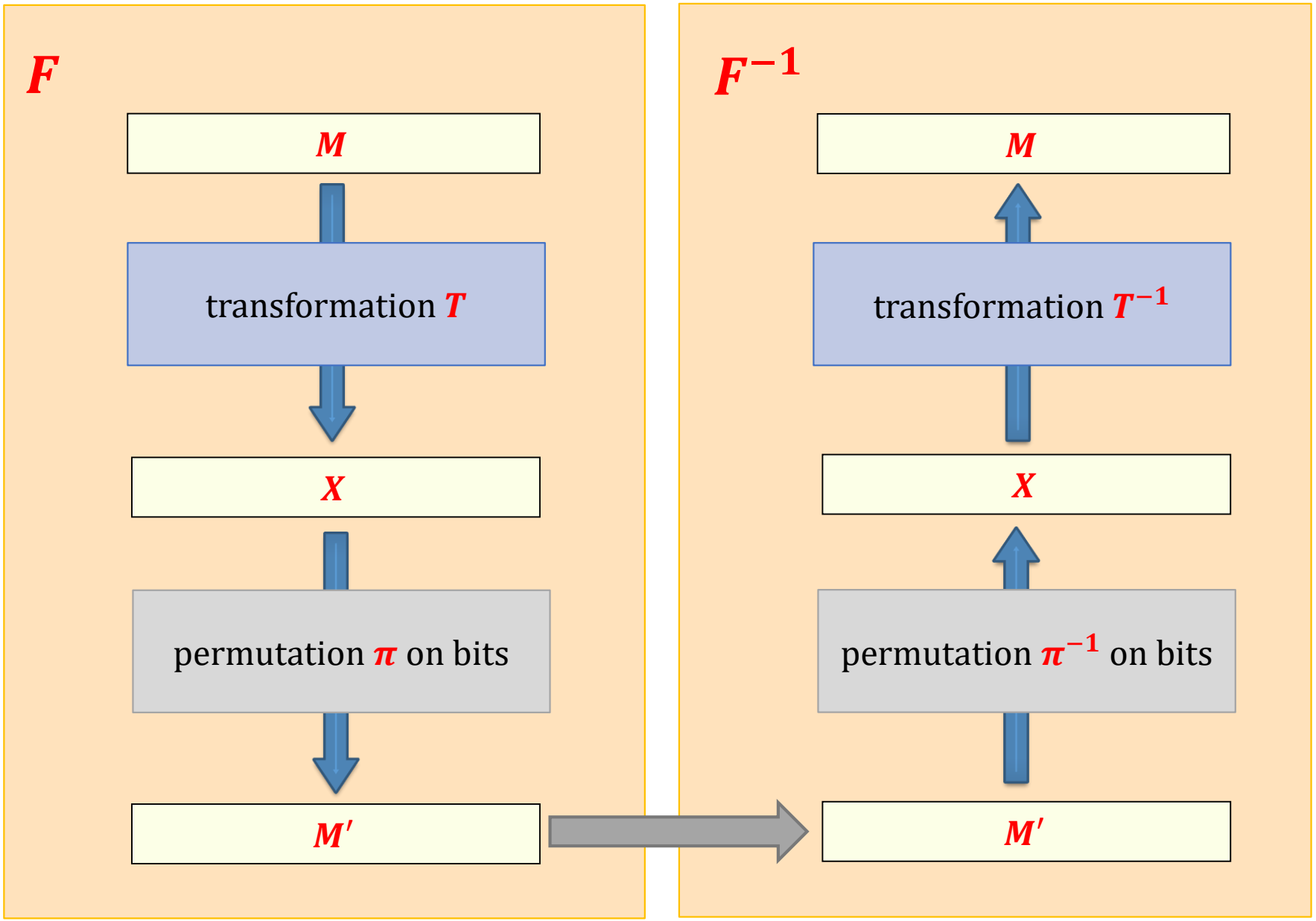
To invert: invert the order and apply F^{-1} instead of F.



A construction of F

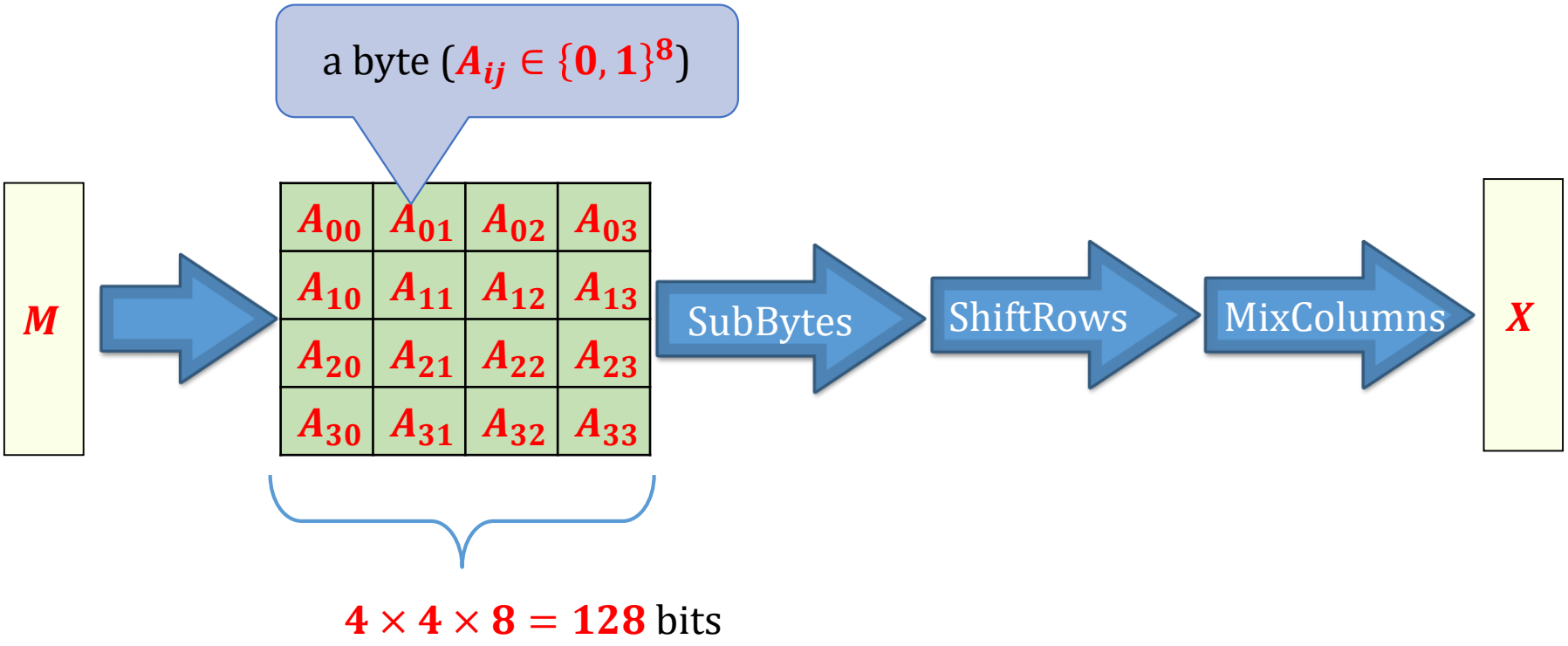


How to construct F^{-1}?



Transformation T in AES

In AES M is represented as a 4×4-matrix of bytes.



Main challenge

This transformation needs to be **invertible**...

On the other hand: it cannot be “too simple”.

AES idea: use **finite field algebra**.

Groups

A **group** is a set G along with a binary operation $\circ$ such that:

- [**closure**] for all $g, h \in G$ we have $g \circ h \in G$,
- there exists an **identity** $e \in G$ such that for all $g \in G$ we have
$$e \circ g = g \circ e = g,$$
- for every $g \in G$ there exists an **inverse of**, that is an element h such that
$$g \circ h = h \circ g = e,$$
- [**associativity**] for all $g, h, k \in G$ we have
$$g \circ (h \circ k) = (g \circ h) \circ k$$
- [**commutativity**] for all $g, h \in G$ we have
$$g \circ h = h \circ g$$

if this holds, the group is called **abelian**

Additive/multiplicative notation

Convention:

[**additive notation**]
If the groups operation is denoted with $+$, then:
the inverse of g is denoted with $-g$,
the neutral element is denoted with 0,
$g + \cdots + g$ (n times) is denoted with ng.

[**multiplicative notation**]
If the groups operation is denoted "$\times$" or "$\cdot$", then:
sometimes we write gh instead of $g \cdot h$,
the inverse of g is denoted g^{-1} or $1/g$.
the neutral element is denoted with 1,
$g \cdot \cdots \cdot g$ (n times) is denoted with g^n
$(g^{-1})^n$ is denoted with g^{-1}.

Fields

$(F, +, \times)$ is a **field** if

- $(F, +)$ is an **additive group** with **neutral element 0**
- $(F \setminus \{0\}, \times)$ is a **multiplicative group**
- **Distributivity of multiplication over addition**:
  for all $a, b, c \in F$, we have
  $$a \times (b + c) = a \times b + a \times c$$

How to define a “field over bytes”?

A very simple additive group over $\{0,1\}^n$:

$$(\{0,1\}^n, +)$$

where

$$(a_1, \ldots, a_n) + (b_1, \ldots, b_n) = (a_1 \oplus b_1, \ldots, a_n \oplus b_n)$$

xor

Extremely efficient to implement.

How to extend $(\{0,1\}^n, +)$ to a field?

"**Galois fields**" $\mathbf{GF}(2^n)$:

Represent each $(a_0, \ldots, a_{n-1}) \in \{0,1\}^n$ as a polynomial A over Z_2 of degree $n-1$.

$$A(x) = a_0 + a_1 x + \cdots + a_{n-1} x^{n-1}$$

Note: if $(b_1, \ldots, b_{n-1})$ is represented as a polynomial $B(x) = b_0 + b_1 x + \cdots + b_{n-1} x^{n-1}$, then

$$(A + B)(x) = \sum_{i=0}^{n-1} (a_i + b_i) \cdot x^i \pmod 2$$

Observe: this is the same as the **xor** operation on bits.

How to multiply?

Suppose:

$$A(x) = a_0 + a_1 x + \cdots + a_{n-1} x^{n-1}$$
$$B(x) = b_0 + b_1 x + \cdots + b_{n-1} x^{n-1}.$$

Then $A \times B$ is a polynomial of max degree $2n - 2$.

How to reduce this degree?

do not exits non-constant polynomials q_0, q_1 such that $p = q_0 \cdot q_1$

Take an **irreducible polynomial** p of degree n, and compute

$$C = A \cdot B \pmod{p}$$

Then C is a polynomial of degree $n - 1$.

Write $C(x) = c_0 + c_1 x + \cdots + c_{n-1} x^{n-1}$.

Define: $(a_0, \ldots, a_{n-1}) \times (b_1, \ldots, b_{n-1}) = (c_1, \ldots, c_{n-1})$

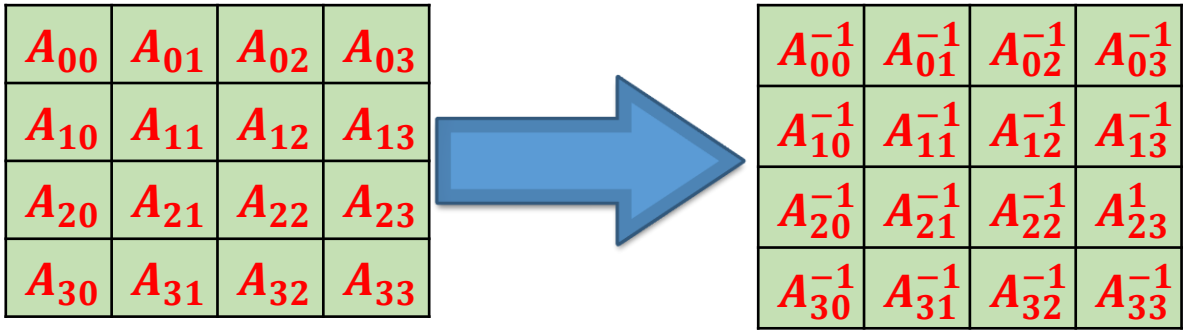
Fact from algebra: this defines a field.

AES field

AES uses $\mathbf{GF(2^8)}$, where the polynomial $\boldsymbol{p}$ is defined as

$$\boldsymbol{p(x) = 1 + x + x^3 + x^4 + x^8}$$

First step of SubBytes



Invert every $\boldsymbol{A_{ij}}$ (in the multiplicative group of $\mathbf{GF}(\boldsymbol{2^n})$).

Convention: $\boldsymbol{0^{-1} = 0}$.

Another observation

We can look at Z_2^n as a **linear space**.

AES defines the following affine transformation:

$$\varphi(x_1, \dots, x_8) :=$$

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

$*$

x_1
x_2
x_3
x_4
x_5
x_6
x_7
x_8

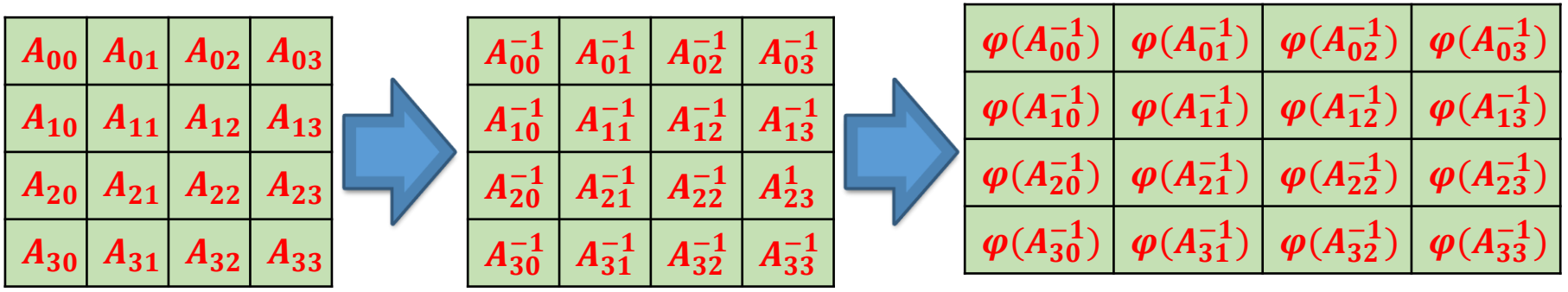
$+$

1
1
0
0
0
1
1
0

Advantages:

- **SubBytes** is not an operation only in $\mathbf{GF}(2^n)$.
- The constant vector is chosen in such a way that there are no
  - fixpoints $\varphi(X) = X$
  - anti-fixpoints $\varphi(X) = \overline{X}$
- φ is invertible.

Complete SubBytes



Observe that

$$A_{ij} \mapsto A_{ij}^{-1} \mapsto \varphi(A_{ij}^{-1})$$

is invertible (since $A_{ij} \mapsto A_{ij}^{-1}$ and φ are invertible)

ShiftRows

Cyclic shifts of rows:

B_{00}	B_{01}	B_{02}	B_{03}
B_{10}	B_{11}	B_{12}	B_{13}
B_{20}	B_{21}	B_{22}	B_{23}
B_{30}	B_{31}	B_{32}	B_{33}

B_{00}	B_{01}	B_{02}	B_{03}
B_{11}	B_{12}	B_{13}	B_{10}
B_{22}	B_{23}	B_{20}	B_{21}
B_{33}	B_{30}	B_{31}	B_{32}

shift **1** cell left

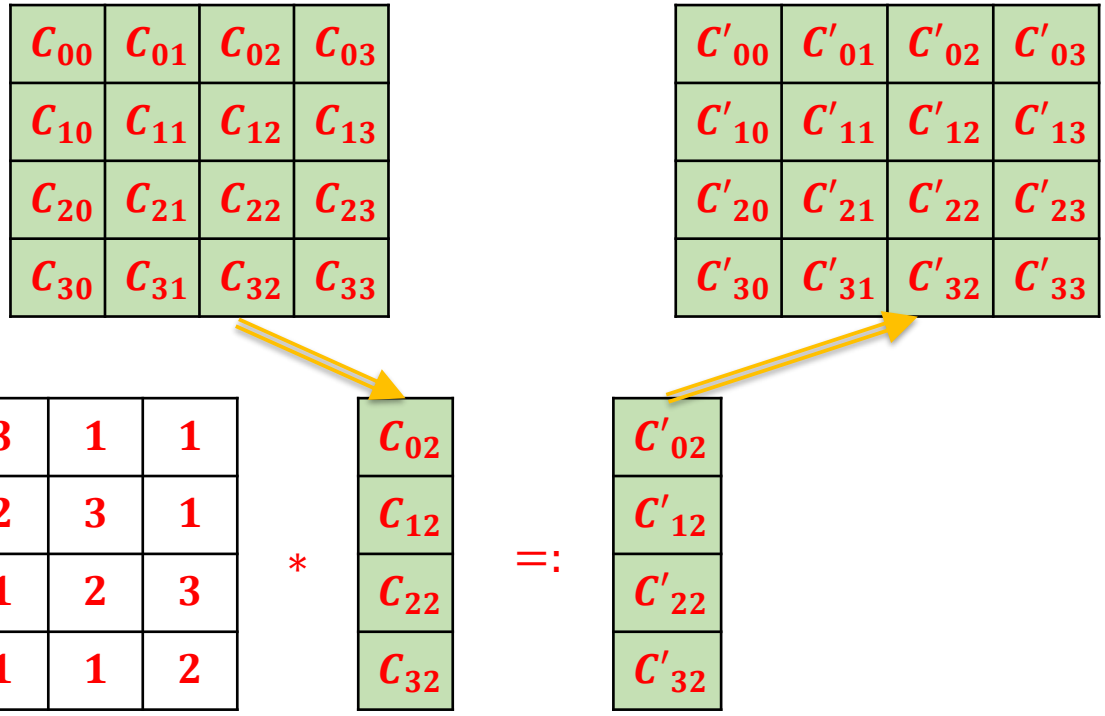
shift **2** cells left

shift **3** cells left

Clearly: **ShiftRows** is invertible.

MixColumns

Multiply every column by a matrix $\boldsymbol{M}$ (in $\mathbf{GF}(\mathbf{2^8})$):



Clearly $\boldsymbol{M}$ is invertible, so the whole operation also is.

AES construction – more details

Concrete parameters:

key size: 128, **192** or **256** bit,
block size: **128** bits

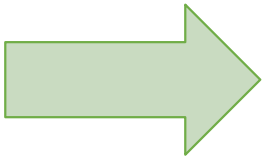
We omit the description of the key schedule.

Security:

best known attack: **biclique attacks [Bogdanov, Khovratovich, and Rechberger, 2013]**:

- **AES-128** complexity $2^{126.1}$,
- **AES-192** complexity $2^{189.7}$,
- **AES-256** complexity $2^{254.4}$.

Plan



1. Substitution-permutation networks
2. Cascade ciphers
3. Practical considerations

An idea

The main problem of **DES** is the short key!

Maybe we could increase the length of the key?

But how to do it?

Idea: cascade the ciphers!

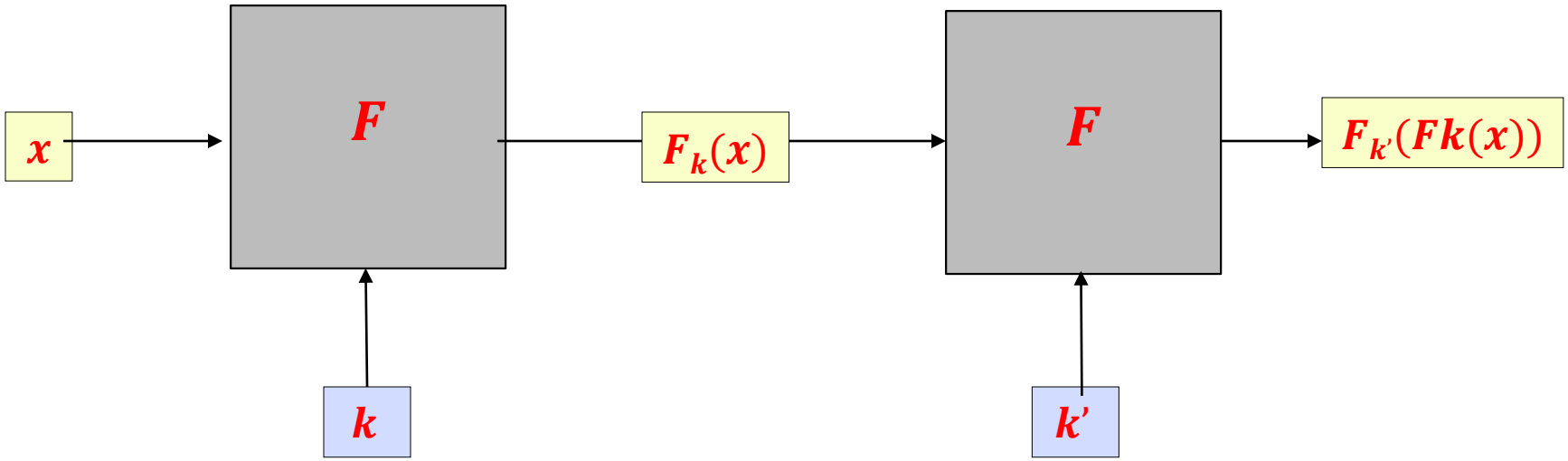
We now describe it in an abstract way (for any block cipher F)

How to increase the key size?

Cascade encryption.

For example **double encryption** is defined as:

$$F'_{(k,k')}(x) := F_{k'}(F_k(x))$$



Does it work?

- Double encryption – not really...
- Triple encryption is much better!

Double encryption

n = block length = key length

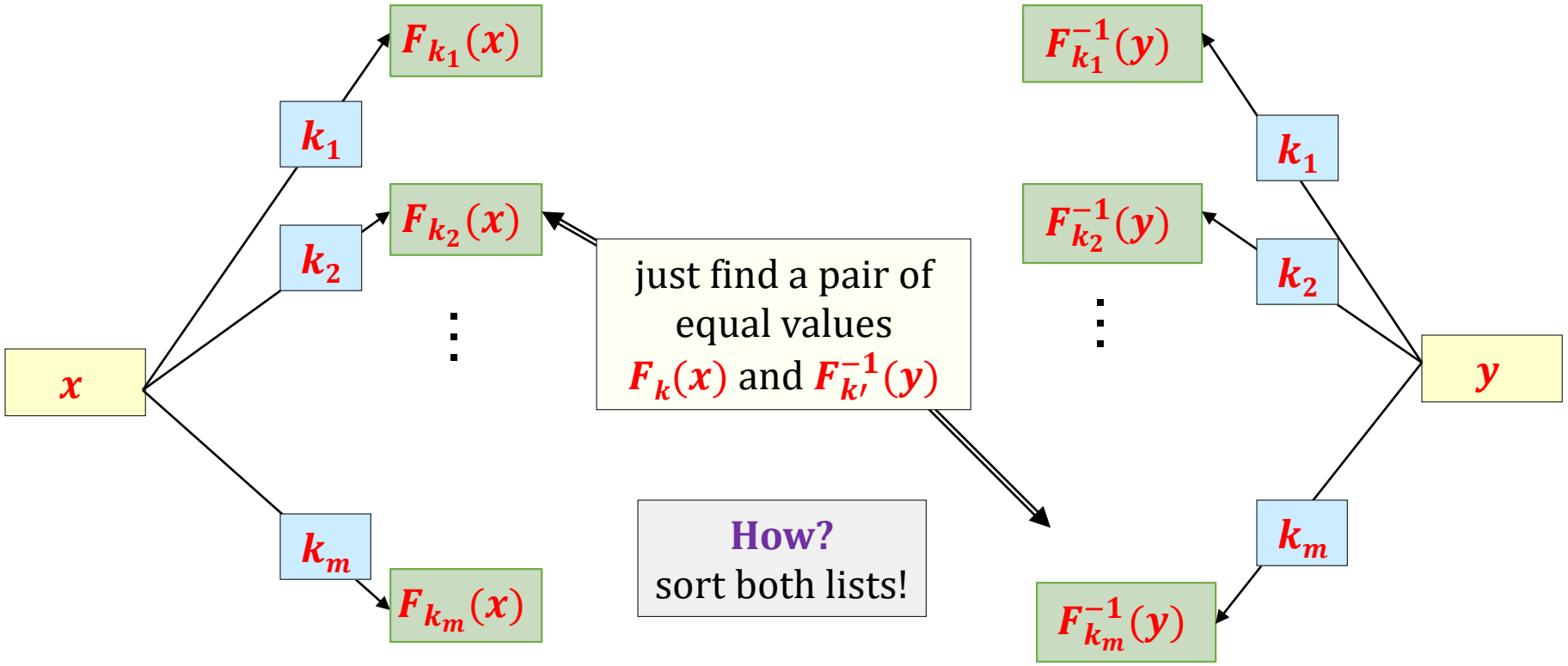
Double encryption can be broken using

- time $O(2^n)$,
- space $O(2^n)$,
- and **3 (plaintext,ciphertext)** pairs.

The attack is called "**meet in the middle**".

Meet-in-the middle attack – the idea

Goal: Given (x, y) find (k, k') such that $y = F_{k'}(F_k(x))$ $m = 2^n$



Meet-in-the middle attack – the algorithm

Goal: Given (x, y) find (k, k') such that $y = F_{k'}(F_k(x))$.

Algorithm:

1. For each k compute $z = F_k(x)$ and store (z, k) in a list L.
2. For each k compute $z = F_k^{-1}(y)$ and store (z, k') in a list L'.
3. Sort L and L' by their first components.
4. Let S denote the list of all pairs all pairs (k, k') such that for some z we have
   $(z, k) \in L$ and $(z, k') \in L'$.
5. Output S.

Meet-in-the middle attack – analysis [1/2]

Suppose: n = block length = key length, x and y are fixed

P (**a random pair** (k, k') **satisfies** $y = F_{k'}(F_k(x))) \approx 2^{-n}$

why? because $F_{k'}(F_k(x))$ can take 2^n values

The number of all pairs (k, k') is equal to 2^{2n}. Therefore

$$E(|S|) \approx 2^n \cdot 2^{-n} = 2^n.$$

So we have around 2^n "candidates" for the correct pair (k, k').

How to eliminate the "false positives"?

For each "positive" check it against another pair (x', y').

Meet-in-the middle attack – analysis [2/2]

The probability that (k, k') is a false positive for (x, y) and for (x', y') is around

$$2^{-n} \cdot 2^{-n} = 2^{-2n}.$$

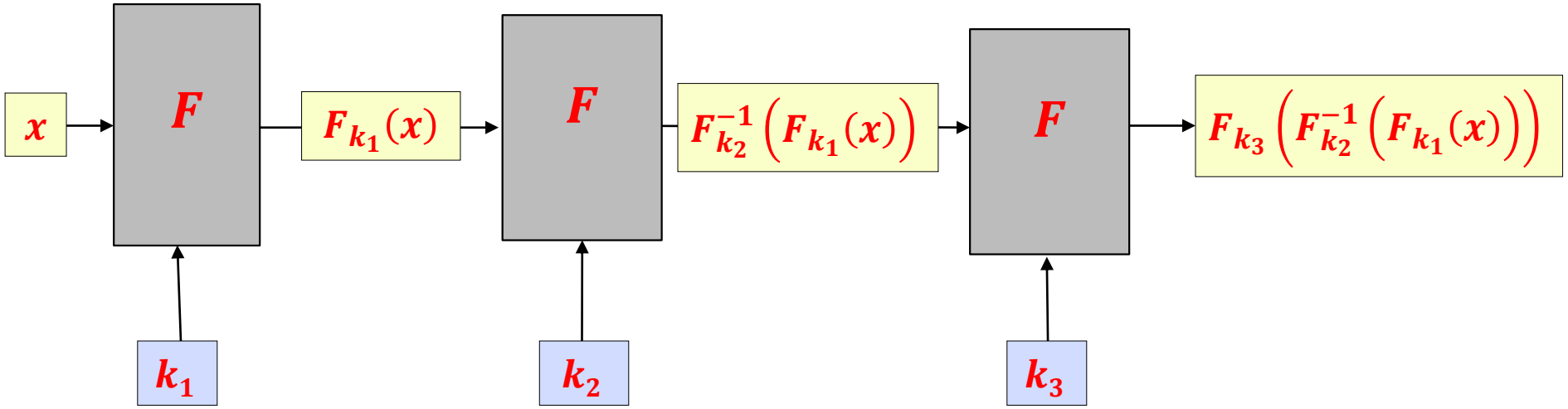
Hence, the expected number of "false positives" is around

$$2^{2n} \cdot 2^{-2n} = 1$$

An additional pair (x'', y'') allows to eliminate the false positive.

A much better idea: triple encryption

$$F_{(k_1,k_2,k_3)}(x) := F_{k_3}\left(F_{k_2}^{-1}\left(F_{k_1}(x)\right)\right)$$



Sometimes $k_1 = k_3$.

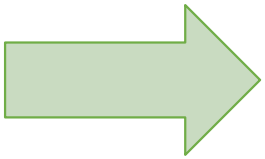
Triple DES (3DES) is a standard cipher.

Disadvantages:

- rather slow,
- small block size.

Plan

1. Substitution-permutation networks
2. Cascade ciphers
3. Practical considerations



Benchmarks

	Algorithm	MiB/Second	Cycles Per Byte
stream	Salsa20	643	2.7
	Sosemanuk	727	2.4
block	AES/CTR (128-bit key)	139	12.6
	AES/CTR (192-bit key)	113	15.4
	AES/CTR (256-bit key)	96	18.2
	AES/CBC (128-bit key)	109	16
	AES/CBC (192-bit key)	92	18.9
	AES/CBC (256-bit key)	80	21.7
	DES/CTR	32	54.7
	DES-EDE3/CTR	13	134.5

Source: www.cryptopp.com/benchmarks.html
All algorithms coded in C++, compiled with Microsoft Visual C++ 2005 SP1 (whole program optimization, optimize for speed), and ran on an Intel Core 2 1.83 GHz processor under Windows Vista in 32-bit mode.

Hardware implementations of AES

(taken from J Daemen, V Rijmen The design of Rijndael, 2001):

Example of a hardware record:

ASIC. H. Kuo and I. Verbauwhede report a throughput of 6.1 Gbit/s, using 0.18 μm standard cell technology [55] for an implementation without pipelining. Their design uses 19 000 gates. B. Weeks et al. report a throughput of 5 Gbit/s [91] for a fully pipelined version. They use a 0.5 μm standard cell library that is not available outside NSA.

Stream ciphers vs. block ciphers

- Stream ciphers are a **bit more efficient**.
- But they appear to be "**less secure**".
- It is easier to misuse them (use the same stream twice).
- If you encrypt a stream of data you can always use a block cipher in a **CTR** mode.
- Probably at the moment **block ciphers are a better choice** for most of the applications.