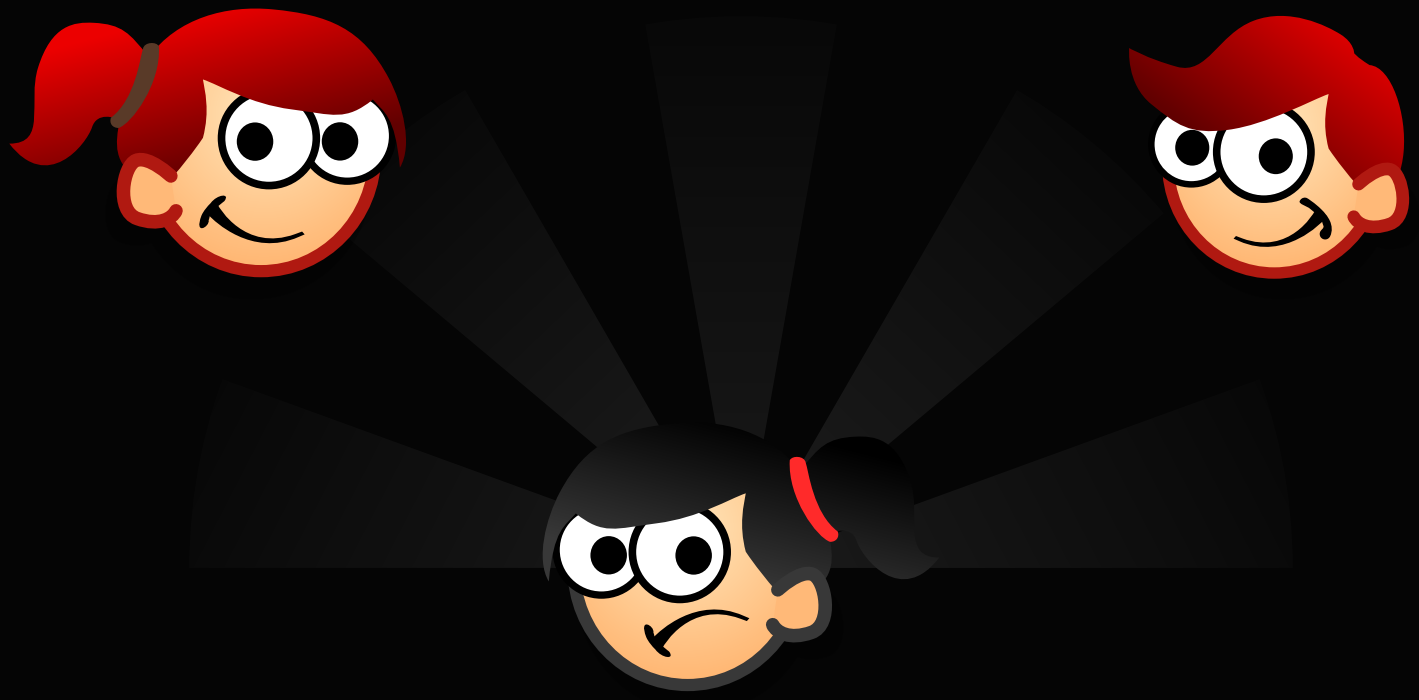
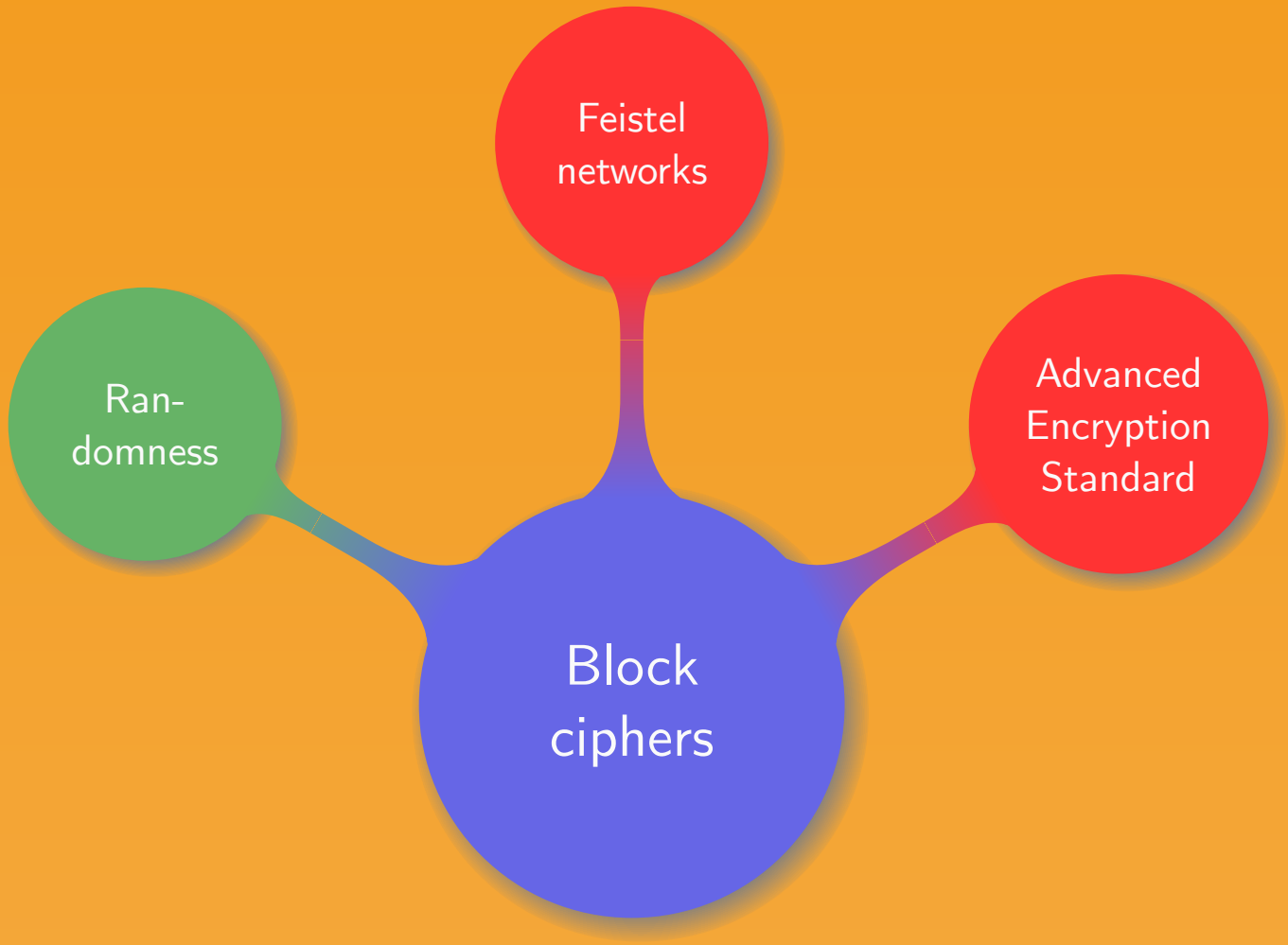


Introduction to Cryptography

2. Block ciphers

Manuel – Summer 2019

Chapter organisation



Block ciphers

A *block cipher* is composed of two functions, inverse of each other:

$$E : \{0,1\}^n \times \{0,1\}^k \to \{0,1\}^n \qquad D : \{0,1\}^n \times \{0,1\}^k \to \{0,1\}^n$$
$$(P, K) \mapsto C \qquad (C, K) \mapsto P$$

where n and k are the sizes of a block and the key, respectively.

Block ciphers

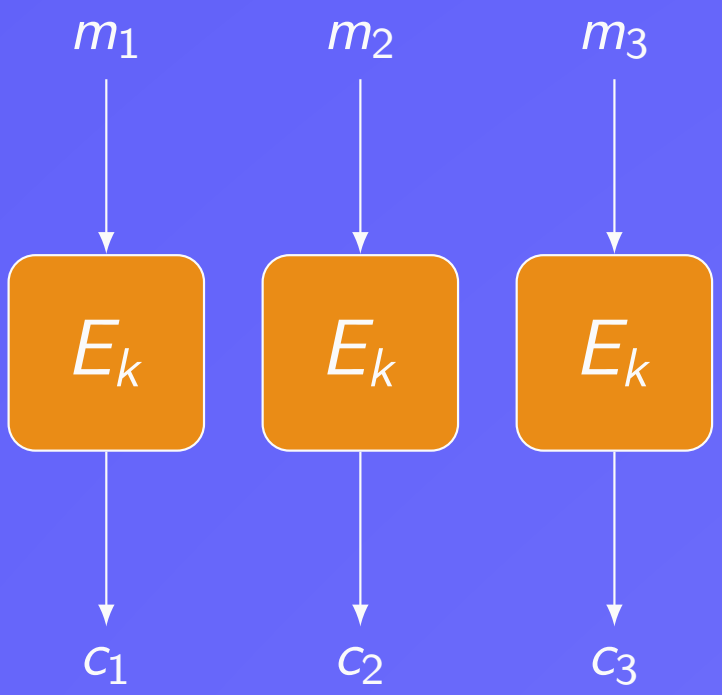
A *block cipher* is composed of two functions, inverse of each other:

$$E : \{0,1\}^n \times \{0,1\}^k \to \{0,1\}^n \qquad D : \{0,1\}^n \times \{0,1\}^k \to \{0,1\}^n$$
$$(P,K) \mapsto C \qquad (C,K) \mapsto P$$

where n and k are the sizes of a block and the key, respectively.

Goal: given a key K, design an invertible function E whose output cannot be distinguished from a random permutation over $\{0,1\}^n$.

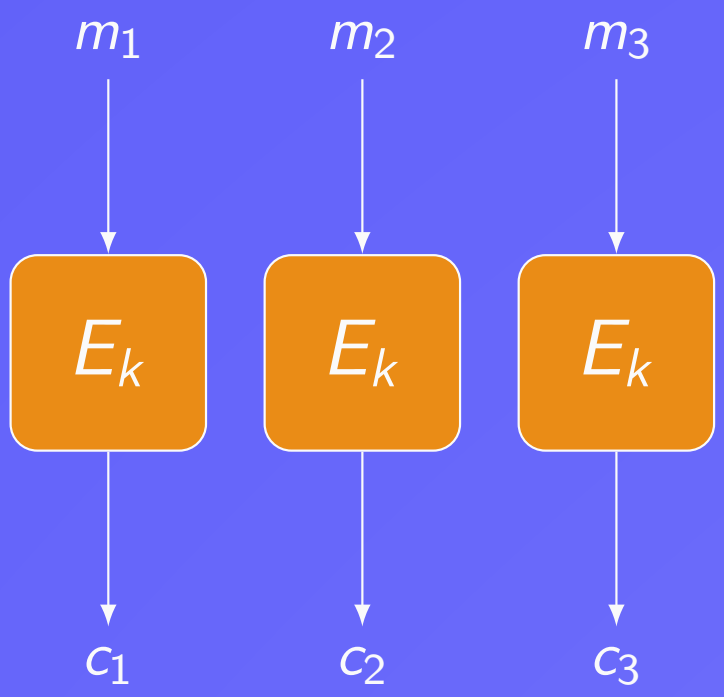
Mode ECB



Basic principle:

- Split the plaintext in blocks of size n
- Encrypt each block with a function E and a key K
- Electronic Code Block (ECB) mode

Mode ECB

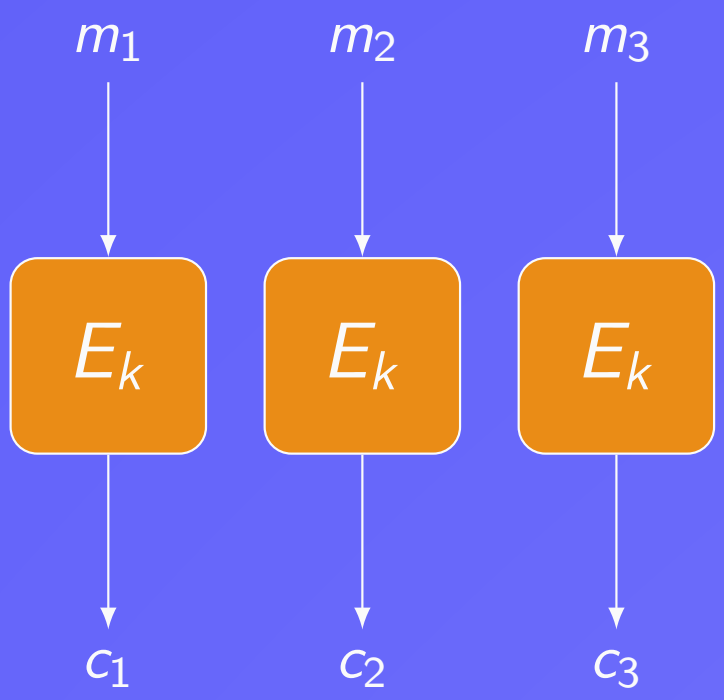


Basic principle:

- Split the plaintext in blocks of size n
- Encrypt each block with a function E and a key K
- Electronic Code Block (ECB) mode

Limitation: what if a block is repeated several times over the message?

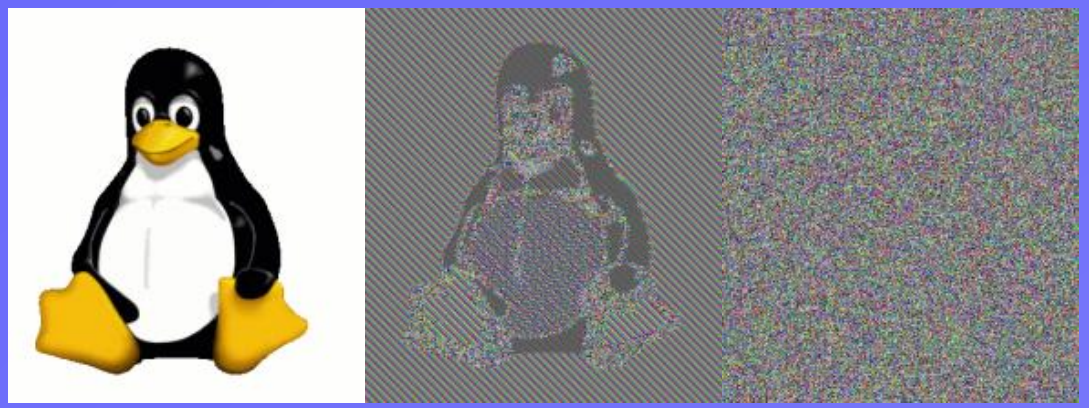
Mode ECB



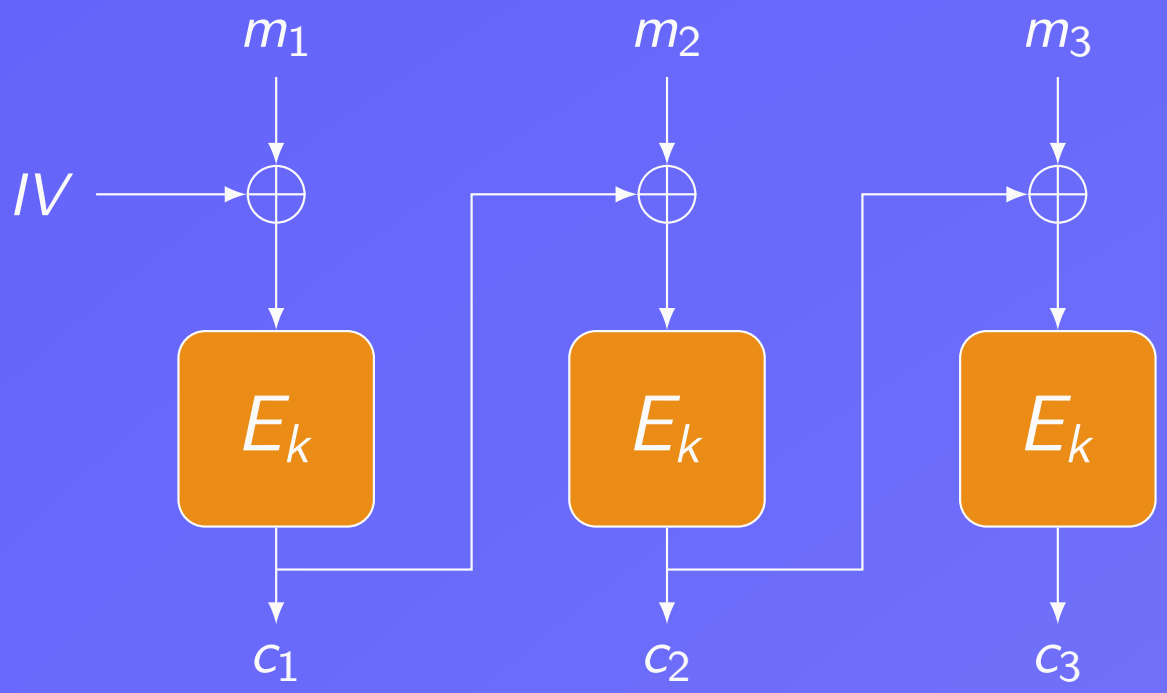
Basic principle:

- Split the plaintext in blocks of size n
- Encrypt each block with a function E and a key K
- Electronic Code Block (ECB) mode

Limitation: what if a block is repeated several times over the message?



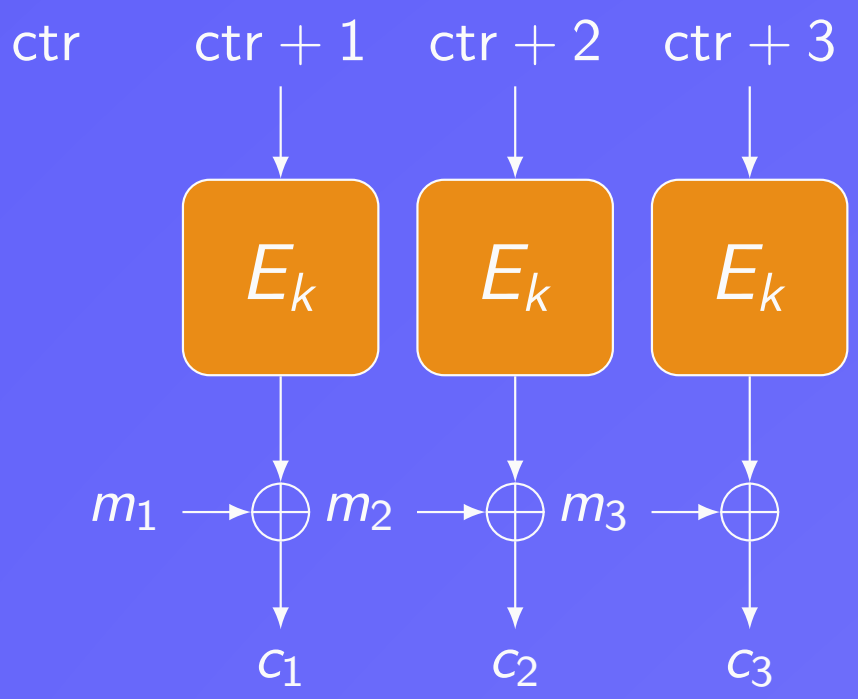
Mode CBC



Basic principle:

- Cipher Block Chaining (CBC)
- Most commonly used mode
- Uses an Initialization Vector
- Can it be parallelized?

Mode CTR



Basic principle:

- CTR stands for counter
- The counter acts like an IV
- The E_K function randomizes the counter
- Can be run in parallel

What is randomness?

```
int getRandomNumber()
{
    return 4;  // chosen by fair dice roll.
               // guaranteed to be random.
}
```

Randomness

Definition (Kolmogorov randomness)

Let x be a string.

- We say that x is *random* if and only if it is not shorter than any program that can produce it in any language.
- The *entropy* of x is the minimum number of bits necessary to describe x.

Randomness

Definition (Kolmogorov randomness)

Let x be a string.

- We say that x is *random* if and only if it is not shorter than any program that can produce it in any language.
- The *entropy* of x is the minimum number of bits necessary to describe x.

Remark.
A random string of length k cannot be compressed in any way, therefore it has entropy k

Example.
3434 3434 3434 3434 3434 3434 3434 3434
ed71 b38f 4316 6907 a8ea 75d3 c141 735f

Random bits generation

Generating true randomness is not simple:

- Toss a coin
- Measure physical phenomena that are expected to be random
- In case of a lack of entropy the output is blocked

Random bits generation

Generating true randomness is not simple:

- Toss a coin
- Measure physical phenomena that are expected to be random
- In case of a lack of entropy the output is blocked

Example.
The thermal noise from a semiconductor resistor
A nuclear decay radiation source measured by a Geiger counter

Pseudo-random bits generation

Random function from the C standard:

```c
/*  Linear congruential generator */
static unsigned long next = 1;

/* RAND_MAX assumed to be 32767 */
int rand(void) {
  next = next * 1103515245 + 12345;
  return((unsigned)(next/65536) % 32768);
}

void srand(unsigned int seed) {
  next = seed;
}
```

Pseudo-random bits generation – BBS generator

A secure method from Blum, Blum and Shub:

1. Generate two large primes p and q, both being 3 mod 4
2. Set $n = pq$
3. Choose a random integer x coprime to n
4. Define
$$\begin{cases} x_0 & \equiv & x^2 \bmod n \\ x_{i+1} & \equiv & x_i^2 \bmod n \end{cases}$$
5. At each iteration select the least significant bit of x_i

Why is BBS secure?

Can bits generated using BBS be predicted?

Problem (Quadratic Residuosity (QR))

Let $n = pq$ be the product of two primes. Given an integer y, is it a square mod n, i.e. is there an x such that $x^2 \equiv y \bmod n$?

This loose formulation will be refined in the next chapter (3.32).

Why is BBS secure?

Can bits generated using BBS be predicted?

Problem (Quadratic Residuosity (QR))

Let $n = pq$ be the product of two primes. Given an integer y, is it a square mod n, i.e. is there an x such that $x^2 \equiv y \bmod n$?

This loose formulation will be refined in the next chapter (3.32).

Strategy:

- Prove that the QR problem is hard
- If this is hard the previous bit cannot be predicted
- A sequence a pseudo-random bits generated by BBS cannot be compressed

Reminder

In order to prove that the QR problem is hard we first recall and prove few results from number theory. The goal is to prove that solving the QR problem is as hard as factoring. That is, knowing how to solve one implies knowing how to solve the other one.

Theorem (Fermat's little theorem)

Let $p \in \mathbb{N}$ and $a \in \mathbb{Z}$. If p is prime and $p \nmid a$, then

$$a^{p-1} \equiv 1 \bmod p.$$

More generally, for any $p \in \mathbb{N}$ and $a \in \mathbb{Z}$,

$$a^{p} \equiv a \bmod p.$$

Squares modulo a prime

Lemma

If $p \equiv 3 \bmod 4$ is prime, then the equation $x^2 \equiv -1 \bmod p$ has no solution.

Proof.
Suppose such an x exists. Then raising it to the power of $(p-1)/2$ and applying Fermat's little theorem (2.13) yields

$$(x^2)^{\frac{p-1}{2}} \equiv x^{p-1} \equiv 1 \bmod p.$$

On the other hand $p \equiv 3 \bmod 4$, implies $(p-1)/2$ odd and

$$(-1)^{\frac{p-1}{2}} \equiv -1 \bmod p.$$

Square roots modulo a prime

Proposition

Let $p \equiv 3 \bmod 4$ be a prime, y be an integer and $x \equiv y^{\frac{p+1}{4}} \bmod p$.

- If y has a square root mod p, then its square roots are $\pm x \bmod p$
- If y has no square root mod p, then the square roots of $-y$ are $\pm x \bmod p$

Proof.
The case $y \equiv 0 \bmod p$ being trivial, we assume $y \not\equiv 0 \bmod p$.
Applying Fermat's little theorem (2.13) we get

$$x^4 \equiv y^{p+1} \equiv y^2 y^{p-1} \equiv y^2 \bmod p. \tag{2.1}$$

Square roots modulo a prime

Proof (continued).

Since p is prime all the non zero elements have a multiplicative inverse (prop. 1.33). Therefore rewriting eq. (2.1) into

$$(x^2 - y)(x^2 + y) \equiv 0 \bmod p,$$

implies $x^2 \equiv \pm y \bmod p$. Hence at least one of y and $-y$ is a square mod p.

Suppose that both y and $-y$ are square mod p, i.e. there exist a and b such that $y \equiv a^2 \bmod p$ and $-y \equiv b^2 \bmod p$.

Then $\left(b^{-1}a\right)^2 \equiv -1 \bmod p$, that is -1 is a square mod p, contradicting lem. 2.14↯.

Hence exactly one of y and $-y$ has square roots $\pm x \bmod p$.

□

Reminder

Keeping in mind the initial goal of studying the BBS generator where the squares are computed mod $n = pq$, with both p and q congruent to 3 modulo 4, we recall the following result.

Theorem (Chinese Remainder Theorem (CRT))

Let $m_1, \dots, m_k \in \mathbb{N}\setminus\{0\}$ be pairwise relatively prime and $a_1, \dots, a_k \in \mathbb{Z}$. Then the system of congruences

$$\begin{cases} x \equiv a_1 \bmod m_1, \\ x \equiv a_2 \bmod m_2, \\ \vdots \\ x \equiv a_n \bmod m_k. \end{cases}$$

has a unique solution modulo $m = m_1 m_2 \dots m_k$.

Square roots for a composite modulus

Example.
Find x such that $x^2 \equiv 71 \bmod 77$.

As $77 = 7 \times 11$, the congruency can be rewritten

$$\begin{cases} x^2 \equiv 71 \equiv 1 \bmod 7 \\ x^2 \equiv 71 \equiv 5 \bmod 11. \end{cases}$$

As both 7 and 11 and 3 mod 4, from prop. 2.15 we derive

$$\begin{cases} x \equiv \pm 1 \bmod 7 \\ x \equiv \pm 4 \bmod 11. \end{cases}$$

Finally, by applying the CRT (2.17) the four solutions can be recombined modulo 77 such as to get

$$x \equiv \pm 15, \ \pm 29 \bmod 77.$$

From square roots to factors

In the previous example we used the factorisation of n in order to calculate the square root of x modulo n. We now show that if we know the square root then we can factorize n.

Proposition

Let n be a product of two unknown primes p and q, both being 3 mod 4. Let $x \equiv \pm a,\ \pm b \bmod n$ be the four solutions to $x^2 \equiv y \bmod n$. Then $\gcd(a - b, n)$ is a non-trivial factor of n.

Proof.

From the construction of a and b, we know that $a \equiv b \bmod p$ and $a \equiv -b \bmod q$ (or the other way around). Therefore $p|(a - b)$ while $q \nmid (a - b)$, which means that $\gcd(a - b, n) = p$. □

Remarks on the BBS generator

We showed that:

- Solving the factorization problem allows to solve the QR problem
- Solving the QR problem gives the factorization of the modulus

The previous reasoning is:

- Not a formal security reduction
- Enough to "informally" consider BBS as a secure pseudo-random number generator

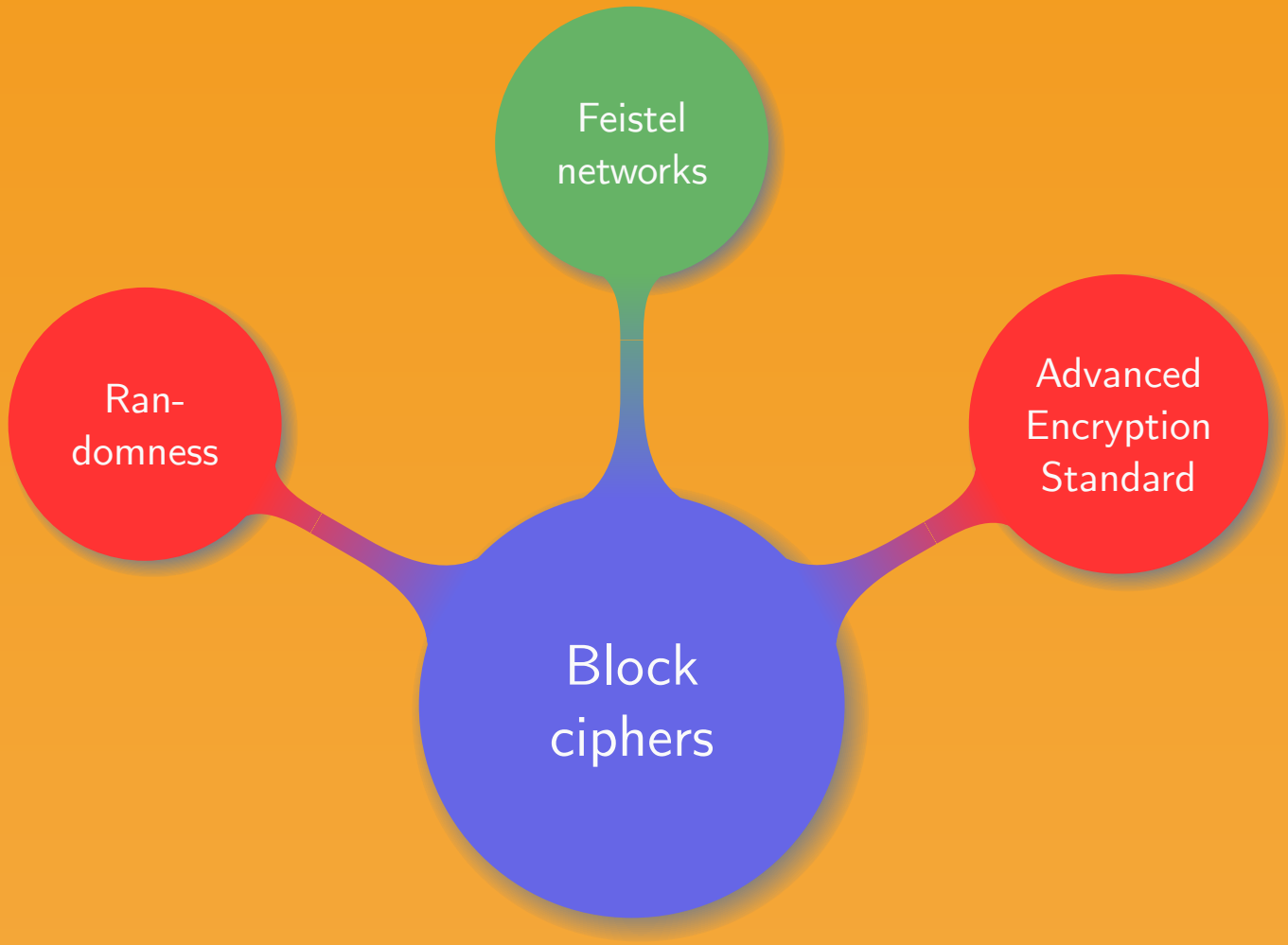
Building a block cipher

A few informal definitions:

- A *random oracle* is a "black box" that returns a truly uniform random output on an input. Submitting the same input more than once leads to the same output.
- A *pseudorandom function* is a function that emulates a random oracle

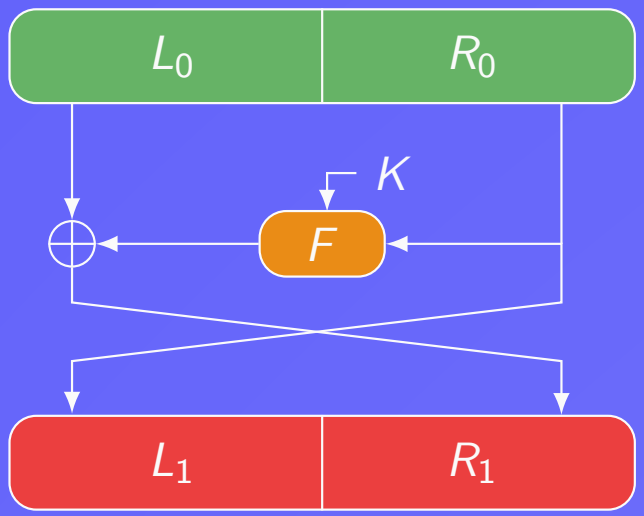
- A pseudorandom function that cannot be distinguished from a random permutation is called *pseudo random permutation*
- A *blockcipher* is a pseudorandom permutation
- A *one way function* is a function easy to evaluate but hard to invert

Chapter organisation



Basic Feistel network

We want to build a random bijection over $2n$ bits



- Size of a block: $2n$ bits
- Split the block into two blocks of n bits each
- $F : \{0,1\}^n \times \{0,1\}^k \to \{0,1\}^n$

We define the function

$$\Psi_F : \{0,1\}^{2n} \longrightarrow \{0,1\}^{2n}$$
$$[L, R] \longmapsto [R, L \oplus F(R, K)]$$

Inverse function

Proposition

For any function F, Ψ_F is a bijection and $\Psi_F^{-1} = \sigma \circ \Psi_F \circ \sigma$, with

$$\sigma : \{0,1\}^{2n} \longrightarrow \{0,1\}^{2n}$$
$$[L, R] \longmapsto [R, L]$$

Proof.

By definition of Ψ_F, $\Psi_F\left([L_0, R_0]\right) = [R_0, L_0 \oplus F(R_0, K)] = [L_1, R_1]$.

Equivalently,

$$\begin{cases} R_0 &= L_1 \\ L_0 &= R_1 \oplus F(L_1, K). \end{cases}$$

Moreover

$$\begin{aligned} \sigma \circ \Psi_F \circ \sigma([L_1, R_1]) &= \sigma \circ \Psi_F \circ \sigma([R_0, L_0 \oplus F(R_0, K)]) \\ &= \sigma\left(\Psi_F([L_0 \oplus F(R_0, K), R_0])\right) \\ &= \sigma\left(R_0, L_0 \oplus F(R_0, K) \oplus F(R_0, K)\right) \\ &= [L_0, R_0]. \end{aligned}$$

□

Attacking Feistel networks

Setting up two black boxes, a random oracle and a Feistel network, the goal for an attacher is to distinguish them.

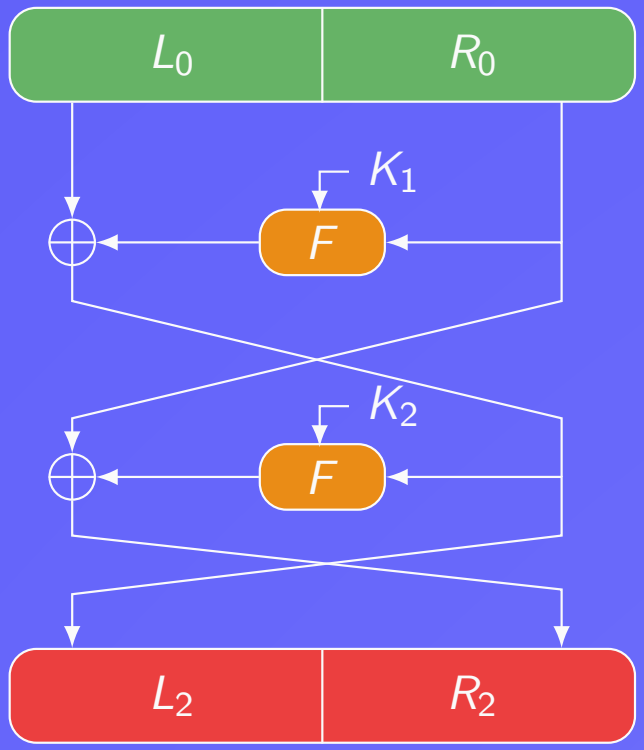
Attacking Feistel networks

Setting up two black boxes, a random oracle and a Feistel network, the goal for an attacher is to distinguish them.

Number of messages necessary to reach the goal:

Rounds	KPA	CPA	CPCA
1	1	1	1
2	$\mathcal{O}\left(\sqrt{2^n}\right)$	2	2
3	$\mathcal{O}\left(\sqrt{2^n}\right)$	$\mathcal{O}\left(\sqrt{2^n}\right)$	3
4	$\mathcal{O}\left(2^n\right)$	$\mathcal{O}\left(\sqrt{2^n}\right)$	$\mathcal{O}\left(\sqrt{2^n}\right)$

Two rounds – CPA

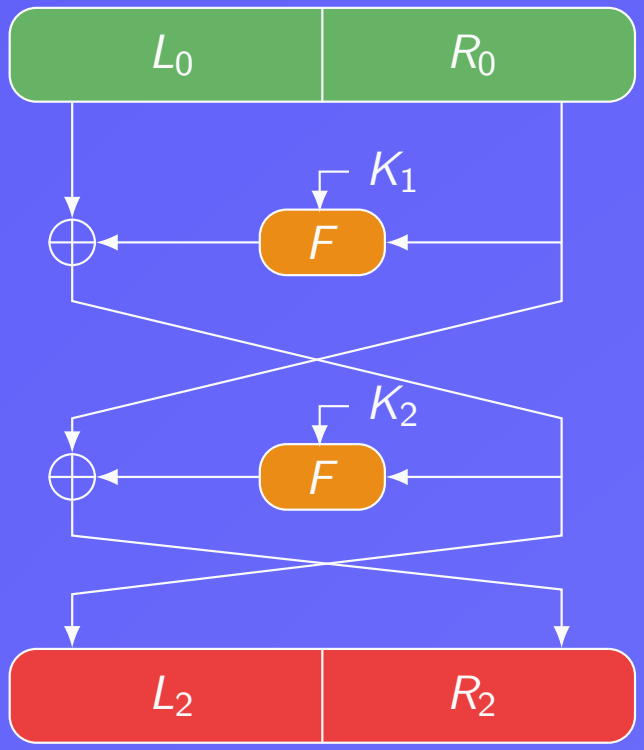


Attack strategy:

- For simplicity we denote $F(X, K)$ by $F_k(X)$ and $\Psi_{F_{k_2}} \circ \Psi_{F_{K_1}}$ by $\Psi^2_{F_{K_1},F_{K_2}}$
- $\Psi^2_{F_{K_1},F_{K_2}}\left([L_0, R_0]\right) = [L_2, R_2]$ with $L_2 = L_0 \oplus F_{K_1}(R_0)$ and $R_2 = R_0 \oplus F_{K_2}(L_2)$
- The inverse of $\Psi^2_{F_{K_1},F_{K_2}}$ is

$$\begin{aligned}\Psi^{-2}_{F_{K_1},F_{K_2}} &= \Psi^{-1}_{F_{K_1}} \circ \Psi^{-1}_{F_{K_2}} \\ &= \sigma \circ \Psi_{F_{K_1}} \circ \sigma \circ \sigma \circ \Psi_{F_{K_2}} \circ \sigma \\ &= \sigma \circ \Psi^2_{F_{K_2},F_{K_1}} \circ \sigma\end{aligned}$$

Two rounds – CPA



Attack strategy:

- For simplicity we denote $F(X, K)$ by $F_k(X)$ and $\Psi_{F_{k_2}} \circ \Psi_{F_{K_1}}$ by $\Psi^2_{F_{K_1}, F_{K_2}}$
- $\Psi^2_{F_{K_1}, F_{K_2}}\left([L_0, R_0]\right) = [L_2, R_2]$ with $L_2 = L_0 \oplus F_{K_1}(R_0)$ and $R_2 = R_0 \oplus F_{K_2}(L_2)$
- The inverse of $\Psi^2_{F_{K_1}, F_{K_2}}$ is

$$\begin{aligned}\Psi^{-2}_{F_{K_1}, F_{K_2}} &= \Psi^{-1}_{F_{K_1}} \circ \Psi^{-1}_{F_{K_2}} \\ &= \sigma \circ \Psi_{F_{K_1}} \circ \sigma \circ \sigma \circ \Psi_{F_{K_2}} \circ \sigma \\ &= \sigma \circ \Psi^2_{F_{K_2}, F_{K_1}} \circ \sigma\end{aligned}$$

What if we use $m_1 = [m_{1_L}, m_{1_R}]$ and $m_2 = [m_{2_L}, m_{2_R}]$ such that

$$\begin{cases} m_{1_L} \neq m_{2_L} \\ m_{1_R} = m_{2_R} \end{cases}$$

Two rounds – KPA

Number of plaintext/ciphertext pairs needed: $\mathcal{O}(\sqrt{2^n})$

1. Find a collision over the m_{i_R}, $1 \leq i \leq 2^n$
2. If a collision is found for m_j and m_l check if

$$m_{j_{L_2}} \oplus m_{l_{L_2}} = m_{j_{L_0}} \oplus m_{l_{L_0}}$$

Two rounds – KPA

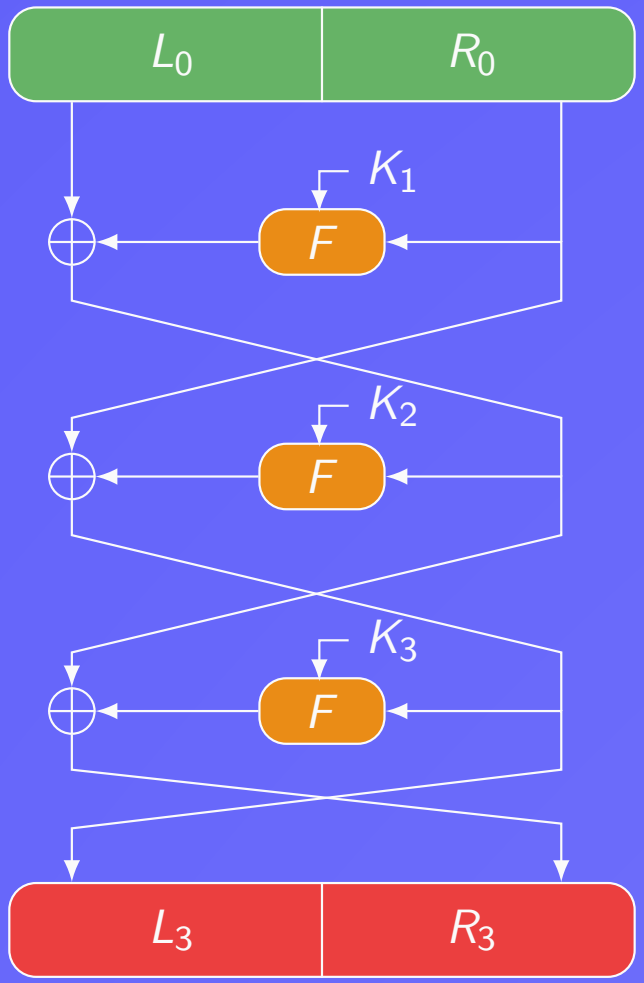
Number of plaintext/ciphertext pairs needed: $\mathcal{O}(\sqrt{2^n})$

1. Find a collision over the m_{i_R}, $1 \leq i \leq 2^n$
2. If a collision is found for m_j and m_l check if

$$m_{j_{L_2}} \oplus m_{l_{L_2}} = m_{j_{L_0}} \oplus m_{l_{L_0}}$$

- Step 1: $\mathcal{O}(\sqrt{2^n})$ messages (birthday paradox, 4.11)
- Step 2: no better than $\mathcal{O}(\sqrt{2^n})$:
  - Collision on $m_{i_{L_2}} = m_{i_{L_0}} \oplus F_{K_1}(m_{iR_0})$ for two messages
  - The variables $m_{i_{L_2}}$, $m_{i_{L_0}}$, and $m_{i_{R_0}}$ are fixed
  - It only depends on F_{K_1}, which can take 2^n different values
  - From l messages $\frac{l(l-1)}{2}$ pairs can be constructed
  - Probability of collision: $\approx \frac{l(l-1)}{2 \cdot 2^n}$

Three rounds – CPCA



Attack strategy:

- $\Psi^3_{F_{k_1},F_{k_2},F_{k_3}}([L_0, R_0]) = [L_3, R_3]$ with

$$\begin{cases} L_3 = R_0 \oplus F_{K_2}(L_2) \\ R_3 = L_2 \oplus F_{K_3}(L_3) \end{cases}$$

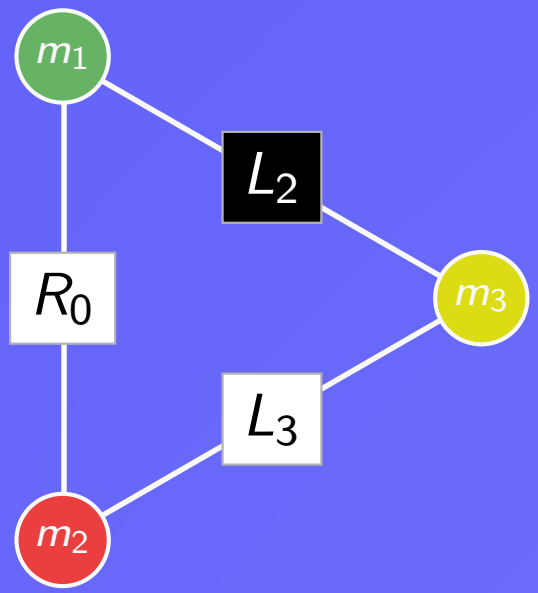
and $L_2 = L_0 \oplus F_{K_1}(R_0)$

- Notice for a pair of messages (m_a, m_b)

$$\begin{aligned} m_{a_{R_0}} = m_{b_{R_0}} &\Rightarrow m_{a_{L_2}} \oplus m_{b_{L_2}} = m_{a_{L_0}} \oplus m_{b_{L_0}} \\ m_{a_{L_2}} = m_{b_{L_2}} &\Rightarrow m_{a_{L_3}} \oplus m_{b_{L_3}} = m_{a_{R_0}} \oplus m_{b_{R_0}} \\ m_{a_{L_3}} = m_{b_{L_3}} &\Rightarrow m_{a_{R_3}} \oplus m_{b_{R_3}} = m_{a_{L_2}} \oplus m_{b_{L_2}} \end{aligned} \tag{2.2}$$

Three rounds – CPCA

Graph for Eq. (2.3):



Attack strategy:

- We take m_1, m_2, and m_3 such that

$$\begin{cases} m_{2_{R_0}} & = m_{1_{R_0}} \\ m_{3_{L_3}} & = m_{2_{L_3}} \\ m_{3_{L_2}} & = m_{1_{L_2}} \end{cases} \tag{2.3}$$

- From eq. (2.2) and (2.3) we derive

$$\begin{cases} m_{2_{R_0}} & = m_{1_{R_0}} \\ m_{3_{L_3}} & = m_{2_{L_3}} \\ m_{3_{R_3}} & = m_{2_{R_3}} \oplus m_{1_{L0}} \oplus m_{2_{L_0}} \end{cases}$$

- We finally get

$$m_{3_{R_0}} = m_{1L_3} \oplus m_{3L_3} \oplus m_{1_{R_0}}$$

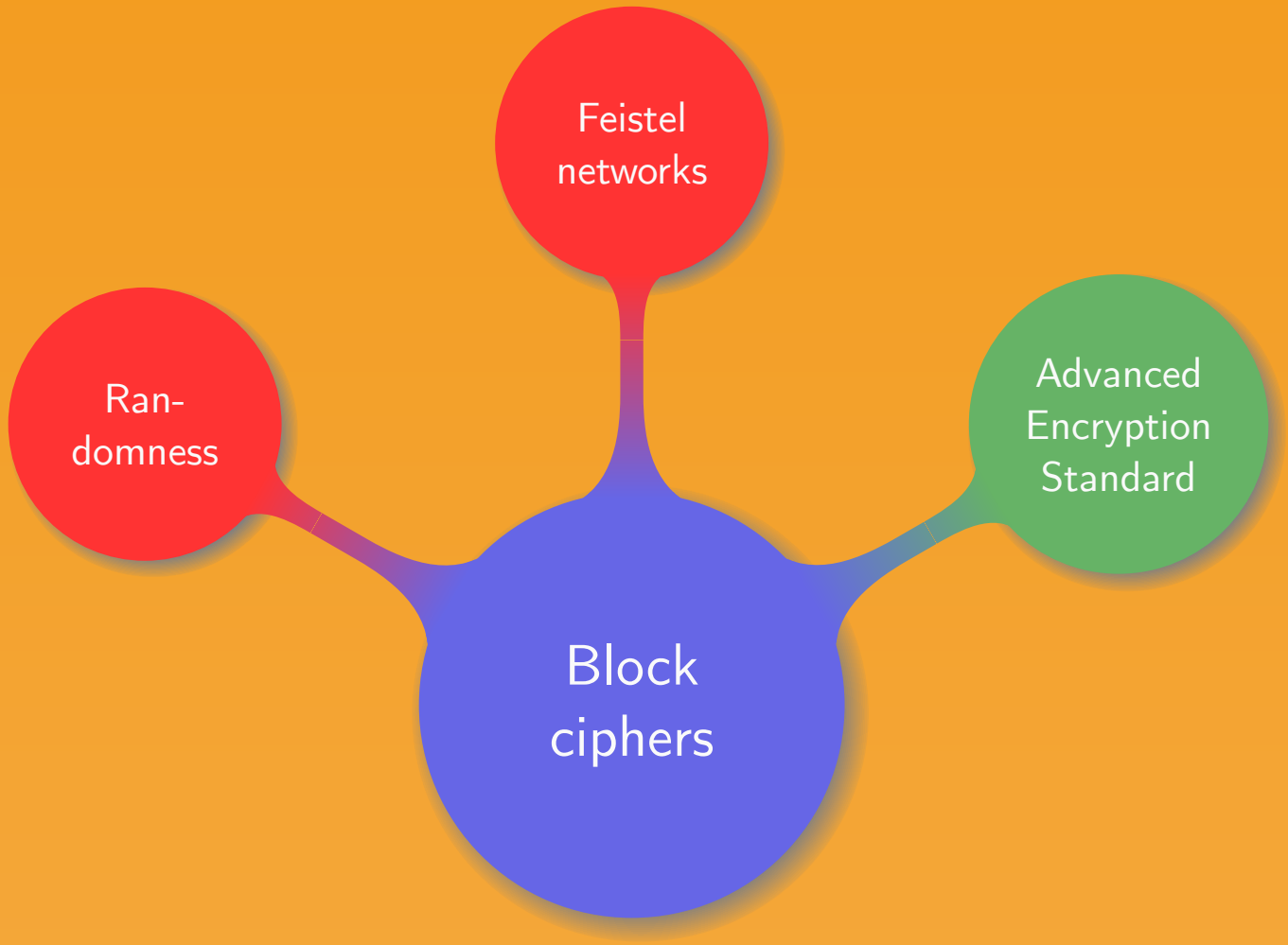
- What can be concluded?

Data Encryption Standard

Data Encryption Standard (DES):

- 1974: IBM uses Feistel networks to create LUCIFER
- 1975: LUCIFER is sent to NSA for review and modifications
- 1977: renamed DES and becomes the official encryption standard
- 2002: DES is not secure anymore and is replaced by AES

Chapter organisation



Advanced Encryption Standard

Advanced Encryption Standard (AES):

- 1997: call for candidates to replace DES
- Requirements:
  - Possible key sizes: 128, 192 and 256 bits
  - Input block size: 128 bits
  - Work on various hardware (e.g. 8-bit processors)
  - Speed
- Five finalists: MARS, RC6, Rijndael, Serpent, and Twofish
- 2001: Rijndael is chosen to become AES

AES – General description

Brief outline of AES:

- 10 rounds for a 128-bit key (12 and 14 for 192 and 256-bit)
- A round is formed of layers
  - *SubBytes*: substitution operation
  - *ShiftRows*: linear mixing step on the rows
  - *MixColumns*: linear mixing on the columns
  - *AddRoundKey*: apply a round key derived from the main key

AES – Encryption

AES setup:

- The 128 bits are grouped into 16 bytes
- Each byte is composed of 8 bits:

$$a_{0,0}, a_{1,0}, a_{2,0}, a_{3,0}, a_{1,1}, \cdots, a_{3,3}$$

- Bits are arranged in a 4×4 matrix:

$$\begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix}$$

Invertible elements

So far we worked with the set $S = \{0, \cdots, n-1\}$ using modular congruences (def. 1.19). In the proof of prop. 2.15 we noted that when n is prime all the non-zero elements of S are invertible.

Example.

i. The set $S = \{0, \cdots, 4\}$ has five elements, and since five is prime all the non-zero elements are invertible. Indeed,

$$1 \cdot 1 \equiv 1 \bmod 5, \ 2 \cdot 3 = 6 \equiv 1 \bmod 5, \text{ and } 4^2 = 16 \equiv 1 \bmod 5.$$

ii. The set $S = \{0, \cdots, 5\}$ has six elements, and as six is not prime some non-zero elements are not invertible. In fact since

$$2 \cdot 3 = 6 \equiv 0 \bmod 6,$$

we conclude that 2 and 3 are not invertible mod 6.

Finite fields

Loosely speaking a set where the addition and multiplication operations are defined and such that every non-zero element is invertible for the multiplication is called a *field*.

When a field has a finite number of elements it is called *finite field*. For each prime p and positive integer n there exists a finite field with p^n elements, often denoted $\mathrm{GF}(p^n)$ or $\mathbb{F}_{p^n}$ (GF standing for Galois Field).

Remark.
The set $S = \{0, \cdots, 8\}$ has $9 = 3^2$ elements and is not a field since 3 is not invertible. Therefore the question remaining to answer is "how to construct a finite field with nine elements", or more generally with p^n elements.

Polynomials over finite fields

Similarly to how polynomials are defined over common fields such as the real numbers, they can also be defined over finite fields. The main difference relies on their coefficients which take their values in the base field.

In a field, a polynomial which cannot be written as the product of two polynomials of lower degree is said to be *irreducible*.

Example.

i. In $\mathbb{F}_2[X]$, $X^2 + 3X + 1$ and $X^2 + X + 1$ are equal.

ii. In $\mathbb{F}_5[X]$, $X^3 + X + 3 = (X + 4)(X^2 + X + 2)$ is not irreducible.

iii. In $\mathbb{F}_{17}[X]$, $X^3 + X + 3$ is irreducible.

Non-prime fields

Theorem

Let $P(X)$ be an irreducible polynomial of degree n in $\mathbb{F}_p[X]$, and F be the set of all the polynomials of degree less than n. Then F is a finite field with p^n elements.

Proof.

Assuming addition and multiplication are properly defined we need to prove that F has p^n elements and that all but 0 are invertible.

It is simple to see that F has p^n elements since each of the n monomials (from degree 0 to $n-1$) can take p different values (from 0 to $p-1$).

Non-prime fields

Proof (continued).

Let $A(X)$, $B(X)$ and $C(X)$ be three distinct non-zero polynomials such that

$$A(X)B(X) \equiv A(X)C(X) \bmod P(X).$$

This implies $A(X)\left(B(X) - C(X)\right) \equiv 0 \bmod P(X)$, which is not possible since $P(X)$ is irreducible.

Hence multiplying a polynomial $A(X)$ by all the non-zero elements of F results in covering all the non-zero polynomials of F, meaning that there is a polynomial $B(X)$ such that

$$A(X)B(X) \equiv 1 \bmod P(X).$$

□

Finite fields in the AES

In Rijndael $\mathbb{F}_{2^8}$ is used:

- $P(X) = X^8 + X^4 + X^3 + X + 1$ is the irreducible over $\mathbb{F}_2[X]$
- Each element of $\mathbb{F}_{2^8}$ is a polynomial of the form

$$a_7X^7 + a_6X^6 + a_5X^5 + a_4X^4 + a_3X^3 + a_2X^2 + a_1X + a_0$$

- The polynomial is described as a byte $a_7a_6a_5a_4a_3a_2a_1a_0$
- The sum of two polynomials is the XOR of their bit representation

Finite fields in the AES

In Rijndael $\mathbb{F}_{2^8}$ is used:

- $P(X) = X^8 + X^4 + X^3 + X + 1$ is the irreducible over $\mathbb{F}_2[X]$
- Each element of $\mathbb{F}_{2^8}$ is a polynomial of the form

$$a_7X^7 + a_6X^6 + a_5X^5 + a_4X^4 + a_3X^3 + a_2X^2 + a_1X + a_0$$

- The polynomial is described as a byte $a_7a_6a_5a_4a_3a_2a_1a_0$
- The sum of two polynomials is the XOR of their bit representation
- Multiplying a polynomial $Q(X)$ by X:
  1. Shift left the byte representation of $Q(X)$ and append a 0
  2. If the first bit is 0 stop and otherwise XOR with $P(X)$
- Multiplying $Q(X)$ by $R(X)$:
  1. Split $R(X)$ into the monomials $M_i(X)$, $i \leq \deg R(X)$
  2. For $M_i(X)$ applying the multiplication by X $\deg M_i(X)$ times
  3. Add all the results using XOR

Finite fields in the AES

Example.

Let $Q(X) = X^7 + X^4 + X + 1$ and $R(X) = X^2 + 1$. Determine the product $Q(X)R(X)$ in $\mathbb{F}_{2^8}[X]$.

1. Regular strategy: multiply and reduce mod $P(X)$
   - $Q(X)R(X) = X^9 + X^7 + X^6 + X^4 + X^3 + X^2 + X + 1$
   - Since $P(X) = 0$, $X^9 = X^5 + X^4 + X^2 + X$ and

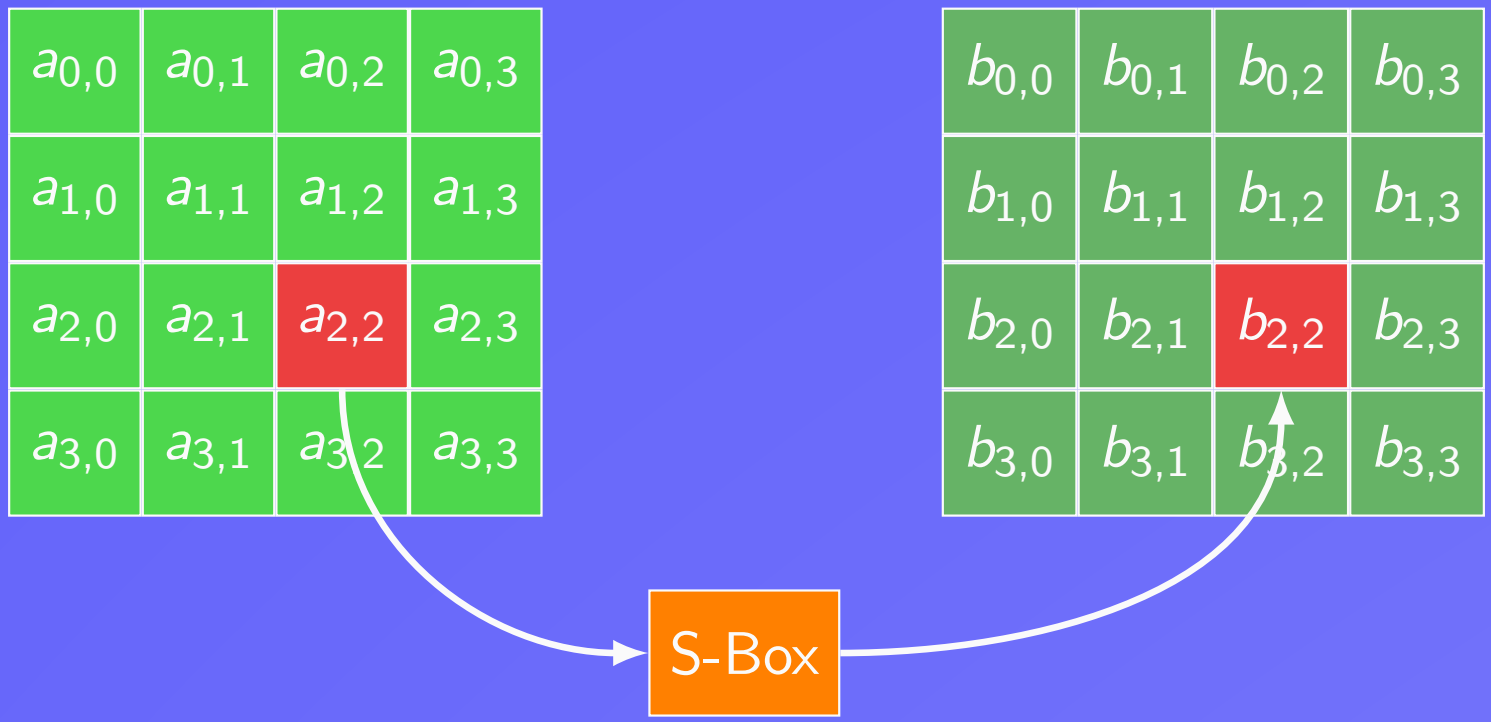
$$Q(X)R(X) \equiv X^7 + X^6 + X^5 + X^3 + 1 \bmod P(X)$$

2. Represent polynomials as bytes and apply XOR operations:

Write $Q(X) = 10010011$ and decompose $R(X)$ as $X \cdot X + 1$

   - $Q(X) \cdot X = 100100110 \oplus 100011011 = 000111101$
   - $(Q(X) \cdot X) \cdot X = 001111010$
   - $(Q(X) \cdot X) \cdot X + Q(X) = 01111010 \oplus 10010011 = 11101001$
   - $Q(X)R(X) \equiv X^7 + X^6 + X^5 + X^3 + 1 \bmod P(X)$

The SubBytes layer



For each byte in the matrix:

- Split it into two 4-bit numbers a and b
- Find byte c in the S-Box table at row a and column b
- Replace the original byte by c

S-Box

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	99	124	119	123	242	107	111	197	48	1	103	43	254	215	171	118
1	202	130	201	125	250	89	71	240	173	212	162	175	156	164	114	192
2	183	253	147	38	54	63	247	204	52	165	229	241	113	216	49	21
3	4	199	35	195	24	150	5	154	7	18	128	226	235	39	178	117
4	9	131	44	26	27	110	90	160	82	59	214	179	41	227	47	132
5	83	209	0	237	32	252	177	91	106	203	190	57	74	76	88	207
6	208	239	170	251	67	77	51	133	69	249	2	127	80	60	159	168
7	81	163	64	143	146	157	56	245	188	182	218	33	16	255	243	210
8	205	12	19	236	95	151	68	23	196	167	126	61	100	93	25	115
9	96	129	79	220	34	42	144	136	70	238	184	20	222	94	11	219
10	224	50	58	10	73	6	36	92	194	211	172	98	145	149	228	121
11	231	200	55	109	141	213	78	169	108	86	244	234	101	122	174	8
12	186	120	37	46	28	166	180	198	232	221	116	31	75	189	139	138
13	112	62	181	102	72	3	246	14	97	53	87	185	134	193	29	158
14	225	248	152	17	105	217	142	148	155	30	135	233	206	85	40	223
15	140	161	137	13	191	230	66	104	65	153	45	15	176	84	187	22

Generating the S-Box

Simple construction:

- For a in $\mathbb{F}_{2^8}^*$ compute its inverse $b = a^{-1}$ or set $b = 0$ if $a = 0$
- Represent b as a column vector $B = (b_0, \cdots, b_7)$
- Compute

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \end{pmatrix}$$

- The entry located at row $(a_7 \cdots a_4)_2$ and column $(a_3 \cdots a_0)_2$ of the S-Box is $(c_7 \cdots c_0)_2$

Generating the S-Box

Example.
Find the S-Box entry corresponding to the byte 11001011?

Generating the S-Box

Example.
Find the S-Box entry corresponding to the byte 11001011?

The byte 11001011 stands for $a(X) = X^7 + X^6 + X^3 + X + 1$, and we observe that

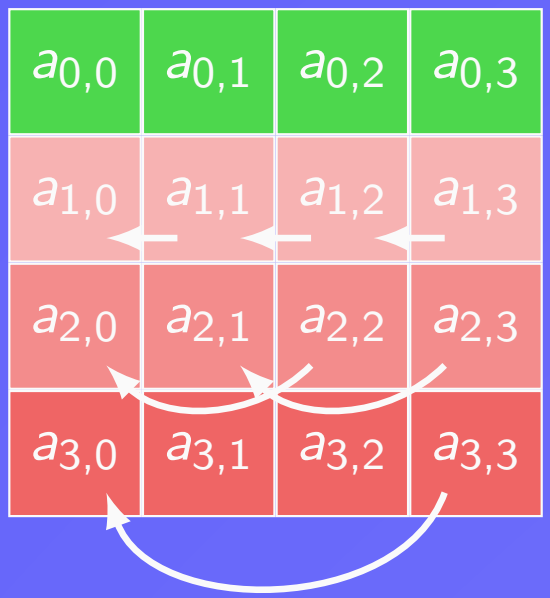
$$\begin{aligned} a(X) \cdot X^2 &= X^9 + X^8 + X^5 + X^3 + X^2 \\ &\equiv X^8 + X^4 + X^3 + X \bmod P(X) \\ &\equiv 1 \bmod P(X). \end{aligned}$$

Therefore we calculate

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix},$$

and finally conclude that the entry at row 12 and column 11 is 31.

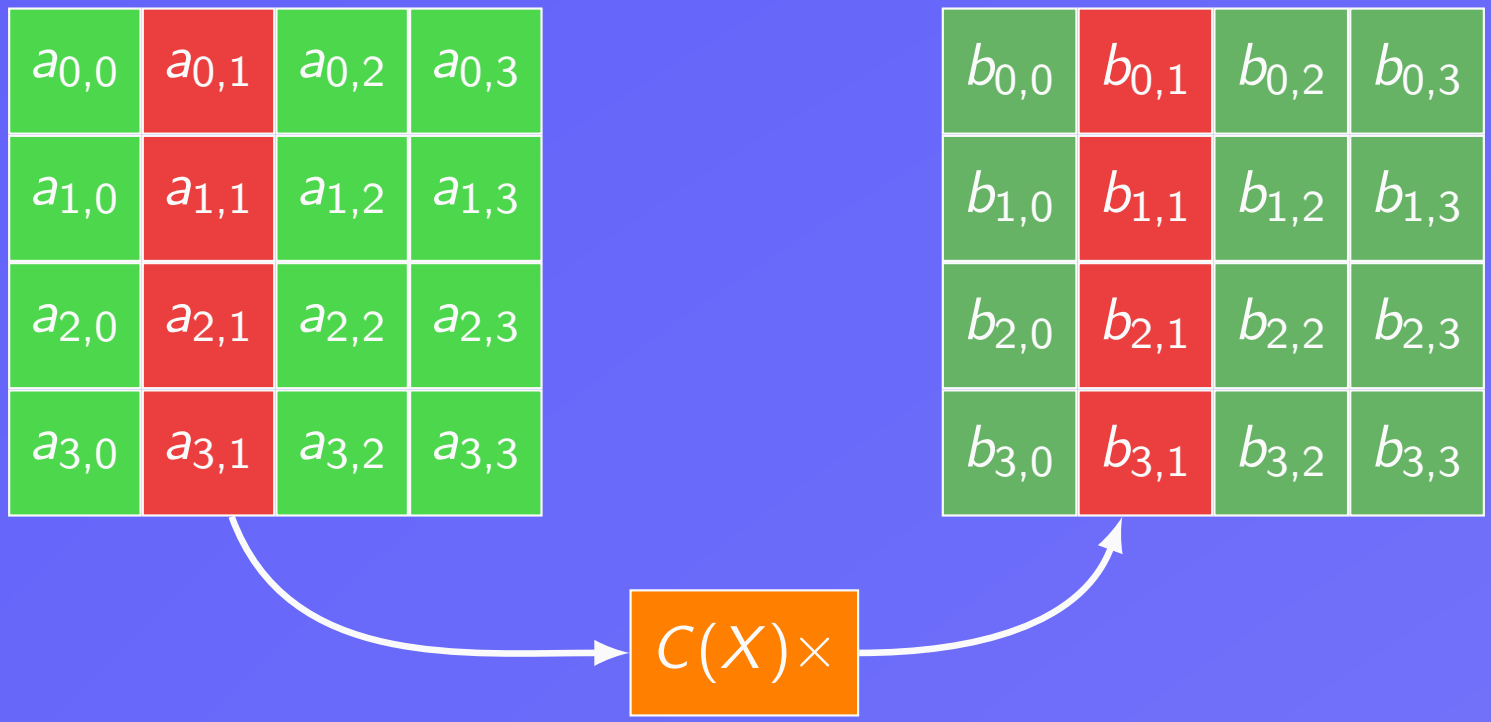
The ShiftRows layer



$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,0}$
$a_{2,2}$	$a_{2,3}$	$a_{2,0}$	$a_{2,1}$
$a_{3,3}$	$a_{3,0}$	$a_{3,1}$	$a_{3,2}$

Cyclically shift to the left row i by offset i, $0 \leq i \leq 3$

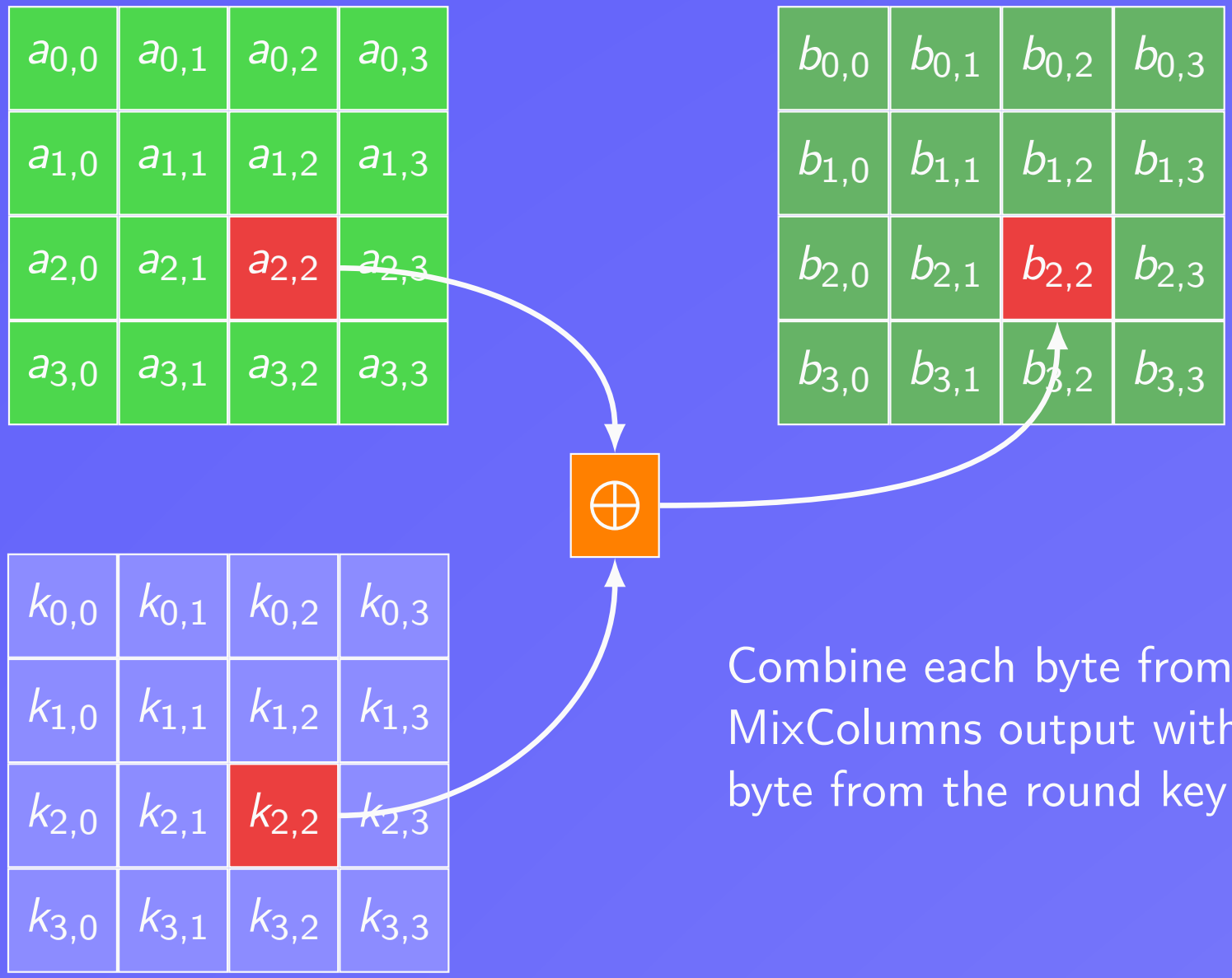
The MixColumns layer



Left multiply the output of ShiftRows by the matrix

$$C(X) = \begin{pmatrix} 00000010 & 00000011 & 00000001 & 00000001 \\ 00000001 & 00000010 & 00000011 & 00000001 \\ 00000001 & 00000001 & 00000010 & 00000011 \\ 00000011 & 00000001 & 00000001 & 00000010 \end{pmatrix}$$

The AddRoundKey layer



Combine each byte from the MixColumns output with a byte from the round key

Generating a round key

The original 128 bits key is arranged into a 4×4 matrix $K(X)$

Label the first four columns $K(0), \cdots, K(3)$ and add forty more:

- $K(i) = K(i-4) \oplus K(i-1)$, for $i \not\equiv 0 \bmod 4$
- $K(i) = K(i-4) \oplus T(K(i-1))$, for $i \equiv 0 \bmod 4$

Generating a round key

The original 128 bits key is arranged into a 4×4 matrix $K(X)$

Label the first four columns $K(0), \cdots, K(3)$ and add forty more:

- $K(i) = K(i-4) \oplus K(i-1)$, for $i \not\equiv 0 \bmod 4$
- $K(i) = K(i-4) \oplus T(K(i-1))$, for $i \equiv 0 \bmod 4$

The transformation $T(K(i-1))$ is defined over the column i:

- Compute $r(i) = 00000010^{\frac{i-4}{4}}$
- Cyclically top shift the elements of the column by 1
- Apply the SubBytes layer (2.42) to each byte of the column and get the column vector (a, b, c, d)
- Finally return the column vector

$$T(K(i-1)) = (a \oplus r(i), b, c, d)$$

Generating a round key

The original 128 bits key is arranged into a 4×4 matrix $K(X)$

Label the first four columns $K(0), \cdots, K(3)$ and add forty more:

- $K(i) = K(i-4) \oplus K(i-1)$, for $i \not\equiv 0 \bmod 4$
- $K(i) = K(i-4) \oplus T(K(i-1))$, for $i \equiv 0 \bmod 4$

The transformation $T(K(i-1))$ is defined over the column i:

- Compute $r(i) = 00000010^{\frac{i-4}{4}}$
- Cyclically top shift the elements of the column by 1
- Apply the SubBytes layer (2.42) to each byte of the column and get the column vector (a, b, c, d)
- Finally return the column vector

$$T(K(i-1)) = (a \oplus r(i), b, c, d)$$

The i-th round key is given by the columns $K(4i), \cdots, K(4i+3)$

Generating a round key

Example.
$K(i)$ being simple to generate for $i \not\equiv 0 \bmod 4$, we focus on the case $i \equiv 0 \bmod 4$. For instance if $i = 40$ and $K(39)$ is the column vector $(10001100, 00001100, 11000110, 11110011)$, then

- Cyclical top shit: $(00001100, 11000110, 11110011, 10001100)$
- SubBytes transformation:

$$\begin{array}{llllll} 00001100 & \to & 11111110, & 11000110 & \to & 10110100 \\ 11110011 & \to & 00001101, & 10001100 & \to & 01100100 \end{array}$$

- $r(40) = X^9 \equiv X^5 + X^4 + X^2 + X \bmod P(X) = 00110110$
- Get the final column vector $T(K(39))$

$$\begin{aligned} T(K(39)) &= (11111110 \oplus 00110110, 10110100, 00001101, 01100100) \\ &= (11001000, 10110100, 00001101, 01100100) \end{aligned}$$

- Finally define $K(40)$ as $K(36) \oplus T(K(39))$

Decryption

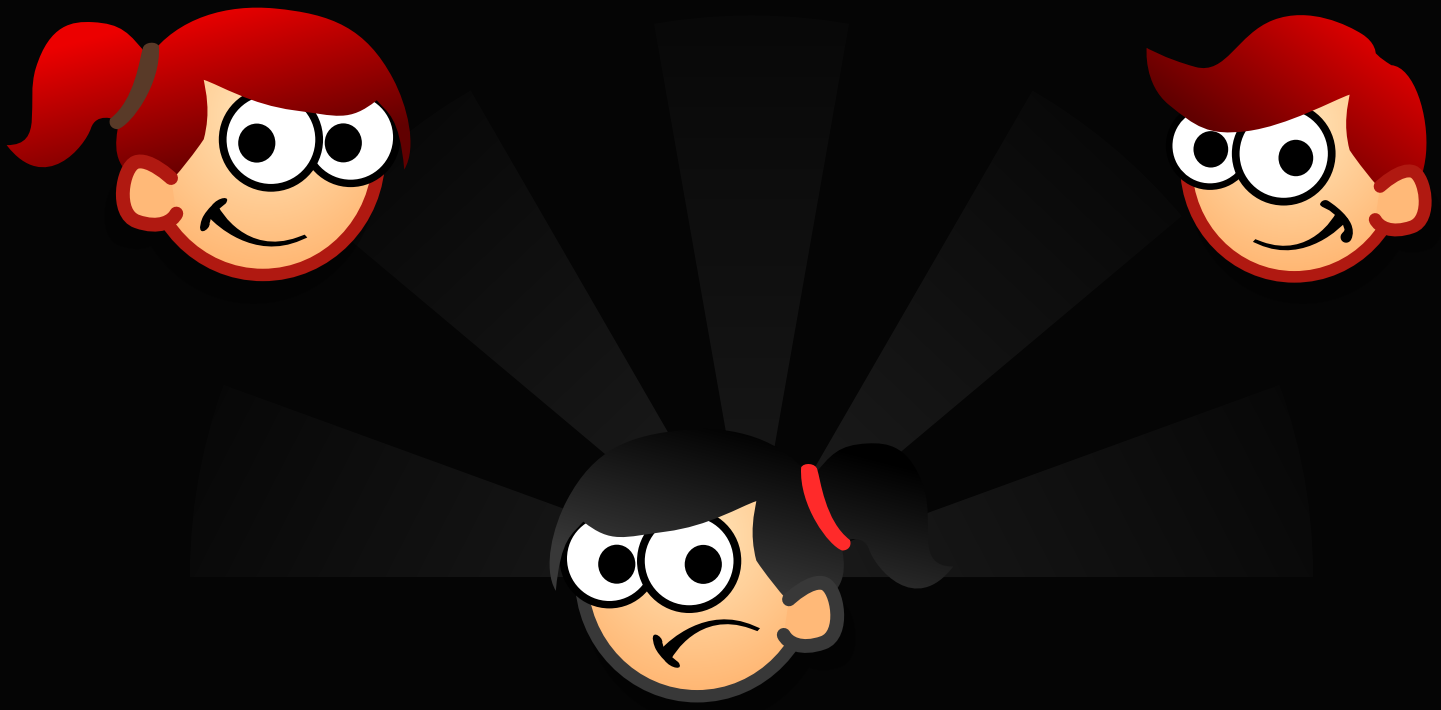
The decryption process is simple:

- Perform all the operations in reverse order
- Replace the SubBytes, ShitRows and MixColumns operations by their inverse

Remark.
It is possible to construct an inverse cipher performing decryption by applying a sequence of inverse operations in the same order as it is done for encryption

Key points

- What does it mean to be random?
- Recall Fermat's little theorem
- Recall the CRT
- How is a Feistel network organised?
- Describe AES



Thank you!

References I

2.4 https://upload.wikimedia.org/wikipedia/commons/5/56/Tux.jpg

2.4 https://upload.wikimedia.org/wikipedia/commons/f/f0/Tux_ecb.jpg

2.4 https://upload.wikimedia.org/wikipedia/commons/a/a0/Tux_secure.jpg

2.7 https://www.xkcd.com/221/