Lecture 3
Ciphers and Information Theory

Douglas Wikström
KTH Stockholm
dog@csc.kth.se

January 30, 2015

AES

Advanced Encryption Standard (AES)

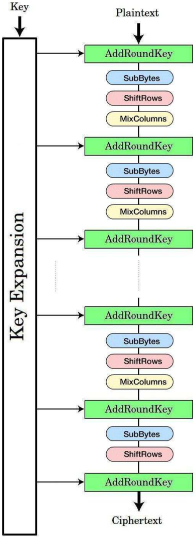
- Chosen in worldwide **public competition** 1997-2000. Probably no back-doors. Increased confidence!
- Winning proposal named "Rijndael", by Rijmen and Daemen
- Family of 128-bit block ciphers:

Key bits	128	192	256
Rounds	10	12	14

- The first key-recovery attacks on full AES due to Bogdanov, Khovratovich, and Rechberger, published **2011**, is faster than brute force by a factor of about **4**.
- ... algebraics of AES make some people uneasy.

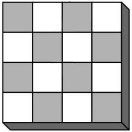
AES

- **AddRoundKey**: XOR With Round Key
- **SubBytes**: Substitution
- **ShiftRows**: Permutation
- **MixColumns**: Linear Map



Similar to SPN

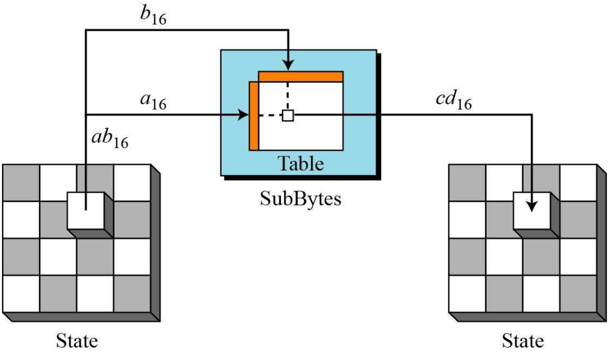
The 128 bit state is interpreted as a 4×4 matrix of bytes.



Something like a mix between substitution, permutation, affine version of Hill cipher. In each round!

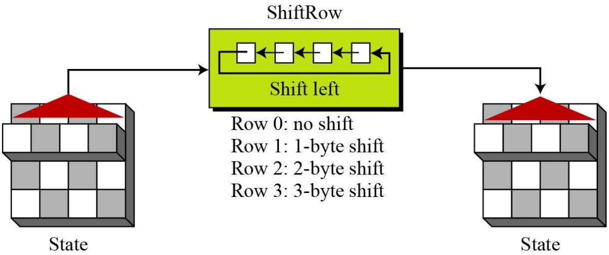
SubBytes

SubBytes is field inversion in $\mathbb{F}_{2^8}$ plus affine map in $\mathbb{F}_2^8$.



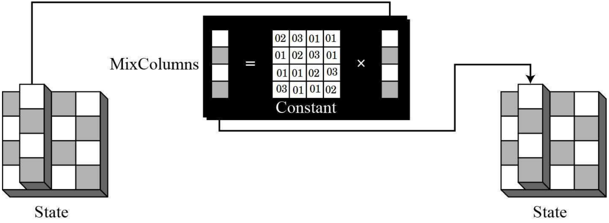
ShiftRows

`ShiftRows` is a cyclic shift of bytes with offsets: 0, 1, 2, and 3.



MixColumns

MixColumns is an invertible linear map over $\mathbb{F}_{2^8}$ (with irreducibile polynomial $x^8 + x^4 + x^3 + x + 1$) with good diffusion.



Decryption

Uses the following transformations:

- **AddRoundKey**
- **InvSubBytes**
- **InvShiftRows**
- **InvMixColumns**

Feistel Networks

Feistel Networks

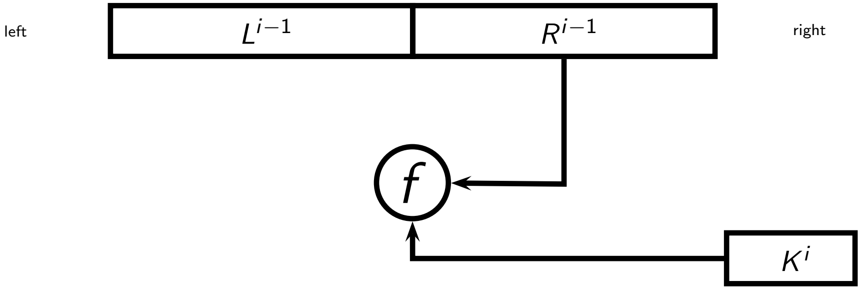
- Identical rounds are iterated, but with different round keys.
- The input to the ith round is divided in a left and right part, denoted L^{i-1} and R^{i-1}.
- f is a function for which it is somewhat hard to find pre-images, but f is typically **not invertible**!
- One round is defined by:

$$L^i = R^{i-1}$$
$$R^i = L^{i-1} \oplus f(R^{i-1}, K^i)$$

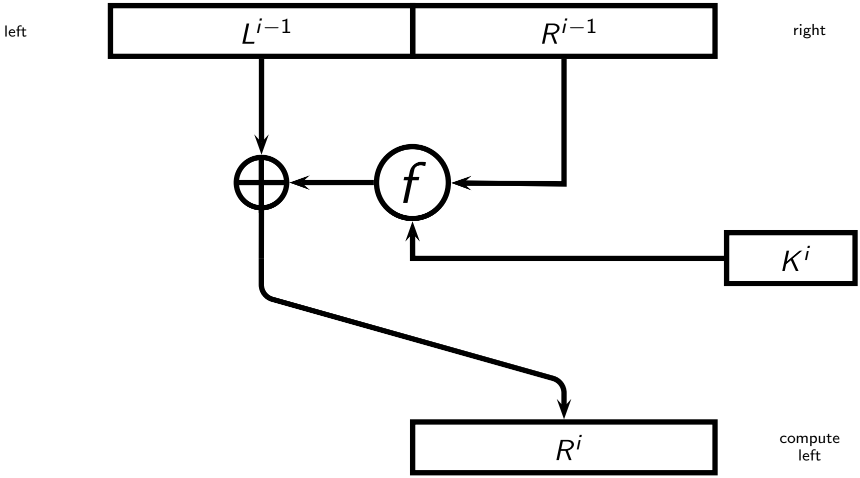
where K^i is the ith round key.

Feistel Round

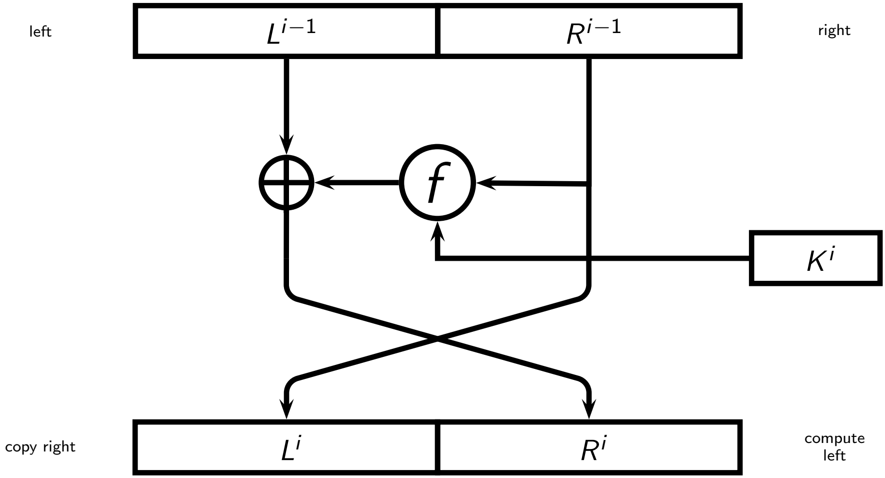
Feistel Round



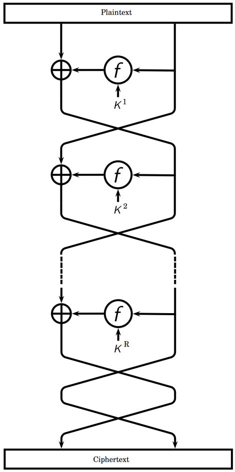
Feistel Round



Feistel Round



Feistel Cipher



Inverse Feistel Round

Feistel Round.

$$L^i = R^{i-1}$$
$$R^i = L^{i-1} \oplus f(R^{i-1}, K^i)$$

Inverse Feistel Round

Feistel Round.

$$L^i = R^{i-1}$$
$$R^i = L^{i-1} \oplus f(R^{i-1}, K^i)$$

Inverse Feistel Round.

$$L^{i-1} = R^i \oplus f(L^i, K^i)$$
$$R^{i-1} = L^i$$

Reverse direction and swap left and right!

Idealized Four-Round Feistel Network

Definition. Feistel round (H for "Horst Feistel").

$$H_{F_K}(L, R) = (R, L \oplus F(R, K))$$

Idealized Four-Round Feistel Network

Definition. Feistel round (H for "Horst Feistel").

$$H_{F_K}(L, R) = (R, L \oplus F(R, K))$$

Theorem. (Luby and Rackoff) If F is a pseudo-random family of functions, then

$$H_{F_{k_1},F_{k_2},F_{k_3},F_{k_4}}(x) = H_{F_{k_4}}(H_{F_{k_3}}(H_{F_{k_2}}(H_{F_{k_1}}(x))))$$

(and its inverse) is a pseudo-random family of permutations.

Idealized Four-Round Feistel Network

Definition. Feistel round (H for "Horst Feistel").

$$H_{F_K}(L, R) = (R, L \oplus F(R, K))$$

Theorem. (Luby and Rackoff) If F is a pseudo-random family of functions, then

$$H_{F_{k_1}, F_{k_2}, F_{k_3}, F_{k_4}}(x) = H_{F_{k_4}}(H_{F_{k_3}}(H_{F_{k_2}}(H_{F_{k_1}}(x))))$$

(and its inverse) is a pseudo-random family of permutations.

Why do we need four rounds?

DES

Quote

The news here is not that DES is insecure, that hardware algorithm-crackers can be built, or that a 56-bit key length is too short. ... The news is how long the government has been denying that these machines were possible. As recently as 8 June 98, Robert Litt, principal associate deputy attorney general at the Department of Justice, denied that it was possible for the FBI to crack DES. ... My comment was that the FBI is either incompetent or lying, or both.

– Bruce Schneier, 1998

Data Encryption Standard (DES)

- Developed at IBM in 1975, or perhaps...

Data Encryption Standard (DES)

- Developed at IBM in 1975, or perhaps...
- at National Security Agency (NSA). Nobody knows for certain.

Data Encryption Standard (DES)

- Developed at IBM in 1975, or perhaps...
- at National Security Agency (NSA). Nobody knows for certain.
- 16-round Feistel network.

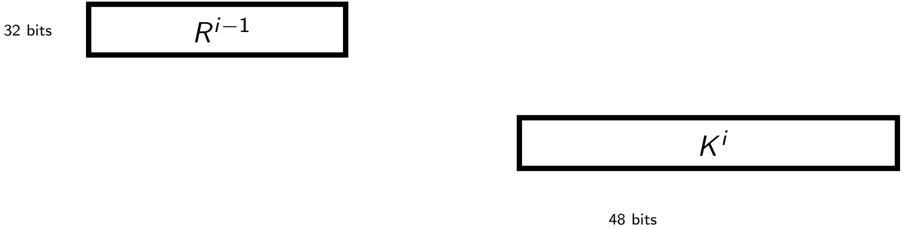
Data Encryption Standard (DES)

- Developed at IBM in 1975, or perhaps...
- at National Security Agency (NSA). Nobody knows for certain.
- 16-round Feistel network.
- Key schedule derives permuted bits for each round key from a 56-bit key. Supposedly not 64-bit due to parity bits.

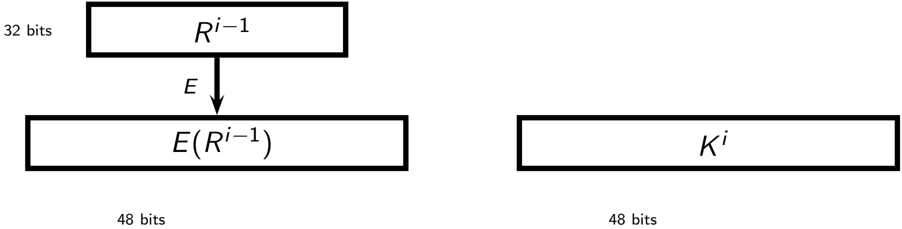
Data Encryption Standard (DES)

- Developed at IBM in 1975, or perhaps...
- at National Security Agency (NSA). Nobody knows for certain.
- 16-round Feistel network.
- Key schedule derives permuted bits for each round key from a 56-bit key. Supposedly not 64-bit due to parity bits.
- Let us look a little at the Feistel-function f.

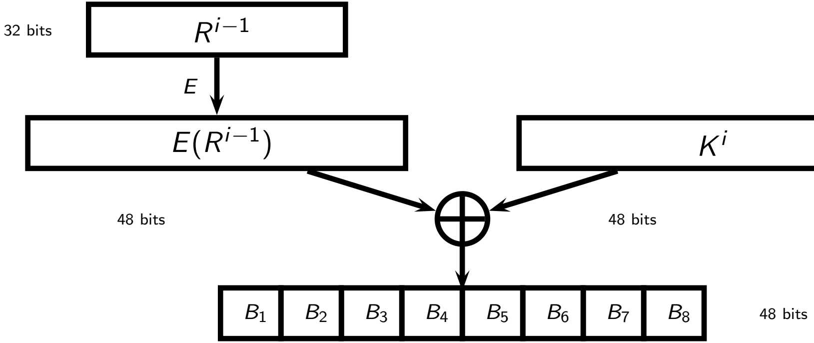
DES's f-Function



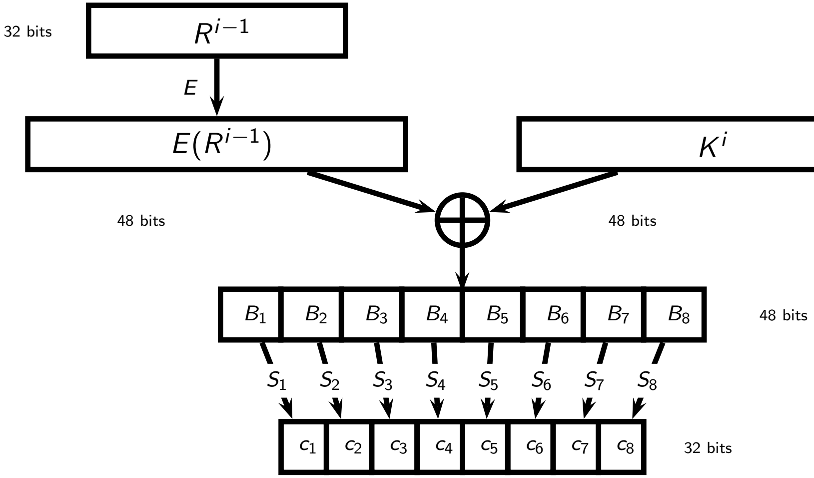
DES's f-Function



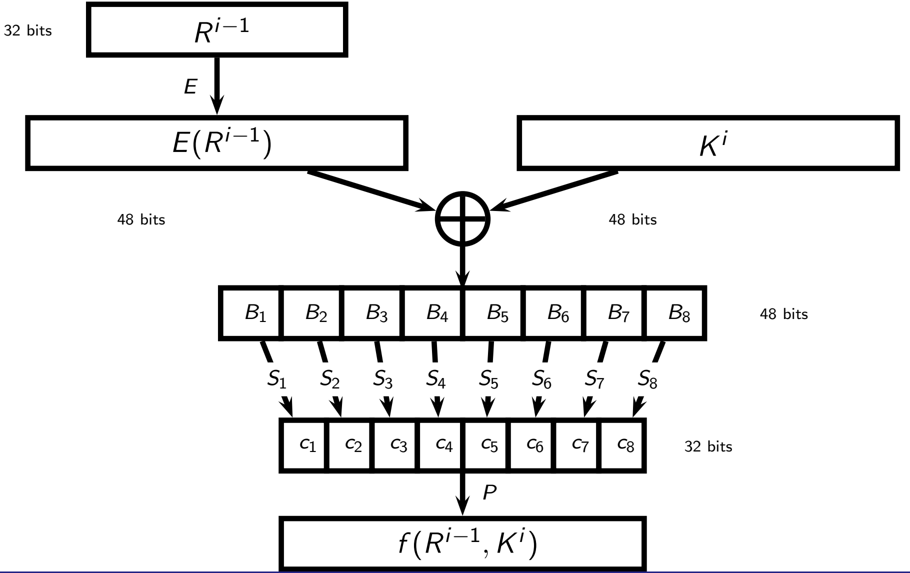
DES's f-Function



DES's f-Function



DES's f-Function



Security of DES

- **Brute Force.** Try all 2^{56} keys. Done in practice with special chip by Electronic Frontier Foundation, 1998. Likely much earlier by NSA and others.
- **Differential Cryptanalysis.** 2^{47} chosen plaintexts, Biham and Shamir, 1991. (approach: late 80'ies). Known earlier by IBM and NSA. DES is surprisingly resistant!
- **Linear Cryptanalysis.** 2^{43} known plaintexts, Matsui, 1993. Probably **not** known by IBM and NSA!

Double DES

We have seen that the key space of DES is too small. One way to increase it is to use DES twice, so called “double DES”.

$$\mathrm{2DES}_{k_1,k_2}(x) = \mathrm{DES}_{k_2}(\mathrm{DES}_{k_1}(x))$$

Is this more secure than DES?

This question is valid for any cipher.

Meet-In-the-Middle Attack

- Get hold of a plaintext-ciphertext pair (m, c)
- Compute $X = \{x \mid k_1 \in \mathcal{K}_{\mathrm{DES}} \wedge x = \mathsf{E}_{k_1}(m)\}$.
- For $k_2 \in \mathcal{K}_{\mathrm{DES}}$ check if $\mathsf{E}^{-1}_{k_2}(c) = \mathsf{E}_{k_1}(m)$ for some k_1 using the table X. If so, then (k_1, k_2) is a good candidate.
- Repeat with (m', c'), starting from the set of candidate keys to identify correct key.

Triple DES

What about triple DES?

$$3\mathrm{DES}_{k_1,k_2,k_3}(x) = \mathrm{DES}_{k_3}(\mathrm{DES}_{k_2}(\mathrm{DES}_{k_1}(x)))$$

- Seemingly 112 bit "effective" key size.
- 3 times as slow as DES. DES is slow in software, and this is even worse. One of the motivations of AES.
- Triple DES is still considered to be secure.

Modes of Operation

Modes of Operation

- Electronic codebook mode (ECB mode).
- Cipher feedback mode (CFB mode).
- Cipher block chaining mode (CBC mode).
- Output feedback mode (OFB mode).
- Counter mode (CTR mode).

ECB Mode

Electronic codebook mode

Encrypt each block independently:

$$c_i = \mathrm{E}_k(m_i)$$

ECB Mode

Electronic codebook mode

Encrypt each block independently:

$$c_i = \mathrm{E}_k(m_i)$$

- Identical plaintext blocks give identical ciphertext blocks.

ECB Mode

Electronic codebook mode

Encrypt each block independently:

$$c_i = \mathsf{E}_k(m_i)$$

- Identical plaintext blocks give identical ciphertext blocks.
- How can we avoid this?

CFB Mode

Cipher feedback mode

xor plaintext block with previous ciphertext block **after** encryption:

$$c_0 = \text{initialization vector}$$
$$c_i = m_i \oplus \mathsf{E}_k(c_{i-1})$$

CFB Mode

Cipher feedback mode

xor plaintext block with previous ciphertext block **after** encryption:

$$c_0 = \text{initialization vector}$$
$$c_i = m_i \oplus \mathsf{E}_k(c_{i-1})$$

- Sequential encryption and parallel decryption.

CFB Mode

Cipher feedback mode

xor plaintext block with previous ciphertext block **after** encryption:

$$c_0 = \text{initialization vector}$$
$$c_i = m_i \oplus \mathsf{E}_k(c_{i-1})$$

- Sequential encryption and parallel decryption.
- Self-synchronizing.

CFB Mode

Cipher feedback mode

xor plaintext block with previous ciphertext block **after** encryption:

$$c_0 = \text{initialization vector}$$
$$c_i = m_i \oplus \mathsf{E}_k(c_{i-1})$$

- Sequential encryption and parallel decryption.
- Self-synchronizing.
- How do we pick the initialization vector?

CBC Mode

Cipher block chaining mode

xor plaintext block with previous ciphertext block **before** encryption:

$$c_0 = \text{initialization vector}$$
$$c_i = \mathsf{E}_k\left(c_{i-1} \oplus m_i\right)$$

CBC Mode

Cipher block chaining mode

xor plaintext block with previous ciphertext block **before** encryption:

$$c_0 = \text{initialization vector}$$
$$c_i = \mathsf{E}_k\left(c_{i-1} \oplus m_i\right)$$

- Sequential encryption and parallel decryption

CBC Mode

Cipher block chaining mode

xor plaintext block with previous ciphertext block **before** encryption:

$$c_0 = \text{initialization vector}$$
$$c_i = \mathsf{E}_k\left(c_{i-1} \oplus m_i\right)$$

- Sequential encryption and parallel decryption
- Self-synchronizing.

OFB Mode

Output feedback mode

Generate stream, xor plaintexts with stream (emulate "one-time pad"):

$$s_0 = \text{initialization vector}$$
$$s_i = \mathsf{E}_k(s_{i-1})$$
$$c_i = s_i \oplus m_i$$

OFB Mode

Output feedback mode

Generate stream, xor plaintexts with stream (emulate "one-time pad"):

$$s_0 = \text{initialization vector}$$
$$s_i = \mathsf{E}_k(s_{i-1})$$
$$c_i = s_i \oplus m_i$$

- Sequential.

OFB Mode

Output feedback mode

Generate stream, xor plaintexts with stream (emulate "one-time pad"):

$$s_0 = \text{initialization vector}$$
$$s_i = \mathsf{E}_k(s_{i-1})$$
$$c_i = s_i \oplus m_i$$

- Sequential.
- Synchronous.

OFB Mode

Output feedback mode

Generate stream, xor plaintexts with stream (emulate "one-time pad"):

$$s_0 = \text{initialization vector}$$
$$s_i = \mathsf{E}_k(s_{i-1})$$
$$c_i = s_i \oplus m_i$$

- Sequential.
- Synchronous.
- Allows batch processing.

OFB Mode

Output feedback mode

Generate stream, xor plaintexts with stream (emulate "one-time pad"):

$$s_0 = \text{initialization vector}$$
$$s_i = \mathsf{E}_k(s_{i-1})$$
$$c_i = s_i \oplus m_i$$

- Sequential.
- Synchronous.
- Allows batch processing.
- Malleable!

CTR Mode

Counter mode

Generate stream, xor plaintexts with stream (emulate "one-time pad"):

$$
\begin{aligned}
s_0 &= \text{initialization vector} \\
s_i &= \mathsf{E}_k(s_0 \| i) \\
c_i &= s_i \oplus m_i
\end{aligned}
$$

CTR Mode

Counter mode

Generate stream, xor plaintexts with stream (emulate "one-time pad"):

$$s_0 = \text{initialization vector}$$
$$s_i = \mathrm{E}_k(s_0 \| i)$$
$$c_i = s_i \oplus m_i$$

- Parallel.

CTR Mode

Counter mode

Generate stream, xor plaintexts with stream (emulate "one-time pad"):

$$
\begin{aligned}
s_0 &= \text{initialization vector} \\
s_i &= \mathsf{E}_k(s_0 \| i) \\
c_i &= s_i \oplus m_i
\end{aligned}
$$

- Parallel.
- Synchronous.

CTR Mode

Counter mode

Generate stream, xor plaintexts with stream (emulate "one-time pad"):

$$s_0 = \text{initialization vector}$$
$$s_i = \mathrm{E}_k(s_0 \| i)$$
$$c_i = s_i \oplus m_i$$

- Parallel.
- Synchronous.
- Allows batch processing.

CTR Mode

Counter mode

Generate stream, xor plaintexts with stream (emulate "one-time pad"):

$$
\begin{aligned}
s_0 &= \text{initialization vector} \\
s_i &= \mathrm{E}_k(s_0 \| i) \\
c_i &= s_i \oplus m_i
\end{aligned}
$$

- Parallel.
- Synchronous.
- Allows batch processing.
- Malleable!

Ideal Block Cipher

Negligible Functions

Definition. A function $\epsilon(n)$ is negligible if for every constant $c > 0$, there exists a constant n_0, such that

$$\epsilon(n) < \frac{1}{n^c}$$

for all $n \geq n_0$.

Motivation. Events happening with negligible probability can not be exploited by polynomial time algorithms! (they "never" happen)

Pseudo-Random Function

"Definition". A function is pseudo-random if no efficient adversary can distinguish between the function and a random function.

Pseudo-Random Function

"Definition". A function is pseudo-random if no efficient adversary can distinguish between the function and a random function.

Definition. A family of functions $F : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$ is pseudo-random if for all polynomial time oracle adversaries A

$$\left| \Pr_K \left[A^{F_K(\cdot)} = 1 \right] - \Pr_{R:\{0,1\}^n \rightarrow \{0,1\}^n} \left[A^{R(\cdot)} = 1 \right] \right|$$

is negligible.