Private-Key Encryption

Federico Pintore[1]

[1]Mathematical Institute

Outline

Outline

Block Ciphers

- Block ciphers are designed to behave like (strong) pseudo-random permutations.
- They are maps of the form $F : \{0,1\}^n \times \{0,1\}^\ell \to \{0,1\}^\ell$ s.t. $F_k : \{0,1\}^\ell \to \{0,1\}^\ell$, $F_k(x) = F(k,x)$ is an efficiently computable permutation.
- Representing a permutation with ℓ-bit block size necessitates $\ell \cdot 2^\ell$ bits (infeasible for $\ell > 50$).
- Problem: how to represent permutations with big enough size for ℓ (for modern block ciphers $\ell \geq 128$)?

Confusion-Diffusion Paradigm

- **Confusion**: construct a random-looking permutation F_k with a large block length (e.g. 128 bits) using a set of random-looking permutations $\{f_i\}$ with smaller block length (e.g. 8 bits).
- Example: given $x \in \{0,1\}^{128}$, parse it as $x_1, \cdots, x_{16}$, where $|x_i| = 8$ bits and define

$$F_k(x) = f_1(x_1)||\cdots||f_{16}(x_{16}).$$

- If two inputs x and x' have only one different bit, then $F_k(x)$ and $F_k(x')$ have only one different byte.
- **Diffusion**: use a mixing permutation to make a change in one bit affect the entire output block!
- f_i is called *round function*; the confusion/diffusion steps together are called *round*

Substitution-permutation Networks (SPNs)

- A substitution-permutation network is an implementation of the confusion-diffusion paradigm.
- Using a fixed public algorithm called *key schedule*, we derive sub-keys $k_1, \ldots, k_{r+1}$ from a master key k.
- Different permutations $\{S_i\}$ with small block length are used to define the round functions:

$$f_i(x_i) = S_i(x_i + k_{\cdot,i})$$

- S_i is called *S-box*

Key-Schedule: simple example

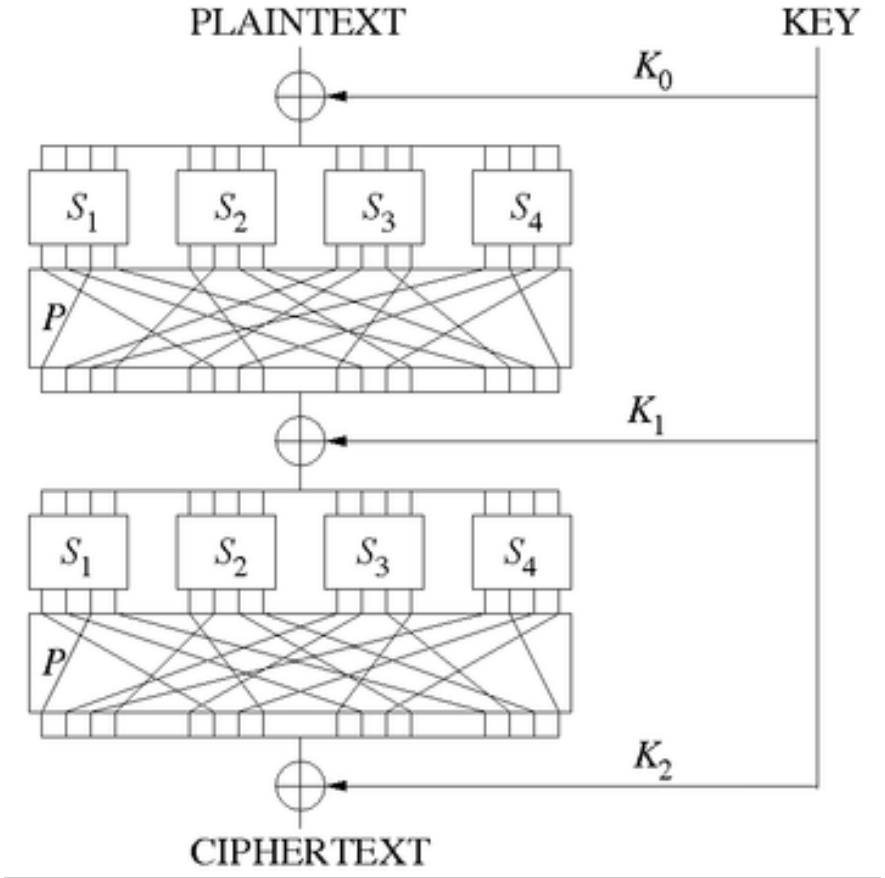
Suppose that the master key is as follows:

$$k = 1110\ 0111\ 0110\ 0111\ 1001\ 0000\ 0011\ 1101$$

Let k_i be 16 consecutive bits of k starting at bit $4i - 3$:

- $k_1 = 1110\ 0111\ 0110\ 0111$
- $k_2 = 0111\ 0110\ 0111\ 1001$
- $k_3 = 0110\ 0111\ 1001\ 0000$
- $k_4 = 0111\ 1001\ 0000\ 0011$
- $k_5 = 1001\ 0000\ 0011\ 1101$

SPN - Example



SPN - Example

Algorithm

Inputs: plaintext, S-boxes, mixing permutation P, $(k_1, \ldots, k_{r+1})$
Output: ciphertext

state = plaintext
For round $j = 1$ to r do
    key-mixing: state = state $\oplus k_j$
    substitution: apply S-boxes to the m sub-strings of state
    permutation: apply P to state
end do
ciphertext = state $\oplus k_{r+1}$

SPNs

Avalanche effect

- S-boxes: designed in a way so that a change of 1-bit in the input determines a change of *at least* two bits in the output.
- mixing permutation P: make sure that the bits of the output of one S-box will be fed to *multiple* S-boxes in the next round.

SPNs

- The Advanced Encryption Standard (AES) has similar structure (will see it soon).
- The security of a SPN depends on the number of rounds.
- for a SPN with a single round with no key-mixing at the final step, it is easy to recover the master key
- A one round SPN is also not secure
- Same for a two round SPN!

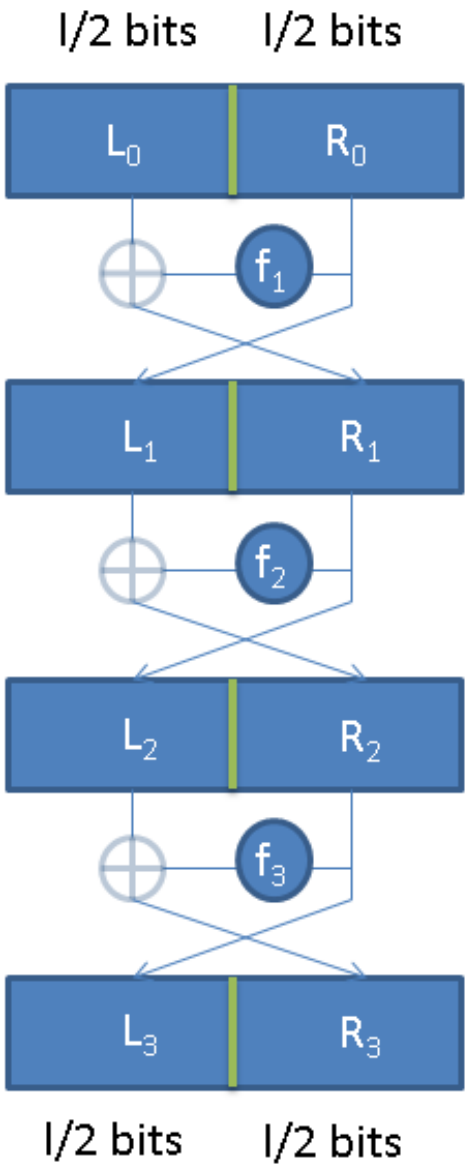
Feistel Networks

- Different approach to construct block ciphers following the confusion-diffusion paradigm.
- Advantage over SPNs: the round functions need not be permutations.
- Feistel Network: Given functions $f_1, \cdots, f_r$, where $f_i : \{0,1\}^{\ell/2} \to \{0,1\}^{\ell/2}$, construct a **permutation** $F_k : \{0,1\}^{\ell} \to \{0,1\}^{\ell}$.

Encryption:

Li=Ri-1

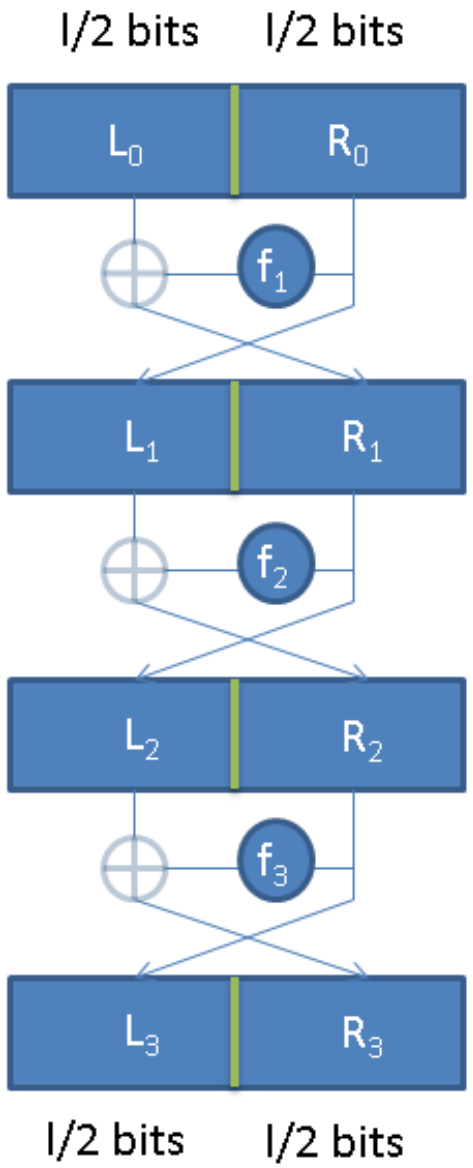
Ri=Li-1 $\oplus$ fi(Ri-1)



Encryption:

Li=Ri-1

Ri=Li-1 $\oplus$ fi(Ri-1)



Decryption:

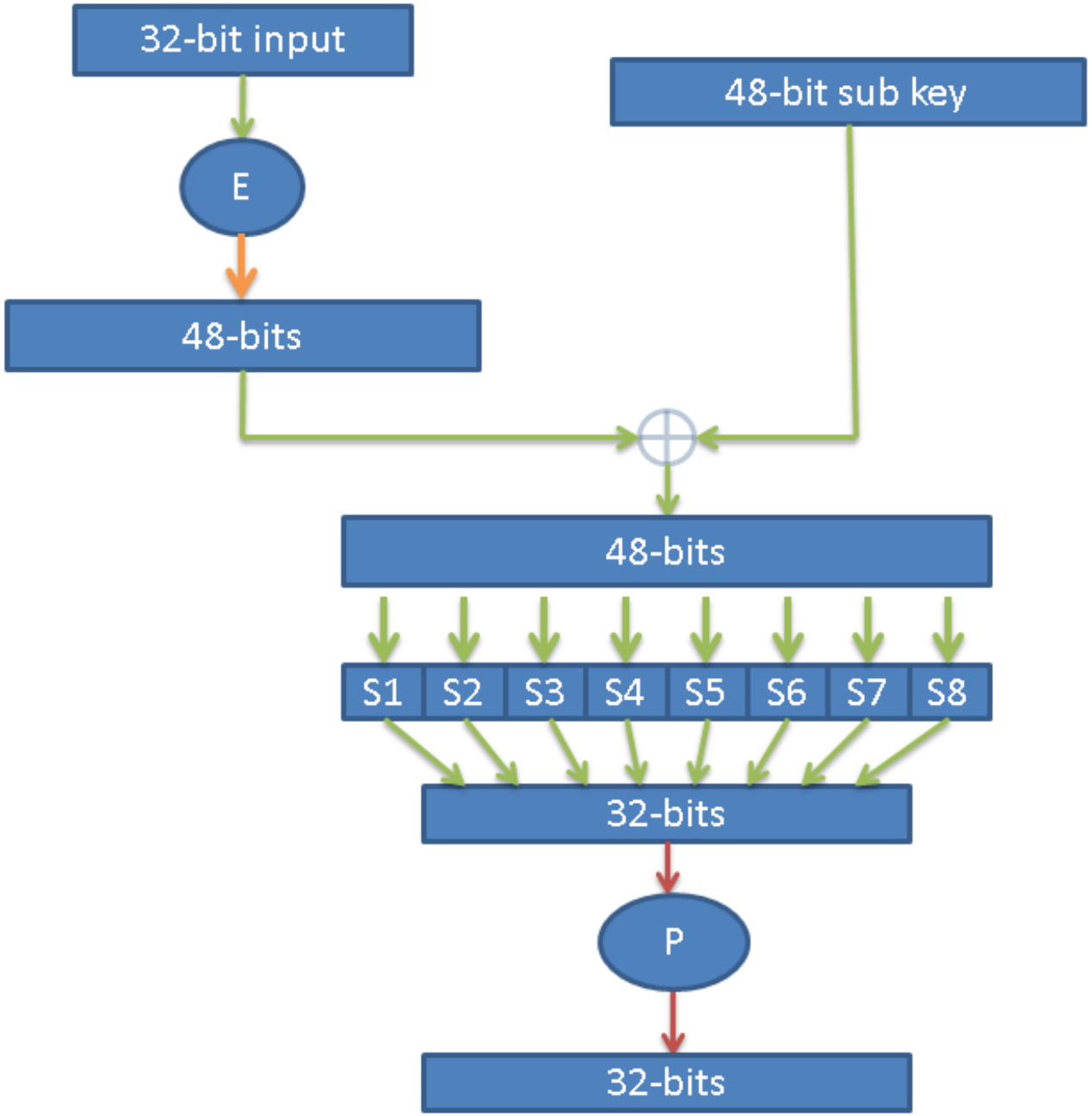
Ri-1=Li

Li-1=Ri $\oplus$ fi(Ri-1)

Outline

The Data Encryption Standard (DES)

- It is a 16 round Feistel Network.
- Block length $\ell = 64$; key length $n = 56$.
- The same round function f is used in all the 16 rounds.
- The key schedule derives 16 sub-keys of size 48 bits, $k_1, \cdots, k_{16}$, from the **secret** master key.
- $f : \{0,1\}^{32} \times \{0,1\}^{48} \rightarrow \{0,1\}^{32}$
- An *expansion function* $E : \{0,1\}^{32} \rightarrow \{0,1\}^{48}$ is used to duplicate half of the bits.
- It also uses 8 different and non invertible S-boxes, $S_1, \cdots, S_8$, where S_i takes a 6-bit input and produces a 4-bit output.
- A simple animation for DES, `http://kathrynneugent.com/animation.html`



The DES function

S-box example

Think of S-boxes as code books, i.e. they replace words with other words.

S_5		Middle 4 bits of input															
		0000	**0001**	**0010**	**0011**	**0100**	**0101**	**0110**	**0111**	**1000**	**1001**	**1010**	**1011**	**1100**	**1101**	**1110**	**1111**
Outer bits	**00**	0010	1100	0100	0001	0111	1010	1011	0110	1000	0101	0011	1111	1101	0000	1110	1001
	01	1110	1011	0010	1100	0100	0111	1101	0001	0101	0000	1111	1010	0011	1001	1000	0110
	10	0100	0010	0001	1011	1010	1101	0111	1000	1111	1001	1100	0101	0110	0011	0000	1110
	11	1011	1000	1100	0111	0001	1110	0010	1101	0110	1111	0000	1001	1010	0100	0101	0011

Security of DES

- 1970: Horst Feistel designs Lucifer (DES' precursor) at IBM, where $n = \ell = 128$
- 1976: NBS adopts DES as a federal standard $n = 56, \ell = 64$
- 1997: first successful brute-force attack on DES broken (DESCHALL project, using 96 days)
- State-of-the-art: brute-force attack takes $\approx$ 23 hours (DES cracking box by PICO computing)
- Conclusion: the key length used by DES is too short!

Security of DES

Can we do better than brute-force attacks on DES?

- *Differential cryptanalysis* - by Biham-Shamir late 1980s: takes time 2^{37} (DES computations) but requires 2^{47} **chosen** plaintexts.
- Theoretically speaking, it was a breakthrough, but not a realistic attack regarding the number of encryptions of chosen plaintext.
- *Linear cryptanalysis* - by Matsui mid 1990s: takes time 2^{43} and requires 2^{42} of **known** plaintexts (still a huge number, but with the advantage of being known rather than chosen).

2DES

- DES's main problem is its short key size.
- Changing the internal structure of DES was not recommended.
- What if we double the key length defining

$$F'_{k_1,k_2} \leftarrow F_{k_2} \circ F_{k_1}$$

- Not a great idea! A meet-in-the-middle attack takes time $\mathcal{O}(n \cdot 2^n)$ and requires space $\mathcal{O}((n+\ell) \cdot 2^n)$.

2DES: meet-in-the-middle-attack

Given a pair of input/output $(x, y = F_{k_2^*}(F_{k_1^*}(x)))$, to minimise the set of candidate keys, the adversary can maintain the two lists L_1 and L_2 as follows;

- $\forall k_1 \in \{0,1\}^n$, compute $z \leftarrow F_{k_1}(x)$, and store $L_1 \leftarrow (z, k_1)$
- $\forall k_2 \in \{0,1\}^n$, compute $z \leftarrow F_{k_2}^{-1}(y)$, and store $L_2 \leftarrow (z, k_2)$

The adversary creates a third list M containing all the *match* pairs (k_1, k_2) for which their corresponding z_1 and z_2 in L_1 and L_2 are equal.

The entry $(k_1^*, k_2^*) \in M$ can be identified with very high probability.

3DES

We have two versions:

- Choose independent keys $k_1, k_2, k_3 \in \{0,1\}^n$ and define

$$F''_{k_1,k_2,k_3} \leftarrow F_{k_3} \circ F_{k_2}^{-1} \circ F_{k_1}$$

- Meet-in-the-middle attack takes 2^{2n}.

- Use two keys $k_1, k_2 \in \{0,1\}^n$ and define

$$F''_{k_1,k_2} \leftarrow F_{k_1} \circ F_{k_2}^{-1} \circ F_{k_1}$$

- Best attack takes time 2^{2n} (if the attacker is only given a small number of plaintext/ciphertext pairs).

3DES

We have two versions:

- Choose independent keys $k_1, k_2, k_3 \in \{0,1\}^n$ and define

$$F''_{k_1,k_2,k_3} \leftarrow F_{k_3} \circ F_{k_2}^{-1} \circ F_{k_1}$$

- Meet-in-the-middle attack takes 2^{2n}.

- Use two keys $k_1, k_2 \in \{0,1\}^n$ and define

$$F''_{k_1,k_2} \leftarrow F_{k_1} \circ F_{k_2}^{-1} \circ F_{k_1}$$

- Best attack takes time 2^{2n} (if the attacker is only given a small number of plaintext/ciphertext pairs).

Security of 3DES

- It was standardized in 1999.
- Drawbacks: it has small block length and it runs slowly (it requires three block cipher executions!)
- The best security level that it can offer is 2^{112} whereas the current recommendation is 2^{128}
- For higher security levels, check this to know about magic numbers: `http://www.iacr.org/conferences/eurocrypt2012/Rump/shamir.pdf`
- Any alternative?

Outline

The Advanced Encryption Standard (AES)

- In 1997, the United States National Institute of Standards and Technology (NIST) called for a competition for a new block cipher called AES.
- In 2000, Rijndael, a new block cipher, designed by Vincent Rijmen and Joan Daemen, won the competition.
- AES block cipher, has a 128-bit block length.
- The key lenght can be of 128, 192, or 256 bits.
- AES is a substitution-permutation network (SPN).
- The *state* in AES is a 4×4 array of bytes that will be modified each round. The initial value of the state is the input to the block cipher.

AES' round: the four stages

- **AddRoundKey**: Xor the state with a 128-bit round key, that is generated using the master key.
- **SubBytes**: Apply a fixed S-box to each byte of the state. The S-box is a bijection over $\{0,1\}^8$.
- **ShiftRows**: Shift the bytes in each row of the state to the left. The number of steps is 0, 1, 2 and 3, respectively.
- **MixColumns**: Apply a linear transformation (a matrix multiplication over the Galois field F_{2^8}).

AES

A nice animation of AES:

`http://www.formaestudio.com/rijndaelinspector/archivos/Rijndael_Animation_v4_eng.swf`

Security of AES

- No practical attacks that are notably better than brute-force search for the key.
- A great implementation for a (strong) pseudo-random permutation.
- Free, standardized, and efficient.

Outline

Linear Attacks

No Linear combination of output bits should be too close to a linear combination of the input bits

- The linearity here refers to $\oplus$ (a mod 2 bit-wise operation)
- Given the inputs $\{X_1, \cdots, X_s\}$ and the corresponding outputs $\{Y_1, \cdots, Y_t\}$, compute

$$L = \bigoplus_{i=1}^{s} X_i \bigoplus_{j=1}^{t} Y_i$$

- Define the *linear probability bias* as $P_L = |\Pr[L = 0] - 1/2|$.
- The higher P_L, the more applicable the linear attacks (i.e. fewer known plaintexts are required)

Differential Attacks

- Given a pair of inputs (X_1, X_2) and corresponding outputs (Y_1, Y_2), exploit the relationship between $\Delta X = X_1 \oplus X_2$ and $\Delta Y = Y_1 \oplus Y_2$.
- Ideally, $P_D = \Pr[\Delta Y = d_2 | \Delta X = d_1] = 1/2^n$, for some d_1, d_2 (n is the number of bits of X_i, Y_i).
- "Differential Cryptanalysis" is interested in $(\Delta X, \Delta Y)$ s.t. $P_D \gg 1/2^n$.
- It is a chosen plaintext attack, so attacker aims at encrypting particular plain-texts $\{X_{i_1}, X_{i_2}\}$ for which he knows that a certain ΔY_i occurs with high probability.

Quantum attacks on Block Ciphers

Search problem: let $f : X \to \{0,1\}$ be a function. Find the only $x \in X$ s.t. $f(x) = 1$.

- Classical computers: the best algorithm is a generic algorithm which runs in time $\mathcal{O}(|X|)$.
- Quantum computers (when they exist?): according to [Grover'96], the runnning time is $\mathcal{O}\left(\sqrt{|X|}\right)$ (quadratic speedup).
- Given m and $c = \text{Enc}(k, m)$, define $f(k) = 1$ if $\text{Enc}(k, m) = c$ and 0 otherwise. Quantum algorithm can find the key k in time $\mathcal{O}\left(\sqrt{|\mathcal{K}|}\right)$.
- **Conclusion:** Symmetric key lengths should be doubled to protect against quantum attacks, e.g. we will need AES-256 to achieve 2^{128} post-quantum security.

Outline

Fixed-length Encryption scheme using a PRG

Let G be a pseudorandom generator with expansion factor $\ell(n)$. For messages of length $\ell(n)$, we define the following encryption scheme $E = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$:

- $\mathsf{KeyGen}(n)$: It picks a uniform bit string k of length n, i.e. $k \in \{0,1\}^n$.
- Enc : it takes as input a key $k \in \{0,1\}^n$ and a message $m \in \{0,1\}^{\ell(n)}$, it outputs

$$c \leftarrow \mathsf{Enc}(G(k), m) = G(k) \oplus m$$

- Dec : it takes as input a key $k \in \{0,1\}^n$ and a ciphertext $c \in \{0,1\}^{\ell(n)}$, it outputs

$$m \leftarrow \mathsf{Dec}(G(k), c) = G(k) \oplus c.$$

Encryption using PRP

Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a pseudo-random permutation. We define the following encryption scheme $E = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$:

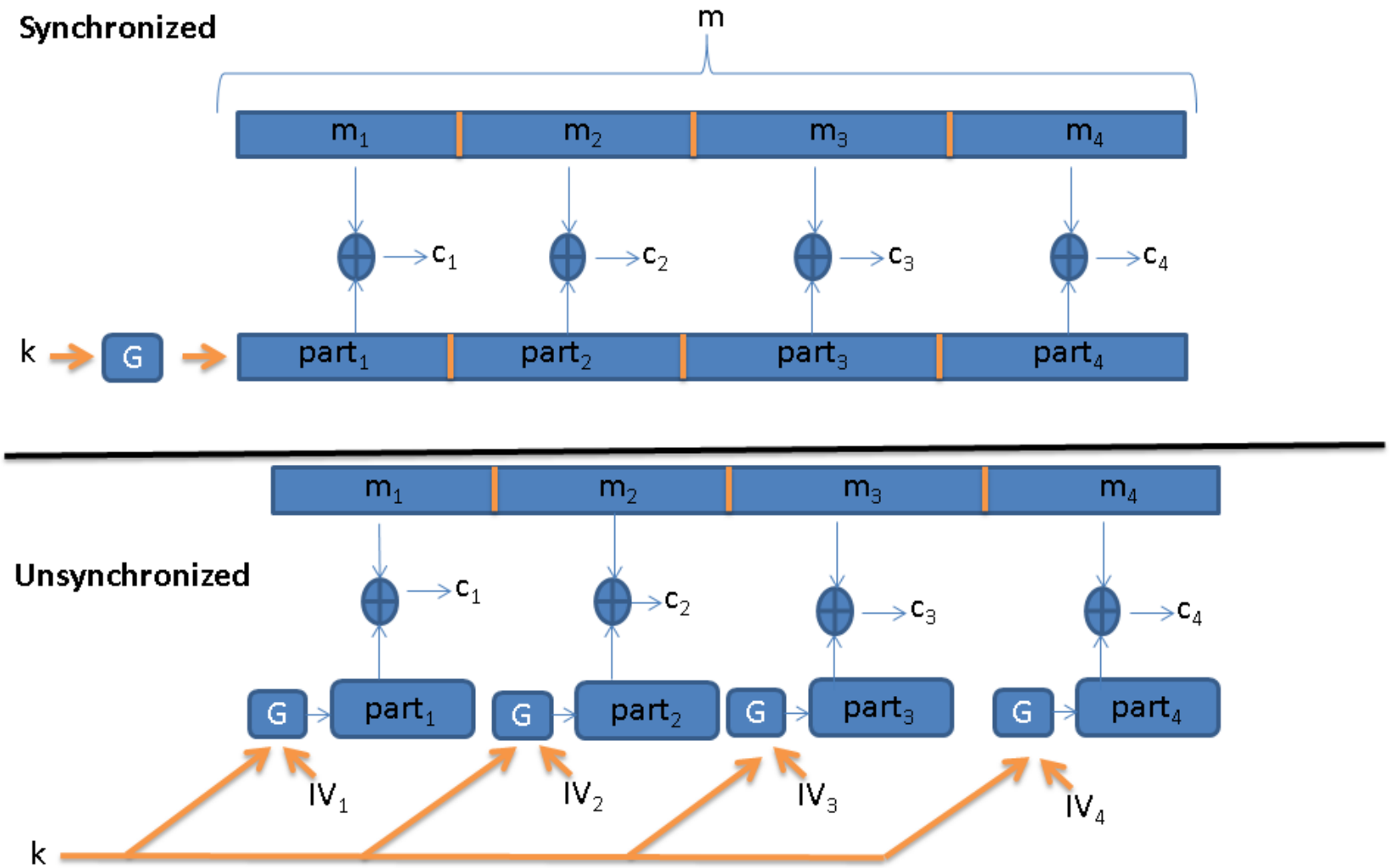
- KeyGen : it takes n and outputs a key $k \in \{0,1\}^n$.
- Enc: it takes a key $k \in \{0,1\}^n$ and a message m, it picks a random $r \leftarrow \{0,1\}^n$, and outputs

$$(c_0, c_1) \leftarrow (r, F_k(r) \oplus m).$$

- Dec: it takes a key k and a ciphertext $c = (c_0, c_1)$ and outputs

$$m \leftarrow (F_k(c_0) \oplus c_1).$$

Stream Cipher Modes of Operation



Stream Cipher Modes of Operation

- Synchronised mode
  - It gives a stateful encryption scheme
  - Sender and Receiver must be synchronised
  - It generates a long pseudo-random stream (different parts of it are used to encrypt different messages)
  - Therefore, messages should be received in order

- Unsynchronised mode
  - It needs initialisation vectors
  - It gives *stateless* CPA-secure encryption.

Stream Cipher Modes of Operation

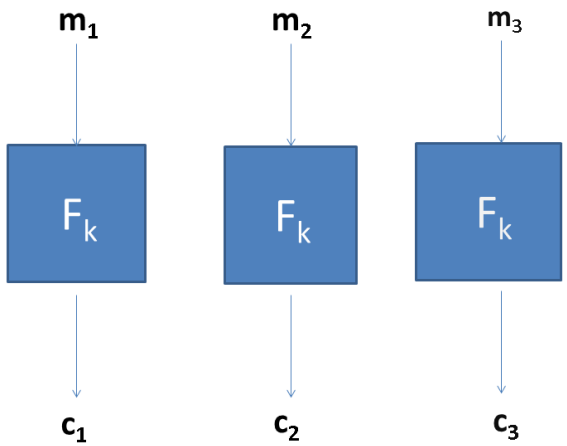
- Synchronised mode
  - It gives a stateful encryption scheme
  - Sender and Receiver must be synchronised
  - It generates a long pseudo-random stream (different parts of it are used to encrypt different messages)
  - Therefore, messages should be received in order

- Unsynchronised mode
  - It needs initialisation vectors
  - It gives *stateless* CPA-secure encryption.

Block Ciphers Modes of Operation

- Block ciphers are secure instantiations of pseudo-random permutations where key length and block length are fixed.
- Block ciphers modes of operation allow to encrypt *arbitrary-length* messages with shorter chipertexts (the aforementioned CPA-secure encryption scheme produces ciphertexts which are double the length of the plaintexts).
- F is a block cipher with block length n, all messages are assumed to have length multiple of n.

Electronic Code Book (ECB) mode

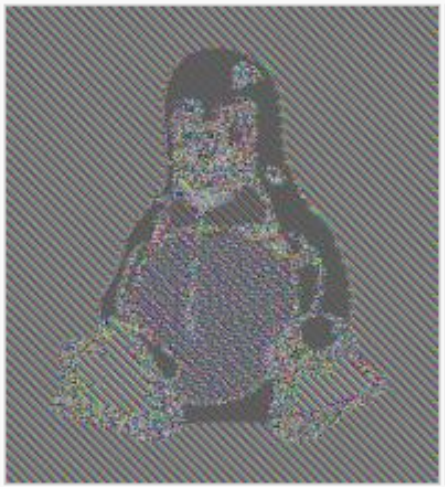


- ECB mode is deterministic $\Rightarrow$ cannot be CPA secure.
- Repetition of blocks in plaintext $\Rightarrow$ repetition of blocks in ciphertext!
- Doesn't even have indistinguishable encryptions in the presence of an eavesdropper.

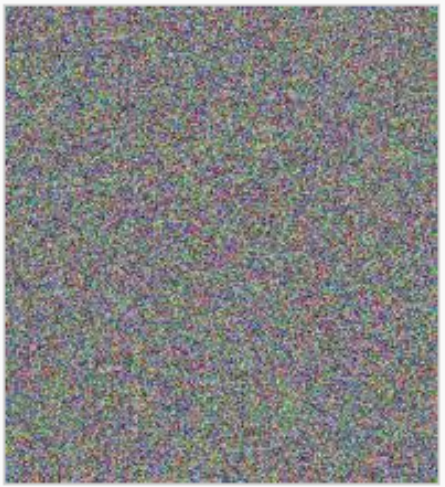
Electronic Code Book (ECB) mode

Source: Wikipidea.

Original image



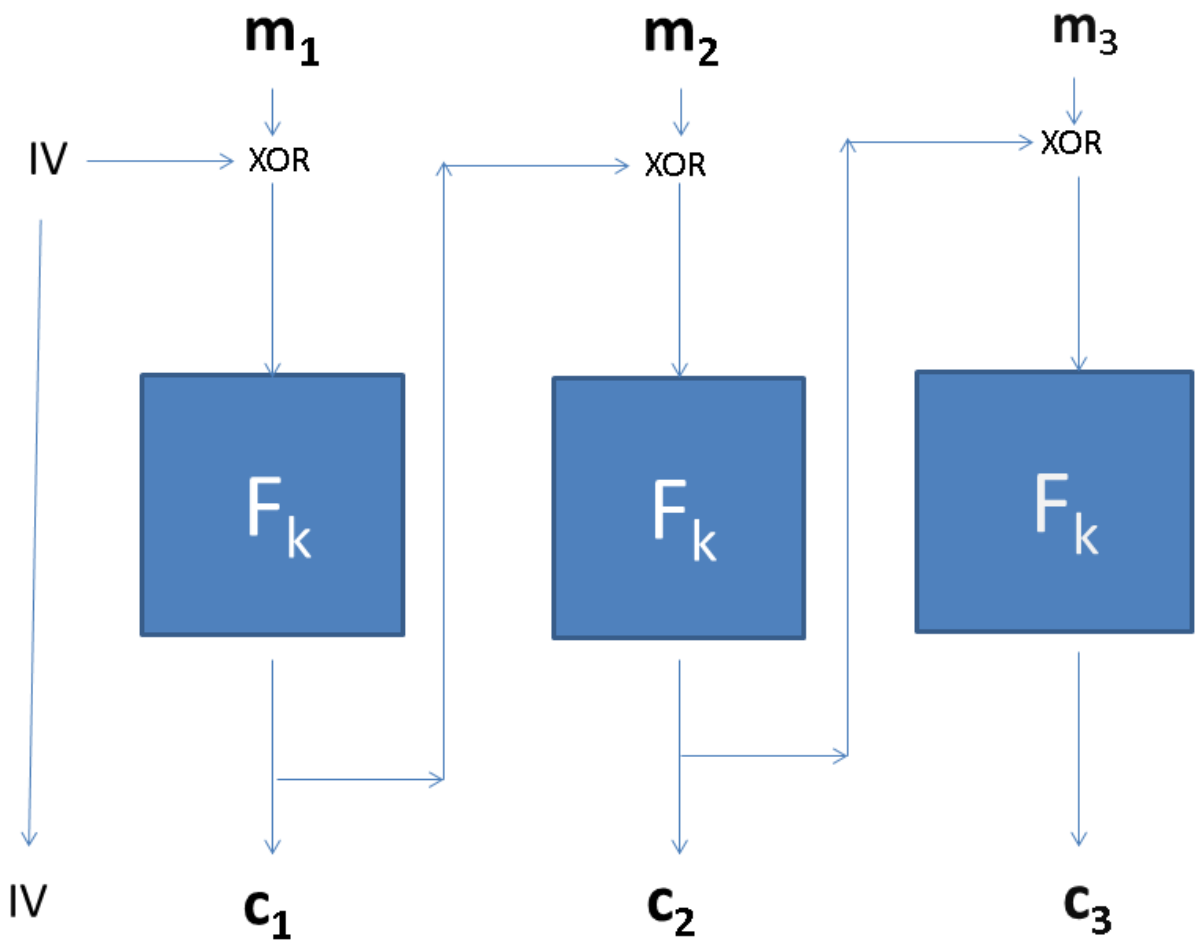
Encrypted using ECB mode



Modes other than ECB result in pseudo-randomness

The image on the right is how the image might appear encrypted with CBC, CTR or any of the other more secure modes—indistinguishable from random noise. Note that the random appearance of the image on the right does not ensure that the image has been securely encrypted; many kinds of insecure encryption have been developed which would produce output just as "random-looking".

Cipher Block Chaining (CBC) mode



Cipher Block Chaining (CBC) mode

- Enc: On input $m = m_1 m_2 \cdots m_\ell$ and a block cipher of block length n, i.e. F_k, outputs

$$c_i \leftarrow F_k(c_{i-1} \oplus m_i), \text{ for } i = 1 \cdots \ell.$$

  where $c_0 = IV$.

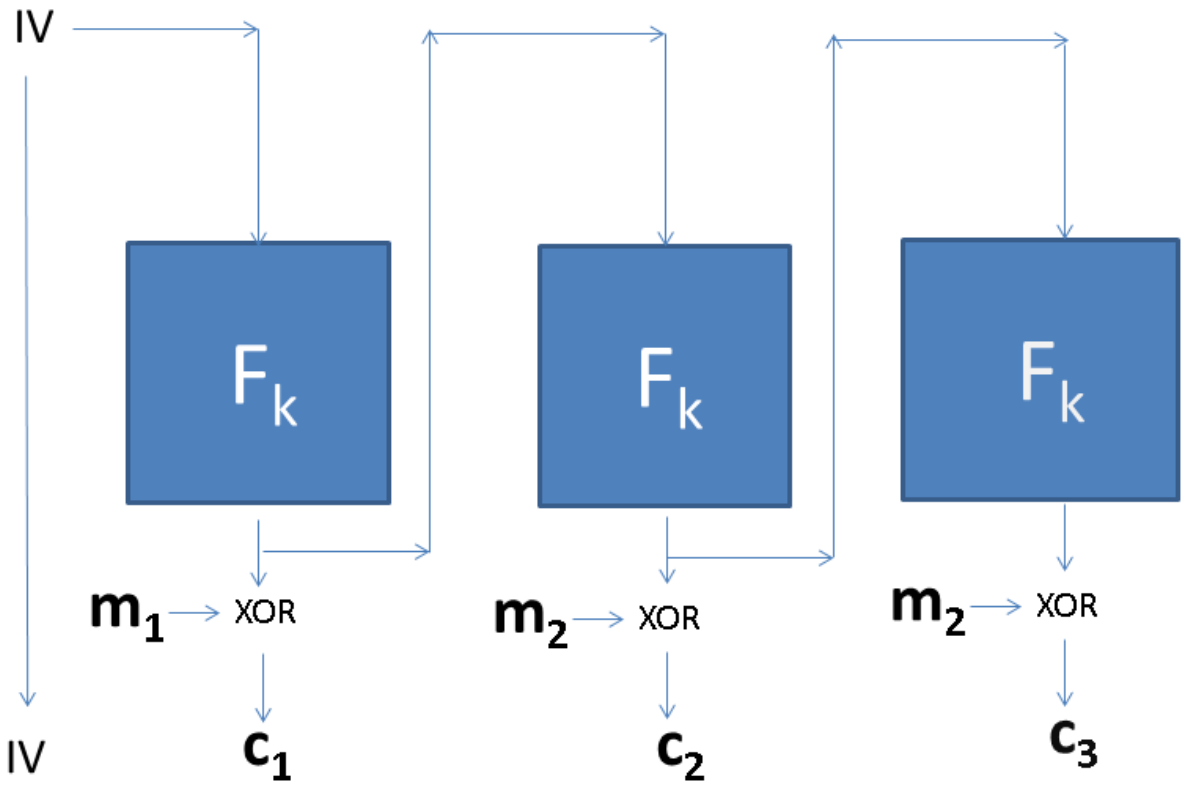
- Dec: On input $c = c_0 c_1 c_2 \cdots c_\ell$ and a block cipher of block length n, i.e. F_k^{-1}, outputs

$$m_i \leftarrow F_k^{-1}(c_i) \oplus c_{i-1}, \text{ for } i = 1 \cdots \ell.$$

CBC mode

- CBC is probabilistic.
- If F is a pseudo-random permutation, than CBC-mode encryption is CPA-secure.
- *Stateful* variant of CBC: the last block of a given ciphertext is used as the IV to encrypt the next message.
- This variant of CBC is called *Chained* CBC. It is used in SSL 3.0 and TLS 1.0, but it is not CPA-secure!
- Efficiency: encryption must be executed sequentially (no parallel processing).

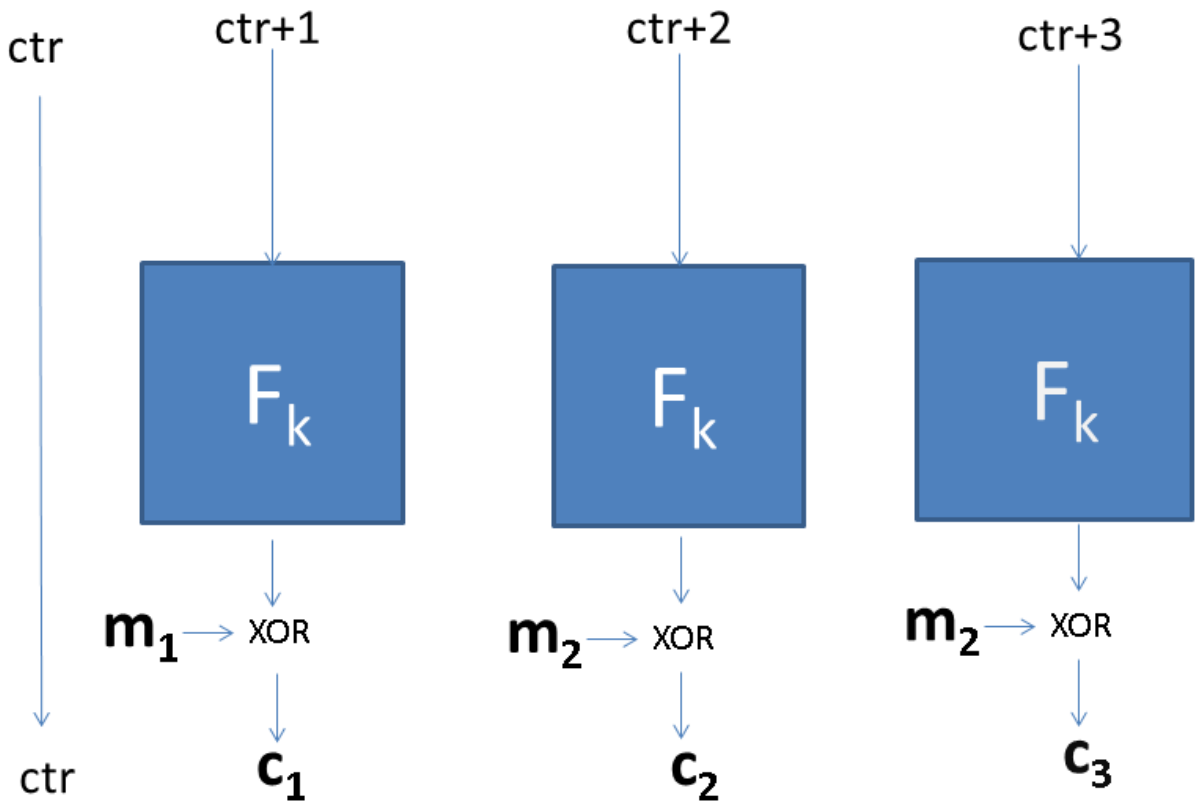
Output Feedback (OFB) mode



OFB mode

- The OFB mode works as follows:
  - Choose $IV \in_R \{0,1\}^n$ uniformly.
  - Let $y_0 = IV$, set $y_i = F_k(y_{i-1})$.
  - Enc: $c_0 = y_0$, $c_i \leftarrow y_i \oplus m_i$.
  - Dec: $m_i \leftarrow y_i \oplus c_i$.
- F_k does not have to be invertible.
- if F is a pseudo-random function, then the OFB mode is CPA-secure.
- Most of the computation can be done before encrypting/decrypting.
- Its stateful variant is secure

Counter (CTR) mode



CTR mode

- The CTR mode works as follows:
  - Choose ctr $\in_R \{0,1\}^n$ uniformly.
  - Compute $y_i = F_k(\text{ctr} + i \mod 2^n)$.
  - Enc: $c_i \leftarrow y_i \oplus m_i$.
  - Dec: $m_i \leftarrow y_i \oplus c_i$.
- F_k does not have to be invertible.
- Parallel processing is possible.

CTR mode

Theorem

If F is a pseudo-random function, then CTR mode is CPA-secure.

Proof.

Similar to the previous proof.

We obtain:

$$\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},E}(n) = 1] < 1/2 + 2q(n)^2/2^n + \mathsf{negl}(n)\,.$$

where $q(n)$ is a polynomial upper-bound on the number of encryption-oracle queries made by the adversary $\mathcal{A}$. □

The stateful version of CTR mode is secure.

Initialisation Vector IV

- CBC, OFB, and CTR all use a random IV.
- One has to make sure that the IV is not repeating!
- Even if F is a secure pseudo-random permutation, the size of the block cannot be too short.
- The block length for DES is $\ell = 64$; after you encrypt data of size $2^{32} bits \approx 34$ gigabytes, you can expect a repeated IV! (hint: see birthday paradox - we will cover it)
- In practice, if you have a repeated IV, then CBC is better that OFB and CTR.

Initialisation Vector IV

- if IV repeats with OFB or CTR, the attacker can XOR the two resulting ciphertexts to learn about the encrypted plaintext.
- In CBC mode, after few blocks the inputs to the block cipher F_k will "diverge".
- To solve the IV issue, either use stateful variants of OFB and CTR, or the regular CBC mode.
- Remember, in OFB/CTR stateful variants, the final value y_ℓ, i.e. $y_\ell = F_k(y_{\ell-1})$ or $y_\ell = F_k(\text{ctr} + \ell \mod 2^n)$, will play the role of the IV when encrypting the next message.

Further Reading (1)

- Don Coppersmith.
  The data encryption standard (DES) and its strength against attacks.
  IBM journal of research and development, 38(3):243–250, 1994.
- Itai Dinur, Orr Dunkelman, Masha Gutman, and Adi Shamir.
  Improved top-down techniques in differential cryptanalysis.
  Cryptology ePrint Archive, Report 2015/268, 2015.
  `http://eprint.iacr.org/`.

Further Reading (2)

- Itai Dinur, Orr Dunkelman, Nathan Keller, and Adi Shamir. Efficient dissection of composite problems, with applications to cryptanalysis, knapsacks, and combinatorial search problems. Cryptology ePrint Archive, Report 2012/217, 2012. http://eprint.iacr.org/.
- Itai Dinur, Orr Dunkelman, Nathan Keller, and Adi Shamir. New attacks on feistel structures with improved memory complexities. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology – CRYPTO 2015*, volume 9215 of *Lecture Notes in Computer Science*, pages 433–454. Springer Berlin Heidelberg, 2015.

Further Reading (3)

- Lov K Grover.
  A fast quantum mechanical algorithm for database search.
  In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219. ACM, 1996.
- Howard M Heys.
  A tutorial on linear and differential cryptanalysis.
  Cryptologia, 26(3):189–221, 2002.