Private-Key Encryption

Ali El Kaafarani

[1]Mathematical Institute
[2] PQShield Ltd.

Outline

Outline

Block Ciphers – Modes of Operations

- Block ciphers are secure instantiations of pseudo-random permutations where key length and block length are fixed.
- Block ciphers modes of operations allow to encrypt *arbitrary-length* messages with ciphertext shorter than our aforementioned CPA-secure encryption scheme (in which the ciphertext was double the length of the plaintext).
- All messages are assumed to have length multiple of n.
- F is a block cipher with block length n.

Block Ciphers – Modes of Operations

- Block ciphers are secure instantiations of pseudo-random permutations where key length and block length are fixed.
- Block ciphers modes of operations allow to encrypt *arbitrary-length* messages with ciphertext shorter than our aforementioned CPA-secure encryption scheme (in which the ciphertext was double the length of the plaintext).
- All messages are assumed to have length multiple of n.
- F is a block cipher with block length n.

Block Ciphers – Modes of Operations

- Block ciphers are secure instantiations of pseudo-random permutations where key length and block length are fixed.
- Block ciphers modes of operations allow to encrypt *arbitrary-length* messages with ciphertext shorter than our aforementioned CPA-secure encryption scheme (in which the ciphertext was double the length of the plaintext).
- All messages are assumed to have length multiple of n.
- F is a block cipher with block length n.

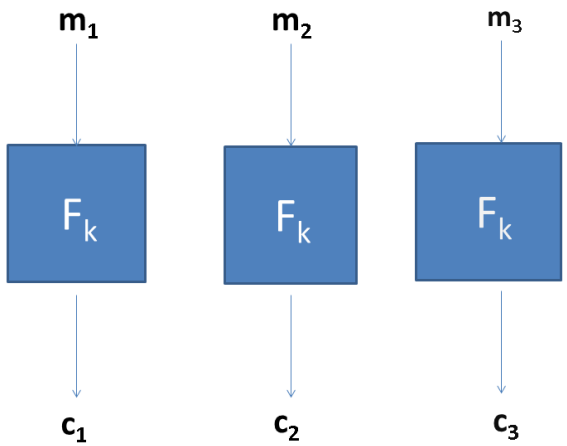
Block Ciphers – Modes of Operations

- Block ciphers are secure instantiations of pseudo-random permutations where key length and block length are fixed.
- Block ciphers modes of operations allow to encrypt *arbitrary-length* messages with ciphertext shorter than our aforementioned CPA-secure encryption scheme (in which the ciphertext was double the length of the plaintext).
- All messages are assumed to have length multiple of n.
- F is a block cipher with block length n.

Block vs. Stream Ciphers

- Block ciphers process plaintexts in large blocks ($|\text{block}| \geq 64$ bits).
- Functions in block ciphers (usually) don't have a memory (stateless). The same function is used to encrypt different blocks in a given message.
- Stream ciphers process plaintext in shorter blocks (down to 1 bit!)

Electronic Code Book (ECB) mode

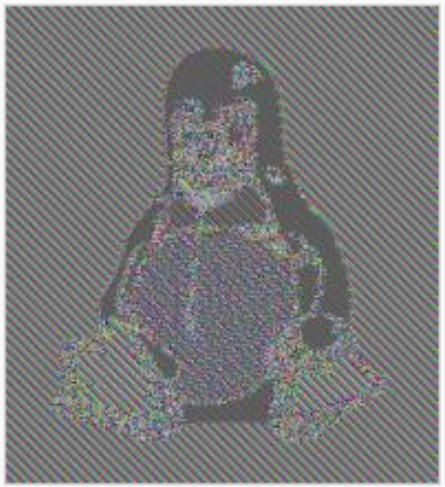


- ECB mode is deterministic $\Rightarrow$ can't be CPA secure.
- Repetition of blocks in plaintext $\Rightarrow$ repetition of blocks in ciphertext!
- Doesn't even have indistinguishable encryptions in the presence of an eavesdropper.

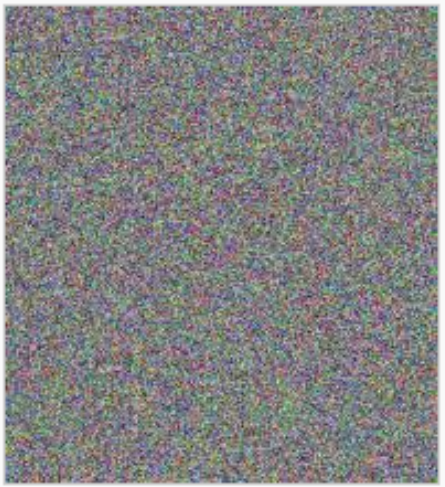
Electronic Code Book (ECB) mode

Source: Wikipidea.

Original image



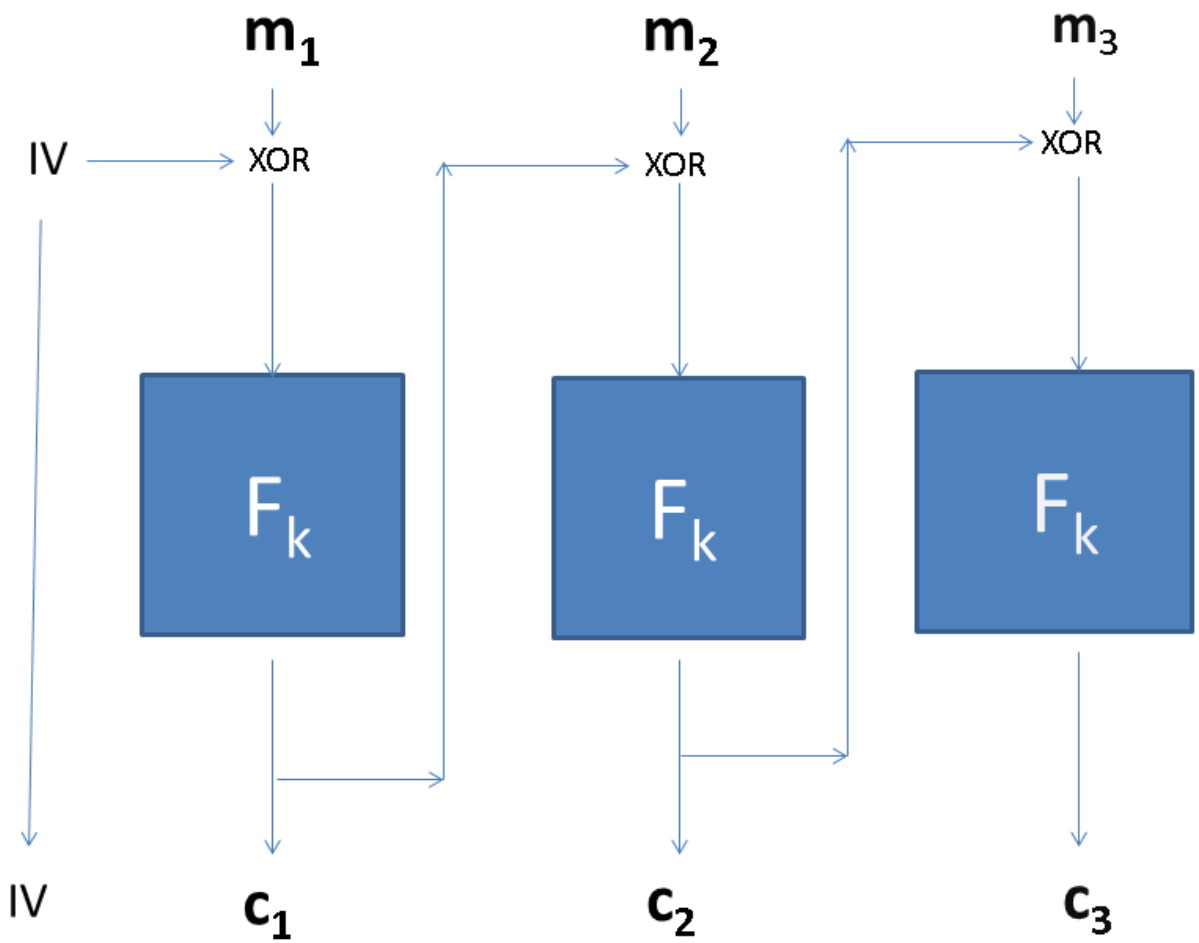
Encrypted using ECB mode



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 inputs $m = m_1 m_2 \cdots m_\ell$ and a block cipher of block length n, i.e. F_k , output

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

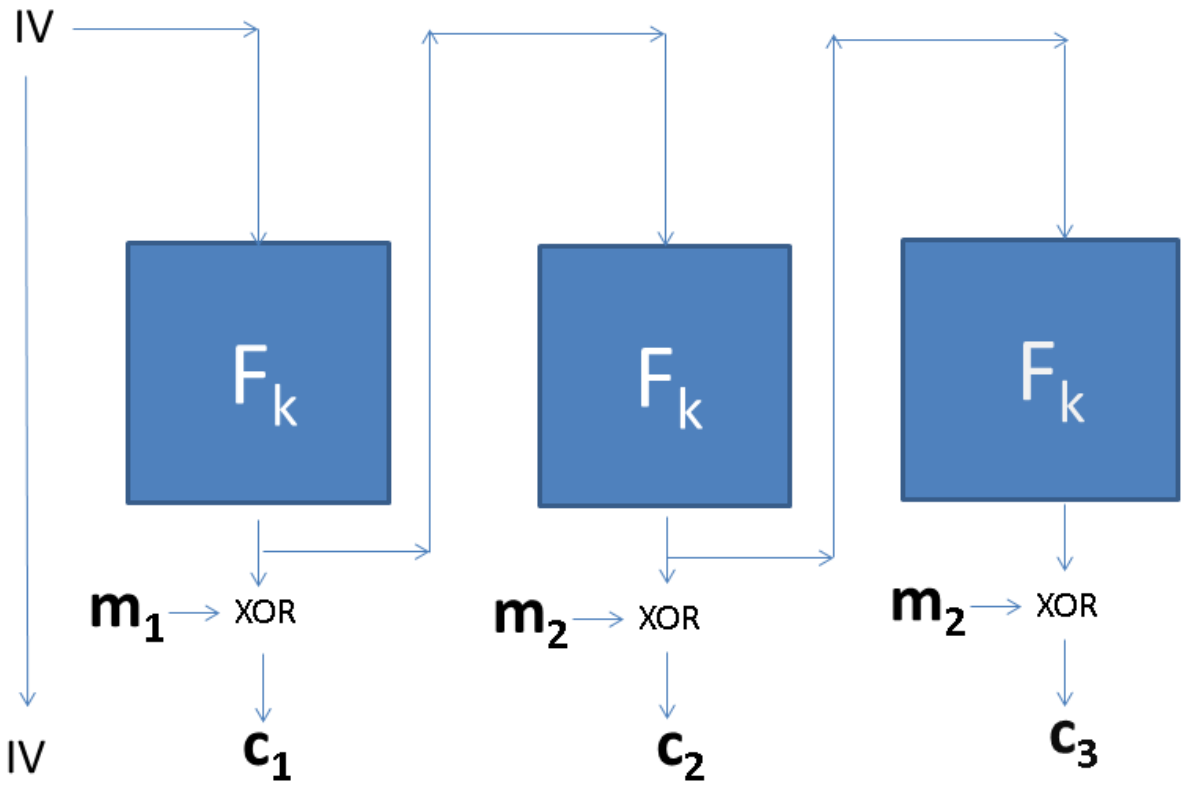
- Dec: On inputs $c = c_1 c_2 \cdots c_\ell$ and a block cipher of block length n, i.e. F_k^{-1}, output

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

CBC mode

- CBC is probabilistic.
- If F is a pseudo-random permutation, than CBC-mode encryption is CPA-secure.
- *Stateful* variant of CBC is when the last block of 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
- Chained CBC is not CPA-secure! (why?)
- Efficiency: is parallel processing possible?

Output Feedback (OFB) mode

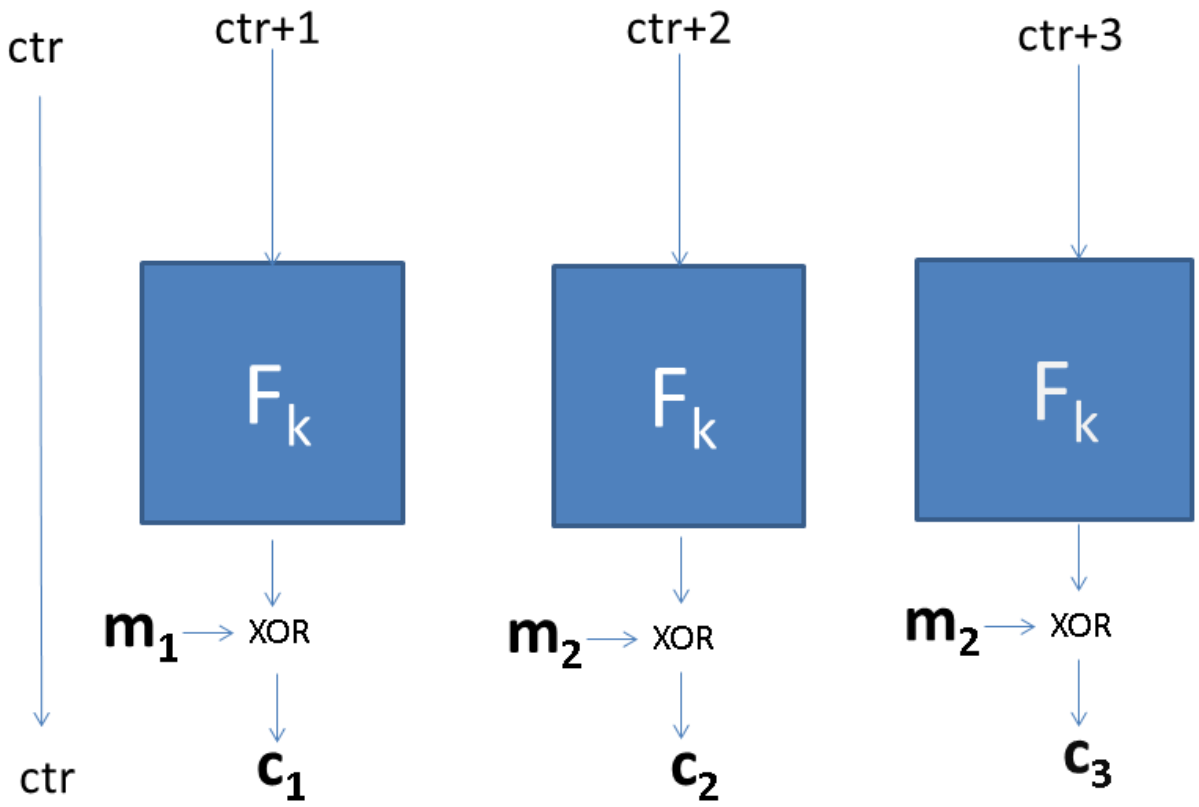


OFB

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: $c_i \leftarrow y_i \oplus m_i$.
- Dec: $m_i \leftarrow y_i \oplus c_i$.
- F doesn't have to be invertible.
- if F is a pseudo-random function, then the OFB mode is CPA-secure.
- Parallel processing is NOT possible.
- It can be viewed as an unsynchronised stream-cipher.
- Its stateful variant is equivalent to a synchronized stream cipher and is secure.

Counter (CTR) mode



- 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 doesn't have to be invertible.
- if F is a pseudo-random function, then the CTR mode is CPA-secure.
- Parallel processing is possible.
- It can be viewed as an unsynchronised stream-cipher.
- The stateful version of CTR mode is secure.

CTR mode

Theorem

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

Proof.

Similar to the previous proof, we can get:

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

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

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 (e.g. a block cipher called DES)
- The block length for DES is $\ell = 64$, then after you encrypt data of size $2^{32} \approx 34$ gigabytes, you can expect a repeated IV! (hint: see birthday paradox- we will cover it soon)
- In practice, if you have a repeated IV, then CBC is better that OFB and CTR. (Why?)

Initialisation Vector IV

- if IV repeats with OFB or CTR, the attacker can XOR the two resulting ciphertexts to learn about the encrypted plaintext, whereas in CBC mode, after few blocks the inputs to the block cipher 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.

Block Ciphers

- Block ciphers are expected to behave like random permutation.
- on ℓ-bit strings, we have $2^\ell!$ permutations.
- Problem: it is infeasible to represent permutation with big enough size for ℓ.
- For modern block ciphers, $\ell \geq 128$.
- NOTE: representing a permutation with ℓ-bit block size necessitates $\log(2^\ell!) \approx \ell \cdot 2^\ell$ bits.
- This is infeasible for $\ell > 50$.

Substitution-Permutation Networks (SPNs)

Confusion-Diffusion paradigm:

- Confusion: given a set of random permutations $\{f_i\}$ with small block length (e.g. 8 bits), construct a random-looking permutation F with a large block length (e.g. 128 bits).
- Now, given $x \in \{0,1\}^{128}$, parse as $x_1, \cdots, x_{16}$, where $|x_i| = 8$ bits. Define

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

- So far, regardless of the key k of F, if two inputs x and x' have only one different bit (say the 1st), then $F_k(x)$ and $F_k(x')$ have only one different byte.
- Diffusion solves this problem: we use a mixing permutation to make the aforementioned change in the the first bit affect the entire output block instead of only affecting the first byte in it!

SPNs

- Each confusion/diffusion step is called a round.
- A substitution-permutation network is an implementation of the confusion-diffusion paradigm.
- Using a fixed public algorithm, we derive a *key schedule* from a master key.

Key-Schedule: simple example

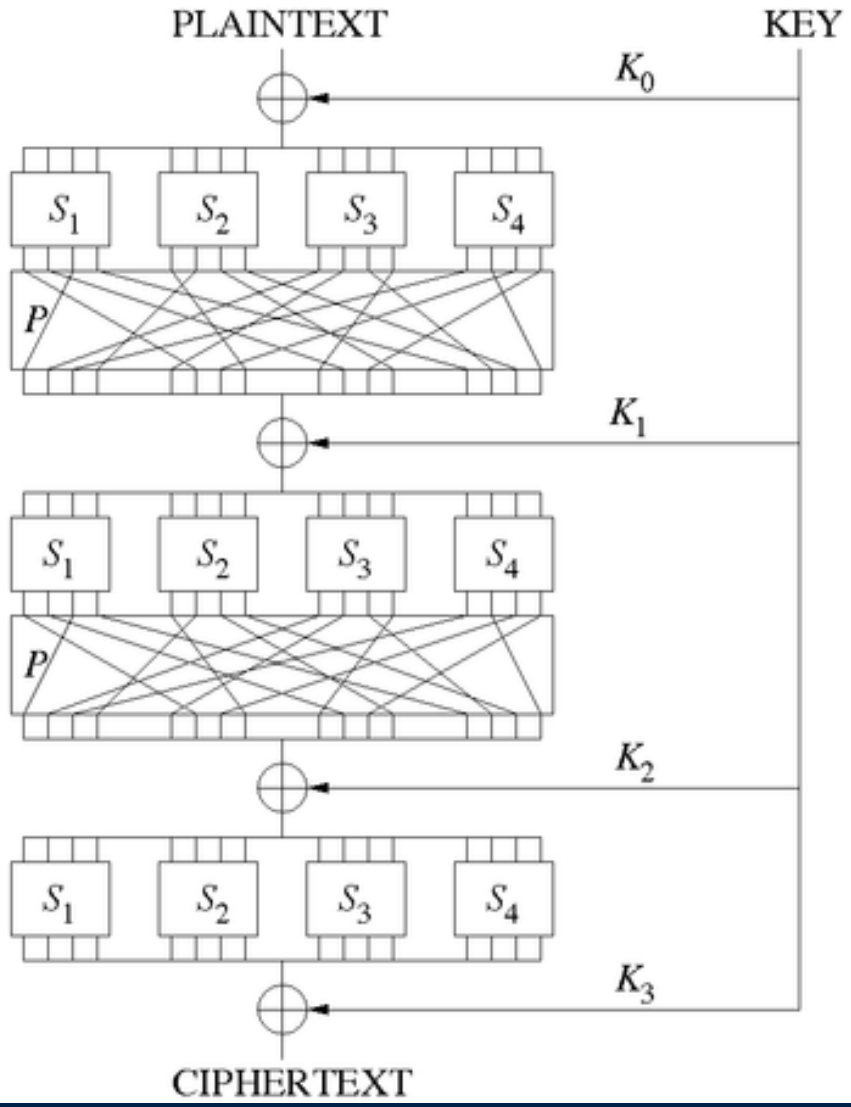
Suppose that the master key is as follows:

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

Our simple key schedule works as follows, we let k_i be 16 consecutive bits of KEY starting at bit $4i - 3$ as follows:

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

SPN


PLAINTEXT
KEY
K_0
S_1 S_2 S_3 S_4
P
K_1
S_1 S_2 S_3 S_4
P
K_2
S_1 S_2 S_3 S_4
K_3
CIPHERTEXT

SPN

Algorithm

Inputs: plaintext block, S-box , P-box , $(k_1, \cdots, k_{Nr+1})$
Output: ciphertext block

state = plaintext block
For round $r = 1$ to $Nr - 1$ do
key-mixing: state = state $\oplus k_r$
substitution: apply S-box to m strings of ℓ bits of state
permutation: apply P-box to ℓm bits of state
end do
x-or: state = state $\oplus k_{Nr}$
substitutions: apply S-box to m strings of ℓ bits of state
ciphertext block = state $\oplus k_{Nr+1}$

SPN- example

Avalanche effect

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

SPN

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

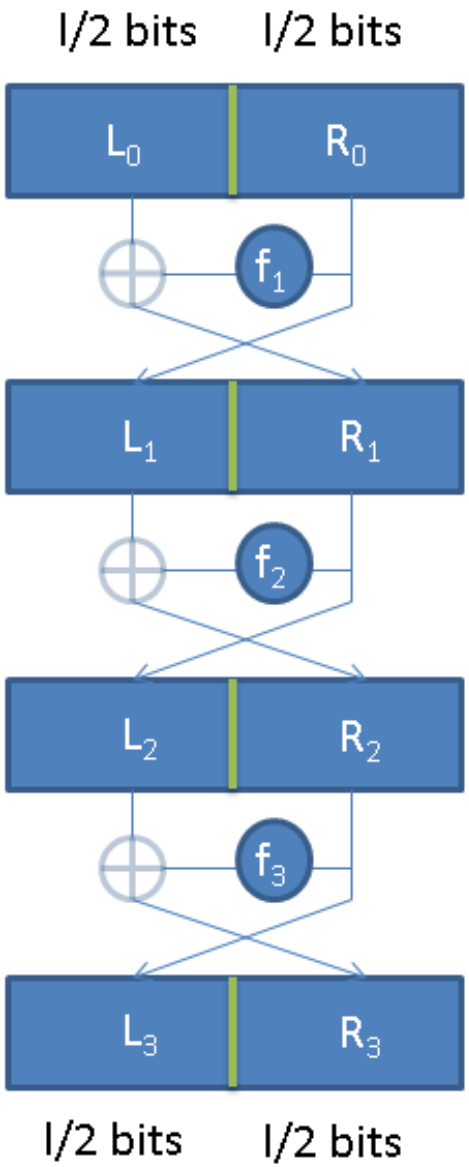
Feistel Networks

- Different approach to construct block ciphers.
- Advantage over SPN: the underlying function need not be invertible like S-boxes used in SPN.
- Feistel Network: Given functions $f_1, \cdots, f_d$, where $f_i : \{0,1\}^{\ell/2} \to \{0,1\}^{\ell/2}$, construct an **invertible** function $F : \{0,1\}^{\ell} \to \{0,1\}^{\ell}$.

Encryption:

Li=Ri-1

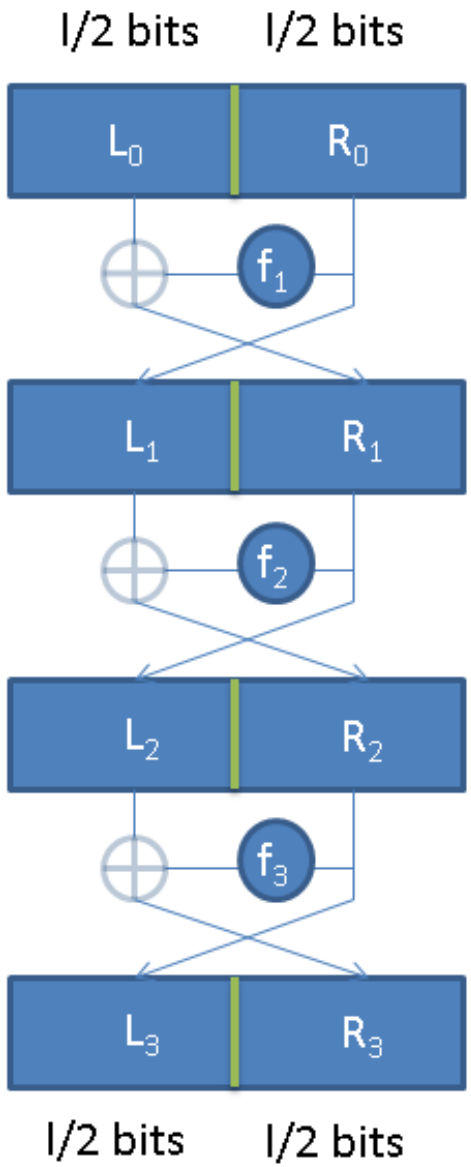
Ri=Li-1 $\oplus$ fi(Ri-1)



Encryption:

Li=Ri-1

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



Decryption:

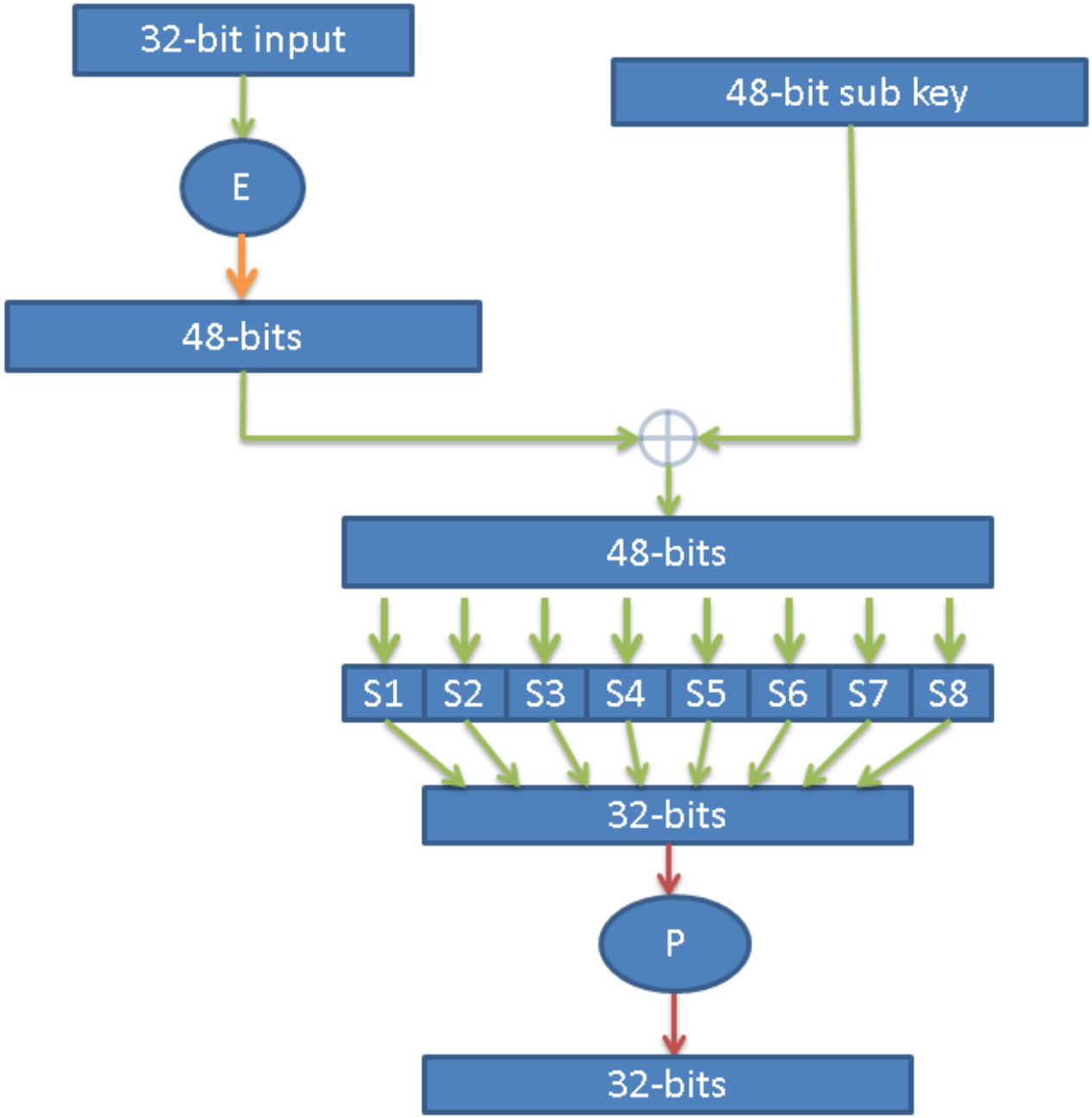
Ri-1=Li

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

Outline

The Data Encryption Standard (DES)

- DES is a 16 round Feistel network.
- Block length $\ell = 64$ and a master key length 56 bits.
- The same function f is used in all the 16 rounds.
- The **public** key schedule of DES is a 16 sub-keys of size 48 bits, i.e. $k_1, \cdots, k_{16}$ all derived from the **secret** master key.
- $f : \{0,1\}^{32} \times \{0,1\}^{48} \rightarrow \{0,1\}^{32}$
- It uses an *expansion function* E, $E : \{0,1\}^{32} \rightarrow \{0,1\}^{48}$. It simply duplicates 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. replace words by 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

- In 1970, Horst Feistel designs Lucifer at IBM, where |key| = 128 bits and |block|= 128 bits
- 1976: NBS adopts DES as a federal standard |key| = 56 bits and |block|= 64 bits
- 1997: DES broken by exhaustive search (DESCHALL project, using 96 days)
- State-of-the-art: can find a DES key in $\approx$ 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) and requires 2^{47} **chosen** plaintexts to work.
- 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** plaintext, which is still a big number, but has the advantage of being known rather than chosen

2DES

- DES's main problem was its short key size, therefore changing the internal structure of DES was not recommended.
- What if we double the encryption, i.e.

$$F'_{k_1,k_2} \leftarrow F_{k_2}(F_{k_1}(x))$$

- Not a great idea! A meet-in-the-middle attack takes time $\mathcal{O}(n \cdot 2^n)$ even if both keys are in $\{0,1\}^n$, 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 minimize the set of possible 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 will then create a third list M that contains 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

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

- Meet-in-the-middle attack takes 2^{2n}.
- The second variant uses two keys $k_1, k_2 \in \{0,1\}^n$ s.t.

$$F''_{k_1,k_2} \leftarrow F_{k_1}(F_{k_2}^{-1}(F_{k_1}(x)))$$

- 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 slow (it requires three block cipher operations!)
- 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 for AES can be of 128, 192, or 256-bit length.
- 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 cipher.

AES: the four stages

- **AddRoundKey**: Xor the state with a 128-bit sub-key, that is generated using the master.
- **SubBytes**: Apply a fixed S-box to each byte of the state. The S-box is represented by a lookup table which is a bijection over $\{0,1\}^8$.
- **ShiftRows**: You shift the bytes in each row of the state to the left and in a cyclic way starting from the first 0, 1,2 and 3 respectively.
- **MixColumns**: Apply a linear transformation which is in fact 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 exhaustive 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 $\{X_1, \cdots, X_{\ell_1}\}$ and outputs $\{Y_1, \cdots, Y_{\ell_2}\}$, compute

$$L = \bigoplus_{i=1}^{\ell_1} X_i \bigoplus_{j=1}^{\ell_2} Y_i$$

- 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 outputs (Y_1, Y_2), exploit the relationship between ΔX and ΔY, where $\Delta X = X_1 \oplus X_2$ 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 where 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 paintexts $\{X_{i_1}, X_{i_2}\}$ for which he knows that a certain ΔY_i occurs with high probability.

Quantum attacks on Block Ciphers

- Generically, a search problem can be defined as follows: Let $f : X \to \{0, 1\}$ be a function. Find $x \in X$ s.t. $f(x) = 1$.
- On a classical computer, the best algorithm is a generic algorithm which runs in time $\mathcal{O}(|X|)$.
- On a quantum computer (when they exist?), according to [Grover'96], the runnning time is $\mathcal{O}\left(\sqrt{|X|}\right)$ (quadratic speedup).
- Given m and $c = \mathrm{Enc}(k, m)$, define $f(k) = 1$ if $\mathrm{Enc}(k, m) = c$ 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.

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.