ENEE 457: Computer Systems Security
09/05/18

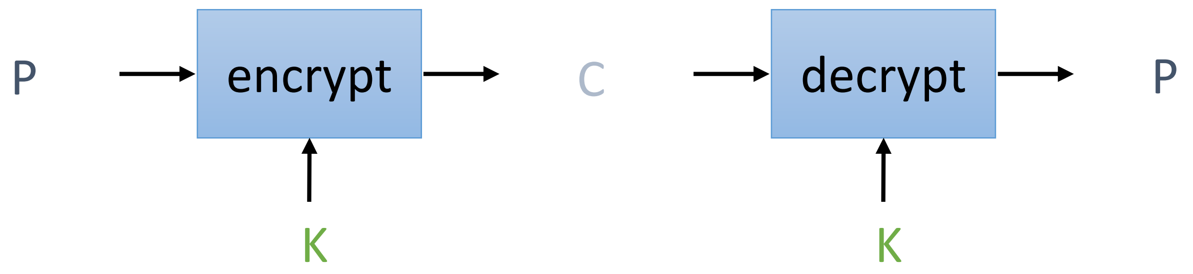
Lecture 3
Symmetric Crypto I

Charalampos (Babis) Papamanthou

Department of Electrical and Computer Engineering
University of Maryland, College Park

Symmetric Cryptosystem

- Scenario
  - Alice wants to send a message (plaintext P) to Bob
  - The communication channel is insecure and can be eavesdropped
  - If Alice and Bob have previously agreed on a symmetric encryption scheme and a secret key K, the message can be sent encrypted (ciphertext C)
- Issues
  - What is a good symmetric encryption scheme?
  - What is the complexity of encrypting/decrypting?
  - What is the size of the ciphertext, relative to the plaintext?



Basic Notions

- Notation
  - Secret key K
  - Encryption function $E_K(P)$
  - Decryption function $D_K(C)$
  - Plaintext length typically the same as ciphertext length
  - Encryption and decryption are permutation functions (bijections) on the set of all n-bit arrays
- Efficiency
  - functions E_K and D_K should have efficient algorithms
- Consistency
  - Decrypting the ciphertext yields the plaintext
  - $D_K(E_K(P)) = P$

Attack on all schemes: Brute-Force Attack

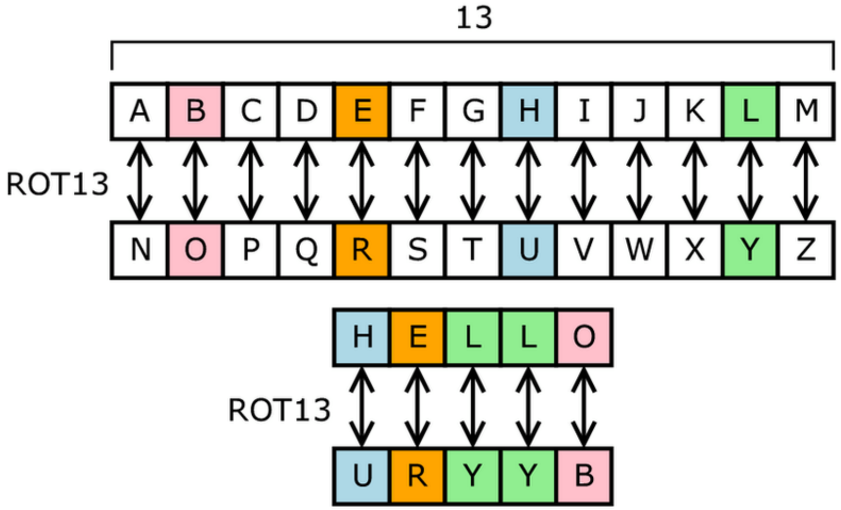
- Try all possible keys K and determine if $D_K(C)$ is a likely plaintext
  - Requires some knowledge of the structure of the plaintext (e.g., PDF file or email message)
- Key should be a sufficiently long random value to make exhaustive search attacks unfeasible

Image by Michael Cote from http://commons.wikimedia.org/wiki/File:Bingo_cards.jpg

Candidate scheme: Substitution Ciphers

- Each letter is uniquely replaced by another
- There are 26! possible substitution ciphers

- One popular substitution “cipher” for some Internet posts is ROT13



Public domain image from http://en.wikipedia.org/wiki/File:ROT13.png

Or…Substitution Boxes

- Substitution can also be done on binary numbers.
- Such substitutions are usually described by substitution boxes, or S-boxes.

	00	01	10	11
00	0011	0100	1111	0001
01	1010	0110	0101	1011
10	1110	1101	0100	0010
11	0111	0000	1001	1100

(a)

	0	1	2	3
0	3	8	15	1
1	10	6	5	11
2	14	13	4	2
3	7	0	9	12

(b)

Figure 8.3: A 4-bit S-box (a) An S-box in binary. (b) The same S-box in decimal.

Attack on Substitution ciphers: Frequency Analysis

- Letters in a natural language, like English, are not uniformly distributed
- Knowledge of letter frequencies, including pairs and triples can be used in cryptologic attacks against substitution ciphers

a:	8.05%	b:	1.67%	c:	2.23%	d:	5.10%
e:	12.22%	f:	2.14%	g:	2.30%	h:	6.62%
i:	6.28%	j:	0.19%	k:	0.95%	l:	4.08%
m:	2.33%	n:	6.95%	o:	7.63%	p:	1.66%
q:	0.06%	r:	5.29%	s:	6.02%	t:	9.67%
u:	2.92%	v:	0.82%	w:	2.60%	x:	0.11%
y:	2.04%	z:	0.06%				

Letter frequencies in the book *The Adventures of Tom Sawyer*, by Twain.

What would a great symmetric encryption scheme satisfy?

- What if we could devise a system such that we can encrypt and the ciphertext does not reveal anything about the plaintext (apart from its length)
- Let’s express it mathematically

Perfect security

- Pick messages m_1 and m_2
- Pick a ciphertext c
- Encrypt m_1
- Encrypt m_2
- Compute the probability $Pr[Enc(m_1)=c]$ (over the choice of the random key)
- Compute the probability $Pr[Enc(m_2)=c]$ (over the choice of the random key)
- Enc is secure if for all messages m_1 and m_2 and for all ciphertexts c
    - $Pr[Enc(m_1)=c]= Pr[Enc(m_2)=c]$

One-time pad

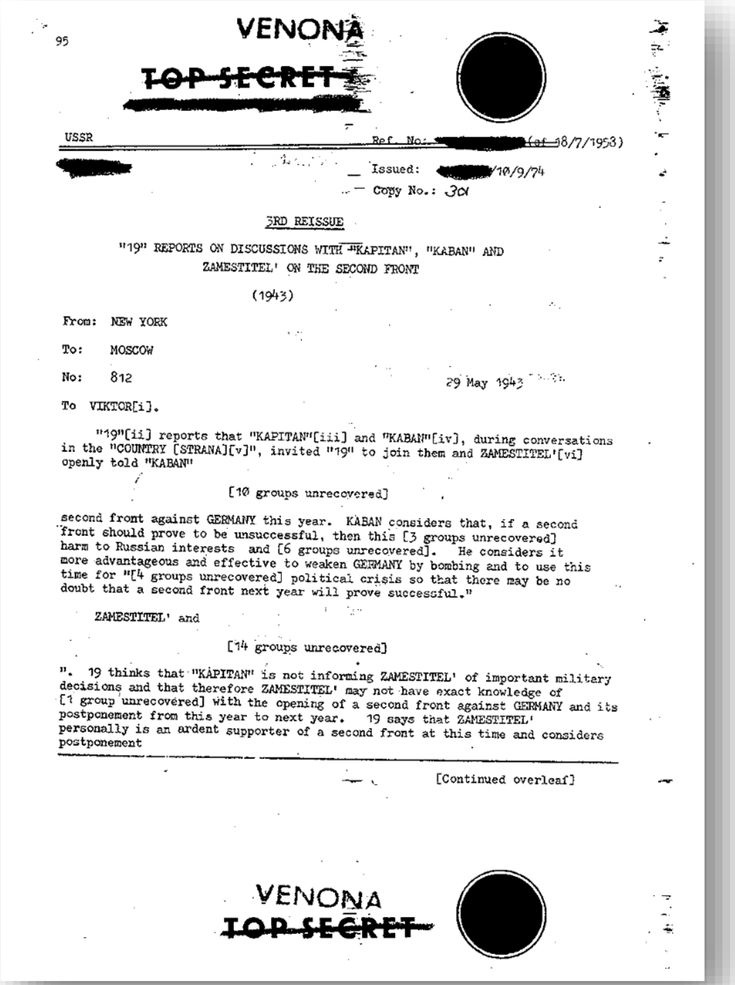
- $K \leftarrow$ KeyGen(n): Pick a random key K of n bits
- $E_K(A)$: On input plaintext A, compute ciphertext B=A XOR K
- $D_K(B)$: On input ciphertext B, compute plaintext A=B XOR K
- Correctness: B XOR K= (A XOR K) XOR K= A XOR 0 = A
- Security?
  - Note that $Enc_K(m_1)=c$ is the event m_1 XOR K = c which is the event K = m_1 XOR c
  - K is chosen at random (irrespective of m_1 and m_2, and therefore the probability is 2^{-n}
  - Namely ciphertext does not reveal anything about the plaintext

Key space and message space in one-time pad

- Key space should be at least equal to the message space
- Suppose not and the key space is missing one element and does not contain 0000000…00
- For a given c, there exists a message m such that
  - Pr[Enc(m) = c]=0
  - E.g., If key does not contain 00000000000…00, then m = c
  - But for all other messages m' that are not equal to c we have that
    - Pr[Enc(m') = c]>0=1/(2^{n}-1) (why is that?)
  - Therefore the definition does not hold.
  - In particular, if I see a ciphertext, I have excluded one possibility

One-time pad is not practical

- In spite of their perfect security, one-time pads have some weaknesses
- The key has to be as long as the plaintext
- Keys can never be reused
  - Repeated use of one-time pads compromised communications during the cold war


95

VENONA

~~TOP SECRET~~ [redacted]

USSR [redacted]

Ref. No: [redacted] (of 18/7/1953)

Issued: [redacted] /10/9/74

Copy No.: 301

3RD REISSUE

"19" REPORTS ON DISCUSSIONS WITH "KAPITAN", "KABAN" AND ZAMESTITEL' ON THE SECOND FRONT

(1943)

From: NEW YORK

To: MOSCOW

No: 812 29 May 1943

To VIKTOR[i].

"19"[ii] reports that "KAPITAN"[iii] and "KABAN"[iv], during conversations in the "COUNTRY [STRANA][v]", invited "19" to join them and ZAMESTITEL'[vi] openly told "KABAN"

[10 groups unrecovered]

second front against GERMANY this year. KABAN considers that, if a second front should prove to be unsuccessful, then this [3 groups unrecovered] harm to Russian interests and [6 groups unrecovered]. He considers it more advantageous and effective to weaken GERMANY by bombing and to use this time for "[4 groups unrecovered] political crisis so that there may be no doubt that a second front next year will prove successful."

ZAMESTITEL' and

[14 groups unrecovered]

". 19 thinks that "KAPITAN" is not informing ZAMESTITEL' of important military decisions and that therefore ZAMESTITEL' may not have exact knowledge of [1 group unrecovered] with the opening of a second front against GERMANY and its postponement from this year to next year. 19 says that ZAMESTITEL' personally is an ardent supporter of a second front at this time and considers postponement

[Continued overleaf]

VENONA

~~TOP SECRET~~

What do we want to use in practice?

- Size: Small, one-time key (128 bits) and also encrypting the same thing twice should give different things
- Security: It turns out that **perfect secrecy** is very strong if we want to achieve both small key and one key
  - How about if we improve the best strategy of the attacker, which is still going to be really bad for practical purposes
- Answer: **Computational Secrecy**
- **Intution: The ciphertext does not reveal anything about the plaintext as long as our attacker runs in time polynomial (like all machines in this class)**
- If attacker can run in time 2^{size_of_key}, all bets are off
- But this is too long…

Pseudorandom permutations (PRPs)

- We say that a length-preserving keyed function F: $\{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$, is a keyed permutation if and only if each F_k is a bijection
- Also, it is pseudorandom if an adversary could not distinguish between the following two worlds with probability more than ½+2^{-k}
  - He sends *x* to World1, World1 chooses a random permutation A and returns A[x]
  - He sends *x* to World2, World2 chooses a random key k and returns $F_k(x)$
- How do we encrypt using PRPs a message m of n bits?
- First attempt: Pick secret key k. Return F_k(m). Problem?
  - $\mathbf{Enc}_k(m)$: $c := \langle r, F_k(r) \oplus m \rangle$
    - where $r \leftarrow \{0,1\}^n$ is chosen at uniform random
  - $\mathbf{Dec}_k(c)$: given $c=\langle r, s \rangle$, $m := F_k(r) \oplus s$
- Let's call the above scheme **`First_Symmetric`**

Question 2

- Why `First_Symmetric` is secure?

Intuitively this is secure: so long as r is not used for different messages, $F_k(r)$ should look completely random

- But this is just intuition

Semantic security (CPA)

- I give you a symmetric encryption scheme (Enc,Dec,K)
- What do you need to prove in order to say that it is secure?
- A strong notion used is “semantic security”
- We are going to define it as an interaction between the adversary **A** and a trusted party **T** that has the secret key.
- Informally:
    1. **T** picks a random secret key
    2. **A** picks messages m_i and receives ciphertexts Enc_K(m_i) from **T**.
    3. **A** picks message m_0 and m_1 and sends them to **T**.
    4. **T** flips a coin b and computes t_b=Enc_K(m_b).
    5. **T** sends t_b to the **A**.
- The scheme is secure if **A** has no better chance of finding whether t_b corresponds to m_0 or m_1 than ½+2^{-k}
- This should hold even if it is repeated many (polynomial) times

Question 3

- What behavior of the adversary does this definition model?

- Think emails…

Question 4

- Why **`First_Symmetric`** without randomness r is **not** semantically secure?

- Provide an attack where the adversary's chance of finding where t_b corresponds to is 1.

Task 1

- Prove **`First_Symmetric`** is semantically secure
  - Suppose it is not. That means that the adversary A, given
    - m_0 and m_1
    - c_b = $F_k(r) \oplus$ m_b (where b = 0 or b = 1)

  can figure out whether b = 0 or b = 1. But due to the "randomness" of $F_k(r)$, $F_k(r)$ appears "random", so $F_k(r) \oplus$ m_b appears "random" " and does not give any information about m_b, a contradiction.

More advanced security (CCA)

- Informally:
  - **T** picks a random secret key
  - **A** picks messages m_i and receives ciphertexts Enc_K(m_i) from **T**.
  - **A** picks message m_0 and m_1 and sends them to **T**.
  - **T** flips a coin b and computes t_b=Enc_K(m_b).
  - **T** sends t_b to the **A**.
  - **A** sends a ciphertext of its choice**, different than t_b**, for decryption
  - The scheme is secure if **A** has no better chance of finding whether t_b corresponds to m_0 or m_1 than ½+2^{-k}
- This should hold even if it is repeated many (polynomial) times

Question 5

- What behavior of the attacker does this model?

- Lunch-time attacks…

Is **`First_Symmetric`** CCA-secure?

- Ask encryption for m_0 = 0000…00 and m_1 = 1111…11
- You get $c_b = < s_b, r_b >$, where $s_b = F_k(r_b) \oplus m_b$
- How to find b is you are allowed to send decryption queries?
- Construct new new ciphertext
    - $c = < s_b \oplus 1000\ldots00, r_b > = < F_k(r_b) \oplus m_b \oplus 1000\ldots00, r_b >$
    - Decryption of this will give $m_b \oplus$ 1000…00
        - 1000…00, if s_b was encryption of m_0 = 0000…00
        - 01111…1, if s_b was encryption of m_1 = 1111111....1111
- So we can distinguish!
- Conclusion: **`First_Symmetric`** is not CCA-secure.

How do we construct a PRP in practice?

- What is the main property we want?
  - Even a single bit change in the input should yield a completely independent result
- This implies that
  - Every bit of the input should affect every bit of the output…
  - Or…every change in an input bit should change each output bit with probability roughly ½
- This takes some work…

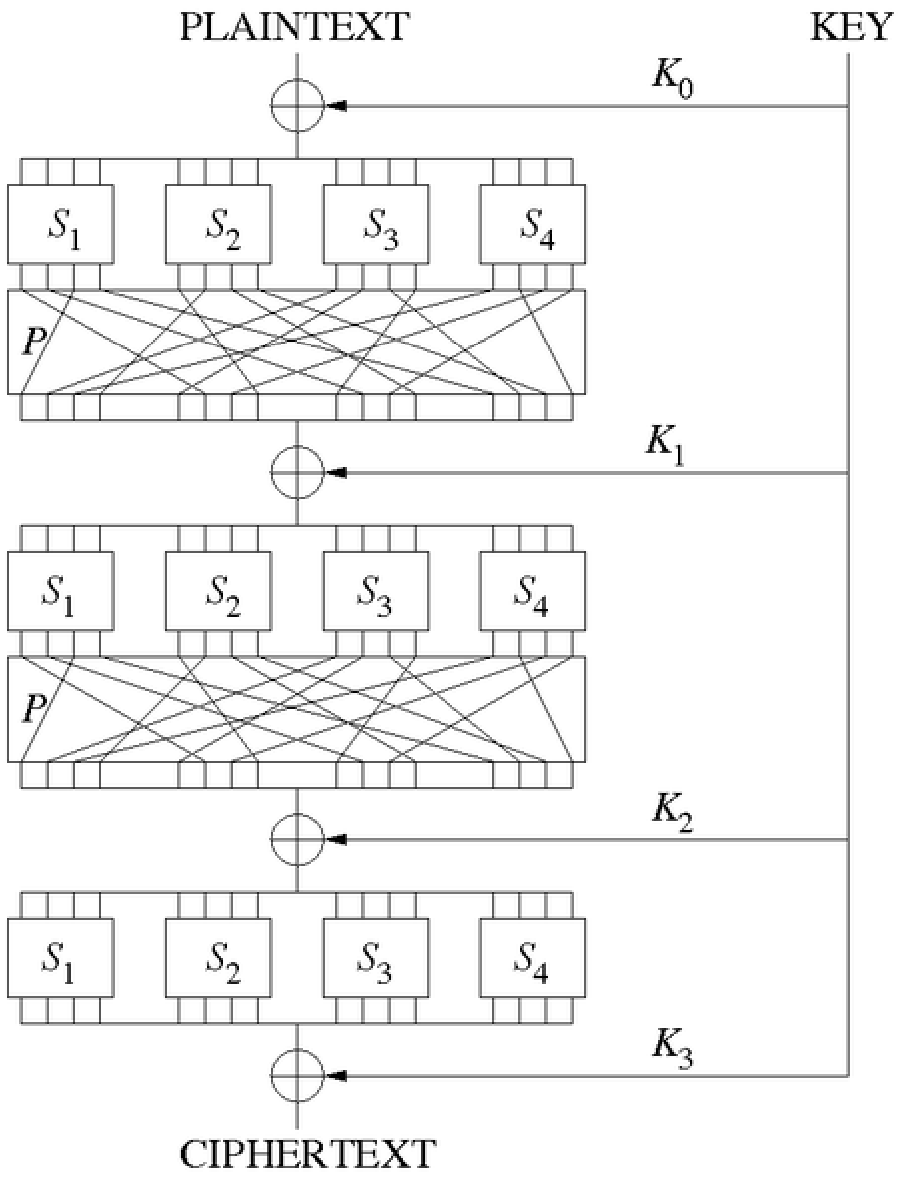
A first idea (Shannon)

- Construct block cipher from many smaller random (or random-looking) permutations
- **Confusion:** e.g., for block size 128, uses 16 8-bit random permutation
  - $F_k(x) = f_1(x_1) \ldots f_{16}(x_{16})$
  - Where key k selects 16 8-bit random permutation.
  - Does $F_k(\cdot)$ look like a random permutation?
- **Diffusion:** bits of $F_k(x)$ are permuted (re-ordered)
- Multiple rounds of confusion and diffusion are used.

Substitution-Permutation Networks

- A variant of the Confusion-Diffusion Paradigm
  - $\{f_i\}$ are fixed and are called s-boxes
  - Sub-keys are XORed with intermediate result
    - Sub-keys are generated from the master key according to a key schedule
- Each round has three steps
  - Message XORed with sub-key
  - Message divided and went through s-boxes
  - Message goes through a mixing permutation (bits reordered)

Substitution-Permutation Networks



Design Principles:

---A single-bit difference in each s-box results in changes in at least two bits in output

---The mixing permutation distributes the output bits of any s-box into multiple s-boxes

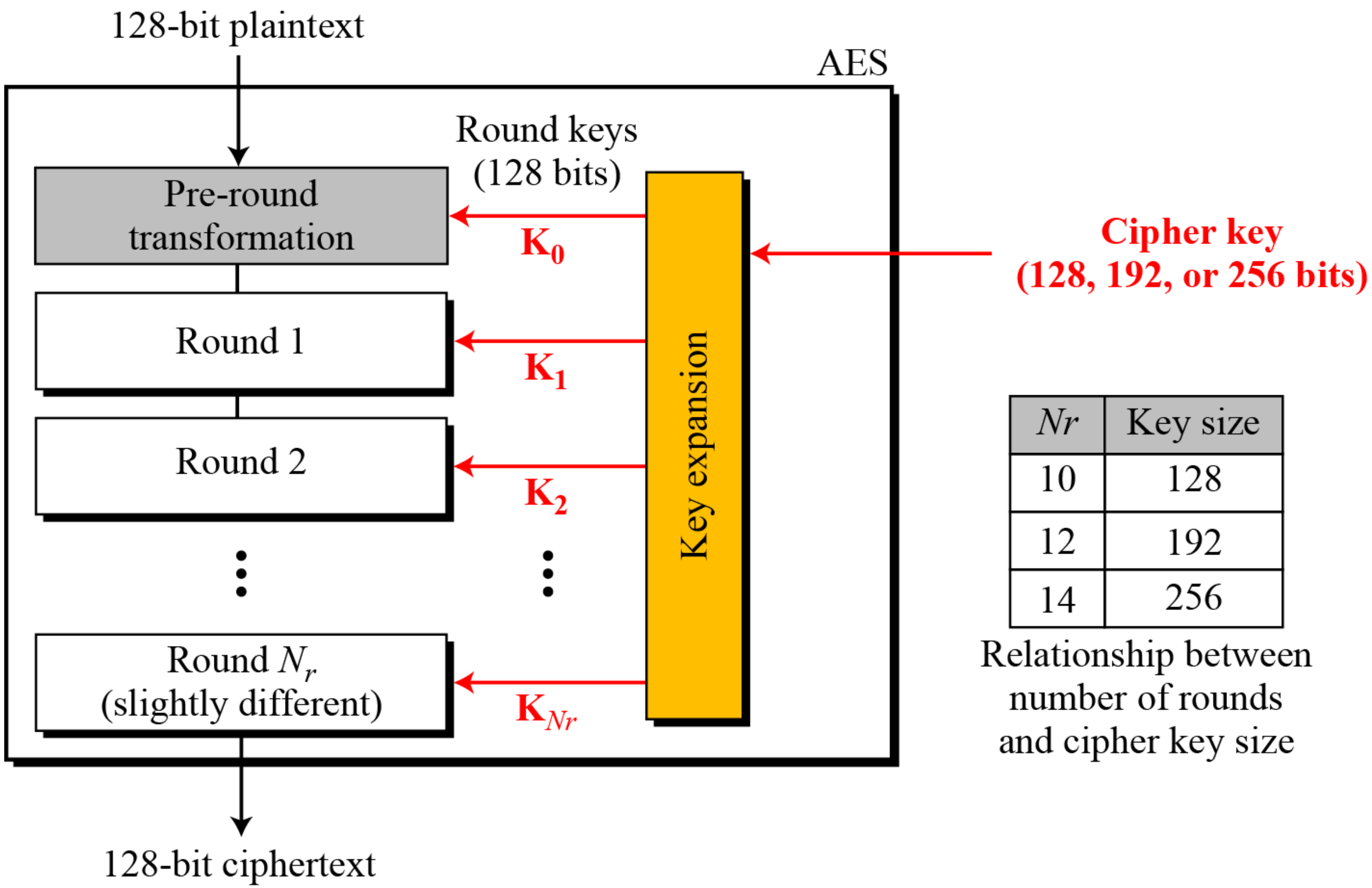
The above, with sufficient number of rounds, achieves the avalanche effect.

AES encryption, the algorithm of choice in today's Internet communications is using the above framework

Question 6

- Say you have the pair of ciphertext and plaintext.
- How can you attack one round?
- How can you attack two rounds?

AES structure

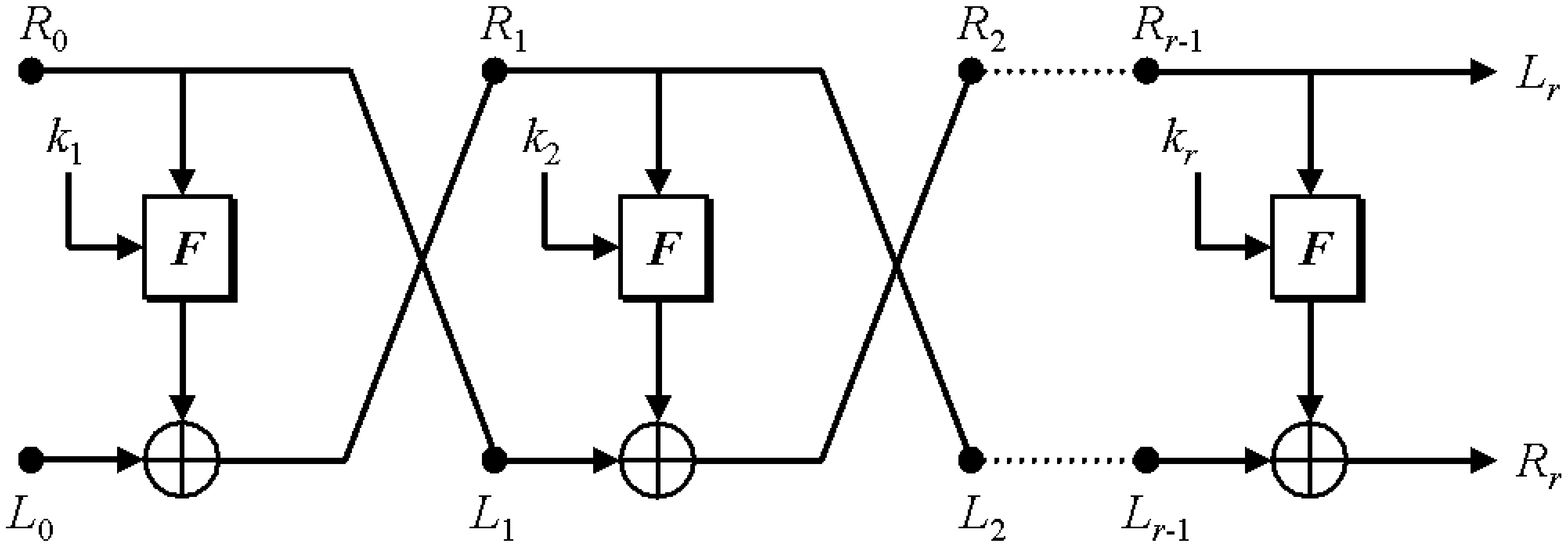


Nr	Key size
10	128
12	192
14	256

Relationship between number of rounds and cipher key size

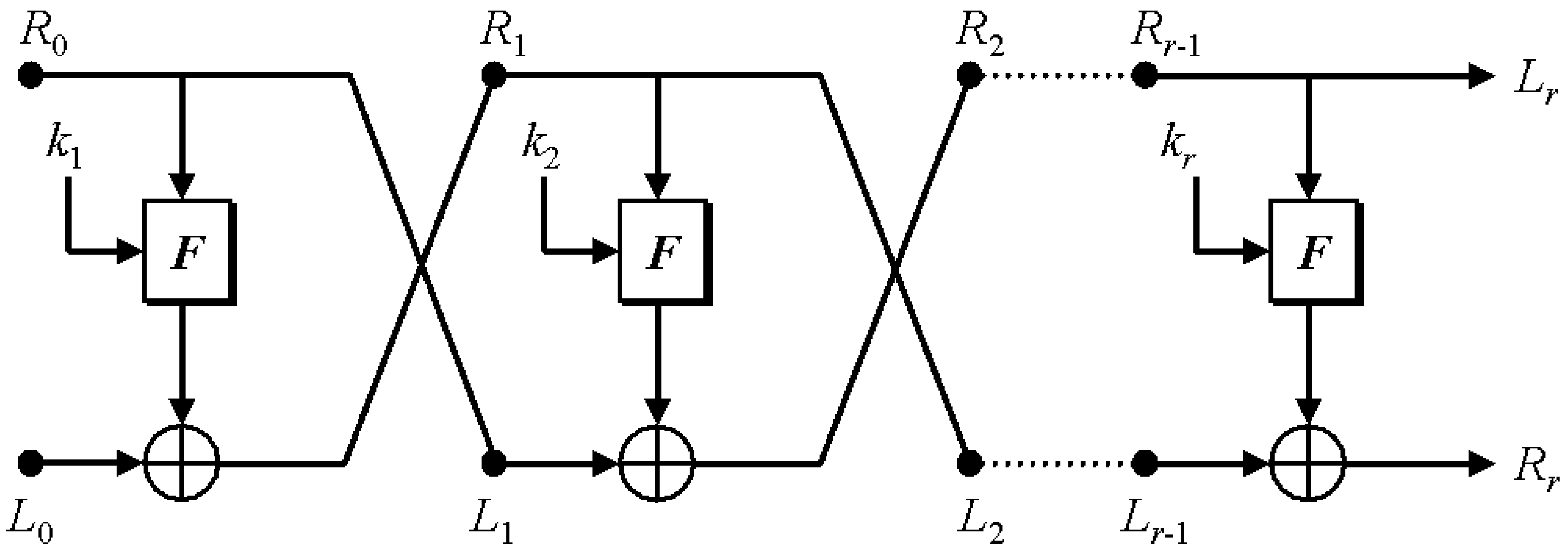
Second approach: Feistel Network

- Feistel Networks



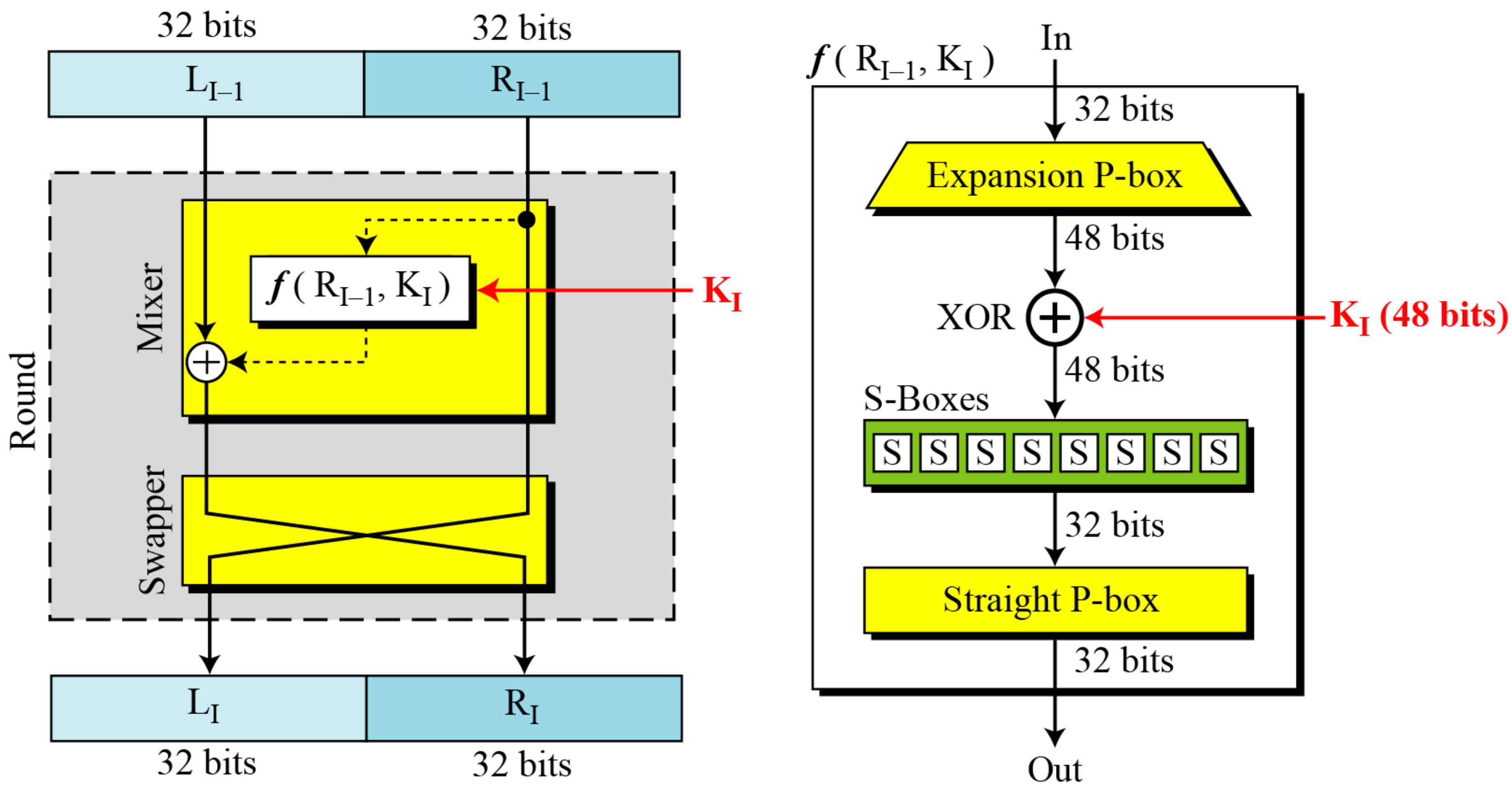
Feistel Network

- Main difference: F does not have to be invertible
- In practice: It is a Substitution-permutation network
- DES was based on that (broken, not because of bad design, but due to the size of the key)



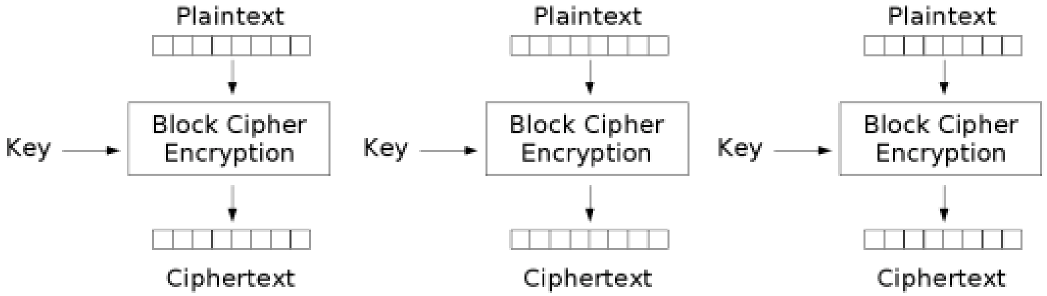
DES function

The DES function applies a 48-bit key to the rightmost 32 bits to produce a 32-bit output



Block Cipher Modes

- So far we have described how to encrypt a string of fixed length
- How do we encrypt a 4GB file?
- Electronic Code Book (ECB) Mode (is the simplest):
  - Block P[i] encrypted into ciphertext block C[i] = E_K(P[i])
  - Block C[i] decrypted into plaintext block M[i] = D_K(C[i])



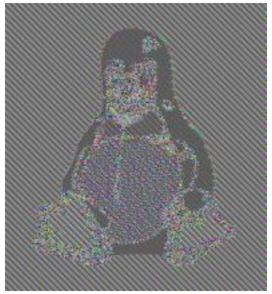
Electronic Codebook (ECB) mode encryption

Strengths and Weaknesses of ECB

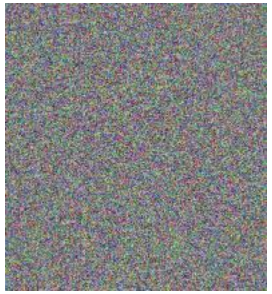
- Strengths:
  - Is very simple
  - Allows for parallel encryptions of the blocks of a plaintext
  - Can tolerate the loss or damage of a block

- Weakness:
  - Documents and images are not suitable for ECB encryption since patterns in the plaintext are repeated in the ciphertext:

ECB

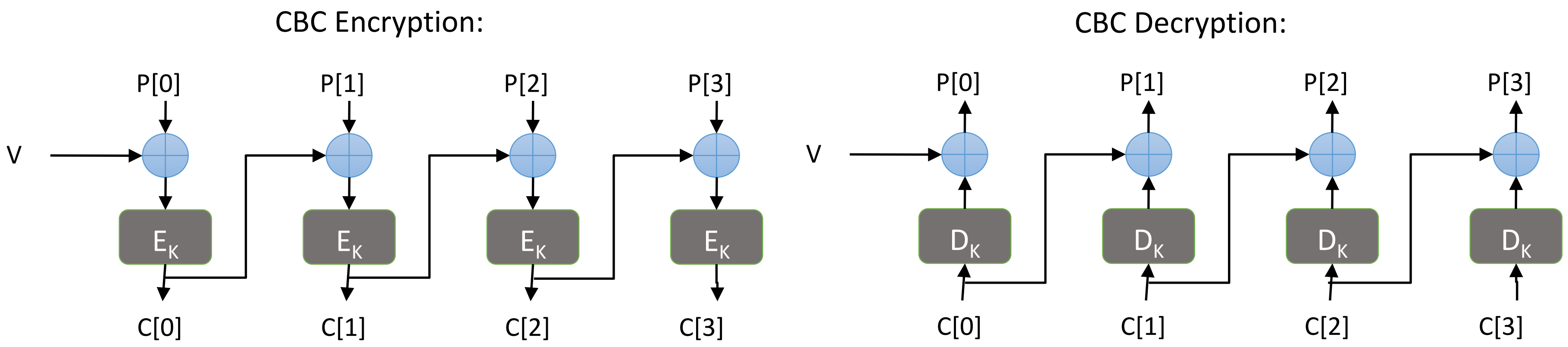


CBC



Cipher Block Chaining (CBC) Mode

- In Cipher Block Chaining (CBC) Mode
  - The previous ciphertext block is combined with the current plaintext block $C[i] = E_K(C[i-1] \oplus P[i])$
  - $C[-1] = V$, a random block separately transmitted encrypted (known as the initialization vector)
  - Decryption: $P[i] = C[i-1] \oplus D_K(C[i])$



Question 7

- Is CBC encryption parallelizable?
- Is CBC decryption parallelizable?

OpenSSL encryption decryption

- openssl aes-256-cbc -a -in plaintext.txt -out ciphertext.txt –base64

- openssl aes-256-cbc -a -d -in ciphertext.txt -out plaintext.txt