CS 88: Security and Privacy

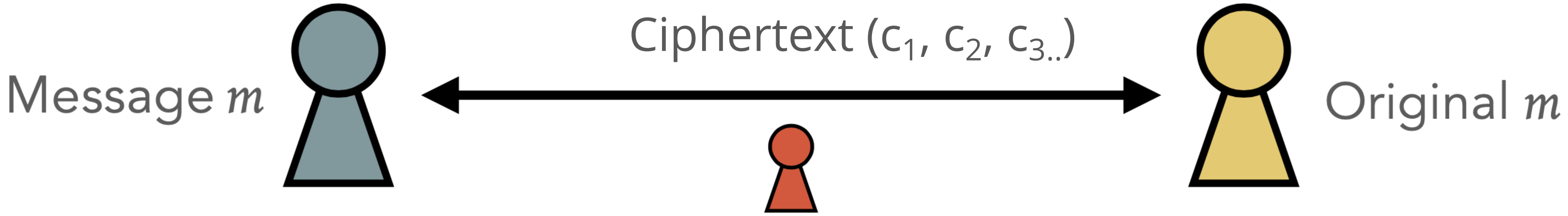
13: Symmetric Key Cryptography

10-20-2022

slides adapted from Dave Levine, Jonathan Katz, Kevin Du

SWARTHMORE COLLEGE

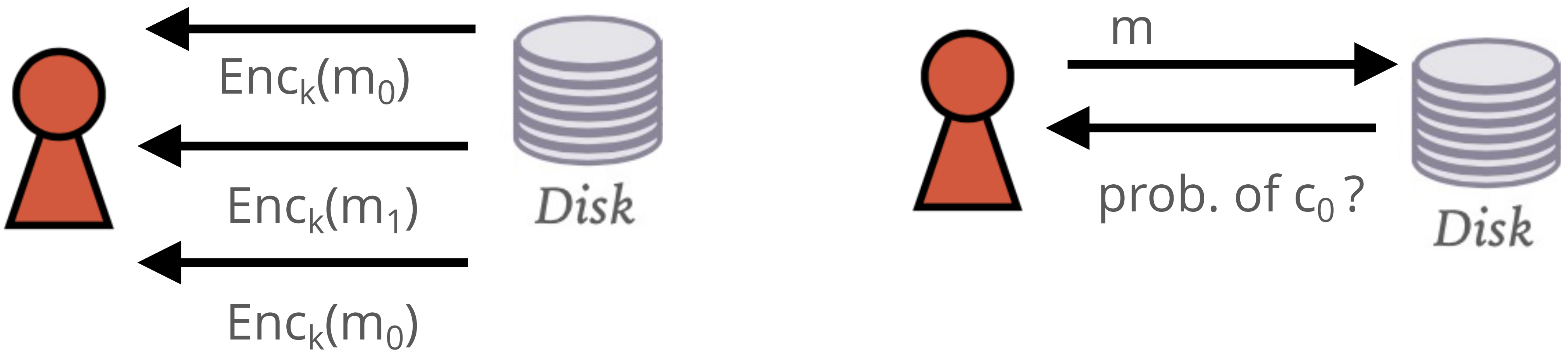
Multiple message secrecy



We are not going to formally define a notion of multiple-message secrecy

- Instead, define something stronger: <u>security against chosen-plaintext attacks (CPA-security)</u>
- *minimal notion of security an encryption scheme should satisfy*

Security against Chosen Plaintext Attack: Impossible?



It really is a problem if an attacker can tell when the same message is encrypted twice!

This attack only works if encryption is deterministic!

Random Functions

- Functions map from some set X to a set F(X) = Y.
  - (think of this mapping as a hash table mapping from x -> y)

- $Func_n$: all mappings from X: $\{0, 1\}^n$ -> F(X)= Y:$\{0, 1\}^n$
  - i.e., for all input bit strings of length n, there is a mapping to an output bit string also of length n
  - *all possible mappings? $2^{n.(2^n)}$!! astronomically large!*

Random Functions

Out of all possible functions between X and Y we choose one uniformly at random.

- e.g. for a 2 bit string mappings between X: $\{0, 1\}^2$ and Y: $\{0, 1\}^2$
- one possible mapping that we could choose:

x	00	01	10	11
F(x)	01	11	00	10

Properties of function F(X) chosen uniformly at random:

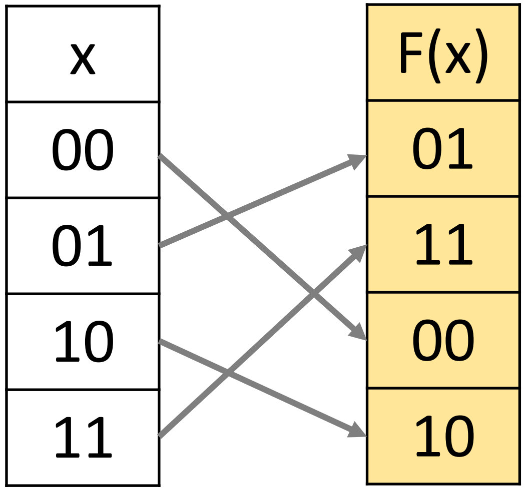
- for any given $x \in X$, the probability that $F(x) = y$ is $1/2^n$
- in our example example:
  - given $x \in X$, the probability that $F(x) = 1/2^2 = ¼ = 0.25$
- F(x) property:
  - if x changes by one bit to give x’ then
  - *F(x’) is completely independent of F(x).*

Random Permutations

- Variant of random function is random permutation
  - treat them equivalently for our purposes .

- E.g.: random permutation over bit strings of length 2

  Encryption: $\{0, 1\}^2 \rightarrow \{0, 1\}^2$



Important Property of the Random Permutation:

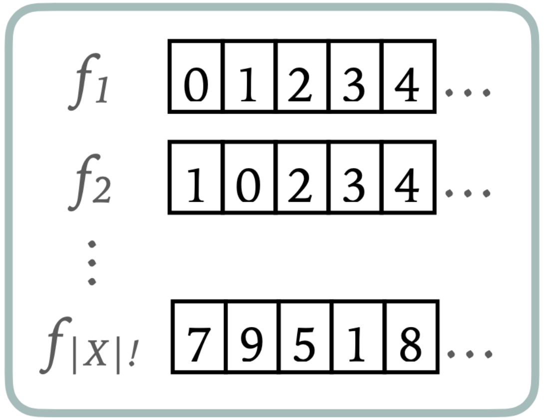
A permutation is invertible (bijective) function

Given F(x) it is impossible to determine x without resorting to a brute force attack.

If |X| is very large? brute force not possible by an efficient (probabilistic polynomial time) attacker.

What we have, ideally: Random Functions

Consider the set of all permutations $F_k: X \to X$



Think of X as all 128-bit bit strings

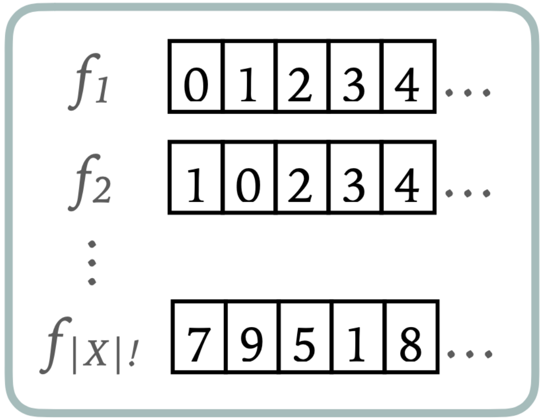
If you know k, then $F_k(x)$ is trivial to invert

If you don't know k, then $F_k(x)$ is one-way

<u>One-way trapdoor function</u>

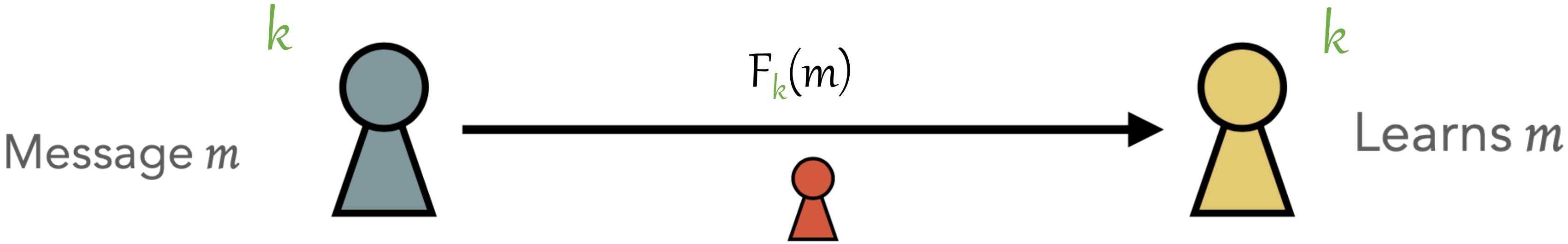
What we have, ideally: Random Functions

Consider the set of all permutations $F_k: X \to X$



Think of X as all 128-bit bit strings

Shared secret: index k chosen uniformly, at random

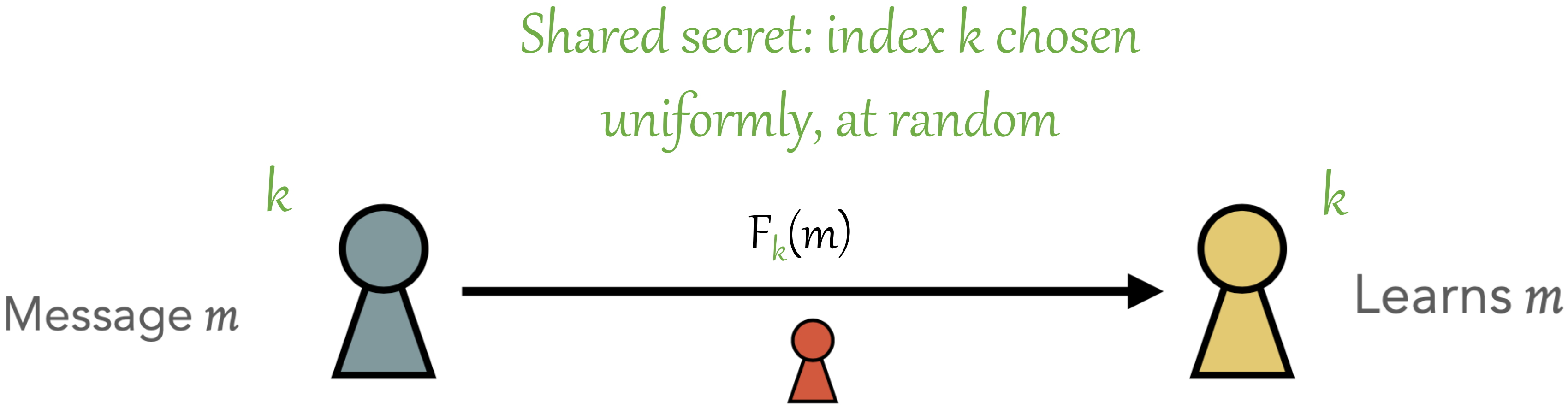


Without knowing *k*, Eve learns nothing about *m*

k is our key!

What we have, ideally: Random Functions

k is our key!



Without knowing *k*, Eve learns nothing about *m*

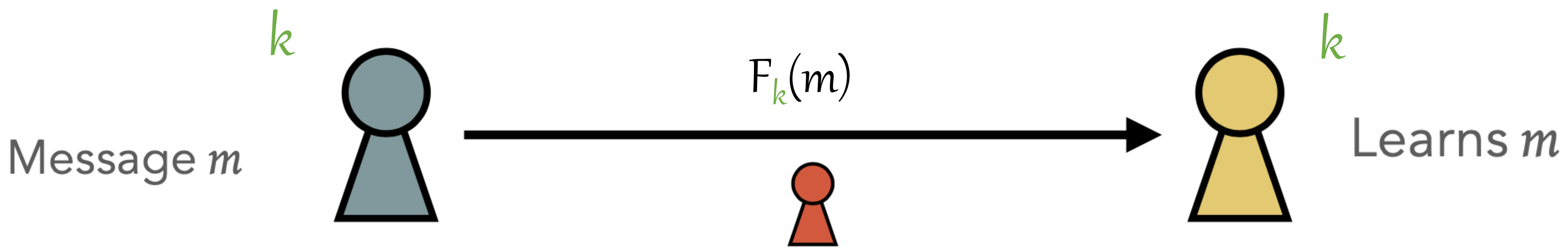
In essence, this protocol is saying “Let’s use the ith permutation function”

Infeasible to store all permutation functions – so instead cryptographers construct *pseudorandom functions*

What we have, approximately: Pseudo-Random Functions

Shared secret: index k chosen uniformly, at random

k is our key!



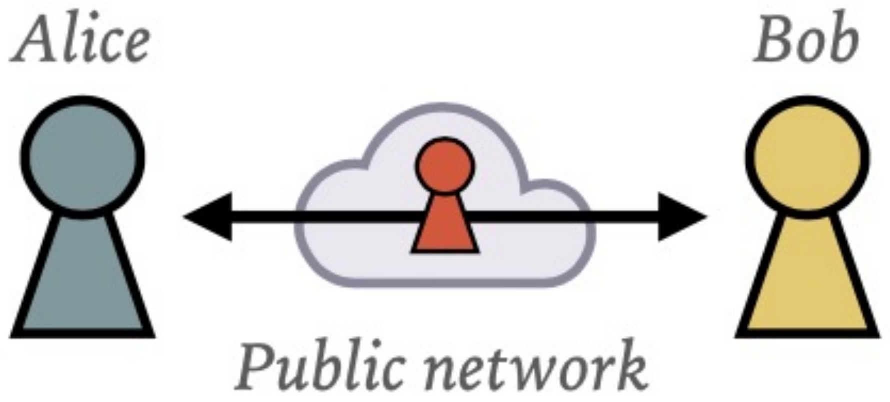
Without knowing *k*, Eve learns nothing about *m*

In essence, this protocol is saying “Let’s use the ith permutation function”

Infeasible to store all permutation functions – so instead cryptographers construct *pseudorandom functions*

A Perfectly Secure Encryption Scheme

Regardless of any prior information the attacker has about the plaintext
the ciphertext observed by the attacker
should leak no additional information about the plaintext.



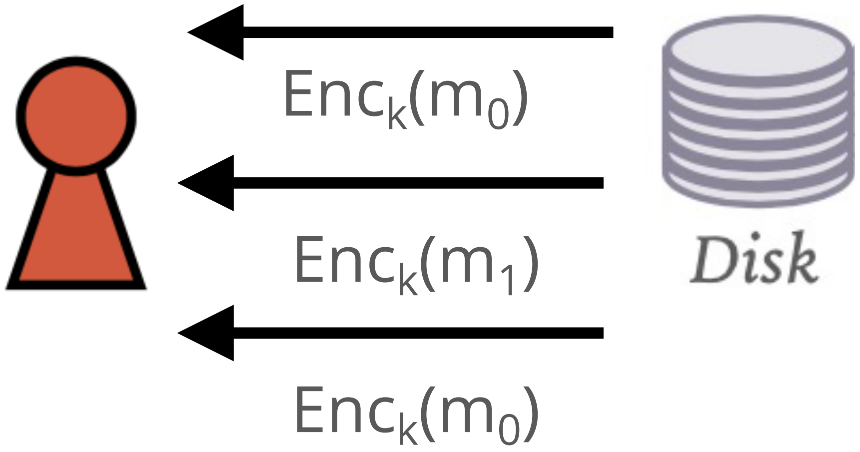
<u>Alice can only observe one ciphertext going over the network</u>

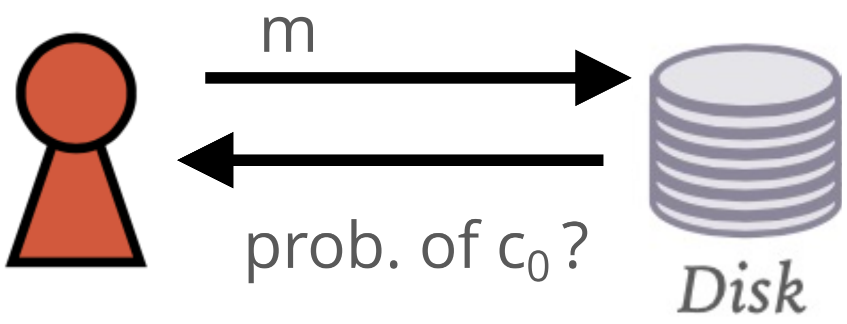
Computational Secrecy

Would be okay if a scheme leaked information with a tiny probability to eavesdroppers with bounded computational resources.

- Allowing security to fail with a tiny probability (negligible in key length n)
  - how tiny is tiny? 2^{-60} : probability of an event occurring every 100 billion years!
- Only consider efficient attackers (bounded in polynomial time by key length)
  - attackers that can brute-force the key space in bounded time.
  - try testing 2^{112} keys? Would take a supercomputer since Big Bang!
  - modern key space? 2^{128} or more!

Multiple message secrecy: Impossible?





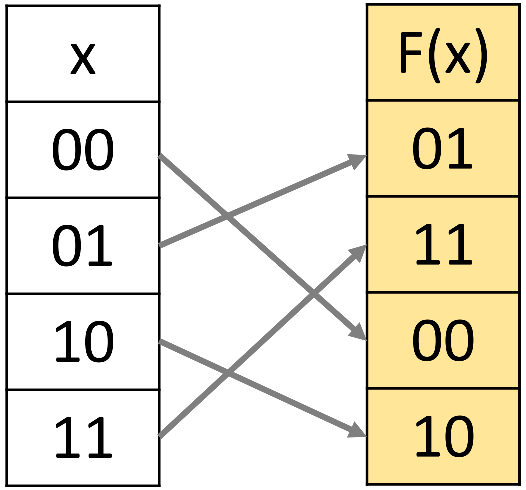
It really is a problem if an attacker can tell when the same message is encrypted twice!

This attack only works if encryption is deterministic!

Random Permutations

- Variant of random function is random permutation
  - treat them equivalently for our purposes .

- E.g.: random permutation over bit strings of length 2
  Encryption: $\{0, 1\}^2 \rightarrow \{0, 1\}^2$



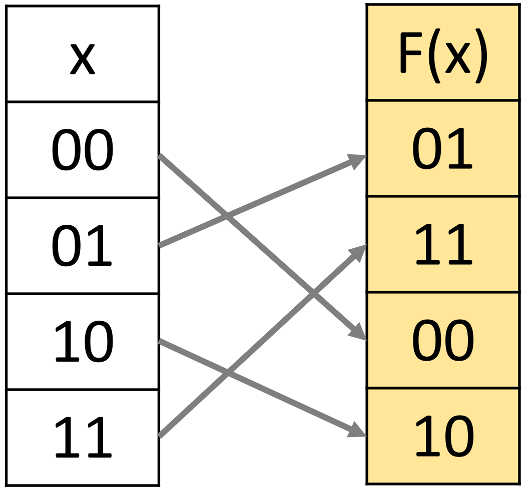
Important Property of the Random Permutation:
A permutation is invertible (bijective) function

Given F(x) it is impossible to determine x without resorting to a brute force attack.

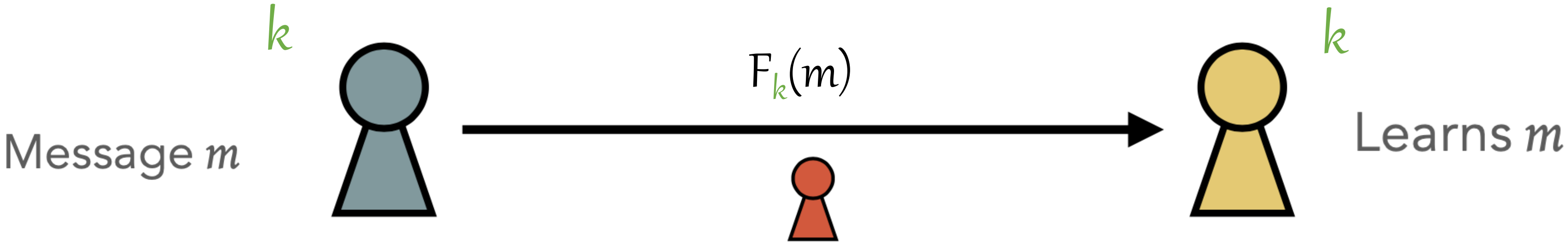
If |X| is very large? brute force not possible by an efficient (probabilistic polynomial time) attacker.

What we have, ideally: Random Functions

Consider the set of all permutations $F_k : X \to X$



Shared secret: index k chosen uniformly, at random



Without knowing k, Eve learns nothing about m

If you know k, then $F_k(x)$ is trivial to *invert*

If you don't know k, then $F_k(x)$ is *one-way*

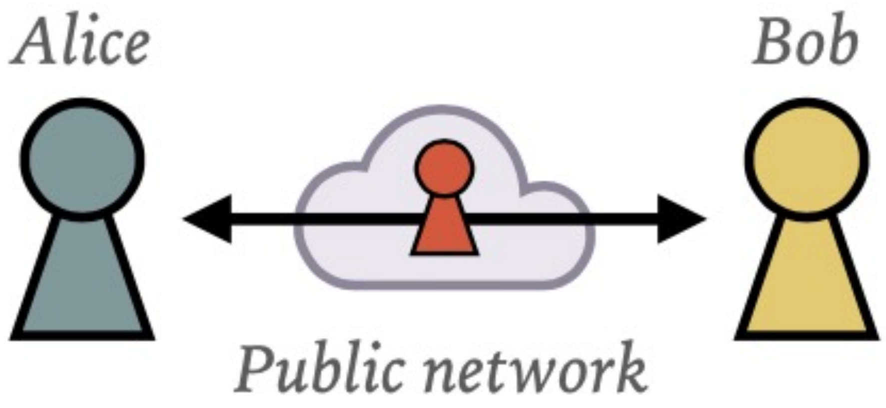
k is our key!

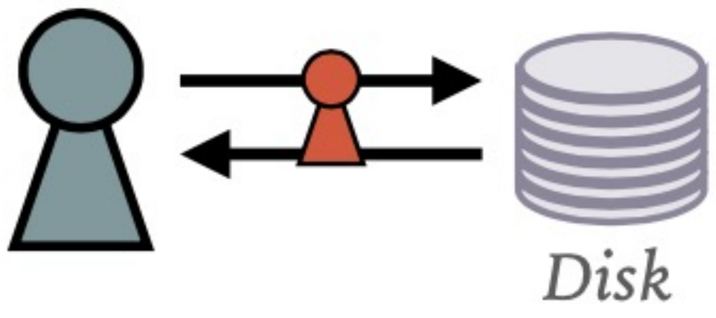
BLACKBOXES

To this end, we'll cover several "blackboxes": what properties do they provide, and how can we responsibly put them together

Block ciphers **MACs** **Hash functions** **Public key crypto**

Scenarios and Goals

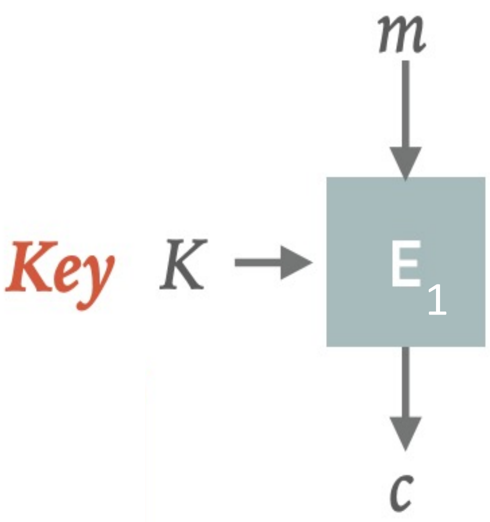




Confidentiality	Keep others from reading Alice’s messages/data	Block Ciphers
Integrity	Keep others from undetectably tampering with Alice’s messages/data	
Authenticity	Keep others from undetectably impersonating Alice (keep her to her word too!)	

Block Ciphers

ENCRYPTION



Encryption Function: E: $\{0, 1\}^k \times \{0, 1\}^n \rightarrow \{0, 1\}^n$

Fix the key K, then, E_k: $\{0, 1\}^n \rightarrow \{0, 1\}^n$

- plaintext size: n
- ciphertext size:n

E_k: permutation on n-bit strings.

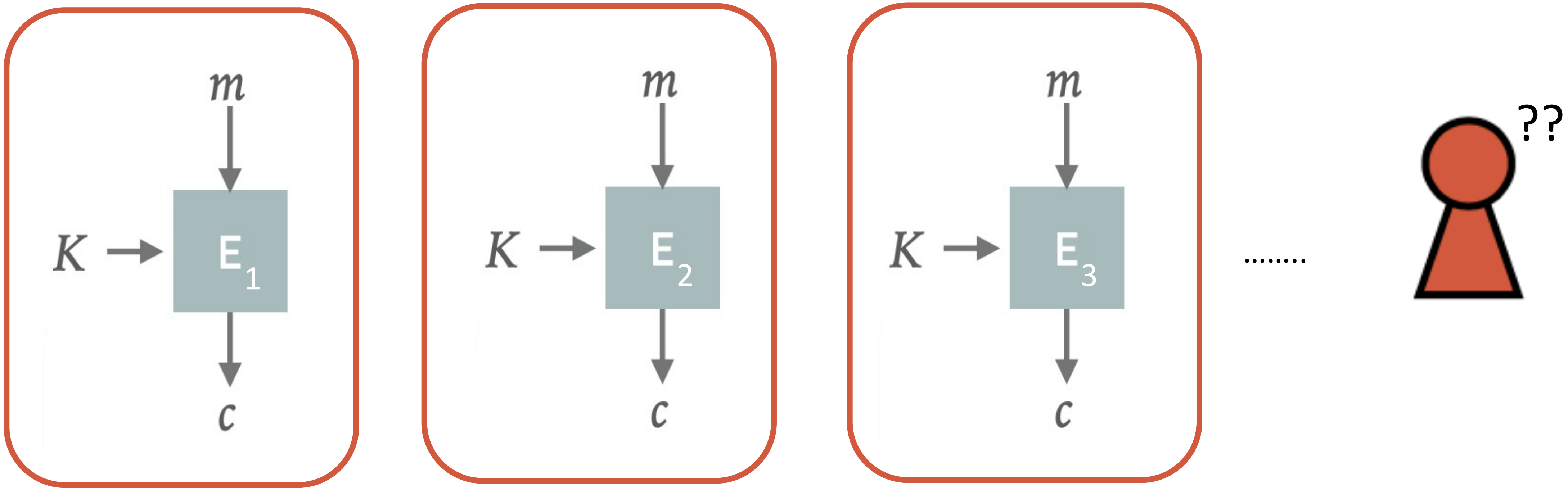
- invertible (bijective function) given the key

Once the key is fixed: E(k,m) is indistinguishable from a function chosen uniformly at random from all possible functions between block-sized binary strings.

Block Ciphers

ENCRYPTION

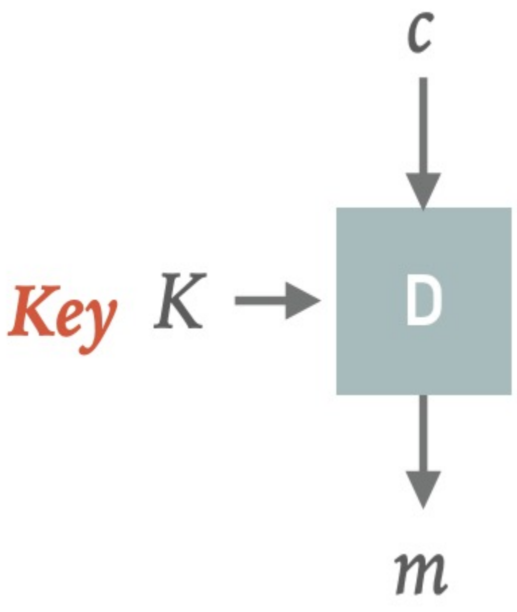
Once the key is fixed: E(k,m) is indistinguishable from a function chosen uniformly at random from all possible functions between block-sized binary strings.



Attacker has no way of knowing which random function was chosen to permute the plaintext to the ciphertext

Block Ciphers

DECRYPTION



Inverse mapping of the permutation is the decryption algorithm, given the key

$D_k(E_k(M)) = M$

without the key: best attack is a brute force exhaustive search over the entire key space!

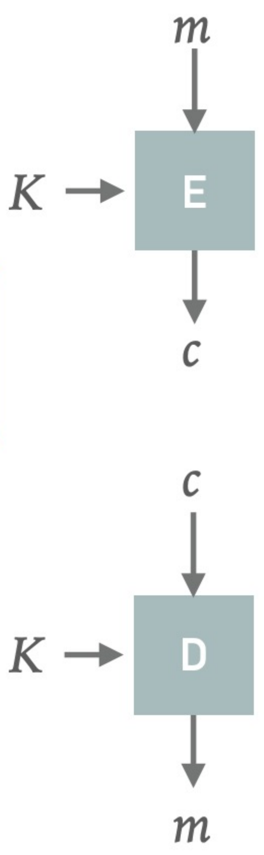
Attacker has no way of knowing which random function was chosen to permute the plaintext to the ciphertext

Block Ciphers

$\{0, 1\}^k \times \{0, 1\}^n \rightarrow \{0, 1\}^n$

ENCRYPTION

Key



AES key sizes:
128, 192, 256

DECRYPTION

Plaintext ("message")

Same fixed block size
(AES: 128 bits, 3DES: 64 bits)

Ciphertext

PROPERTY:
Block ciphers are deterministic
For a given m and K,
E(K,m) always returns the same c

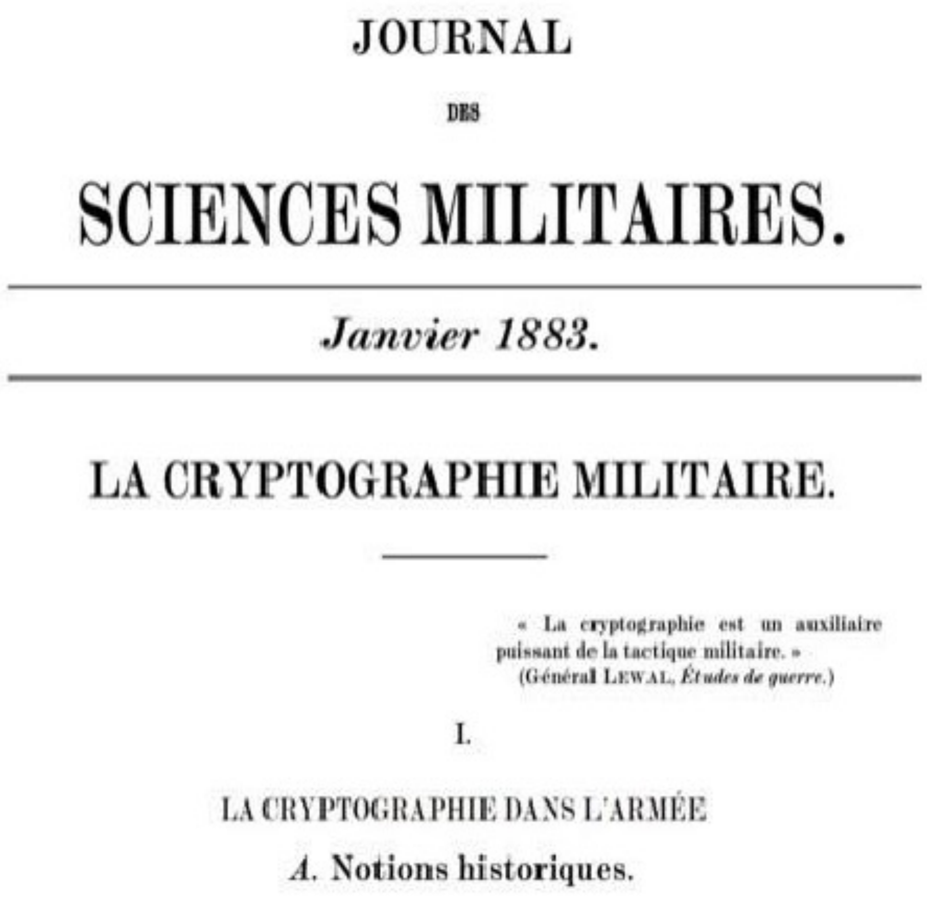
PROPERTY: ***Small changes to the inputs cause big changes in the output***

Confusion: *Each bit of the ciphertext should depend on each bit of the key*

Diffusion: *Flipping a bit in m should flip each bit in c with Pr = 1/2*

Kerckhoffs' principle

Encryption and Decryption and Key Generation Algorithm are publicly known. *The only unknown is the shared secret key*



JOURNAL

DES

SCIENCES MILITAIRES.

Janvier 1883.

LA CRYPTOGRAPHIE MILITAIRE.

« La cryptographie est un auxiliaire puissant de la tactique militaire. »
(Général LEWAL, *Études de guerre*.)

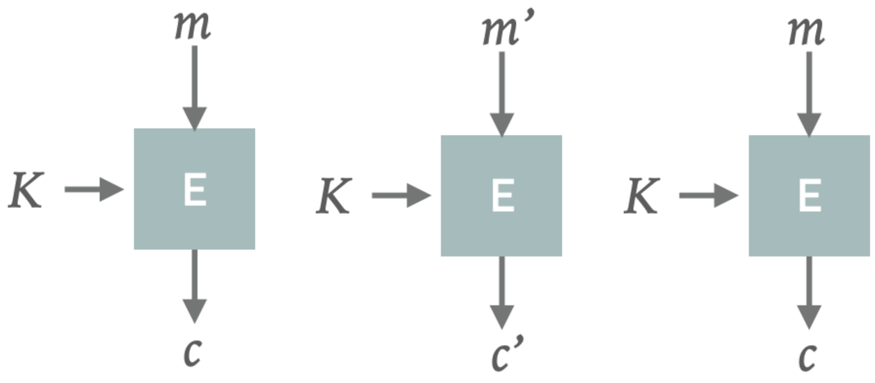
I.

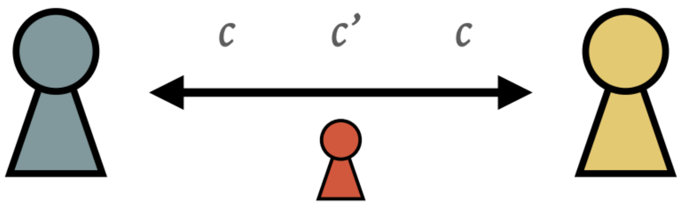
LA CRYPTOGRAPHIE DANS L'ARMÉE

A. Notions historiques.

Wikipedia

Problem #1: Block Ciphers Are Deterministic





PROPERTY:

Block ciphers are deterministic

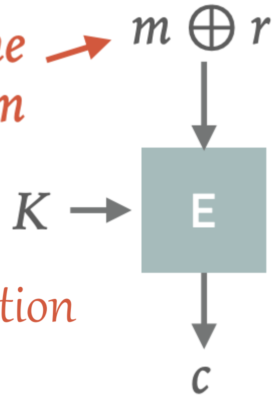
For a given m and K,
E(K,m) always returns the same c

An eavesdropper could determine when messages are re-sent

A FIX:

Note: $m \oplus r$ is the same size as m

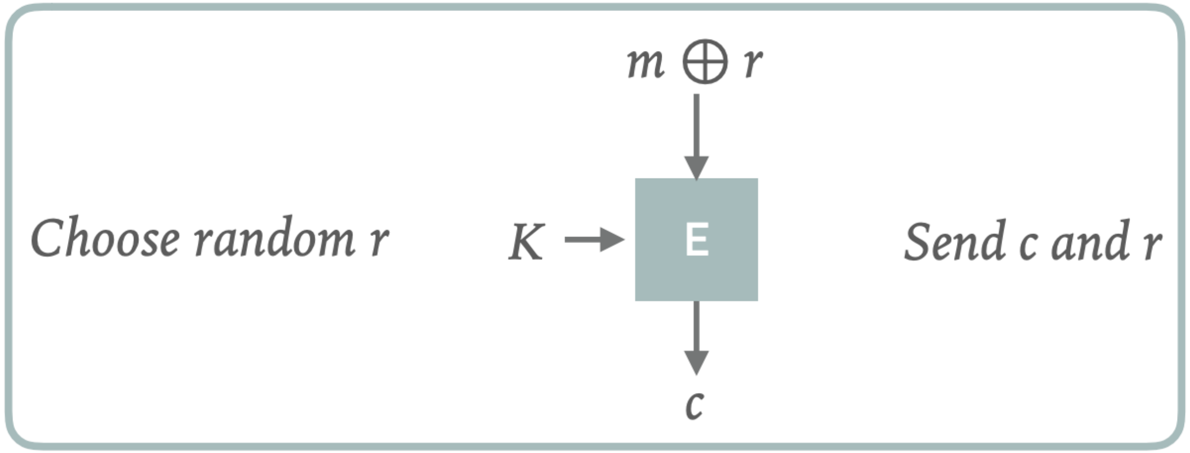
Choose random r



Send c and r

Also known as an Initialization Vector or Nonce

Initialization Vector (nonce)



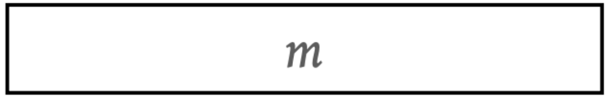
IV or r needs to be different (unpredictable) each time

Random: *Must send r with the message*
This is good if messages can be reordered

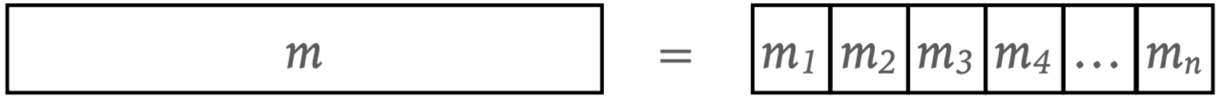
Counter: *Don't need to send r; the receiver can infer it from the message number*
This is good if messages are delivered in-order

Problem #2: Block Ciphers have fixed size

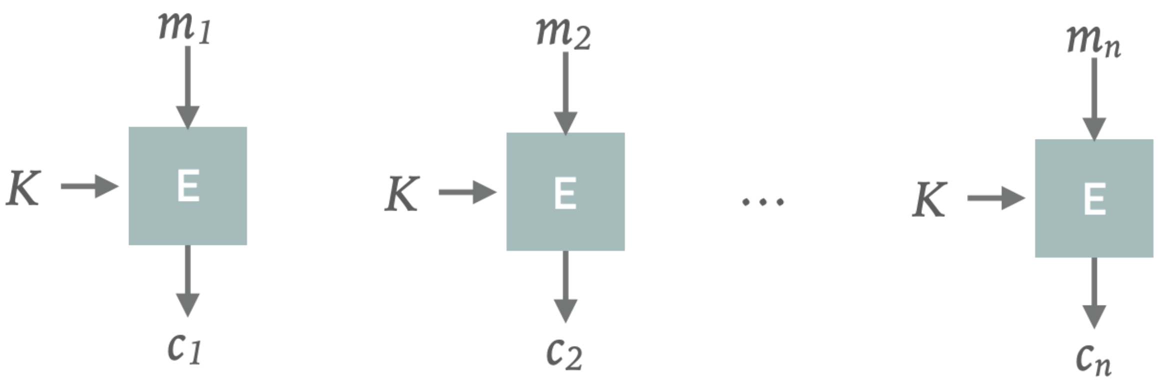
Fixed block size m



If we want to encrypt a message larger than the block size (128 bits), we simply break up the message into block-size length pieces...



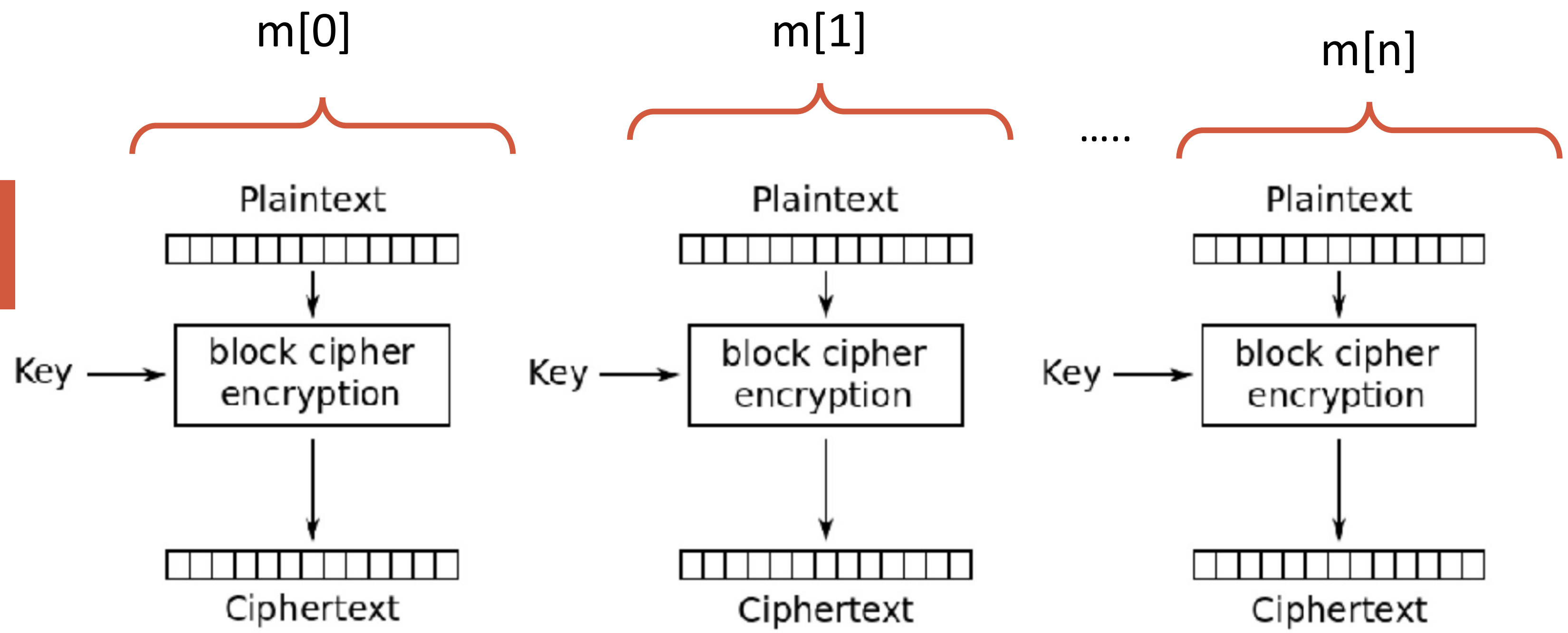
...and encrypt each block



But recall: it can be deterministic. We must choose good initialization vectors. How?

Modes of Encryption: Electronic Codebook Mode (ECB)

each block in AES 128 bits



Electronic Codebook (ECB) mode encryption

Encryption:

inputs: plaintext: m, key: k,

ciphertext: c[i] = E(k, m[i])

Decryption:

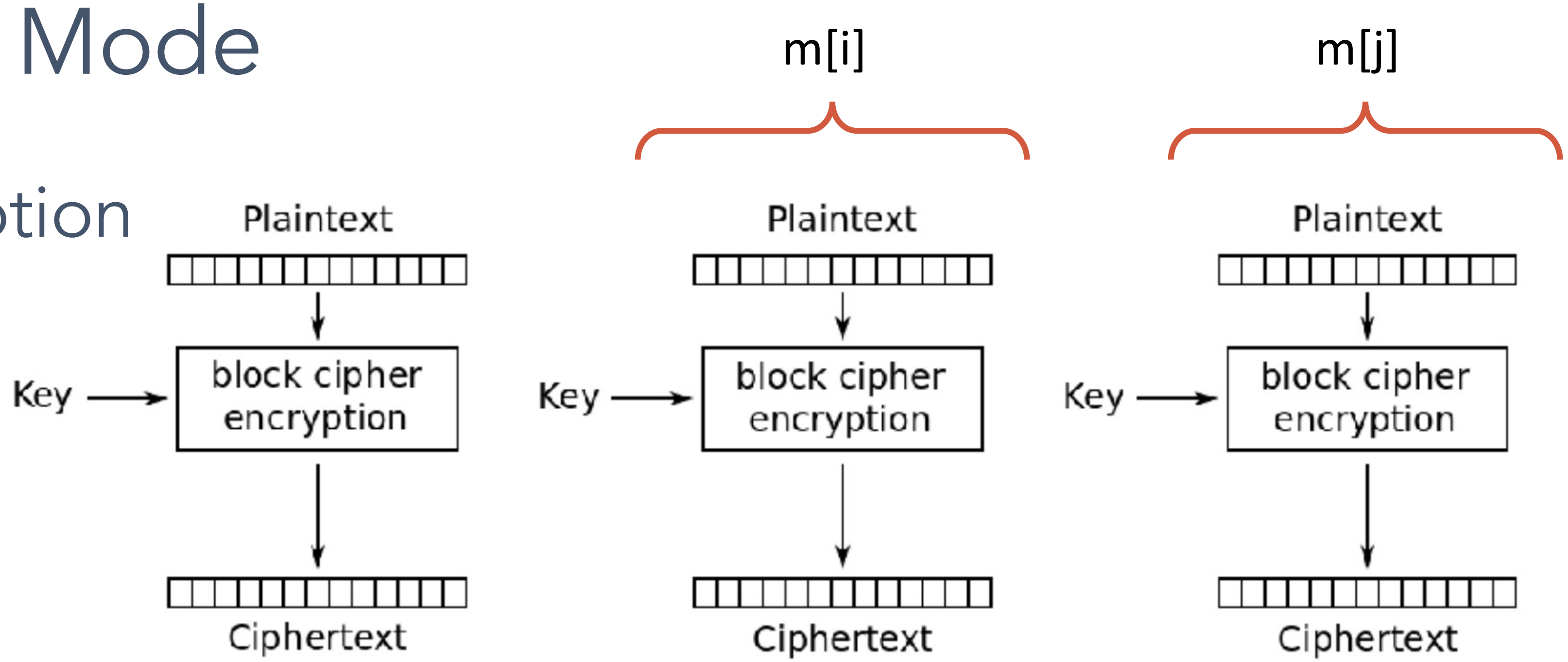
inputs: ciphertext: c, key: k,

plaintext: m[i] = D(k, c[i])

spot the problem?

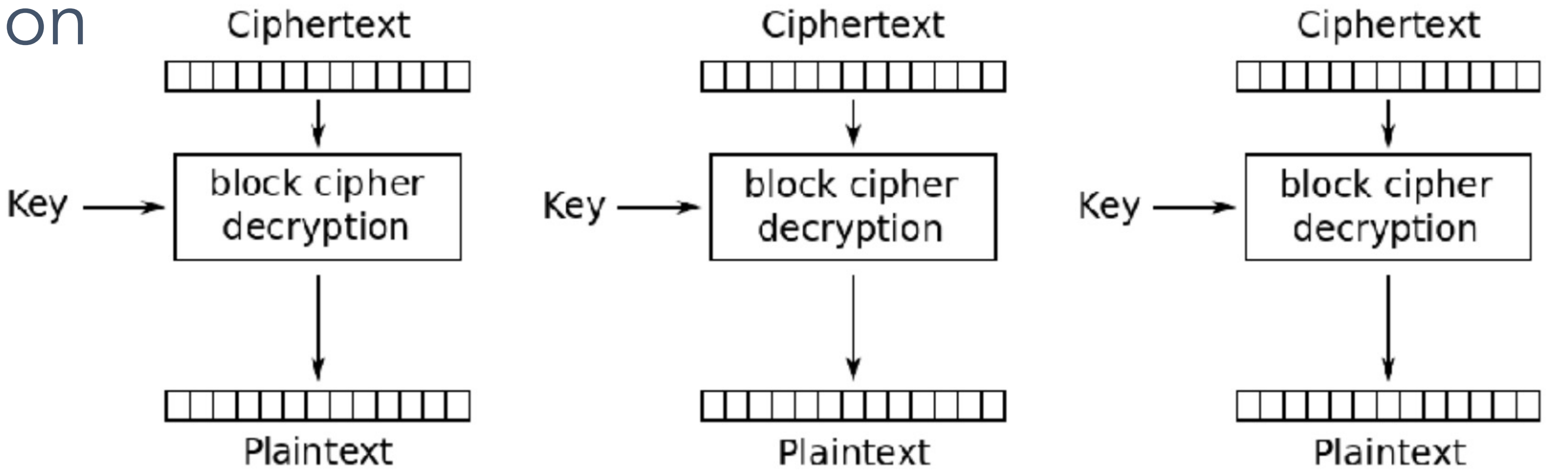
ECB Mode

Encryption



If two separate segments are equal, m[i] = m[j]. Then Eve can detect this by noting c[i] = c[j]

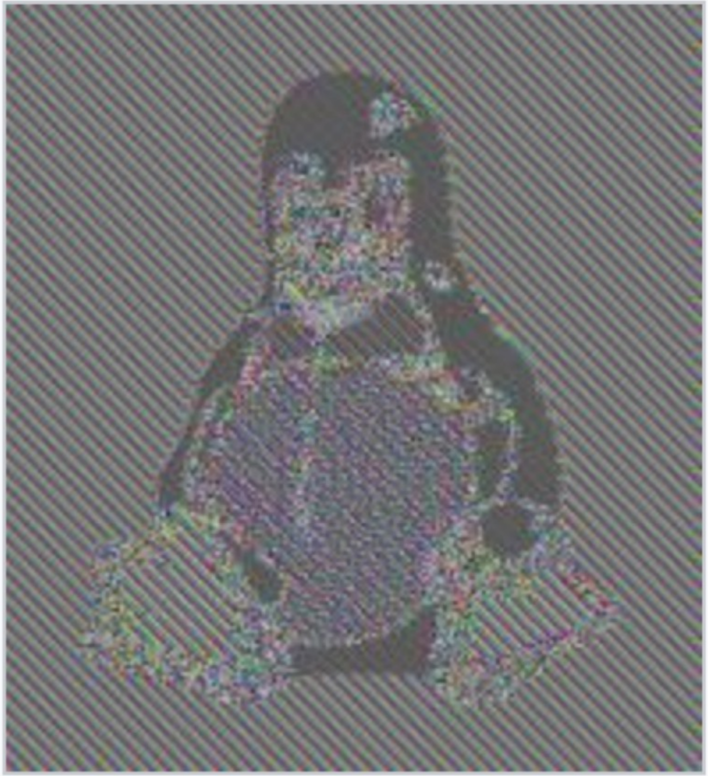
Decryption



Same issue that led us to use Initialization Vectors!

NEVER USE THE ECB MODE!

Original image



Encrypted using ECB mode

NEVER use ECB
(but over 50% of Android apps do)

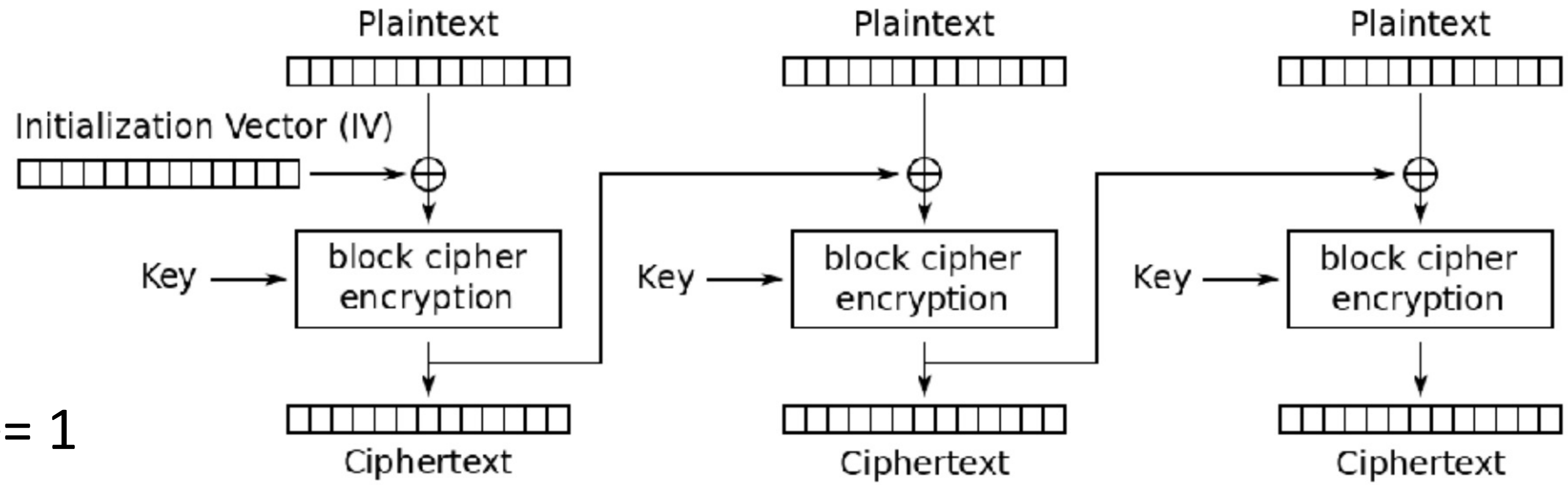
Modes of Encryption: Cipher Block Chaining Mode (CBC)

Encryption

input: plaintext m,
key k,
initialization vector IV

c[0] = IV

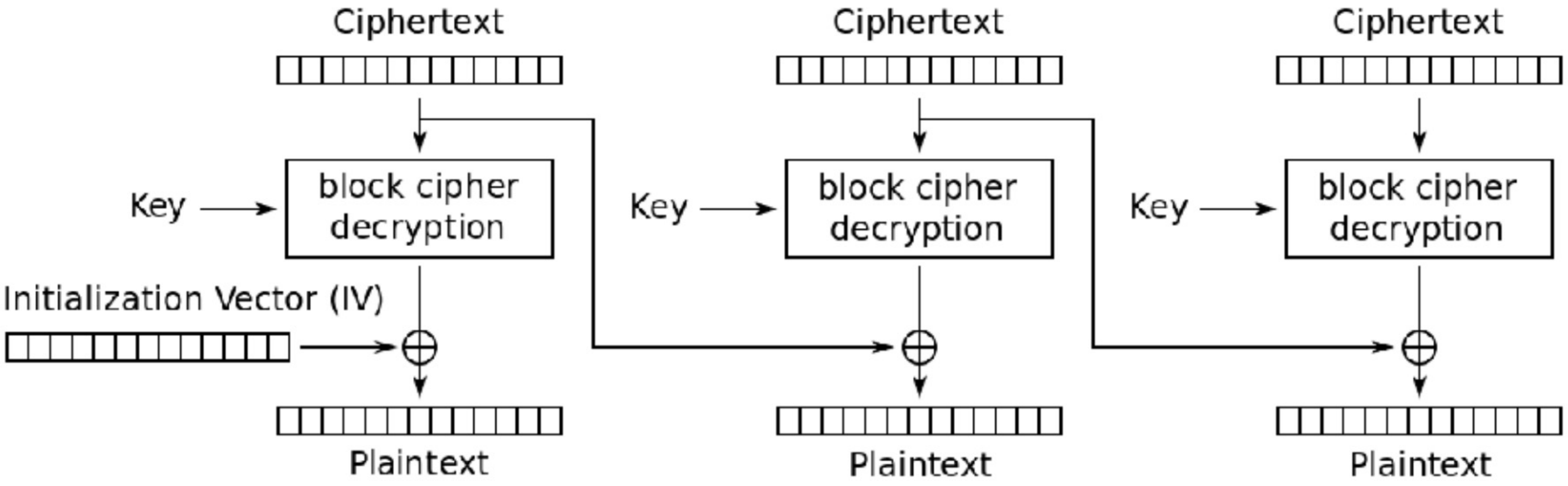
c[i] = E(k, m[i] $\oplus$ c[i−1]) for i >= 1



Decryption

input: ciphertext c,
key k,
initialization vector IV

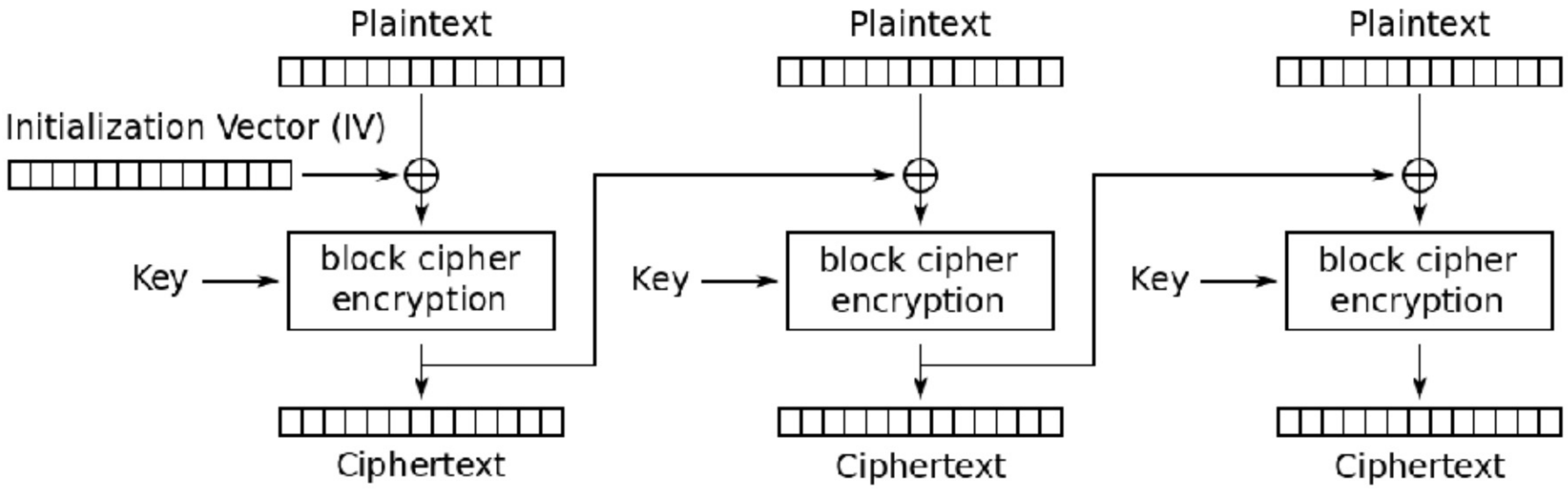
m[i] = D(k, c[i]) $\oplus$ c[i−1]



Modes of Encryption: Cipher Block Chaining Mode (CBC)

Security

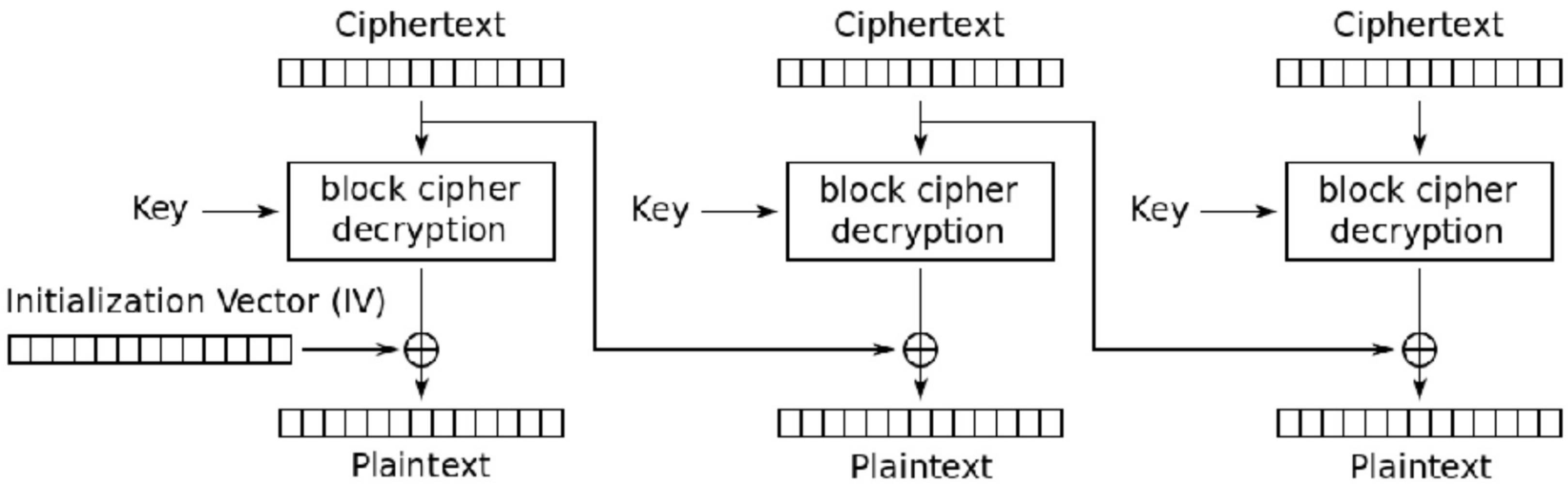
Input to the Encryption algorithm at each step is extremely likely to be different from the previous step.



Performance

Encryption: Not Parallelizable

Decryption: Parallelizable recovering m[i] does not require m[i-1]. Only requires c[i-1] which is already known.



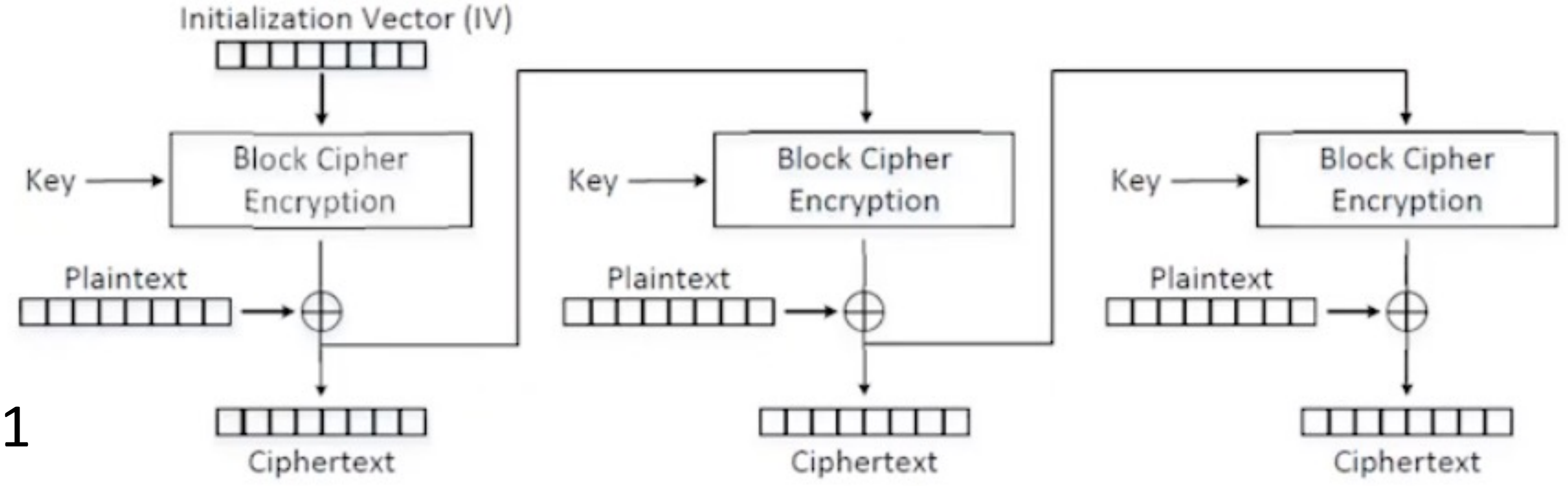
Modes of Encryption: Cipher Feedback Mode (CFB)

Encryption

input: plaintext m,
key k,
initialization vector IV

c[0] = IV

c[i] = E(k, c[i-1]) ⊕ m[i] for i >= 1

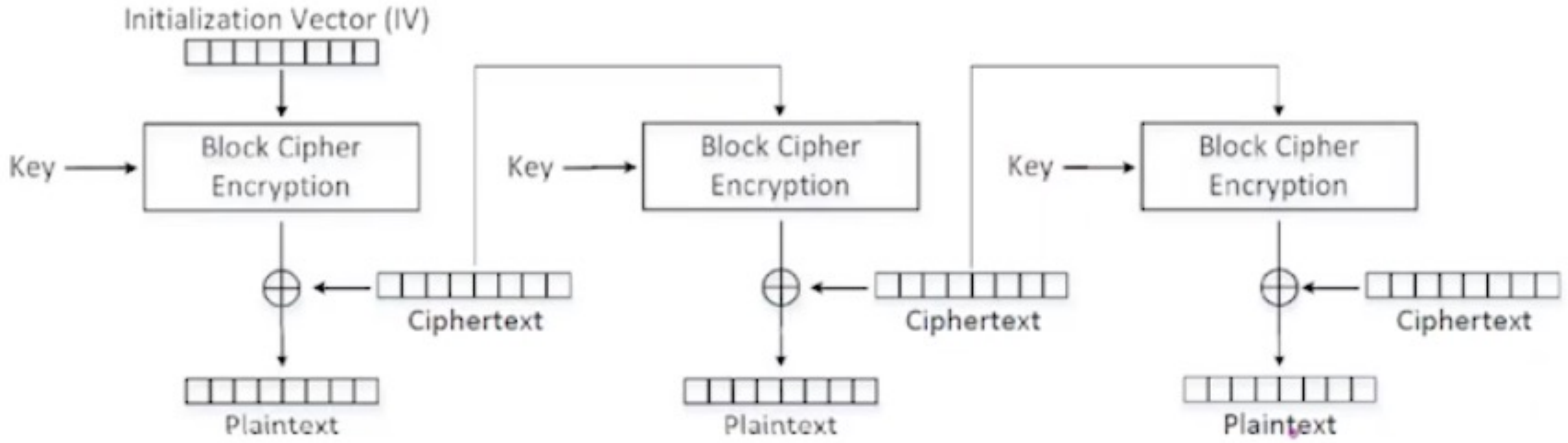


(a) Cipher Feedback (CFB) mode encryption

Decryption

input: ciphertext c,
key k,
initialization vector IV

m[i] = E(k, c[i-1]) ⊕ c[i]



(b) Cipher Feedback (CFB) mode decryption

Doesn't make use of the decryption function!

Modes of Encryption: Cipher Feedback Mode (CFB)

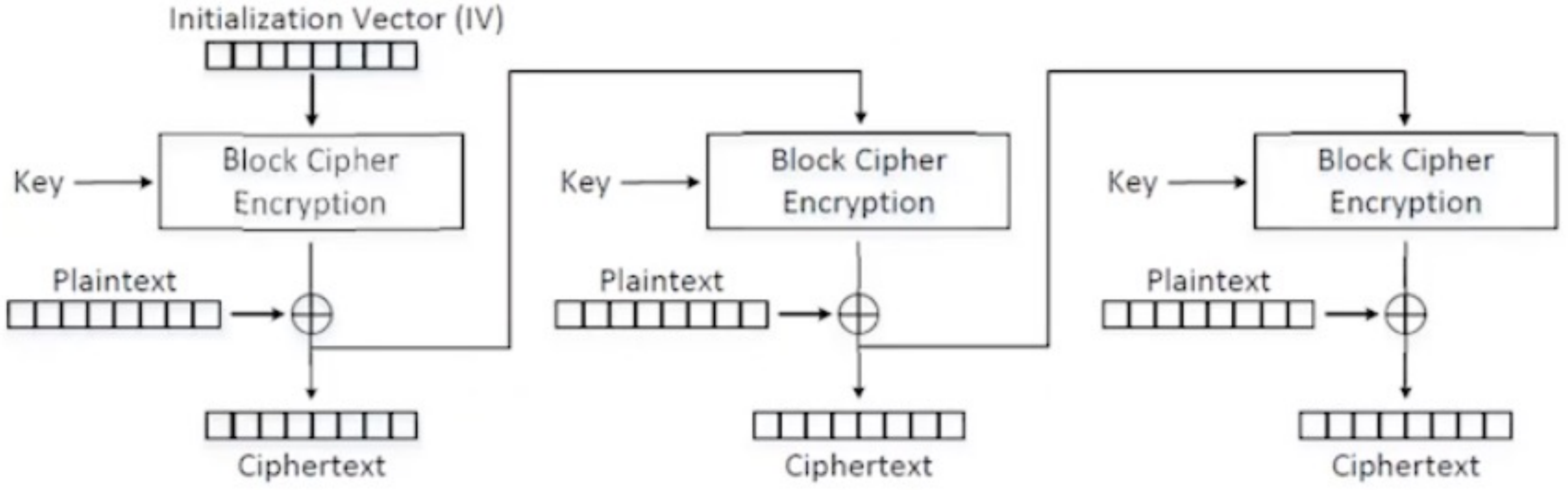
Security:

c[i] != c[j] for m[i] = m[j]

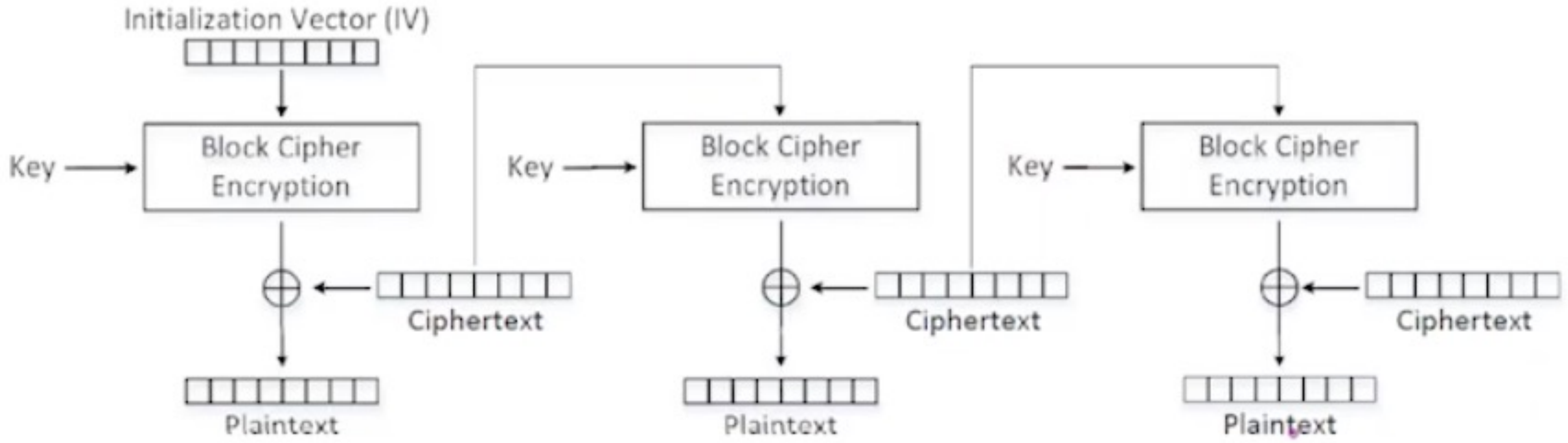
Performance

Encryption: Still Not Parallelizable

Decryption: Parallelizable recovering m[i] does not require m[i-1]. Only requires c[i-1] which is already known.



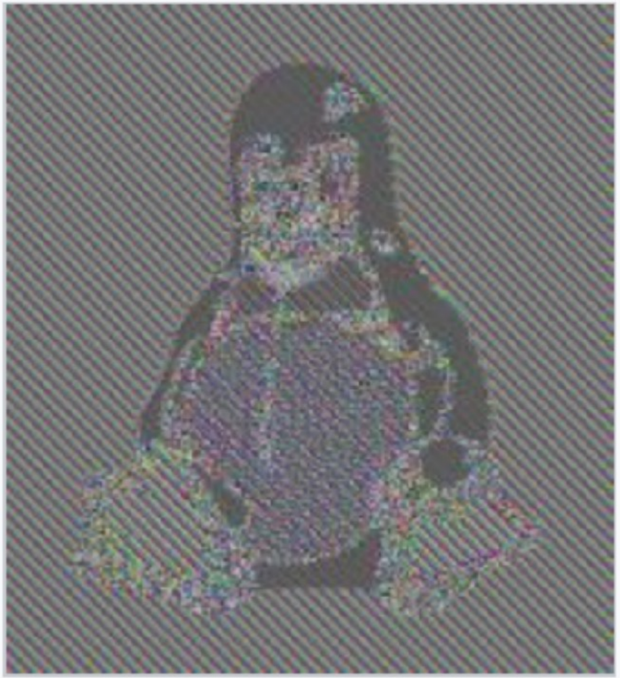
(a) Cipher Feedback (CFB) mode encryption



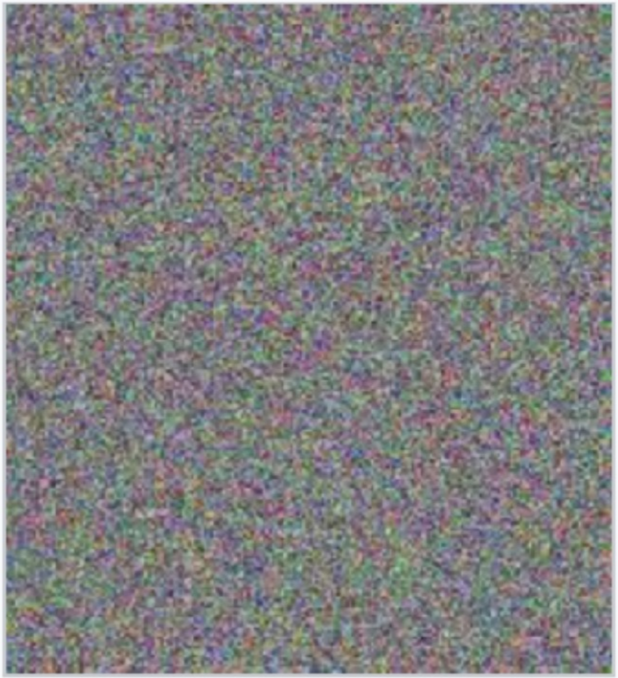
(b) Cipher Feedback (CFB) mode decryption

Doesn't make use of the decryption function!

Original image



Encrypted using ECB mode



Modes other than ECB result in pseudo-randomness

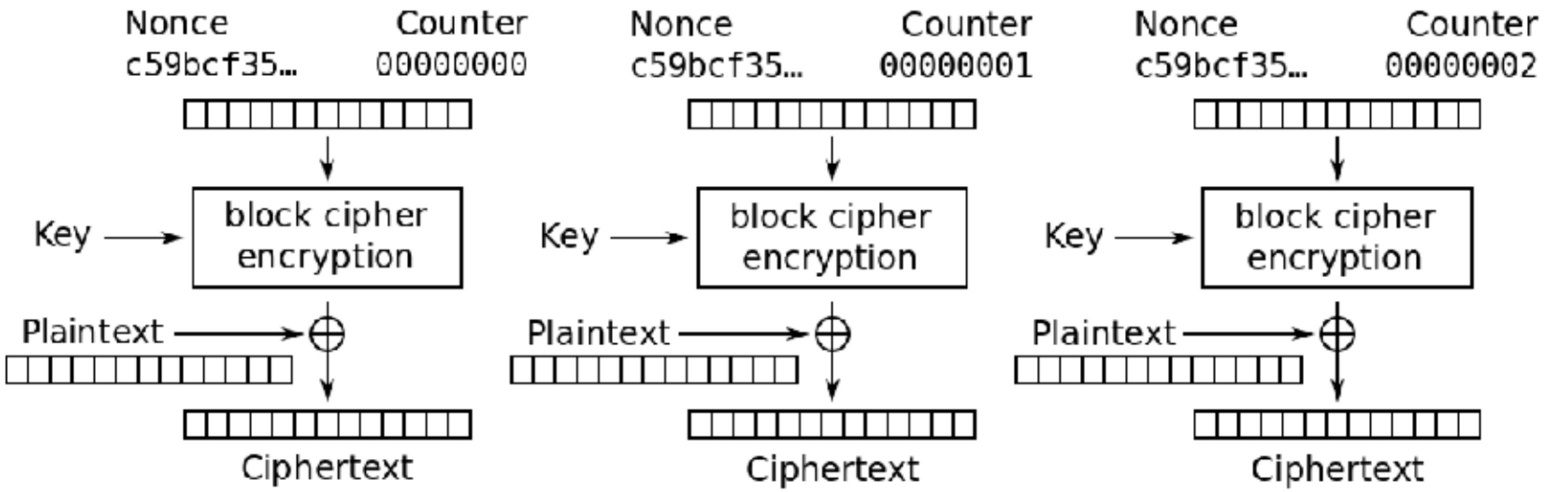
Modes of Encryption: Counter Mode (CTR)

Encryption

input: plaintext m,
key k,
initialization vector IV

c[0] = IV

c[i] = E(k, IV+i]) ⊕ m[i] for i >= 1



Decryption

input: ciphertext c,
key k,
initialization vector IV

m[i] = E(k, IV+i) ⊕ c[i]