Security in Informatics and in the Organizations (2018/2019)

Practical Class #3 (#4, #5):

Applied Cryptography

Vitor Cunha (vitorcunha@ua.pt)

Important Dates

Project

November 15th (Thursday) - Provide a written description of the security mechanisms to be used. No implementation is required and it will not involve any grading.

December 31st (Monday) - Final delivery of the code and report through CodeUA

1st week of January - Presentation and demo of the solution implemented

Theoretical Component

Intermediate Test - TBD *(optional)*

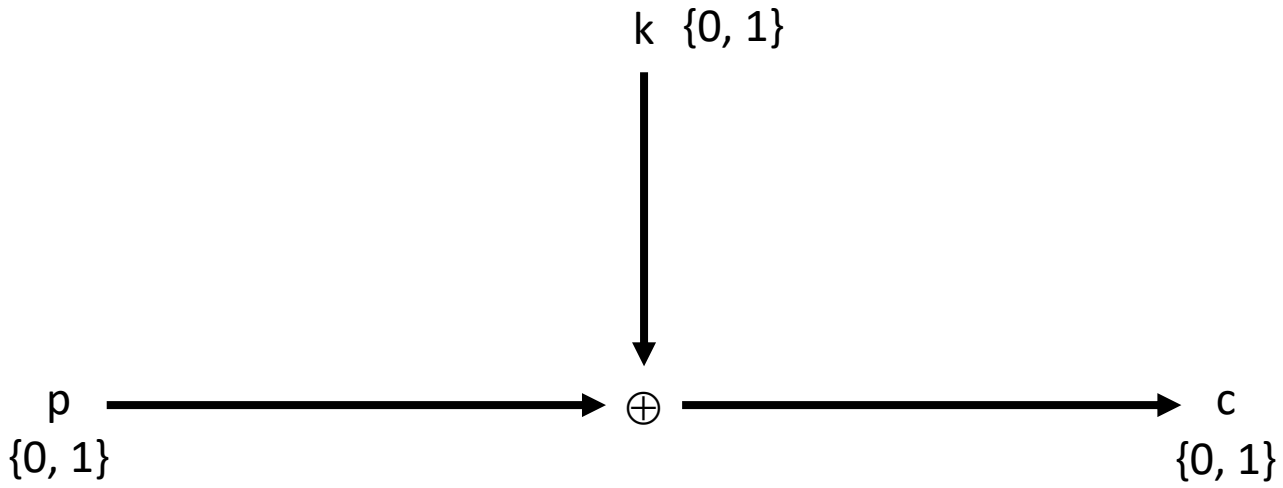
1 week before: OT to address the first two practical classes; ARP attacks, XSS and SQL injection.

Symmetric Cryptography

That is, the same key is used both to encrypt and decrypt.

One Time Pad

(Vernam, as per US Patents Office -- disputed)

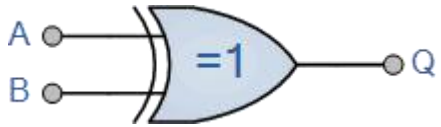


k	p	c
0	0	0
0	1	1
1	0	1
1	1	0

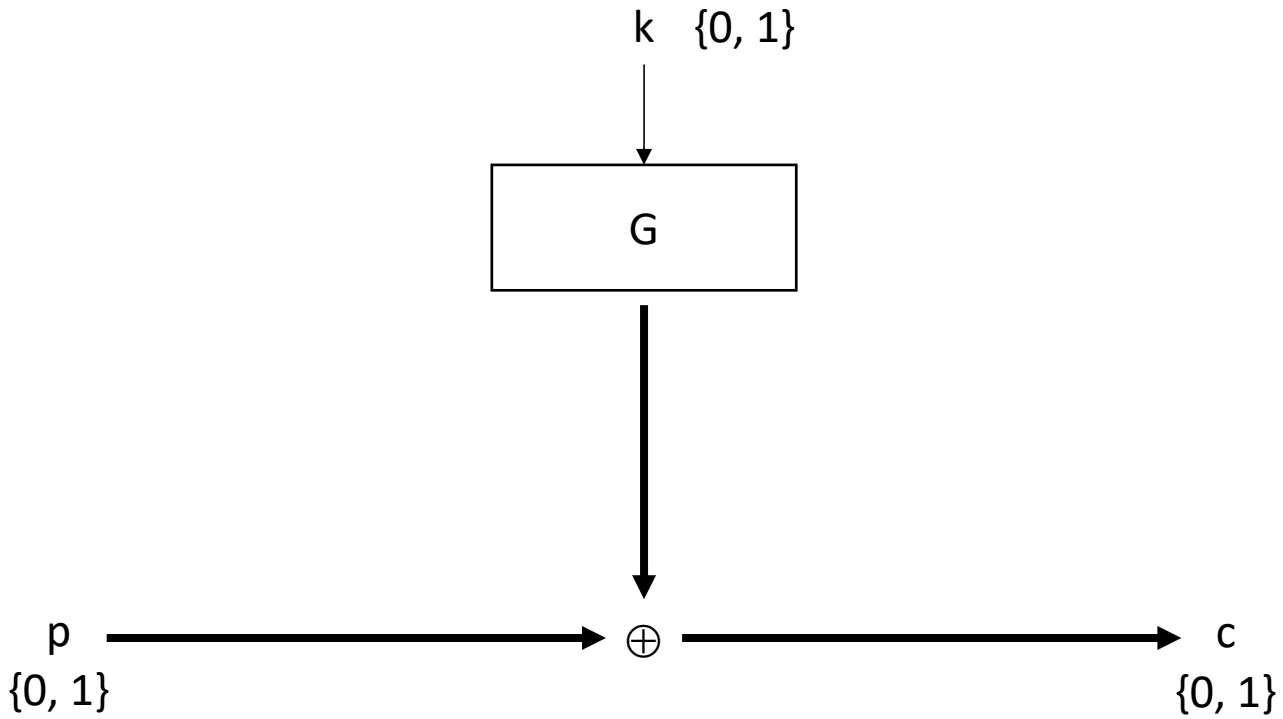
If k is truly random:

$P_c(0) = 1/2$

$P_c(1) = 1/2$



Symmetric Cryptography



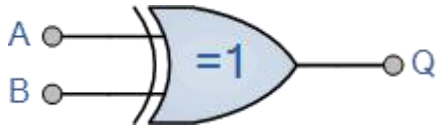
G	p	c
0	0	0
0	1	1
1	0	1
1	1	0

If G is not truly random:

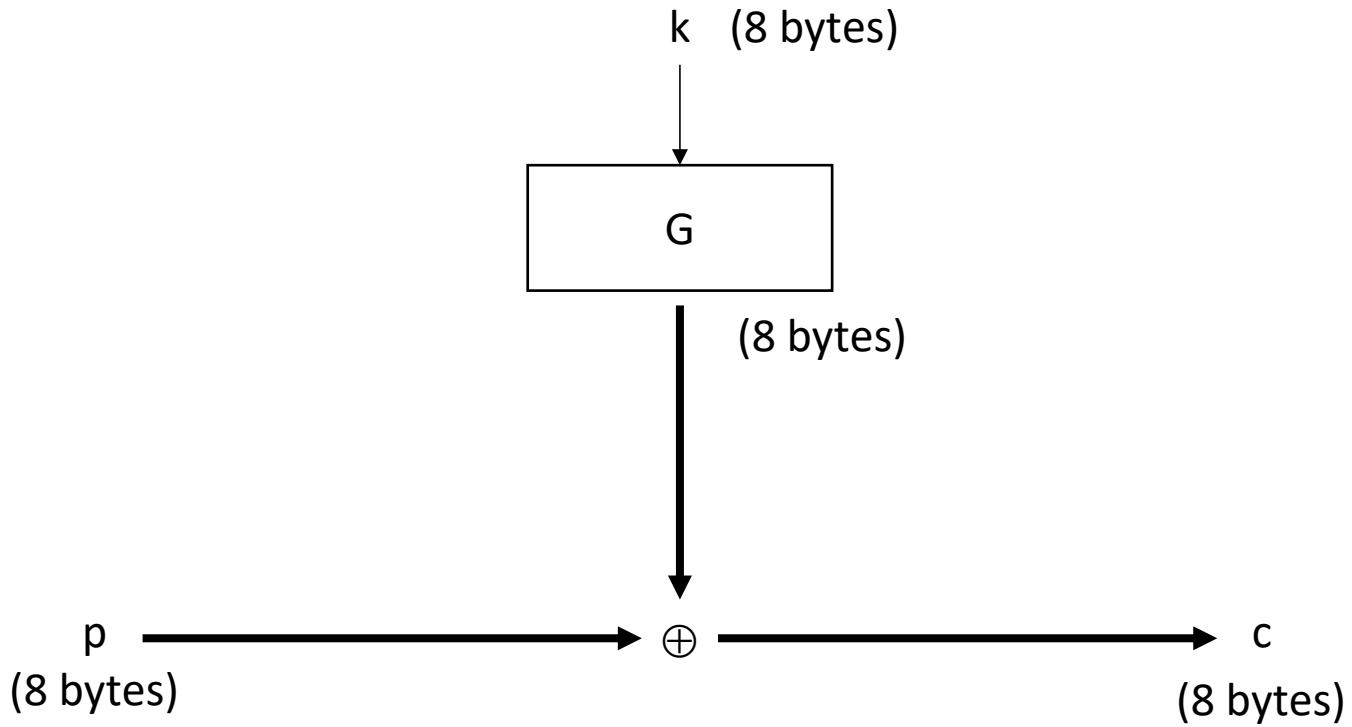
$P_c(0) = ½ + \varepsilon$

$P_c(1) = ½ + \varepsilon$

Secure if $\varepsilon < E$

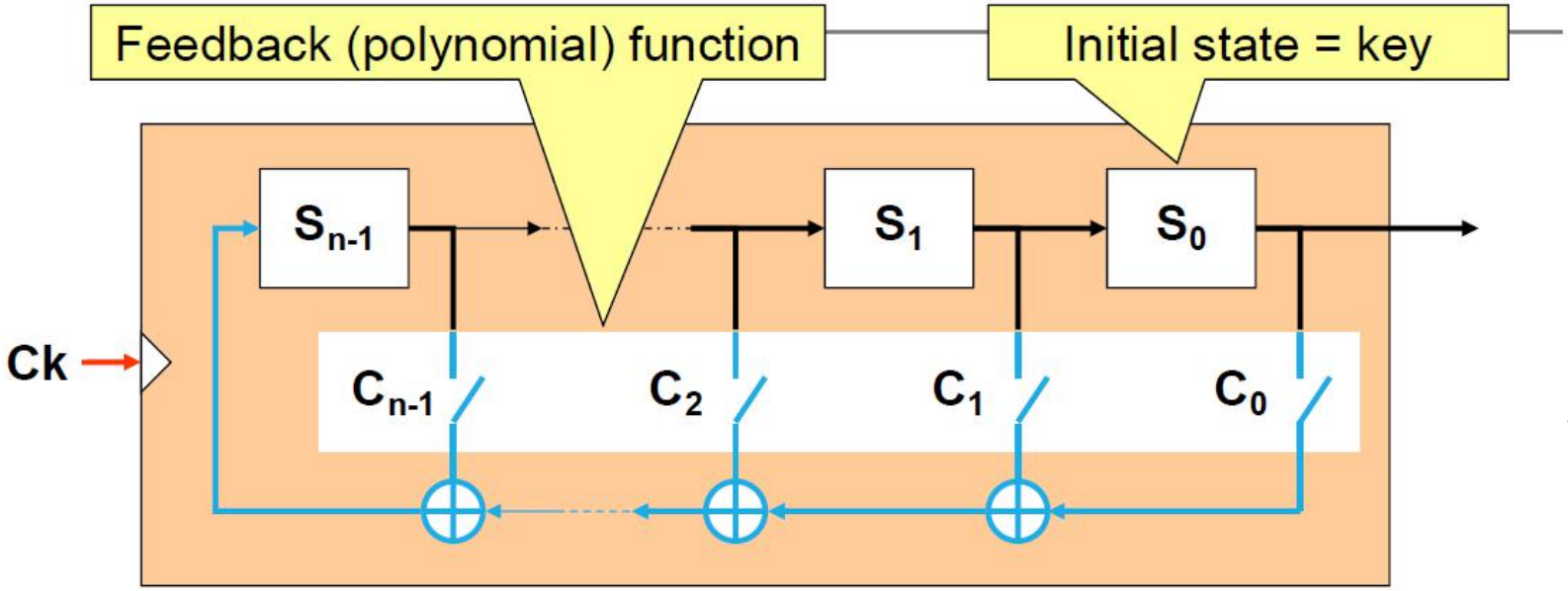


Symmetric Cryptography



Block ciphers allowed to control $\varepsilon < E$

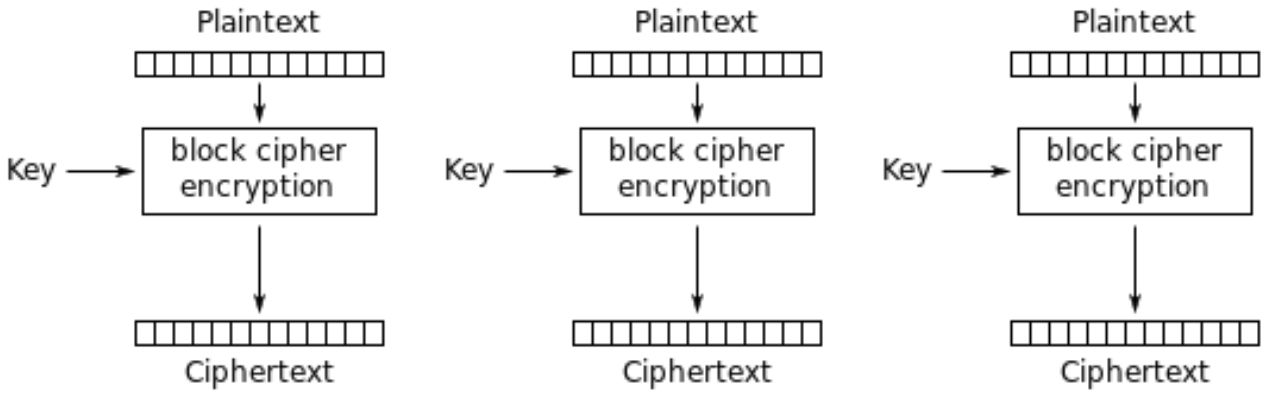
Symmetric Cryptography



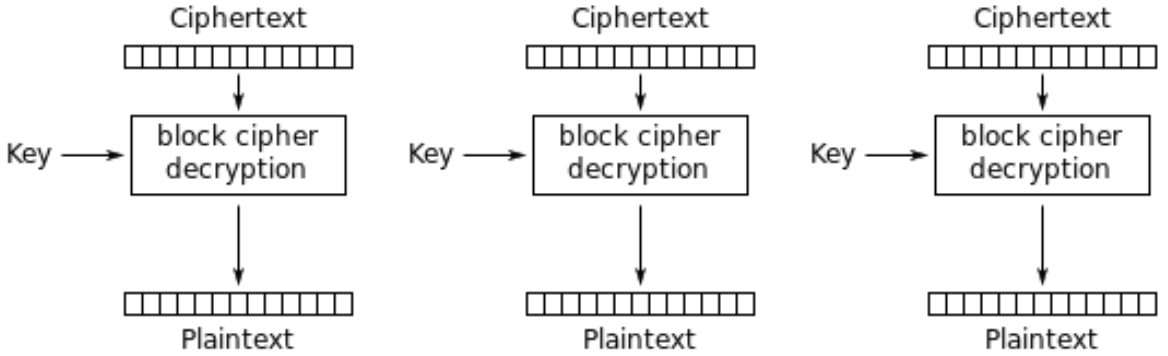
Linear Feedback
also controls ε *< E (*)*

see this morning slides for caveats

Symmetric Cryptography (cipher modes)

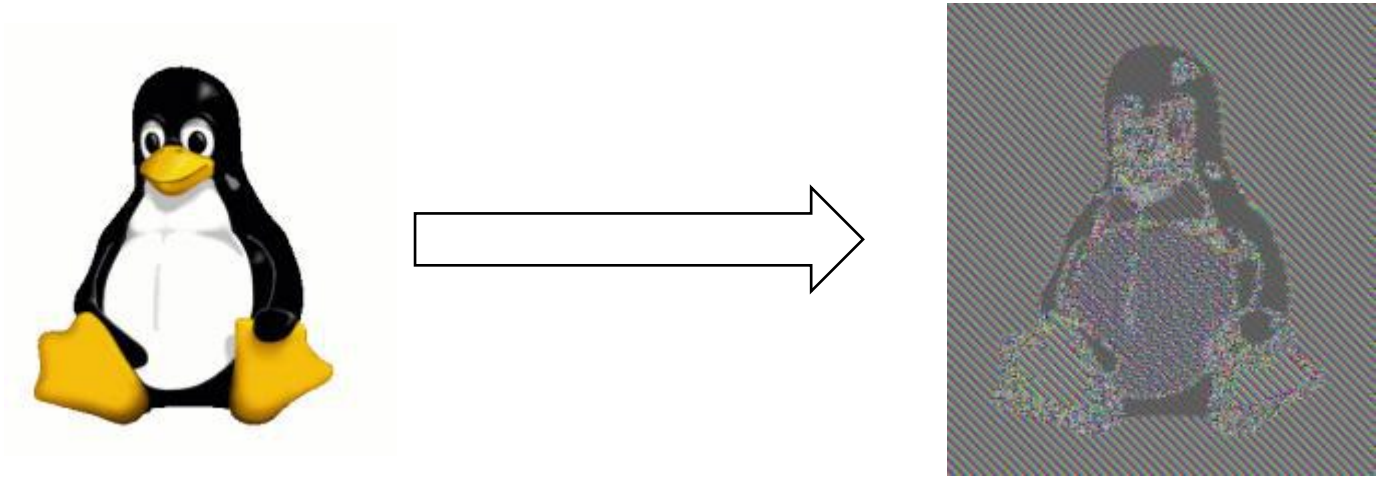


Electronic Codebook (ECB) mode encryption

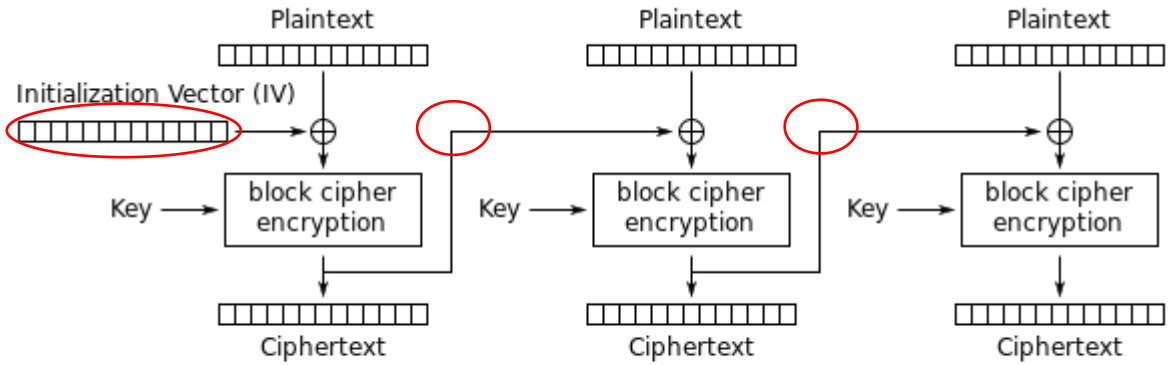


Electronic Codebook (ECB) mode decryption

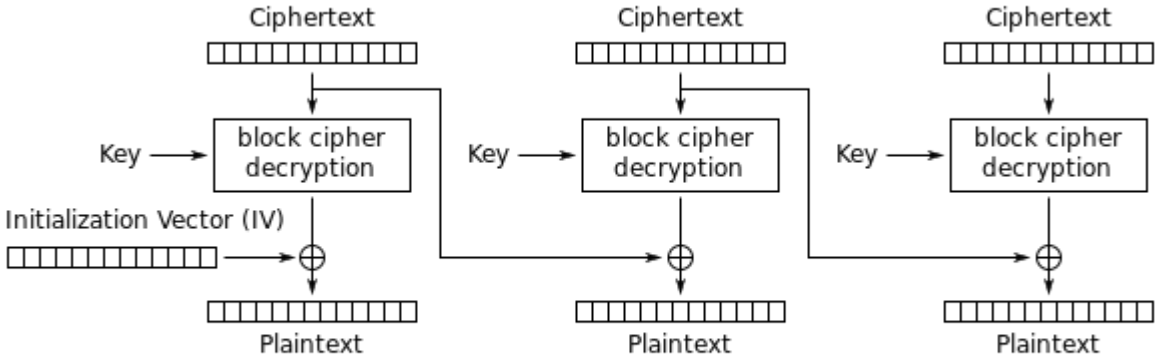
Symmetric Cryptography (ECB)



Symmetric Cryptography (CBC)

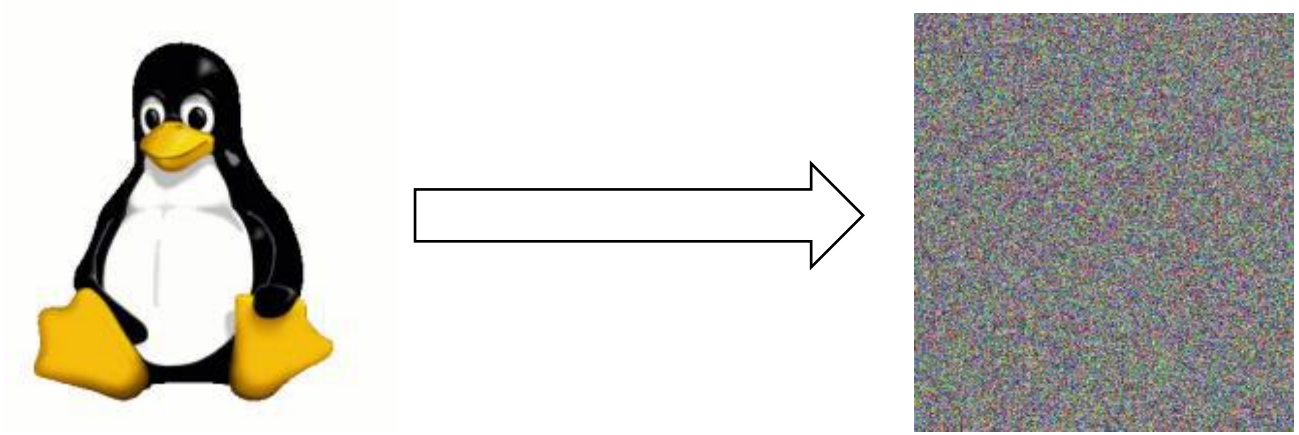


Cipher Block Chaining (CBC) mode encryption

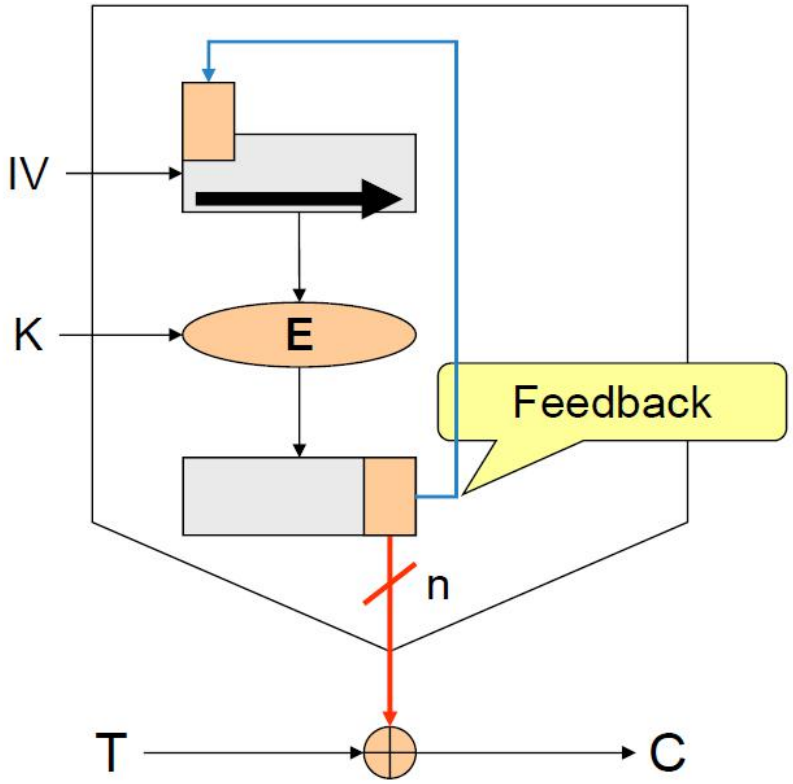


Cipher Block Chaining (CBC) mode decryption

Symmetric Cryptography (CBC)



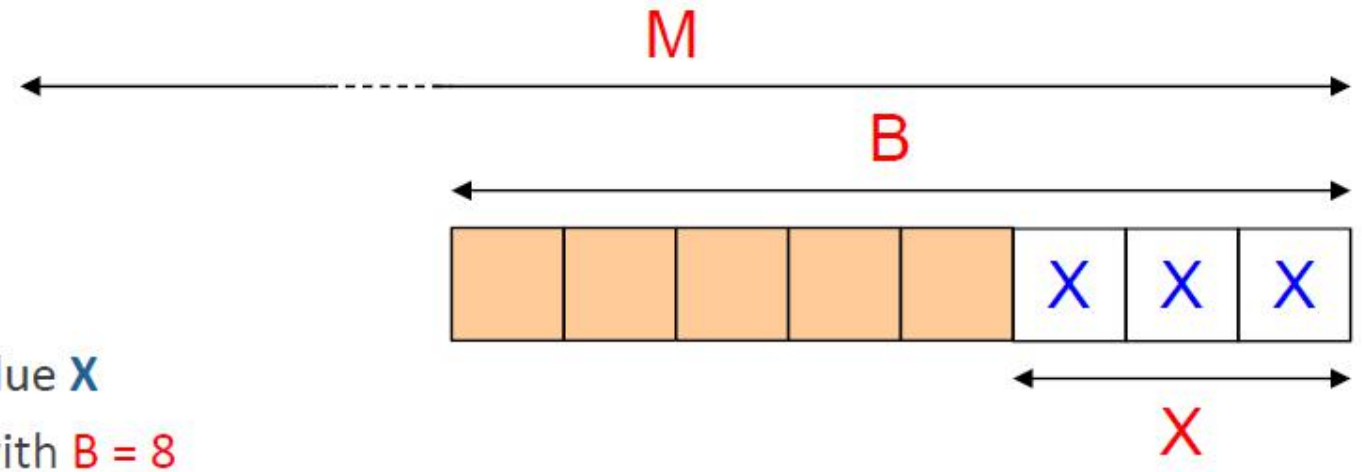
Symmetric Cryptography (OFB)

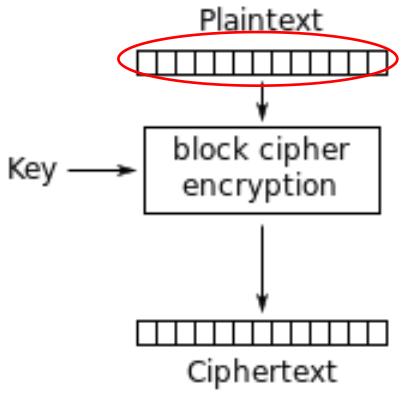


Turn a block cipher into a stream cipher

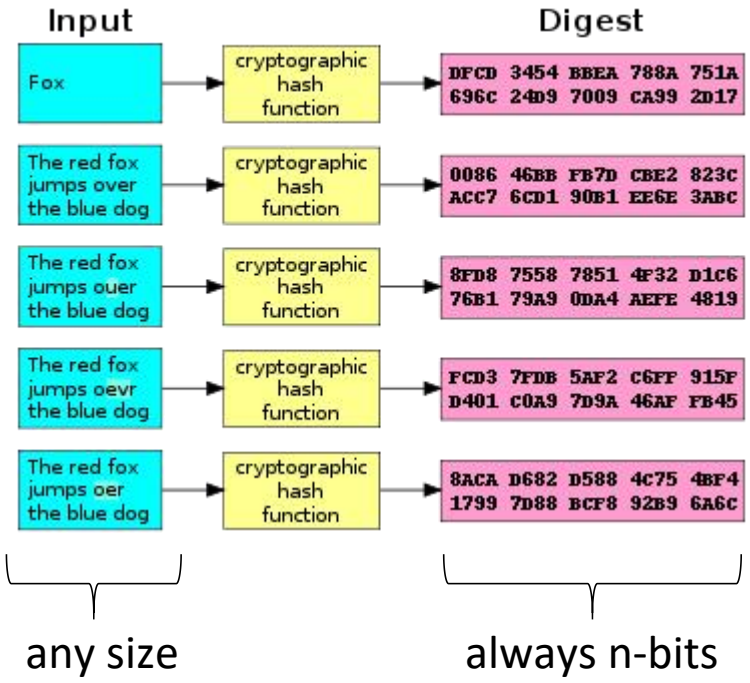
Symmetric Cryptography (Padding)

- Padding
  - Of last block, identifiable
  - PKCS #7
    - **X = B – (M mod B)**
    - **X** extra bytes, with the value **X**
  - PKCS #5: Equal to PKCS #7 with B = 8
- Different processing for the last block
  - Adds complexity





Digest Function (aka “Hash”)



1 - Pre-image resistance

Given a hash value h it should be difficult to find any message m such that h = hash(m). This concept is related to that of a one-way function. Functions that lack this property are vulnerable to preimage attacks.

2 - Second pre-image resistance

Given an input m1, it should be difficult to find a different input m2 such that hash(m1) = hash(m2). Functions that lack this property are vulnerable to second-preimage attacks.

3 - Collision resistance

It should be difficult to find two different messages m1 and m2 such that hash(m1) = hash(m2). Such a pair is called a cryptographic hash collision. This property is sometimes referred to as strong collision resistance. It requires a hash value at least twice as long as that required for pre-image resistance; otherwise collisions may be found by a birthday attack.

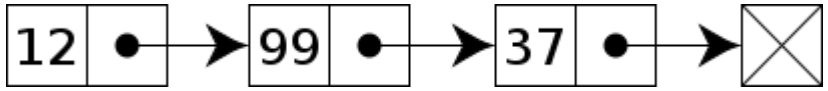
Asymmetric ciphers in next classes!

That is, when you cannot share the same key for encryption and decryption.

hint: useful for authentication (project)

10 seconds introduction to a simple blockchain

linked list



Same concept as a linked list, only that each new insertion validates the actual content before it (instead of just holding the reference)

Uses digests (hash) to validate that content. However, is that enough to ensure integrity?

Any Project Questions?