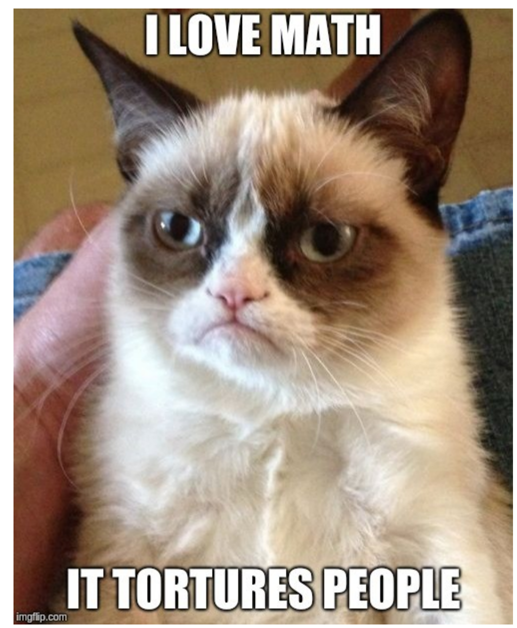
Communication Secrecy

Contents

Introduction: providing confidentiality

- We'd like to provide confidential communication
  - Only the intended recipient(s) should be able to read the data

- Two types of encryption and decryption
  - Symmetric ciphers
  - Asymmetric ciphers



Symmetric Ciphers

- A cipher defined over (*K*, *M*, *C*) is a pair of "comp. eff." algorithms (***E***, ***D***), where

$$\boldsymbol{E}: K \times M \to C$$
$$\boldsymbol{D}: K \times C \to M$$

  s. t. for all *k* in *K* and *m* in *M*:

$$\boldsymbol{D}(k, \boldsymbol{E}(k, m)) = m$$

- ***E*** is often randomized, ***D*** is always deterministic

Perfect Secrecy

- What is a "secure" cipher?
  - Shannon: Cipher text should reveal "no information" about the plain text
- A cipher (***E***, ***D***) over (*K, M, C*) has **perfect secrecy** if for all $m_0, m_1 \in M$ ($|m_0|=|m_1|$) and for all $c \in C$

$$\Pr[\boldsymbol{E}(k, m_0) = c] = \Pr[\boldsymbol{E}(k, m_1) = c]$$

  where $k \in K$ is randomly chosen
  - Given cipher text c, one cannot tell whether c is a cryptogram of m_0 or m_1

One Time Pad

- Vernam (1917)
  - $M = C = K = \{0, 1\}^n$
  - $E(k,m) = k \oplus m$
  - $D(k, c) = k \oplus c$
- Features
  - Given a truly random key, OTP has ***perfect secrecy***
  - Key has to be ***random*** and it must be used ***only once***
  - <u>Impractical:</u> Shannon shows that perfect secrecy requires keys to be at least as long as the plain text

Pseudo Random Generator

- Idea: Replace a "random" with a "pseudorandom" key

$$\boldsymbol{G}: \{0, 1\}^s \to \{0, 1\}^n \quad \text{where } n >> s$$

- Pseudo Random Generator (PRG) is a function ***G*** that maps *seed space* to *key space*
  - Is "efficiently" computable by a deterministic algorithm
  - Its output (keys) "looks random"
- Stream ciphers
  - $\boldsymbol{E}(k, m) := m \oplus \boldsymbol{G}(k)$
  - $\boldsymbol{D}(k, c) := c \oplus \boldsymbol{G}(k)$
- Examples: RC4, CSS, eStream, Salsa 20
- Can stream ciphers have perfect secrecy, why?

Stream Ciphers: perfect secrecy?

- Stream ciphers cannot have perfect secrecy
  - Keys (seeds) are shorter than messages
- Can stream ciphers ever be secure?
  - Need a new definition of security
  - Security will depend on PRG used

Pseudo Random Generators: defs

- **Statistical test** is an algorithm **A**: $\{0, 1\}^n \rightarrow \{0, 1\}$
  - Returns 1 if it *thinks* the input string is random, 0 otherwise
- **Advantage** of st. test ***A*** against PRG ***G***:
  $$\mathrm{Adv}_{\mathrm{PRG}}[A,G] = \left|\Pr_{k \xleftarrow{R} K}[A(G(k)) = 1] - \Pr_{r \xleftarrow{R} \{0,1\}^n}[A(r) = 1]\right|$$
  - If *close* to 0, ***A*** cannot distinguish ***G*** from random
  - Otherwise, ***A*** can distinguish ***G*** from random
- **<u>Def.</u>** A PRG G is secure, if for all eff. stat. tests A: $\mathrm{Adv}_{\mathrm{PRG}}[A,G]$ is negligible.

Negligible? Assume less than 2^{-80}

Pseudo Random Generators: defs

- <u>Def:</u> A PRG is **unpredictable** if given an initial sequence of bits (a prefix), one cannot *efficiently* predict the next bit (with probability higher than ½ + ε)
- **<u>Thm:</u>** A PRG is secure iff. it is unpredictable.
- In practice
  - Unknown if there are provably secure PRG
  - But we have heuristic candidates

Perfect secrecy, threat model

- (Recall) A cipher (***E***, ***D***) over (*K, M, C*) has **perfect secrecy** if for all $m_0, m_1 \in M$ ($|m_0|=|m_1|$) and for all $c \in C$
  $$\Pr[\boldsymbol{E}(k, m_0) = c] = \Pr[\boldsymbol{E}(k, m_1) = c]$$
  where $k \in K$ is randomly chosen
  - Given cipher text c, one cannot tell whether c is a cryptogram of m_0 or m_1
- **Threat model:** basis for reasoning about security
  - **Adversary's power:** what can she do
  - **Adversary's goal:** what is she trying to achieve

Semantic security: def

(for one-time key; adv. sees only one CT)

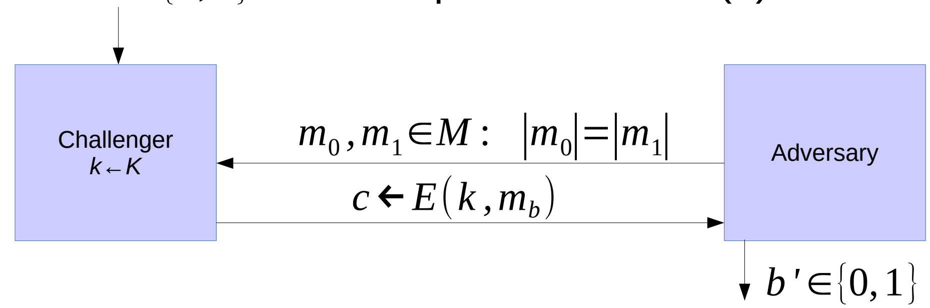
- Adversary's power: **observe <u>one</u> ciphertext**
  - Every message is encrypted with its own key; a particular key is used only once
- Adversary's goal: **learn about the plaintext**

Adaptation of: Dan Boneh, Cryptography I, Stanford.

Semantic security: def

(for one-time key; adv. sees only one CT)

- For $b \in \{0,1\}$ define experiments EXP(b) as



- Def: $\zeta = (E, D)$ is **semantically secure** if for all eff. adversaries A $\mathrm{Adv}_{SS}[A, \zeta]$ is negligible.

$$\mathrm{Adv}_{SS}[A, \zeta] := \left| \Pr[\mathrm{EXP}(0) = 1] - \Pr[\mathrm{EXP}(1) = 1] \right|$$

Adaptation of: Dan Boneh, Cryptography I, Stanford.

Semantic security

- Informally
  - A cipher has **semantic security** if given only cipher text, an attacker cannot *practically* derive any information about the plain text
- **<u>Thm:</u>** Given a secure PRG, derived stream cipher is semantically secure

Adaptation of: Dan Boneh, Cryptography I, Stanford.

Final thoughts

- Two-time pad attack
  - Never use stream-cipher key to encrypt more than one message
    - later we show a secure a multi-message exchange

$$c_1 \leftarrow m_1 \oplus \boldsymbol{G}(k)$$
$$c_2 \leftarrow m_2 \oplus \boldsymbol{G}(k)$$

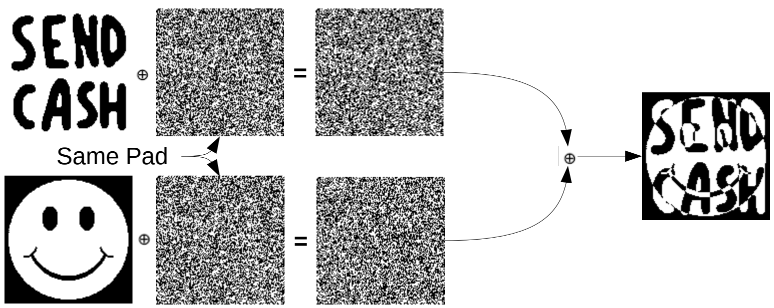
$$m_1 \oplus m_2 \leftarrow c_1 \oplus c_2$$

  - Redundancy in natural languages and in encoding schemes (ASCII, UTF-8, …) to separate $m_1 \oplus m_2 \rightarrow m_1, m_2$
  - http://www.crypto-it.net/eng/attacks/two-time-pad.html

https://crypto.stackexchange.com/questions/59/taking-advantage-of-one-time-pad-key-reuse

Final thoughts

- Two-time pad attack



https://crypto.stackexchange.com/questions/59/taking-advantage-of-one-time-pad-key-reuse

Final thoughts

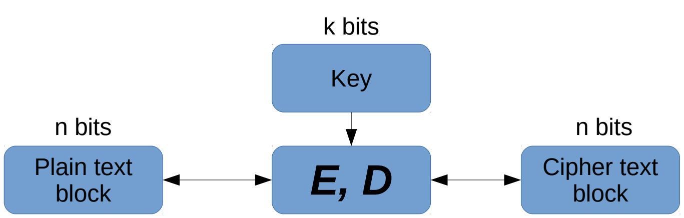
- Malleability
  - Modifications to CT are not detected and have predictable impact on the plain text

```
Encrypt: c  ← m  ⊕  k
 Modify: c’ ← c  ⊕  p
Decrypt: m’ ← c’ ⊕  k
```

  - What is the relation between *m* and *m’*?

Block Ciphers

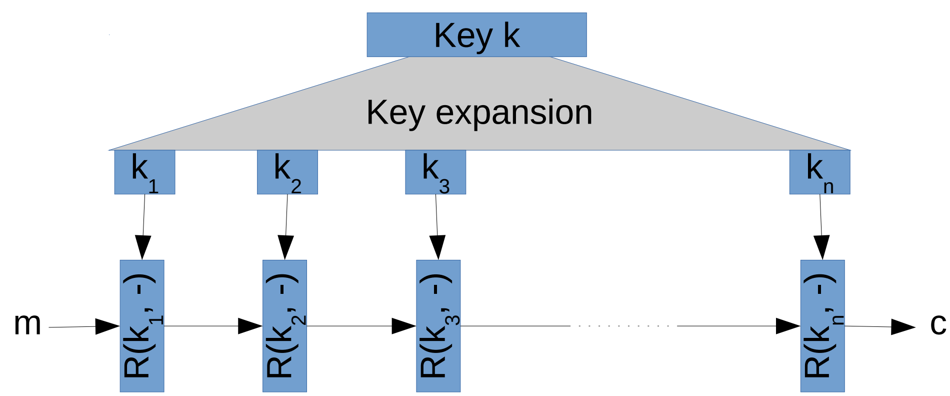
- Notable examples
  - 3DES: n = 64 bits, k = 168 bits
  - AES: n = 128 bits, k = 128, 192, 256 bits



Adaptation of: Dan Boneh, Cryptography I, Stanford.

Block Ciphers: Built by iteration

- R(k, m) is a round function
  - 3DES (n = 48)
  - AES (n = 10)



Adaptation of: Dan Boneh, Cryptography I, Stanford.

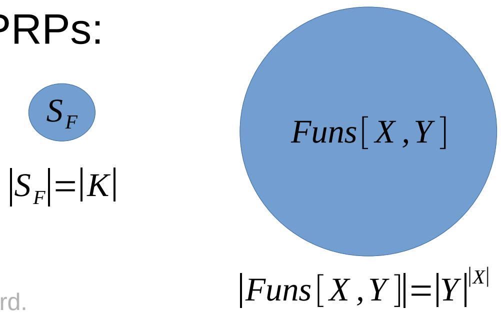
Abstracting BC: PRF and PRP

- Pseudo Random Function (PRF) defined over (K, X, Y):
  F: K × X → Y
  - We can evaluate F(k, x) efficiently
- Pseudo Random Permutation (PRP) defined over (K, X):
  E: K × X → X
  - We can evaluate E(k, x) efficiently
  - E(k, -) has an inverse
  - We have an efficient inversion algorithm D(k, x)
  - (All PRPs are PRFs.)

Secure PRF

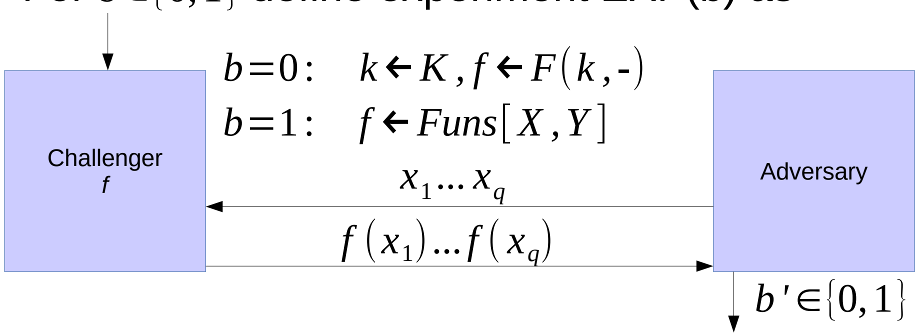
- Let $F : K \times X \to Y$ be a PRF
  - $Funs[X, Y]$ the set of all functions from X to Y
  - $S_F = \{F(k, -) : \forall k \in K\} \subseteq Funs[X, Y]$

  Intuitively
  - A PRF is secure if a random function in $Funs[X, Y]$ is indistinguishable from a random function in S_F
- Believed to be secure PRPs:
  - AES, 3DES, Blowfish



Secure PRF (def.)

- For $b \in \{0, 1\}$ define experiment EXP(b) as

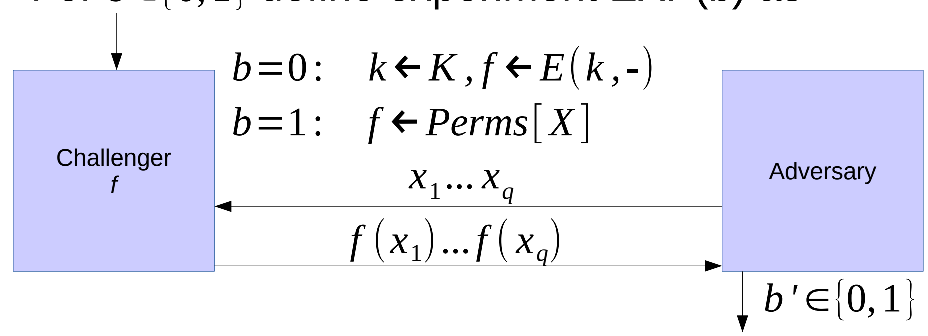


- Def: *F* is a secure PRF if for all eff. adversaries A $\mathrm{Adv}_{\mathrm{PRF}}[A, F]$ is negligible.

$$\mathrm{Adv}_{\mathrm{PRF}}[A, F] := \left|\Pr[\mathrm{EXP}(0)=1] - \Pr[\mathrm{EXP}(1)=1]\right|$$

Secure PRP (def.)

- For $b \in \{0, 1\}$ define experiment EXP(b) as



- Def: *E* is a secure PRP if for all eff. adversaries A $\mathrm{Adv}_{\mathrm{PRP}}[A, E]$ is negligible.

$$\mathrm{Adv}_{\mathrm{PRP}}[A, E] := \left|\Pr[\mathrm{EXP}(0)=1] - \Pr[\mathrm{EXP}(1)=1]\right|$$

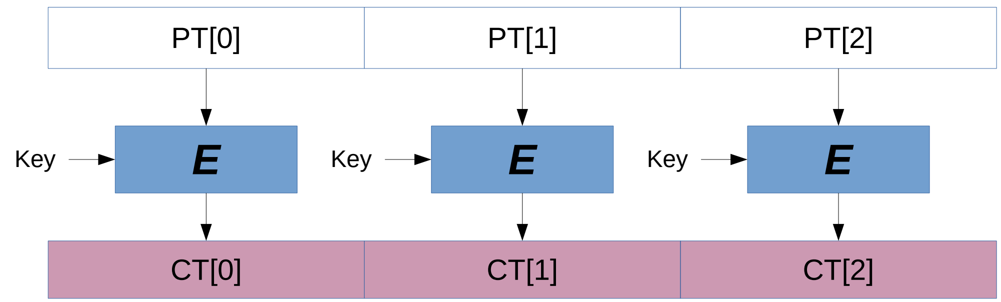
Block Ciphers: Modes of Operation

- Goal: How do we build a secure encryption from secure PRP (e.g. AES)
  - A PRP encrypts a single data block. How do we encrypt larger data?
- Semantic security (still for one-time key only)
  - Adversary's power: **observe one ciphertext**
  - Adversary's goal: **learn about plaintext**

Adaptation of: Dan Boneh, Cryptography I, Stanford.

MO: Electronic Code Book

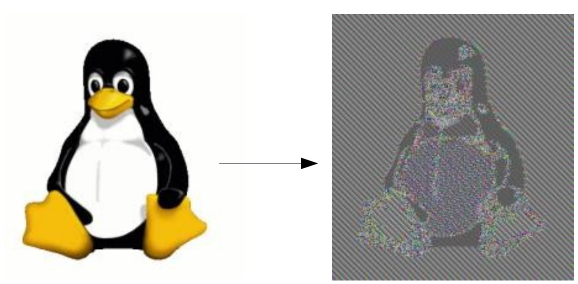
- "Solution" Electronic Code Book (ECB):
  - Split the data into blocks
    - if needed, extend the last block with padding bits
  - Independently encrypt each block



Adaptation of: Dan Boneh, Cryptography I, Stanford.

MO: Electronic Code Book

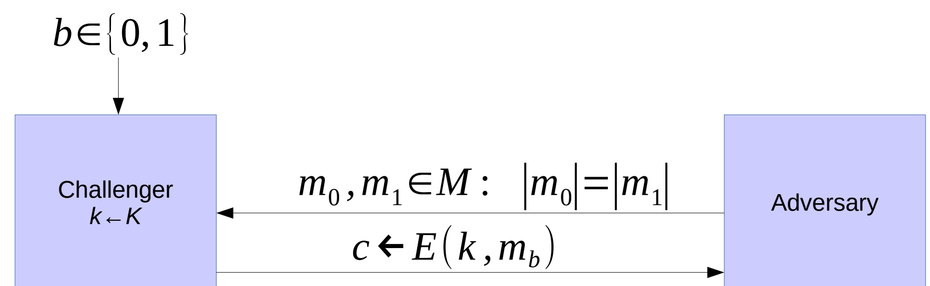
- Problem: If PT[0] == PT[1], then CT[0] == CT[1]
  - If two plaintext blocks are the same, so are the corresponding ciphertexts blocks



How does the Adversary win the semantic security game against ECB?

https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation#Electronic_Codebook_.28ECB.29

ECB is not semantically secure



- What should be m_0, m_1 ?
- What should the adversary output?

$?$

Adaptation of: Dan Boneh, Cryptography I, Stanford.

MO: Deterministic counter mode

- Deterministic counter from a pseudorandom function (PRF)

	PT[0]	PT[1]	PT[2]
$\oplus$	PRF(k, 0)	PRF(k, 1)	PRF(k, 2)
	CT[0]	CT[1]	CT[2]

- Creates a stream cipher from a PRF
- Secure (but only for encrypting a single message which may consists of multiple blocks)

Semantic security for many-time key

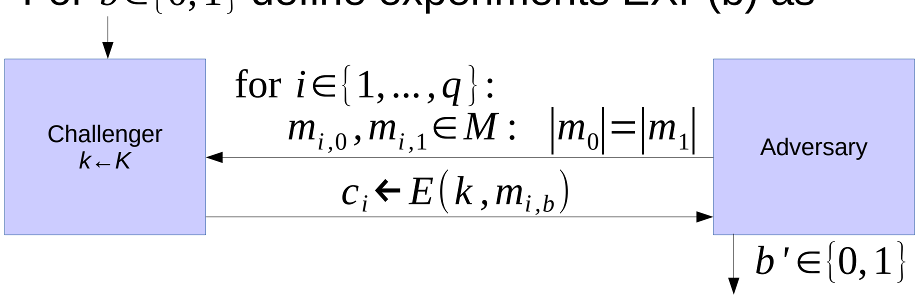
- Key is used more than once: adversary sees many CTs encrypted with the same key
- Adversary's power: **chosen-PT attack (CPA)**
  - Can obtain the encryption of any message of her choice
- Adversary's goal: **break semantic security**
  - Learn about the PT from the CT

Adaptation of: Dan Boneh, Cryptography I, Stanford.

Semantic security for CPA (def)

(for many-time key)

- Let $\zeta=(E,D)$ be a cipher defined over (K,M,C)
- For $b\in\{0,1\}$ define experiments EXP(b) as



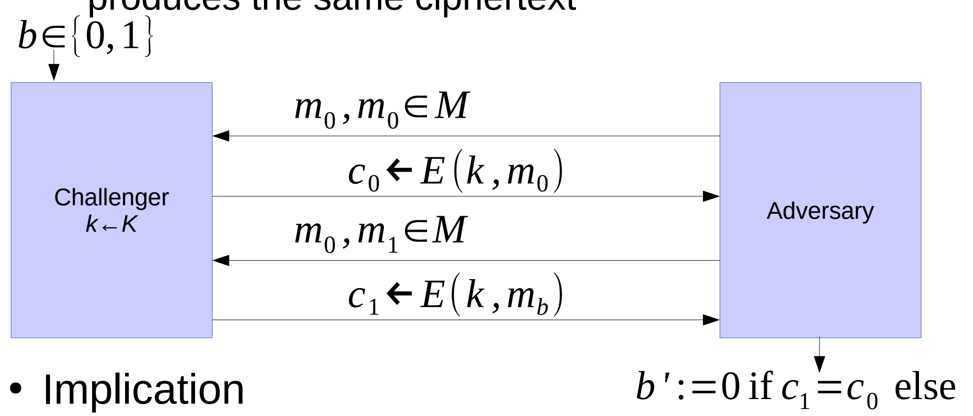
- Def: $\zeta=(E,D)$ is **semantically secure under CPA** if for all eff. adversaries A $\mathrm{Adv}_{\mathrm{CPA}}[A,\zeta]$ is negligible.

$$\mathrm{Adv}_{\mathrm{CPA}}[A,E]:=|\Pr[\mathrm{EXP}(0)=1]-\Pr[\mathrm{EXP}(1)=1]|$$

Adaptation of: Dan Boneh, Cryptography I, Stanford.

Ciphers insecure under CPA

- Suppose a cipher is deterministic
  - Given some message *m*, the cipher always produces the same ciphertext



- Implication
  - An attacker can learn that two encrypted elements (files, packets, ...) are the same

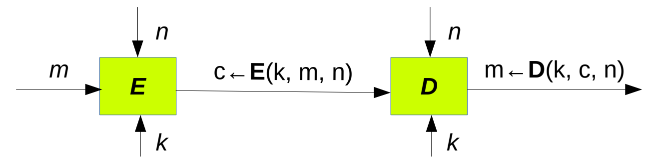
Adaptation of: Dan Boneh, Cryptography I, Stanford.

Ciphers insecure under CPA

- If a key is to be used multiple times, the encryption should be **non-deterministic**:
  - Encrypting the same PT twice, must produce different CTs
- Solutions
  - Randomized encryption
  - Nonce-based encryption

Non-deterministic encryption

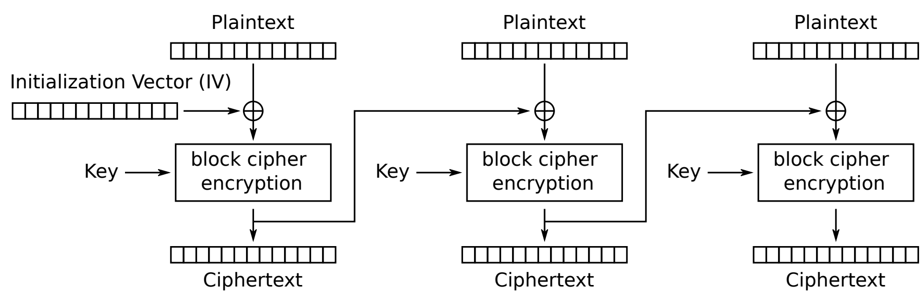
- Nonce ***n***: a value that changes from message to message
  - Pair *(key, n)* **must never repeat**
- Method 1: Nonce is a random value (AES-CBC)
- Method 2: Nonce is a counter (AES-CTR)



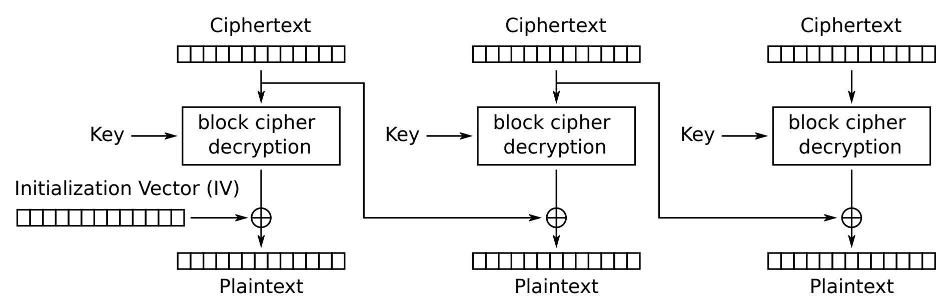
- CPA system should be secure even when the adversary chooses nonces

Modes of Operation: CBC

- Randomize the encryption with an initialization vector (IV)
  - Sent unencrypted
  - Must generate new random IV for every message: pair ***(key, IV)*** must never repeat
  - IV must be unpredictable
- Forces encryption to be sequential
  - Decryption may be parallelized



Cipher Block Chaining (CBC) mode encryption

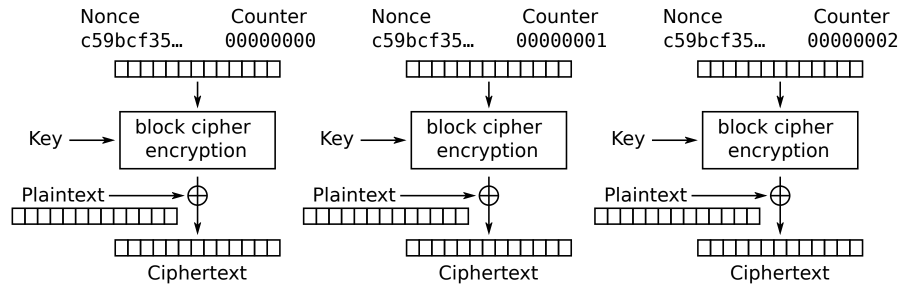


Cipher Block Chaining (CBC) mode decryption

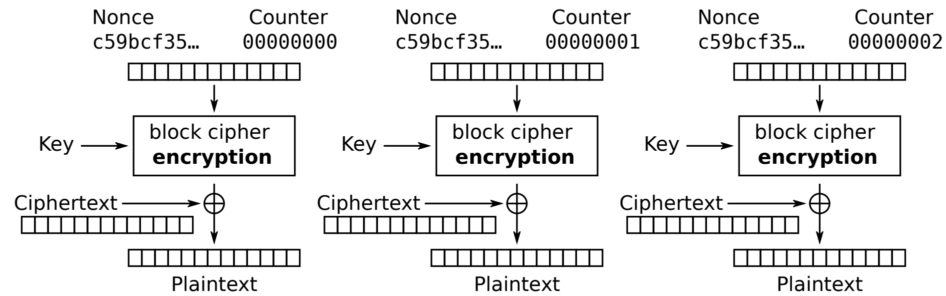
https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

Counter Mode

- The random element is a counter
  - Or a combination of a random IV and a counter
  - The combination must not repeat for the lifetime of the key
- Encryption and decryption can be done in parallel
- In effect, creates a stream cipher out of a block cipher



Counter (CTR) mode encryption



Counter (CTR) mode decryption

https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

Summary

- Two security notions
  - Semantic security against one-time CPA
  - Semantic security against many-time CPA
- Only covered secrecy against passive attackers
  - Adversaries can see, but not modify cipher text
  - We'll cover integrity next week

Power / Goal	One-time key	Many-time key (CPA)
Semantic security	**Stream-ciphers Deterministic CTR-mode**	**Rand CBC Rand CTR-mode**