CSIT 495/595 - Introduction to Cryptography
Private Key Encryption

Bharath K. Samanthula
Department of Computer Science
Montclair State University

Outline

- Computation Security Definition
- Pseudorandom Generators
- Stream Ciphers
- Block Ciphers

Perfect Secrecy Vs. Computation Security

- Perfect secrecy requires absolutely no information about the plaintext is leaked to an adversary with unlimited computational power
- Computational secrecy allows to leak only a tiny amount of information to the adversary with bounded computational power
- For many applications, the notion of computational secrecy is sufficient
- Example: an encryption scheme that leaks information with probability at most 2^{-80} to attackers investing upto 200 years of computational effort on the world's fastest super computer is adequate for any real-world application

Computational Security: Basic Idea

- Perfect security[1] is hard to achieve in real-world applications
- Modern cryptography is based on the notion of Computational Secrecy
- Key question: Can we use a short key (say 128 or 192 bits long) and still able to encrypt many long messages (say file sizes of gigabytes or terabytes) with reasonable security guarantees?

[1] Also called as information-theoretic security

Computational Security: Relaxations

1. Security is only guaranteed against efficient adversaries that run for certain amount of time
2. Adversaries can potentially succeed, but with very very small probability
3. Key questions:
   - How do we define efficient adversaries?
   - How can we quantify very small probability?

Computational Security: Necessity of Relaxations

- The two relaxations are needed for developing practical encryption schemes
- Suppose key space is much smaller than message space
- Given a ciphertext c, the attacker can try to decrypt it using all possible k values
- Since $|\mathcal{K}| < |\mathcal{M}|$, such a brute force attack leaks information about the message that corresponds to c

Computational Security: Necessity of Relaxations

- Say the attacker carries out a known-plaintext attack, i.e., knows $m_1, \ldots, m_\ell$ and their encrypted values $c_1, \ldots, c_\ell$
- The attacker can find out the key k, such that $m_i = \mathrm{Dec}_k(m_i)$, for $1 \leq i \leq \ell$
- After this, he/she can decrypt any ciphertext with probability 1
- The attacker can perform the above attack in time linear with $|\mathcal{K}|$
- Therefore, for using smaller key sizes, we need to restrict the computational power of the adversary (with a small success probability)

Computational Security: A Concrete Approach

- A scheme is said to (t, ϵ) secure if:
  1. adversary can run for at most time t (seconds or CPU cycles)
  2. the probability to succeed is at most ϵ
- In modern private-key encryption schemes:
  - For a key of size n, an adversary running for time t succeeds in breaking the system with probability at most $ct/2^n$
- Example: Let $c = 1$ and $n = 60$. On a 4 GHz PC, 2^{60} CPU cycles require about 9 years. However, a supercomputer with $2 * 10^{16}$ FLOPS, 2^{60} operations can be done in about one minute

Computational Security: A Concrete Approach

Security Claim: No adversary running for 5 years can break a scheme with probability better than ϵ
Limitations:

- What kind of computing power does the adversary uses?
  - Desktop PC
  - supercomputer
  - network of computer nodes
- Does the security claims take Moore's Law into account
- What will be the success probability if the adversary runs for 10 years?

Computational Security: An Asymptotic Approach

- A security parameter (denoted by n) is used to parameterize cryptographic schemes and participating parties (honest parties + attacker)
- n corresponds to the key length: selected by the honest parties during key generation algorithm
- n is assumed to be known to the attacker
- Realizing relaxations:
  1. *efficient adversaries* - randomized (or probabilistic) algorithms running in time polynomial in n
  2. *very small probability* - an inverse polynomial in n (consider to be negligible)

Computational Security: An Asymptotic Approach

Example 1:

- Suppose an adversary running for n^3 minutes succeed with probability $2^{40} * 2^{-n}$
- Scenario 1: For $n \leq 40$, adversary running for 40^3 minutes can break the system with probability 1 (is this acceptable?)
- Scenario 2: For $n = 50$, adversary running for 50^3 minutes can break the system with probability 1/1024 (is this acceptable?)
- Scenario 3: When $n = 500$, it takes 200 years for the adversary to break the system with probability 2^{-460} (is this acceptable?)

Computational Security: An Asymptotic Approach

Example 2:

- Let honest parties run $10^6 * n^2$ cycles and adversaries running for $10^8 * n^4$ cycles can succeed with probability at most $2^{-n/2}$
- Scenario 1: say $n = 80$ and all parties use 2 GHz PCs. Honest parties run for 3.2 seconds. Attacker running for 3 weeks can succeed with probability only 2^{-40}
- Scenario 2: say $n = 160$ and all parties upgraded to 8 GHz. Honest parties still take 3.2 seconds. Attacker takes 13 weeks to achieve a success probability of 2^{-80}. Adversary's job became harder here

Computational Security: Formal Definition

- Suppose m_0 and m_1 be two messages of same length

A private-key encryption scheme $\Pi = \langle \text{Gen, Enc, Dec} \rangle$ is computationally secure if for every polynomial-time algorithm $\mathcal{A}$, n-bit key, there exists a negligible function ϵ, such that

$$|Pr[Out(\mathsf{A}_\Pi(n, 0)) = 1] - Pr[Out(\mathsf{A}_\Pi(n, 1)) = 1]| \leq \epsilon(n)$$

where $Out(\mathsf{A}_\Pi(n, b)))$ denote the output bit of the experiment being run to find out that the encrypted message is m_b and $b \in [0, 1]$

Encryption and Plaintext Length

- Encryption scheme should consider whether it can leak plaintext length
- Leaking plaintext can become problematic:
  - Numeric/categorical data: salary, yes/no, etc.
  - Database searches: user's query and number of records returned can leak some information
- Necessary steps need to be taken to avoid leaking plaintext length, such as padding all messages to some predetermined length

Pseudorandom Generator

- A pseudorandom generator G is an efficient, deterministic algorithm:
  - Input: a short, uniform string, called the seed
  - Ouput: a longer, uniform looking string
- Cryptographic schemes are impossible without pseudorandom generators
- They are used often in
  - generating keys
  - initialization vectors
  - public-key cryptosystems
  - other cryptographic algorithms

Pseudorandom Generator: Seeds and its Length

- Seed must be chosen uniformly and be kept secret from adversaries
- Well-known ways to select a seed:
  - delays between network events, hard-disk access times, thermal noise, etc
- To avoid brute-force attacks,
  - seed must be long enough
  - One approach is to set the length of the seed equal to the security parameter

Security Under Chosen-Plaintext Attacks (CPA)

- CPA is a stronger notion of security compared to Ciphertext-only and known-plaintext attacks
- Have been used in the past to break military encryption schemes
- Example 1: During world war II, the British placed mines at chosen locations, which were identified by Germans who have encrypted them and send back to their headquarters. The cryptanalysts at Bletchley park used these encrypted messages to break the German encryption scheme
- Example 2: In May 1942, US Navy cryptanalysis used the CPA-attack to crack messages encrypted by Japanese encryption scheme

Security Under Chosen-Plaintext Attacks (CPA)

- Key Idea: encryption of a plaintext should completely yield different ciphertexts
- Nowadays, security against CPA is the minimal notion of security an encryption scheme should satisfy
- Is one-time pad encryption scheme considered CPA-secure?

Types of Private-Key Encryption

- Stream Ciphers
- Block Ciphers

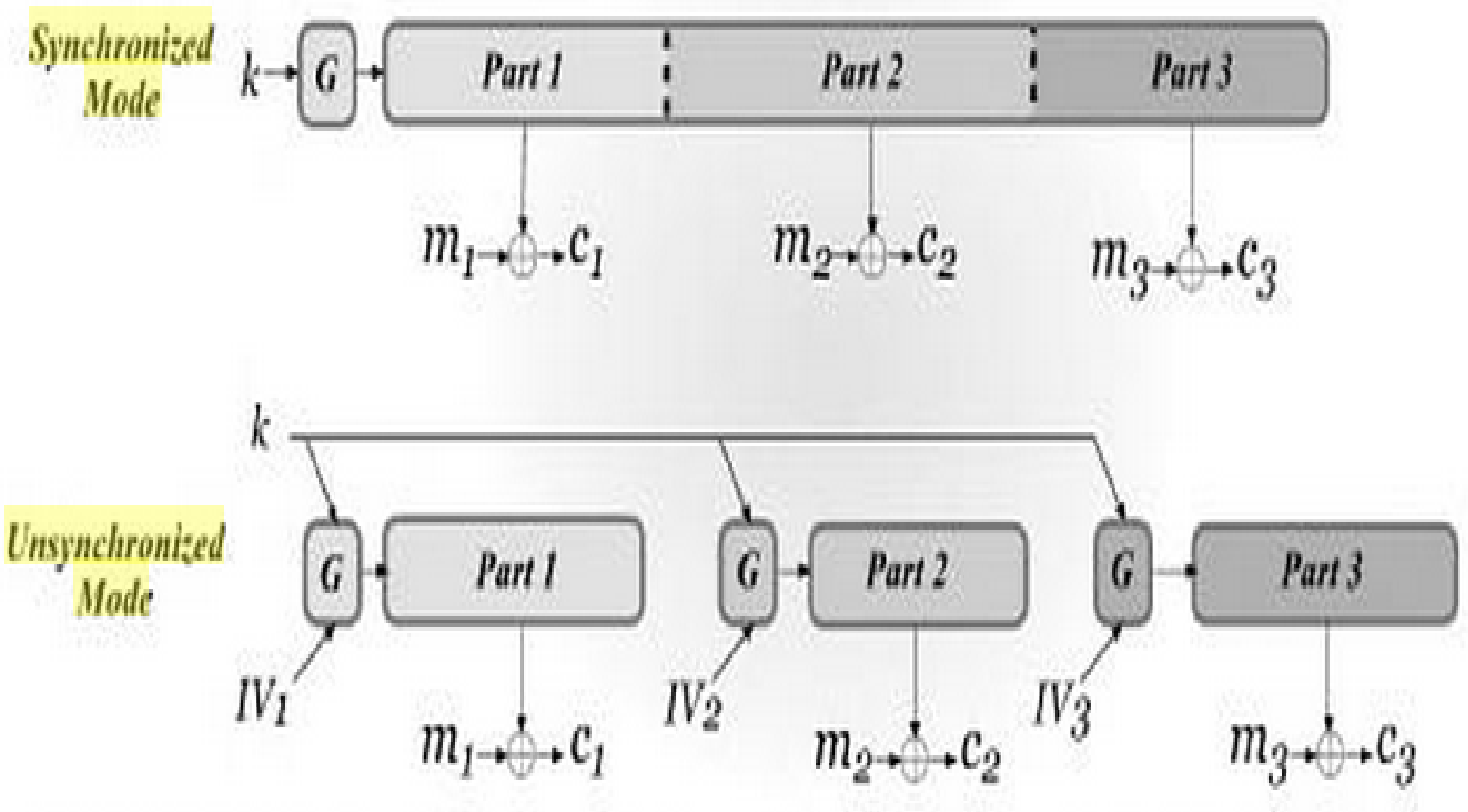
Stream Ciphers

- Used for encrypting streamed data
- Encryption/decryption is done one bit at a time
- Usually faster and have a lower hardware complexity
- Used for cell phones or small embedded devices (e.g., A5/1 stream cipher is used as a standard in GSM mobile phones for voice encryption)
- Basic Idea:
  - Keystream: a pseudorandom sequence of bits, $s_1, s_2, \ldots, s_\ell$
  - Encryption: $c_i \leftarrow m_i + s_i \bmod 2 = m_i \oplus s_i$
  - Decryption: $m_i \leftarrow c_i + s_i \bmod 2 = c_i \oplus s_i$

Stream Ciphers: Modes of Operation

- Synchronized mode
- Unsynchronized mode

Stream Ciphers: Modes of Operation



where IV denotes the initialization vector or nonce

Block Ciphers

- Encryption is done block by block
- Each block typically consists of 64-bit or more
- Encrypts each block with the same key
- Some well-known block ciphers - DES, AES (more details on these later)
- Basic Idea:
  - Divide the message into blocks (each of equal size)
  - If text in a block is less than its size, use padding
  - Choose the mode of operation
  - Apply the mode of operation on the blocks

Block Ciphers: Modes of Operation

- Electronic Code Book (ECB)
- Cipher Block Chaining (CBC)
- Output Feedback (OFB)
- Counter (CTR)

Block Ciphers: Evaluation Criteria

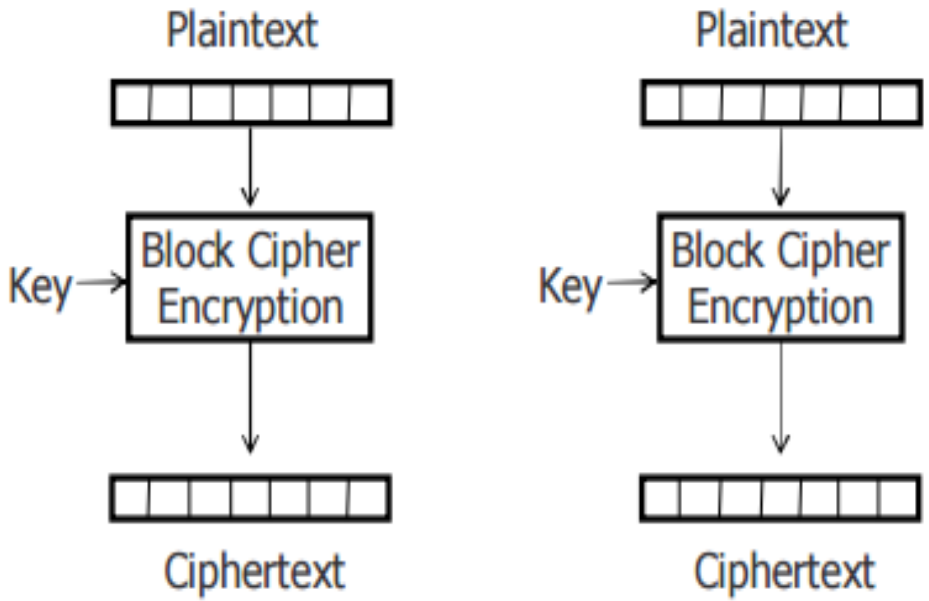
- Identical messages
  - under which conditions ciphertext of two identical messages are the same
- Chaining dependencies
  - how adjacent plaintext blocks affect encryption of a plaintext block
- Error propagation
  - resistance to channel noise
- Efficiency
  - preprocessing
  - parallelization: random access

Slides adopted from *Modes of operation*, Murat Kantarcioglu

Electronic Code Book (ECB) Mode 1

- Direct use of the block cipher/ pseudorandom functions
- Apply block cipher to each plaintext block
- Used primarily to transmit encrypted keys
- Never use it for general-purpose encryption, such as for a file or a image (why??)

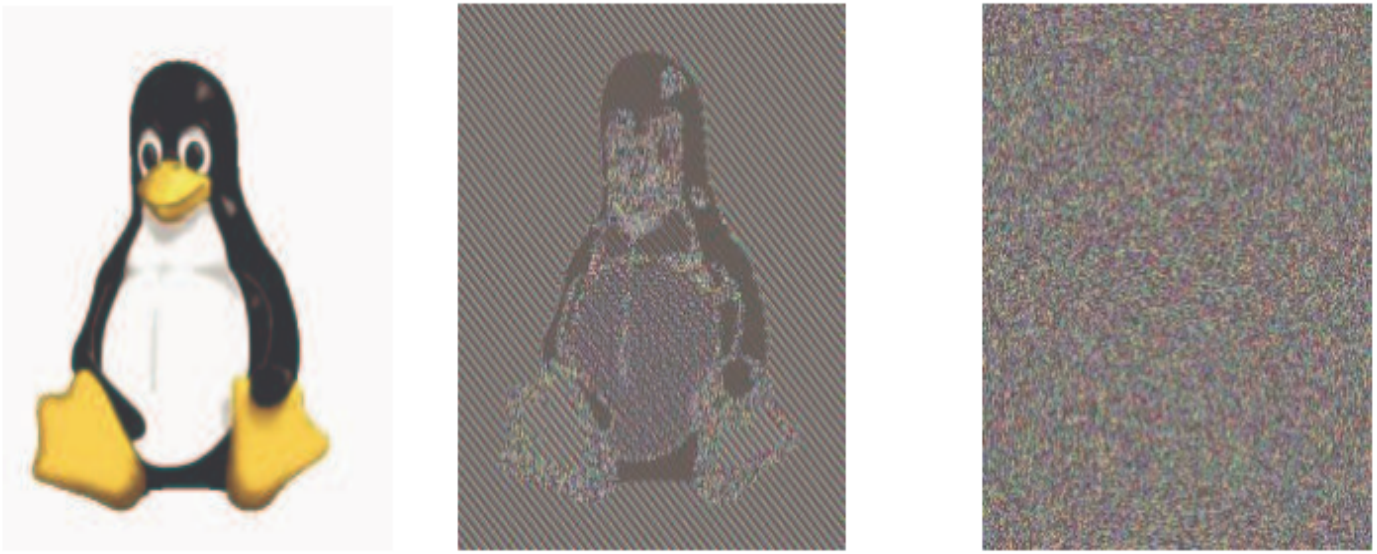
Electronic Code Book (ECB) Mode 2



- Each block encrypted independently
- Identical plaintexts encrypted similarly
- No chaining, no error propagation

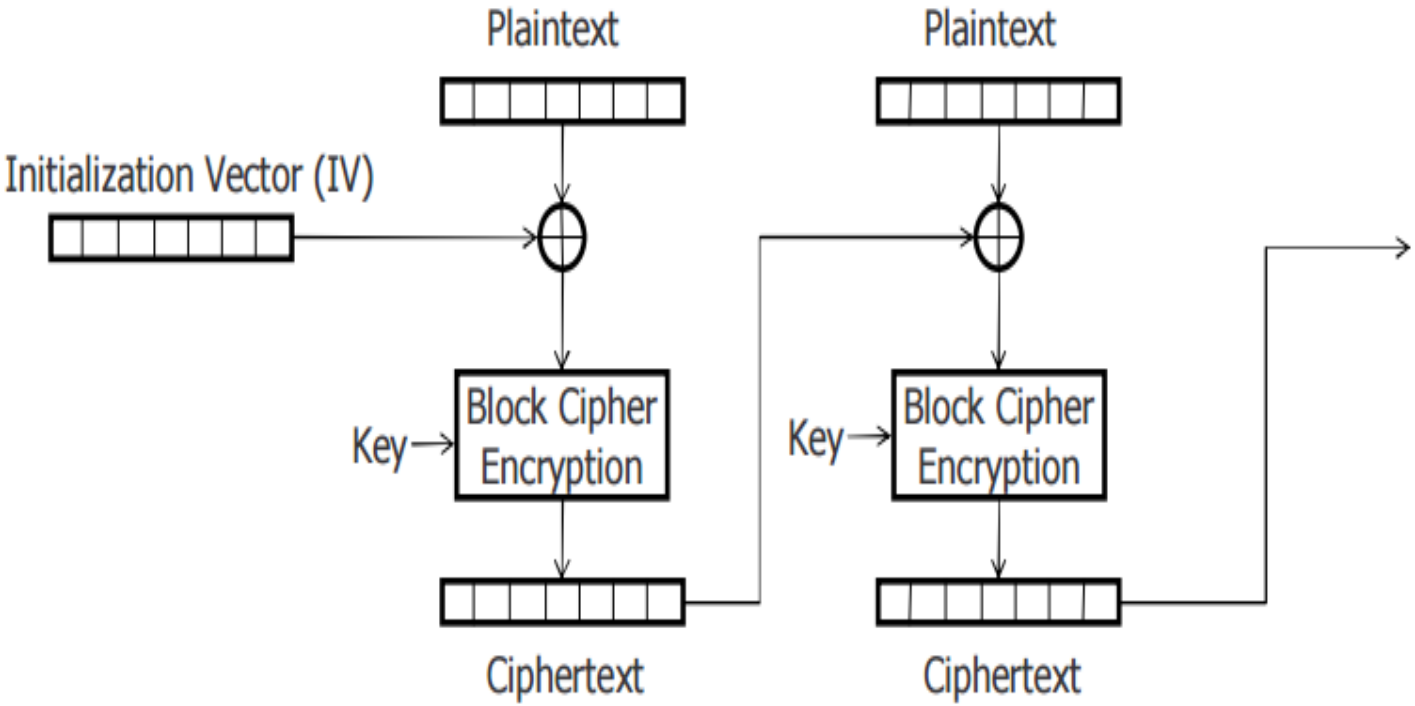
Electronic Code Book (ECB) Mode 3

- Does not hide data patterns, unsuitable for long messages
  - Wiki example: pixel map using ECB



- Susceptible to replay attacks
  - Example: a wired transfer transaction can be replayed by re-sending the original message)

Cipher Block Chaining (CBC) Mode

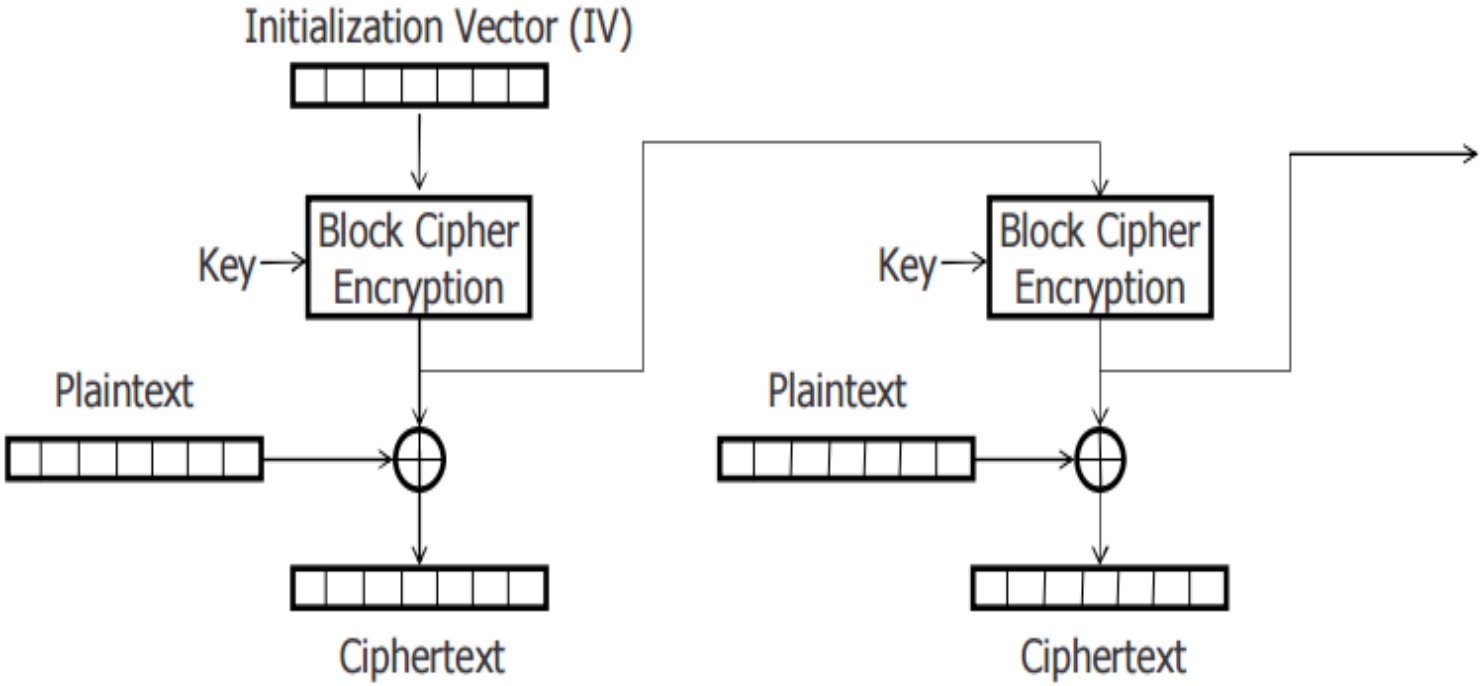


- Allows random access to ciphertext
- Decryption is parallelizable
  - Plaintext block x_j requires ciphertext blocks c_j and c_{j-1}

Cipher Block Chaining (CBC) Mode

- Identical messages: changing IV or the first plaintext block results in different ciphertext
- Chaining: Ciphertext block c_j depends on x_j and all preceding plaintext blocks (dependency contained in c_{j-1})
- Error propagation: Single bit error on c_j may flip the corresponding bit on x_{j+1}, but changes x_j significantly.
- IV need not be secret, but its integrity should be protected

Output Feedback (OFB) Mode

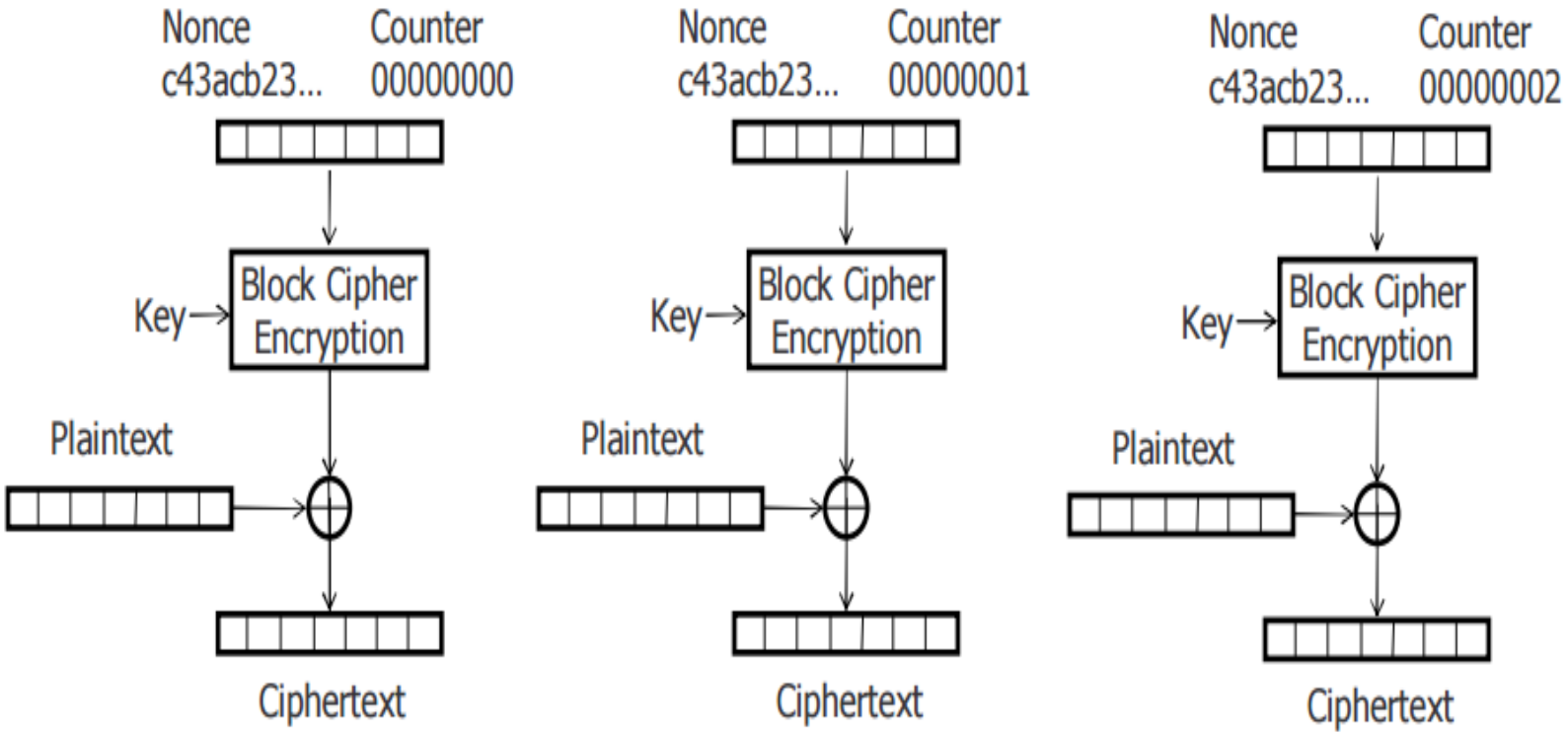


- Preprocessing possible (keep enc/decrypting previous output block)
- No random access, not parallelizable

Output Feedback (OFB) Mode

- Identical messages: same as CBC
- No chaining dependencies
- Error propagation: Single bit error on c_j may only affect the corresponding bit of x_j
- IV need not be secret, but should be changed if a previously used key is to be used again

Counter (CTR) Mode



- Preprocessing possible (inc/decrement and enc/decrypt counter)
- Allows random access

Counter (CTR) Mode

- Both encryption & decryption are parallelizable
  - Encrypted counter is sufficient to enc/decrypt
- Identical messages: changing nonce results in different ciphertext
- No chaining dependencies
- No error propagation
- Nonce should be random, and should be changed if a previously used key is to be used again

Effect of Modes of operation

- Choice of encryption mode affects
  - Encryption/decryption speed
  - Security against active adversaries (bit flips)
  - Security against passive adversaries (ECB)
  - Error propagation

Summary

- Computational Security
- Pseudorandom Generators
- Stream Ciphers and its modes of operation
- Block Ciphers and its modes of operation

Useful References

- Chapter 3, Introduction to Modern Cryptography by Jonathan Katz and Yehuda Lindell, 2nd Edition, CRC Press, 2015.
- `http://arxiv.org/ftp/arxiv/papers/1405/1405.0398.pdf`
- `http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf`