Cristina Nita-Rotaru

CY2550: Foundations of Cybersecurity Section 03

Crypto Module: Computational security. Block ciphers. Public-key cryptography.

Outline

- Computational security
- Block cipher
- Public-key cryptography

Computational security

Computationally-bounded adversaries

Restriction: **Eve is computationally-bounded**

We will construct schemes that in **principle can be broken** if the adversary has a huge computing power or is extremely lucky.

- E.g., break the scheme by enumerating all possible secret keys. ("**brute force attack**")
- E.g., break the scheme by guessing the secret key.

Goal: cannot be broken with reasonable computing power with reasonable probability.

Towards computational security

- Computational security uses two relaxations:
  1. **Security is preserved only against computationally bounded adversaries**
     - Limits on computational power and storage
     - Polynomial-time adversaries
  2. **Adversaries may successfully crack encryption with a very small probability**
     - So small that (we hope) it becomes negligible
     - Example negligible probability: $\frac{1}{2^{128}}$
- Computational assumptions are part of the threat model

Eavesdropping security

- Ciphertext INDistinguishability under an EAVesdropping attacker (IND-EAV)

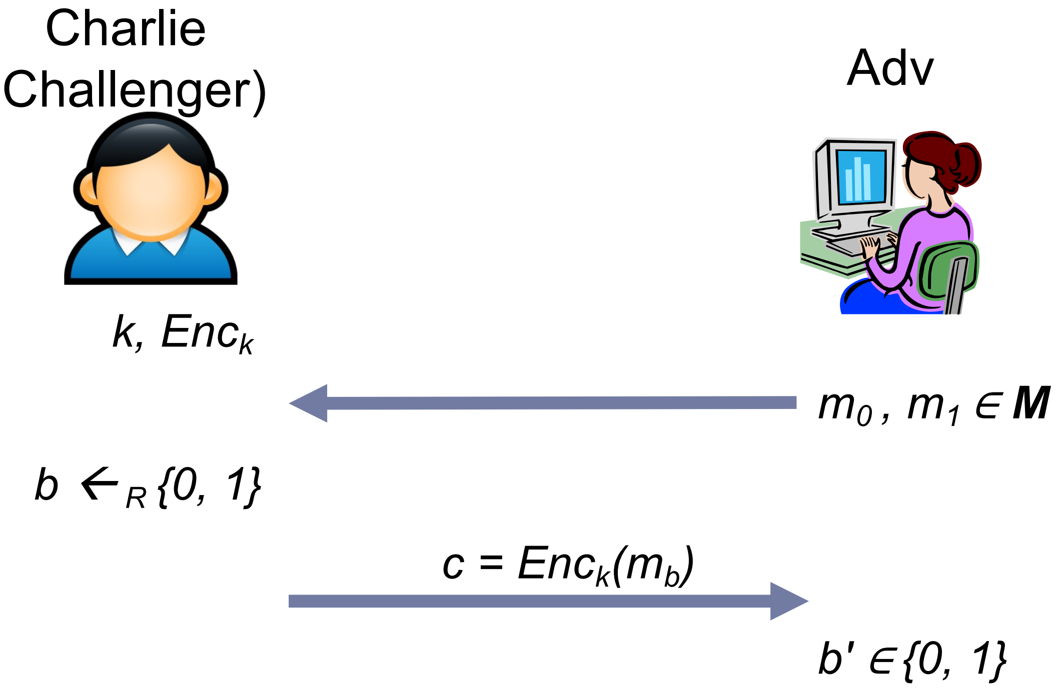
Round 1: Charlie chooses k and encryption algo

Round 2: Adv chooses two plaintext messages

Round 3: Charlie chooses a random binary number

Round 4: Charlie encrypts the corresponding message

Round 5: Adv guesses a the value of *b*



Charlie (Challenger)

Adv

$k,\ Enc_k$

$m_0\ ,\ m_1 \in \mathbf{M}$

$b \leftarrow_R \{0,\ 1\}$

$c = Enc_k(m_b)$

$b' \in \{0,\ 1\}$

Adversary wins if $b = b'$

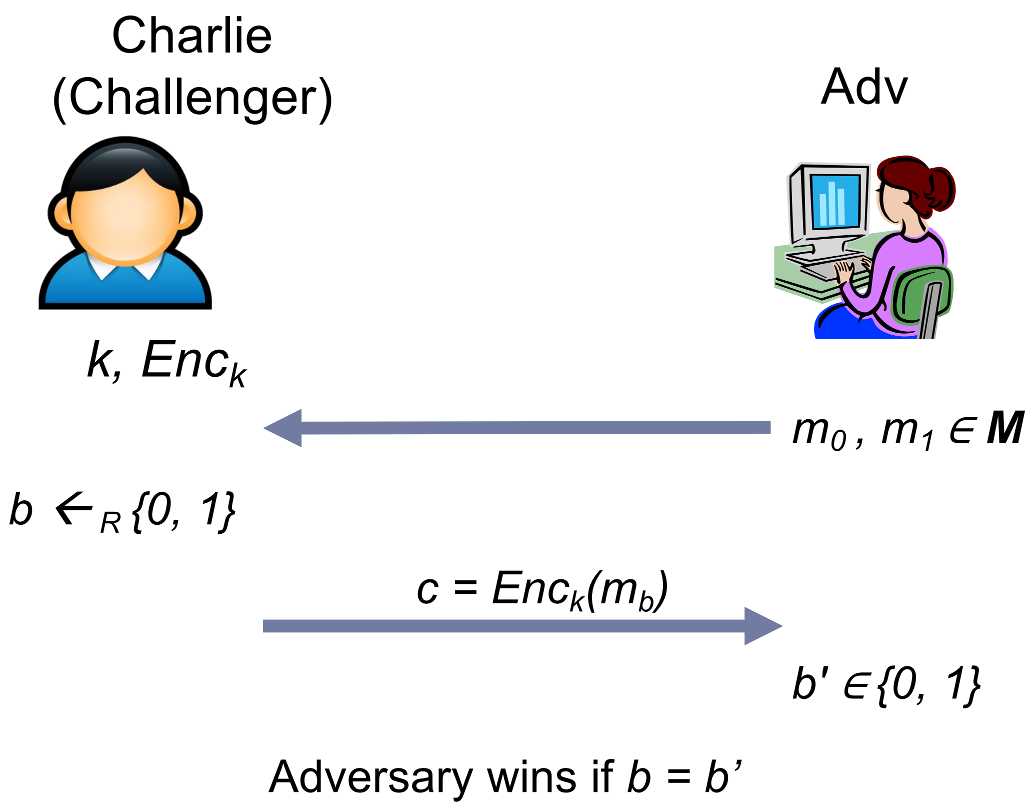
Examples

- If E is a perfectly secure algorithm (e.g., OTP), what is the probability that $b = b'$?

$P(Adv\ wins)$ = ½ SECURE

- If E is a Caesar shift, what is the probability that $b = b'$?

$P(Adv\ wins) = 1$ NOT SECURE



Charlie (Challenger)

Adv

k, Enc_k

$m_0, m_1 \in \mathbf{M}$

$b \leftarrow_R \{0, 1\}$

$c = Enc_k(m_b)$

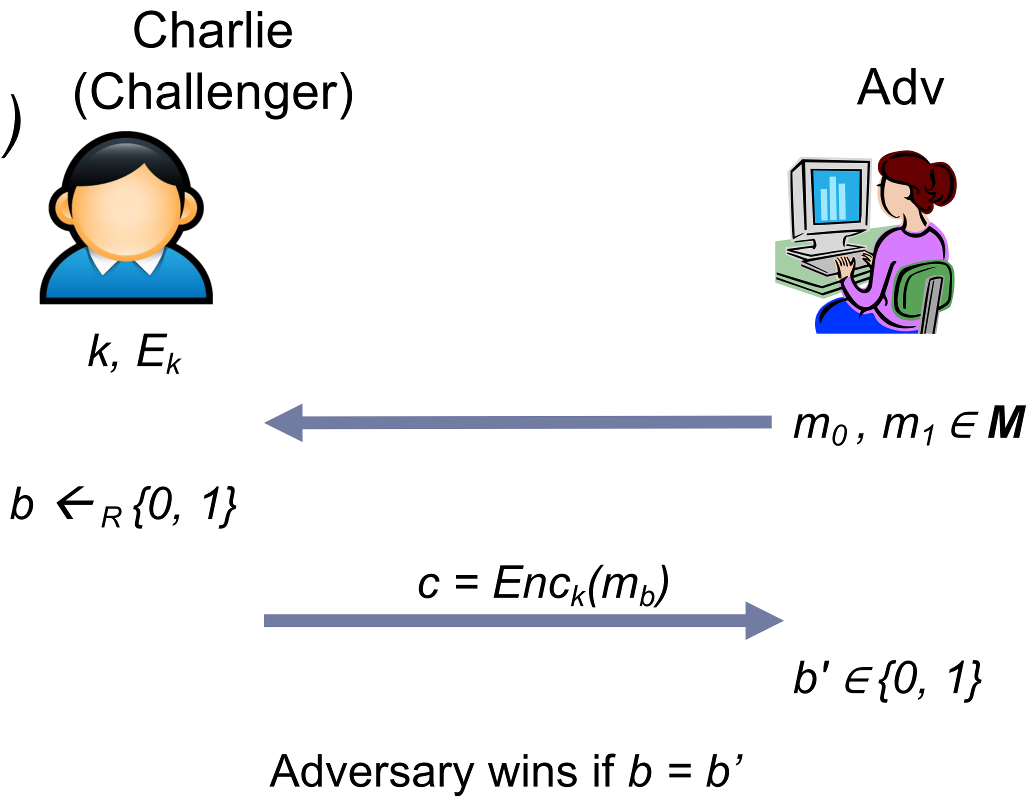
$b' \in \{0, 1\}$

Adversary wins if $b = b'$

Computational secure IND-EAV

- If *Enc* is a computationally secure algorithm, what is the probability that *b* = *b*'?

P(Adv wins) = ½ + *negligible(|k|)*



Charlie (Challenger)

Adv

$k,\ E_k$

$m_0,\ m_1 \in \boldsymbol{M}$

$b \leftarrow_R \{0,\ 1\}$

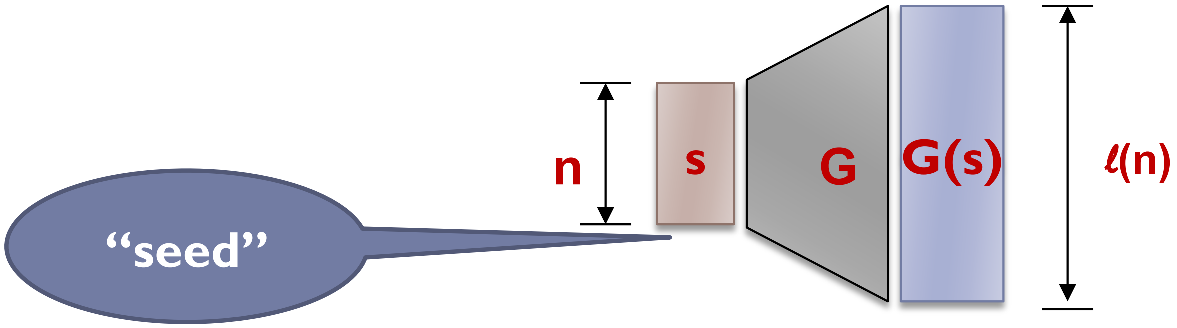
$c = Enc_k(m_b)$

$b' \in \{0,\ 1\}$

Adversary wins if *b* = *b*'

How to achieve this?

Pseudorandom generators (PRG)



A pseudorandom generator is a deterministic algorithm

$G : \{0,1\}^n \rightarrow \{0,1\}^{\ell(n)}$.

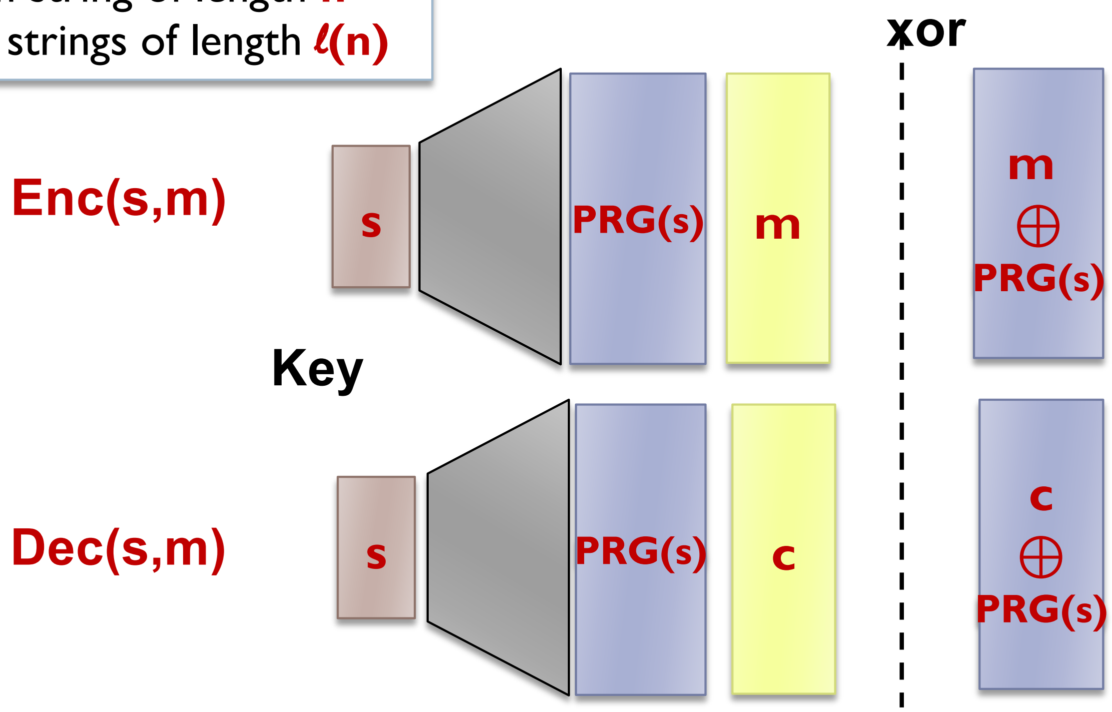
- **Output length:** $\ell(n)$ for all s with $|s| = n$ we have $|G(s)| = \ell(n)$.
- **Stretch:** $\ell(n) - n$

Goal (imprecise): If s chosen randomly from $\{0,1\}^n$, then $G(s)$ "looks" like it was chosen randomly from $\{0,1\}^{\ell(n)}$.

Using a PRG to build efficient OTP

Use PRGs to "shorten" the key in the one time pad

Key: random string of length **n**
Plaintexts: strings of length **ℓ(n)**



Enc(s,m)

s

PRG(s)

m

xor

m $\oplus$ PRG(s)

Key

Dec(s,m)

s

PRG(s)

c

c $\oplus$ PRG(s)

STREAM CIPHER

Examples: RC4, Salsa20

IND-EAV secure one-time pad

Adversarial capability

- Ciphertext-only attack: Perfect security, IND-EAV
  - Adversary observes ciphertext(s)
  - Infer information about plaintext
- Chosen-plaintext attack: IND-CPA
  - Adversary can encrypt messages of his choice
- Chosen-ciphertext attack: IND-CCA
  - Adversary can decrypt ciphertexts of its choice
  - Learn plaintext information on other ciphertext

IND-CPA security

- Ciphertext Indistinguishability under Chosen-Plaintext Attack (CPA)
- Adv can encrypt messages of its choice

Round 1: Charlie chooses k and encryption algo

Round 2: Adv can encrypt messages

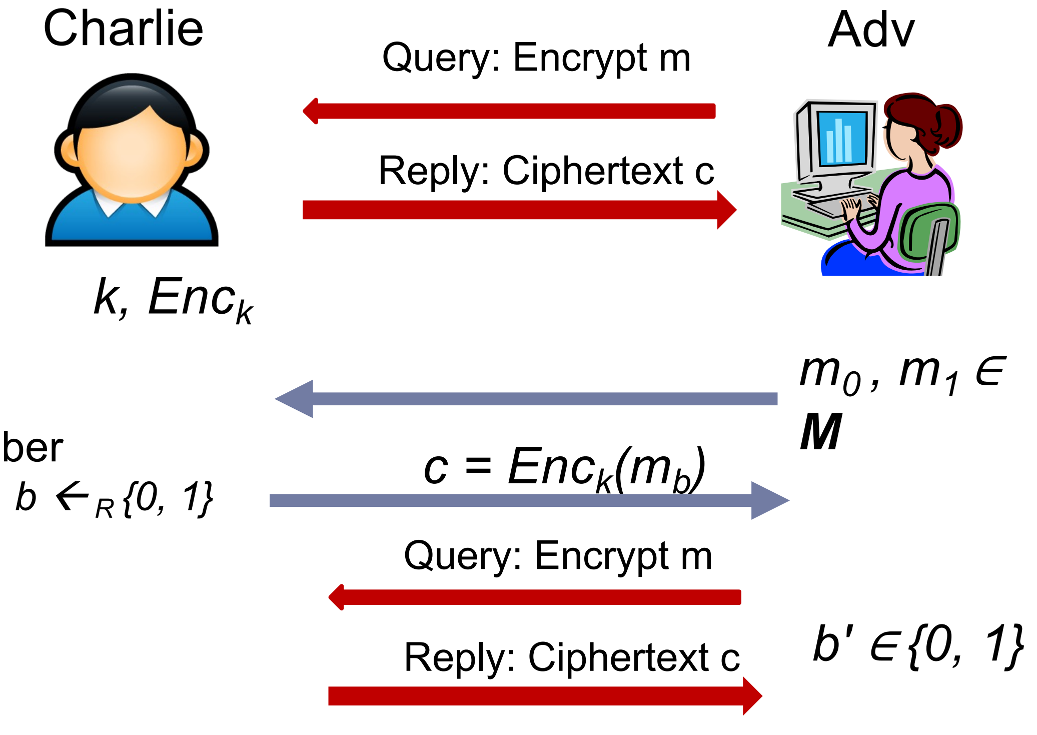
Round 3: Adv chooses two plaintext messages

Round 4: Charlie chooses a random binary number

Round 5: Charlie encrypts the corresponding message

Round 6: Adv can encrypt messages

Round 7: Adv guesses the value of *b*



Charlie

Adv

Query: Encrypt m

Reply: Ciphertext c

k, Enc_k

m_0, $m_1 \in \boldsymbol{M}$

$b \leftarrow_R \{0, 1\}$

$c = Enc_k(m_b)$

Query: Encrypt m

Reply: Ciphertext c

$b' \in \{0, 1\}$

Adversary wins if *b* = *b’*

IND-CPA Security

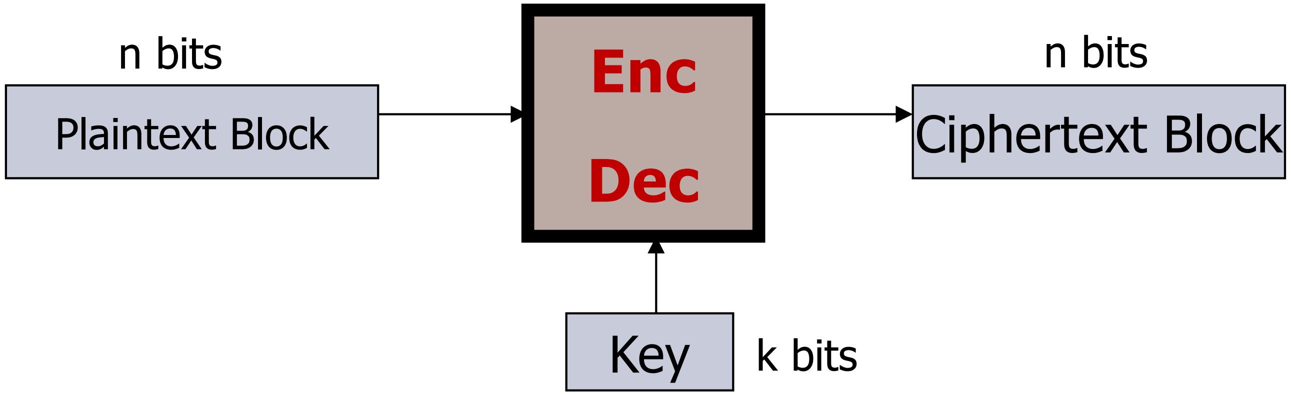
- Adversary can encrypt messages of his choice
  - Including m_0 , m_1
- Adversary can encrypt any message before and after seeing the ciphertext c
- CPA adversary is stronger than EAV
- A scheme secure under CPA is also secure under EAV
- But not the other way around!
  - The One-time pad is IND-EAV secure, but not IND-CPA secure
  - IND-CPA is strictly stronger than IND-EAV (for symmetric-key encryption)
- How to design IND-CPA secure ciphers?

Block ciphers

Symmetric key cryptography

- Algorithms that use a single key for encryption and decryption
  - i.e. the algorithm is reversible
  - $\forall k \ \forall m \quad Dec_k(Enc_k(m)) = m$ where m is a message, k is a key, and Dec_k and Enc_k are decryption and encryption using k
- Historic examples:
  - Caeser shift, mono and polyalphabetic substitution, OTP
- Modern examples (block ciphers):
  - DES, 3DES, RC4, Blowfish, Twofish, AES
  - **Warning**: many of these methods are known to be vulnerable

Block ciphers



Canonical examples:

1. DES: n=64 bits, k=56 bits
2. Triple DES: n=64 bits, k=168 bits
3. AES: n=128 bits, k=128, 192, 256 bits

Desired properties:

1. Change one bit of plaintext completely changes ciphertext
2. Good mixing properties
3. Ciphertext looks random

The Data Encryption Standard (DES)

- Early 1970s: Horst Feistel designs Lucifer at IBM

  key-len = 128 bits ; block-len = 128 bits

- 1973: NBS asks for block cipher proposals.

  IBM submits variant of Lucifer.

- 1976: NBS adopts DES as a federal standard

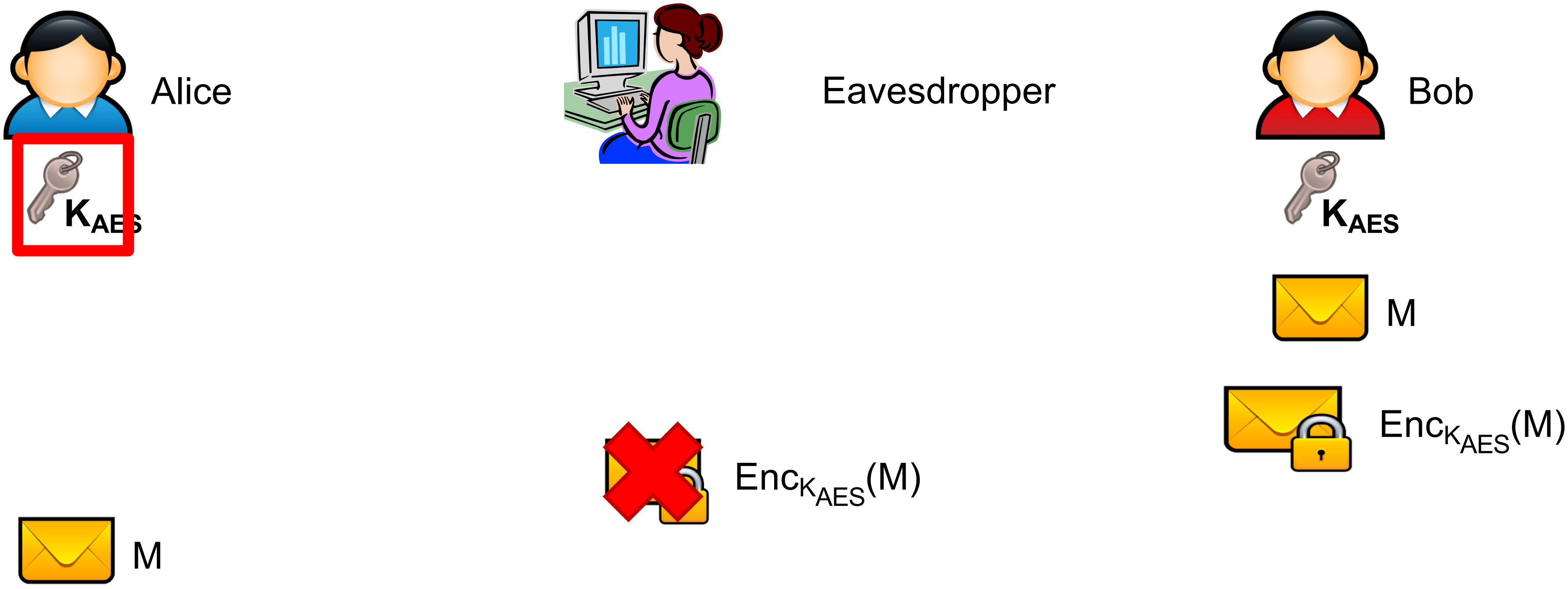
  key-len = 56 bits ; block-len = 64 bits

- 1997: DES broken by exhaustive search (short keys)
- 2000: NIST adopts Rijndael as AES to replace DES

Advanced Encryption Standard (AES)

- In 1997, NIST made a formal call for algorithms stipulating that the AES would specify an unclassified, publicly disclosed encryption algorithm, available royalty-free, worldwide
- Goal: replace DES for both government and private-sector encryption.
- The algorithm must implement symmetric key cryptography as a block cipher and (at a minimum) support block sizes of 128-bits and key sizes of 128-, 192-, and 256-bits.
- In 1998, NIST selected 15 AES candidate algorithms.
- In 2000, NIST selected Rijndael (invented by Joan Daemen and Vincent Rijmen) as the AES
- Designed to be efficient in both hardware and software

AES example



Alice

Eavesdropper

Bob

K_{AES}

K_{AES}

M

$Enc_{K_{AES}}(M)$

$Enc_{K_{AES}}(M)$

M

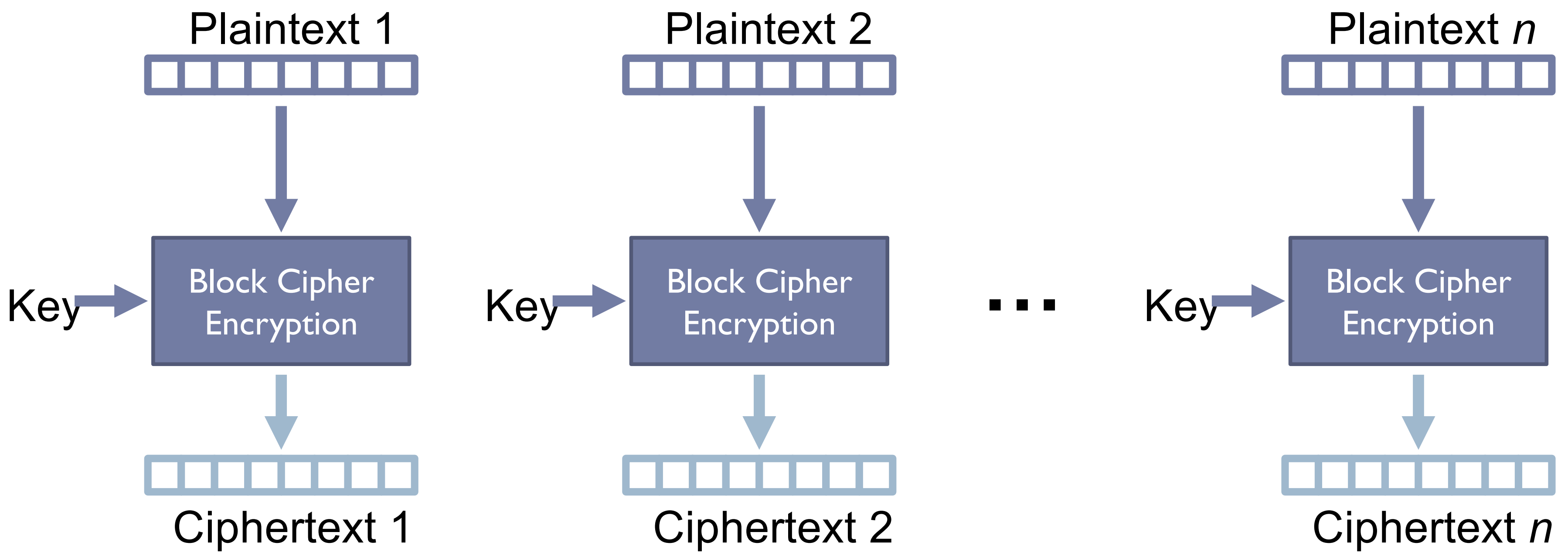
- AES is assumed to be secure (aka ciphertext is pseudorandom)!
- This is backed up by years of crytanalysis
- Block cipher: encrypts blocks of fixed size

Need for encryption modes

- A block cipher encrypts only one block
  - But a message may be longer than one block
- Need a way to extend the algorithm to encrypt arbitrarily long messages
- Need to ensure that if block cipher is secure, then whole encryption is secure
  - Whole operation should be secure if block cipher is secure

ECB encryption mode

- Message is broken into independent blocks
- Electronic Code Book (ECB): each block is encrypted separately



Cryptanalysis of ECB

- Deterministic
  - The same data block always gets encrypted the same way
    - Reveals patterns when data repeats!
  - *m* encrypted with *k* always produces the same *c*
  - This is the same problem we had with the Vigenère cipher
- Is the ECB mode IND-CPA secure?
- Is the ECB mode IND-EAV secure?
- **Do not use ECB mode in practice**

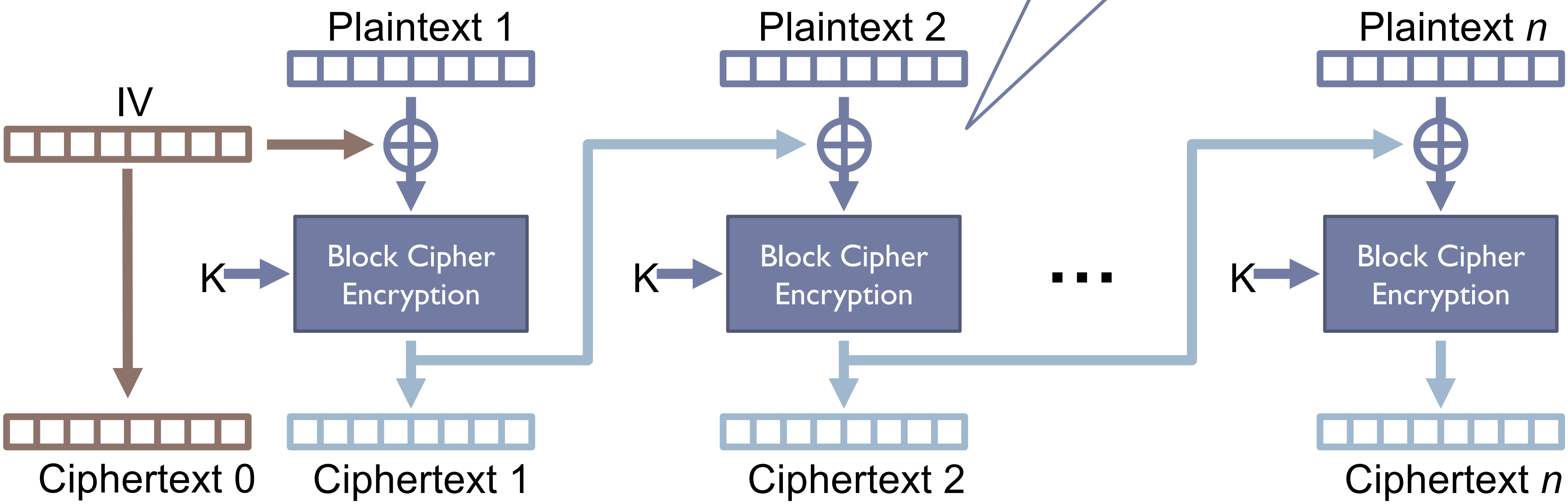
Lessons on IND-CPA security

- ECB uses deterministic encryption
  - Encryption of a message m is always the same
  - Adv can trivially win the IND-CPA game
- Deterministic encryption is not IND-CPA secure!
- CPA secure encryption needs to be randomized!
  - How is that achieved?

CBC encryption mode

- Cipher Block Chaining (CBC)
  - Uses a random Initialization Vector (IV)
  - Block *i* depends on block *i-1*

⊕ is exclusive bitwise OR (XOR)



Cryptanalysis of CBC

- CBC randomizes the encryption
  - IV ensures initial block is randomized
  - Dependency between blocks propagates randomness
- CBC is IND-CPA secure assuming
  - Block cipher itself is secure (pseudorandom permutation)
  - IV is truly random
  - IV is sufficiently large
  - Use the key for limited number of encryptions (key needs to be changed afterwards)
- Usage in practice: choose random IV and protect its integrity
  - The IV is not secret (it becomes part of the ciphertext)
  - Do not let the adversary control the IV (needs to be unpredictable)!

An example CBC analysis

q = # messages encrypted with k
L = length of message (in blocks)

Suppose we want **Pr[Attacker wins CPA game]** $\leq 1/2 + 1/2^{32}$

$$q^2 L^2 / 2^n < 1/ 2^{32}$$

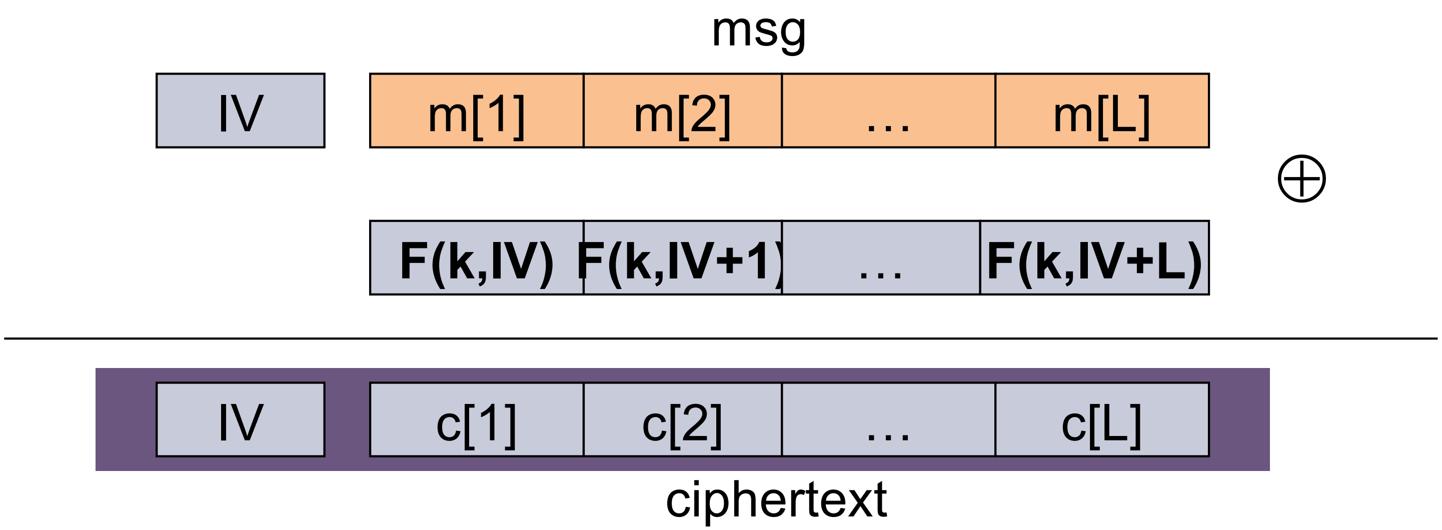
- AES: $2^n = 2^{128} \quad \Rightarrow \quad q\,L < 2^{48}$

  So, after 2^{48} AES blocks, must change key

CTR-mode encryption

Let F be a secure block cipher (e.g., ENC-AES)

Enc(k,m): choose a random IV and do:



msg

IV | m[1] | m[2] | … | m[L]

$\oplus$

F(k,IV) | F(k,IV+1) | … | F(k,IV+L)

IV | c[1] | c[2] | … | c[L]

ciphertext

$$c_i = F_k(IV + i) \oplus m_i$$

Comparison of CBC and CTR mode

- Both are IND-CPA secure assuming
  - Block cipher itself is secure (pseudorandom permutation)
  - IV is truly random with size of block cipher
  - Use the key for limited number of encryptions (key needs to be changed afterwards)
- CTR mode has better security bounds
- CTR mode is parallelizable, while CBC is sequential
- In CTR encryption can be done off line
- In CTR blocks can be independent decrypted, no other blocks needed for decryption of a given block