CO 487 Course Notes: Applied Cryptography

Winter 2026 - Samuel Jaques

Talha Yildirim,
tyildir [at] uwaterloo [dot] ca

Contents

1 Intro and History 3
2 Symmetric Encryption 6
2.1 Block Ciphers 6
2.2 Block Cipher Modes of Operations 14

1 Intro and History

Definition 1.1 (Cryptography)

Cryptography is about securing communications in the presence of malicious adversaries

♣

Remark

States of information

- Data at rest
- Data at transit
- Data while processing

Corollary 1.1.1 (Fundamental Goals of Cryptography)

- **Confidentiality** - Keeping data secret from all but those authorized to see it
- **Integrity** - Ensuring data has not been altered by unauthorized means
- **Authentication** - Corroborating the source of data or identity of an entity
- **Non-repudiation** - Preventing an entity from denying previous commitments or actions
- **Deniability** - Allowing an entity to deny previous commitments or actions
- **Consensus** - Ensuring a number of entities agree on the state of some data
- **Availability** - Ensuring a computer system works even if parts of it fail or are tampered with

♡

Remark

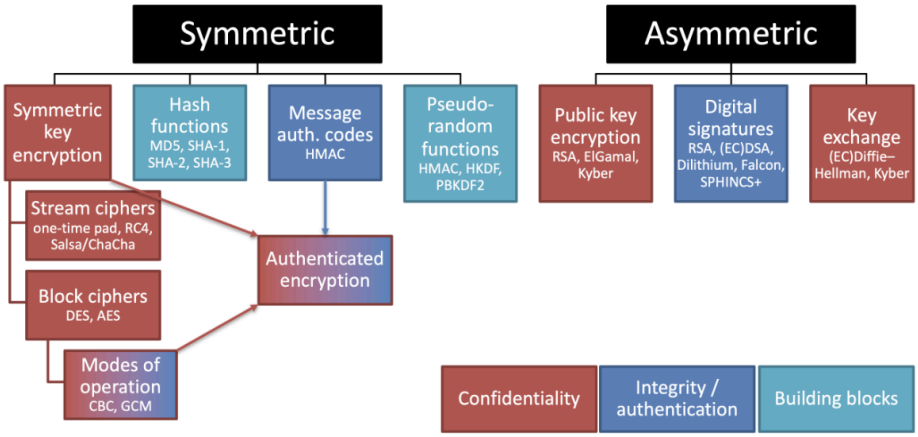


Figure 1: Cryptography building blocks

Symmetric cryptography: Uses a single shared secret to protect data. Both parties use the same secret, so it is simple and efficient, but requires the secret to be shared securely.

Asymmetric cryptography: Uses two related keys, one public and one private. This allows parties to communicate or verify identity without sharing a secret first, at the cost of being more complex and slower.

Definition 1.2 (Caesar Cipher)

Caesar cipher is a simple substitution cipher that encrypts text by shifting each letter a fixed number of positions forward in the alphabet, wrapping around at the end

Encrypt(m) where $m \in \{A, \dots, Z\}^*$

1 : **for** $i = 1, \dots, |m|$
2 : $x \leftarrow \text{Encode}(m_i)$
3 : $y \leftarrow x + 23 \bmod 26$
4 : (or equivalently $y \leftarrow x - 3 \bmod 26$)
5 : $c_i \leftarrow \text{Decode}(y)$
6 : **return** c

Decrypt(c) where $c \in \{A, \dots, Z\}^*$

1 : **for** $i = 1, \dots, |c|$
2 : $x \leftarrow \text{Encode}(c_i)$
3 : $y \leftarrow x - 23 \bmod 26$
4 : (or equivalently $y \leftarrow x + 3 \bmod 26$)
5 : $m_i \leftarrow \text{Decode}(y)$
6 : **return** m

Encode / Decode:

A	B	C	D	E	F	G	H	I	J	K	L	M
0	1	2	3	4	5	6	7	8	9	10	11	12
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
13	14	15	16	17	18	19	20	21	22	23	24	25

This scheme is **NOT** secure

Remark

Note the difference between "encrypt" and "encode", and "decrypt" and "decode"

- **encode and decode** map letters to numbers without trying to add security – just mapping to a more convenient space
- **encrypt and decrypt** try to add security

Definition 1.3 (Shift Cipher)

Idea: Modify Caeser cipher by introducing a secret key

Key space

$\mathcal{K} = \{0, \dots, 25\}$
Randomly sample key $k \xleftarrow{\$} \mathcal{K}$

Encrypt(k, m) where $m \in \{A, \dots, Z\}^*$

1 : **for** $i = 1, \dots, |m|$
2 : $x \leftarrow \text{Encode}(m_i)$
3 : $y \leftarrow x + k \bmod 26$
4 : $c_i \leftarrow \text{Decode}(y)$
5 : **return** c

Decrypt(k, c) where $c \in \{A, \dots, Z\}^*$

1 : **for** $i = 1, \dots, |c|$
2 : $x \leftarrow \text{Encode}(c_i)$
3 : $y \leftarrow x - k \bmod 26$
4 : $m_i \leftarrow \text{Decode}(y)$
5 : **return** m

This scheme is **NOT** secure

How might we break the shift cipher?

Assumptions

Definition 1.4 (Kerckhoff's Principal)

We know the encryption / decryption algorithm being used but not any secret keys

and,

We are given a cipher text to break

Two approaches

1. Try all 26 possible secret key values $k = 0, \ldots, 25$ (brute force / exhaustive key search)
2. Frequency Analysis

Definition 1.5 (Frequency Analysis)

Compare the distribution of letters in the cipher text with the distribution of letters in the underlying plain text space

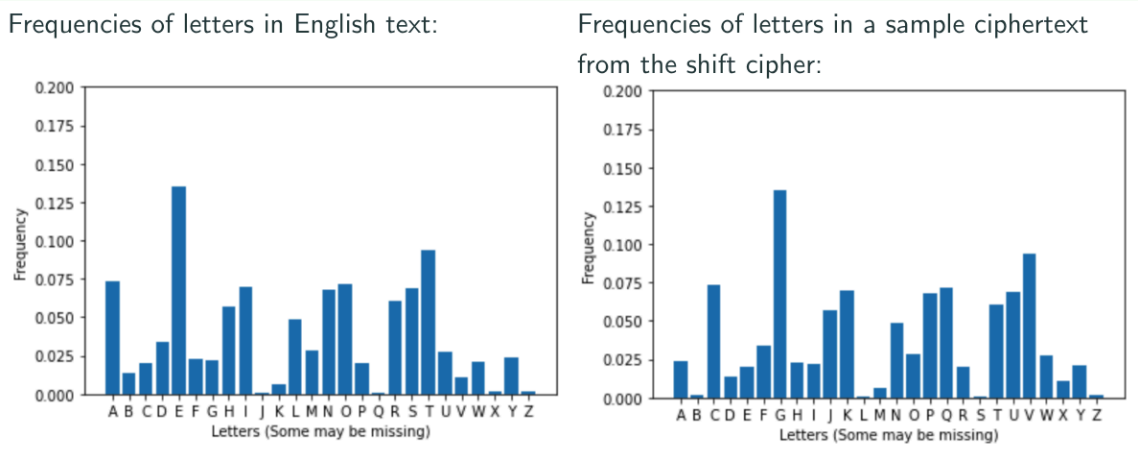


Figure 4: $k = 2$

Definition 1.6 (Vigenere Cipher)

Use different shift ciphers for different parts of the message to reduce effect of frequency analysis

Key space	**Message and ciphertext space**
$\mathcal{K} = \{A, \ldots, Z\}^B$	$\mathcal{M} = \mathcal{C} = \cup_{i \geq 0}\{A, \ldots, Z\}^{iB}$

Encrypt(k, m)

1 : **for** $i = 1, \ldots, |m|$
2 : $\quad c_i \leftarrow m_i + k[i \bmod B] \bmod 26$
3 : **return** c

Decrypt(k, c)

1 : **for** $i = 1, \ldots, |c|$
2 : $\quad m_i \leftarrow c_i - k[i \bmod B] \bmod 26$
3 : **return** m

Example with block length $B = 6$

$m =$	t	h	i	s		i	s		a	m	e	s	s	a	g	e
$+ k =$	C	R	Y	P		T	O		C	R	Y	P	T	O	C	R
$c =$	V	Y	G	H		B	G		C	D	C	H	L	O	I	V

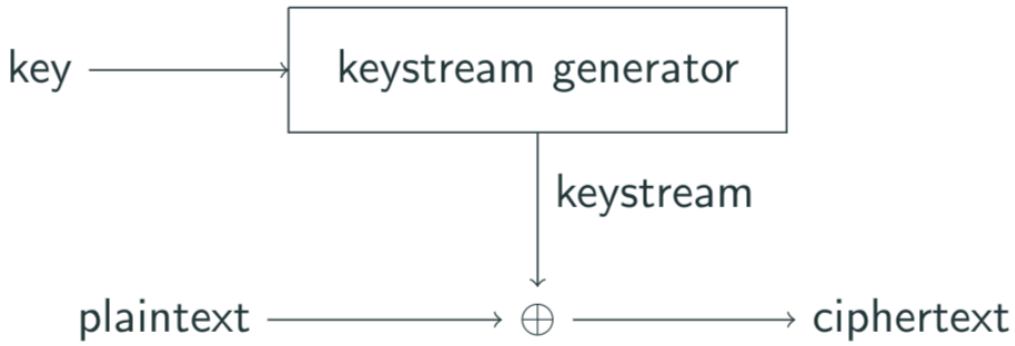
This scheme is **NOT** secure

2 Symmetric Encryption

2.1 Block Ciphers

Definition 2.1.1 (Stream Cipher)

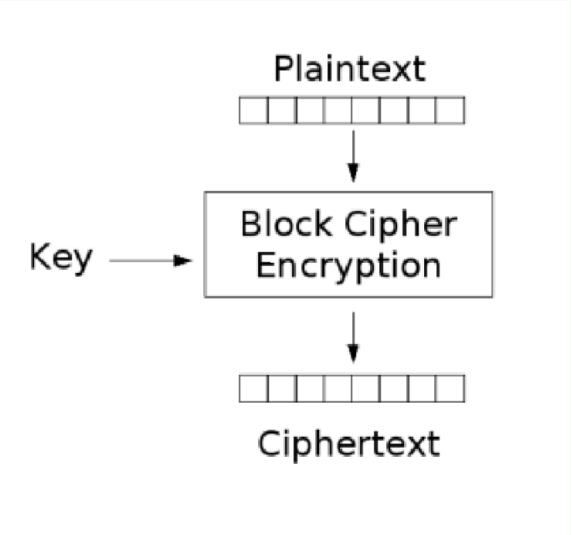
A stream cipher is a symmetric key encryption scheme in which each successive character of plaintext determines a single character of ciphertext



Definition 2.1.2 (Block Cipher)

A block cipher is a symmetric key encryption scheme in which a fixed length block of plaintext determines an equal sized block of ciphertext

e.g. AES, DES

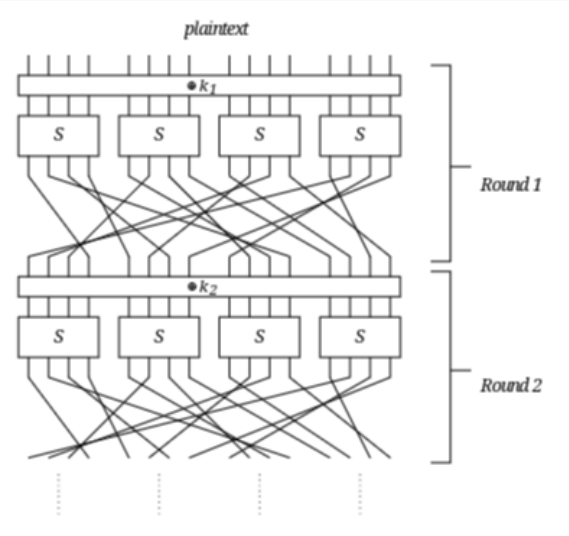


Corollary 2.1.2.1 (Desirable Properties of Block Ciphers)

- **Security**
  - **Diffusion:** Each ciphertext bit should depend on all plaintext and all key bits
  - **Confusion:** The relationship between key bits, plaintext bits, and ciphertext bits should be complicated
  - **Cascade or Avalanche Effect:** Changing one bit of plaintext or key should change each bit of ciphertext with 50% probability
  - **Key Length:** Should be small, but large enough to preclude exhaustive key search
- **Efficiency**
  - Simplicity (easier to implement and analyze)
  - High encryption and decryption rate
  - Suitability for hardware or software

Definition 2.1.3 (Substitution Permutation Networks)

A substitution permutation network (SPN) is a multiple round iterated block cipher where each round consist of a substitution operation followed by a permutation operation. During each round, a round key is `XOR`ed into the state. The round keys k_i are derived from the main key k using a key schedule function.



♣

Warning

Despite being called a Substitution Permutation Network, the permutation can be any invertible linear function.

Remark

To a mathematician, any bijective (one-to-one) function on a finite set can be considered a "permutation". To avoid confusion lets clarify permutation and substitution in a substitution permutation network.

1. **Bijection or Substitution**

The S-box will (often) be a one-to-one function on some small space

e.g. 4 or 8 bit strings

11110001 $\rightarrow$ 10100001

11110010 $\rightarrow$ 01000101

In this course this will be referred to as a bijection or substitution

2. **Permutation**

The permutation will be a one-to-one function of positions of bits.

e.g. A block of 128 bits

$p : \{0, 1, 2, ..., 127\} \rightarrow \{0, 1, 2, ..., 127\}$

This tells us to move the bit at position i to position $p(i)$

Remark

Two perspevtives on a permutation

1. As a function from bit position to bit position (i.e a one-to-one function on $\{0, 2, ...\}, 127)$
2. As a function from bit strings to bit strings (i.e take each bit in the bit string and rearrange it according to how the positions are permuted)

As an example, we could have a permutation like this:

Start Position	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
End Position	11	13	15	0	8	7	9	14	5	2	10	12	6	1	4	3

which would act on 16-bit strings as dictated above, which could also be done by a matrix:

$$\begin{pmatrix} 0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0 \\ 0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

Definition 2.1.4 (The Advanced Encryption Standard (AES))

Requirements

- **Key Sizes:** 128, 192 and 256 bits
- **Block Sizes:** 128 bits
- Efficient on both hardware and software platforms
- Availability on a worldwide, non-exclusive, royalty-free basis
- AES is a SPN where the "permutation" operation consist of two linear transformations (one of which is a permutation)
- All operations are byte oriented

Number of rounds depends on the key length

key length	number of rounds h
128	10
192	12
256	14

- The substitution operation (S-box) is the only non-linear component
- The permutation operations (permutation and linear transformations) spread out the non-linearities in each round

Corollary 2.1.4.1 (AES Round Operations)

- Each round updates a variable called `State` which consist of a $4x4$ array of bytes (notes: $4 \times 4 \times 8 = 128$, the block size)
- `State` is initialized with the 128-bit plaintext

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$

$\longleftarrow$ *plaintext*

- After h rounds are completed, one final additional round key is `XORed` with `State` to produce the ciphertext
- The AES round function uses four operations:
  - `AddRoundkey` , key mixing
  - `SubBytes` , S-box
  - `ShiftRows` , permutation
  - `MixColumns` , matrix multiplication / linear transformation

Corollary 2.1.4.2 (AES Encryption)

From the key k, derive $h+1$ round keys $k_0, k_1, ..., k_h$ via the key schedule

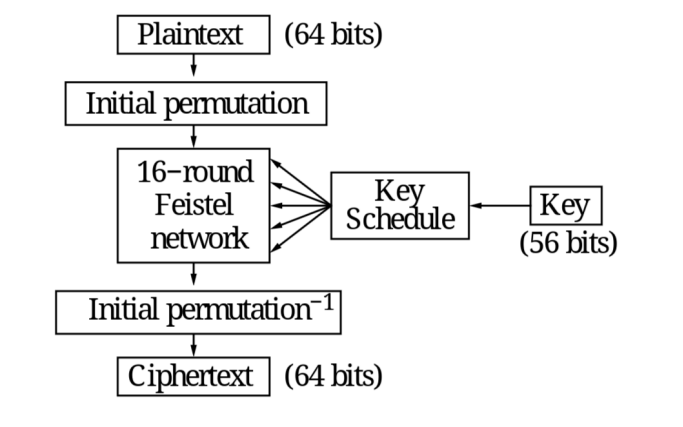
Encryption function:

State $\leftarrow$ plaintext
for $i = 1 \ldots h-1$ do
    State $\leftarrow$ State $\oplus\ k_{i-1}$
    State $\leftarrow$ SubBytes(State)
    State $\leftarrow$ ShiftRows(State)
    State $\leftarrow$ MixColumns(State)
State $\leftarrow$ State $\oplus\ k_{h-1}$
State $\leftarrow$ SubBytes(State)
State $\leftarrow$ ShiftRows(State)
State $\leftarrow$ State $\oplus\ k_h$
ciphertext $\leftarrow$ State

Not in the final round `MixColumns` is not applied

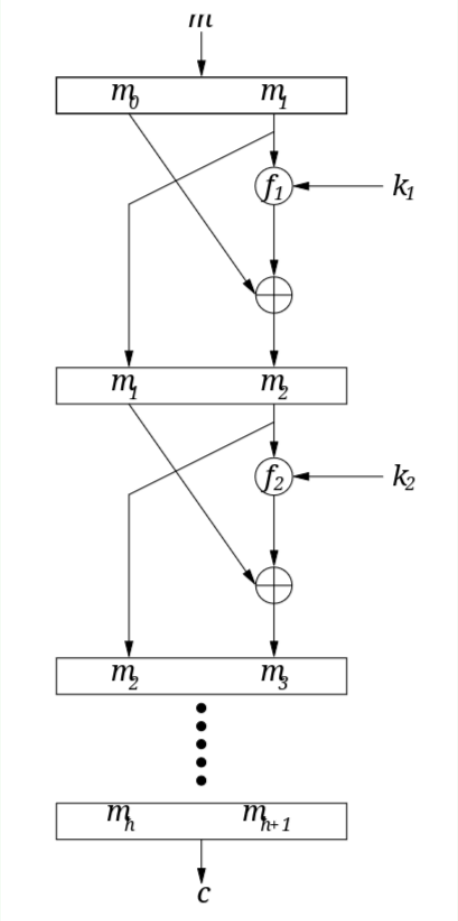
Definition 2.1.5 (Data Encryption Standard DES)

Block cipher with 64-bit blocks, 56-bit key, and 16 rounds of operation



Definition 2.1.6 (Feistel Network Design)

- DES uses a Feistel network design
- Plaintext is divided into two halves
- Key is used to generate subkeys $k_0, k_1, \ldots, k_h$
- f_i is a component function whose output value depends on k_i and m_i



Corollary 2.1.6.1 (Feistel Ciphers: A Class of Block Ciphers)

- **Components of a Feistel Cipher:**
  - Parameters: n (half the block length), h (number of rounds). l (key size)
  - $M = \{0,1\}^{2n}$, $C = \{0,1\}^{2n}$, $K = \{0,1\}^{l}$
  - A key scheduling algorithm which determines subkeys $k_0, k_1, ..., k_h$ from a key k
  - Each subkey k_i defines a component function $f_i : \{0,1\}^l \times \{0,1\}^n \to \{0,1\}^n$
- **Encryption takes h rounds:**
  - Plaintext is $m = (m_0, m_1)$, where $m_i \in \{0,1\}^n$
  - **Round 1:** $(m_0, m_1) \to (m_1, m_2)$, where $m_2 = m_0 \oplus f_1(k_1, m_1)$
  - **Round 2:** $(m_1, m_2) \to (m_2, m_3)$, where $m_3 = m_1 \oplus f_2(k_2, m_2)$
  - **Round h:** $(m_{h-1}, m_h) \to (m_h, m_{h+1})$, where $m_{h+1} = m_{h-1} \oplus f_h(k_h, m_h)$
  - Ciphertext is $c = (m_h, m_{h+1})$
- **Decryption:** Given $c = (m_h, m_{h+1})$ and k, to find $m = (m_0, m_1)$
  - Compute $m_{h-1} = m_{h+1} \oplus f_h(k_h, m_h)$
  - Similarly, compute $m_{h-2}, ..., m_1, m_0$

♡

Remark

- No restrictions on the functions f_i in order for the encryption procedure to be invertible
- **Underlying principle:** Take something "simple" and use it several times; hope that the result is "complicated"

Corollary 2.1.6.2 (DES Problem: Small Key Size)

Exhaustive seach on key space takes 2^{56} steps and can be easily parallelized

♡

Definition 2.1.7 (Multiple Encryption)

Re-encrypt the ciphertext one or more times using independent keys, and hope that this operation increase the effective key length

Multiple encryption does not always increase security. e.g. Substitution cipher

♣

Definition 2.1.8 (Double Encryption)

- **Double DES:** Key is $k = (k_1, k_2), k_1, k_2 \underset{R}{\in} \{0,1\}^{56}$
- **Encryption:** $c = E_{k2}(E_{k1}(m))$
- **Decryption:** $c = E_{k1}^{-1}(E_{k2}^{-1}(m))$
- Key length of Double DES is $l = 112$, so exhaustive key search takes 2^{112} steps (infeasible)

♣

Corollary 2.1.8.1 (Attack on Double DES)

Main idea: If $c = E_{k2}(E_{k1}(m))$, then $E_{k2}^{-1}(c) = E_{k1}(m)$ (meet-in-the-middle)

1. Given: Known plaintext pairs (m_i, c_i) , $i = 1, 2, 3, ...$
2. For each $h_2 \in \{0,1\}^{56}$:
   - Compute E_{h2}^{-1}, and store $[E_{h2}^{-1}, h_2$ in a table
3. For each $h_1 \in \{0,1\}^{56}$ do the following
   - Compute $E_{h_1}(m_1)$
   - Search for $E_{h_1}(m_1)$ in the table
   - If $E_{h_1}(m_1) = E_{h_2}^{-1}(c_1)$:
     - Check if $E_{h_1}(m_1) = E_{h_2}^{-1}(c_2)$
     - Check if $E_{h_1}(m_1) = E_{h_2}^{-1}(c_3)$
     - Etc
     - If all checks pass, then output (h_1, h_2) and stop

Complexity of the attack is $\approx 2^{57}$

♡

Remark

- Number of known plaintext / ciphertext pairs required to avoid false keys: 2 suffice, with high probability
- Number of DES operations $\approx 2^{56} + 2^{56} + 2 \times 2^{48} \approx 2^{57}$
- Space requirements: $2^{56}(64 + 65)$ bits $\approx 1,080,863$ Tbytes

Double DES effectively has the same key length as DES and isn't much more secure

Definition 2.1.9 (Three Key Triple Encryption)

- **Triple DES:** Key is $k = (k_1, k_2, k_3), k_1, k_2, k_3 \underset{R}{\in} \{0,1\}^{56}$
- **Encryption:** $c = E_{k3}(E_{k2}(E_{k1}(m)))$
- **Decryption:** $c = E_{k1}^{-1}\left(E_{k2}^{-1}\left(E_{k_3}^{-1}(m)\right)\right)$
- Key length of Triple DES is $l = 168$, so exhaustive key search takes 2^{168} steps (infeasible)

♣

Corollary 2.1.9.1 (Attack on Triple DES)

- meet-in-the-middle attack takes $\approx 2^{112}$ steps
- So, the effective key length of Triple DES against exhaustive key search is ≤ 112 bits
- No proof that Triple DES is more secure than DES
- Block length is 64 bits, and now forms the weak link:
  - Adversary stores a large table (of size $\leq 2^{64}$) pf (m, c) pairs (dictionary attack)
  - To prevent this attack $\rightarrow$ change secret keys frequently

♡

2.2 Block Cipher Modes of Operations

Remark

Recall,

Stream Cipher: A symmetric key encryption scheme in which each a pseudorandom sequence of arbitrary length is generated to encrypt successive character of plaintext of ciphertext

Block Cipher: A symmetric key encryption scheme in which a fixed length block of plaintext determines an equal sized block of ciphertext

Lemma 2.2.1 (Encrypting Bulk Data)

What if one needs to encrypt large quantities of data?

- With a stream cipher, just encrypt each character
- With a block cipher, there are some complications if:
  - The input is larger than one block
  - The input does not fill an integer number of blocks

To deal with these problems we use a mode of operation, which means a specification for how to encrypt multiple and / or partial data blocks using a block cipher

♡

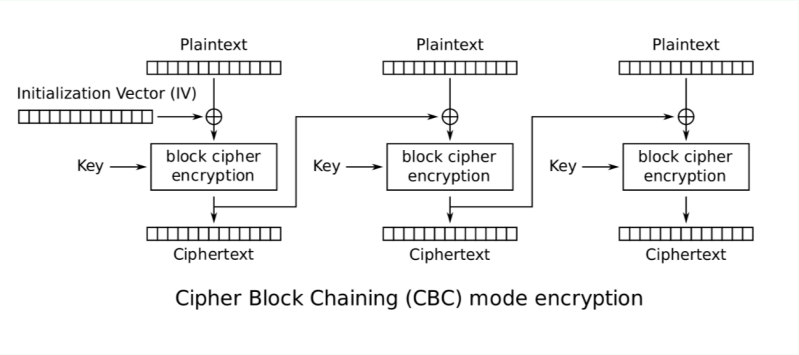
Definition 2.2.2 (Padding)

- Some modes namely ECB and CBC, require the plaintext to consist of one or more complete blocks
- Padding method
  - Append a single '1' bit to the data string
  - Pad the resulting string by as few '0' bits, possibl none, as are necessary to complete the ifnal block
- The padding bits can be removed unambiguously, if the receiver known that this padding method is used

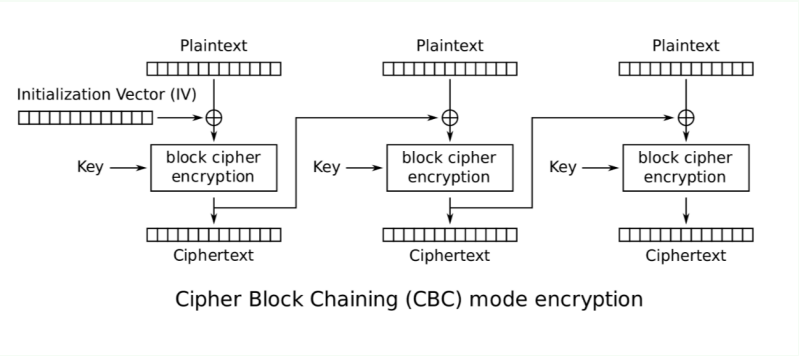
♣

Definition 2.2.3 (Cipher Block Chaining mode (CBC))

Use a single initialization vector for the first block of plaintext, and "chain" the (pseudorandom) ciphertext blocks as the next blocks initialization vectors. The IV is included as part of the ciphertext



Cipher Block Chaining (CBC) mode encryption



Cipher Block Chaining (CBC) mode encryption

- **Encryption:**
  - $C_i = E(K, P_i \vee C_{i-1})$, where $C_0 = IV$
  - P_i is XORed with the previous ciphertext block C_{i-1}, and encrypted with the key K to produce ciphertext block C_i. For the first plaintext block IV is used for the value of C_0
  - **Decryption:**
    - $P_i = D(K, C_i) \oplus C_{i-1}$
    - C_i is decrypted with the K ,and XORed with the previous ciphertext block C_{i-1} to produce plaintext block P_i. As in encyption, IV is used in place of C_0