

# CS2107

## Introduction to Information Security

AY2022/23 Semester 2

Notes by Jonathan Tay

Last updated on January 19, 2023

---

### Contents

<b>I</b>	<b>Cryptography</b>	<b>1</b>
<b>1</b>	<b>Encryption</b>	<b>1</b>
1.1	Classical Ciphers . . . . .	1
1.2	Modern Ciphers . . . . .	1
1.3	Attacks . . . . .	2

---

## Part I

# Cryptography

## 1 Encryption

**Symmetric-key encryption** uses the same key  $k$  for encryption and decryption.

Any plaintext  $x$  that is encrypted to produce a ciphertext  $c = E_k(x)$  must also be decryptable to recover the plaintext  $D_k(E_k(x)) = x$  — (**correctness**).

It must also be difficult or impossible to derive useful information of the key or the plaintext from the ciphertext — (**security**).

### attack models: information

Given access to an **oracle** with varying levels of information, adversaries can conduct several models of attack:

- **ciphertext only (COA)**: only ciphertexts and properties of the plaintext are known
- **known-plaintext (KPA)**: ciphertexts of corresponding plaintexts are known
- **chosen-plaintext (CPA)**: ciphertexts can be derived from arbitrary plaintexts
- **chosen-ciphertext (CCA2)**: plaintexts can be derived from arbitrary ciphertexts

### attack models: goals

Adversaries may have varying goals:

- **total break**: find the encryption key
- **partial break**: decrypt a ciphertext or obtain information about a plaintext
- **indistinguishability**: distinguish ciphertexts of different plaintexts better than random chance

### 1.1 Classical Ciphers

#### substitution cipher

Construct a **substitution table** used as the key from some permutation of all possible characters used in the plaintext.

Replace each character in the plaintext with the corresponding character in the substitution table to produce the ciphertext.

Substitution ciphers are vulnerable to various attack models:

- **general**: brute-force exhaustive search on all keys
- **KPA**: substitution table reconstruction by comparison of characters
- **COA**: frequency analysis of characters between ciphertext and plaintext

#### permutation ciphers

Group characters of the plaintext into blocks of equal size, then rearrange the characters in each block according to a permutation key.

Extremely vulnerable under KPA and COA attacks as the permutation key can be easily reconstructed, especially if the plaintext language is known.

Note that applying substitution or permutation ciphers several times alone in succession is equivalent to applying it once — the ciphertext is not any more secure!

#### one-time pad

Generate a key of equal bit length as the plaintext, then XOR them together to encrypt. XOR the ciphertext with the key to decrypt:

**correctness**:

1.  $c = x \text{ XOR } k$
2.  $(x \text{ XOR } k) \text{ XOR } k = x \text{ XOR } (k \text{ XOR } k) = x \text{ XOR } 0 = x$

**security**: Even if an adversary obtains a ciphertext-plaintext pair and derives the key (trivial), the key is useless as it is not re-used.

Exhaustive search does not work on OTPs because no meaningful information about the plaintext can be learnt (**perfect secrecy**) — every decrypted plaintext is equally probable.

### 1.2 Modern Ciphers

**DES** (data encryption standard) has a key length of 56 bits, and as such is vulnerable to exhaustive search today.

**AES** (advanced encryption standard) has a block length of 128 bits, and a key length of 128, 192, or 256 bits.

Both DES and AES are **block ciphers** which inputs and outputs of fixed length. This means plaintext must be padded and divided into blocks before encryption.

There are several methods of extending an encryption from single to multiple blocks.

**ECB (electronic code book) mode**

Encrypt each block independently, then concatenate the ciphertexts.

If the encryption scheme is *deterministic*, then any two blocks of plaintext encrypts to the same ciphertext, which leaks information.

AES is inherently deterministic, and as such requires an **initialization vector (IV)** for probabilistic encryption.

An IV is an arbitrary plaintext value (i.e. not secret) that should be unique for each encryption:

1. Let  $X$  and  $Y$  be two different plaintexts,  $U$  and  $V$  their ciphertexts, and  $K$  the IV.
2.  $U \oplus V = (X \oplus K) \oplus (Y \oplus K)$ .
3. By associativity and commutativity,  $U \oplus V = (X \oplus Y) \oplus (K \oplus K) = X \oplus Y$ .
4.  $X \oplus Y$  can leak information.
5. Hence, we need a unique IV for each encryption.

The IV is also needed for decryption and as such is sent together with the ciphertext.

**CBC (cipher block chaining) mode**

XOR the first block with the IV, then encrypt it.

Subsequent blocks are XOR'd with the previous ciphertext block before encryption.

**CTR (counter) mode**

Generate a cryptographically-secure pseudo-random sequence from the secret key and IV *and then* encrypt it (not the plaintext).

Plaintext blocks can be streamed in are XOR'd with the encrypted sequence (**stream cipher**).

Decryption uses the same process.

**1.3 Attacks****meet-in-the-middle attack (MITM) ( $\in$  KPA)**

Suppose we apply DES twice sequentially using keys  $K_1$  and  $K_2$  to encrypt a plaintext  $p$  into ciphertext  $c = E_{K_2}(E_{K_1}(p))$ .

Recall that in a KPA, an adversary aiming for a total break wants to find  $K_1$  and  $K_2$  given  $c$  and  $p$ .

Define the intermediate encryption value  $x = E_{K_1}(p)$ . For decryption, we also have  $x' = D_{K_2}(c)$ .

Now, the adversary can pre-compute and store all  $2^{56}$  possible values of  $x$  and all  $2^{56}$  possible values of  $x'$  in two hash tables.

With  $2^{57}$  possible pairs of  $(x, x')$ , exhaustive search will yield the keys  $K_1$  and  $K_2$ .

Hence, instead of the expected key strength of  $2 \times 56 = 112$ , by trading space for time, the key strength is only 57.

Any sequential encryption scheme is vulnerable to MITM.

Since MITM only reduces key strength, it can be remedied by increasing the number of rounds in the encryption scheme, e.g. **triple DES** (3DES).

**padding oracle attack ( $\in$  CCA2)**

Suppose AES CBC with **PKCS#7 padding** is used to encrypt a plaintext  $p$  into ciphertext  $c$  with 3 blocks of size of 4 bytes, giving a total of 12 bytes.

**PKCS#7 padding**

If a plaintext does not have a length that is exactly a multiple of the block size, let  $l$  be the number of bytes needed to fill (pad) the last block.

The last  $l$  bytes of the plaintext are set to  $l$ .

Otherwise, an additional block is added with all bytes set to 0.

An adversary wants a partial break, i.e. decrypt  $c$  to get  $p$ , with access to a **padding oracle** which returns true if the padding is valid, and false otherwise.

To decrypt AES CBC, each block of  $c$  is first decrypted with the key, then XOR'd with the previous block (or the IV for the first block).

This means that a change in any byte of  $c$ , say  $c_8$ , will result in a different value for the plaintext byte in the same position of that and every subsequent block, i.e.  $p_8$  and  $p_{12}$ .

Therefore, by exhausting all 256 possible values of  $c_8$ , the padding oracle will return true for the one correct value of  $p'_{12} = D_k(c_{12}) \oplus c'_8$ .

Since  $p'_{12} = p_{12} \oplus c_8 \oplus c'_8$ , and the values of  $p'_{12}$ ,  $c_8$  and  $c'_8$  are known,  $p_{12} = p'_{12} \oplus c_8 \oplus c'_8$ !

**forcing a specific value  $x$  for  $p'_i$** 

To proceed further with the attack on  $p_{11}$  via  $c_7$ , we need to force the value of  $p'_{12}$  to be  $0x2$ , so that we can brute-force  $c'_7$  to get  $p'_{11}$  to be  $0x2$ .

Hence, we want a value for  $c''_8$  which achieves this, which is  $c_8 \oplus p_{12} \oplus x$ , where  $x = 0x02$ .

We can repeat this process for  $p_{11}$  after setting  $c_8 = c''_8$ , and so on, truncating  $c$  so that the padding does not exceed one block.

Any encryption scheme using CBC is vulnerable to the padding oracle attack, and so is CTR if it uses padding.