Faculty of Engineering and Technology
Computer Science Department

Encryption Theory (COMP438)

Assignment-2 Report
“One-Time-Pad & Rail-Fence Ciphers”

Prepared by: Motasem Nabeel Ali - 1210341

Instructor: Mohammad Alkhanafseh

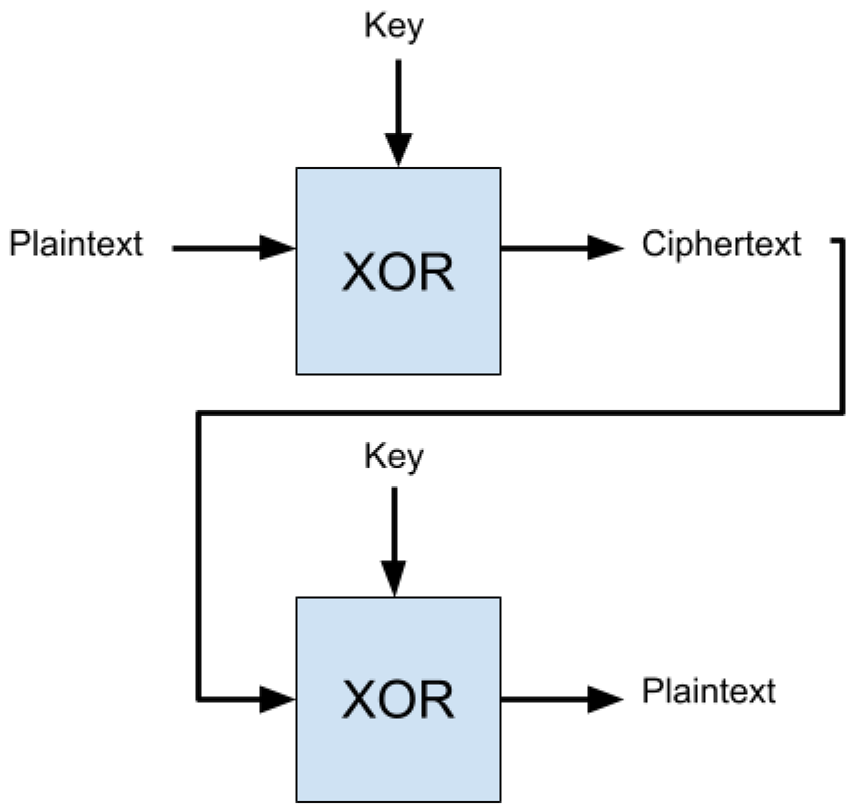
Section: 1

Date: 03/1/2024

Table of Contents

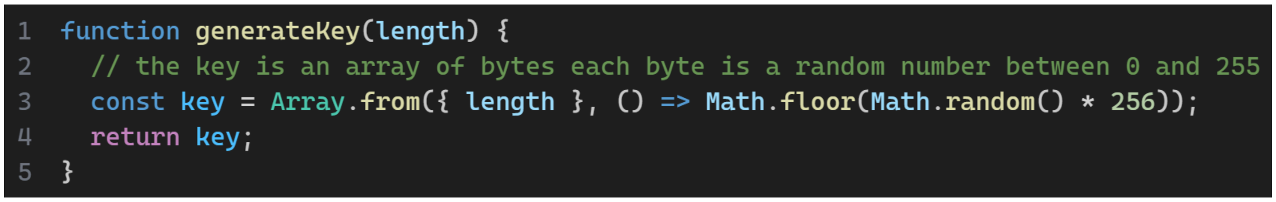
1. One Time Pad Cipher

It is a type of a symmetric encryption technique that uses the same key for encryption and decryption process. The key size is the same as the plaintext size. An XOR operation is used as a function to encrypt and decrypt the message because XOR is a reversible operation.



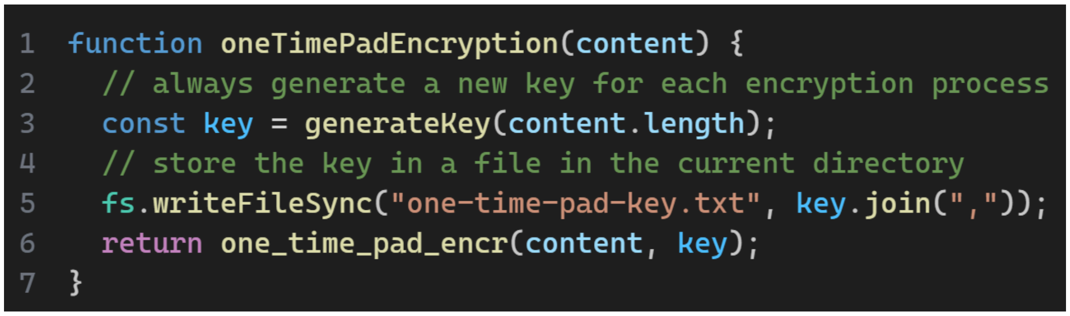
1.1 Procedure

1. Encryption:
    a. the plaintext is converted into an array of bytes, each character to its corresponding byte.
    b. The key is generated using Math.random() function and the length of the key is the number of characters in the plaintext (the generated key is an array of bytes).



```
function generateKey(length) {
  // the key is an array of bytes each byte is a random number between 0 and 255
  const key = Array.from({ length }, () => Math.floor(Math.random() * 256));
  return key;
}
```

    c. New key is generated each time a new encryption process happens.



```
function oneTimePadEncryption(content) {
  // always generate a new key for each encryption process
  const key = generateKey(content.length);
  // store the key in a file in the current directory
  fs.writeFileSync("one-time-pad-key.txt", key.join(","));
  return one_time_pad_encr(content, key);
}
```

d. The key is stored in a secure place inside a txt file.
e. The plaintext is converted to an array of bytes each character corresponding to its byte representation.

```
const messageBinary = plaintext.split("").map((char) => char.charCodeAt(0));
```

f. XOR the plaintext with the key to get the ciphertext.

```
const encryptedBinary = messageBinary.map((byte, i) => {
    return byte ^ key[i];
  });
```

2. Decryption:
    a. The method takes the ciphertext which is an array of bytes and the key fetched from the secure file.

```

function oneTimePadDecryption(content) {
  const key = fs.readFileSync("one-time-pad-key.txt", "utf-8").split(",");
  return one_time_pad_decr(content, key);
}
```

b. XOR the ciphertext with the key, each byte of the ciphertext XOR each byte of the key.

```
const decryptedBinary = ciphertext.map((byte, i) => {
    return byte ^ key[i];
  });
```

c. At the end the key is cleared out and we convert the result of step "b" into the original plaintext.

3. Reading Content from Files
    a. The user could encrypt a file content and decrypt it into a new file.

1.2 Reusing The Key Weakens Security

Reusing the same key or part of it for more than one time will compromise the encryption process entirely. If you use the same key to encrypt two different plaintexts, the attacker could infer the relationship between the plaintexts by exploiting properties of the XOR operation:

1.2.1 Ciphertext1 XOR Ciphertext2

Ci : ciphertext_i
Pi : plaintext_i
K : key

The attacker can start with XORing the two ciphertexts that are encrypted with the same key until reaching the two plaintexts.

C1 = P1 XOR K
C2 = P2 XOR K

- C1 XOR C2 ⇒ (P1 XOR K) XOR (P2 XOR K)
- Associative ⇒ P1 XOR (K XOR (P2 XOR K))
- Commutative ⇒ P1 XOR (K XOR (K XOR P2))

- Associative ⇒ P1 XOR ((K XOR K) XOR P2)
- Inverse ⇒ P1 XOR (0 XOR P2)
- Identity ⇒ P1 XOR P2

What the attacker did here is he removed the key from the equation and now he remains with the combination of the both plaintexts (remember that the plaintexts aren't directly known).

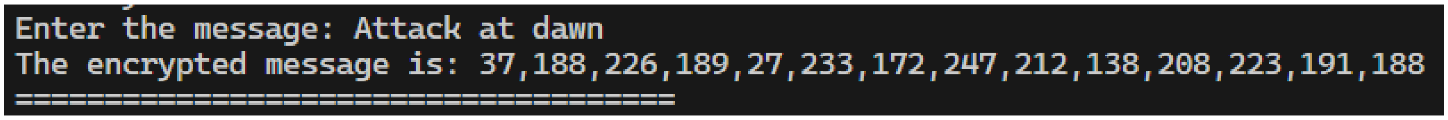
1.2.2 Types of Attacks

1. **Known Plaintext Attack:** if the attacker knows or figured out one of the plaintexts he could obtain the other immediately by doing this:

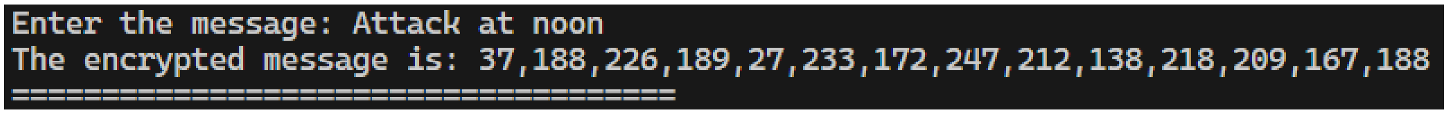
   P1 (Known)
   P2 (Unknown)

   (P1 XOR P2) XOR P1 ⇒ (P2 XOR P1) XOR P1 ⇒ (P2 XOR (P1 XOR P1) ⇒ P2 XOR 0 ⇒ P2

2. **Pattern Analysis Attack:** Even if the attacker could not know one of the plaintexts he could identify patterns in P1 XOR P2



```
Enter the message: Attack at dawn
The encrypted message is: 37,188,226,189,27,233,172,247,212,138,208,223,191,188
=====================================
```



```
Enter the message: Attack at noon
The encrypted message is: 37,188,226,189,27,233,172,247,212,138,218,209,167,188
=====================================
```

- Based on the following conclusion that Ciphertext1 XOR Ciphertext2 will lead to Plaintext1 XOR Plaintext2 the result is as follows:
  0 0 0 0 0 0 0 0 0 0 10 14 24 0
- The 0s indicate similar characters in plaintexts “Attack at “, and the numbers (10, 14, 24) indicates difference between the palintexts “dawn” / “noon”
- Now if the attacker knows some information about one of the plaintexts along with the result of XORing them, he could construct the other plaintext:
  - The attacker knows the first plaintext: P1
  - The attacker does: P1 XOR (P1 XOR P2) so he can get P2
  - P1 last 4 chars are: “d” “a” “w” “n” ⇒ 100, 97, 119, 110
  - P1 XOR P2 the different characters are: 10 14 24
  - Following the equation: P1 XOR (P1 XOR P2)
    - dawn XOR 10 14 24 0 ⇒
    - 100 XOR 10 ⇒ 110 ⇒ “n”
    - 97 XOR 14 ⇒ 111 ⇒ “o”
    - 119 XOR 24 ⇒ 111 ⇒ “o”
    - 110 XOR ⇒ 110 ⇒ “n”
    - “noon” which is the different part in P2

So even with partial information about one of the plaintexts that could be repeated in other plaintexts could reveal the others.

2. Rail Fence Cipher

It's a symmetric encryption technique that uses the concept of arrays to encrypt and decrypt messages. The key in this cipher represents the number of rows your message will be spread on. And the number of columns equals the number of characters in the message.

2.1 Procedure

Plaintext: “Hello, World!”
Key: 3

1. **Encryption:**
    a. The encryption method takes the key and the plaintext as arguments
    b. We construct an empty array with the proper number of rows and columns
    c. # of columns equals # of characters without punctuation/spaces/…etc which is 10

====>

H				o				l	
	e		l		W		r		d
		l				o			

    d. Ciphertext would be: “Hol elwrd lo” which is each rail followed by a blank space
2. **Decryption:**
    a. First we need to figure out the key which is the number of blocks in the ciphertext which is 3
    b. Then the number of columns is the number of characters in the ciphertext without the spaces which is 10
    c. Then we fill in zigzag order starts “ * “

*				*				*	
	*		*		*		*		*
		*				*			

    d. Then we start reading the ciphertext and substitute the stars with the characters from the ciphertext row by row

H				o				l	
	e		l		w		r		d
		l				o			

    e. Then you read the array using the zigzag order to obtain the plaintext again.
    f. The way we move in zigzag is by using a pointer on rows that keep going up and down, and another pointer that traverse all the columns