The University of Newcastle
School of Information and Physical Sciences

SENG2250/6250 System and Network Security
Week 3 Lab

PART 1: Cryptanalysis

Q1: Consider the following ciphertext, which has been obtained by applying monoalphabetic substitution cipher. Apply cryptanalysis to reveal the (meaningful) plaintext.

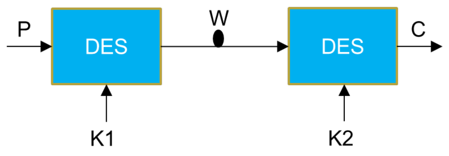
KRHPH FL BX BAAWH ZX KRH KPHH

PART 2: Triple-DES

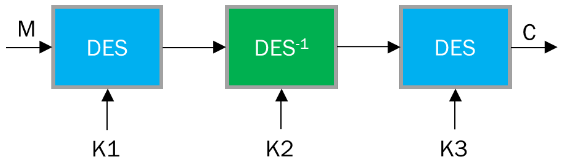
Meet-in-the-middle (MITM) attacks: The meet-in-the-middle attack (MITM) is a cryptographic technique used to break certain encryption schemes more efficiently.

- It involves dividing the encryption process into two parts and storing intermediate results in a table. By matching these results, the attacker can find the encryption key faster than a traditional exhaustive search.
- MITM is often employed against double encryption or other schemes that can be split into two separate operations.

Q1: Why does double-DES have (approx.) 2^{57} security level?



Q2: Why does Triple-DES (3 independent keys) have (approx.) 2^{112} security level?



Q3: Why is the middle portion of the triple-DES being decryption rather than encryption?

Part 3: Hash Functions

A hash function is like a magical machine that takes any input, such as a word, a sentence, or even a file, and turns it into a unique fixed-size jumbled-up code, which we call the "hash value" or "hash code. Think of it as a black box: you put something in, and you get a scrambled output that looks completely different from the original input. The cool part is that ***no matter how big or small the input is, the hash function always produces the same size of output***.

A **cryptographic hash function** is a special type of hash function designed for use in cryptography and security applications. It takes an input (also known as a message or data) and produces a fixed-size output, which is a unique jumbled-up code, called the "hash value" or "hash code."

The key characteristics of cryptographic hash functions are:

1. **Deterministic**: For the same input, the hash function always produces the same hash value.
2. **Fixed Size**: Regardless of the size of the input, the hash function always generates a fixed-length hash value.
3. **Irreversibility**: It should be extremely difficult (ideally, practically impossible) to reverse-engineer the original input from the hash value. This property is known as "pre-image resistance."
4. **Collision Resistance**: It should be computationally infeasible to find two different inputs that produce the same hash value. In other words, the chance of a collision (two different inputs having the same hash) should be negligible.
5. **Avalanche Effect**: A tiny change in the input should cause a significant change in the resulting hash value.

Q1: (a) Is the following function H a hash function? Why?

$$H(x) = x \bmod 65537, x \in \mathbb{N}$$

(b) Is the above function H a cryptographic hash function? If not, why?

PART4: Programming

Fast modular exponentiation ($b^e \bmod n$) is an essential operation for many modern security algorithms. In Lab 1, you have implemented the modular exponentiation function. But it is slow if the base and exponent are very large.
We introduce a fast modular exponentiation as below. Write a (C/C++/Java/Python) program to implement the fast modular exponentiation operation based on the following pseudocode.

```
function powmod2(base b, exponent e, modulus n) {
    if n  = 1
        return 0
    rs = 1
    while (e > 0) {
        if (e & 1) == 1
            rs = (rs * b) mod n
        e = e >> 1
        b = (b*b) mod n
    }
    return rs
}
```

Use the above implementation to find the solutions of the followings.

$$3^3 \bmod 7$$
$$10^8 \bmod 133$$
$$3785^{8395} \bmod 65537$$
$$17^{45} \times 17^{61} \bmod 1023$$
$$17^{45+61} \bmod 1023$$