Cryptography

Mainack Mondal

CS60112
Spring 2025

Today's class

- Basic Encryption Schemes
  - XOR
  - Substitution Cipher
  - Caesar Cipher
- Symmetric Encryption: AES and Modes of Operation
- Asymmetric Encryption: RSA

Basic Encryption Schemes

Basic Encryption Schemes: XOR

- An XOR or *eXclusive OR* is a bitwise operation

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

$$
\begin{array}{c}
111010100101 \\
\oplus \\
010101110100 \\
\hline
\text{Result} \\
\downarrow \\
101111010001
\end{array}
$$

XOR in CTFs

- XOR is a cheap way to encrypt data with a password.
- It can be used for encryption as follows:

```
>>> data = 'CAPTURETHEFLAG'
>>> key = 'A'
>>> encrypted = ''.join([chr(ord(x) ^ ord(key)) for x in data])
>>> encrypted
'\x02\x00\x11\x15\x14\x13\x04\x15\t\x04\x07\r\x00\x06'
>>> decrypted = ''.join([chr(ord(x) ^ ord(key)) for x in encrypted])
>>> decrypted
'CAPTURETHEFLAG'
```

Substitution Cipher

- An alphabet substituted by a different alphabet
  - Eg: A by F, B with Y etc.

```
key = {
    "a": "f", "b": "y", "c": "a", "d": "b", "e": "z", "f": "c",
    "g": "m", "h": "s", "i": "n", "j": "t", "k": "o", "l": "h",
    "m": "q", "n": "v", "o": "r", "p": "x", "q": "w", "r": "i",
    "s": "k", "t": "u", "u": "l", "v": "j", "w": "p", "x": "g",
    "y": "d", "z": "e"
}

# The message we want to encrypt
message = "Welcome to Information Security at IIT KGP"

# Remove non-alphabetic characters and convert to lowercase
secret = filter(str.isalpha, message.lower())

# Encrypt the message by mapping each character
encrypted = "".join([key[i] for i in secret])
print(encrypted)
# Outputs: pzharqzurnvcriqfunrvkzalinudfunnuomx
```

Solving Substitution Cipher CTFs

- Language Entropy:
  - "*predict*" the occurrence of a certain letter based on it's usage
  - Ever observed “vowels are used in most words”?

- Can you solve the following puzzle with this hint?

  “Rbo rpktigo vcrb bwucja wj kloj hcjd, km sktpqo, cq rbwr loklgo vcgg cjqcqr kj skhcja wgkja wjd rpycja rk ltr rbcjaq cj cr”

Caesar Cipher

Core Idea: Shifting the alphabets by a fixed amount

- Basically, `enc(ch) = (ch + shift)%26`
- Eg: If shift = 8, the alphabets: ABCDEFGHIJKLMNOPQRSTUVWXYZ becomes IJKLMNOPQRSTUVWXYZABCDEFGH

How to solve these ? Brute Force (try with all shifts from 0 to 25)

Try this: “`iwtgt xh cd gxvwi pcs lgdcv. iwtgth dcan ujc pcs qdgxcv.`”

There are many more such ciphers like Vigenere Cipher, Hill Cipher etc.

Symmetric Encryption

AES - Advanced Encryption Standard

- One of the most commonly used symmetric encryption schemes
  - 128 bits plain text and 128/192/256 bit keys
  - Resistant against known attacks
  - Both software and hardware friendly

- Encryption process is spread over multiple rounds, which perform similar operations (10 rounds for 128-bit key).

- AES decryption process is very similar to the encryption process, except that inverses of each step are used and key schedule is different.

Modes of Operation : Block ciphers

- Package a message into evenly distributed 'blocks' which are encrypted one at a time.
- A sequence of bytes, called Initialization Vector (IV), is used to randomize encryption even if the same plaintext is encrypted.
- If the length of message is not an exact multiple of the block size, padding may be necessary for the last block.

Modes of Operation : Block ciphers

- Common types and some vulnerabilities:
  - Electronic Codebook Book (ECB)
    - Decrypt suffix (data after plaintext)
  - Cipher Block Chaining (CBC)
    - Bit-flipping Attack
    - Padding Oracle Attack
  - Cipher FeedBack (CFB)
    - Predictable Output
  - Counter (CTR) Mode
    - Known Plaintext Attack

Modes of Operation : Stream ciphers

- Encrypt by XORing pseudorandom sequences with bits of plaintext in order to generate ciphertext.

- Two types:
  - Synchronous : Generates keystream based on internal states not related to the plaintext or ciphertext.
  - Self-synchronizing / Asynchronous : uses the previous N digits in order to compute the keystream used for the next N characters.

Modes of Operation : Stream ciphers

- Used for continuous data streams or when actual length of message to be transmitted is unknown, e.g., audio or video encryption during transmission.

- Vulnerabilities:
  - Key Reuse
  - Bit-Flipping Attack

Asymmetric Encryption

Asymmetric Encryption

- Each user generates two keys: public key (**pk**) and private key (**pr**)

- Suppose Alice needs to send a message to Bob
  - Alice encrypts the message with Bob's public key **pk_Bob**
  - Bob decrypts with its secret key **pr_Bob**

Asymmetric Encryption

- Example: RSA

- Three procedures : **KeyGen, Encrypt,** and **Decrypt**
  - **KeyGen**: Generate public key **pk** and private key **pr**
    - Generate *large* primes **p** and **q**
    - **pk** = (**N, e**), where **N = p** * **q**
    - **pr** = **d** such that **e . d =** 1 mod **ϕ(N)**, where **ϕ(N)** = (**p** - 1) (**q** - 1)

Asymmetric Encryption

- Example: RSA

- Three procedures : **KeyGen, Encrypt,** and **Decrypt**
  - **Encrypt**: To encrypt a message **m**, **c = m ^ e** mod **N**

  - **Decrypt**: To decrypt, compute **c ^ d** mod **N**

Asymmetric Encryption

- Issues to avoid in RSA
  - Primes **p** and **q** need to be *large*. Otherwise factorization is possible!
  - **e** and **ϕ(N)** need to be co-prime.
  - Avoid small values of **d**.
  - Avoid reusing randomness in **KeyGen**.