SOS Cryptography

Kavin Arvind

Presented by: Kavin Arvind (22B1019)
Mentored by: Nilabha Saha

Contents

1 Intoduction

Cryptography refers to secure information and communication techniques derived from mathematical concepts and a set of rule-based calculations called algorithms, to transform messages in ways that are hard to decipher. In this digital era, as the usage of digital information has increased exponentially, it is also important to secure them. Each time you make an online purchase, conduct a banking transaction, or ping your email client, cryptography is working in the background. It secures all transmitted information in our IoT world, to authenticate people and devices, and devices to other devices.

2 What is cryptography?

The art of scrambling of data using a key and an algorithm so that a third party cannot understand the data and unscrambling it again for the authorised person to understand is called cryptography. Let's see its types:

2.1 Symmetric and assymetric encription

Symmetric encription: Symmetric encryption, also known as secret-key or shared-key encryption, uses a single secret key to both encrypt and decrypt data. The same key is used by both the sender and the recipient. The key needs to be kept secret and securely shared between the communicating parties.

Assymetric encription: Asymmetric encryption, also called public-key encryption, employs a pair of mathematically related keys: a public key and a private key. The public key is freely distributed and used for encryption, while the private key remains secret and is used for decryption.

2.2 Stream Ciphers and Block Ciphers

Stream ciphers: Stream ciphers encrypt bits individually. This is achieved by adding a bit from a key stream to a plaintext bit. Thus the ciphertext depends only on the keystream and individual bits and does not depend on other bits in the plaintext.

Block Ciphers: Block ciphers encrypt an entire block of plaintext bits at a time with the same key. This means that the encryption of any plaintext bit in a given block depends on every other plaintext bit in the same block.

Refer section 5 for more depth.

3 Early methods

Ancient ciphers date back to Egypt, where substitution ciphers were used.

3.1 Caesar's cipher

Julius Caesar encrypted by rotating the letters of the alphabet by 3 places i.e, a was replaced with D, b with E, and so on. Of course, at the end of the alphabet, the letters wrap around and so x was replaced with A, y with B and z with C. Here, 3 is the key.
Encryption: $y_i = x_i + k \pmod{26}$
Decryption: $x_i = y_i - k \pmod{26}$
k - key
y_i - individual cypher text letters

x_i - individual plain text letter.

3.1.1 Attack method:

An eavesdropper can check by rotating by different keys, and by brute force method, he can find the key and find the plain text as there are only 26 keys available in this method. Thus, we need a better method to encrypt.

3.2 Mono-alphabetic substitution

Each and every letter in english alphabet are substituted by some other alphabet and the mapping is the key for the encription. Thus, there are totally 26! keys available which is approximately, 2^{88}. A brute force attack on the key space for this cipher takes much longer than a lifetime, even using the most powerful computer known today.

3.2.1 Attack method:

Though there are 26! keys, remember that the hacker won't take method that we expect them to take. They take the smater way. The attack works by tabulating the probability distribution of the ciphertext and then comparing it to the known probability distribution of letters in English text.(refer fig[1])

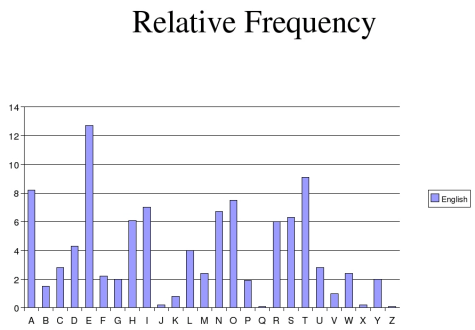


Figure 1: Average letter frequencies in the English language

Associate the letters of the English alphabet with the numbers $0, \ldots, 25$. Let p_i, for $0 \leq i \leq 25$, denote the probability of the ith letter in normal English text.

$$\sum_{i=0}^{25} p_i^2 \approxeq 0.065$$

Let q_i denote the probability of the ith letter in this ciphertext. Then, for key k for each alphabet in cyphertext q_i, $I_k = \sum_{i=0}^{25} p_i q_{i+k(mod 26)} \approxeq 0.065$. Thus, corresponding keys can be found out.

3.3 The Vigen'ere (poly-alphabetic shift) cipher

In this method, we would have a secret word(not necessarily meaningful) as a key. It is repeated many times, and added to the plain text to get the ciphertext. In the given example, the key

word *cake* is repeated and added to the plain text to obtain the ciphertext. This method provides a smoothening effect on the probability distribution of letters in ciphertext as at different places, different letters are mapped to different letters.

Plaintext	A	L	L	I	S	W	E	L	L
Key	C	A	K	E	C	A	K	E	C
Ciphertext	C	L	V	M	U	W	P	P	N

Figure 2: Vigen'ere cipher

3.3.1 Attack method:

For $\tau = 1, 2, \ldots$ we look at $c_1, c_{1+\tau}, c_{1+2\tau}, \ldots$, analyse its letter probability distribution like the previous section and find the key word.
These were refered from [3]

4 Modular Arithematic

Modular Arithematic plays a crucial role in understanding cyptography and verifying its reliability. Thus, we introduce some important properties and theorems of it.

Theorem 4.1. *Let a and b be positive integers. Then the equation*

$$au + bv = gcd(a, b)$$

always has a solution in integers u and v. For proof, refer [2]

Theorem 4.2. *Let a be an integer. Then $a.b \equiv 1(mod\ m)$ for some integer b if and only if gcd(a,m)=1 . Here, b is called the (multiplicative) inverse of a modulo m. For proof, refer [2] Here, we note that the inverse element is unique if it exists.*

Theorem 4.3 (Fermat's Little Theorem)**.** *Let p be a prime number and let a be any integer. Then,*

$$a^{p-1} \equiv \begin{cases} 1(modp), if\ p \nmid a, \\ 0(modp), if\ p \mid a. \end{cases}$$

5 Stream Ciphers and Block Ciphers

5.1 Stream Ciphers

Definition 5.1. *The plaintext, the ciphertext and the key stream consist of individual bits, i.e., $x_i, y_i, s_i \in 0, 1$.*
Encryption: $y_i = e_{s_i}(x_i) \equiv x_i + s_i\ mod\ 2$.
Decryption: $x_i = d_{s_i}(y_i) \equiv y_i + s_i\ mod\ 2$.

5.1.1 Random number generation

- **True Random Number Generators (TRNG):** Numbers produced are truely random and it is not possible to calculate the outcome. These are based on physical processes

like coin flipping, rolling of dice, semiconductor noise, clock jitter in digital circuits and radioactive decay. True random numbers are necessary to produce unpredictable keys to be distributed to the sender and reciever.

- **(General) Pseudorandom Number Generators (PRNG):** They are calculated from an intitial seed value. Knowing the intitial seed value allows us to calculate the rest of the outcomes. Eg-
  S_0 = seed
  $S_{i+1} = AS_i + B \bmod m$,$i = 0, 1, \ldots$
  Let's say A and B are secret. By knowing first 4 outputs, A and B can be uniquely determined and key stream pproduced by PRNGs can be easily determined.

- **Cryptographically Secure Pseudorandom Number Generators (CSPRNG):**
  CSPRNGs are a special type of PRNG which prodece unpredictable numbers i.e, given n output random numbers produced, its not possible to calculate the next number produced. Eg-
  Linear Feedback Shift Registers (LFSR): LFSR is initially loaded with initial values $s_{m-1}, \ldots, s_0$. the output sequence can be described as:
  $s_{i+m} \equiv \sum_{j=0}^{m-1} p_j.s_{i+j} \bmod 2$; $s_i, p_j \in 0, 1$; $i = 0, 1, 2, \ldots$

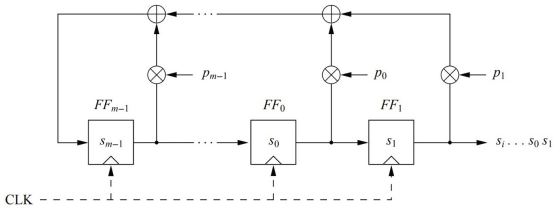


Figure 3: General LFSR with feedback coefficients p_i and initial values $s_{m-1}, \ldots, s_0$

Here, if the eavesdropper knows 2m output bits of an LFSR of degree m, the pi coefficients can exactly be constructed by merely solving a system of linear equations. Thus practically, combinations of many such LSFRs are used to build strong cryptosystems.

5.1.2 The One-Time Pad

Here, the key stream is produced are truely random and its size should be same as the plaintext size. New key should be used everytime a plaintext is sent. These makes it impractical. But otherwise, it is theoretically a very safe encription. We can prove easily that it is unconditionally secure. For every ciphertext bit we get an equation of this form:
$y_0 \equiv x_0 + s_0 \bmod 2$
$y_1 \equiv x_1 + s_1 \bmod 2$
.
.
Each individual relation is a linear equation modulo 2 with two unknowns. They are impossible to solve. Thus it is theoretically very secure.
Let us see some practical stream Ciphers.

5.1.3 Trivium

Trivium is a relatively new stream cipher which uses an 80-bit key. It is based on a combination of three shift registers. Even though these are feedback shift registers, there are nonlinear components used to derive the output of each register, unlike the LFSRs that we studied in section

5.1.1.

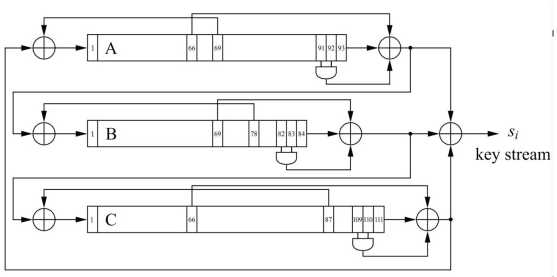


Figure 4: Internal structure of the stream cipher Trivium

As shown in figure 4, at the heart of Trivium are three shift registers, A, B and C. The lengths of the registers are 93, 84 and 111, respectively. The XOR-sum of all three register outputs forms the key stream s_i.

Encryption with Trivium:
We use Initialization Vector(IV) that serves as a randomizer and should take a new value for every encryption session. The key stream is produced dependent on key and IV. IV does not have to be kept secret, it merely must change for every session so that same plaintext is not encrypted to same ciphertext everytime and knowledge about keystream of one session doesn't give knowledge of other session's key stream and the key.
The steps are as follows:-

- **Initialization:** Initially, an 80-bit IV is loaded into the 80 leftmost locations of register A, and an 80-bit key is loaded in the 80 leftmost locations of register B. All other register bits are set to zero with the exception of the three rightmost bits of register C, i.e., bits c_{109}, c_{110} and c_{111}, which are set to 1.
- **Warm-up Phase:** In the first phase, the cipher is clocked $4\times288 = 1152$ times. No cipher output is generated.
- **Encryption Phase:** The bits produced hereafter, i.e., starting with the output bit of cycle 1153, form the key stream.

5.2 Block Ciphers

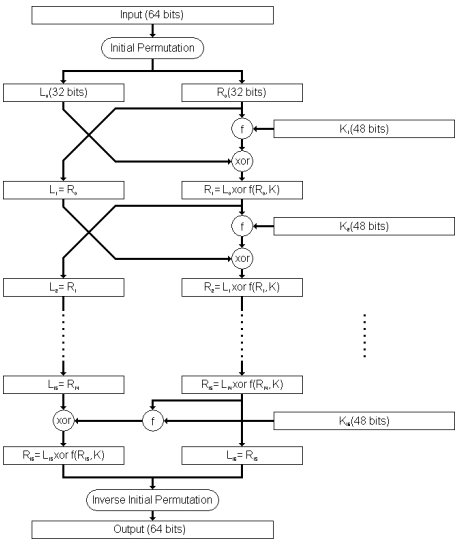
Block ciphers are a class of symmetric encryption algorithms that operate on fixed-size blocks of data. A block cipher divides the plaintext into fixed-size blocks, typically 64 or 128 bits, and encrypts each block separately. The encryption process involves a series of transformations applied to each block using a secret encryption key.

5.2.1 The Data Encryption Standard (DES)

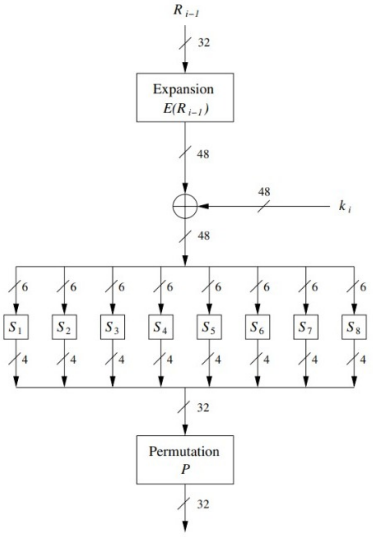
- **Confusion** is an encryption operation where the relationship between key and ciphertext is obscured.
- **Diffusion** is an encryption operation where the influence of one plaintext symbol is spread over many ciphertext symbols with the goal of hiding statistical properties of the plaintext.

Key Features and Characteristics:
DES operates on 64-bit blocks and employs a 56-bit key. It uses a Feistel network structure,

where the input block undergoes a series of transformations through multiple rounds. These transformations include substitution boxes (S-boxes) and permutation boxes (P-boxes), providing confusion and diffusion properties.



(a) Flowchart of the 16 rounds



(b) F function

Disadvantage:
DES is of short key length of 56 bits. With advances in computing power, it has become feasible to launch brute-force attacks against DES by trying all possible keys.
Over time, various cryptanalysis techniques have been developed that can exploit specific weaknesses in the DES algorithm. Differential cryptanalysis and linear cryptanalysis are two such techniques that can reduce the effective security of DES.
Thus modified versions of DES were intoduced.

5.2.2 Advanced Encryption Standard (AES)

AES, selected as the successor to DES, offers key sizes of 128, 192, and 256 bits. It employs substitution and permutation operations on 128-bit blocks and operates on a larger number of rounds compared to DES. AES has gained widespread adoption and is considered highly secure.

5.2.3 Triple DES (3DES)

3DES applies DES three times using two or three different keys. It operates on 64-bit blocks and supports key sizes of 112 and 168 bits. While providing improved security over DES, 3DES is computationally more expensive.

5.2.4 Electronic Codebook Mode (ECB)

Each plaintext block is individually encrypted with key k by function $e()$.
Encryption: $y_i = e_k(x_i)$,
Decryption: $x_i = e_k^{-1}(y_i)$
Very simple algorithm and so if some plaintexts and corresponding ciphertexts are known, it is easy to know about key k and $e()$.

5.2.5 Cipher Block Chaining Mode (CBC)

Here, the encryption of all blocks are "chained together" such that ciphertext y_i depends not only on block x_i but on all previous plaintext blocks as well. The encryption is also randomized by using an initialization vector (IV). IV is used at the first step of encryption and decryption and thus the randomization is carried over to all the blocks.

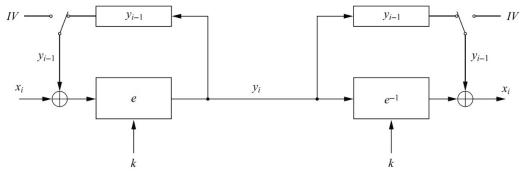


Figure 6: Encryption and decryption in CBC mode

Every block depends on every other previous blocks as the encryption is done recursively.

5.2.6 Output Feedback Mode (OFB)

OFB is a mode of operation that transforms a block cipher into a synchronous stream cipher. It generates a keystream by encrypting an initialization vector (IV) using the block cipher. The keystream is then combined with the plaintext using a bitwise XOR operation to produce the ciphertext. OFB provides keystream independence, error propagation containment, and efficient random access.

5.2.7 Cipher Feedback Mode (CFB)

CFB is a mode of operation that turns a block cipher into a self-synchronizing stream cipher. It encrypts the previous ciphertext block to generate the keystream, which is then XORed with the plaintext to produce the ciphertext. CFB provides error propagation containment, random access, and support for smaller block sizes compared to OFB.

5.2.8 Counter Mode (CTR)

CTR is a mode of operation that uses a counter as the input to the block cipher. The counter values are encrypted to generate the keystream, which is then XORed with the plaintext to produce the ciphertext. CTR mode offers parallel encryption and decryption, efficient random access, and the ability to support high-speed encryption and decryption operations.

5.2.9 Galois Counter Mode (GCM)

GCM combines the Counter Mode (CTR) with the Galois Field Multiply (GFM) operation to provide authenticated encryption. It offers confidentiality, integrity, and authenticity of the data. GCM mode uses a counter for encryption, a message authentication code (MAC) for integrity protection, and additional authenticated data (AAD) for additional security requirements.
Refered from [1]

References

[1] Jan Pelzl Christof Paar. *Understanding Cryptography: A Textbook for Students and Practitioners*.

[2] Joseph H. Silverman Jeffrey Hoffstein Jill Pipher. *An Introduction to Mathematical Cryptography*. Ed. by Second Edition.

[3] Yehuda Lindell Jonathan Katz. *Introduction to Modern Cryptography*. Ed. by First Edition.