Cryptography and its Types | GeeksforGeeks

Modular Arithmetic and Cryptography!

Caesar Cipher in Cryptography | GeeksforGeeks

Implementation of Affine Cipher | GeeksforGeeks

Vigenère Cipher | GeeksforGeeks

What is Steganography? | GeeksforGeeks

Modular arithmetic can be used to implement basic encryption algorithms like the Caesar cipher and the Affine cipher. These ciphers rely on shifting letters in the alphabet based on a mathematical key. Modular arithmetic ensures that the shifted letter stays within the bounds of the alphabet.
Caesar Cipher
The Caesar cipher is a simple substitution cipher where ea
ch letter in the plaintext is shifted a certain number of positions down the alphabet.
Encryption:

1. Convert letters to numbers: Assign numerical values to each letter (A=0, B=1, ..., Z=25).
2. Shift the numbers: Add the key (the shift value) to each letter's number.
3. Apply modular arithmetic: Take the result modulo 26 (e.g., 32 mod 26 = 6).
4. Convert back to letters: Find the letter corresponding to the resulting number.

Decryption: The reverse of encryption. Subtract the key, take modulo 26, and convert back to letters.
Example:

- Plaintext: "HELLO"
- Key: 3
- Encryption:
    - H (7) + 3 = 10 (K)
    - E (4) + 3 = 7 (H)
    - L (11) + 3 = 14 (O)
    - L (11) + 3 = 14 (O)
    - O (14) + 3 = 17 (R)

- Ciphertext: "KHOOR"
- Decryption:
    - K (10) - 3 = 7 (H)
    - H (7) - 3 = 4 (E)
    - O (14) - 3 = 11 (L)
    - O (14) - 3 = 11 (L)

    - R (17) - 3 = 14 (O)

• Plaintext: "HELLO"

Affine Cipher
The Affine cipher is a more complex substitution cipher that uses both a multiplicative and an additive key.
Encryption:

1. Convert letters to numbers: As in the Caesar cipher.
2. Multiply by the multiplicative key (a): Multiply each letter's number by the key (a).
3. Add the additive key (b): Add the key (b) to the result.
4. Apply modular arithmetic: Take the result modulo 26.
5. Convert back to letters: As in the Caesar cipher.

Decryption:

1. Find the multiplicative inverse of 'a' (a^-1) modulo 26: 'a' and 26 must be relatively prime (their greatest common divisor is 1). The inverse is the number that, when multiplied by 'a', gives 1 modulo 26.
2. Multiply by the inverse: Multiply the ciphertext number by the inverse of 'a'.
3. Subtract the additive key (b): Subtract the additive key (b).
4. Apply modular arithmetic: Take the result modulo 26.
5. Convert back to letters: As in the Caesar cipher.

Example:

• Plaintext: "HELLO"
• Multiplicative key (a): 5
• Additive key (b): 8
• Encryption:
    - H (7) * 5 + 8 = 43 mod 26 = 17 (R)
    - E (4) * 5 + 8 = 28 mod 26 = 2 (C)
    - L (11) * 5 + 8 = 63 mod 26 = 11 (L)
    - L (11) * 5 + 8 = 63 mod 26 = 11 (L)
    - O (14) * 5 + 8 = 78 mod 26 = 0 (A)

• Ciphertext: "RCLLA"

Briefly, how the Enigma encodes:
The operator sets some initial settings:

The plugboard wiring

The selection of rotors and reflector

Initial versions had three rotors and one reflector

Later versions had capacity for more rotors

Each rotor and reflector is essentially a substitution cipher

Each rotor has their own predefined notch points where additional rotor stepping occurs

The rotor ring settings (Ringstellung)

The rotor positions / rotor offset (Grundstellung)

The operator uses a keyboard to type their message, and the resulting encrypted letter would be highlighted for them.

Key understandings when trying to program an Enigma machine:
Assuming a set up of rotors I, II, III (III being the right most / first rotor the signal is passed through).

Each letter pressed steps the first (right most) rotor by one step. This moves the rotor offset by one.

The rotor offset is essentially a caesar cipher effect on the input. So letter 'A' is for example shifted by one letter to 'B' on the first input before it is run through the rotor's substitution cipher.

The ring settings are also essentially a caesar cipher that affect the input before it is run through the rotor's wiring / substitution cipher. The ring settings offset the internal wiring (substitution cipher) of the rotors.

The rotor offset and ring setting offsets need to be reversed before the final output is given to the next rotor. Each rotor will need their own rotor offset and ring setting offsets calculated in this way...

Understanding the notch points. At the notch point, the rotor steps the rotor offset of the following rotor by one. Note: in traditional enigma machines, the rotors after the 3rd rotor do not rotate. The double step effect occurs specifically for the middle rotor. This is a specific condition where the middle rotor will rotate once because the right most rotor is at its notch point (and therefore steps the rotor offset of the following rotor by one), and this rotation will bring the middle rotor into its notch point; the middle rotor will then rotate again on the next letter input because the middle rotor itself is at its notch point. This is what is meant by the double step effect.

e.g. you have the rotor positions of Q, D, V (V is the notch point for the right-most rotor, or the first rotor to be passed, E is the notch point for the middle rotor).

V will become W (as the right most rotor always rotates), and as it is at its notch point, it will rotate the middle rotor from D to E which is the middle rotor's notch point.

Now with notch position Q, E, W, on the next letter, the middle rotor will rotate itself, and it will step the rotor offset of the following rotor by one (becoming R, F, X). This is the unique double step effect where the rotor itself will rotate when it is at its notch point even while the preceding rotor is not at its notch point.

See this video for a physical demonstration of why this happens at 0:28 timestamp
https://youtu.be/hcVhQeZ5gI4?t=28

(run it at slower speed to analyse it).

https://en.m.wikipedia.org/wiki/Enigma_rotor_details