Caesar Cipher

Simple Substitution Cipher

Keane J. Moraes

1. History

The Caesar cipher is one of the simplest and most widely known encryption techniques. It was named after Julius Caesar (100BC - 44BC) who implemented the cipher to communicate with his generals on the battlefield. According to the Roman historian Gaius Suetonius Tranquillus, in his book *De vita Caesarum*, Caesar used a shift of 3 (A→D).
One can assume that this cipher was reasonably secure in those days due to the fact that most of Caesar's enemies were illiterate or assumed that the text was some strange foreign language. The earliest record of cryptanalysis of simple substitution ciphers was **frequency analysis** given through the works of Al-Kind of the Arab World in the 9th century AD.

2. Modern Usage

Nowadays, the Caesar cipher offers little to no security as an encryption scheme. A few relatively modern uses of the Caesar cipher are discussed below.
During WW1, the Russian army used the Caesar cipher to communicate troop movements because their proprietary service code *(RSK)* with daily key shifts was too complicated for the troops to use. German and Austrian cryptanalysists had a field day decrypting this scheme. They were able to predict the Russian invasion of East and West Prussia. Colonel Max Ronge, head of Austro-Hungarian intelligence called this "the most brilliant period of our interception services".[1]

Another prominent modern use of this cipher was in 2006. The fugitive Mafia don, Bernardo Provenzano, implemented an inept version of the Caesar cipher where letters of the alphabet were encoded as numbers (A → 4; B → 5 ...). He was soon captured in Sicily not long after[2].

Implementations of the cipher are present in other schemes like the Vigenère cipher and the modern ROT13 cipher.

[1] Kahn, David (1967). The Codebreakers. pp. 348–50. ISBN 978-0-684-83130-5.
[2] Leyden, John (2006-04-19). "Mafia boss undone by clumsy crypto". The Register.

3. Encryption and Decryption Algorithms

The encryption and decryption algorithms for the Ceasar Cipher are pretty straightfoward. To encrypt shift certain units to the right and to decrypt shift those same units to the left. But the tricky part is doing a 'wrap-around' without `if` statements.

3.1. Encryption

Encryption of the plain text is done by shifting all the letters to the right of their respective letter number* by a given shift length. Any letter numbers that exceed 26 are wrapped around.
*Letter Number - an ordered numbering of the alphabet. ($A \to 0, B \to 1 \ldots Z \to 25$). We will perform a letter-by-letter encryption. We must take into account the conversion of the ASCII character values $\big(a(97) - z(122)\big)$ or $\big(A(65) - Z(90)\big)$ into the standard format $(0-25)$ else the modular arithmetic will not work with a modulo of 26.
Let E_t be the cipher text.
P_t be the plain text having length 'n'.
S be the shift, where $S \in \mathbb{N}$
$E_t = E_1E_2E_3 \ldots E_n$ where E_i is the i^{th} character of the cipher text
The i^{th} character of the cipher text is computed -
If the letter is between 65 and 90 (Uppercase)

$$E_i = \Big[\big((P_i - 65) + S\big) \mod 26\Big] + 65$$

If the letter is between 97 and 122 (Lowercase)

$$E_i = \Big[\big((P_i - 97) + S\big) \mod 26\Big] + 97$$

The ± 65 and ± 97 merely bring the ASCII character values to 0-25 range. We can then perform the necessary operation and raise them back up to their ASCII values so they can be converted back into characters.
What does **mod** mean? Refer to the Why Modular Arithmetic section

3.2. Decryption

Decryption of the plain text is done by shifting all the letters to the right of their respective letter number by the given shift length. Any numbers that are below 0 are wrapped around. As always, we must convert the letters into standard format(0-25)
Let E_t be the cipher text having length of n
P_t be the plain text
S be the shift, where $S \in \mathbb{N}$
$P_t = P_1P_2P_3 \ldots P_n$ where P_i is the i^{th} character of the plain text
The i^{th} character of the plain text is -
If the letter is between 65 and 90 (Uppercase)

$$P_i = \Big[\big((E_i - 65) - S\big) \mod 26\Big] + 65$$

If the letter is between 97 and 122 (Lowercase)

$$P_i = \Big[\big((E_i - 97) - S\big) \mod 26\Big] + 97$$

NOTE
The % operator in Java computes remainder and **NOT** the modulus. This means that for negative operand, the % will not work properly

For e.g: $-2\%3 = -2$

$-2 \mod 3 \equiv 1$

To account for this, the following correction is made :

```
int a = operand;
int m = modulus;
int result = ((a%m)+m)%m;
```

Other languages like C, C++, Javascript and Perl use the % operator as a division remainder operator too. Thus one must be careful in implementation.

4. Why Modular Arithmetic?

We use modular arithmetic because it simplifies the action of wrapping around.
Imagine a circle of m units numbered from 0 to $(m-1)$ where m represents the modulus. Let us take a rope of a units and slowly 'wrap' this rope around the circle. The number that we eventually land on 'b' is the value of $a \mod m$.[3] So even if a exceeds m, the modulo operation automatically performs the wrapping of the operand. This works even if we go below 0. If we did not use modular arithmetic, we would have to add if statements to account for the excess to be wrapped, making that part of the code extraneous.

5. Variants

Only the Affine cipher is discussed here. ROT13 is found in a separate documentation.

5.1. Affine Cipher

This variant uses 2 more parameters -'a' and 'b' for encryption and decryption. These parameters are kept constant for all shifts. The encryption and decryption algorithms for upper case letters are listed below in order :

$$E_i = \Big[\big(a(P_i - 65) + b\big) \mod 26\Big] + 65$$

$$P_i = \Big[\big(c(E_i - 65) - b\big) \mod 26\Big] + 65$$

Here $a \cdot c \equiv 1 \mod 26$. Here instead of one value begin the shift it is a pair of values (a, b). Being a monoalphabetic cipher, it inherits the weaknessed of the class. It is still prone to attacks from frequency analysis and brute force attacks.

[3] Burton, David (1976). "Basic Properties of Congruence". Elementary Number Theory. pp. 63-76. ISBN 978-0-073-38314-9

6. Further Reading

For Modular Arithmetic :

"The Theory of Congruences". Elementary Number Theory by David Burton.

High School Mathematics Extensions is a great Wikibooks article for those interested in learning concepts outside the high school maths syllabus.

For Frequency Analysis :

"Cryptanalysis of Simple Substitution Ciphers". Introduction to Mathematical Cryptography by Hoffstein, Pipher & Silvermann.