Cryptosaurus manual

Cryptosaurus user manual

Sweb

July 2023

1 Credits

Programming : Sweb
Dinosaur ASCII art : Christopher Johnson (This ASCII pic can be found at `https://www.asciiart.website/index.php?art=animals/reptiles/dinosaurs`)

2 Introduction

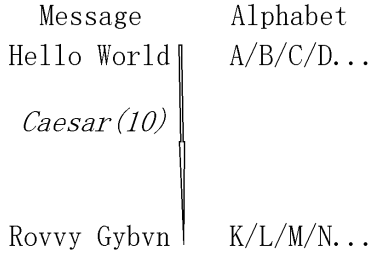
Cryptosaurus is a simple Python application that allows users to easily encode and decode messages to share with their friends for fun.

The goal of this manual is to explain the different functionalities of the software.

3 Caesar offset

This is one of the oldest and easiest ways to encode a message. It takes an integer n as input and modifies the alphabet by shifting all letters by n steps.

Figure 1: Example of a Caesar encoding

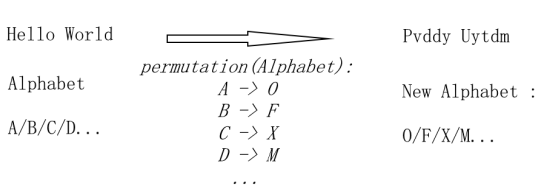


To decrypt the message, one only needs to have the value of n, so that the inverse operation can be applied.

4 Permutations

This is similar to the Caesar offset, in the sense that one uses a new alphabet, which is a function of the old alphabet, to crypt the message.

Figure 2: Example of a permutation encoding



4.1 Random permutation

Cryptosaurus can generate a random permutation to encrypt the message. In that case, the user needs to respond "n" to "Generate random permutation ?".

4.2 Custom permutation

If "y" is entered, it will be taken from `perm/custom_permutation.json`. This file contains an array of characters that the sender can edit freely to create their own permutation. For example, if the user writes "R" as the first element of the array, all A letters will be replaced by R. *Of course, there must be a one-to-one correspondence*, i.e there cannot be twice the letter R in the array, for instance (unless you deliberately want to create ambiguities, in that case only the first instance of the duplicate letter will be properly translated).

5 RSA

RSA (Rivest–Shamir–Adleman) is a public-key cryptosystem, one of the oldest, that is widely used for secure data transmission (*source : Wikipedia*).

To crypt, simply write your message. You will be prompted to save 3 files where you desire. `RSA_Result.txt` (default name) contains both keys and the final crypted message. In the same directory, you will then find `private_key.pem` and `crypted_message.bin`. The .txt file is only here for your own information, and what the reciever actually needs to be sent are `private_key.pem` and `crypted_message.bin`.

To decrypt, you will need to have `private_key.pem` and `crypted_message.bin`, and will be prompted by the program to select them. If the files are valid, the decrypted message will be displayed.

6 Bijection

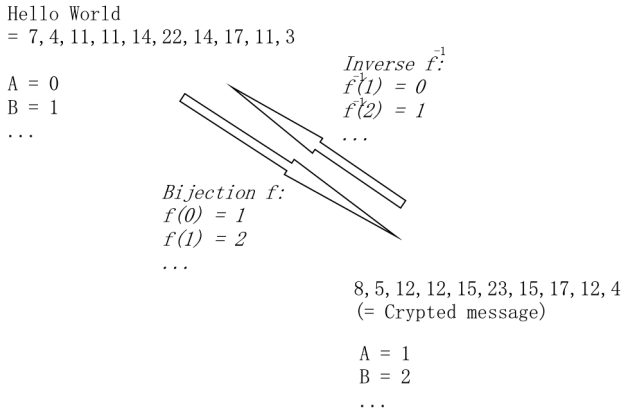
In mathematics, a bijection, also known as a bijective function, one-to-one correspondence, or invertible function, is a function between the elements of two sets, where each element of one set is paired with exactly one element of the other set, and each element of the other set is paired with exactly one element of the first set; there are no unpaired elements between the two sets.

In our case, we use bijections to encode an alphabet (left functions) and their inverse to decode the alphabet (right functions).

Figure 3: Currently available bijections

Crypt	inverse of ⟷	Decrypt
$f : \mathbb{Z} \to \mathbb{N}$ $k \to \begin{cases} 2k, & \text{if } k >= 0, \\ -(2k+1), & \text{if } k < 0. \end{cases}$		$f : \mathbb{N} \to \mathbb{Z}$ $l \to \begin{cases} l/2, & \text{if } l \text{ even}, \\ \frac{-l-1}{2}, & \text{if } l \text{ odd}. \end{cases}$
$f(x) = ae^{bx+c} + d$ $a \neq 0, b \neq 0$		$f(x) = \frac{\frac{ln(x-d)}{a} - c}{b}$ $a \neq 0, b \neq 0$
$f(x) = ax + b$ $a \neq 0$		$f(x) = \frac{x - b}{a}$ $a \neq 0$
$f(x) = ay^{bx+c} + d$ $a \neq 0, b \neq 0$		$f(x) = \frac{\frac{log_y(x-d)}{a} - c}{b}$ $a \neq 0, b \neq 0$

Figure 4: How the bijection encoding works. Every letter is mapped by a number, then run through the bijection to get a secret message. The inverse function gives us back the message.



6.1 Custom alphabet

Every letter is coded by a number. By default, these numbers are simply the letters' indexes in the alphabet array (A = 0, B = 1,...). However, the sender can choose to code each letter by whatever real number they want by editing the `bij/custom_alphabet.json` file. By definition, running the same bijection will not yield the same results if the alphabet is different. For instance, is we use the exponential function, if we decide that A = 0 then A will be crypted by exp(0) = 1. However if we decide that A = -10 then A will be crypted by exp(-10) = 0.000045... This is a good way to muddy the waters.

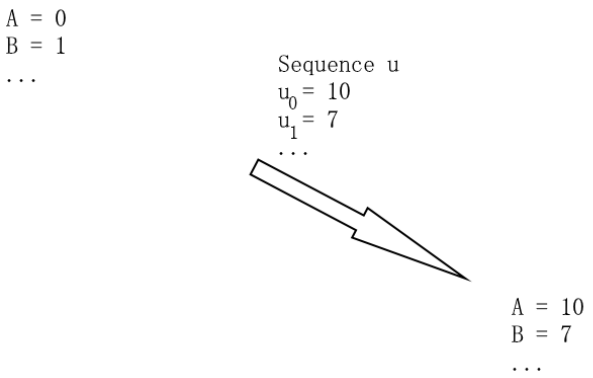
After the crypting is complete, the sender will be asked to save a file named `mapped_alphabet.json` on their computer. This file contains the image of the alphabet by the chosen bijection. To decrypt, the receiver will need to choose the appropriate inverse function, and then input the message as well as this file.

7 Sequence

This is similar to the bijection method as each letter is mapped by a number. What's special here is that the letters are coded by a sequence of the form $u_{n+2} = a \times u_{n+1} + b \times u_n$. The objective of the receiver is to find $A = u_0, B = u_1, ...$ To do so, they need to write u_n as

$$u_n = \lambda \times r_1^n + \mu \times r_2^n \text{ with } r_1 = \frac{a + \sqrt{a^2 + 4b}}{2}, r_2 = \frac{a - \sqrt{a^2 + 4b}}{2} \text{ and } \mu + \lambda = u_0$$

Figure 5: How the sequence encoding works.



Therefore, the sender will need to communicate a, b, λ and μ for the receiver to decrypt the message.