SYMMETRIC ALGEBRAIC CIPHER

Gutik O. V.[1], Popadiuk O. B. [1], Vlasov V. A. [1]

[1]*Ivan Franko National University of Lviv*
Lviv, Ukraine
oleg.gutik@lnu.edu.ua, olha.popadiuk@lnu.edu.ua,
vitaly.vlasov@lnu.edu.ua

Herein we describe a symmetric algebraic cipher utilizing matrix representation of message symbols. Code implementation is provided.

Keywords: algebraic, modulo, Rust, Python, symmetric cipher

Introduction

We propose a symmetric algebraic cipher with the following structure.

First, we define cipher parameters:

- an alphabet A with size $\mathrm{L} \equiv |\mathrm{A}|$;
- a matrix $M$with size $\mathrm{N} \gg \mathrm{L}$;
- σ_0, σ_1 are some permutations $\mathrm{N} \to \mathrm{N}$;
- ϕ is some bit sequence of prime length P: $\phi \in \{0,1\}^{\mathrm{P}}$.

A triple $(\sigma_0, \sigma_1, \phi)$ will be our secret key.

We define each symbol z by a corresponding set of diagonals D in the matrix M, so that

$$\forall(\mathrm{x}, \mathrm{y}) \in \mathrm{D}: \mathrm{x} - \mathrm{y} = \mathrm{z} \pmod{\mathrm{L}}$$

(see Figure 1).

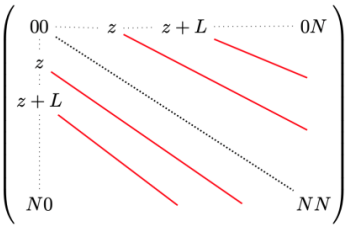

Figure 1

Encryption/Decryption algorithms

Here we describe the steps of the encryption algorithm.

Suppose we receive some text T containing symbols to be encoded.

For each $t_i \in T$, we first obtain its numeric representation $z_i \in [0, L)$.

Then we map each z_i to a pair of matrix coordinates (x_i, y_i) such that:

1. first, we pick a random $x_i \in [0, N)$ (e.g., horizontal coordinate in a matrix);
2. then, we randomly pick some $y_i \in [0, N)$ such that:

$$x_i - y_i = z_i \ (mod\ L).$$

Having thus obtained a sequence $\{(x_i, y_i),\ i \in [0, |T|)\}$, we now apply permutations $\sigma_k : [0, N) \to [0, N),\ k = 0,1$:

$$\text{ciphertext } (\xi_i, \eta_i) \ := \left(\sigma_{k_0}(x_i), \sigma_{k_1}(y_i)\right),$$

where

$$k_j = \phi[P\ mod\ (2 * i + j)].$$

Below are the steps of the decryption algorithm:

1. receive encoded ciphertext $\{(\xi_i, \eta_i)\}$;
2. apply inverse permutations: $(x_i, y_i) = \left(\sigma_{k_0}^{-1}(\xi_i), \sigma_{k_1}^{-1}(\eta_i)\right)$;
3. find z_i: $z_i = x_i - y_i \ (mod\ L)$.

Implementation

We have implemented a small library consisting of two modules:

- low-level Rust code for encryption/decryption parts;
- high-level Python wrapper.

These are implemented using PyO3 [1] - a library providing Rust bindings for Python extension modules. Its users include Qiskit, Python Cryptography package, and others.

To show an example of encoder output, consider the following cipher parameters:

- alphabet size: $L = 64$;
- matrix dimensions: $N = 1000$.

For these we obtain the following ciphertext:

Original string: Hello, world!

Generated pairs: [(60, 14), (746, 196), (489, 458), (620, 68), (947, 434), (943, 501), (18, 265), (913, 856), (196, 639), (965, 800), (28, 111), (693, 195), (22, 482), (614, 434), (367, 725), (932, 195), (102, 434), (388, 195)]

Conclusions

We have proposed and implemented an algebraic symmetric cipher relying on matrix representation of message symbols, using a set of permutations as its secret key. Python package with low-level Rust code is used for testing encryption and decryption algorithms.

References

1. https://github.com/PyO3/pyo3