Network Working Group
Internet Engineering Task Force (IETF)
Request for Comments: XXXX
Category: Informational

T. Developer
(Your Organization)
(Your Email)
(Today’s Date)

RFC SULA-ALUS: A REVERSAL ENCRYPTION ALGORITHM

Status of This Memo

Abstract

This document describes the SULA-ALUS encryption algorithm—a lightweight, reversible transformation whose principal operation is to reverse the input plaintext. The algorithm’s name pays homage to the Latvian wordplay where reversing “sula” gives “alus.” Although trivial in its operation, this algorithm may serve educational, demonstrative, or low-risk obfuscation purposes.

Table of Contents

1. Introduction

The SULA-ALUS algorithm is founded on a straightforward principle: encryption is performed by reversing the order of characters (or bytes) of the input data. While the transformation is simple and not intended for modern security-critical applications, it can be useful in contexts where a reversible obfuscation method is acceptable or for educational purposes.

2. Conventions Used in This Document

The key words “MUST,” “SHOULD,” and “MAY” are to be interpreted as described in RFC 2119 [1].

3. The Reversal Encryption Algorithm

The core idea behind SULA-ALUS is that the encryption operation is identical to the decryption operation – they both consist of reversing the input sequence.

3.1. Encryption Process

Let P represent the plaintext composed of a sequence of characters:

$$P = p_1, p_2, ..., p_n$$

The ciphertext C is produced by applying the reversal transformation:

$$C = \text{reverse}(P) = p_n, p_{n-1}, ..., p_1$$

For example, if $P = \text{sula}$, then:

$$C = \text{alus}$$

3.2. Decryption Process

Given that the reversal operation is an involution (its own inverse), the decryption process involves applying the same transformation. Let C be the ciphertext; then the plaintext P is recovered as:

$$P = \text{reverse}(C)$$

4. Implementation Considerations

A simple pseudocode implementation of the algorithm is as follows:

```
function encrypt(plaintext):
return reverse(plaintext)

function decrypt(ciphertext):
return reverse(ciphertext)
```

This algorithm may be implemented in any programming language. It is important to note that due to its simplicity, SULA-ALUS is only appropriate for applications where robust cryptographic security is not a requirement.

5. Security Considerations

The SULA-ALUS algorithm offers minimal protection and should not be used as a substitute for established cryptographic methods. Its reversible nature and lack of key-dependent randomness make it vulnerable to trivial analysis and brute-force reversal. Users are cautioned against employing this algorithm in scenarios where the confidentiality or integrity of sensitive data is paramount.

6. IANA Considerations

This document has no actions for the Internet Assigned Numbers Authority (IANA).

7. Acknowledgments

The authors thank the community for insights into minimalist cryptographic mechanisms and for inspiring creative approaches that celebrate cultural references—in this case, the reversal of the Latvian word “sula” to “alus.”

References

[1] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels.* 1997.