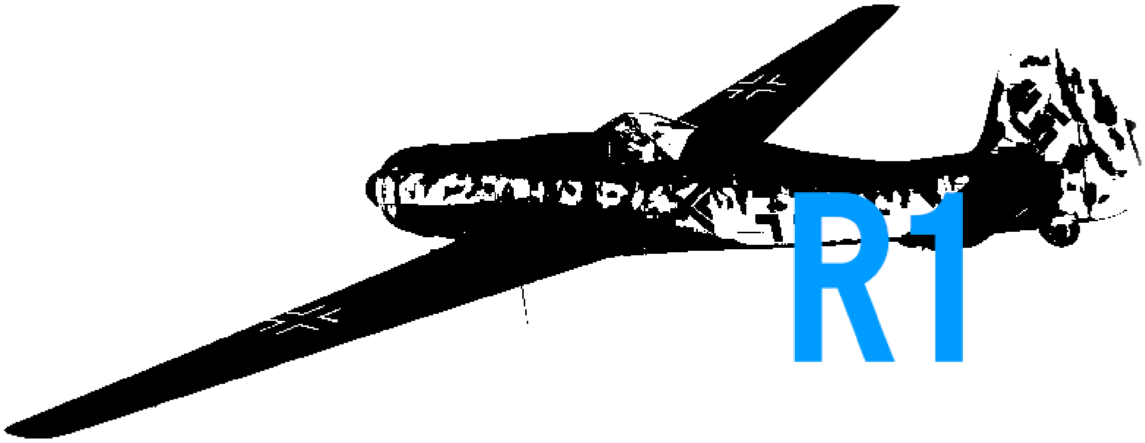
TA-152-R1

Authored and Designed By Riyan Dahiya



Cipher Specifications

Contents

1. About

TA-152-R1 is a custom symmetric encryption scheme, with an experimental scope. It operates using a streaming substitution cipher with a continuously evolving permutation state matrix, with attempts at optional keystream XOR-based confusion.

The design focusses on:

1. Per-byte state evolution.
2. Deterministic reversibility, with flexibility to achieve non-determinism using an IV.

3. Minimal runtime dependencies.
4. Explicit, inspectable behaviour.

THE ALGORITHM IS CURRENTLY EXPERIMENTAL, AND IS NOT SUITABLE FOR PRODUCTION CRYPTOGRAPHY.

2. Design Characteristics

a. Overview

TA-152-R1 is a stateful cipher, with an evolving 256-byte permutation. The state matrix covers all possible byte permutations *(0x00 – 0xFF)*. It uses a 128-bit key, processed one byte per round, and has a block size of 1 byte. The scheme allows for use of a randomly generated 16-byte IV to introduce non-determinism. Currently, this scheme only provides a degree of confidentiality, and lacks an authentication mechanism.

b. Key

Keys are 16-bytes in length, are consumed cyclically, and influence permutation evolution. They can also optionally influence keystream generation when used together with a 16-byte IV. Key reuse is allowed, although, is questionably effective without use of IV in keystream.

i. *Key Format:* Raw Binary (16 Bytes)
ii. *Weak Keys:* Not formally analysed

c. Internal State

The algorithm maintains the following internal state during operation:

i. `base_mx[256]`
Current substitution permutation

ii. `inverse_mx[256]`
Inverse substitution permutation, maintained incrementally

iii. `keypos`
Index for key byte *(mod 16)*

iv. `S`
Optional keystream

v. `Counter`
Optional counter with keystream

The state evolves with every byte processed.

d. Permutation Evolution

For each byte processed:

1. A round’s *chunk size* is derived from the key value.
   ```
   key = 0 or 1 >> chunk_size = 2
   Otherwise, chunk_size = key
   ```

2. The permutation array is partitioned into contiguous chunks.

3. Each chunk is *reversed* in place.

4. If any tail bytes remain, they are also reversed as a contiguous chunk.

The operation is involutive only when mirrored during decryption. Hence, strict ordering symmetry is required.

3. Encryption

For each plaintext byte `P`:

1. Update permutation using current key byte.

2. Substitute:
   ```
   C = base_mx[P]
   ```

3. If IV based keystream is enabled:
   ```
   C = C XOR S
   ```

4. Output `C`.

5. If IV based keystream is enabled, update keystream:
   ```
   1. S = (S * 131 + key_byte + (counter & 0xFF)) mod 256
   2. counter++
   ```

6. Advance key index modulo 16.

4. Decryption

For each plaintext byte `C`:

1. If IV based keystream is enabled:
   ```
   C’ = C XOR S
   ```

2. Update permutation using current key byte.

3. Substitute:
   `P = inverse_mx[C]`

4. Output `P`.

5. If IV based keystream is enabled, update keystream:

```
1. S = (S * 131 + key_byte + (counter & 0xFF)) mod 256
2. counter++
```

5. Initialization Vector

IV usage is controlled by `status` byte in header

- *i.* *IV Size:* 16 bytes
- *ii.* *Generation:* OS entropy (`getrandom`)
- *iii.* *Usage:*
  S_0 = key[0] XOR iv[0] XOR iv[1], counter = 0

When IV mode is disabled, keystream stage is skipped. The IV is stored, but is unused.

6. File Format

a. Header

Field	Size	Description
magic_number	4 bytes	0x54313532 *(T152)*
version	1 byte	VERSION NUMBER

status	1 byte	STATUS = 1 (USE IV)
		STATUS = 0 (DONOT USE IV)

iv	16 bytes	RANDOM
offset_a	4 bytes	RESERVED (FOR FUTURE)
offset_b	2 bytes	RESERVED (FOR FUTURE)
file_size	4 bytes	ORIGINAL PLAINTEXT SIZE

Header is written as a packed struct. Endian-ness is host-dependent (little-endian is assumed). Payload follows immediately after the header.

b. File Extension

The encryption function implemented appends `*.t152e` file extension to the encrypted file. This file extension is purely cosmetic, and is implemented for easier visibility of encrypted data.

The presence of this extension is trivial, and its absence in encrypted ciphertext does not affect the parsing of a file for decryption.