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 and ciphertext-feedback-based diffusion.

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.
5. Simple, byte-oriented operation without reliance on block primitives.

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

2. Design Characteristics

2.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 the use of a randomly generated 16-byte IV to introduce non-determinism. In addition to permutation evolution and optional keystream XOR, the cipher employs ciphertext feedback between successive bytes to increase diffusion across the plaintext stream.

Currently, this scheme only provides a degree of confidentiality, and lacks an authentication or integrity mechanism.

2.b. Key

Keys are 16 bytes in length, are consumed cyclically, and influence permutation evolution on a per-byte basis. Keys also influence keystream generation when used together with a 16-byte IV, and additionally seed the ciphertext feedback mechanism.

Key reuse is allowed, although it is questionably effective without the use of IV-based keystream and feedback seeding.

Key Format: Raw Binary (16 Bytes)

Weak Keys: Not formally identified

2.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 to allow efficient decryption.

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

iv. S
Optional keystream byte (enabled only when IV mode is active).

v. counter
Optional counter used for keystream evolution.

vi. mix_byte
Ciphertext feedback byte used to introduce inter-byte dependency.

The complete state evolves with every byte processed.

2.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 of chunk_size.

3. Each chunk is reversed in place.

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

The permutation update is involutive only when mirrored exactly during decryption. Hence, strict ordering symmetry between encryption and decryption is required.

3. Encryption

For each plaintext byte P:

1. A feedback-mixed input is computed:
   P' = P XOR mix_byte

2. Update permutation using the current key byte.

3. Substitute:
   C = base_mx[P']

4. If IV-based keystream is enabled:
   C = C XOR S

5. Output C.

6. Update feedback state:
   mix_byte = C

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

8. Advance key index modulo 16.

The ciphertext feedback causes each ciphertext byte to depend on all previous ciphertext bytes, increasing diffusion across the stream.

4. Decryption

For each ciphertext byte C:

1. If IV-based keystream is enabled:
   C' = C XOR S

2. Update permutation using the current key byte.

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

4. Recover plaintext:
   P = P' XOR mix_byte

5. Output P.

6. Update feedback state:
   mix_byte = C

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

8. Advance key index modulo 16.

Due to ciphertext feedback, decryption must proceed sequentially from the beginning of the stream.

5. Initialization Vector

IV usage is controlled by the status byte in the header.

IV Size: 16 bytes

IV Generation: OS entropy (getrandom)

Used for ciphertext and keystream initialization.

Keystream initialization:
S0 = key[0] XOR iv[0] XOR iv[1]
counter = 0

Ciphertext feedback initialization:
mix_byte = key[0] XOR iv[15]

When IV mode is disabled, the keystream stage is skipped, and the IV is zeroed and unused. Ciphertext feedback is seeded as: mix_byte = key[0]

6. File Format

6.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 *(don’t use IV)*
iv	16 bytes	RANDOM
offset_a	4 bytes	RESERVED *(for future features)*
offset_b	2 bytes	RESERVED *(for future features)*
file_size	4 bytes	ORIGINAL PLAINTEXT SIZE

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

6.b. File Extension

The encryption function appends a .t152e file extension to the encrypted file. This file extension is purely cosmetic, and is implemented for easier visibility of encrypted data. The presence or absence of this extension does not affect the parsing or decryption of a ciphertext file.

7. Notes and Limitations

The cipher exhibits catastrophic error propagation due to ciphertext feedback. A single corrupted ciphertext bit will corrupt all subsequent decrypted output. This also prevents random-access decryption.

The algorithm does not provide integrity, authenticity, and only provides limited tamper checking mechanisms. In its current state, the construction should be treated as experimental.