Fluid Sponge, SPN Cryptographic Core

Kirill Papka

October 2025

Overview

The idea behind the **Fluid Sponge Cipher** is to make the *fluid motion itself* act as the encryption. Each "stirring" step moves bytes or pixels around like a flow field. The motion depends on a secret key. After several rounds, the image looks completely mixed and unreadable. Because every operation is reversible, the receiver can "unstir" the data with the same key and recover the original image.

1. The Fluid Flow (Mixer)

Imagine the image as a grid of pixels floating on a pond. The secret key generates a *velocity field*—arrows showing how each pixel should shift.

To keep the system reversible, pixels never blend or merge. Instead, each round performs *integer circular shifts*:

- Each row is shifted left or right by a key-derived amount.
- Each column is shifted up or down by another key-derived amount.

Doing one horizontal shift and one vertical shift is one **fluid step**. Each step is perfectly reversible because we can shift back by the opposite amount.

This step provides the "motion" of the fluid.

2. The S-box (Color Twist)

If we only move pixels, the brightness pattern stays visible. After each flow step, we also change each pixel's value using a small **lookup table** (called an S-box).

The S-box is a shuffled list of all 256 possible byte values, created from the key. Each pixel's value is replaced by the corresponding value from this table. This adds a nonlinear change, similar to turbulence in a real fluid.

3. Mixing with Neighbors (Diffusion)

To ensure that small changes spread through the image, we mix nearby pixels. For example, we can take small 2×2 blocks and blend their four bytes with a fixed, reversible formula. The receiver can undo this mixing exactly, because the transformation has a mathematical inverse.

After several rounds, a single changed pixel influences many others.

4. Rounds

One **round** consists of:

1. The fluid shift (move pixels).
2. The S-box substitution (change values).
3. The local 2×2 mixing (diffusion).

All three operations are perfectly reversible, so the entire round is reversible. Typical implementations use 8–12 rounds with different key-derived flows and S-boxes. After all rounds, the image appears completely scrambled.

5. Encrypting Data (Sponge Mode)

The fluid mixer becomes the core of a **sponge** or **duplex** system:

1. Start with a secret internal state generated from the key and a unique *nonce*.
2. For each block of the file or image:
   - Combine (XOR) the block with part of the state (*absorb* step).
   - Run the fluid rounds to stir the state.
   - Take some bytes from the state as a *keystream* and XOR them with the block to form the ciphertext.
3. After the last block, extra bytes from the state can be taken as a **tag** (a checksum) to detect tampering.

Decryption performs the same sequence with the same key. Because the fluid process is deterministic and reversible, it regenerates the same keystream and recovers the original data.

6. What Makes It Fluid

- The pixel motion truly follows a key-generated velocity field.
- The flow can be run backward to decrypt the data.
- If intermediate rounds are animated, the encryption looks like waves mixing dye in water.
- Each round is a **bijective map**—no data is lost.

Thus, the encryption is both functional and fluid-like.

7. Summary of Components

Purpose	**How It Is Done**	**Why It Matters**
Move data like fluid	Key-derived horizontal and vertical shifts	Hides structure
Non-linear change	Byte substitution (S-box)	Prevents simple attacks
Spread influence	Mix 2×2 neighbor blocks	One change affects many bytes
Multiple rounds	8–12 repetitions	Ensures strong mixing
Reversibility	Each step has an exact inverse	Allows perfect decryption
Integrity check	Optional tag from final state	Detects tampering

8. Next Steps

1. Implement the three reversible steps:
   - Shift (geometry),
   - S-box (non-linearity),
   - Block-mix (diffusion).
2. Verify that applying the inverse restores the original data.
3. Combine the three steps into a round function and test 8–12 rounds on small images.
4. Wrap this in the full sponge encryption loop.

Once these components work together, the result is a fully functional, fluid-based encryption system that can both scramble and unscramble images exactly.

9. Mathematical Formulation of the Cipher

9.1 Data Representation

Let the plaintext image or data block be represented as a two-dimensional array

$$X = [x_{i,j}] \in \{0, 1, \ldots, 255\}^{H \times W},$$

where H and W denote the height and width in pixels (or bytes). Each $x_{i,j}$ is an 8-bit value.

We define the encryption process as a composition of several *bijective maps* acting on X.

9.2 Key and Round Parameters

A master key K and nonce N are used to generate a sequence of round parameters:

$$\mathcal{P}_r = \big(s_r(y),\, t_r(x),\, S_r,\, M_r,\, \Pi_r\big), \quad r = 1, \ldots, R,$$

where

- $s_r(y)$ and $t_r(x)$ are integer-valued shift functions (horizontal and vertical),
- S_r is a round-dependent byte substitution table (S-box),
- M_r is an invertible diffusion matrix,
- Π_r is an optional permutation of 2×2 or 4×4 blocks.

The round parameters are derived using HKDF with HMAC-SHA-256 as:

$$\mathcal{P}_r = \mathrm{HKDF\_}HMAC - SHA256(K, N \| \text{“}round''\| r).$$

Each component s_r, t_r, S_r, M_r uses domain-separated labels (“sr-r”, “tr-r”, “sbox-r”, “mat-r”) to avoid collisions.

9.3 Fluid Shift (Incompressible Flow Step)

The geometric “fluid” motion is realized by two shears:

$$(x, y) \mapsto (x + s_r(y) \bmod W,\ y), (x, y) \mapsto (x,\ y + t_r(x) \bmod H).$$

Applying these sequentially gives the key-dependent permutation $\mathcal{F}_r$:

$$\mathcal{F}_r = \mathrm{Y-shear} \circ \mathrm{X-shear}.$$

$$\mathcal{F}_r^{-1} = \mathrm{X-shear}^{-1} \circ \mathrm{Y-shear}^{-1}.$$

The inverse is obtained by reversing the order and negating the shifts:

$$P_r^{-1} = \mathrm{X-shear}^{-1} \circ \mathrm{Y-shear}^{-1}.$$

Each P_r is a *bijective map* on the coordinate grid (x, y), ensuring perfect reversibility.

9.4 Nonlinear Substitution (S-box)

For each round, a byte substitution $S_r : \{0, \ldots, 255\} \to \{0, \ldots, 255\}$ is defined as a key-derived permutation.

Applied elementwise:

$$x'_{i,j} = S_r(x_{i,j}),$$

with inverse

$$x_{i,j} = S_r^{-1}(x'_{i,j}).$$

This step introduces nonlinearity (confusion).

9.5 Local Diffusion (Mixing)

Divide the array into disjoint 2×2 blocks

$$B = abcd.$$

Each block is linearly transformed using an invertible matrix $M_r \in \mathrm{GL}(4, \mathrm{GF}(2^8))$:

$$a'b'c'd' = M_r abcd.$$

The inverse operation uses M_r^{-1}. This step ensures local diffusion: a change in one byte affects all bytes in the block after decryption.

9.6 One Encryption Round

A single round r is a composition of the above three reversible operations:

$$X_r = \Pi_r \circ \mathrm{Mix}(M_r) \circ S_r \circ P_r(X_{r-1}),$$

where

- P_r is the coordinate permutation (fluid flow),
- S_r is the nonlinear byte substitution,
- $\mathrm{Mix}(M_r)$ is the diffusion layer,
- Π_r is an optional block permutation for additional diffusion.

The full encryption permutation after R rounds is:

$$\pi_{K,N} = \pi^{(R)} \circ \pi^{(R-1)} \circ \cdots \circ \pi^{(1)},$$

which is also bijective.

Decryption applies the inverse rounds in reverse order:

$$X_0 = (\pi^{(1)})^{-1} \circ \cdots \circ (\pi^{(R)})^{-1}(X_R).$$

9.7 Sponge Mode Encryption

Let S denote an internal state of $H \times W$ bytes, divided into:

$$S = \big(S_{rate},\, S_{capacity}\big),$$

with $r + c = H \times W$.

Initialization:

$$S_0 = \mathrm{KDF}(K, N, \text{“}init''\text{)}.$$

Encryption loop (for message blocks M_i):

1. **Squeeze:** Extract keystream bytes from the state:

$$Z_i \leftarrow S_{rate}.$$

2. **Encrypt:** XOR with the plaintext to produce the ciphertext:

$$C_i = M_i \oplus Z_i.$$

3. **Absorb:** Mix the plaintext into the state:

$$S_{rate} \leftarrow S_{rate} \oplus M_i.$$

4. **Permute:** Apply the internal permutation:

$$S \leftarrow \pi_{K,N}(S).$$

After processing all blocks, one more permutation is applied and a tag T is squeezed for integrity:

$$T = \mathrm{Squeeze}(S).$$

Output: Ciphertext $C = (C_1, C_2, \ldots)$, and optional tag T derived by squeezing additional bytes from S.

Decryption repeats the same steps to reconstruct $M_i = C_i \oplus Z_i$.

9.8 Reversibility

Every layer (P_r, S_r, $\mathrm{Mix}(M_r)$, Π_r) is invertible. Therefore, the overall permutation $\pi_{K,N}$ is bijective:

$$\pi_{K,N}^{-1} = (\pi^{(1)})^{-1} \circ \cdots \circ (\pi^{(R)})^{-1}.$$

Running the inverse rounds with the same key and nonce exactly restores the original data.

9.9 Summary of Mathematical Steps

Step	Operation	Mathematical Form
1. Fluid shift	Coordinate permutation	$\mathcal{F}_r = \mathrm{Y-shear} \circ \mathrm{X-shear}$
2. S-box	Byte substitution	$x'_{i,j} = S_r(x_{i,j})$
3. Diffusion	Linear mix	$\mathbf{b}' = M_r\mathbf{b}$
4. Block permutation	Shuffle of tiles	Π_r on indices

—

10. Conceptual Summary

- The fluid step moves data like an incompressible flow.
- The S-box acts as nonlinear turbulence.
- The diffusion matrix spreads local changes, simulating molecular mixing.
- Repeating the process creates thorough but reversible chaos.
- Because the full map is bijective, decryption perfectly reverses encryption.