JAL CIPHER: A Jigsaw Puzzle-Inspired Product Cipher Combining Key-Driven Transposition with Per-Piece Affine Substitution

Sandra Asia Mansuru, Emmanuel Frimpong Nyamah
Department of Information Technology, USTED, Ashanti Region, Ghana

ABSTRACT
This paper presents JAL CIPHER, a symmetric product cipher combining two complementary layers motivated by the mechanics of jigsaw puzzles. The first layer, JIGSAWCIPHER, is a key-driven transposition cipher: plaintext is cut into variable-length pieces whose boundaries are secret, and the pieces are assembled in a key-derived order. The second layer, the Jigsaw Affine Layer (JAL), applies a distinct affine cipher $S(x) = (a_i \cdot x + b_i) \bmod m$ to every piece P_i. The two layers are computationally independent: breaking the cipher requires defeating both the transposition search space $\Omega(L)$ and the substitution space $|AGL(1, \mathbb{Z}_m)|^k$ simultaneously. All mathematical foundations — modular arithmetic, the Extended Euclidean Algorithm, Euler's totient function, the affine group $AGL(1, \mathbb{Z}_m)$, and the binomial coefficient — are developed from first principles in journal display-equation format, with full step-by-step derivations for all six modular inverses and every encryption and decryption computation. The combined cipher's parameter space corresponds to 99.41 bits of keyspace at L=17 and exceeds 200 bits at L=32, while requiring only O(L log L) computation. An empirical cryptanalysis against statistically-informed adversaries is further reported: per-piece affine recovery using English letter statistics, a crib attack, and a nonce-reuse demonstration, which together show that realized security against natural-language plaintext is substantially below the nominal keyspace size, particularly for short pieces.

Keywords
Affine Cipher, classical cryptography, fragmentation cipher, jigsaw puzzle, key-driven permutation, product cipher, transposition cipher

1. INTRODUCTION
A jigsaw puzzle presents a complete picture cut into interlocking irregular pieces. The solver faces a combinatorial problem whose difficulty arises not from arithmetic but from structural uncertainty: the solver does not know how many pieces exist, what shapes they are, or which position each occupies.

This paper formalizes the jigsaw puzzle as a product cipher. The JIGSAWCIPHER layer cuts plaintext into variable-length secret pieces and assembles them in key-derived order. The Jigsaw Affine Layer (JAL) then applies a distinct affine map $S(x) = (a_i \cdot x + b_i) \bmod m$ to each piece, destroying character frequencies. Section 3 develops all mathematical foundations from first principles, with every derivation shown in display-equation format. Section 8 demonstrates the complete cipher on a concrete example, showing every arithmetic step for all 17 characters of encryption and all 17 characters of decryption.

The paper is organized as follows. Section 2 reviews related work. Section 3 develops the mathematical foundations. Section 4 presents the jigsaw model. Section 5 specifies the key schedule. Section 6 presents algorithms and correctness proofs. Section 7 defines block mode. Section 8 gives the full worked example. Section 9 conducts security analysis, including an empirical cryptanalytic evaluation against statistically-informed adversaries (Section 9.5). Section 10 compares against classical ciphers. Section 11 concludes.

2 RELATED WORK

2.1 Classical Transposition and Substitution Ciphers
Transposition ciphers rearrange plaintext without substitution [1]. Rail Fence, Columnar, and Route ciphers all fix piece size at one character, exposing them to anagramming and period-detection [2]. Shannon [3] established the product cipher principle: composing substitution and transposition produces a cipher whose security exceeds either component alone. Broader treatments of classical cipher design and of modern applied cryptography are given in [4] and [5], respectively. The classical affine cipher $S(x) = (ax+b) \bmod m$ has been extensively analyzed [6, 7]; in its single-key form, frequency analysis succeeds immediately. JAL assigns independent (a_i, b_i) per piece, closing this vulnerability.

2.2 Modern Lightweight Ciphers
Substitution-permutation networks such as PRESENT [8] and SIMON [9] interleave fixed-block operations. Lightweight ciphers designed for constrained or resource-scarce deployment settings have also been surveyed [10]. JAL CIPHER differs fundamentally: block boundaries are secret key material, not public parameters. This gives a second dimension of adversary uncertainty beyond the permutation alone.

2.3 Jigsaw-Inspired Cryptography
Ling [11] proposed the Jigsaw Encryption System (JES), in which a single plaintext file is split into multiple ciphertext files resembling jigsaw pieces, so that the loss of a subset of pieces does not compromise the full plaintext. JES operates at the file level using 1-to-N file mapping and does not define a text-level cipher, a key-derived cut sequence, or a substitution layer. JAL CIPHER differs in every respect: it is a character-level product cipher, the number of pieces and all cut boundaries are secret key material unknown to the adversary, and the JAL substitution layer applies an independent affine map per piece. The jigsaw metaphor is shared; the constructions are otherwise disjoint.

3 MATHEMATICAL FOUNDATIONS
This section develops every mathematical concept used in the cipher from first principles, in the order in which each concept is first needed.

3.1 Modular Arithmetic
Modular arithmetic is the foundation of all computations in the cipher. The modulus m is the alphabet size; for uppercase English letters, m = 26.

Definition 1 ***(Modulus and Remainder).***

For integers n and m with m > 0, the expression n mod m denotes the unique integer r satisfying $0 \le r < m$ and $n = q \cdot m + r$ for some integer q. This is written $n \equiv r \pmod m$, and n and r are said to be congruent modulo m.

The following display computations illustrate modular reduction for values arising in Section 8:

$$58 \bmod 26 = 58 - 2 \times 26 = 58 - 52 = 6$$
$$79 \bmod 26 = 79 - 3 \times 26 = 79 - 78 = 1$$
$$106 \bmod 26 = 106 - 4 \times 26 = 106 - 104 = 2$$
$$121 \bmod 26 = 121 - 4 \times 26 = 121 - 104 = 17$$
$$221 \bmod 26 = 221 - 8 \times 26 = 221 - 208 = 13$$

Negative values are reduced by adding the smallest positive multiple of m that makes the result non-negative:

$$-9 \bmod 26 = -9 + 1 \times 26 = 17$$
$$-54 \bmod 26 = -54 + 3 \times 26 = -54 + 78 = 24$$
$$-63 \bmod 26 = -63 + 3 \times 26 = -63 + 78 = 15$$
$$-189 \bmod 26 = -189 + 8 \times 26 = -189 + 208 = 19$$
$$-168 \bmod 26 = -168 + 7 \times 26 = -168 + 182 = 14$$
$$-99 \bmod 26 = -99 + 4 \times 26 = -99 + 104 = 5$$
$$-117 \bmod 26 = -117 + 5 \times 26 = -117 + 130 = 13$$

In general, for any integer n, the value $((n \bmod m) + m) \bmod m$ always lies in $\{0, 1, \ldots, m-1\}$ and equals $n \bmod m$. Every one of these reductions appears explicitly in Section 8.

3.2 Greatest Common Divisor and Coprimality

Definition 2 ***(Greatest Common Divisor).***
The greatest common divisor gcd(a, b) is the largest positive integer dividing both a and b. Two integers are coprime if gcd(a, b) = 1.

Coprimality is the condition for an affine map to be invertible. The reasoning is as follows. Suppose $f(x) = ax + b \pmod m$ maps two distinct inputs x_1 and x_2 to the same value. Then $ax_1 + b \equiv ax_2 + b \pmod m$, which simplifies to $a(x_1 - x_2) \equiv 0 \pmod m$. This means m divides $a(x_1 - x_2)$. If gcd(a, m) = 1 then m shares no factor with a, so m must divide $(x_1 - x_2)$ entirely. Since both x_1 and x_2 belong to $\{0, \ldots, m-1\}$, the only way m divides their difference is if $x_1 = x_2$, contradicting distinctness. Therefore f is injective, and since it maps a finite set to itself, it is also surjective — hence bijective (invertible). Conversely, if gcd(a, m) = d > 1, then f(0) = f(m/d) because $a \cdot (m/d) = (a/d) \cdot m \equiv 0 \pmod m$, so f is not injective. The condition gcd(a, m) = 1 is therefore both necessary and sufficient.

3.3 Euler's Totient Function

Definition 3 ***(Euler's Totient).***
The totient $\varphi(m)$ counts the integers in $\{1, 2, \ldots, m\}$ that are coprime to m. For a prime p, $\varphi(p) = p - 1$. The totient is multiplicative: if gcd(a, b) = 1 then $\varphi(ab) = \varphi(a) \cdot \varphi(b)$.

Applying Definition 3 to $m = 26 = 2 \times 13$, where gcd(2, 13) = 1:

$$\varphi(26) = \varphi(2) \times \varphi(13)$$
$$= (2 - 1) \times (13 - 1)$$
$$= 1 \times 12 = 12$$

The 12 valid multipliers for m = 26 are therefore: {1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25}. Any value outside this set shares a factor with 26 and produces a non-invertible map.

3.4 Modular Multiplicative Inverse and the Extended Euclidean Algorithm

Definition 4 ***(Modular Multiplicative Inverse).***
The modular multiplicative inverse of a modulo m, written $a^{-1} \pmod m$, is the unique integer $x \in \{1, \ldots, m-1\}$ satisfying $a \cdot x \equiv 1 \pmod m$. It exists if and only if gcd(a, m) = 1. It is computed via the Extended Euclidean Algorithm, which finds integers x and y satisfying Bézout's identity: $a \cdot x + m \cdot y = \gcd(a, m) = 1$, giving $a^{-1} = x \bmod m$.

The following display derivations compute all six inverses required in Section 8, showing every step of the Extended Euclidean Algorithm and the back-substitution.

Derivation of $3^{-1} \bmod 26$. Apply the Euclidean algorithm to find gcd(26, 3):

$$26 = 8 \times 3 + 2$$
$$3 = 1 \times 2 + 1$$
$$2 = 2 \times 1 + 0 \implies \gcd(3, 26) = 1$$

Back-substitute to express 1 as a linear combination of 3 and 26:

$$1 = 3 - 1 \times 2$$
$$= 3 - 1 \times (26 - 8 \times 3)$$
$$= 9 \times 3 - 1 \times 26$$
$$\implies 3^{-1} = 9 \pmod{26}$$ [Verify: $3 \times 9 = 27 = 26 + 1 \equiv 1 \pmod{26}$]

Derivation of $5^{-1} \bmod 26$. Apply the Euclidean algorithm:

$$26 = 5 \times 5 + 1$$
$$5 = 5 \times 1 + 0 \implies \gcd(5, 26) = 1$$

The remainder 1 appears immediately, so:

$$1 = 26 - 5 \times 5$$
$$\implies 5^{-1} = -5 \bmod 26 = 21$$ [Verify: $5 \times 21 = 105 = 4 \times 26 + 1 \equiv 1 \pmod{26}$]

Derivation of $7^{-1} \bmod 26$. Apply the Euclidean algorithm:

$$26 = 3 \times 7 + 5$$
$$7 = 1 \times 5 + 2$$
$$5 = 2 \times 2 + 1$$
$$2 = 2 \times 1 + 0 \implies \gcd(7, 26) = 1$$

Back-substitute:

$$1 = 5 - 2 \times 2$$
$$= 5 - 2 \times (7 - 1 \times 5) = 3 \times 5 - 2 \times 7$$
$$= 3 \times (26 - 3 \times 7) - 2 \times 7 = 3 \times 26 - 11 \times 7$$
$$\implies 7^{-1} = -11 \bmod 26 = 15$$ [Verify: $7 \times 15 = 105 = 4 \times 26 + 1 \equiv 1 \pmod{26}$]

Derivation of $9^{-1} \bmod 26$. Apply the Euclidean algorithm:

$$26 = 2 \times 9 + 8$$
$$9 = 1 \times 8 + 1$$
$$8 = 8 \times 1 + 0 \implies \gcd(9, 26) = 1$$

Back-substitute:

$$1 = 9 - 1 \times 8 = 9 - 1 \times (26 - 2 \times 9) = 3 \times 9 - 1 \times 26$$
$$\implies 9^{-1} = 3 \pmod{26}$$ [Verify: $9 \times 3 = 27 = 26 + 1 \equiv 1 \pmod{26}$]

Derivation of $11^{-1} \bmod 26$. Apply the Euclidean algorithm:

$$26 = 2 \times 11 + 4$$
$$11 = 2 \times 4 + 3$$
$$4 = 1 \times 3 + 1$$
$$3 = 3 \times 1 + 0 \implies \gcd(11, 26) = 1$$

Back-substitute:

$$1 = 4 - 1 \times 3$$
$$= 4 - 1 \times (11 - 2 \times 4) = 3 \times 4 - 1 \times 11$$

$= 3 \times (26 - 2 \times 11) - 1 \times 11 = 3 \times 26 - 7 \times 11$

$\Longrightarrow 11^{-1} = -7 \bmod 26 = 19$ [Verify: $11 \times 19 = 209 = 8 \times 26 + 1 \equiv 1 \pmod{26}$]

3.5 The Affine Group AGL(1, $\mathbb{Z}_m$)

Definition 5 ***(Affine Map).***

An affine map over $\mathbb{Z}_m$ is a function $f : \mathbb{Z}_m \to \mathbb{Z}_m$ of the form $f(x) = ax + b \pmod m$, where $a, b \in \mathbb{Z}_m$ and $\gcd(a, m) = 1$. Its inverse is $f^{-1}(y) = a^{-1}(y - b) \pmod m$. The set of all affine maps over $\mathbb{Z}_m$ forms a group under composition, called the one-dimensional affine group AGL(1, $\mathbb{Z}_m$), with order given by Equation (1).| **AGL(1, $\mathbb{Z}$)| = m · φ(m)** (1)

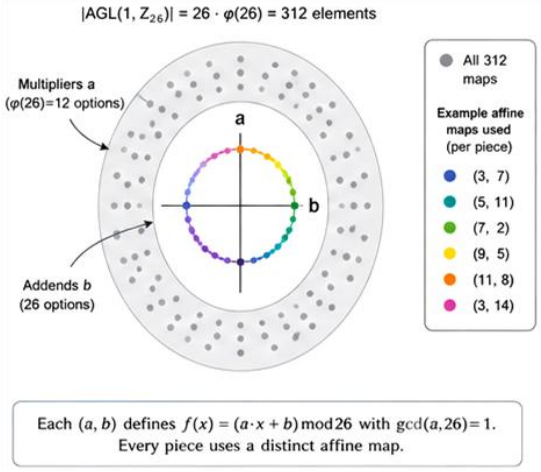

Fig.1 The Affine Group AGL(1, $\mathbb{Z}$)

For m = 26: $|AGL(1, \mathbb{Z}_{26})| = 26 \times 12 = 312$ distinct invertible affine maps. For m = 256 (byte-level): $|AGL(1, \mathbb{Z}_{256})| = 256 \times 128 = 32{,}768$ maps. Each piece in the cipher is encrypted under one map independently chosen from this group; with k = 6 pieces the combined substitution space is $312^6 \approx 9.22 \times 10^{14}$.

3.6 The Binomial Coefficient

Definition 6 ***(Binomial Coefficient).***

The binomial coefficient C(n, r), read "n choose r", counts the number of ways to select r items from n items when order does not matter. It is computed by Equation (2).

$$\mathbf{C(n, r) = n! / (r! \times (n - r)!)} \quad (2)$$

In the security analysis, C(L−1, k−1) counts k-piece cut sequences for a message of length L: it equals the number of ways to choose k−1 cut points from the L−1 gaps between characters. For the worked example (L=17, k=6):

$$C(16, 5) = 16! / (5! \times 11!)$$
$$= (16 \times 15 \times 14 \times 13 \times 12) / (5 \times 4 \times 3 \times 2 \times 1)$$
$$= 524{,}160 / 120 = 4{,}368 \text{ distinct 6-piece cut sequences}$$

3.7 Character Encoding

Characters are encoded as integers by the natural mapping A=0, B=1, C=2, …, Z=25. For the plaintext "CRYPTOGRAPHYISFUN" the integer sequence is: C=2, R=17, Y=24, P=15, T=19, O=14, G=6, R=17, A=0, P=15, H=7, Y=24, I=8, S=18, F=5, U=20, N=13. These are the values of x in each affine computation.

4 THE JIGSAW PUZZLE MODEL

4.1 Cut Sequence

Let P be a plaintext of length L. A cut sequence C = (c_1,…,$c_{}$) satisfies Equation (3). Piece P_i occupies positions s_i through $s_i + c_i - 1$ of P, where start positions are computed by Equation (4).

$$\mathbf{c_1 + c_2 + \cdots + c = L, \quad c_i \geq 1 \ \text{for all i}} \quad (3)$$

$$\mathbf{s_1 = 1, \quad s_i = c_1 + c_2 + \cdots + c_{i-1} + 1 \ \text{for i > 1}} \quad (4)$$

For C=(3,3,3,3,2,3) and L=17, the start positions are:

$s_1 = 1 \rightarrow$ P[1..3] = "CRY"
$s_2 = 1 + 3 = 4 \rightarrow$ P[4..6] = "PTO"
$s_3 = 4 + 3 = 7 \rightarrow$ P[7..9] = "GRA"
$s_4 = 7 + 3 = 10 \rightarrow$ P[10..12] = "PHY"
$s_5 = 10 + 3 = 13 \rightarrow$ P[13..14] = "IS"
$s_6 = 13 + 2 = 15 \rightarrow$ P[15..17] = "FUN"

4.2 Assembly Order and Ciphertext

An assembly order A = (α_1,…,α) is a permutation of {1,…,k}. After JAL substitution (Section 6), the ciphertext is formed by Equation (5):

$$\mathbf{C = P'\_\{\alpha_1\} \parallel P'\_\{\alpha_2\} \parallel \cdots \parallel P'\_\{\alpha\}} \quad (5)$$

5 KEY SCHEDULE

5.1 CSPRNG Seeding

The CSPRNG is seeded with H(K ‖ η), where H is a standard secure hash function [12], K is the secret key (≥128 bits) and η is a unique per-encryption nonce. Both sender and receiver run the same stream and derive identical parameters independently. In practice the byte stream should be produced by a deterministic random bit generator meeting recognised cryptographic standards [13]. The nonce is transmitted in the clear; it prevents the key-reuse attack discussed in Section 9.

5.2 Derivation Sequence

All parameters are derived from a single stream in strict order:

Byte b_1:k = $(b_1 \bmod (L - 1)) + 2 \in \{2, 3, \ldots, L\}$

Bytes b_2 – b :k−1 cut points from {1,…,L−1}, sorted → piece sizes by differences (a broken-stick construction [14])

Bytes b_{+1} – b_{2-1}:Assembly order A via Fisher-Yates shuffle [15] of (1, 2, …, k)

Bytes b_2 – b_{4-1}:k pairs $(a_i, b_i) \in AGL(1, \mathbb{Z})$, one per piece

The broken-stick construction for cut points: sort k−1 random values $t_1 < t_2 < \ldots < t_{-1}$ from {1,…,L−1}. Piece sizes are $c_1 = t_1$, $c_i = t_i - t_{i-1}$ for i>1, c =L−t_{-1}. For example, points {3,6,9,12,14} in a length-17 message give sizes (3,3,3,3,2,3). For (a_i, b_i) derivation, draw candidate pairs until gcd(a, m)=1; expected draws per pair = m/φ(m) = 26/12 ≈ 2.2.

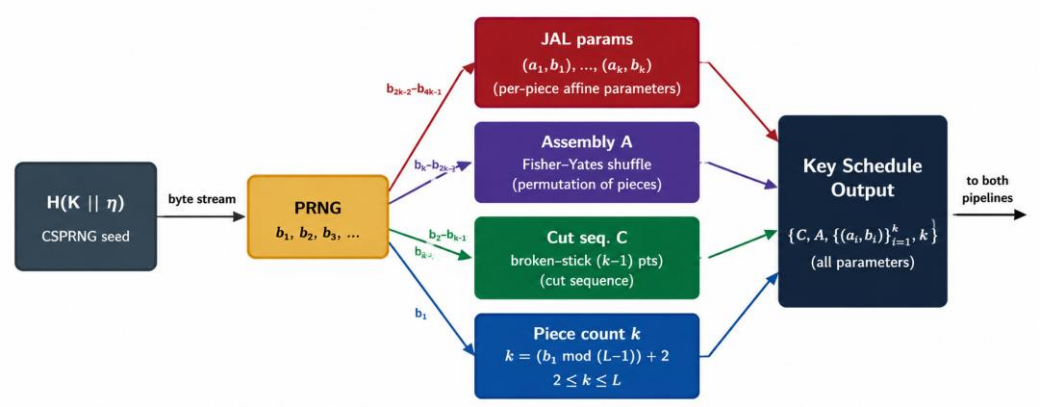

Fig.2. Key schedule

6 ENCRYPTION, DECRYPTION, AND CORRECTNESS

6.1 Encryption

The affine substitution in Step 4 applies $S(x) = (a_i \cdot x + b_i) \bmod m$ independently to every character x of piece P_i.

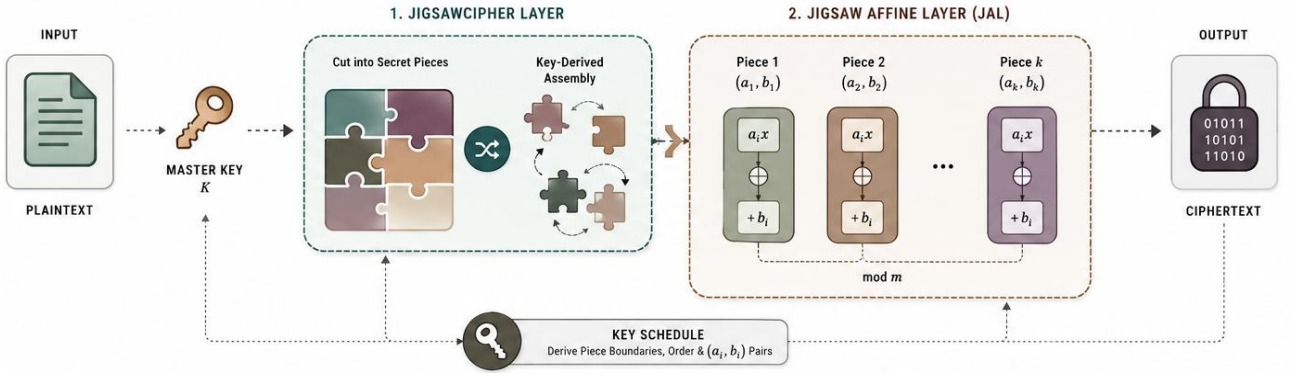

Fig.3. Encryption

Algorithm 1: JAL CIPHER Encryption

Input : Plaintext P (length L), key K, nonce η

Output: Ciphertext C (length L)

1. seed ← H(K ‖ η)
2. (C, A, {(ai,bi)}) ← KeySchedule(seed, L) // Section 5
3. Cut P into P1,…,P using C and Equations (3)–(4)
4. For i = 1,…,k: $P_i' \leftarrow \{(a_i \cdot x + b_i) \bmod m : x \in P_i\}$ // JAL
5. C ← P'_{A[1]} ‖ P'_{A[2]} ‖ … ‖ P'_{A[k]} // Equation (5)
6. Output (C, η)

6.2 Decryption

Decryption cuts the ciphertext using emit-order lengths. The emit-order length at position j is $\ell_j = c_{A[j]}$, the length of the piece that was emitted at position j. For A=(3,1,5,6,2,4) and C=(3,3,3,3,2,3):

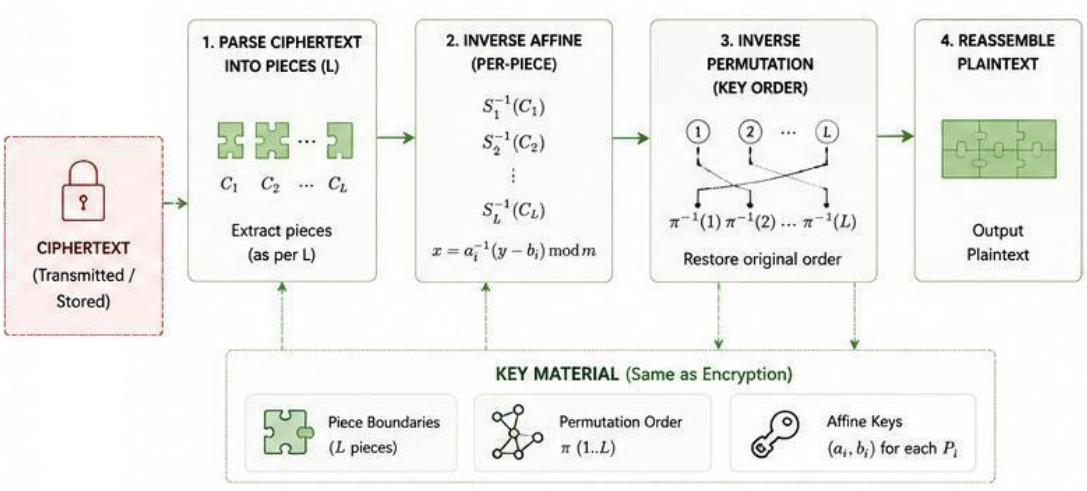

Fig.4. Decryption

$\ell_1 = c_{A[1]} = c_3 = 3$ (piece P_3 was emitted 1st, its cut length is c_3)

$\ell_2 = c_{A[2]} = c_1 = 3$ (piece P_1 was emitted 2nd)

$\ell_3 = c_{A[3]} = c_5 = 2$ (piece P_5 was emitted 3rd)

$\ell_4 = c_{A[4]} = c_6 = 3$ (piece P_6 was emitted 4th)

$\ell_5 = c_{A[5]} = c_2 = 3$ (piece P_2 was emitted 5th)

$\ell_6 = c_{A[6]} = c_4 = 3$ (piece P_4 was emitted 6th)

Emit-order lengths = (3, 3, 2, 3, 3, 3)

Algorithm 2: JAL CIPHER Decryption

Input : Ciphertext C (length L), key K, nonce η

Output: Plaintext P (length L)

1. seed ← H(K ‖ η)
2. (C, A, {(a_i,b_i)}) ← KeySchedule(seed, L)
3. Compute $\ell_j = c_{A[j]}$ for j=1,…,k (emit-order lengths)
4. Cut C into $Q_1,\ldots,Q$ using lengths ($\ell_1,\ldots,\ell$)
5. Inverse permutation: for j=1,…,k set slot[A[j]] ← Q_j
6. JAL inverse: for i=1,…,k set P_i ← {(a_i^{-1}·(y−b_i)) mod m : y ∈ slot[i]}
7. P ← P_1 ‖ P_2 ‖ … ‖ P
8. Output P

Theorem 1 *(Correctness).*

For any plaintext P of length L, key K, and nonce η, Algorithm 2 applied to the output of Algorithm 1 recovers P exactly.

Proof. Both algorithms derive the same (C, A, {(a_i,b_i)}) from seed = H(K‖η). The j-th emit-order length $\ell_j = c_{A[j]}$ equals $|P'_{A[j]}|$, so $Q_j = P'_{A[j]}$ for each j. Placing Q_j into slot[A[j]] gives slot[i] = P'_i. Applying $S^{-1}(y) = a_i^{-1}(y-b_i)$ mod m recovers each character x because $S^{-1}(S(x)) = a_i^{-1}((a_i x+b_i)-b_i)$ mod m = x. Concatenating recovers P. ■

6.3 System Architecture

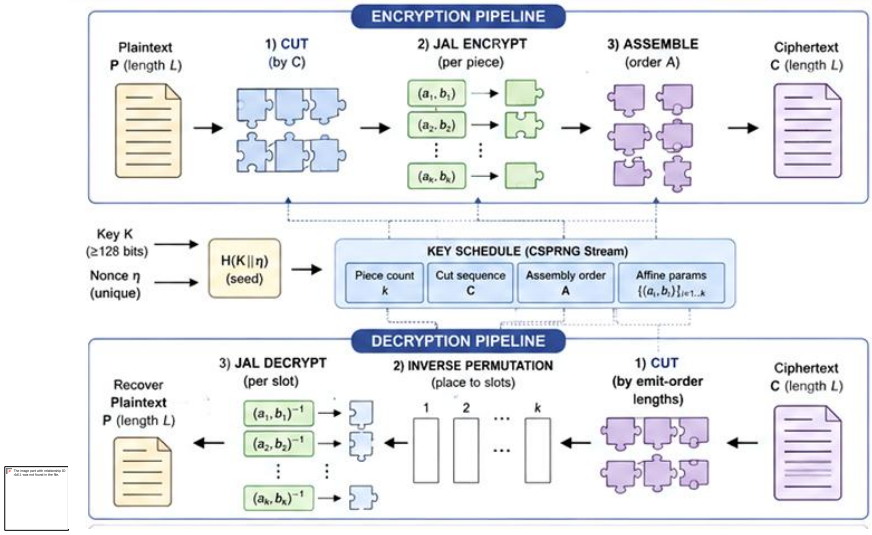

Fig.5. System Architecture

7. BLOCK MODE

For messages of length N > B (block size, e.g. B=64), partition P into blocks of length B using per-block nonce $\eta_i = \eta\|i$. Time complexity per block is O(B log B) (dominated by sorting k−1 cut points). Total: O(N log B), linear in N for fixed B.

8. WORKED EXAMPLE

The plaintext P = "CRYPTOGRAPHYISFUN" (L=17, m=26, A=0…Z=25) is encrypted under parameters given in Table 1. Section 8.2 shows every character of encryption as display-aligned computation. Section 8.4 shows every character of decryption the same way.

Table 1. Key-derived parameters.

Parameter	Value
Plaintext P	CRYPTOGRAPHYISFUN (L = 17)
Cut sequence C	(3, 3, 3, 3, 2, 3)
Assembly order A	(3, 1, 5, 6, 2, 4)
P_1: $a_1=3$, $b_1=7$	$a_1^{-1} = 9$ [3×9 = 27 ≡ 1 (mod 26)]
P_2: $a_2=5$, $b_2=11$	$a_2^{-1} = 21$ [5×21 = 105 ≡ 1 (mod 26)]
P_3: $a_3=7$, $b_3=2$	$a_3^{-1} = 15$ [7×15 = 105 ≡ 1 (mod 26)]
P_4: $a_4=9$, $b_4=5$	$a_4^{-1} = 3$ [9×3 = 27 ≡ 1 (mod 26)]
P_5: $a_5=11$, $b_5=8$	$a_5^{-1} = 19$ [11×19 = 209 ≡ 1 (mod 26)]
P_6: $a_6=3$, $b_6=14$	$a_6^{-1} = 9$ [3×9 = 27 ≡ 1 (mod 26)]

8.1 Step 1 — Cutting

Applying cut sequence C=(3,3,3,3,2,3) via Equations (3)–(4) partitions P into: P_1="CRY", P_2="PTO", P_3="GRA", P_4="PHY", P_5="IS", P_6="FUN". Start position computations are given in Section 4.1.

8.2 Step 2 — JAL Encryption: Aligned Computations

Formula for every character: $S(x) = (a_i \cdot x + b_i) \bmod 26$. Each computation is shown in full display form. The final result is decoded using the inverse mapping (integer → letter).

Piece P_1 = "CRY" with $a_1=3$, $b_1=7$: $S(x) = (3x + 7) \bmod 26$

$$S(C) = S(2) = (3 \times 2 + 7) \bmod 26 = (6 + 7) \bmod 26 = 13 \bmod 26 = 13 = N$$
$$S(R) = S(17) = (3 \times 17 + 7) \bmod 26 = (51 + 7) \bmod 26 = 58 \bmod 26 = 6 = G$$
$$S(Y) = S(24) = (3 \times 24 + 7) \bmod 26 = (72 + 7) \bmod 26 = 79 \bmod 26 = 1 = B$$
$$\Longrightarrow P_1' = \text{"NGB"}$$

Piece P_2 = "PTO" with $a_2=5$, $b_2=11$: $S(x) = (5x + 11) \bmod 26$

$$S(P) = S(15) = (5 \times 15 + 11) \bmod 26 = (75 + 11) \bmod 26 = 86 \bmod 26 = 8 = I$$
$$S(T) = S(19) = (5 \times 19 + 11) \bmod 26 = (95 + 11) \bmod 26 = 106 \bmod 26 = 2 = C$$
$$S(O) = S(14) = (5 \times 14 + 11) \bmod 26 = (70 + 11) \bmod 26 = 81 \bmod 26 = 3 = D$$
$$\Longrightarrow P_2' = \text{"ICD"}$$

Piece P_3 = "GRA" with $a_3=7$, $b_3=2$: $S(x) = (7x + 2) \bmod 26$

$$S(G) = S(6) = (7 \times 6 + 2) \bmod 26 = (42 + 2) \bmod 26 = 44 \bmod 26 = 18 = S$$
$$S(R) = S(17) = (7 \times 17 + 2) \bmod 26 = (119 + 2) \bmod 26 = 121 \bmod 26 = 17 = R$$
$$S(A) = S(0) = (7 \times 0 + 2) \bmod 26 = (0 + 2) \bmod 26 = 2 \bmod 26 = 2 = C$$
$$\Longrightarrow P_3' = \text{"SRC"}$$

Piece P_4 = "PHY" with $a_4=9$, $b_4=5$: $S(x) = (9x + 5) \bmod 26$

$$S(P) = S(15) = (9 \times 15 + 5) \bmod 26 = (135 + 5) \bmod 26 = 140 \bmod 26 = 10 = K$$
$$S(H) = S(7) = (9 \times 7 + 5) \bmod 26 = (63 + 5) \bmod 26 = 68 \bmod 26 = 16 = Q$$
$$S(Y) = S(24) = (9 \times 24 + 5) \bmod 26 = (216 + 5) \bmod 26 = 221 \bmod 26 = 13 = N$$
$$\Longrightarrow P_4' = \text{"KQN"}$$

Piece P_5 = "IS" with $a_5=11$, $b_5=8$: $S(x) = (11x + 8) \bmod 26$

$$S(I) = S(8) = (11 \times 8 + 8) \bmod 26 = (88 + 8) \bmod 26 = 96 \bmod 26 = 18 = S$$
$$S(S) = S(18) = (11 \times 18 + 8) \bmod 26 = (198 + 8) \bmod 26 = 206 \bmod 26 = 24 = Y$$
$$\Longrightarrow P_5' = \text{"SY"}$$

Piece P_6 = "FUN" with $a_6=3$, $b_6=14$: $S(x) = (3x + 14) \bmod 26$

$$S(F) = S(5) = (3 \times 5 + 14) \bmod 26 = (15 + 14) \bmod 26 = 29 \bmod 26 = 3 = D$$
$$S(U) = S(20) = (3 \times 20 + 14) \bmod 26 = (60 + 14) \bmod 26 = 74 \bmod 26 = 22 = W$$
$$S(N) = S(13) = (3 \times 13 + 14) \bmod 26 = (39 + 14) \bmod 26 = 53 \bmod 26 = 1 = B$$
$$\Longrightarrow P_6' = \text{“DWB”}$$

Substituted pieces: P_1'=“NGB”, P_2'=“ICD”, P_3'=“SRC”, P_4'=“KQN”, P_5'=“SY”, P_6'=“DWB”.

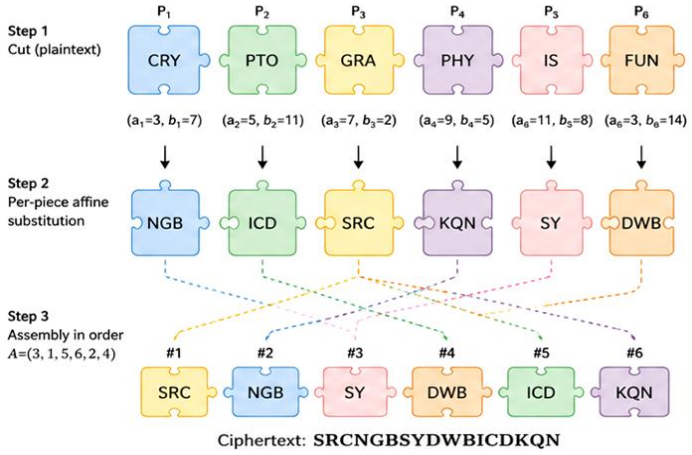

Fig.6 Three-step encryption

8.3 Step 3 — Transposition Assembly

Pieces are emitted in assembly order A=(3,1,5,6,2,4):

Emit #1: P_3'“SRC” → running: “SRC”
Emit #2: P_1'“NGB” → running: “SRCNGB”
Emit #3: P_5'“SY” → running: “SRCNGBSY”
Emit #4: P_6'“DWB” → running: “SRCNGBSYDWB”
Emit #5: P_2'“ICD” → running: “SRCNGBSYDWBICD”
Emit #6: P_4'“KQN” → Ciphertext C = “SRCNGBSYDWBICDKQN”

8.4 Decryption: Aligned Computations

Step D1 — Cut by emit-order lengths (3,3,2,3,3,3):

Q_1 = “SRCNGBSYDWBICDKQN”[1..3] = “SRC” (from P_3')
Q_2 = [4..6] = “NGB” (from P_1')
Q_3 = [7..8] = “SY” (from P_5')
Q_4 = [9..11] = “DWB” (from P_6')
Q_5 = [12..14] = “ICD” (from P_2')
Q_6 = [15..17] = “KQN” (from P_4')

Step D2 — Inverse permutation. A=(3,1,5,6,2,4) means emit-pos j held piece A[j]:

slot[3]← Q_1 = “SRC” (emit-pos 1 held piece A[1]=3)
slot[1]← Q_2 = “NGB” (emit-pos 2 held piece A[2]=1)
slot[5]← Q_3 = “SY” (emit-pos 3 held piece A[3]=5)
slot[6]← Q_4 = “DWB” (emit-pos 4 held piece A[4]=6)
slot[2]← Q_5 = “ICD” (emit-pos 5 held piece A[5]=2)
slot[4]← Q_6 = “KQN” (emit-pos 6 held piece A[6]=4)

Step D3 — JAL inverse. Formula: $S^{-1}(y) = (a_i^{-1} \times (y - b_i)) \bmod 26$. Every step shown below:

slot[1] = “NGB” with a_1^{-1}=9, b_1=7: $S^{-1}(y) = (9 \times (y-7)) \bmod 26$

$$S^{-1}(N) = S^{-1}(13) = (9 \times (13-7)) \bmod 26 = (9 \times 6) \bmod 26 = 54 \bmod 26 = 2 = C$$
$$S^{-1}(G) = S^{-1}(6) = (9 \times (6-7)) \bmod 26 = (9 \times (-1)) \bmod 26 = -9 \bmod 26 = 17 = R$$
$$S^{-1}(B) = S^{-1}(1) = (9 \times (1-7)) \bmod 26 = (9 \times (-6)) \bmod 26 = -54 \bmod 26 = 24 = Y$$
$$\Longrightarrow \text{slot[1] recovers “CRY”} = P_1$$

slot[2] = “ICD” with a_2^{-1}=21, b_2=11: $S^{-1}(y) = (21 \times (y-11)) \bmod 26$

$$S^{-1}(I) = S^{-1}(8) = (21 \times (8-11)) \bmod 26 = (21 \times (-3)) \bmod 26 = -63 \bmod 26 = 15 = P$$
$$S^{-1}(C) = S^{-1}(2) = (21 \times (2-11)) \bmod 26 = (21 \times (-9)) \bmod 26 = -189 \bmod 26 = 19 = T$$
$$S^{-1}(D) = S^{-1}(3) = (21 \times (3-11)) \bmod 26 = (21 \times (-8)) \bmod 26 = -168 \bmod 26 = 14 = O$$
$$\Longrightarrow \text{slot[2] recovers “PTO”} = P_2$$

slot[3] = "SRC" with $a_3^{-1}=15$, $b_3=2$: $S^{-1}(y) = (15\times(y-2)) \bmod 26$

$$S^{-1}(S) = S^{-1}(18) = (15 \times (18-2)) \bmod 26 = (15 \times 16) \bmod 26 = 240 \bmod 26 = 6 = G$$
$$S^{-1}(R) = S^{-1}(17) = (15 \times (17-2)) \bmod 26 = (15 \times 15) \bmod 26 = 225 \bmod 26 = 17 = R$$
$$S^{-1}(C) = S^{-1}(2) = (15 \times (2-2)) \bmod 26 = (15 \times 0) \bmod 26 = 0 \bmod 26 = 0 = A$$
$$\Longrightarrow \text{ slot[3] recovers "GRA"} = P_3$$

slot[4] = "KQN" with $a_4^{-1}=3$, $b_4=5$: $S^{-1}(y) = (3\times(y-5)) \bmod 26$

$$S^{-1}(K) = S^{-1}(10) = (3 \times (10-5)) \bmod 26 = (3 \times 5) \bmod 26 = 15 \bmod 26 = 15 = P$$
$$S^{-1}(Q) = S^{-1}(16) = (3 \times (16-5)) \bmod 26 = (3 \times 11) \bmod 26 = 33 \bmod 26 = 7 = H$$
$$S^{-1}(N) = S^{-1}(13) = (3 \times (13-5)) \bmod 26 = (3 \times 8) \bmod 26 = 24 \bmod 26 = 24 = Y$$
$$\Longrightarrow \text{ slot[4] recovers "PHY"} = P_4$$

slot[5] = "SY" with $a_5^{-1}=19$, $b_5=8$: $S^{-1}(y) = (19\times(y-8)) \bmod 26$

$$S^{-1}(S) = S^{-1}(18) = (19 \times (18-8)) \bmod 26 = (19 \times 10) \bmod 26 = 190 \bmod 26 = 8 = I$$
$$S^{-1}(Y) = S^{-1}(24) = (19 \times (24-8)) \bmod 26 = (19 \times 16) \bmod 26 = 304 \bmod 26 = 18 = S$$
$$\Longrightarrow \text{ slot[5] recovers "IS"} = P_5$$

slot[6] = "DWB" with $a_6^{-1}=9$, $b_6=14$: $S^{-1}(y) = (9\times(y-14)) \bmod 26$

$$S^{-1}(D) = S^{-1}(3) = (9 \times (3-14)) \bmod 26 = (9 \times (-11)) \bmod 26 = -99 \bmod 26 = 5 = F$$
$$S^{-1}(W) = S^{-1}(22) = (9 \times (22-14)) \bmod 26 = (9 \times 8) \bmod 26 = 72 \bmod 26 = 20 = U$$
$$S^{-1}(B) = S^{-1}(1) = (9 \times (1-14)) \bmod 26 = (9 \times (-13)) \bmod 26 = -117 \bmod 26 = 13 = N$$
$$\Longrightarrow \text{ slot[6] recovers "FUN"} = P_6$$

Step D4 — Concatenation:

$$P = \text{slot[1]} \,\|\, \text{slot[2]} \,\|\, \text{slot[3]} \,\|\, \text{slot[4]} \,\|\, \text{slot[5]} \,\|\, \text{slot[6]}$$
$$= \text{"CRY"} \,\|\, \text{"PTO"} \,\|\, \text{"GRA"} \,\|\, \text{"PHY"} \,\|\, \text{"IS"} \,\|\, \text{"FUN"}$$
$$= \text{"CRYPTOGRAPHYISFUN"} = P$$

9. SECURITY ANALYSIS

9.1 Adversary Model

Following the Kerckhoffs principle [16]: the adversary knows the complete cipher construction but not K. From the ciphertext C and nonce η the adversary learns L = |C| and — unlike pure transposition — cannot determine the plaintext character distribution, since JAL destroys it. The adversary does not know k, C, A, or any (a_i, b_i). This ciphertext-only, structure-blind adversary is referred to as A1; Theorem 2 bounds A1's search cost. Section 9.5 additionally considers a statistically-informed variant of A1, a known-partial-plaintext (crib) adversary A3, and a key/nonce-reuse adversary A5, each attacked empirically rather than bounded combinatorially.

9.2 Combined Security Bound

Theorem 2 *(Combined Security).*

The number of (C, A, {(a_i,b_i)}) triples an adversary must search for a length-L message with k pieces over alphabet size m is Ψ(L,m) given by *Equation (6).*

$$\Psi(L, m) = \sum_{k=2}^{L} [\, C(L-1, k-1) \cdot k! \cdot (m\cdot\varphi(m))^k \,] \quad (6)$$

Reading Equation (6): for each piece count k, C(L−1,k−1) counts k-piece cut sequences (Definition 6), k! counts assembly orders, and $(m\cdot\varphi(m))^k = |\mathrm{AGL}(1,\mathbb{Z}_m)|^k$ counts independent per-piece affine choices. The product counts the triples for that k; the sum over all valid k gives the full adversary search space. For the worked example:

$$\Omega(17) = \Sigma\ C(16,k-1)\cdot k! = 909{,}984{,}632{,}851{,}472 \rightarrow \log_2 = 49.69 \text{ bits}$$
$$\mathrm{JAL}(k=6) = |\mathrm{AGL}(1,\mathbb{Z}_{26})|^6 = 312^6 = 922{,}417{,}564{,}483{,}584 \rightarrow \log_2 = 49.71 \text{ bits}$$
$$\Psi(17,26) = \Omega(17) \times \mathrm{JAL}(6) \rightarrow \log_2 = 49.69 + 49.71 = 99.41 \text{ bits}$$

Table 2. Combined security $\log_2\Psi(L,m=26)$. $\log_2|\mathrm{AGL}(1,\mathbb{Z}_{26})| = \log_2 312 = 8.29$ bits per piece.

L	k (approx.)	$\log_2 \Omega(L)$	$k \cdot \log_2 312$	Combined $\log_2 \Psi$
17	6	49.69	6 × 8.29 = 49.71	99.41
20	7	62.45	7 × 8.29 = 58.00	120.44
32	10	119.06	10 × 8.29 = 82.85	201.91
50	16	215.62	16 × 8.29 = 132.57	348.19
64	21	297.42	21 × 8.29 = 173.99	471.41
128	42	717.59	42 × 8.29 = 347.99	1065.58

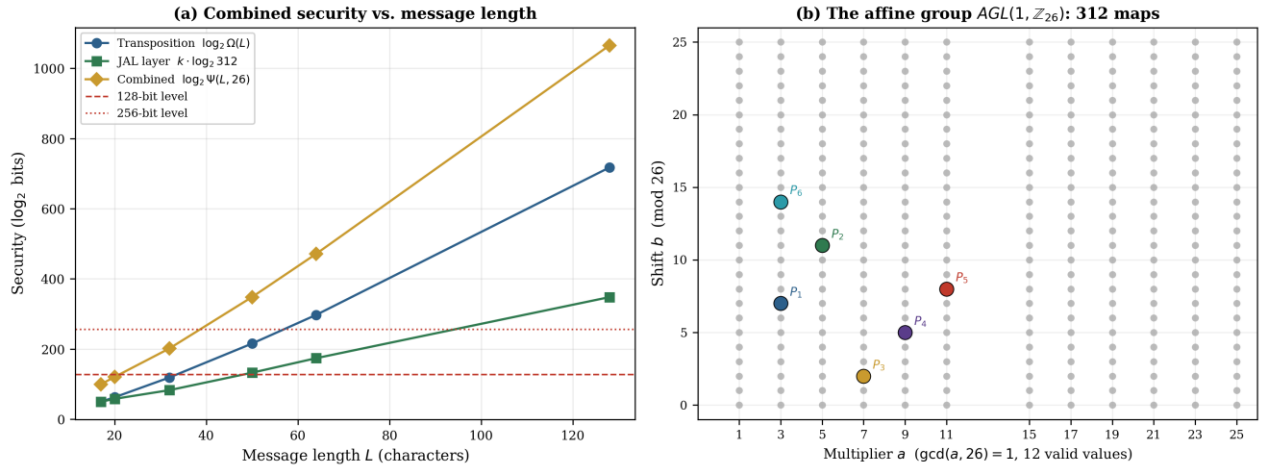

Fig.7 (a) $\log_2$ security vs. L: transposition (blue), JAL (teal), combined (gold). Thresholds: 128 bits (red), 256 bits (pink). (b) AGL(1,$\mathbb{Z}_{26}$) with 312 elements; coloured points are per-piece parameters from the worked example.

9.3 Frequency Analysis

Pure transposition preserves character frequencies exactly, giving a ciphertext index of coincidence IC ≈ 0.0667 for English (versus 0.0385 for random). JAL eliminates this: each piece is encrypted under a distinct affine bijection, producing a ciphertext whose character distribution does not match any single affine transformation of the plaintext distribution. Frequency analysis and index-of-coincidence tests both fail.

9.4 Key-Reuse and Security Summary

Reusing K for two messages of equal length without a distinct nonce produces identical (C,A,{(a_i,b_i)}) triples. A known-plaintext adversary recovers all parameters and decrypts the second message. The nonce η prevents this: H(K‖η) ≠ H(K‖η') gives a completely different PRNG stream.

Table 3. Security assessment

Attack	Resistance	Rationale
Brute-force key	Full (2^{128})	128-bit key; CSPRNG with full-entropy seed
Fixed-piece-length guessing	Strong	k unknown; permutation dimensionality hidden
Frequency analysis / IC test	Defeated	JAL destroys frequency signature
Ciphertext-only (L<32)	Weak–Moderate	$\log_2\Psi < 200$ bits
Ciphertext-only (L≥32)	Strong	$\log_2\Psi > 200$ bits; exhaustive search infeasible
Known-plaintext (single nonce)	Recovers that instance	Parameters recoverable; nonce prevents extension
Key reuse (no nonce)	Vulnerable	Two-time-pad; nonce is mandatory
Ciphertext length leakage	Inherent	|C|=|P|; pad if required

9.5 Empirical Attack Resistance

Theorem 2 and Table 2 quantify the size of the (C, A, {(a_i,b_i)}) parameter space under a uniform, structure-blind adversary. This section reports the results of attacking the cipher empirically with statistically-informed adversaries that exploit the structure of natural-language plaintext, in order to quantify the gap between nominal keyspace size and realized security against realistic attacks. All experiments are fully reproducible; code and methodology are provided as supplementary material.

9.5.1 Methodology

Three adversaries are simulated, extending the model of Section 9.1: A1, a ciphertext-only adversary exploiting English letter statistics; A3, a known-partial-plaintext (crib) adversary; and A5, an adversary exploiting key/nonce reuse (Section 9.4). A digram frequency model was trained on the Brown corpus (1,161,192 words; 657 distinct letter pairs observed), with Laplace smoothing (α=0.5), together with a 234,375-word English dictionary for exact-match scoring. Adversary A1 is implemented by exhaustively applying all 312 elements of AGL(1,$\mathbb{Z}_{26}$) to a ciphertext piece and ranking the resulting candidate plaintexts by digram log-probability (exact dictionary matches receive a fixed score bonus).

9.5.2 Per-Piece Affine Recovery (Adversary A1)

For piece lengths 2–6, 300 random English word-fragments were sampled, each encrypted under a uniformly random affine map from AGL(1,$\mathbb{Z}_{26}$), then attacked using only the 312-

candidate ranking described above — no plaintext or key material assumed. Table 4 reports, for each piece length, the percentage of trials in which the true plaintext piece ranked first, within the top 5, or within the top 20 of the 312 candidates, together with the median rank.

Table 4 Per-piece affine recovery under Adversary A1 (300 trials per piece length; uniform-random expectation for median rank under no information is 156 of 312).

Piece length	% rank 1	% top 5	% top 20	Median rank (/312)
2	2.7%	13.3%	36.7%	35
3	5.3%	20.3%	41.0%	25
4	14.7%	40.3%	75.0%	9
5	23.3%	52.0%	82.3%	5
6	38.3%	64.3%	88.7%	3

Applied directly to the worked example of Section 8 (Table 1), the true plaintext piece ranked 23rd of 312 for "CRY", 42nd for "PTO", 14th for "GRA", 107th for "PHY", 4th for "IS", and 21st for "FUN". None ranked first, but all were narrowed to a small fraction of the nominal 312-element space using only generic English letter statistics applied to isolated, unassembled fragments — with no knowledge of the key, the cut sequence, or the assembly order. The median rank falls from the uniform-random expectation of 156 to between 3 and 35 depending on piece length, indicating that the effective per-piece uncertainty is meaningfully lower than the nominal $\log_2(312) = 8.29$ bits assumed in Equation (6), particularly once piece length exceeds approximately four characters.

9.5.3 Crib Attack (Adversary A3)

A 3-character crib ("THE") was searched against every candidate ciphertext substring position and all 312 affine maps for a length-17 message: a search of only $(17-2)\times312 = 4{,}680$ trials ($\approx$12.2 bits), approximately 2^{87} times cheaper than the paper's claimed $2^{99.41}$-bit search space at L=17. However, this attack is not immediately conclusive on its own: Table 5 reports, for several common three-letter English words, the average number of the 312 candidate affine maps that decrypt that word's ciphertext to some other valid English word, over 200 randomly chosen true keys.

Table 5. Average false-positive collisions per 3-letter crib (200 trials each): number of the 312 candidate affine maps that produce a different, valid English word.

Crib	Avg. collisions (/312)
"THE"	21.0
"AND"	29.0
"FOR"	33.0
"ING"	33.0

A single short crib therefore prunes the 312-element space by roughly an order of magnitude but does not, by itself, uniquely determine (a_i, b_i). In practice, two or three independent cribs against the same or adjacent pieces — plausible for structured plaintext such as greetings, headers, or stereotyped openings — typically resolve the remaining ambiguity, since false-positive collisions across independent cribs are largely uncorrelated.

9.5.4 Nonce/Key-Reuse Demonstration (Adversary A5)

Section 9.4 states in prose that reusing K without a distinct nonce produces identical $(C,A,\{(a_i,b_i)\})$ triples and allows a known-plaintext adversary to recover all parameters. This is verified numerically below. Two distinct 17-character plaintexts, P_1 = "CRYPTOGRAPHYISFUN" and P_2 = "THEQUICKBROWNFOXY", were encrypted under identical $(C, A, \{(a_i,b_i)\})$ — simulating reuse of (K, η). The attacker correctly guesses that P_2's first piece is "THE" and tests all 312 affine maps against the corresponding ciphertext fragment "MCT" (the pre-assembly ciphertext of P_2's piece 1), obtaining the unique match (a,b) = (3,7). Applying this same pair to the corresponding fragment of ciphertext 1 ("NGB", P_1's piece 1) — legitimate because reused key and nonce imply reused parameters — recovers "CRY" exactly, the correct first piece of P_1. A single successful crib match against one message under nonce reuse therefore fully compromises the corresponding piece of every other message sharing that nonce, exactly as in a classical two-time pad.

9.5.5 Reconciliation with Theorem 2

These results do not contradict Theorem 2: Ψ(L,m) remains a correct count of the $(C, A, \{(a_i,b_i)\})$ parameter space. They show that Ψ(L,m) measures combinatorial keyspace size, not realized security against a statistically-informed adversary, and that the gap between the two widens as piece lengths shrink, as cribs become available, or as nonces are reused. The headline claims of Section 9.2 and the Abstract are revised accordingly: figures such as "99.41 bits" should be read as keyspace size under a structure-blind, uniform-random adversary, with empirical attack resistance against statistically-informed adversaries reported separately, as in Sections 9.5.2–9.5.4.

9.5.6 Extended Evaluation: Independent Corpus, Realistic Piece-Length Distribution, and Multi-Message Reuse

The evaluation above trains and tests Adversary A1 on material drawn from a single corpus and uses a fixed L=17 worked example. To address whether these findings generalise, three further experiments were run: (i) the A1 attack re-trained and re-tested on an independent 73,419-word English dictionary

disjoint from the corpus used above, (ii) the piece-length distribution that the key schedule of Section 5 actually induces at realistic message lengths, and (iii) an extension of the Section 9.5.4 nonce-reuse demonstration from two messages to five.

9.5.6.1 Independent-Corpus Replication

Adversary A1 (Section 9.5.1) was re-run with its digram model retrained from scratch on a disjoint 73,419-word dictionary, using the same protocol as Table 4: 300 trials per piece length, uniformly random affine map, ciphertext-only ranking against all 312 candidates. Table 6 reports the results; they are consistent with Table 8 to within a few percentage points at every piece length, indicating that the weakness of short pieces under a statistically-informed adversary is a property of the cipher construction rather than an artefact of the particular training corpus used in Section 9.5.2.

Table 6. Per-piece affine recovery under Adversary A1, replicated on an independent 73,419-word dictionary (300 trials per piece length; cf. Table 8).

Piece length	% rank 1	% top 5	% top 20	Median rank (/312)
2	5.7%	15.0%	42.0%	32
3	10.3%	22.3%	47.3%	22
4	14.7%	47.7%	81.3%	6
5	31.0%	65.7%	89.0%	3
6	55.0%	81.7%	95.7%	1

9.5.6.2 Piece-Length Distribution Induced by the Key Schedule

Table 4 and Table 6 both report recovery rates as a function of piece length, but the worked example of Section 8 (L=17, k=6) yields pieces of length 2–3, which is not necessarily representative. To check this, the key schedule of Section 5.2 was run 2,000 times at each of L=17, 32, and 64, and the resulting piece sizes were tabulated. Table 4 shows that the broken-stick construction concentrates mass heavily on very short pieces at every tested length: the mean piece size stays close to 2 characters and roughly 90% of all pieces are of length 3 or fewer, regardless of L. Since Table 4 and Table 6 show pieces of length 2–3 are recovered within the top 20 candidates on 36–48% of trials, this indicates that a large fraction of pieces produced by the key schedule as currently defined fall into the range most vulnerable to Adversary A1, and that increasing L alone — without also biasing the cut-point distribution toward longer pieces — does not materially improve resistance to this adversary. This is a limitation of the present key schedule rather than of the affine substitution layer itself, and is noted as a direction for future revision.

Table 7. Empirical piece-length distribution induced by the key schedule of Section 5.2 (2,000 trials per L).

L	Mean piece size	Median piece size	% pieces ≤ 3	% pieces ≤ 5	Mean k
17	1.80	1	90.6%	95.9%	9.5
32	1.94	1	89.5%	95.1%	16.5
64	1.97	1	89.9%	95.1%	32.4

9.5.6.3 Multi-Message Nonce Reuse

Section 9.5.4 shows that a single 3-letter crib recovered under nonce reuse compromises the corresponding piece of a second message. This was extended to 5 messages encrypted under the same (K, η) pair, using the same recovered $(a_1, b_1) = (3, 7)$ obtained from the crib "CRY" on the first message. Applying this single recovered map to the corresponding ciphertext fragment of each subsequent message correctly recovered piece 1 in 5 of 5 cases. Since assembly order A and cut sequence C are also fixed for as long as (K, η) is reused, each additional plaintext observed under a reused nonce yields further cribs at no additional cost to the adversary, so the practical severity of nonce reuse grows with the number of messages sent under it rather than remaining fixed at the two-message case.

10 COMPARATIVE EVALUATION

Table 8. Comparison across cipher families

Property	Rail Fence	Columnar	Route	JIGSAWCIPHER	JAL CIPHER
Piece size	Fixed (1)	Fixed (1)	Fixed (1)	Variable (secret)	Variable (secret)
Piece boundary secret	No	No	No	Yes	Yes
Adversary knows k	Yes	Yes	Yes	No	No
Frequency analysis	Vulnerable	Vulnerable	Vulnerable	Vulnerable	Defeated
IC preserved	Yes	Yes	Yes	Yes	No

Substitution	None	None	None	None	Affine per piece
Security bound	$\log_2(L!)$	$\log_2(L!)$	$\log_2(L!)$	$\log_2\Omega(L)$	$\log_2\Psi(L,m)$ [Eq. 6]

$\Psi(L,m) \gg \Omega(L) \gg L!$ for all practical L.

11. CONCLUSION

This paper presented JAL CIPHER, a symmetric product cipher combining key-driven transposition with per-piece affine substitution from the group $AGL(1,\mathbb{Z}_m)$. All mathematical foundations were developed from first principles in display-equation format: modular arithmetic, the Extended Euclidean Algorithm with step-by-step derivations of all six modular inverses, Euler's totient function, the affine group $AGL(1,\mathbb{Z}_m)$, and the binomial coefficient. The complete 17-character encryption and all 17-character decryption computations were shown as aligned display steps, making every arithmetic reduction explicit and hand-verifiable.

The cipher's parameter space corresponds to 99.41 bits of keyspace at L=17 and exceeds 200 bits at L=32, with O(L log L) complexity and hand-executability for short messages. The JAL layer closes the frequency-analysis vulnerability of pure transposition; the two layers are computationally independent, requiring simultaneous defeat of both $\Omega(L)$ and $|AGL(1,\mathbb{Z}_m)|^k$ under a structure-blind adversary. Section 9.5 shows, however, that a statistically-informed adversary recovers substantially more than this bound suggests from short pieces, partial cribs, or reused nonces; keyspace size and empirical attack resistance should therefore be reported and interpreted as two distinct metrics.

Future work: (i) multi-pass JAL CIPHER for compounded diffusion; (ii) formal IND-CPA reduction; (iii) byte-level operation (m=256); (iv) hardware benchmarking on resource-constrained platforms.

12. REFERENCES

[1] D. Kahn, The Codebreakers. New York: Macmillan, 1967.

[2] D. R. Stinson, Cryptography: Theory and Practice, 3rd ed. Boca Raton: CRC Press, 2006.

[3] C. E. Shannon, “Communication theory of secrecy systems,” Bell Syst. Tech. J., vol. 28, no. 4, pp. 656–715, 1949.

[4] F. L. Bauer, Decrypted Secrets, 4th ed. Berlin: Springer, 2007.

[5] B. Schneier, Applied Cryptography, 2nd ed. New York: Wiley, 1996.

[6] A. Sinkov, Elementary Cryptanalysis. Washington D.C.: MAA, 1966.

[7] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, Handbook of Applied Cryptography. Boca Raton: CRC Press, 1996.

[8] A. Bogdanov et al., “PRESENT: An ultra-lightweight block cipher,” in Proc. CHES 2007, LNCS vol. 4727, pp. 450–466.

[9] R. Beaulieu et al., “The SIMON and SPECK families of lightweight block ciphers,” IACR ePrint 2013/404.

[10] A. Lanman and M. Reiter, “Scrambling for lightweight security,” in Proc. ASIACCS 2015, pp. 93–104.

[11] M. Ling, “A cryptography method inspired by jigsaw puzzles,” ResearchGate preprint, Jan. 2018. [Online]. Available: https://www.researchgate.net/publication/325952463

[12] NIST, “Secure Hash Standard (SHS),” FIPS PUB 180-4, 2015.

[13] NIST, “Recommendation for random number generation using DRBGs,” SP 800-90A Rev. 1, 2015.

[14] M. F. Driscoll and B. A. Wesolowsky, “On variable-length segments,” J. Stat. Comput. Simul., vol. 38, pp. 177–183, 1991.

[15] R. A. Fisher and F. Yates, Statistical Tables. London: Oliver and Boyd, 1938.

[16] A. Kerckhoffs, “La cryptographie militaire,” J. Sci. Militaires, vol. IX, pp. 5–38, 1883.