

Supplementary Material for “Combinadic Compression: An Order-0 Enumerative Byte-Stream Codec with Size-Preserving Encryption”

Anthony J. Rasmussen

This document collects the formal security analysis (Section S1), the authenticated-encryption construction (Section S2), and the analytical models for the per-block header overhead and the coefficient-cache memory (Section S3) referenced from the main text. References to the main article are given by section number; the one external citation (the method of types) resolves against the article’s shared bibliography and is listed at the end.

S1 Formal security analysis

S1.1 Scheme and leakage

Definition 1 (Size-preserving field cipher Π). Compression maps a block b to an ordered field vector $F(b) = (f_1, \dots, f_m)$ with $f_j \in \mathbb{Z}_{M_j} = [0, M_j)$, where each modulus M_j is *prefix-determined*: it is a function of the block length and of $f_1, \dots, f_{j-1}$ only, never of f_j (the structural fact of §6.1). A mode fixes the encrypted set $E \subseteq \{1, \dots, m\}$ ($E = \{1, \dots, m\}$ in full mode; $E = \{\text{alphabet rank, position ranks}\}$ in the queryable mode of §6.2). With a per-file subkey sk and a keystream generator R (a PRF/stream cipher), encryption is, for $j \in E$,

$$y_j = (f_j + s_j) \bmod M_j, \quad s_j = R(sk, \nu_j) \bmod M_j,$$

and $y_j = f_j$ for $j \notin E$; ν_j is the field nonce of §6.3. Dec inverts front to back, each M_j recovered from the decrypted prefix.

Definition 2 (Leakage function). For a block b define

$$\Lambda_{\text{full}}(b) = L(b) = \sum_{j=1}^m \lceil \log_2 M_j \rceil, \quad \Lambda_{\text{query}}(b) = S(b) = (n_i, k_i)_i.$$

Λ_{full} is the total ciphertext length; under full encryption the counts (hence all rank moduli) are themselves encrypted, so a key-less party can parse no field boundary and L is the *only* public function of b . Λ_{query} is the count *spectrum*: the cleartext counts expose every modulus M_j , and L is a deterministic function of S , so Λ_{full} is a coarsening of Λ_{query} .

Definition 3 (Λ -restricted IND-CPA). In experiment $\text{Exp}_{\Pi, \mathcal{A}}^{\Lambda\text{-cpa}}$ the challenger draws sk and $b \in \{0, 1\}$. $\mathcal{A}$ makes encryption queries (answered under fresh, unique nonces) and submits m_0, m_1 subject to $\Lambda(m_0) = \Lambda(m_1)$; it receives $\text{Enc}_{sk}(m_b)$ and outputs b' . Define $\text{Adv}_{\Pi}^{\Lambda\text{-cpa}}(\mathcal{A}) = |\Pr[b' = b] - \frac{1}{2}|$. Π is IND-CPA secure *relative to* Λ if this is negl (negligible in the security parameter) for every probabilistic polynomial-time (PPT) adversary $\mathcal{A}$.

The equal-leakage restriction $\Lambda(m_0) = \Lambda(m_1)$ is necessary and is the essential content of “size-preserving”: a scheme that never expands cannot hide L , so L (or S) is quotiented out of the secret. Everything else must be concealed — that is what the theorem establishes.

S1.2 Confidentiality

Theorem 1 (Security relative to Λ). *Let R be the keystream generator and let each field draw $\ell_j \geq \lceil \log_2 M_j \rceil$ keystream bits before reduction. For every PPT adversary $\mathcal{A}$ making q queries there is a PPT $\mathcal{B}$ with*

$$\text{Adv}_{\Pi}^{\Lambda\text{-cpa}}(\mathcal{A}) \leq \text{Adv}_R^{\text{prf}}(\mathcal{B}) + \sum_{j \in E} \delta(M_j, \ell_j), \quad \delta(M, \ell) \leq \frac{M}{2^\ell},$$

where $\delta(M, \ell)$ is the statistical distance of a uniform ℓ -bit value reduced mod M from $\text{Uniform}(\mathbb{Z}_M)$. Under a secure R and unique (sk, ν_j) , the bound is negl ; taking $\ell_j = \lceil \log_2 M_j \rceil + \kappa$ makes $\delta \leq 2^{-\kappa}$, and rejection sampling makes it 0.

Proof. A hybrid argument. H_0 : the real experiment $\text{Exp}_{\Pi, \mathcal{A}}^{\Lambda\text{-cpa}}$. H_1 : replace $R(sk, \cdot)$ by a truly random function ρ . Because the nonce construction (Remark 1) makes every ν_j distinct within a file and every file’s sk independent, each invoked $\rho(\nu_j)$ is a fresh uniform string, so $|\Pr[H_0] - \Pr[H_1]| \leq \text{Adv}_R^{\text{prf}}(\mathcal{B})$. H_2 : replace each reduced keystream $s_j = \rho(\nu_j) \bmod M_j$ by $s'_j \leftarrow \text{Uniform}(\mathbb{Z}_{M_j})$; by definition of δ , $|\Pr[H_1] - \Pr[H_2]| \leq \sum_{j \in E} \delta(M_j, \ell_j)$. In H_2 , for each $j \in E$ the map $f_j \mapsto (f_j + s'_j) \bmod M_j$ with s'_j uniform on $\mathbb{Z}_{M_j}$ is a one-time pad: y_j is uniform on $\mathbb{Z}_{M_j}$ and independent of f_j . Hence the challenge ciphertext is a deterministic function of the moduli $\{M_j\}$ and the cleartext fields $\{f_j : j \notin E\}$ together with independent uniform randomness — i.e. of $\Lambda(m_b)$ alone. Since $\Lambda(m_0) = \Lambda(m_1)$, the two ciphertext distributions are identical and $\Pr[H_2] = \frac{1}{2}$. Summing the gaps gives the bound. $\square$

Corollary 1 (Perfect secrecy relative to Λ). *With a truly uniform keystream (rejection sampling, $\delta = 0$), $\text{Adv}_{\Pi}^{\Lambda\text{-cpa}}(\mathcal{A}) = 0$ for all (even unbounded) $\mathcal{A}$. This recovers, at whole-block granularity, the per-field Shannon statement $I(R; Y) = 0$ of §6.3.*

On the bitset reduction. The reduced-bitset encoding that produces the fields $(f_1, \dots, f_m)$ is a public, deterministic, key-independent transform—an adversary can compute and invert it unaided—so it is already subsumed in the prefix-determined moduli of Definition 1 and affects neither the leakage Λ nor the bound of Theorem 1; it fixes only how the fields and their moduli are derived, not what is concealed.

S1.3 Leakage quantification

Proposition 1 (Leakage equals spectrum entropy). *Let X be the block-content random variable and $C = \text{Enc}_{sk}(X)$. In the ideal model of Corollary 1, $C = g(\Lambda(X), U)$ with U independent of X , so*

$$I(X; C) = I(X; \Lambda(X)) = H(\Lambda(X)),$$

i.e. a key-less party learns exactly $H(S)$ bits (queryable) or $H(L) \leq H(S)$ bits (full) about the content, and no more. This is the paper’s $I(\text{content}; \text{leak}) = H(S)$, now derived.

Proposition 2 (Spectrum entropy for an i.i.d. source). *If block symbols are independent and identically distributed (i.i.d.) with law $p = (p_1, \dots, p_A)$ then $S \sim \text{Multinomial}(N, p)$ and*

$$H(S) = \frac{A-1}{2} \log_2(2\pi e N) + \frac{1}{2} \sum_{i=1}^A \log_2 p_i + o(1).$$

Thus $H(S) = \Theta((A-1) \log_2 N)$: leakage grows only logarithmically in the block length, while the block carries $NH_0(p)$ bits of content, so the fraction leaked is $O((A \log_2 N)/(NH_0)) \rightarrow 0$. Skew lowers $\prod_i p_i$ and hence $H(S)$ (the uniform law maximises it), so the homogeneous, skewed streams the codec targets leak least in absolute terms.

The estimate is validated against exact enumeration for $A = 4$ (agreement to < 0.3 bit) and gives:

Source	Law	N	content $\approx NH_0$ (b)	$H(S)$ (b)	leak/content
DNA ($A=4$)	uniform	256	512	14.1	2.8%
DNA ($A=4$)	uniform	1024	2048	17.1	0.8%
DNA ($A=4$)	Zipf(1)	1024	1836	16.7	0.9%
DNA ($A=4$)	highly skewed	1024	803	13.7	1.7%
Logs ($A=16$)	uniform	1024	4096	73.7	1.8%
Logs ($A=16$)	Zipf(1)	4096	13940	84.5	0.6%

A subtlety for highly skewed data: skew reduces the *absolute* leak $H(S)$, but because such blocks also carry less content, the *relative* leak can be marginally higher at small N ; it still vanishes as N grows. Encrypting the counts too (full mode) replaces $H(S)$ by $H(L)$, a single integer’s worth of entropy.

S1.4 Preconditions and scope

Remark 1 (Nonce invariant). The reference implementation uses $\nu = [\text{domain:16} \mid \text{block:32} \mid \text{field:16}]$ with a per-file subkey $sk = \text{KDF}(\text{master}, \text{salt})$ and a random per-file salt in the header. This guarantees the invariant Theorem 1 needs: (sk, ν_j) is unique across all encryptions, so no keystream value is ever reused (no two-time pad). The 32-bit block field bounds a file to 2^{32} blocks per (sk, salt) ; beyond that a fresh salt (hence subkey) must be drawn.

Remark 2 (Integrity is out of scope here). Theorem 1 is a *confidentiality* statement. The additive cipher is malleable, and no strictly size-preserving scheme can supply integrity: a length-preserving encryption is a bijection on the block space, so every ciphertext decrypts to a valid block and tampering cannot be detected without expansion. Authentication is therefore treated separately, as an optional expanding mode with quantified overhead (Section S2).

S2 Authenticated encryption

S2.1 Why integrity cannot be size-preserving

Proposition 3 (Size-preservation precludes integrity). *Let $\Pi = (\text{Enc}, \text{Dec})$ be correct and length-preserving: for each key k and length n , Enc_k maps the plaintext set $\mathcal{M}_n$ into the ciphertext set $\mathcal{C}_n$ with $|\mathcal{C}_n| = |\mathcal{M}_n|$ (no expansion). Then for each k , $\text{Dec}_k : \mathcal{C}_n \rightarrow \mathcal{M}_n$ is a bijection — total, with no rejection ($\perp$) — and Π has ciphertext-integrity advantage $\text{Adv}_{\Pi}^{\text{int-ctxt}} = 1$.*

Proof. Correctness makes Enc_k injective on $\mathcal{M}_n$; with $|\mathcal{C}_n| = |\mathcal{M}_n|$ it is a bijection, so its inverse Dec_k is defined on *all* of $\mathcal{C}_n$ and never rejects. An adversary obtains one ciphertext $c = \text{Enc}_k(m)$ from an encryption query and submits any $c' \neq c$ with $|c'| = |c|$ (e.g. $c' = c \oplus \Delta$, $\Delta \neq 0$). Then $\text{Dec}_k(c') = m' \in \mathcal{M}_n$ is a valid message, and $m' \neq m$ by injectivity, so the forgery is accepted. The advantage is therefore 1. $\square$

Corollary 2 (Expansion is necessary, and how much). *A scheme that can reject (has a $\perp$) must have $|\mathcal{C}_n| > |\mathcal{M}_n|$. If the ciphertext is expanded by e bits, at most a 2^{-e} fraction of length- $(n+e)$ strings are valid, so a blind forgery succeeds with probability $\leq 2^{-e}$, and no smaller bound is attainable with fewer bits. Achieving forgery probability $\leq 2^{-t}$ requires at least t bits of expansion. Integrity and strict size-preservation are thus mutually exclusive; the additive field cipher of §6, a bijection on each $\mathbb{Z}_M$, is malleable for exactly this reason — intrinsically, not incidentally.*

This is the direct answer to “just add a MAC”: a MAC *is* the t bits of expansion Corollary 2 demands, and it must live *outside* the size-preserving code, not within it.

S2.2 An optional authenticated mode

Construction 1 (Layered encrypt-then-MAC with out-of-band tags). Retain the size-preserving encryption of Section S1 as the untouched *data layer*. Partition the block stream into authentication *segments* of G consecutive blocks. For segment j with ciphertext bits c_j , compute

$$\tau_j = \lfloor \text{MAC}_{sk_{\text{mac}}}(\text{hdr} \parallel j \parallel c_j) \rfloor_t$$

(a t -bit tag under an independent MAC subkey) and store the tags $\{\tau_j\}$ in a separate integrity *sidecar* (a footer or index), never interleaved with the data blocks. Decryption verifies τ_j before releasing segment j and returns $\perp$ on mismatch.

Properties.

- *Size-preservation retained.* The data blocks are byte-identical to the unauthenticated mode; all expansion is confined to the sidecar — exactly the $t \cdot \lceil n/G \rceil$ bits Corollary 2 shows to be unavoidable.
- *Standard guarantee.* Encrypt-then-MAC with an independent key yields authenticated encryption: IND-CPA relative to Λ (Theorem 1 of Section S1) *and* INT-CTXT to 2^{-t} , now against an *active* adversary. A wrong key or a tampered block fails cleanly rather than decoding to garbage — the fail-closed behaviour §6.4 requires.
- *Tunable strength.* t trades forgery probability against overhead directly: a 32-bit tag is 4 bytes at 2^{-32} , a 64-bit tag 8 bytes at 2^{-64} .

Granularity knob. G dials the random-access unit against overhead: $G=1$ gives per-block verification at the highest cost; a moderate G verifies a segment independently at low cost; $G=n$ is one whole-file tag at ≈ 0 cost but whole-file verification only.

Remark 3 (Merkle option: per-block random access at near-zero data cost). Build a binary hash tree over the per-block tags and authenticate only the t -bit *root*. Any single block then verifies through its $O(\log_2 n)$ -node authentication path, so random access is retained *without* a per-block tag in the data path; the tree lives in the sidecar and may be materialised (about one hash per block) or recomputed on demand. Thus authentication and random access coexist.

S2.3 Overhead, and reconciliation with random access

Effective expansion is $t/(G\bar{L})$, where $\bar{L}$ is the mean compressed block size in bits. At the default $t = 64$ (forgery $\leq 2^{-64}$; halve the entries for $t=32$, double for $t=128$):

Granularity	$\bar{L}=128$	$\bar{L}=512$	$\bar{L}=2048$
Per-block ($G=1$)	50%	12.5%	3.1%
Per-segment ($G=16$)	3.1%	0.78%	0.20%
Per-segment ($G=256$)	0.20%	0.049%	0.012%
Per-file ($G=n$)	≈ 0 (one tag over the stream)		
Merkle tree	data blocks unchanged; $+t$ -bit root; $O(\log_2 n)$ hashes to verify a block		

The top row is the naive “tag every block”: up to 50–100% on small blocks — precisely why integrity cannot be bolted on for free, and why the knob matters. Dropping to per-segment or a Merkle tree brings it under a percent while keeping random access (per-segment) or full per-block verifiability (Merkle). The size-preserving property of the encoding is never touched — only the optional sidecar expands, and only by the amount Corollary 2 proves is required.

Measured (reference implementation, enwik8, ≈ 65 MB compressed). The default-off `-auth` mode confirms the closed form $t\lceil n/G \rceil$ to the byte: per file a single tag adds **29 bytes** ($4 \times 10^{-5}\%$); at 64 KB segments, **15 933 bytes** (0.024%); at 4 KB segments, **254 669 bytes** (0.39%). Compression time is unchanged within noise ($1.26 \rightarrow 1.29$ s), so the HMAC pass adds no measurable throughput cost at any of these granularities.

Remark 4 (Relation to the existing checksum, §7.6). The codec already appends a 64-bit XXH3 hash of the original stream and re-verifies it on decompression. That is a *non-cryptographic* check: it catches accidental corruption and guarantees a lossless round trip, but an adversary can forge it. The MAC layer above is its cryptographic counterpart for an *active* adversary; the two are complementary — XXH3 for cheap corruption detection, the MAC for authentication — and both are optional layers over the same size-preserving core.

S3 Analytical models

S3.1 Header overhead vs. block size

Proposition 4 (Exact per-block header). *In per-record mode a block’s header is the concatenation of the alphabet cardinality (9 bits, a value in $[0, 256]$), the alphabet rank ($\lceil \log_2 \binom{256}{A} \rceil$ bits), the most-frequent byte (8 bits), a count-width field ($w_{\text{cw}} = \lceil \log_2(1 + \lceil \log_2(N+1) \rceil) \rceil$ bits), and the $A-1$ transmitted counts, each of the shared width $\lceil \log_2(k_{\text{max}}+1) \rceil \leq \lceil \log_2(N+1) \rceil$:*

$$H(N, A) = \underbrace{17 + \lceil \log_2 \binom{256}{A} \rceil + w_{\text{cw}}}_{N\text{-independent}} + \underbrace{(A-1) \lceil \log_2(k_{\text{max}}+1) \rceil}_{\leq (A-1) \lceil \log_2(N+1) \rceil}.$$

The constant $17 = 9 + 8$ collects the cardinality and most-frequent-byte fields. Only the counts term grows with N , and it grows as $(A-1) \log_2 N$: the method-of-types bound (an N -string over A symbols has $\leq (N+1)^{A-1}$ types, [1]), matching Prop. 1’s $O(A \log N)$.

Proposition 5 (A tighter, rank-based header). *The transmitted counts are a composition of N into A positive parts, of which there are exactly $\binom{N-1}{A-1}$. Ranking the composition (as the arrangements are already ranked) replaces the fixed-width counts term by its information-theoretic minimum,*

$$H_{\text{rank}}(N, A) = 17 + \lceil \log_2 \binom{256}{A} \rceil + w_{\text{cw}} + \lceil \log_2 \binom{N-1}{A-1} \rceil,$$

saving $\Delta(N, A) = (A - 1)\lceil \log_2(N+1) \rceil - \lceil \log_2 \binom{N-1}{A-1} \rceil \geq 0$ bits per block. This is the refinement flagged as future work in §4; the table below quantifies the gap.

Relative overhead and the small-block regime. Against a payload of NH_0 bits, the header fraction is $H(N, A)/(H(N, A) + NH_0)$. It is $\Theta((A - 1)\log_2 N/(NH_0)) \rightarrow 0$ as N grows, but is substantial for small blocks: it exceeds one half below a crossover block size $N^*(A)$ where $H(N, A) \approx NH_0$. For a uniform alphabet ($H_0 = \log_2 A$):

N	$H_{\text{fixed}} / H_{\text{rank}}$ (bits) and header/(header+payload)				
	16	64	256	1024	4096
$A=4$	63/57 (66%)	69/64 (35%)	76/71 (13%)	82/77 (3.8%)	88/83 (1.1%)
$A=16$	179/104 (74%)	209/151 (45%)	240/185 (19%)	270/215 (6.2%)	300/245 (1.8%)
$A=64$	—	665/224 (63%)	792/427 (34%)	918/563 (13%)	1044/691 (4.1%)

Crossover N^* (header $\leq$ payload): 33 ($A=4$), 49 ($A=16$), 111 ($A=64$). Two riders: (i) the fixed-width counts term dominates the gap Δ — ranking the composition roughly halves the header for wide alphabets (e.g. $665 \rightarrow 224$ bits at $A=64, N=64$); (ii) the fractions above use a uniform $H_0 = \log_2 A$, the largest possible payload, so low-entropy (skewed) blocks — the codec’s target — carry proportionally *larger* header overhead, which is precisely why the effect is visible in the small-block, low-entropy experiments.

S3.2 Half-diagonal cache memory

Proposition 6 (Cache size). *For maximum block length N the cache stores the folded Pascal triangle $\{\binom{n}{k} : 0 \leq k \leq n/2, 0 \leq n \leq N\}$ as $\lfloor N/2 \rfloor$ diagonals, diagonal i holding $N - 2i$ coefficients, for*

$$M(N) = \left\lfloor \frac{N}{2} \right\rfloor \left(N - \left\lfloor \frac{N}{2} \right\rfloor + 1 \right) = \frac{N^2}{4} + O(N) \quad \text{entries.}$$

Entry (n, k) stores a `big.Int` of bit length $\lceil \log_2 \binom{n}{k} \rceil$ — so $8\lceil \lceil \log_2 \binom{n}{k} \rceil / 64 \rceil$ bytes of words — plus a cached 16-bit length and the Go object overhead (≈ 96 bytes/entry measured: the value struct, its word-slice header and length, plus the allocator’s per-object bookkeeping and size-class rounding of each backing array). Summing the widths,

$$B(N) = \sum_{n=0}^N \sum_{k=0}^{\lfloor n/2 \rfloor} \left\lceil \log_2 \binom{n}{k} \right\rceil = \frac{N^3}{6} \int_0^1 H_2(x) dx + O(N^2) = \frac{N^3}{12 \ln 2} + O(N^2) \approx 0.120 N^3 \text{ bits,}$$

using $\log_2 \binom{n}{k} \approx nH_2(k/n)$ and $\int_0^1 H_2 = \frac{1}{2 \ln 2}$. Bit-packed, these $0.120 N^3$ bits are $0.120/8 = 0.0150 N^3$ bytes; rounding each coefficient up to whole 64-bit words inflates this by $\approx 3\%$ to the measured $0.0154 N^3$ bytes. Hence the footprint is cubic:

$$\text{Mem}(N) \approx \underbrace{0.0154 N^3}_{\text{coefficient storage}} + \underbrace{24 N^2}_{\text{per-entry object overhead}} \quad \text{bytes.}$$

The constants are measured, not just asymptotic: building the cache and reading Go’s heap gives the following (the first four rows measured, the last two extrapolating the cubic term):

N	entries $M(N)$	coefficient storage	incl. object overhead
256	16 512	0.35 MB	1.80 MB
1024	262 656	17.7 MB	41.5 MB
4096	4 196 352	1.06 GB	1.46 GB
8192	16 781 312	8.35 GB	10.3 GB
16384	(extrapolated)	66.8 GB	—
32768	(extrapolated)	535 GB	—

Remark 5 (Consequences, matching §7.2 and the implementation). The cubic law explains the observed ceiling exactly: the moderate blocks of the experiments ($N \leq 8192$) need $\lesssim 8.5$ GB and are comfortable, whereas blocks in the tens of thousands demand tens to hundreds of GB and are infeasible — long before the format’s 16-bit block field (65 535) is reached. It also justifies the implementation’s explicit `maxN ≤ 65000` guard on two counts: the central coefficient’s bit length is $\approx N$, so it stays within the 16-bit `BitLength` field throughout the feasible range, and the memory wall (a ~ 4 TB cache at $N = 65000$) is reached first, so the guard rejects only already-infeasible sizes. The composition-ranking of Prop. 5 does not affect this: the cache serves the arrangement ranks, whose widths are intrinsic to the order-0 code.

References

- [1] Imre Csiszár. The method of types. *IEEE Transactions on Information Theory*, 44(6):2505–2523, 1998.