Cross-Sums Compression: A Structural Approach to Predictable, Lossless Binary Compression

Sam Caldwell

Asymmetric Effort, LLC

scaldwell@asymmetric-effort.com

Table of Contents

[Table of Contents 2](#_Toc198327400)

[Abstract 5](#_Toc198327401)

[Introduction 6](#_Toc198327402)

[Limitations of Conventional Algorithms 6](#_Toc198327403)

[Features of CRSCE 7](#_Toc198327404)

[Theoretical Foundation of Cross Sums Compression 8](#_Toc198327405)

[Matrix Representation 8](#_Toc198327406)

[Cross-Sum Matrices 8](#_Toc198327407)

[Cross-Sum Information Density 9](#_Toc198327408)

[Cryptographic Hash Chain 10](#_Toc198327409)

[Lateral Hash Information Density 11](#_Toc198327410)

[Collisions in SHA-256 11](#_Toc198327411)

[Final Output and Bit-Packing 12](#_Toc198327412)

[Compression Rates (Cr) 13](#_Toc198327413)

[Proof of Losslessness (Sketch) 16](#_Toc198327414)

[Performance and Scalability 16](#_Toc198327415)

[Comparison to Prior Art: Compression 17](#_Toc198327416)

[Decompression 18](#_Toc198327417)

[Decompression Overview 18](#_Toc198327418)

[Deterministic Elimination (DE) 19](#_Toc198327419)

[Game of Beliefs Protocol (GOBP) 20](#_Toc198327420)

[Belief Field Initialization 20](#_Toc198327421)

[Row-Master Thread Model 20](#_Toc198327422)

[Cellular Automaton (CA) Phase 20](#_Toc198327423)

[Row Hash Validation 21](#_Toc198327424)

[Convergence Criteria 21](#_Toc198327425)

[Computational Complexity Analysis 21](#_Toc198327426)

[Deterministic Elimination (DE) 21](#_Toc198327427)

[Game of Beliefs Protocol (GOBP) 22](#_Toc198327428)

[Combined Complexity 23](#_Toc198327429)

[Memory Complexity Analysis 23](#_Toc198327430)

[Performance and GPU Acceleration 26](#_Toc198327431)

[Comparative Analysis: CRSCE vs Traditional Compression Algorithms 27](#_Toc198327432)

[Algorithmic Characteristics 27](#_Toc198327433)

[Computational Complexity 29](#_Toc198327434)

[GPU Suitability 30](#_Toc198327435)

[Summary of Trade-offs 31](#_Toc198327436)

[Appendix A: CSM-to-Cross-Sum Coordinate Translation 31](#_Toc198327437)

[References 33](#_Toc198327438)

# Abstract

Traditional lossless compression algorithms struggle to offer both high efficiency and predictable performance on binary or high-entropy datasets. We present the compression phase of the Cross Sums Compression and Expansion (CRSCE) algorithm, a novel technique grounded in structural analysis of binary matrices. CRSCE bypasses entropy coding and dictionary-based schemes by representing input as a 2D matrix and extracting compact summaries via cross-sum metrics. This method achieves a consistent, deterministic compression ratio of 42.97% for fixed 512×512-bit blocks, outperforming standard compressors in uniform and pseudo-random domains. Theoretical bounds are derived, and compression properties are proven under modular arithmetic. Comparative analysis with LZ77, BWT, and arithmetic coding illustrates CRSCE’s scalability and content-independent efficiency. This paper exclusively addresses the compression process, demonstrating how structure-preserving summarization and cryptographic hashing yield high-confidence, verifiable compression guarantees.

**Keywords:** lossless compression, binary data, modular arithmetic, structural summarization, cross-sums, cryptographic hashing.

# Introduction

Lossless compression remains vital for domains where data integrity is paramount—such as archival systems, distributed databases, and cryptographic pipelines. Classical algorithms, including Huffman coding (Huffman, 1952), LZ77 (Ziv & Lempel, 1977), and DEFLATE, primarily exploit local redundancy or symbol frequency. However, these techniques degrade sharply on binary or random-like datasets, where such patterns vanish (Sayood, 2018).

This work introduces a rigorously designed structural compression scheme—Cross Sums Compression (CRSCE)—that compresses data deterministically using cross-sum reductions of binary matrices. The resulting algorithm avoids entropy dependence, providing content-invariant compression rates and mathematical verifiability.

## Limitations of Conventional Algorithms

The DEFLATE algorithm (Deutsch, 1996), a hybrid of LZ77 and Huffman coding, has become the de facto lossless standard. While efficient on structured or repetitive data, DEFLATE and Huffman will fail to compress uniformly random data (Shannon, 1948). Compression tools like BZIP2 (based on Burrows-Wheeler Transform) and ZPAQ (a context mixing compressor) fare better but lack predictable compression ratios.

## Features of CRSCE

The Cross-Sums Compression and Expansion (CRSCE) algorithm offers advantages over conventional compression:

1. **Effective, Consistent Compression of Arbitrary Inputs.** Given any two arbitrary inputs of the same size, regardless of content, the CRSCE algorithm will produce a compressed result. The user could compress photos or the output of /dev/random with the same effectiveness as a result.
2. **Predictable Compression Rates.** Given only the size of an input (or the compression blocks into which the input will be subdivided), the compression rate can be computed ahead of actual compression. Thus, a user could forecast the storage needs of the compressed output ahead of running the compression algorithm to better control costs.

Both of the above features are missing from conventional compression algorithms.

# Theoretical Foundation of Cross Sums Compression

## Matrix Representation

The CRSCE algorithm represents input data as one or more two-dimensional binary matrices called CSM:

Where—

Because CSM is a square matrix, we can then say the smallest CSM block must be , this yields a 32 KB data block. This is discussed more, later.

## Cross-Sum Matrices

From the CSM, CRSCE computes four (4) structural summaries called “Cross Sums”:

* **Lateral Sum Matrix (LSM):**
* **Vertical Sum Matrix (VSM):**
* **Diagonal Sum Matrix (DSM):**
* **Anti-Diagonal Sum Matrix (XSM):**

These projections provide invariant structural representations of the CSM data. Their mutual redundancy allows reversibility under constraints discussed in later sections. The astute reader will see a non-trivial risk of collision in these sums alone. This risk is mitigated by additional steps, covered below.

### Cross-Sum Information Density

Given the four (4) cross sums, we can compute the information density of the cross-sum representation of CSM as—

where s is the length of any side of CSM and b represents the number of bits needed to represent any cross sum in the LSM, VSM, DSM or XSM matrices.

## Cryptographic Hash Chain

To mitigate the risk of collision in a pure cross-sum environment, CRSCE adds a cryptographic hash chain (LH) computed as follows:

where the start of the hash chain is some pre-shared nonce[[1]](#footnote-1) known to all users of the algorithm. This lateral hash (LH) matrix provides a means of performing row-by-row collision detection and verification. The cryptographic hash chain method used to derive the LH matrix provides a collision detection mechanism for the entire CSM solution, while also further reducing the probability of collision.

### Lateral Hash Information Density

Since each row of CSM must have a hash in the Lateral Hash (LH) matrix, LH must be of s rows itself. Further, because LH is composed of s-number of SHA-256 hashes, and because the output size of the SHA-256 algorithm is 256 bits, we can calculate the number of bits in LH as—

### Collisions in SHA-256

While SHA-256 remains a secure hash function as of 2025, its finite output space (256 bits) necessarily invokes the pigeonhole principle, particularly when compressing fixed-size binary matrices such as the 512-bit rows in CRSCE's Cross-Sum Matrix (CSM). Each row is effectively a unique arrangement of 512 bits; however, these configurations must map to one of possible SHA-256 outputs. Though the cryptographic design of SHA-256 renders practical collision discovery computationally infeasible (Dang, 2012), the mathematical certainty of potential collisions remains. To mitigate this theoretical risk and enhance verification strength, CRSCE implements a chained hash construction. Specifically, each row hash ​ is derived as:

This chaining mechanism ensures that each row’s hash is not only bound to its own content but also transitively tied to all preceding rows, reducing the risk of undetected collision-based tampering. By coupling the structural integrity of the CSM with cryptographic hash dependencies, CRSCE reinforces the fidelity of decompression and integrity verification, even in scenarios where theoretical hash collisions may exist.

## Final Output and Bit-Packing

Once the cross-sums matrices (LSM, VSM, DSM and XSM) have been calculated and the lateral hash (LH) matrix has been computed, the compressed output must be serialized into a packed bit stream. For optimal speed, the output is laid out as LH, LSM, VSM, DSM, XSM, since the LH matrix is byte-aligned and easily serialized. The remaining cross-sums, however, must be packed through a slower process.

Because the value b is not byte-aligned (b=9 for s=512), the compressor must read bits from the cross-sum matrices then push the resulting bits to the output buffer. This incurs a non-trivial amount of processing time. However, this can be parallelized to some extent by interleaving the cross-sum matrix elements as follows:

where is the *i*th bit in the serialized sequence. This interleaving means four bit-reader threads can execute concurrently to pop bits from their respective cross-sum matrix elements then push them to a serialization queue for their respective cross-sum. A fifth independent thread can read from these queues in round-robin order (LSM, VSM, DSM, XSM) to pop one bit at a time from each queue then write it to the output queue (*O*). What results is an bit stream of packed cross-sum values regardless of the b-value written to the output file in less time than if the cross-sums were packed sequentially.

## Compression Rates (Cr)

Let the compression rate (Cr) be the ratio of compressed output to uncompressed input:

Given the above, we can define as the total compressed output size, calculated as—

This is the sum of our cross-sum information density and lateral hash information density, as defined earlier.

Next, we can assume our input size (I) is calculated as—

where we use the CSM size (s) to determine the input size in order to account for any padding which may be necessary to make the raw input fit the square CSM matrix. From these two terms, we can calculate Cr as—

Plugging in s=512, from which we determine that b=9, we find—

This is the empirical benchmark from the CRSCE implementation. A study of this compression rate would show that as the value of s increases, the compression rate Cr also increases. The following table projects the compression rates for various block size (s) values:

|  |  |  |  |
| --- | --- | --- | --- |
| Block Size (s) | b | Total Data Size (s2) | Compression Rate (Cr) |
| 256[[2]](#footnote-2) | 8 | 2562=8KB | -14.5% |
| 512 | 9 | 5122=32KB | 42.97% |
| 1024 | 10 | 10242=128KB | 73.12% |
| 2048 | 11 | 20482=512KB | 90.54% |

As demonstrated, s < 512 is not feasible and will result in data inflation rather than data compression. However, as the compression rate increases, the computational complexity required for decompression also increases, as does the risk of collision. For purposes of this paper, CRSCE is limited to s=512.

NOTE: We explicitly avoided any compressed file header or other overhead in our compression rate calculation as this is an implementation detail reserved for the end user. Most use cases will need to write header information such as the value of s, the name of the compressed data set and other metadata. But there are other use cases where the implementing user may make these values constant. To facilitate a more general use of CRSCE, the compression rate excludes this overhead.

## Proof of Losslessness (Sketch)

Let LSM, VSM, DSM and XSM be the four cross-sum vectors and LH the lateral hash chain, as described earlier. Because SHA-256 is preimage-resistant and collision-resistant (Dang, 2012), the probability of an alternate matching LH is negligible. Therefore, is uniquely decodable with probability . See Merkle-Damgård construction proof (Merkle, 1989).

## Performance and Scalability

CRSCE performs all computations in linear time , with parallelizable phases:

* Cross-sum vectors: tasks over s elements
* Hash chain: sequential without parallelization

By fixing s=512, CRSCE supports consistent performance metrics across all datasets.

# Comparison to Prior Art: Compression

The following table illustrates the comparative performance of CRSCE to conventional compression:

|  |  |  |  |
| --- | --- | --- | --- |
| Algorithm | Average Binary Compression | Predictability | Complexity |
| DEFLATE | <5% | ✗ |  |
| LZMA | 10-30%  (content-dependent) | ✗ |  |
| BWT+RLE | <20% | ✗ |  |
| CRSCE | 42.97% (Fixed) | **✓** |  |

Table 1 - Sources: Salomon & Motta (2010), Sayood (2018)

# Decompression

The decompression phase of the CRSCE algorithm must invert a structure-preserving transformation that maps a binary input matrix into a compact representation of its lateral, vertical, diagonal, and anti-diagonal cross-sums along with row-wise cryptographic. The prior decompression algorithm developed for CRSCE relied on an informed sieve and sift, which was neither scalable nor efficient. What follows proposes a principled approach combining deterministic rule-based inference and probabilistic graphical modeling, using a novel Game of Beliefs Protocol (GOBP) as the backbone of post-elimination inference. The goal of this improved algorithm is to reduce the time required to decompress information and thereby make CRSCE a viable general-purpose compression alternative to traditional algorithms. Where the original sieve-and-sift decompression algorithm was too slow for anything beyond archival use, this approach enables broader applicability by significantly improving performance through deterministic inference and GPU-accelerated probabilistic resolution.

## Decompression Overview

Decompression proceeds as a loop of the following stages until convergence:

1. Deterministic Elimination (DE):

Apply exact inference rules to reduce entropy.

1. Game of Beliefs Protocol (GOBP):

Use cellular automation with belief propagation and row-local updates to resolve uncertain bits.

1. Verification:

Validate resolved rows using SHA-256 hash checks against the LHASH matrix.

Convergence occurs when all rows are verified or the system reaches a stable fixed point.

## Deterministic Elimination (DE)

DE aims to resolve all bits that can be deterministically inferred. This includes:

* Rows, columns, diagonals, or anti-diagonals with sums equal to 0 or s.
* Lines with exactly one unknown bit, inferable by subtraction.
* Early elimination of known-pattern rows via LHASH lookup (e.g., alternating bit patterns).

The output of DE is a partially filled Cross-Sum Matrix (CSM), with many bits fixed and entropy localized in more ambiguous rows. Each element of the CSM includes a lock-bit. Once a bit is solved—using DE or any other mechanism—it is locked via this lock-bit to prevent further modification by downstream inference phases.

## Game of Beliefs Protocol (GOBP)

### Belief Field Initialization

After DE, each unresolved CSM bit is assigned a probability , estimated through Loopy Belief Propagation (LBP). The factor graph consists of variable nodes for each unknown bit and factor nodes representing cross-sum constraints.

### Row-Master Thread Model

Each unsolved row spawns a row-master thread that:

* Receives the fixed bit budget from LSM.
* Initializes the row with exactly that number of set (1) values, based on belief strengths.
* Spawns cellular automaton (CA) threads to manage slices of the row.

Budgeted set (1) values budgeted for a row cannot be transferred to another row, as this would violate the requirements of

### Cellular Automaton (CA) Phase

Each CA thread attempts to reduce local constraint violations by migrating set (1) values within the row:

* Prioritize cells with high belief and minimal residual pressure from VSM, DSM, and XSM.
* Preserve the row's bit budget at all times.
* Synchronize via GPU shared memory within row-local threadblocks.

### Row Hash Validation

Each row-master computes the SHA-256 hash of the current row. If it matches the LHASH entry and cross-sum consistency is preserved, the row is locked and marked as solved. Otherwise, it continues updating for a fixed number of steps or until convergence.

### Convergence Criteria

The protocol terminates when:

* All rows pass SHA-256 verification.
* Or no further changes occur in the belief field after a maximum iteration threshold.

Optionally, rows that fail to converge may be flagged for fallback methods, though brute-force phases are omitted in the current design.

## Computational Complexity Analysis

### Deterministic Elimination (DE)

Let be the dimension of the Cross-Sum Matrix (CSM), fixed at throughout our implementation.

DE iteratively resolves matrix bits using cross-sum rules and hash-based pattern resolution. Each of the four cross-sum vectors (LSM, contains entries. Each entry potentially involves scanning up to ss bits, and updates propagate as elements are solved and locked. This gives:

DE time complexity:  per pass, up to  passes⇒.

In practice, the number of effective passes is reduced by the locking mechanism, which prevents reprocessing of resolved rows/columns.

The DE phase also attempts hash-based resolution for known patterns, requiring. SHA-256 hashes (one per row), each of cost , thus also .

### Game of Beliefs Protocol (GOBP)

Post-DE, remaining bits are handled using probabilistic inference and local constraint optimization. GOBP consists of:

1. **Factor Graph Initialization**:

* Build a bipartite graph with up to variable nodes and factor nodes.

1. **Loopy Belief Propagation (LBP)**:

* For each edge (up to 4), messages are exchanged for iterations:

, assuming  is constant-bound (e.g., 10–20).

1. **Cellular Automata (CA)**:

* Each row contains ss bits. Local adjustments (bit flips, shifts) are attempted under strict LSM budgets.
* Each row thread executes R rounds: , with R constant (e.g., 100).

1. **Row Validation**:

* Each row is hashed and compared to LHASH:  per iteration.

Total GOBP complexity per block is:

With fixed constants T, R, this is asymptotically .

### Combined Complexity

Combining both stages, we get a total decompression complexity per block of . This upper bound is consistent with empirical convergence behavior, particularly for high-entropy inputs.

## Memory Complexity Analysis

Each decompression block requires:

* CSM matrix:
* Cross-sum vectors:
* LHASH matrix:
* Belief matrix:
* Temporary GPU buffers and row automata state:

Total approximate memory per block:

This assumes a fixed block size s.

## Performance and GPU Acceleration

The CRSCE decompression process is explicitly designed for high-parallel throughput on modern GPUs:

* **DE Phase**:

Amenable to thread-parallelism, particularly for parallel reductions along rows, columns, and diagonals.

* **GOBP LBP Phase**:

Maps naturally to CUDA warp/block architectures using message-passing semantics.

* **CA Threads**:

Executed as row-local thread blocks using shared memory for synchronization and bit-shifting operations.

* **Hash Validation**:

Can be batched and overlapped using warp-shuffle and crypto intrinsics.

## Comparative Analysis: CRSCE vs Traditional Compression Algorithms

In this section we compare the Cross Sums Compression and Expansion (CRSCE) algorithm alongside traditional compression algorithms. The analysis which follows is purely theoretical. It is derived from designs, published computational complexity, and the structural properties of the CRSCE and comparative algorithms. No empirical runtime or throughput data is presented in this algorithm. All performance metrics such as time and memory complexity are analyzed under worst-case assumptions using formal algorithmic models. Readers should treat this comparison as a high-confidence analytical projection, not a benchmark.

This projection compares the CRSCE algorithm to DEFLATE (used by gzip), LZMA and BWT+RLE. The following compares these methods across several dimensions relevant to high-entropy, verifiable, and scalable compression workflows. However, it should be worth saying that CRSCE is fundamentally different from traditional algorithms. Nonetheless, there are no comparable algorithms with the novel approach to data compression which could provide a closer comparison.

### Algorithmic Characteristics

The following table presents a theoretical comparison of CRSCE and traditional compression algorithms to highlight the structural and computational distinctions of CRSCE. Of particular importance is CRSCE’s strong losslessness guarantee and its independence from source entropy. Unlike conventional algorithms, which stem from Shannon’s foundational 1948 work that ties compression efficiency to redundancy and statistical repetition in data (Shannon, 1948), CRSCE derives its effectiveness from deterministic structural properties such as cross-sums and cryptographic hash constraints, CRSCE derives its abilities from bit sums.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Feature | CRSCE | DEFLATE (gzip) | LZMA | BWT + RLE |
| Compression Strategy | Structural + Hashing | LZ77 + Huffman | LZMA (LZ77 + Range coding) | Transform + Run-Length |
| Predictable Output Size | ✓ (based on block size) | ✗ (depends on input) | ✗ | ✗ |
| Content-Independence | ✓ | ✗ | ✗ | ✗ |
| Verifiability via Hash | ✓ (cryptographic LHASH) | ✗ (optional checksums only) | ✗ (optional CRC) | ✗ (no intrinsic hash) |
| Losslessness Guarantee | Proven via SHA-256 + sums | Proven but entropy-bound | | |

CRSCE has predictable output size regardless of the input. Knowing only the input message size or block size, the compression rate can be estimated, providing valuable capabilities to the end user. Traditional algorithms lack this ability. Further, larger and larger block sizes increase the compression rates of the CRSCE algorithm, where traditional algorithms only increase their rates if the entropy of the input increases.

### Computational Complexity

The trade-off of CRSCE guaranteed, predictable compression rates and entropy-independence is a greater computational complexity, especially when it comes to decompression. The complexity of decompression is the primary reason the algorithm has taken more than thirty years to develop. In the following table, we see the computational complexity, illustrated:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Metric | CRSCE Decompression | DEFLATE | LZMA | BWT + RLE |
| Time Complexity  (Worst Case) |  |  |  |  |
| Time Complexity  (Best Case) |  |  |  |  |
| Memory Complexity | (fixed) |  |  |  |
| Entropy Sensitivity | None | High | High | High |

The comparative performance of CRSCE to other algorithms is significant. Greater speeds can be achieved using traditional compression. This is even after the significant improvements of the GOBP over the Radditz Sieve algorithm.

### GPU Suitability

To improve decompression speeds, CRSCE is designed to use Graphics Processing Units (GPUs). The GPU allows massive parallel math operations needed for GOBP. We hope that this will allow CRSCE to become suitable for mainstream. We believe that this will also create the potential for quantum computing applications. With this future in mind, we evaluate GPU Suitability for all of our algorithms in the following table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Feature | CRSCE | DEFLATE | LZMA | BWT + RLE |
| GPU Parallelism | Excellent (SIMD/CA/LBP) | Poor | Poor | Moderate |
| Thread Decomposition | Block-row factorized | Sequential bitstream | Sequential | Transform parallelism |
| Suitable for Real-time | ✗ (bounded by GOBP convergence) | ✓ | ✓ | ✓ |

### Summary of Trade-offs

* **CRSCE** performs predictably regardless of input signal entropy.
* **DEFLATE and LZMA** remain faster and more general-purpose for low-entropy or real-time applications.

# Appendix A: CSM-to-Cross-Sum Coordinate Translation

As a two-dimensional matrix, CSM elements are addressed by row, column ordered pair coordinates. However, the CSM coordinates must be translated to one-dimensional coordinate systems for the cross-sums matrices. For simple cross-sums matrices like LSM and VSM, the translation is fairly simple. LSM addresses elements of CSM for a given row (*r*) and any value of . Likewise, VSM addresses elements of CSM for a given column (*c*) and any value . But for the DSM and XSM matrices the translation is more complex.

# References

Dang, Q. (2012). Recommendation for applications using approved hash algorithms. NIST Special Publication 800-107. https://doi.org/10.6028/nist.sp.800-107r1

Deutsch, P. (1996). DEFLATE Compressed Data Format Specification version 1.3. IETF RFC 1951. https://www.ietf.org/rfc/rfc1951.txt

Huffman, D. A. (1952). A method for the construction of minimum-redundancy codes. Proceedings of the IRE, 40(9), 1098–1101. https://doi.org/10.1109/JRPROC.1952.273898

Merkle, R. C. (1989). One way hash functions and DES. In Advances in Cryptology—CRYPTO’89 (pp. 428–446). Springer.

Salomon, D., & Motta, G. (2010). Handbook of Data Compression. Springer.

Sayood, K. (2017). Introduction to Data Compression (5th ed.). Morgan Kaufmann.

**Shannon, C. E. (1948).** A mathematical theory of communication. Bell System Technical Journal, 27(3), 379–423. https://doi.org/10.1002/j.1538-7305.1948.tb01338.x

Stevens, M., Bursztein, E., Karpman, P., Albertini, A., & Markov, Y. (2013). The first collision for full SHA-1. In **Advances in Cryptology – CRYPTO 2017** (pp. 570–596). Springer. https://doi.org/10.1007/978-3-319-63688-7\_19

Ziv, J., & Lempel, A. (1977). A universal algorithm for sequential data compression. IEEE Transactions on Information Theory, 23(3), 337–343.

1. The nonce is not secret, nor can it be. The nonce must be known to allow hashes to be calculated during the collision detection of the decompression process. [↑](#footnote-ref-1)
2. s=256 is an invalid s-value since it is lower than the boundary 292, as established earlier. It is included in this table to highlight how the algorithm will perform without this boundary. [↑](#footnote-ref-2)