TIME SPACE REDUCER · VERSION 0.10.2 · DEMONSTRATION BETA

# TSR

Compress further where others stop, in a time you choose, with an archive that is a function of the data.

## White paper

The TSR method: three ideas, the measurements behind them and the limits stated. For those who evaluate the technology before the product.

- **Time as the unit of compression.** You choose how much time to spend; TSR calibrates the machine and the real files and states duration, space and extraction time before starting.
- **The incompressible layer is taken apart.** PDF, Office, PNG, JPEG and video are re-encoded from the inside and come back byte for byte: first among ten tools on office documents.
- **The archive is a function of the data.** Same bytes on three architectures, a frozen format with a specification and vectors, integrity per block and per file, use without extracting.


| DOCUMENT | EDITION | DISTRIBUTION | OWNER |
|---|---|---|---|
| TSR-WHP-06-EN | 27 September 2026 | Public | Matteo Luigi Feroldi |


**PUBLIC DOCUMENT**
It describes the method and the results; the inner workings of the codec stay in the technical dossier, delivered under a non-disclosure agreement. Every figure carries its corpus, machine, profile and date.

IN BRIEF

# Strong compression already existed. What was missing was making it fast enough to use, and honest enough to trust.

TSR (Time Space Reducer) is a compressor and archiver for macOS, Windows and Linux, with an engine written from scratch in Rust. These pages set out the method that distinguishes it, the measurements behind it and the limits it states, for those who evaluate the technology: architects, infrastructure leads, technical reviewers.

**47.888%**
**Office documents**
1,322 real files from public sources: first among ten tools. Best rival: 51.98%

**21.975%**
**enwik8**
One hundred megabytes of text, in 19 seconds on all cores. 7-Zip at maximum: 24.80% in 41

**−18.2%**
**JPEG photographs**
Re-encoded from the inside and returned bit-identical, where generic archivers stay at zero

**3**
**Architectures**
Apple Silicon, x86-64 and server-class ARM64 produce the same bytes, verified with fingerprints

First idea

### Time as the unit

You choose how much time to spend; TSR calibrates machine and real files, takes the strongest codec that fits and states duration, space and extraction time up front. The strong family runs on independent blocks, one per core.

Second idea

### The incompressible layer is taken apart

Already-compressed files (PDF, Office, PNG, JPEG, video, gzip) are taken apart down to the layer that makes them so; the real data goes to the codec and the layer is put back byte for byte, verified on every file.

Third idea

### The archive is a function of the data

The same input yields the same bytes on three architectures; every block and every file carries its fingerprint; the format is frozen with a specification and vectors; the archive is used while it stays compressed.

## Contents


01 The problem 3
02 Time as the unit of compression 4
03 The incompressible layer 6
04 The archive as a function of the data 8
05 The data stays usable 10
06 Evidence and method 11
07 Declared perimeter 13
08 Verifying and adopting 14
– Glossary and references 15

# 01 Classic compressors have run out of headroom

And the part of the data that grows fastest is already compressed: to a traditional archiver it is indistinguishable from noise.

## Space is the cost item that grows on its own

Every copy of a piece of data pays its compression ratio: the primary archive, the backup, the off-site replica, the tape, every transfer over the network. A point of ratio gained once acts on all copies, for the whole life of the data. Capacity is bought in steps, backup windows stay what they are and bandwidth is paid by the byte: the ratio is the one lever paid for once that pays back every night.

## Two families, and a practical limit

Everyday archivers (zip, 7-Zip, xz, zstd, brotli) belong to the family born with DEFLATE in 1993: they look for repetitions and encode what remains. For years they have been refining the same ideas and move by fractions of a point: on enwik8, one hundred megabytes of text, they sit between 24.8 and 25.7%. The compressors that beat them (PAQ, zpaq, cmix) belong to the context-mixing family, which predicts every bit from dozens of clues and weighs them together: it is worth three to eight points more and costs ten to a thousand times the time on a single core. For twenty years it has remained a laboratory tool.

## The part that grows is already compressed

PDF, Office documents, PNG, JPEG, video, gzip archives: the formats in which data is produced already contain a layer of compression. A generic archiver treats them as noise and gives back 96 to 99.5% of the original: it has run out of headroom. What remains lies inside those formats, in the coding layers of 1992 and 1993, and it is reached only by putting everything back in its place, byte for byte.

WHAT ARCHIVERS GIVE BACK WHERE THE DATA IS ALREADY COMPRESSED

| FILE SET | 7-ZIP AT MAXIMUM | ANOTHER RIVAL |
|---|---|---|
| **JPEG photographs (gallery, 41.6 MB)** | ≈ 100% | generic ≈ 100% |
| **Photos and videos from a phone (“family” scenario)** | 97.41% | zip 98.02% |
| **Films and masters (“videomaker” scenario)** | 98.68% | zip 99.45% |
| **Office documents (1,322 files, 199.5 MB)** | 52.47% | zpaq 51.98% |

**Three costs governed by the ratio**

Space occupied on every tier of storage; the length of backup and replication windows; bytes transferred over the network, paid per use.

**The enwik8 ladder**

bzip2 29.01% · zstd 25.27% · 7-Zip 24.80% · PPMd 21.41% · **TSR 21.975%** · bsc 20.80% · zpaq 19.63%. All on the same machine, at the documented maximum levels; page 5 puts the times alongside.

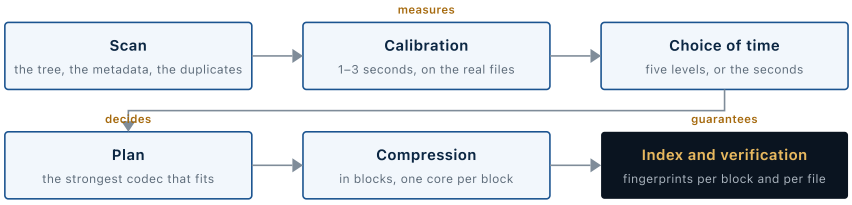
**Where this document comes from**

From a project that wrote every figure with its measurement beside it and published negative verdicts like positive ones. Pages 11–13 give method, corpora and limits.

# 02 Time as the unit of compression

The user chooses how much time to spend; TSR converts that time into ratio on their machine and their files, and states the result before starting.

Levels 1 to 9 in an archiver describe the algorithm: the same level takes a minute on one computer and ten on another, and the space returned is discovered at the end. Time is the one quantity a user knows how to negotiate, and the natural one for a codec that converts time into ratio.




The flow of a time-budgeted compression: calibration and plan repeat on every batch, and when the job ends the estimate is compared with the result.

## Calibrating on the real machine, on the real files

Before compressing, TSR samples the real files of the batch: strips by size band, weighted by the mass of bytes each band carries, and classes by type (text, binary, already-compressed media). On those samples it measures, for every available codec, speed and ratio on the machine doing the work at that moment. Then it probes the filters on the real files, counting what they yield and cost, and estimates the duplicates. All of it takes one to three seconds.

## The plan

The time budget is compared with the estimates and the strongest codec that fits wins. Five levels (Fastest, Fast, Medium, Slow, Slowest) are multiples of the machine's minimum time (×1.1, ×3, ×8, ×20, ×60); a budget written in seconds is taken literally. Before starting TSR shows the estimated duration, space saved and extraction time; at the end it compares the estimate with the result.

## Estimates err on the safe side

The direction of the error is declared: the ratio is measured on strips and whole blocks warm up the model only once, so the real result is almost always better than announced. On a user's two ZIP archives the balanced profile estimated 12 seconds against 13 real and 5.3 MB of savings against 5.27; on a real mail backup the Medium level declared 21 seconds against 20. The extraction-time estimate sits between −2% and +16%.

### Four fixed-codec profiles

For scripts and services there are also profiles without calibration: fast, balanced, backup, maximum. A service is spared two and a half seconds of measurement on every job, and with a fixed codec the same input gives the same archive on every machine.

### When the content is already compressed

Calibration says so at once, before starting: the plan line reports what the probes measured ("pdf 1.9 MB/s · stream ×16 · −15%", "containers: not applicable") and which level is worth proceeding with.

### A budget out of reach

A time shorter than the achievable minimum is reported together with the minimum, and with what it would achieve.

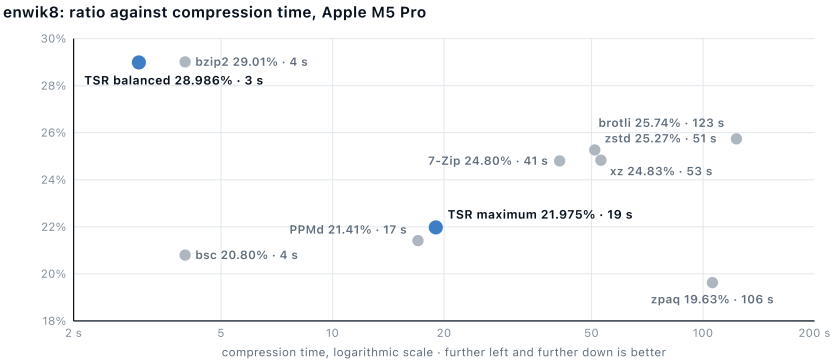
### The codec gives nobody a discount

Calibration times the codecs interleaved strip by strip: a processor drift during the measurement hits all of them alike, and the ratio between speeds, on which the levels decide, stays stable. Verified on 22 September 2026: eight real runs out of eight on the same plan.

## Independent blocks, one block per core

Context mixing is serial by nature: every bit depends on the tables updated by the previous bit. TSR makes it usable on today's machines by splitting the input into independent blocks (2 to 16 MiB depending on the profile, always within a file) and encoding one per core. The models restart from zero at every block: it costs between 1 and 2% of ratio and buys three things at once: parallelism in writing and reading, random access to any point of the archive, and the localisation of damage to a single block. On Silesia the block pipeline compresses 3.4 times faster than per-file batches, extracts 5.5 times and verifies 5.7.

The result on the reference corpus: enwik8 at 21.975% in 19 seconds on all cores of an Apple M5 Pro, where zpaq does 19.63% in 106 seconds on a single core and 7-Zip at its maximum level 24.80% in 41.




Ratio as a percentage of the original against compression time; rivals at their documented maximum levels, measurements of 8 September 2026. **Times are single-run** and give the order of magnitude; sizes are deterministic.

## Three lanes for three needs

The codec has three lanes, and the profile or the plan chooses. The fast lane works on bytes with a grammar of repetitions and table-driven entropy coding: on text 35.9% at 119 MiB/s per core in compression and 617 in decoding, the same ratio as zstd at level 3 at about half its speed. The intermediate lane finds repetitions with optimal parsing and encodes what remains with contexts: on office documents 50.925% in 23 seconds, where zpaq takes 140 for 51.98%. The strong lane is context mixing with election: on wide budgets the codec tries several candidates on the same block and keeps the shortest.

### The graphics accelerator

The search for repetitions can run on the GPU (Metal, Vulkan) under a contract: every result is verified on the CPU before use, and the archive bytes are identical with and without. On the maximum profile the time is the same, on or off, because the bottleneck is context mixing; on the balanced profile it runs 1.4 to 2.9 times faster on the usage scenarios. The command `tsr gpu-check` demonstrates the contract on the evaluator's machine.

### Memory

The memory budget decides how many blocks run together, that is the speed, never the bytes: the same input gives the same archive under any ceiling. With no instruction the budget is an estimate from the cores (between 1 and 8 GB); whoever asks for an explicit ceiling gets a contract, measured at 1.79 GB under a 2 GB ceiling on macOS, paying up to three times the time on trees where the filters do most of the work.

### The cost of blocks, measured and accepted

Solid mode and 64 MiB blocks were tried: 0.335 points of ratio, at the price of random access. Rejected, and the verdict is written.

# 03 Taking the incompressible layer apart, and putting it back byte for byte

To a generic compressor an already-compressed file is eight bits per byte. TSR takes it apart down to the layer that makes it so, re-encodes the real data and proves it can put it back before writing.

## Wrapper and compressed sections

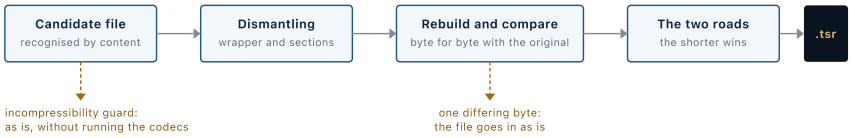
A compressed file splits into a wrapper (headers, indexes, the parts TSR cannot reproduce) and compressed sections. The sections are opened and the raw data goes to the profile's codec; for every section the exact combination of parameters that reproduces it is recorded. On extraction the wrapper comes back as it was and the sections are recompressed with the same library, embedded in the program and identical on every system. This is the road for gzip, ZIP and Office containers, PNGs and PDF streams.

For JPEG the layer to take apart is the Huffman coding of 1992: TSR decodes it down to the transform coefficients, never down to the pixels, and re-encodes them with an adaptive model that predicts every coefficient from the neighbouring blocks. For H.264 video the layer is the arithmetic coding, re-encoded one group of pictures at a time. In every case the extracted file is bit-identical to the original.

## The three guards

- **Every file is rebuilt before writing.** The filter is applied, the file is reassembled and compared byte for byte with the original; at the first differing byte it goes in as is.
- **The two roads are compared.** The dismantled stream and the file as it is are both compressed with the profile's codec, and the shorter wins, file by file. Without this comparison an archive could reach 200% of the original: found by measurement on 6 August 2026, fixed the same day.
- **The incompressible is recognised before any work.** A guard samples the head, quarters and tail of every block and answers "as is" before running the codecs: on random data the maximum profile saves 94% of the time.

A filter can therefore only make the archive smaller.




The path of an already-compressed file; the two dashed exits are what make the operation safe.

### What stays out, measured

Files written with proprietary variants of the algorithm (some PDFs, encrypted streams, images optimised with zopfli or oxipng), formats with no layer to take apart (MP3, AAC, JPEG 2000, WebP) and anything truly random or encrypted go in as they are. Coverage is per stream: the rest of the batch still gains.

### The library that recompresses

The zlib that reproduces the sections is embedded in the program and pinned by the lock file. An update that changed one byte of its output would make archives unrebuildable: the tests that freeze the streams block it.

### On extraction

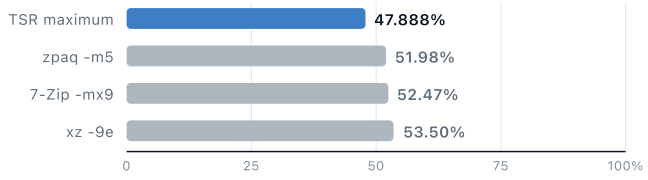
Dismantled files are reassembled by a team of rebuilders, several files at once, within the memory budget: recompressed gzips extract at 178 MB/s aggregate against 23 on a single core (measured 22 September 2026), identical to the source.

## Ten file families, each with its measurement

| FAMILY | WHAT IS TAKEN APART | MEASURED YIELD |
|---|---|---|
| **JPEG** | The Huffman layer, down to the transform coefficients | −18.2% on standard photos, −11.8% on progressive ones; a JPEG-only specialist ≈ −22% |
| **gzip, zlib** | The DEFLATE stream, with its parameters recovered and verified | −32% on the mass of reproducible gzips |
| **ZIP, Office, JAR** | Every DEFLATE member, wrapper intact | −43.5% on modern Office containers and JARs |
| **PNG** | The zlib stream of classic images | −22.9% on classic PNGs; optimised ones travel as they were |
| **PDF** | Only the DEFLATE streams that reproduce | From 87–99% down to 50–79% with producers' standard zlib |
| **H.264 video (MP4, MOV, MKV)** | The arithmetic coding, per group of pictures | 2–3.7% on films and footage, where rivals are at zero; a guard skips videos that would yield little |
| **Executables (x86, ARM64)** | A filter on jump addresses | Executable family 26.66% against 28.22% for xz |
| **PCM audio (WAV, AIFF, CAF)** | Linear prediction per block, channels separated | Audio corpus 59.444% on maximum; on pure PCM six points ahead of FLAC -8 (11.73% against 17.85%) |
| **Containers with irreproducible DEFLATE** | The best of three roads per member, decided on a sample | −18.5% on containers previously left intact |
| **Raw 16-bit images** | A predictor on pixels, residual in byte planes | CT −3.63%, MRI −1.33%; refuses where it would get worse |

## The product's use case: office documents

199.5 MB in 1,322 files from public sources (modern and legacy Office, PDFs from several producers, EPUB, JAR), with 34.9% of the mass inside DEFLATE containers. The ten tools stop between 51.98 and 62.40%; TSR goes down to 47.888%, and the balanced profile does 50.925% in 23 seconds where zpaq takes 140 for 51.98.




Corpus built from public sources (ECMA, Apache POI, Maven, Gutenberg, RFC Editor) with script-enforced criteria: anyone can rebuild it and repeat the measurement. Ten tools measured, the three reference ones charted; Apple M5 Pro, 8 September 2026.

### Where containers are missing

On the Govdocs1 public documents (980 files, 1.1% of the mass in containers) TSR is second: 48.598% against zpaq's 47.51%. The advantage follows the share of containers in the volume, and is declared as such.

### Films and footage

On H.264 films the 2–3.7% is measured on personal footage, hence declared without a ranking. On a video library TSR is worth having for instant access and integrity.

### In the fast profile

The fast profile keeps the two light filters (executables and audio) and leaves recompression to the slower profiles: with the fast codec they yield 0.3 to 1.4 points and cost the time of reassembling them. Decided by a measurement on 21 September 2026.

# 04 The archive is a function of the data

The same input, with the same profile, produces the same bytes on different architectures and operating systems. Every block and every file carries its fingerprint. The format is frozen, with a specification that runs.

## Determinism on three architectures

| TEST | OUTCOME |
|---|---|
| **Whole archives, macOS and Linux, with permissions and dates equalised** | SHA-256 fingerprints identical 4 out of 4 (enwik8 and Silesia, maximum and balanced profiles), also on 11,356 PNGs |
| **Windows** | Compressed content byte-identical on eight archives; the index reflects the permissions that system reports for the files |
| **Per-core fingerprints** | Identical on Apple Silicon, x86-64 and server-class ARM64; with vector instructions on and off, 24 out of 24 |
| **Usage scenarios** | Content identical on the three architectures for eight personas out of nine; the ninth differs only on Windows, for 575 names containing “?” rewritten by the filesystem |
| **Graphics accelerator** | Identical archives with the lane on and off, on Metal and on Vulkan |

The property is possible because the codec works only with integer arithmetic, its streams are frozen by tests that compare byte for byte, and every future evolution takes a new identifier alongside the old ones. This is the precise formulation, and the one that holds up to a check: anyone repeats it with one command on their own machine.

## Integrity per block and per file

Every block carries the BLAKE3 fingerprint of its original content, and every file its own: verification says which file is damaged and in which block; the index has its own fingerprint. Deep verification also rebuilds the dismantled files and compares them with the per-file fingerprint. The same verification exists in a form that goes slowly, stops and resumes where it was (`tsr scrub`, with bandwidth and CPU caps measured from outside) and is scheduled once a week.

**What it is for**

Long-term preservation, chain of custody, verification across sites, upstream deduplication, reproducible builds: all demands that require an archive that is a pure function of the data.

**Format 1.0**

Frozen in September 2026. Every archive written by any version is read by all later ones, forever: for every version of the format there is an archive kept as a vector, which the test suite opens on every run and which is kept as it is. The specification, in English, is verified by a reader written from it alone; the vectors are forty-nine (fourteen historical, five of format 1.0, thirty negative) and three independent readers read them.

**Refusal, stated**

A reader that meets what it does not know says so, naming the thing and the version it reads (“a newer version is needed”): a clean error in place of wrong bytes. Every refusal condition of the specification has its negative vector.

## Encryption as standard

A password-protected archive is encrypted in full with XChaCha20-Poly1305, index included: file names, sizes, dates and permissions stay secret. The key is derived from the password with Argon2id, with the parameters written in the archive, so they can be raised tomorrow without breaking yesterday. The project declares it has written zero cryptography of its own: public, well-tested primitives used as intended. Encryption is per block, so random access, streaming and the mounted volume work identically on encrypted archives; the measured worst-case cost is two hundredths of a second and 4.2 KB on half a gigabyte. A wrong password and a tampered archive produce two distinct errors.

## Repair without the password

On request the archive carries 1 to 30% of Reed–Solomon parity: damage within the quota, scattered or contiguous, is repaired in full and the archive comes back byte-identical, in a new file, leaving the original untouched. The parity covers the physical bytes, so an encrypted archive is repaired without knowing the password: whoever keeps the medium can restore it even without the right to read it. Reference measurement: 34 MB with 10% parity, 100 KiB contiguous plus twenty scattered bytes of damage, 46 shards out of 8,732 rebuilt in a tenth of a second.

## Durability against power loss

Extending an archive writes the new blocks after the old index, then a new index and footer at the end, synced to disk; the old footer stays as an anchor until the new one is safe. The program was killed 2,752 times at random instants during creation, extension and compaction, plus byte-by-byte truncation: zero archives lost, zero crashes. On error or cancellation the disk stays as it was before the job.

> **The rule that holds the three ideas together**
>
> Every byte of the archive can be derived from the data, the profile and the published format; every byte can be verified with its fingerprint; every unknown byte is refused, and said so. From these three properties follow reading across different machines, verification years later and repair without trusting anyone.

**Opening files of unknown origin**

The program is written in Rust, which rules out by construction the classes of defects most exploited in parsers. Every size field has a cap: a hostile 299-byte archive with an inflated size is rejected in a tenth of a second. Coverage-guided fuzzing runs on **18 targets** under a memory-error detector: 11.77 million recorded executions, zero defects; on every push a blocking local check repeats it for ten seconds per target.

**Network and telemetry**

The version for companies works locally and answers only whoever queries it. The public beta is a separate build: it checks for signed updates and sends crashes and metrics only with consent. The streaming service listens by default on the local interface only; exposing it to a network requires an access secret.

**What can be seen from outside**

Of an encrypted archive the total size, the index size and the number of extensions remain visible. Rolling back the whole file remains possible to whoever holds the medium.

# 05 The data stays usable while it is compressed

The independent blocks that make the strong family usable are the same ones that open the archive halfway: only what is touched is decompressed.

## Random access by construction

A read at any point decompresses only the blocks involved, with a cache sized on the budget. A 2.3 GB H.264 film inside an archive plays and seeks with 252 MiB of memory and a 1.6-second jump; in VLC and QuickTime the frame matches the timecode at every jump, and the file stays compressed the whole time.

### The declared price of random access

TSR compresses within the blocks of a file and never across files: on trees of source code and e-mail 7-Zip in solid mode is ahead by 2–7 points. It is the price of direct access to files, and it is written in the perimeter.

## Three doors onto the same archive

- **A local network volume.** `tsr serve` exposes the archive over HTTP with range requests and as a read-only WebDAV volume: it mounts in the Finder or in File Explorer without kernel extensions, and every application opens the files normally.
- **A folder, on Linux.** `tsr mount` mounts the archive in the filesystem: grep, ffmpeg, an editor or a build open the files without knowing they are compressed. Measured: 189 microseconds per file, against 1,198 over the HTTP path and 18 for extracted files.
- **The graphical interface.** Preview of text and images, a tree with search, “where the space goes”, selective extraction, verification and repair, without extracting the archive.

### A service that can be exposed

Outside the local interface the access secret is mandatory and compared in constant time; connection and bandwidth caps; write methods refused; transport encryption left to a reverse proxy in front, with configurations ready. A daemon queues jobs behind a local API, and a system timer runs the weekly verification.

## The second backup costs what has changed

A file identical to one already present (in the batch, or in the archive being extended even months later) becomes a reference; a block identical to one already written is referenced instead of rewritten; with the backup profile blocks are cut where the content says, so an insertion realigns within a megabyte. The three things live in the same container.

### Against a dedicated backup tool

In the benchmark with insertions the backup profile saves as much as restic (97 to 111% of its savings), with the single copy within a tenth of a point.

BACKUPS AND VERSIONS: SCENARIOS FROM A DETERMINISTIC SCRIPT, REPRODUCIBLE BY THIRD PARTIES; JULY 2026

| SCENARIO | TSR | 7-ZIP AT MAXIMUM |
|---|---|---|
| **408 MB SQLite database, two versions after twelve changes** | 100.0 MB (0.90 GB of memory) | 109.8 MB (8.30 GB) |
| **Log growing at the tail** | 29.2 MB in 26.2 s | 35.3 MB in 136.8 s |
| **Disk image modified by mounting it** | 247.1 MB | 427.6 MB |
| **389.5 MB folder plus its full backup** | 36.6% (the second costs 2,190 bytes) | 74.8% |

# 06 Evidence, and the method that governs it

A benchmark is worth its conditions. The rules below govern every figure in this document, and have applied for months to every measurement of the project.

## Four rules

- **Sizes are deterministic.** The size of an archive depends only on the input and the profile: one run is enough, and repeated elsewhere it gives the same bytes.
- **Times require discipline.** Thermal drift is worth up to 20%: where a time supports a claim the measurement is an interleaved A/B comparison over three runs, and the minimum counts; the times in the overview tables are single-run.
- **Whoever builds the corpus decides the result.** Four criteria enforced by a script: no file over ten per cent of the mass; no material already used in the measurement corpora or in training the codec; public sources with verified fingerprints; no selection by type inside a source. The lesson was paid for in-house: an internal corpus used until early August 2026 showed twenty-two points of lead, and on corpora rebuilt this way the lead is about three. Corpus withdrawn, measurement redone, episode written up.
- **Rivals at their best, and negative verdicts published.** Ten tools at their documented maximum levels, including the strongest pure-ratio compressors (zpaq, bsc, PPMd); rankings are declared against all of them, and rejected ideas have a page in the benchmark dossier.

### The four machines

Apple M5 Pro, 15 cores (macOS, reference for times); NVIDIA GB10, twenty ARM cores (Linux arm64); AMD Ryzen 9 5900X, 12 cores, on Windows 10 and on WSL2 Ubuntu 22.04 (x86-64).

### The ten tools

7-Zip -mx9 and the PPMd variant (o32 mem2g), zpaq -m5, xz -9e, zstd --ultra -22, brotli -q11, bzip2 -9, bsc -b100 -e2. On the usage scenarios: zip -9, 7-Zip -mx9 and -mx5, zstd -19 --long=27, xz -9.

### The sentence the numbers support

First on compressed containers; 7-Zip, xz, zstd, brotli and bzip2 beaten on text, mixed data and documents. On pure text zpaq, bsc and PPMd are ahead: it is written here because it is the first thing a reviewer measures.

## Results on the corpora

| CORPUS | TSR MAXIMUM | BEST RIVAL | RANKING |
|---|---|---|---|
| **office-documents (1,322 files, 199.5 MB)** | **47.888%** | **zpaq 51.98%** | **first of ten** |
| **public-documents (Govdocs1, 980 files)** | 48.598% | zpaq 47.51% | second |
| **Silesia (211.9 MB)** | 20.124% | zpaq 18.81% | second |
| **enwik8 (100 MB of text)** | 21.975% | zpaq 19.63% | fourth, behind zpaq, bsc and PPMd |
| **png-graphics (11,356 optimised PNGs)** | 53.373% | zpaq 52.84% | third |
| **sources (13,663 small files, on tar)** | 14.311% | — | 16.555% on the sum of the files |
| **JPEG gallery (5 photographs)** | 81.840% | generic ≈ 100% | −18.2%; a specialist ≈ −22% |

Maximum profile, 8 September 2026, Apple M5 Pro; rivals at maximum levels on the same machine; denominator the corpus tar. 7-Zip, xz, zstd, brotli and bzip2 beaten on text, mixed data and documents.

## Nine simulated usage scenarios

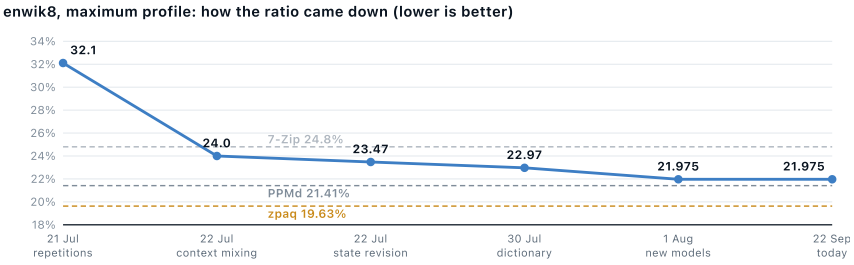
Nine folders that look like a person's, 3 to 12 GB each, built from public sources with a per-file licence (44.9 GiB and 195,753 files in all) and measured on four machines. Here the Mac with the GPU off: TSR with the median of three runs, rivals with one run at their best. Ratio as a percentage of the raw bytes, lower is better; the best per row in bold.

NINE USAGE SCENARIOS: RATIO AS A PERCENTAGE OF RAW BYTES, APPLE M5 PRO, 12–14 SEPTEMBER 2026

| PERSONA | TSR MAXIMUM | ZIP -9 | 7-ZIP -MX9 | ZSTD -19 --LONG | XZ -9 |
|---|---|---|---|---|---|
| **Doctor** | **34.71** | 67.32 | 46.62 | 45.47 | 57.10 |
| **Family** | **83.76** | 98.02 | 97.41 | 97.57 | 97.63 |
| **Creative** | **69.51** | 84.44 | 79.61 | 80.75 | 79.92 |
| **Student** | **69.39** | 80.35 | 74.71 | 75.07 | 75.19 |
| **Legal** | **61.91** | 69.98 | 62.07 | 62.76 | 62.50 |
| **Videomaker** | **97.18** | 99.45 | 98.68 | 98.81 | 98.77 |
| **Office** | 52.73 | 61.11 | **51.02** | 51.52 | 51.41 |
| **IT technician** | 54.51 | 57.15 | **51.57** | 51.76 | 54.51 |
| **Developer** | 43.84 | 52.71 | **36.80** | 38.47 | 39.11 |

First where the filters work (medical images, phone JPEGs, PNG, PDF) with margins of 5 to 13.7 points over the best of the others; behind 7-Zip in solid mode where files are many and small (developer, office), because it compresses within the blocks of a file and never across files. The balanced profile always produces a smaller archive than zip -9; the fast one always finishes before 7-Zip at level 5, 3 to 38 times faster.

## Where we started from




The codec was born on 21 July 2026 and lost to gzip. Since 1 August the enwik8 figure has stood still: the later work went into already-compressed formats, speed and reliability. Every step has a report with its measurement, and the rejected steps sit beside the accepted ones.

**Measured reliability**

Suite of **632 cases** in 49 groups; green on the four platforms at the same commit, on 27 September 2026.

49 conformance vectors read by three independent readers; 13,649 files of 1,540 types from four systems: identical extraction, zero crashes, zero non-determinism.

2,752 shutdowns mid-write without an archive lost; 11.77 million recorded fuzzing executions, zero defects.

Scale: one million and five million files within the memory budget (opening five million files costs 19 MB); a 111 GB archive with 370 thousand files in a single run, with deep verification green and a thousand random files identical.

# 07 Declared perimeter

This page exists because whoever lists only their own advantages forces the reader to discover the rest alone. Here is the rest, with the numbers.

| ASPECT | STATUS DECLARED AT THE DATE OF THIS EDITION |
|---|---|
| **Pure text** | On enwik8 zpaq, bsc and PPMd are ahead (19.63, 20.80 and 21.41% against 21.975); on Silesia and Govdocs1 only zpaq stays ahead. TSR is first where the filters work, and that is where the product works. |
| **Already-compressed media** | MP3, AAC and untreated photographs: 1–2% recovered. H.264 films: 2–3.7%, measured on personal files that cannot be redistributed. On a video library TSR is worth having for access and integrity. |
| **Many small files** | Blocks within a file, never across files: on trees of source code and e-mail 7-Zip in solid mode is ahead by 2–7 points. It is the price of random access. |
| **Cost of the maximum ratio** | The maximum profile keeps the machine busy for tens of seconds per hundred megabytes per core; fluidity comes from parallelism, and the user chooses the time. |
| **Speed on the target hardware** | Four machines declared. On the Ryzen 9 5900X the maximum profile does 4.0 MiB/s aggregate: 50 MB/s on 16–24-core CPUs stays out of reach with the current codec, and speed is declared per profile. |
| **Memory** | Without an explicit ceiling the budget is an estimate, which trees full of photos and videos can exceed; a requested ceiling is a contract that is honoured (1.79 GB under a 2 GB ceiling), paying up to three times the time. The command line says which of the two is in use. |
| **Estimates** | They err on the safe side. The fixed-codec profiles stay cautious by about three times on time, because parallel efficiency is calibrated on the development machine. |
| **JPEG and PNG** | JPEG: 18.2% against about 22% for a specialist. PNG: only classic-compression files; modern optimised graphics travel as they are, and on 11,356 files of that kind TSR is third. |
| **Deduplication** | A scattered change in a large file rewrites every block touched: 13.8 MB changed at random in 1.3 GB cost 101 MB (a dedicated backup tool pays the same). |
| **Scale and endurance tests** | Done up to five million files and 111 GB in a single run. The twenty-four hours of continuous service have their acceptance criterion written and remain to be run. |
| **Certifications and attestations** | No certification; independent penetration testing and cryptographic review still to be commissioned. The artefacts that enable them (bill of materials, evidence, security policy) are ready. |
| **Code signing** | Ad hoc signature on macOS; the Apple and Windows certificates are awaited, and until then the systems warn at first launch. |
| **Release channel** | Version 0.10.2, “demonstration beta”: eight packages and two images tested, and separately a public beta; a pilot with a third party remains to be done. |

## What stays outside the format’s promise

- A new archive on an old program is refused, and said so: the promise is that the new program reads every old archive.
- The same bytes forever for the same input across different versions of the program: a new codec may win the election and change the bytes written. Guaranteed are reading and identical extracted content; where bytes must match (across architectures, with the same version) the measurement says so.
- Encryption leaves the archive size and the number of extensions visible, and rolling back the whole file remains possible to whoever holds the medium.
- Metadata arrives where the filesystem accepts it; extraction succeeds regardless.

# 08 Verifying for yourself, and adopting

Three checks within anyone's reach, without trusting this document; then the form in which the program arrives.

## Three checks

1. **The public corpora.** enwik8 (Large Text Compression Benchmark) and Silesia can be downloaded; the rivals run at their standard maximum levels (`7z -mx9`, `xz -9e`, `zstd --ultra -22`, `brotli -q11`, `bzip2 -9`, `zpaq -m5`). The size of the TSR archive is deterministic: one run is enough, and it matches the declared one byte for byte.
2. **The graphics accelerator's contract.** `tsr gpu-check` runs the search for repetitions on the machine's GPU and CPU and compares the results position by position.
3. **Your own data.** Thirty days of trial with the comparison scripts included: the version for companies runs inside their network, with no outbound communication, and at the end a table of your own numbers remains. Any archive is verified in full with `tsr t --profonda`.

## The form it arrives in

A graphical application for macOS, Windows and Linux (drag and drop, job queue, preview, selective extraction, verification and repair); a command-line tool with JSON output and nine exit codes, the same as those of the C library `libtsr` through which it integrates into other programs; eight packages (zip for macOS and Windows, tarballs, .deb and .rpm for x86-64 and ARM64) and two container images, one of them 2.92 MB with no operating system inside. Every package carries the binary's software bill of materials (SBOM), the security policy, the licence and the attributions; all dependencies are permissive, zero copyleft.

**On a server**

Streaming service with an access secret mandatory outside the local interface; a daemon with a job queue and a root that fences the paths; weekly verification with a system timer; hardened systemd units; a container image with only the static binary inside.

**Related documents**

TSR-COM-01 · Commercial dossier
TSR-TEC-02 · Technical dossier, under non-disclosure agreement
TSR-PUB-03 · Public sheet
TSR-LEG-04 · Contractual framework
TSR-BEN-05 · Benchmark dossier
TSR-2P-07 · On two pages
Every document also in Italian
`SECURITY.md` · the security policy, inside every package

**Contacts**

**Matteo Luigi Feroldi**
Via Carlo Cattaneo 18/1
20055 Vimodrone (MI), Italy
VAT no. IT14667380969

matteo@feroldi.cloud
+39 388 8630382

```
tsr c archive.tsr folder/ -t lento          # time level; -t 300 for a budget in seconds
tsr c archive.tsr folder/ --max --stima     # fixed codec, with the estimates before starting
tsr t archive.tsr --profonda                # verify, rebuilding the dismantled files too
tsr serve archive.tsr                       # HTTP with Range and WebDAV, local interface only
tsr mount archive.tsr /mnt/a                # Linux: the archive as a folder
tsr scrub archive.tsr --banda 20M --cpu 25  # the slow verification, resuming where it left off
```

A percentage point of ratio is gained once and pays back on every copy of the data, for its whole life.

# Glossary and references

The terms used in this document, in the sense in which TSR uses them, and the public sources of the measurements.

| | |
|---|---|
| **Ratio** | The archive size as a percentage of the original; lower is better. A “point” is a percentage point. |
| **Block** | The unit of compression, 2 to 16 MiB depending on the profile, always within a file; every block is compressed, verified and read on its own. |
| **Profile** | The set of choices of a compression: time-budgeted (five levels or a budget in seconds) or fixed-codec (fast, balanced, backup, maximum). |
| **Calibration** | The measurement of speed and ratio of every codec on the real files and the machine in use, before starting. |
| **Filter** | The reversible rewriting of an already-compressed file, proven on every file before writing. |
| **Context mixing** | The family of codecs that predicts every bit from several models and weighs their opinions. TSR’s strong lane. |
| **Election** | On wide budgets the codec tries several candidates on the same block and keeps the shortest. |
| **Incompressibility guard** | The sample that recognises already-compressed or random data before running the codecs. |
| **Determinism** | The same input and the same profile produce the same bytes, on different machines. |
| **Fingerprint** | The BLAKE3 hash of a block, a file or the index: it says whether the content is the original. |
| **Recovery record** | The Reed–Solomon parity at the end of the archive, which repairs damage within the chosen quota. |
| **Conformance vector** | A frozen archive with a declaration of what must come out when reading it; forty-nine in the project. |
| **Scrub** | The verification that goes slowly, stops and resumes where it was. |
| **DEFLATE container** | A file that contains streams compressed with the DEFLATE of 1993: ZIP, Office documents, JAR, gzip, PNG, PDF streams. |

## References


- **enwik8**: the first hundred million bytes of the English Wikipedia, from the Large Text Compression Benchmark, where the results of the strongest compressors are also published.
- **Silesia Corpus**: twelve files, 211.9 MB (text, binaries, databases, medical images), the industry’s reference mixed corpus.
- **Govdocs1**: real documents collected from public websites, made available by Digital Corpora; here thread 000 in full.
- **Rivals**: 7-Zip (also in the PPMd variant), xz, zstd, brotli, bzip2, zpaq, bsc; on the usage scenarios also zip. All at their documented maximum levels, with the parameters written alongside.
- **Cryptographic primitives**: XChaCha20-Poly1305 for authenticated encryption, Argon2id for key derivation, BLAKE3 for fingerprints, Reed–Solomon over GF($2^{16}$) for parity.
- **Measurement dates**: ratios of 8 September 2026; usage scenarios of 12–14 September (the fast-profile column re-measured on the 22nd); scale tests of 18–20 September; packages and parallel rebuild of 22 September, test suite of 23 September 2026.


### How to read the percentages

Every ratio is the archive size divided by the original size: 47.888% means the archive takes less than half. Ratios are compared by difference (the “points”); an 18.2% gain on JPEGs is the reduction in size compared with the starting file.

### Where the details are

The benchmark dossier carries corpora, commands and machines for every figure; the technical dossier, delivered under a non-disclosure agreement, the inner workings of the codec and the format; the public sheet gives the picture in eight pages. At this edition the companion documents are in Italian.

# TSR

Time Space Reducer — compress further where others stop, in a time you choose, with an archive that is a function of the data.


OWNER

Matteo Luigi Feroldi
Via Carlo Cattaneo 18/1
20055 Vimodrone (MI), Italy
VAT no. IT14667380969

CONTACTS

matteo@feroldi.cloud
+39 388 8630382


REPRODUCTION OF THE PUBLIC MEASUREMENTS

enwik8 — Large Text Compression Benchmark
Silesia — Silesia Corpus
Rivals at standard maximum levels