The D'Agapeyeff Cryptogram (1939)
Anatomy, Reconstruction of a Solver, and Statistical Assessment

Ruben Gariazzo

student at the École Polytechnique Féminine

ABSTRACT

In 1939, cartographer and amateur cryptologist Alexander D'Agapeyeff published, at the end of his manual *Codes and Ciphers*, a 196-symbol cryptogram presented as a challenge for his readers. Eighty-seven years later, it remains unsolved, and its author reportedly admitted having forgotten the method used to construct it. This report documents an attempt at computational cryptanalysis carried out through successive iterations: the methodological difficulties encountered, their diagnosis and correction, and the results obtained once these corrections were applied.

The first difficulty concerned not the cipher, but the solver meant to break it. An early version deemed a result promising beyond a score threshold fixed in advance and never empirically verified. After three days of continuous computation, no result had crossed it. An authentic English text, scored *a posteriori* under the same model, did not reach that threshold either, which turned out to be structurally unreachable: a consequence of a language model[1] trained on a corpus of only 8,000 characters, insufficient to statistically distinguish real English from gibberish. This diagnosis motivated a complete reconstruction of the solver. The principle adopted for the rest of the project is to empirically verify any success criterion before using it, rather than assuming it.

The reconstructed solver jointly tests four cipher families (substitution, simple transposition, Fleissner grille[2], and double transposition) over nine competing hypotheses for transcribing the printed text and three candidate languages (English, French, transliterated Hebrew), by means of simulated annealing[3] whose computational effort is distributed by an adaptive bandit[4] across the possible structural combinations. A high score is not sufficient to conclude, however. Once hundreds of thousands of combinations are tested independently, the best observed score drifts upward mechanically even on pure noise, through a simple sample-size effect. Null-baseline calibration[5], the project's central methodology, corrects for this bias. Applied to the Hebrew hypothesis, it showed that a score initially deemed promising was merely an artifact of the smaller transliterated alphabet used, once compared to its own null baseline

[1] Quadgram language model: a measure of a text's plausibility based on the frequency, in a large reference corpus, of each of its four-consecutive-letter windows (a quadgram). The wider the window, the statistically harder it becomes for a nonsensical sequence to obtain a high score by pure chance.

[2] Fleissner grille: a perforated template rotated over a text to reorder its letters by transposition, one of the oldest historical devices for scrambling a message without altering its letters.

[3] Simulated annealing: an optimization method that explores the space of possible keys, occasionally accepting, with a probability decreasing over the course of the search, a momentarily worse solution, so as to avoid getting stuck in a local optimum.

[4] Adaptive bandit: an algorithm that progressively concentrates sampling on the most promising options among several competing candidates, without ever fully excluding any of them.

[5] Null-baseline calibration: comparing the best result obtained not to a fixed threshold, but to the best result the same pipeline achieves, at the same trial volume, on a randomly shuffled version of the cryptogram, which is therefore by construction devoid of any signal.

rather than to an isolated reference.

This same verification requirement was applied to the search tools themselves. A search acceleration effort (full compilation of the simulated-annealing loop, guaranteed systematic coverage of the search space rather than probabilistic) revealed an unforeseen bottleneck in result logging. Its first candidate fix, measured before deployment rather than assumed safe, turned out to be almost as costly as no fix at all and had to be replaced. A second detection criterion, based on the recognizable-word coverage of the decoded text, was likewise delayed by a reference dictionary contaminated with technical jargon, caught on pure noise before any production deployment.

Across the four cipher methods tested, covered exhaustively and with a guarantee on two independent passes, no signal stands out statistically from noise on any of the three metrics measured: the language-model score, recognizable-word coverage, and the length of the longest coherent fragment identified. The null baseline eventually surpassed the best score obtained on the real cryptogram. The longest and most grammatically coherent run of English words found across the entire project, "held if we can win", in fact comes from a randomly shuffled sequence rather than from the cryptogram itself, a direct demonstration that such coherence can arise by pure chance at this search volume. A final analysis, focused on the structural clustering of top scores rather than on any single result, revealed an asymmetry between transcription hypotheses; on verification, this turned out to be produced by the adaptive sampling mechanism itself rather than by the underlying text. This line of analysis was closed for lack of sufficient data.

This exhaustive coverage of the initial space, with no signal detected, redirected effort toward a cipher family absent from the modeled space until then, the Four-square cipher, which substitutes pairs of letters by means of four Polybius grilles rather than single letters. Its mechanics were validated independently over thousands of reciprocal encryption and decryption trials. Validating the search itself revealed a further difficulty. Simulated annealing, effective on the four previous methods, systematically failed to recover a known key with an identical compute budget, on a markedly rougher score landscape. A dedicated tuning effort, based on multiple independent restarts rather than simply a larger compute budget, achieved a recovery reliability of 83% measured over independent trials. This search is now active in production, on a space still too sparsely covered to draw any conclusion.

Nomenclature

N	Length of the cryptogram in symbols ($N = 196$)
w, h	Width and height of a transposition grid ($w \times h = N$)
IC	Index of coincidence of a symbol sequence
$Q(k)$	Quadgram log-likelihood score of a text decoded with key k
f_{abcd}	Smoothed frequency of quadgram *abcd* in the reference corpus
$w_i,\ b_i,\ n_i$	Sampling weight, best score, number of trials of arm i (bandit)
λ_i	Expected average number of trials on stream i (Poisson approximation)
$\mathscr{S}$	Self-test reference score (single-trial calibration)

1. Introduction

The D'Agapeyeff cryptogram differs from the ciphers usually studied in historical cryptanalysis in its status as a published challenge rather than an intercepted message. Unlike a message known to have been exchanged for a specific purpose, this one was published voluntarily by its author, with no certainty that he himself retained the solution. Alexander D'Agapeyeff, a cartographer and Royal Air Force officer of Russian origin naturalized British, had it printed in 1939 at the end of the first edition of his manual *Codes and Ciphers*, in the form of 392 digits grouped in fives. It was removed from subsequent editions; its author reportedly later admitted no longer remembering the method used to construct it. This anecdote is treated here as a genuine working hypothesis rather than a mere biographical curiosity: a failure to recover a solution would constitute proof neither of the search's inadequacy nor of the absence of an underlying plaintext.

This initial uncertainty shapes the approach adopted in this report. The project treats cryptanalysis of the cryptogram as a statistical hypothesis-testing problem rather than a search for a key by intuition. This entails three principles applied systematically: explicitly modeling the space of cipher methods and transcription variants deemed plausible, exploring that space systematically and with guaranteed coverage rather than at random, and comparing any score obtained not to a threshold fixed a priori, but to the best result that a text devoid of any signal by construction would achieve at the same trial volume. This last point is a central requirement, not an incidental precaution. With hundreds of thousands and then millions of combinations tested independently, the best score observed on pure chance also drifts upward, and nothing distinguishes it from a real signal without this point of comparison. Every extension of the hypothesis space, whether an additional cipher method or a candidate language, is motivated by concrete evidence or a concrete clue, not by arbitrary exploration.

This document takes the form of a research log kept up to date over the course of the project, rather than that of a report written after the fact for a method fixed in advance. It documents the methodological dead ends encountered, their diagnosis and correction, on the same footing as the positive or negative results obtained once those corrections were in place. Section 4 presents the design flaw that motivated

a complete reconstruction of the initial solver. Section 5 details the resulting architecture. The sections that follow retrace, in their actual chronological order, each correction, each new statistical detection axis, and each verdict reached, up to the search's current state of progress.

2. Context and Anatomy of the D'Agapeyeff Cipher

2.1 A Cartographer, a Book, an Enigma

Alexander D'Agapeyeff (1902-1955), a cartographer and amateur cryptologist of Russian origin naturalized British, published in 1939 with Oxford University Press *Codes and Ciphers*, an introductory cryptography book commissioned by his publisher as war approached. The first edition ends with a cryptogram meant to test the reader. It was removed from later editions, D'Agapeyeff himself having admitted he no longer remembered the method used to construct it (an anecdote that, as we shall see in section 15, may be more than a mere quip).

2.2 The Full Text

The cryptogram is printed as 392 digits, grouped in fives for telegraphic legibility (a standard typographic convention of the era):

Exhibit 1: The Cryptogram as Printed (1939)

```
75628 28591 62916 48164 91748 58464 74748 28483 81638 18174
74826 26475 83828 49175 74658 37575 75936 36565 81638 17585
75756 46282 92857 46382 75748 38165 81848 56485 64858 56382
72628 36281 81728 16463 75828 16483 63828 58163 63630 47481
91918 46385 84656 48565 62946 26285 91859 17491 72756 46575
71658 36264 74818 28462 82649 18193 65626 48484 91838 57491
81657 27483 83858 28364 62726 26562 83759 27263 82827 27283
82858 47582 81837 28462 82837 58164 75748 58162 92000
```

2.3 An Underlying Polybius Square

The structure of the digits is consistent with a Polybius square[1] 5×5 read in pairs of digits: the first digit of each pair, taken from $\{6,7,8,9,0\}$, encodes the row; the second, taken from $\{1,2,3,4,5\}$, encodes the column, a standard interwar telegraphic convention in which "0" cyclically continues the series after "9". Figure 1 illustrates this reading on the very first pair of the text.

[1] Polybius square: a 5×5 grid assigning each letter of the alphabet a coordinate (row, column); the basic building block of many historical ciphers (ADFGVX, Bifid, Nihilist...).

	column				
	1	2	3	4	5
6	A	B	C	D	E
7	F	G	H	I/J	K
8	L	M	N	O	P
9	Q	R	S	T	U
0	V	W	X	Y	Z

Illustrative grille (arbitrary alphabetical fill, for pedagogical purposes). The actual substitution key (which letter occupies which cell) is precisely what the solver searches for; it is never assumed.

Example: the pair "75" → row 7, column 5 → cell (7,5) → symbol 9 (indexed 0-24) → letter **K** *in this example grille only*.

Figure 1: *Decoding a digit pair into a Polybius grille coordinate.* The I/J merge is the standard convention for fitting 26 letters into a 25-cell grid.

2.4 Why the End of the Text Was Truncated

The last printed group, "92000", raises a direct question: do the cryptogram's 196 pairs stop at "92", or does the text contain 197.5 additional pairs hidden in "000"? Three converging arguments, illustrated in Figure 2, settle in favor of simple typographic padding:

1. **Group alignment.** 78 complete groups of 5 digits precede the last one = 390 digits = 195 exact pairs. The 79$^{\text{th}}$ group, "92000", supplies with its first two digits ("92") exactly the 196$^{\text{th}}$ and final pair. The three remaining digits structurally fall outside any pair.
2. **Parity.** 196 pairs = 392 digits, an even number. Including "000" would bring the total to 395 digits: an odd number, incompatible with any pairwise split whatsoever.
3. **Row/column typing.** Across all 196 pairs retained, no "0" ever appears in column position (which only accepts $\{1,2,3,4,5\}$). Were one to try reading "000" as additional pairs, the pair "00" would immediately violate this rule, never broken elsewhere in the text.

Figure 2: *Breakdown of the last printed group "92000".* The first two digits complete the 196$^{\text{th}}$ pair; the last three belong to no pair and are excluded from the numeric cryptogram analyzed.

The `CIPHER_RAW` used by the solver therefore stops at 392 digits. It would have been just as easy to err in the other direction: blindly including "000" without checking its consistency with the rest of the text. It is precisely this kind of transcription decision, made without supporting evidence, that this project strives to document and explicitly justify rather than assume.

3. Working Hypotheses

Unlike a closed mathematical proof, this project does not rest on four fixed hypotheses but on a set of *competing* hypotheses, grouped into four families (Figure 3) and arbitrated empirically by the adaptive

bandit (section 5.6) rather than chosen *a priori*.

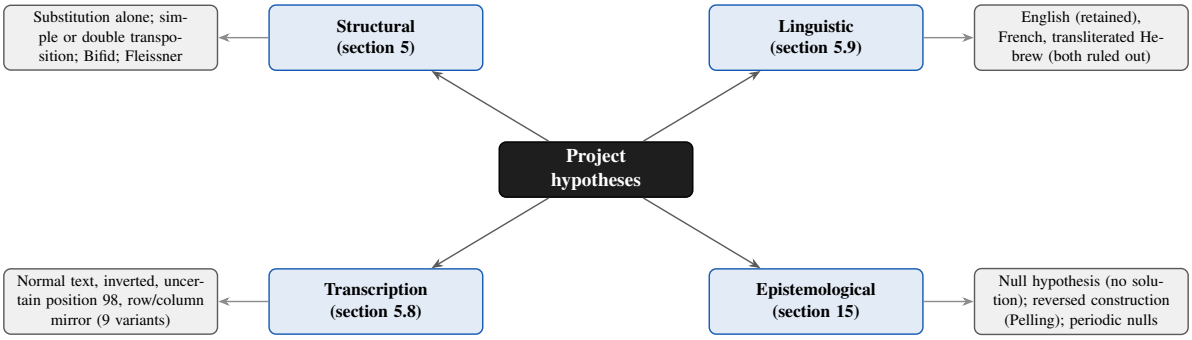

Figure 3: *Taxonomy of the hypotheses tested.* Each branch corresponds to an independent research dimension, combined with the others by the adaptive bandit; none is eliminated *a priori* before being confronted with the data.

Two hypotheses deserve to be highlighted before the rest of the report, as they condition the interpretation of any result obtained:

The null hypothesis is taken seriously. Nothing guarantees that the cryptogram conceals a recoverable text. A construction error by the author, a hypothesis reinforced by the forgetting anecdote and by Pelling's discovery (section 15), remains a possible outcome and is explicitly tested statistically (section 5.9), rather than rejected by default as version 1 of the solver implicitly did.

English is favored, not imposed. D'Agapeyeff, fully integrated into British professional life (cartographer, RAF wing commander), wrote a book in English for an English-speaking audience for whom the cryptogram is the final test. A plaintext in another language would make this test unverifiable for most of the intended readers, a functional argument that adds to the biographical one. This hypothesis nonetheless remains tested against two alternatives (French, Hebrew) rather than accepted without verification.

4. The Project's First Solver and Its Achilles' Heel

The first version of the solver (v1) combined a **trigram** language model (3-letter windows) built on a single embedded corpus of about 8,000 characters (the text of the American Declaration of Independence), a single grid dimension (14×14) explored via 4 fixed reading routes, a hard filter on the index of coincidence, and above all a **success threshold fixed *a priori*** at -500 or -400, never empirically verified (Figure 4).

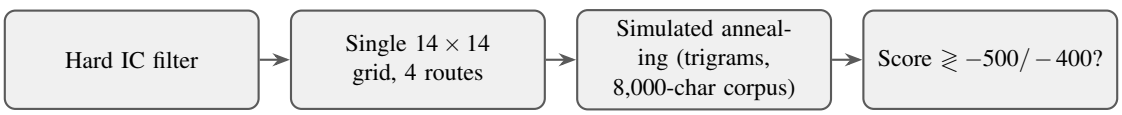

Figure 4: *v1 pipeline: a linear chain, no calibration step.*

After three days of continuous execution, no result exceeded this threshold. The initial reasoning (reason-

able in appearance) concluded that the plaintext was probably not standard English, or that D'Agapeyeff had made a mistake constructing his own cipher. This conclusion proved premature.

4.1 The Diagnosis of a Threshold That Never Existed

To check the threshold's validity, an authentic excerpt of *Pride and Prejudice* was scored with v1's *exact* trigram model, under three regimes: identity key (real text), the best result actually found by v1 (gibberish from the Fleissner grille), and pure random noise.

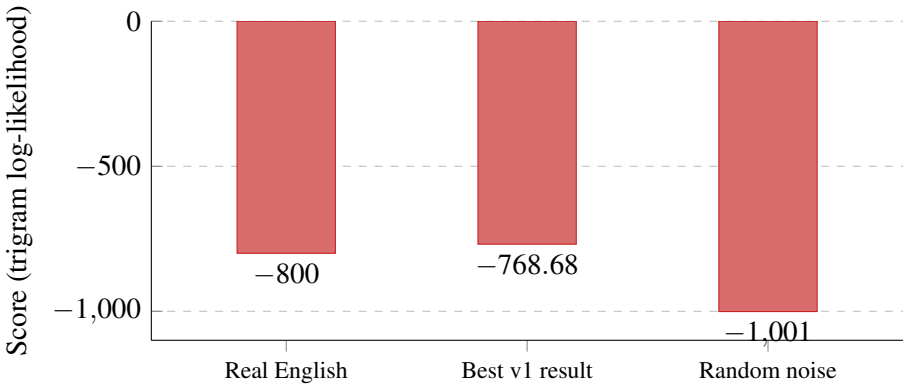

Figure 5: *Retroactive calibration of the v1 trigram model.* The best result v1 ever found (-768.68, gibberish) scores *better* than an authentic English text scored under its own model (-800): the $-500/-400$ threshold was therefore never reachable, independently of any question about the cryptogram.

The explanation lies in corpus size: over only 8,000 characters, most of the $26^3 = 17,576$ possible trigrams never appear, and the model then falls back on near-uniform smoothing that no longer discriminates real English from gibberish. v1's "no English after 3 days" conclusion thus reflected only the limits of its own language model, not a property of the cryptogram. This finding is the turning point of the entire project: it led to the complete abandonment of the v1 architecture in favor of a reconstruction designed to be verifiable at every step.

5. A Solver Rebuilt to Be Statistically Defensible

5.1 A More Discriminating Language Model

The reference corpus is expanded to more than 2.5 million characters (public-domain novels, Project Gutenberg), and the model extended to **quadgrams** (4-letter windows, $26^4 = 456,976$ categories versus $17,576$ for trigrams). The score of a text decoded with key k is expressed as:

$$Q(k) = \sum_{i=1}^{N-3} \log_{10}\left(f_{c_i c_{i+1} c_{i+2} c_{i+3}}\right)$$

where c_i is the i-th letter of the decoded text and f_{abcd} the smoothed frequency (Laplace, $+0.01$) of quadgram *abcd*. The wider the window, the statistically harder it becomes for a nonsensical permutation to obtain a plausible score by pure chance, *provided* the corpus is large enough to populate these categories, hence the simultaneous move to 2.5 million characters.

5.2 A Self-Calibrated Measure Replaces the Assumed Threshold

Before each search, the pipeline encrypts a known authentic text excerpt itself using its own mechanics, then attempts to recover it by simulated annealing. The score obtained $\mathscr{S}$ serves as an *empirical* high reference, specific to the corpus and language actually in use, permanently replacing v1's arbitrary $-500/-400$ threshold.

5.3 Four Cipher Families, Explored Jointly

At each iteration, four methods are tested on the same symbol stream (Figure 6):

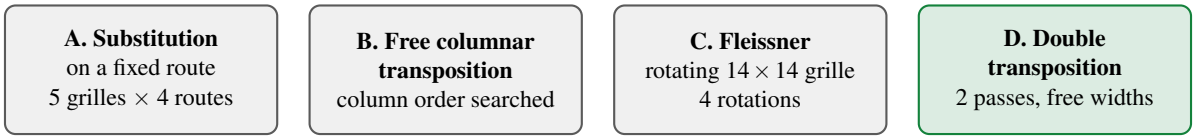

Figure 6: *The four cipher methods searched jointly.* Method D (in green), the most recent addition, is motivated by Pelling's discovery detailed in section 15.

Method D deserves elaboration: a single columnar transposition can only reorder whole blocks of columns of a fixed width w. Two successive passes, of potentially different widths w_1 and w_2, achieve individual symbol rearrangements that no single pass can reproduce (Figure 7): a blind spot in the original architecture, filled after Pelling's discovery.

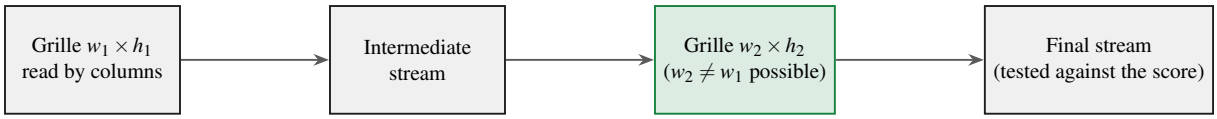

Figure 7: *Mechanics of the double transposition.* Passing through a second grille of different width rearranges symbols individually, not just by column blocks, a search space out of reach for a simple transposition.

5.4 Grids and Reading Routes Shared by Methods A, B, and D

Methods A, B, and D all draw on the same underlying tool before any substitution key is considered: arranging the 196 symbols of a stream into a rectangular grid of $w \times h$ cells, with $w \times h = 196$, and reading them back out along a chosen path. Five grid shapes are tested, the divisor pairs of $196 = 2^2 \times 7^2$ close enough to square to be practically distinct from one another: 14×14, 7×28, 28×7, 4×49, 49×4. The near-linear extremes (1×196, 2×98, and their transposes) are excluded: on such a thin grid,

most reading routes below coincide or nearly so, adding search cost without adding a genuinely distinct hypothesis.

Four canonical reading routes are defined for a given grid shape (Figure 8). Indexing a cell at row r and column c ($0 \le r < h$, $0 \le c < w$) by $\mathrm{idx}(r,c) = r \cdot w + c$, a route is a permutation of the 196 cell indices:

$$\begin{aligned}
\textbf{Rows}: &\ \mathrm{idx}(r,c),\ \text{for } r = 0,\dots,h-1 \text{ (outer)},\ c = 0,\dots,w-1 \text{ (inner)} \\
\textbf{Columns}: &\ \mathrm{idx}(r,c),\ \text{for } c = 0,\dots,w-1 \text{ (outer)},\ r = 0,\dots,h-1 \text{ (inner)} \\
\textbf{Boustro. rows}: &\ \mathrm{idx}(r,c) \text{ if } r \text{ even}, \quad \mathrm{idx}(r, w-1-c) \text{ if } r \text{ odd} \\
\textbf{Boustro. cols}: &\ \mathrm{idx}(r,c) \text{ if } c \text{ even}, \quad \mathrm{idx}(h-1-r, c) \text{ if } c \text{ odd}
\end{aligned}$$

The two boustrophedon routes alternate reading direction every row or every column, after the ancient ox-plowing pattern that gives the technique its name, in case the grid was filled and read in alternating directions rather than uniformly.

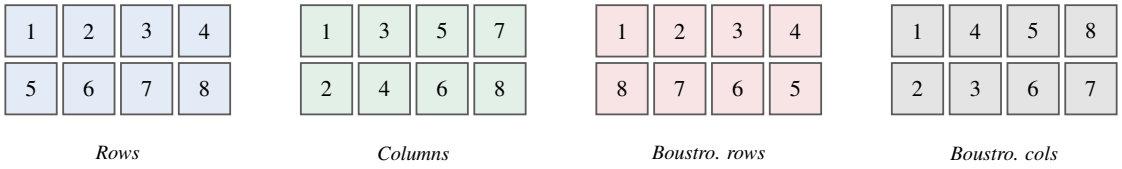

Figure 8: *The four reading routes on a* 4×2 *toy grid.* Each cell shows the rank (1st to 8th) at which it is read out, not the symbol it contains: the physical position is fixed, only the extraction order changes. At the real 196-symbol scale, five grid shapes are tested (main text), not just this 4×2 illustration.

For method A, the grid shape and route are treated as one discrete choice among $5 \times 4 = 20$, an arm of the adaptive bandit (section 5.6); once drawn, the 196 symbols are reindexed once according to that choice, and only the substitution key is then searched by simulated annealing (section 5.7) against this fixed reindexing. Method B instead fixes the route to a column-by-column reading but searches the *column order itself* as a free permutation of $\{0,\dots,w-1\}$, jointly with the substitution key, rather than restricting it to one of the four canonical routes above; concretely, extraction reshapes the stream into an $h \times w$ array, permutes its columns by the searched order, and reads the result column by column. Method D chains two such column-order searches through two independently chosen grid shapes (Figure 7 above), the first pass's output feeding the second.

5.5 Method C: Searching a Turning-Grille Pattern

Method C searches a different structural space entirely, native to the Fleissner grille itself rather than to a grid/route pair. A 14×14 turning grille has 49 physical holes; as the template is rotated 90° three times in succession (four positions in total), each hole exposes a different one of 196 cells, so that all 196 cells are exposed exactly once across the four rotations, in 49 disjoint groups of four *orbit-mates*. Which of the four orbit-mates is the true, physically cut hole, for each of the 49 positions, is not known in advance and is exactly what is searched: a 49-entry vector, each entry in $\{0,1,2,3\}$, perturbed one position at a time

and annealed jointly with the substitution key (section 5.7), the same joint-search scheme as methods B and D. Reading proceeds rotation by rotation (0°, then 90°, 180°, 270°); at each rotation, the 49 cells currently exposed are read in the natural row-major order in which they appear on the page, exactly as someone physically reading through the perforated template would.

5.6 Adaptive Bandit

Each search dimension (stream category, grille dimension, grille pair) is a multi-armed bandit arm. The sampling weight of arm i combines its best observed score b_i and an exploration bonus decreasing with the number of trials already carried out n_i:

$$w_i = \max\left(\underbrace{\frac{b_i - b_{\min}}{b_{\max} - b_{\min}}}_{\text{exploitation}} + \underbrace{\frac{c}{\sqrt{n_i + 1}}}_{\text{exploration}},\ \varepsilon\right), \qquad c = 0.6,\ \varepsilon = 10^{-3}$$

No arm is ever fully excluded ($w_i \geq \varepsilon$): a search area deemed unpromising continues to be tested, with decreasing but never zero frequency, the same logic as the coincidence-index-weighted draw applied to super-encryption streams.

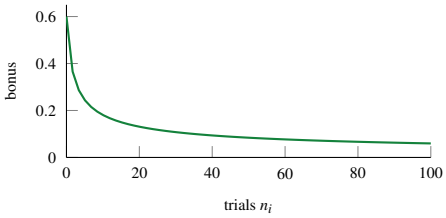

Figure 9: *The exploration bonus decreases but never reaches zero.*

5.7 Simulated Annealing: the Metropolis Criterion

Every combination tested, for every method, whatever its structural search space (a substitution key alone for A; a key and a column order for B; a key and a 49-entry grille pattern for C; a key and two column orders for D), is optimized the same way: by simulated annealing, a local search that occasionally accepts a worse candidate so as not to get trapped in the first local optimum it finds.

Let $S(\cdot)$ denote the quadgram score (section 5) of the text decoded under a given candidate state. At each iteration, a small random perturbation is proposed: for the substitution key, two of its 26 entries are swapped; for a structural state, one component is perturbed in place (a column swap for B and D, a rotation-orbit change for C). Writing $\Delta = S(\text{candidate}) - S(\text{current})$ and T for the current temperature, the candidate is accepted according to the Metropolis criterion:

$$P(\text{accept}) = \begin{cases} 1 & \text{if } \Delta > 0 \\ \exp(\Delta/T) & \text{if } \Delta \leq 0 \end{cases}$$

An improving move is always kept; a worsening move is kept with a probability that shrinks as Δ becomes more negative or as T drops, letting the search escape shallow local optima early on while behaving as

an increasingly strict hill-climb later. The temperature itself decreases geometrically at every iteration, $T \leftarrow \max(T_{\min},\ \alpha T)$, with $\alpha = 0.9995$ and a floor $T_{\min} = 0.05$ below which further cooling has no practical effect on the acceptance probability.

Two variants of this scheme are used (Figure 10). Method A perturbs only the substitution key, against a route already fixed by the bandit draw (section 5.4); it starts at $T_0 = 10$ and runs for 20,000 iterations. Methods B, C, and D perturb the key and the structural state jointly, alternating between the two at each iteration with equal probability; they start at $T_0 = 15$ and run for 20,000 to 30,000 iterations depending on the method. With this schedule, temperature reaches its floor after roughly 11,400 iterations for $T_0 = 15$, the same figure independently noted for the Four-square cipher's own tuning (section 14.5, which reuses this exact joint-search scheme), and somewhat earlier, around 10,600 iterations, for $T_0 = 10$.

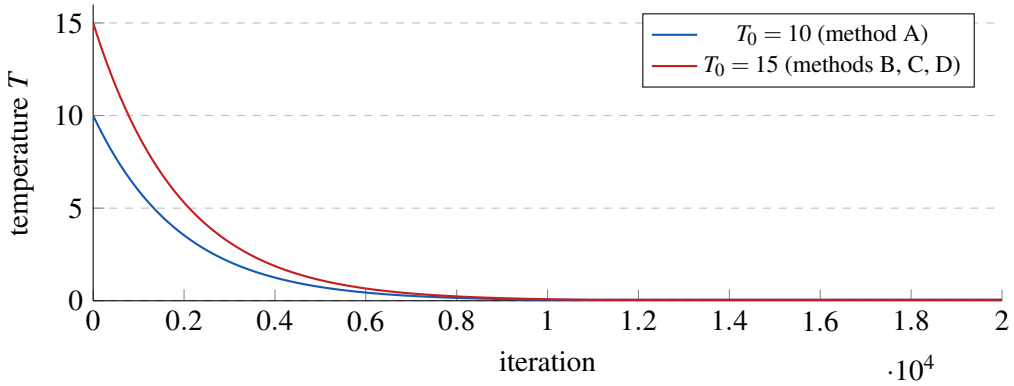

Figure 10: *Geometric cooling, $T \leftarrow \max(0.05,\ 0.9995\,T)$, for the two starting temperatures used.* Both curves flatten out at the floor well before the iteration budget (20,000 to 30,000) is exhausted, leaving the remaining iterations to behave as an increasingly strict hill-climb around the best state found so far.

This is a substantially more powerful search than matching ciphertext symbol frequency ranks directly to the theoretical English letter-frequency order (E, T, A, O, N, I, S, R, H, ...), the classical starting point of manual cryptanalysis. That single-shot approach implicitly assumes the observed frequency ranking on the ciphertext already matches the theoretical one exactly; on a text as short as 196 letters, spread over up to 25 symbol categories, sampling noise alone is enough to invert the rank of two letters of close theoretical frequency (N and I, or R and H, for instance), which silently invalidates the entire resulting key. Simulated annealing makes no such single-shot assumption: each combination explores thousands of candidate keys, scored not on isolated letter frequency but on the coherence of overlapping four-letter windows (section 5), and is measured, once found, against the null-baseline calibration (section 5.9) rather than trusted on its score alone.

5.8 Nine Transcription Hypotheses

Beyond the text as printed, eight additional variants are tested jointly (Table 1), each motivated by a concrete clue rather than by arbitrary exploration.

Table 1: *The 9 base text hypotheses.*

Variant	No.	Justification
Normal	1	Transcription as printed (reference)
Inverted	1	Sequence read backwards
Position 98 (r6-r9)	4	Pair #98 ("04") is the only one to break a pattern noted independently by a third-party researcher; its 4 plausible alternative readings $\{6, 7, 8, 9\}$ are tested rather than a single arbitrary correction
Row / column / both mirror	3	A rotation of the digit→index mapping is already covered by the Z5 key search; a mirror genuinely alters the cyclic structure exploited by the modular re-encryption and the Bifid

5.9 Three Languages, One Methodology to Adjudicate Between Them

The full pipeline was replicated on a **French** model (Gutenberg corpus, accents normalized) and a **transliterated Hebrew** model (22 Hebrew consonants → 22 distinct Latin letters, unpointed text). A methodological problem arose immediately: with hundreds of thousands of combinations tested independently, the *best* observed score naturally drifts upward, even on pure chance (a multiple-comparisons effect). The single-trial self-test reference is not sufficient to judge a score obtained after a large search volume.

Exhibit 2: Null-Baseline Calibration Methodology

Run the exact same pipeline on a randomly shuffled sequence, with the same frequency distribution (same index of coincidence) as the real cryptogram, and compare the two *at equal trial volume*. A score that does not exceed what pure noise achieves at the same volume is not a signal.

This methodology proved its worth from its very first application: the Hebrew hypothesis produced a promising score (-993.83), even exceeding its own self-test reference. At a comparable volume (41,271 trials on each side), the Hebrew null baseline reached -985.48, *better* than the real result (Figure 11). The explanation retained: the transliterated Hebrew alphabet (22 letters) produces a less sparse quadgram model ($22^4 = 234,256$ categories versus $456,976$), statistically easier to satisfy by chance. French, for its part, showed no comparable signal (best score -1138.93, worse than its own self-test reference -1032.6).

5.10 Optimization

The score is computed via JIT compilation (`numba`) and vectorized `numpy`. A bottleneck was identified in the column-reading function (a pure Python loop); vectorizing it produced a speedup factor of ≈ 5.6, benefiting all transposition methods.

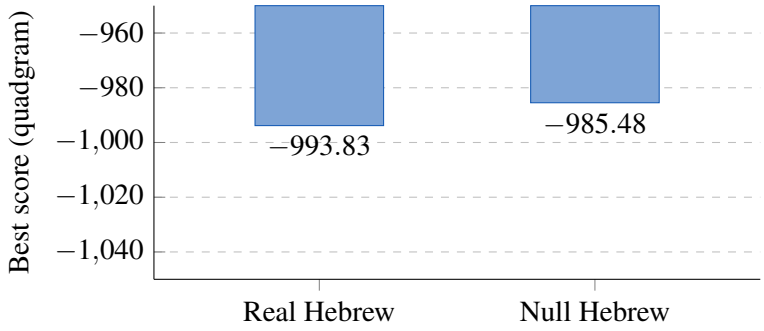

Figure 11: *Pure noise outperforms the real cryptogram under the Hebrew model, at equal trial volume.* The −993.83 score is not a signal, but an artifact of the smaller transliterated alphabet.

6. The Project's Statistical Metrics for Measuring Success

6.1 Index of Coincidence

For a sequence of n symbols over a 25-value alphabet, with n_s occurrences of symbol s:

$$IC = \frac{\sum_{s=0}^{24} n_s(n_s-1)}{n(n-1)}$$

Invariant under substitution (depends only on frequencies, not order), the IC weights (without ever excluding) super-encryption streams that are statistically closest to a plausible monoalphabetic text.

6.2 Quadgram Score and Bandit Weights

Defined in section 5. The first is the underlying metric; the second dynamically steers computational effort toward the most promising areas, without ever closing off any option.

6.3 Null-Baseline Calibration

Defined in section 5.9. It addresses an unavoidable statistical limitation: over a large number of independent trials, the *maximum* score achieved by pure noise follows an extreme-value distribution that grows with the volume tested. A valid comparison must therefore pit two distributions of maxima obtained at the same volume against each other, never an isolated score against a single-trial reference.

6.4 Search Space Coverage

Since sampling is weighted and not exhaustive, "average" coverage (trials ÷ space size) overestimates actual coverage. The fraction of streams never sampled is approximated by a Poisson distribution:

$$\text{fraction never drawn} \approx \frac{1}{M}\sum_{i=1}^{M} e^{-\lambda_i}, \qquad \lambda_i = N_{\text{trials}} \times \frac{w_i}{\sum_j w_j}$$

used in section 7 to honestly quantify the search's actual progress, beyond the simple trials/space ratio.

7. State of Play After 48 Hours of Computation

7.1 Calibrations

Table 2 summarizes, for the three languages tested, the gap between an authentic text and pure noise under each model: the margin within which a result must fall to merit examination.

Table 2: *Self-test (single-trial) calibration by language.*

Language	**Real-text score**	**Noise score (range)**	**Corpus**
English	≈ -935 to -1140	-1500 to -1590	3.29 M characters
French	-1032.6	-1594 to -1600	3.04 M characters
Transliterated Hebrew	-1055 to -1080	-1340 to -1480	746,000 characters

7.2 Space Size and Actual Coverage

With 9 base text hypotheses and 146,529 super-encryption streams generated, the combined space of the four methods represents approximately 5.97 million distinct structural combinations.

7.3 Best Scores and Verdict

The best score across all methods, at this stage of the first 48 hours, is -1078.76 (simple transposition, `pos98_r7` hypothesis), to be compared with the English null baseline, which at a comparable volume already reached -1107 to -1119. The residual gap (11 to 17 points) remains in a zone where the statistical origin of the signal cannot be established with confidence, and no decoded text shows lexical coherence beyond isolated fragments plausible by chance ("THE", "AND", "FOR"...).

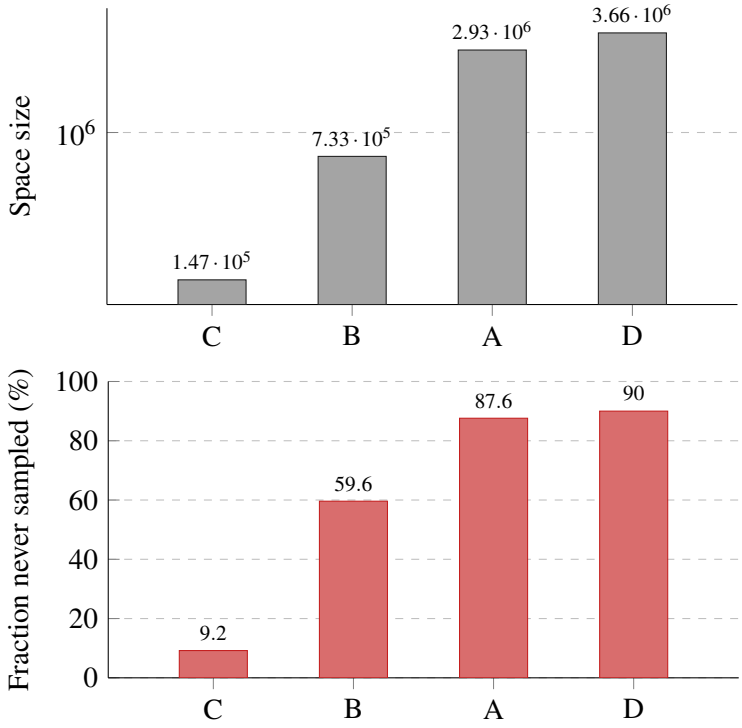

Figure 12: *Space size by method (left, log scale) and fraction never tested after ≈48h (right).* C = Fleissner, B = simple transposition, A = substitution+route, D = double transposition. Methods A and D, with the largest spaces, remain three to ten times less covered than B and C.

Exhibit 3: Verdict at This Stage (48h)

No statistically distinguishable signal from noise has been detected, in English, in French, or in transliterated Hebrew, over the ≈10 to 100% of the modeled search space covered after two days of continuous computation on 12 cores.

This best score continued to drift upward after the acceleration described in section 8: −904.18 at the time this update was written (double transposition, `normal` hypothesis). The corresponding decoded text nevertheless remains gibberish with no lexical coherence (`QFFRESKTHOFTHNISORN...`, 196 letters in total), which directly illustrates the methodological principle of section 5.9: at a much higher search throughput, the extreme-value distribution mechanically pushes up the *maximum* reached by pure noise, independently of any real signal being present. The English null baseline described in section 5.9 dates from the throughput before the acceleration and therefore no longer constitutes a valid reference at the current throughput; renewing it at the new throughput is part of the immediate work ahead (section 15).

8. Guaranteed Coverage and Full Compilation to Accelerate the Search

The state of play in the previous section revealed two structural limitations of the search engine, independent of the validity of the hypotheses tested. An examination of the sampling mechanism and of the

execution profile made it possible to correct them without reopening the question of the hypothesis space itself: the methodology remains that of section 5, only its execution is revised.

8.1 The Hidden Cost of Random Sampling With Replacement

The coverage formula from section 5 (paragraph "Search Space Coverage") has a counter-intuitive consequence, left implicit until now: at a trial volume N_{trials} equal to the space size M (i.e. $\lambda = 1$, what might be called a *nominal pass*), the fraction of combinations *never* drawn stays at $e^{-1} \approx 36.8\%$. Weighted sampling, however useful for concentrating effort on promising areas, therefore never guarantees complete coverage: more than a third of the space can remain entirely unexplored by pure sampling chance, even after a number of trials equal to its size.

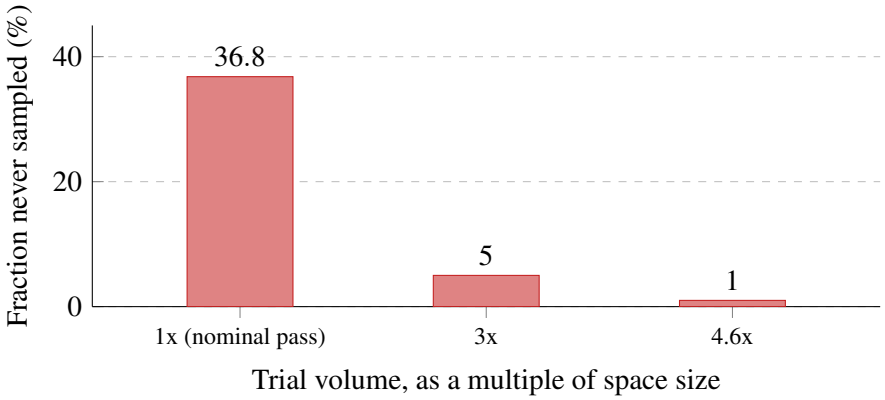

Figure 13: *Fraction of the space never sampled as a function of trial volume (Poisson approximation, uniform weights).* Reaching 95% coverage (at least one draw per combination) requires ≈ 3 times the volume of a nominal pass; 99% requires ≈ 4.6 times.

In other words, guaranteeing reasonable confidence that no combination was left untested requires 3 to 4.6 times more trials than a simple nominal pass, not because of a computational error, but because of a structural property of random sampling with replacement (the so-called “coupon collector” problem).

8.2 Two-Phase Systematic Sweep

The fix adopted retains the existing weighted draw for refinement, but adds to it a preliminary exhaustive-sweep phase guaranteeing, before any recourse to randomness, that no combination is ever distributed twice. For each of the four methods, an integer counter shared across the 12 processes (`multiprocessing.Value`, natively locked) distributes a unique index within that method’s combined stream $\times$ arm space, via lock-protected atomic increment (Figure 14). Each process that obtains an index decodes, by simple integer division, which stream and which structural arm it must test; two concurrent processes can never receive the same index, the guarantee being provided at the operating-system level by the counter’s lock, not by a programming convention.

This mechanism applies independently to the four methods: previously, a single stream was drawn per

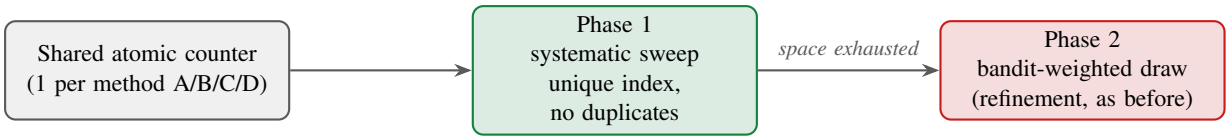

Figure 14: *Mechanism of the two-phase systematic sweep.* Phase 1 guarantees 100% coverage before Phase 2 resumes the exploitation/exploration refinement role already described in section 5.6.

iteration and shared among A, B, C and D; it is now drawn separately for each, a necessary condition for each method to cover its own stream $\times$ arm space without being coupled to the other three. Exact coverage (no longer estimated via the Poisson formula) has since been logged continuously to a status file, readable without recomputation.

8.3 Full numba Compilation of the Search Loop

Before this change, only the quadgram score computation $Q(k)$ was JIT-compiled (section 5, paragraph "Optimization"); the surrounding simulated-annealing loop, key permutation, structural extraction, Metropolis acceptance test, cooling, remained in interpreted Python, with that language's usual call overhead at every iteration.

A prototype limited to method A alone (the simplest of the four algorithms, with a single possible perturbation) first allowed measuring a $\times 4.85$ gain before undertaking the heavier port of methods B, C and D. Their common generic function (joint structure + key search, section 5) relies on Python closures passed as arguments, a pattern that `numba` compilation in *nopython* mode does not support as-is: three specialized loops had to be written, one per structural family (column order, Fleissner grille, order pair), rather than reusing the existing generic abstraction.

Exhibit 4: Validation Before Deployment

Two systematic checks precede any replacement of the Python loop with its compiled equivalent: (1) **numerical fidelity**, the `numba` score and the current Python score, evaluated on the same text and the same key, must match to within machine rounding (measured difference: exactly 0); (2) **search quality**, at an identical iteration budget, on a case with a known signal (an authentic English excerpt encrypted by the pipeline itself), the best score reached by the compiled version must remain within the same range as the Python version. Both conditions were verified before production deployment.

8.4 New Time Estimate

Table 3 recalculates the time needed for guaranteed 100% coverage (section 8.2) of each method, based on the throughput measured after the acceleration.

The bottleneck shifts methods: D, previously the slowest to cover despite its larger space, gives way to

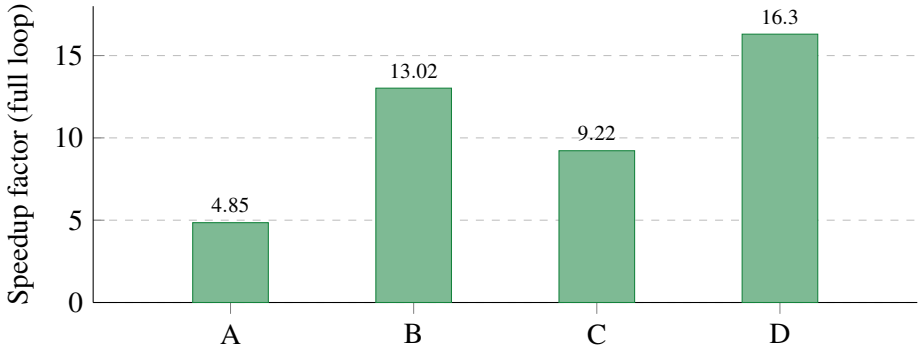

Figure 15: *Speedup factor measured per method, full search loop (score and perturbation logic compiled).* Method A, simpler, had proportionally less Python overhead to eliminate than B, C and D, which additionally handle a structural state extracted at every perturbation.

Table 3: *Estimated time to guaranteed 100% coverage, before and after the acceleration.*

Method	**Space size**	**Speedup**	**Before**	**After**
C (Fleissner)	146,529	×9.22	≈0.75 d	≈2 h
B (simple transposition)	732,645	×13.02	≈3.8 d	≈7 h
A (substitution+route)	2,930,580	×4.85	≈15 d	**≈3.1 d**
D (double transposition)	3,663,225	×16.30	≈18-19 d	≈1.1-1.2 d

A, whose measured speedup is the lowest of the four. Guaranteed 100% coverage of the four methods is now expected in ≈**3 days**, versus 18 to 19 days before this change, with no combination drawn twice before all others have been tested at least once.

8.5 The Logging Bottleneck

A few hours after deploying sections 8.2 and 8.3, throughput observed in production dropped by about 30% and measured CPU load stayed stuck at 35-53% instead of ≈100% despite 12 active processes. A secondary factor was identified first: the project folder, synced by a cloud client, was continuously attempting to re-sync the results file, already large and growing fast; pausing it, however, reduced CPU load only marginally (37-53%), a sign that it was not the main cause.

The actual cause lay in the logging logic itself. The search function only wrote a result if its score exceeded the best score already observed *for that exact combination*, a dictionary initialized to $-\infty$ at each process's startup. Before the systematic sweep, a combination could be redrawn several times by the weighted draw, so this filter eliminated most duplicates. But Phase 1 (section 8.2) guarantees by construction that a combination is *never* revisited: the test "did I do better than last time on this combination" then becomes trivially true on the very first (and only) visit. Measured in production: ≈97% of draws triggered a write (23,281 writes for 23,996 draws over a measurement interval), saturating the logging lock shared across the 12 processes, each opening, writing to, then closing an ever-growing file on almost every iteration: it was this I/O bottleneck, not cloud syncing, that was blocking the CPU.

Fix. The per-unique-combination dictionary was replaced with four scalars, a personal record per process and per method (A/B/C/D), restoring the original intent: log only genuine improvements, increasingly rare as the search progresses. Measured after the fix: ≈0.05% of draws trigger a write, and CPU load returned to 100% across repeated measurements after redeployment.

This fix required restarting the processes, which would otherwise have reset the Phase 1 counters to zero. Persistence was therefore added at the same time: at startup, each script re-reads the existing status file and resumes the systematic sweep exactly where it left off, instead of starting over. Any subsequent restart (crash, code update, machine reboot) automatically benefits from this.

8.6 Can the Personal-Record Safety Net Miss the Right Answer?

The previous fix raises a fundamental question, distinct from performance: can a personal record per process and per method cause a correct result to be missed? Yes, in principle: if a process encounters, early on, a noise combination that, through optimization, scores better than an authentic English text does (empirically observed, section 9.3), a "real English" quality result encountered *later* by that same process would never beat its noise record, and would therefore never be logged, even though the search did test and correctly score it.

A first correction, an absolute score threshold (score $> \mathscr{S} - 50$, in addition to the personal record), was **tested and rejected**: measured in isolation before any deployment, it would have triggered a write on ≈43% of draws (Figure 16): simulated annealing pushes almost any 196-symbol sequence toward a "plausible" score, signal or not, so the "as good as real English text" zone is not rare once optimized. Such a threshold would have recreated the I/O bottleneck of section 8.5.

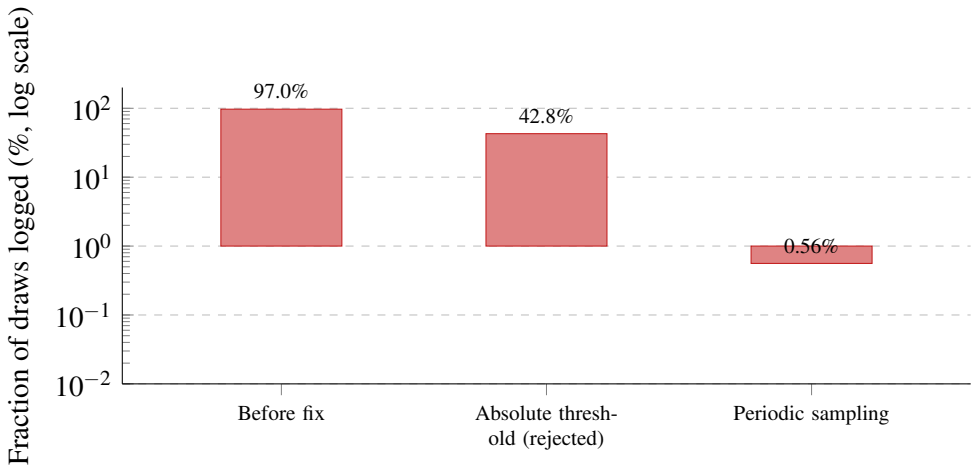

Figure 16: *Three measured logging regimes.* The absolute score threshold, though designed as an infrequent safety net, turned out to be almost as costly as no filter at all: the "plausible" score zone is not rare once simulated annealing is applied.

The solution adopted, **periodic systematic sampling**, unconditionally logs one draw in 2,000 (independent of its score), in addition to the personal record: representative visibility into what is being tested,

without depending on a score threshold that filters almost nothing. Measured in steady state after deployment: ≈0.56% of draws logged (Figure 16), CPU load still at 100%.

Exhibit 5: Acknowledged Limitation of the Safety Net

Periodic sampling does not guarantee capturing precisely the right answer if it exists: it provides representative, not exhaustive, visibility into correct-but-not-record results. Nothing is lost on the *computation* side (Phase 1 guarantees that every combination is tested and scored at least once); the limitation concerns only *logging*, and hence the human visibility of the result after the fact.

9. Repairing the Null Baseline, the Score Race, and the Updated Verdict

9.1 Repairing the English Null Baseline

The English null-baseline script (section 5.9) calls the main solver's search function directly, which caused it to automatically inherit the accelerations of section 8, but also two incompatibilities introduced by those same changes.

1. **Incompatible signature.** Adding the systematic-sweep counters to the search function (section 8.2) broke the calibration script's call, which did not supply them: an immediate, reproducible crash, confirmed by running the script rather than merely re-reading the code.
2. **Status never logged.** The calibration script offsets its process IDs (to distinguish them from the main run in the logs), but writing the status file was conditioned on a specific ID, never reached with this offset. The computation was progressing normally but remained invisible: discovered after an hour of computation with no progress data appearing.

Both were fixed (counters and resume point specific to the calibration script; generalized logging condition) and verified by execution before redeployment, consistent with the practice already applied in section 8.3. The null baseline has since run with 4 processes, a compromise between rigor (adequately covering its space, nine times smaller than the main run's since only one transcription hypothesis is tested there versus nine) and contention (≈18-21% slowdown of the main run was measured and deemed acceptable). Its Phase 1 has since reached 100% on its four methods.

9.2 The Score Race

The best score of each run was tracked at regular intervals as the null baseline caught up in volume (Table 4, Figure 17).

Table 4: *Timeline of the score race, real run versus null baseline.*

Stage	Real score	Null score	Null volume tested
Before repairing the null baseline	−904.18	(unavailable)	0
Null baseline catching up	−893.55	−905.74	≈96,500
Null Phase 1 finished at 100%	−878.89	**−876.55**	≈830,000+

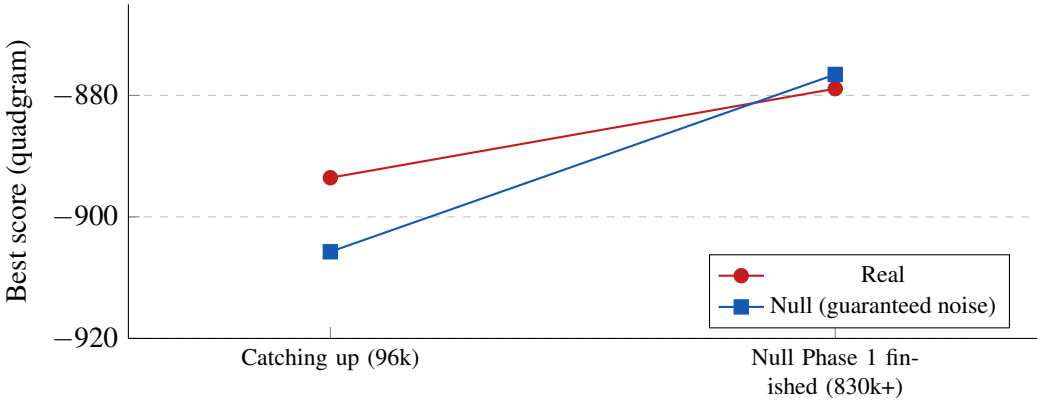

Figure 17: *The null baseline catches up to and then surpasses the real score as its tested volume grows.* The same pattern as the debunking of the Hebrew hypothesis (Figure 11): a gap that seemed favorable to the real signal vanishes once the volume is compared fairly.

Exhibit 6: Updated Verdict

Once the null baseline reaches full coverage of its space, its best score (−876.55) *exceeds* the best real score (−878.89). No statistically distinguishable signal from noise is established at this stage, including for the best result found to date on the real cryptogram.

Far from closing, this gap has since widened. The best real score has remained stable at −877.84 across all subsequent checkpoints, while the null baseline continued to progress to −860.37 after completing its second systematic sweep (section 11) over its four methods, widening the gap to 17.47 points against the real signal, versus 1.29 points at the time of the table above. The text corresponding to this new null record, inspected directly, shows no trace of lexical coherence beyond the isolated fragments already characteristic of noise (section 12).

9.3 Comparison to an Authentic English Text

The self-test calibrations (Table 2) give the score range achieved by a real 196-letter English text under this model: ≈ −935 to −1160 depending on the excerpt. Yet the two current best scores, real *and* null, *both* exceed this range (Figure 18).

This finding, consistent with the best result's decoded text remaining gibberish with no lexical coherence (section 9.5), strips the score alone of any discriminating filter value at this search volume: exceeding an authentic English text is no longer surprising, neither for the real run nor for guaranteed noise.

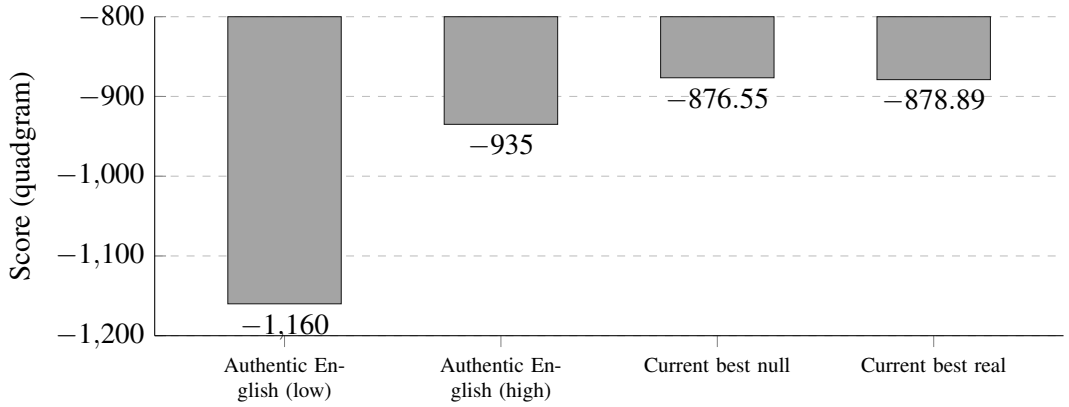

Figure 18: *Both current best scores exceed the range achieved by an authentic English text.* With enough combinations tested, optimization finds arrangements that satisfy the quadgram metric better than a random English passage does, without being real English: a multiple-comparisons effect, not a signal.

9.4 Reconstructing the Pipeline and Key for the Best Real Result

The full pipeline that produced -878.89 was reconstructed and verified algebraically, not merely reaffirmed from the log: transcription variant `pos98_r6` (section 5.8), Z5 modular super-encryption with key $(1,1)$ for rows and $(3,3)$ for columns (constant shift -1 and -3 modulo 5), method D with two transposition passes on a 49×4 grid (two distinct column orders, logged).

The substitution key itself was not logged originally, only the final decoded text was. It was recovered by direct deduction, not by a new search: replaying the known transformations (above) on the raw cryptogram yields the symbol sequence just before substitution is applied; comparing it position by position with the already-logged decoded text, each symbol recurs on average 7 to 8 times across the 196 positions, and its associated letter must be identical at every occurrence if the reconstruction is correct. Zero contradiction across all positions confirms the reconstruction's accuracy, further verified by exact recomputation of the index of coincidence and the score (Table 5).

Table 5: *Verification of the pipeline reconstruction for the* -878.89 *result.*

Quantity	**Logged**	**Recomputed**
Index of coincidence	0.0705	0.0705
Decoded text (196 letters)	(in the log)	identical character for character
Quadgram score	-878.89	-878.89

9.5 Exhaustive Lexical Intelligibility Scan

The score and the comparison to noise (sections 9.2-9.3) concern a single result at a time. A distinct question arises: across all results logged since the project began, is there any trace of lexical coherence anywhere other than in the single best score? Both results files were scanned exhaustively: **3,868,347 decoded texts** in total.

Method. A dictionary of ≈2,528 common English words was extracted from the project corpus. For each decoded text, a trie structure (prefix tree) searches, at every position, for the longest dictionary word starting there, then advances; the coverage rate is the fraction of characters thus explained by chained real words.

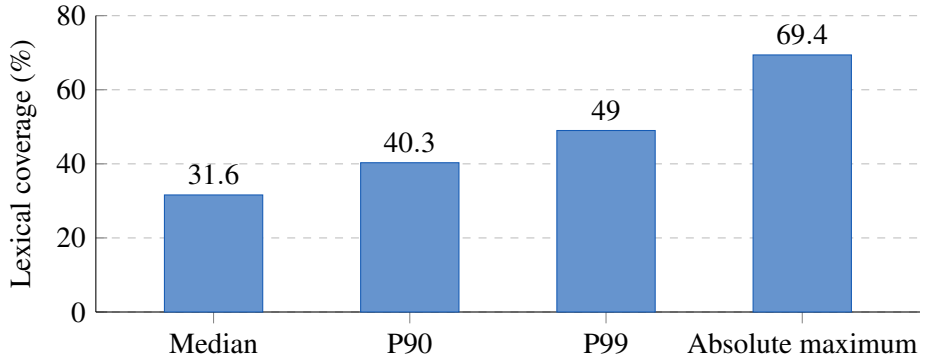

Figure 19: *Distribution of the lexical coverage rate over the 3,868,347 logged decoded texts.* Even the absolute maximum (69.4%) corresponds only to isolated short words ("OR", "ONLY", "THEM"...) with no run forming a sentence.

Result. Median coverage 31.6%, P90 40.3%, P99 49.0%, absolute maximum 69.4% across the entire results corpus (Figure 19). No text, at any coverage level, shows a run of words forming a recognizable sentence. Notable fact: the text with the largest total number of detected words (55, of which 31 are three letters or longer) comes from the **null baseline**, guaranteed noise, not from the real run, ahead of the entirety of the authentic cryptogram's results on this criterion too.

Exhibit 7: Verdict of the Intelligibility Scan

Neither the quadgram score nor a lexical coherence measure independent of the score distinguishes the real cryptogram from pure noise at this stage of coverage. Across nearly 3.9 million decoded texts examined, none shows recognizable linguistic structure beyond what a dictionary of short words explains by chance.

10. The Lexical Intelligibility Score, a Second Detection Axis

The preceding sections establish that no signal stands out from noise on the quadgram score, the only selection metric used so far. A distinct question follows: could this score, by construction, have kept a readable but poorly scored result out of the logs under this model, particularly among draws that predate the safety net of section 8.6? This section formalizes a second criterion, independent of the first, designed to answer that.

10.1 Why Not Drive Simulated Annealing With a Lexical Score

The most direct option would be to substitute, within the simulated-annealing loop itself, a real-word coverage score for the quadgram score $Q(k)$. This option was ruled out for a structural reason, well documented in automated cryptanalysis: the fitness landscape of a lexical score is nearly flat everywhere. A randomly drawn key, over 196 characters, practically never produces a recognizable word; the algorithm would then have no gradient to follow for the majority of the two-letter swaps tested. The quadgram score, by contrast, varies continuously with every swap (each letter contributes to several overlapping windows), providing the dense gradient needed for annealing to converge. Driving the optimization with a lexical score would thus amount, over most of its trajectory, to a near-random search, presumably less effective than the existing setup.

Simulated annealing therefore continues to maximize $Q(k)$, unchanged. The intelligibility measure is computed *a posteriori*, once per combination tested, once the optimal key has already been found: its cost is negligible compared to the optimization's own 20,000 to 30,000 iterations.

10.2 Definition of the Criterion

For a decoded text, two quantities are retained, computed by longest-prefix segmentation (trie structure) against a reference dictionary:

- Cov, the fraction of the 196 characters explained by a chain of real words, possibly interspersed with unexplained letters;
- Run, the length (in words) of the longest *continuous run*, with no unexplained character whatsoever, a requirement strictly stronger than coverage alone and closer, by construction, to what a recognizable sentence would require.

Each process keeps, independently of the quadgram score record, a personal record of Cov and of Run per method. A result is logged as soon as it beats any one of the three records (score, coverage, run), in addition to the periodic sampling already in place (section 8.6): the mechanism thus recovers, without reproducing the I/O bottleneck of section 8.5, results that are highly readable but poorly scored by the quadgram model.

10.3 A Contaminated Dictionary, Caught Before Any Deployment

The dictionary used in the exploratory scan of section 9.5 was, by ad hoc design, limited to the 2,528 most frequent words in the project corpus. A first attempt at expanding it combined this corpus with a list of "common" English words fetched online (derived from web usage frequencies), bringing the dictionary to 19,208 entries.

Consistent with the practice already applied in section 8.6 (always measure a new filter's effect on noise before trusting it), this expanded dictionary was tested on purely random noise before any deployment. The result invalidated the attempt: a median coverage of 72.4% on guaranteed noise (Figure 20), far above the 31 to 37% range observed until then. Inspection revealed the cause: the external list, derived from web frequencies rather than a linguistic dictionary, contained 404 two-letter "words" out of the 676 possible combinations, overwhelmingly acronyms and technical jargon (`AOL`, `API`, `ACM`, `AMD`) rather than genuine English words, which match any letter sequence by pure combinatorial chance.

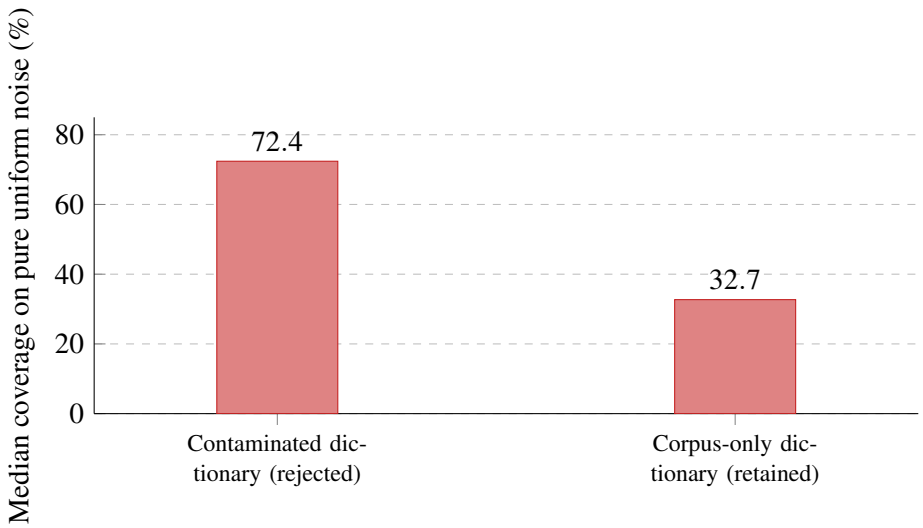

Figure 20: *Effect of a contaminated dictionary on the noise floor of the coverage criterion.* The external list (19,208 words) more than doubles the coverage measured on guaranteed noise, rendering the criterion nearly non-discriminating; the dictionary retained is limited to the project corpus, with no arbitrary cap (23,310 words, frequency ≥ 1).

The dictionary ultimately retained discards the external list and keeps only the words from the novel corpus already used for the language model, consistent with the ordinary-English-prose hypothesis (section 3), without the arbitrary 2,528-word cap of the initial ad hoc scan: the full set of distinct words with frequency ≥ 1, i.e. 23,310 entries. Tested on the same reference noise, this dictionary yields a median coverage of 32.7% (Figure 20), consistent with the earlier measurements of section 9.5.

Exhibit 8: Deployment

The criterion was deployed after double verification in isolation (`-workers 2`): a first trial with the contaminated dictionary confirmed the predicted symptom (93 to 97% coverage on both real *and* null results, a sign of a non-discriminating criterion) before any production deployment; once fixed, the second trial produced coverages of 65 to 81%, consistent on both sides, with no error. Redeployed at full scale with no loss of coverage in the sweeps under way.

11. The Second Systematic Sweep of the Refinement Phase

The two-phase mechanism of section 8.2 guarantees that no combination is left aside on the first pass, but its former Phase 2 (bandit-weighted draw, section 5.6) suffers from a limitation revealed by the score race in section 9.2: the null baseline's best score for method D stayed frozen for more than eleven hours despite more than a day of weighted sampling, a sign of near-zero returns once Phase 1 is exhausted.

Decisive observation. Nearly all of the first systematic sweep, all combinations of methods A, B, C and most of D, took place *before* the intelligibility criterion was introduced (section 10): these combinations were therefore never evaluated under this second criterion. Yet the former Phase 2, weighted by the best *quadgram* score observed per arm, in no way guarantees revisiting arms that score poorly under this model but are potentially rich in lexical intelligibility, exactly the combinations one seeks to catch up on.

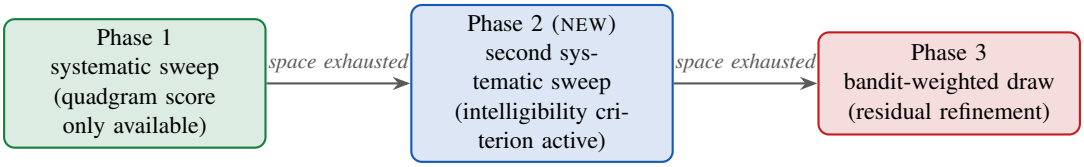

Figure 21: *Three-tier architecture, replacing the direct Phase 1 → bandit chain of Figure 14.* The second sweep (Phase 2, unbiased by score, guaranteed no duplicates like Phase 1) ensures every combination is tested at least once under both selection criteria; the bandit resumes its refinement role only once both guaranteed passes are exhausted.

The implementation reuses the atomic counter mechanism from section 8.2, duplicated for a second independent sweep of the same space (Figure 21): deterministic, no duplicates, not weighted by score. The time cost of a full second pass is identical to that of the first (same space size, same annealing cost per combination); at the measured production throughput, several additional days are needed to complete the second sweep of methods A and D, deemed acceptable given the project's exhaustiveness objective.

12. Hunting for Lexically Coherent Fragments

The coverage scan of section 9.5 measures a global property (fraction of characters explained) but does not guarantee that a fragment of several consecutive words, the only genuinely convincing indicator of linguistic coherence, is absent from the accumulated results. This section documents a targeted search, using two complementary methods, for such fragments.

12.1 Verification by Word N-grams Attested in the Corpus

A first automated method searches, among the consecutive, gap-free words of a decoded text, for pairs or triplets that reproduce a sequence actually present, in that order, in the training corpus. Applied to all texts with coverage $\geq 50\%$, this method reveals an immediate limitation: 85% of texts contain at least one attested word pair, a rate far too high to be informative, since the most frequent function words of

English (*it, is, of, to, as...*) form attested pairs with an already substantial probability by pure chance. Restricting to triplets reduces this rate to 163 occurrences out of $6,282$ texts examined, but nearly all remain composed of the same function words ("so is it", "it is the"...). An additional filter, requiring that all three words be at least three letters long, leaves only **3 occurrences across all texts examined**.

This method, however, has a limitation of principle, acknowledged before drawing any definitive conclusions from it: it can only confirm that a lexical coincidence is *probably* fortuitous (by establishing that it reproduces a sequence already encountered), but it can never prove the absence of an original, coherent fragment, which would have no reason to reproduce word for word a sentence from the five novels in the training corpus.

12.2 Exhaustive Direct Reading

As a complement, the 515 texts with the highest lexical coverage ($\geq 80\%$, a deliberately high threshold, well above the $\approx 33\%$ noise floor established in section 10.3) were read in full with direct semantic judgment, with no mechanical intermediary. The pattern observed throughout the scan in section 9.5 is confirmed there without exception: isolated short words, sometimes grouped two to four at a time, always embedded in nonsensical sequences on both sides.

12.3 The Most Striking Discovery of the Project to Date

A result from the null baseline, guaranteed noise by construction, contains an unbroken run of six real dictionary words, four of which form a grammatically flawless English clause (Figure 22). The sequence "if we can" was verified as actually attested, in that order, in the training corpus, confirming that this is not a segmentation error.

```
ZMOAFINKOFLEDDOCKORIKECERNTNOTOWACALLFONEUTSLIKIN
STWARONGUDMASOLDSTAPAGUPSSNORANDDOIMUMMINDLXNSNOS
URESUPSOFCOUNTHANSWEWWILLONSTREMT HELDIFWECANWIN VE
RPRUSKDARKAQGTOSURVEHUMANDARWHORGAMOOUKNORLXDEDFE
```

Figure 22: *A six-word consecutive fragment with no unexplained character, extracted from a null-baseline result* (`Score: -988.82`, DoubleTransp $49 \times 4 \rightarrow 49 \times 4$, `Cov:0.857 Run:12`). Segmented: "HELD IF WE CAN WIN VE"; the sub-fragment "IF WE CAN WIN" forms a grammatically correct English conditional clause. Provenance: pure noise, unrelated to the real cryptogram.

This result is significant not for what it might reveal about the cryptogram (it has nothing to do with it, coming from a randomly shuffled sequence), but for what it demonstrates about the power of chance at this search volume: a grammatically coherent run of six words can spontaneously appear in pure noise once hundreds of thousands of independent draws have accumulated. It is the most direct demonstration to date that a result's apparent lexical coherence, however impressive on isolated reading, constitutes no proof of signal whatsoever.

12.4 Summary of the Best Fragments Identified

Combining total detected word count and the presence of a recognizable fragment, the project's three best results are summarized in Table 6. Notably, the two texts holding the absolute lexical-coverage records (section 10, 98% on the real side and 100% on the null side) contain, once inspected directly, *no* coherent fragment whatsoever despite their superior coverage score: raw coverage can be inflated by dozens of two-letter word coincidences without any of them chaining into anything readable.

Table 6: *The three best lexically coherent fragments identified across the whole project.*

Fragment	**Provenance**	**Total words (text)**	**Fragment length**
"WALK OUT"	Real	56	59-word gap-free run
"FOR HER TO TRY"	Real	56	4 consecutive grammatical words
"THE WAY HAD"	Null (noise)	50	3 consecutive words

13. Searching for a Second-Order Structural Signal

A natural question, once it is established that no individual result stands out from noise (sections 9.2, 10, 12), is whether results that "look like something" might nonetheless serve as a starting point for further refinement, on the model of simulated annealing, which exploits a score gradient within a given combination. The answer in principle is no: since pure noise reaches or exceeds the real signal on all three metrics already measured, a result's "resemblance" carries, by construction, no information about its proximity to any real answer; restarting from these results would amount to optimizing for coincidence.

13.1 Clustering by Structural Parameter

A more indirect avenue remains conceivable: if one particular structural combination (grille, reading route, transcription hypothesis) is the right one, its best scores should cluster distinctively, even if no individual score taken in isolation stands out. Clustering the top 200 scores of each run by method and grille dimension is unambiguously shared by noise: 90.5% of the real top 200 and 80.5% of the null top 200 come from method D on the $49 \times 4 \rightarrow 49 \times 4$ grid. This convergence does *not* constitute a signal; it reflects the combinatorial power specific to this structural family (the largest space, the most rearrangement degrees of freedom), independently of any truth about the plaintext.

13.2 Success Rate Normalized by Transcription Variant

A finer analysis, restricted to the dominant grid, compares for each transcription hypothesis (section 5.8) its success rate in the top 200, normalized by the volume actually tested (rather than the raw count alone, which is misleading: the `normal` variant alone accounts for 56% of the volume tested on this

grid, versus 5 to 6% for each of the other eight). The result, summarized in Figure 23, shows a clear hierarchy: `inverse` reaches the top 200 about four times more often per draw than `normal`, followed by `lignes_miroir`, `pos98_r9`, and `colonnes_miroir`.

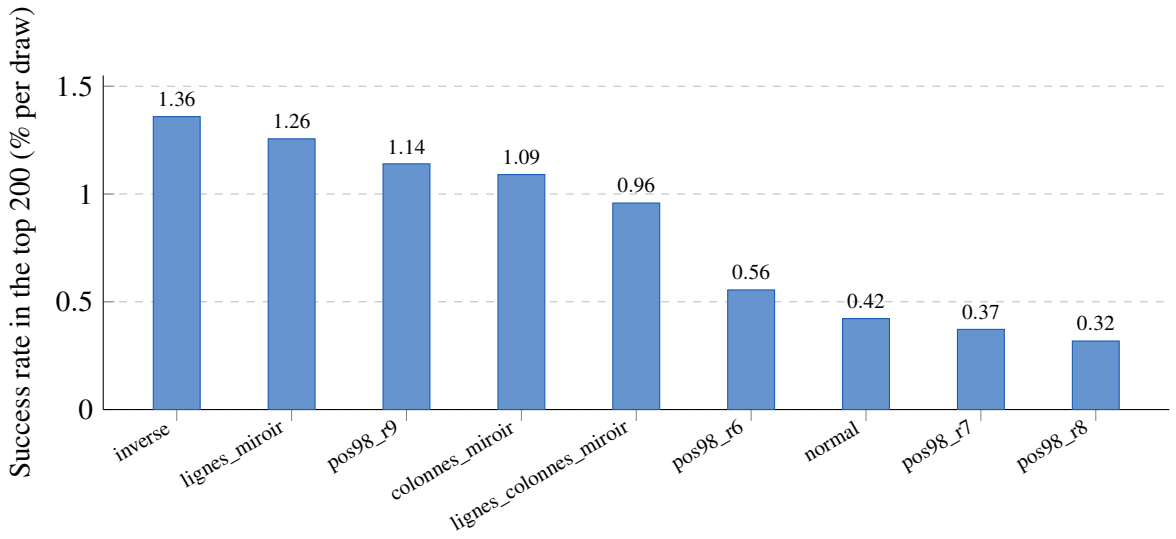

Figure 23: *Success rate in the top 200 of real scores, by transcription variant, normalized by volume tested.* The `normal` variant, despite 56% of the total volume tested, shows one of the lowest success rates; `inverse` succeeds about four times more often per draw.

13.3 A Confound Identified Before Any Conclusion

This result, however, cannot be interpreted as a signal as it stands: it is confounded by the very mechanism of the adaptive bandit (section 5.6, now Phase 3 in Figure 21). An arm massively resampled because it scored well early on keeps accumulating volume, but each additional draw mechanically has less chance of beating a record already close to the local optimum found by simulated annealing; a rarely resampled arm, conversely, benefits from "fresher" draws, with a higher probability of novelty. The `normal` variant could thus show a low success rate *because it has already been extensively exploited*, independently of any question about the validity of the hypothesis itself. The current data does not allow these two explanations to be distinguished.

Exhibit 9: Phase Tagging, to Settle the Question With Future Data

The draw function was modified to return, in addition to the combination index, the number of the phase that produced it (1, 2, or 3), now logged with every result. Phases 1 and 2 (Figure 21) are guaranteed unbiased by construction, each stream being tested there exactly the same number of times; a future analysis restricted to these entries alone will eliminate the bandit self-reinforcement bias identified above. Deployed on 2026-08-11 and verified in isolation before full-scale redeployment; the two second systematic sweeps (section 11) on methods A and D, still far from complete at the time of writing, will provide the unbiased volume needed for a conclusion.

13.4 Checking the Available Volume and Closing the Thread

Phase tagging (Exhibit above) was designed to allow an unbiased repeat of the previous analysis once sufficient volume had accumulated. A check of the volume actually available, carried out once both systematic sweeps were complete on all four methods (main run as well as null baseline), nevertheless revealed a gap between the coverage of combinations *tested* and the volume of results *logged* under this tagging: logging remains deliberately sparse (one draw in 2000, plus personal-record improvements, section 8.6), mechanically capping the usable volume at a fixed fraction of the number of combinations swept, regardless of further progress. No entry carries the Phase 1 tag (all four methods had already completed their first sweep before the tagging itself was deployed); the Phase 2 tag counts only 2290 entries for method A and 2508 for method D, concentrated on three of the nine transcription variants for A and five for D, with the others showing no entries at all.

Widening the filter to untagged entries (predating the tagging's deployment, potentially contaminated by the old non-systematic bandit) was then attempted as a fallback volume. The result, summarized in Table 7, reveals contamination more severe than the bias initially suspected: the ranking produced by this widened filter faithfully follows the chronological order in which the nine variants were introduced into the code (`normal` and `inverse` first, the three "mirror" variants last, section 5.8) rather than any intrinsic quality difference, the longest-tested variant mechanically accumulating the most logging history. Further proof: this ranking *reverses* the one obtained by the first, biased attempt (section 13), where `inverse` dominated and `normal` showed the lowest rate: two biased measurements of the same quantity, mutually contradictory, cannot constitute a signal.

Variant	Rate A (%)	Rate D (%)
`normal`	**60.0**	**64.4**
`inverse`	21.4	8.1
`pos98_r6` – `r9`	9.1 – 9.6	7.9 – 8.2
"mirror" variants (3)	7.0 – 7.1	6.3 – 6.3

Table 7: *Rate of logged entries per volume tested, filter widened to untagged data.* The resulting ranking exactly reproduces the chronological order in which variants were introduced into the code, not a quality difference.

Neither the strict filter (insufficient, patchy volume) nor the widened filter (contaminated by a historical-exposure artifact) allows a reliable comparison with the current instrumentation. This line of analysis is therefore **closed for lack of data**, neither confirmed nor refuted on the merits; properly reopening it would require a design change (dedicated reinforced sampling, or systematic timestamping of logged entries).

14. The Four-square Cipher, a New Research Effort

14.1 Exhaustion of the A-D Space and Deliberate Shutdown

Guaranteed coverage of the two systematic sweeps on the four methods A through D reached 100%, main run and null baseline alike. The best real score, in the meantime, set a new record (-853.60, versus -877.84 previously), exceeding the null baseline's best score (-860.37) for the first time. The corresponding decoded text was nevertheless checked directly and contains no grammatically coherent run of words ("and low ward won… did tall one… wont dis and strove…"), consistent with the finding already established in section 9.2: large-scale optimization eventually produces, through the sheer effect of multiple comparisons, arrangements slightly better scored by the quadgram model without being real English. This score crossover is therefore not a signal.

Since Phase 3 (adaptive bandit, section 5.6) has historically contributed only a marginal gain once both systematic sweeps are exhausted (already observed for method D in the null baseline, section 9), and the A-D hypothesis space is now covered exhaustively with no signal emerging from it, both processes were cleanly stopped (coverage-state persistence unchanged, resumable without loss at any time) to devote effort to a cipher family absent from the modeled space until now.

14.2 Motivation

The most advanced independent research project on this cryptogram (dagapeyeffresearch.com) reports its most promising, though inconclusive, result under a "Four-square" model ($Q = -692.13$ under its own scoring system, weaker than ours): a cipher family structurally different from the substitution and transposition explored so far, combining substitution *and* diffusion within a single operation on pairs of letters rather than single letters.

14.3 Mechanics

The Four-square cipher operates on *digrams* (pairs of consecutive letters) by means of four 5×5 Polybius grilles arranged in a square: the top-left and bottom-right grilles carry the unmodified standard alphabet, while the top-right and bottom-left grilles are *keyed* (a permutation of the alphabet, to be determined). To decrypt an encrypted digram (C_1, C_2): the position of C_1 in the top-right grille gives row r_1; the position of C_2 in the bottom-left grille gives row r_2 and, crossed with the columns of the other grille, the two plaintext letters P_1 and P_2 are read off the fixed grilles at coordinates (r_1, c_1) and (r_2, c_2) (Figure 24), following the "rectangle rule", a four-grille generalization of the well-known Playfair cipher rule.

Applied to the cryptogram, the coordinate stream is first interpreted as 196 letters via the non-keyed Polybius grille (direct interpretation of the symbol's row/column coordinates as a grille position), then split into 98 consecutive digrams. The same 146,529 streams as method C (nine transcription variants $\times$

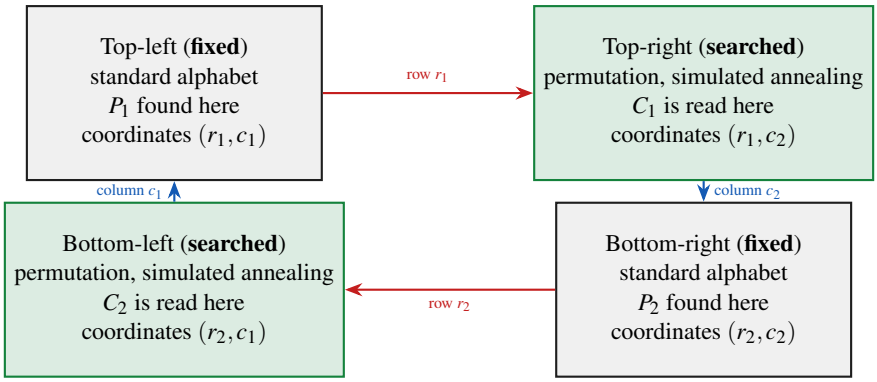

Figure 24: *Four-grille rectangle rule.* The two "searched" grilles (green) carry the effective key, determined by simulated annealing; the two fixed grilles serve as a geometric anchor for the search. Each digram of the cryptogram (196 symbols $\rightarrow$ 98 digrams) is decrypted independently by this same rule.

Z5/Bifid super-encryption, section 5.8) serve as input, guaranteeing reuse of the entire already-validated hypothesis-generation infrastructure.

14.4 Independent Validation of the Mechanics

Before any integration into the main pipeline, the implementation was validated separately: 2500 trials of encryption followed by decryption with the same random key recover the exact text in every case; the published reference case (keys `EXAMPLE/KEYWORD`, digram "HELP") was reproduced digit by digit through manual calculation independent of the code ("FYNF"), confirming that keyword-based grille construction and the rectangle rule conform to the standard definition; finally, the score obtained by the compiled `numba` kernel and its equivalent recomputed in plain Python match exactly (difference 0.00×10^0), the same verification standard applied during the full compilation of methods A through D (section 8).

14.5 An Unexpected Search Landscape

Searching for the optimal grille pair, unlike searching for the single substitution key of methods A through D, cannot be presumed trivial simply because the decoding mechanics are validated. A dedicated test (encrypting a known 196-letter English excerpt with a random Four-square key, then checking that simulated annealing recovers it, the same protocol as the self-calibration in section 5) revealed that the standard budget used for the other methods (20 to 30 thousand iterations, geometric cooling at 0.9995 per iteration) systematically fails: the score obtained remains rigorously identical from 30,000 to 300,000 iterations, a sign of premature freezing in a local optimum rather than genuine convergence. With this cooling schedule, temperature bottoms out after roughly 11,400 iterations, well before the standard budget even ends, which suffices for the plain-substitution landscape but proves insufficient for the markedly rougher landscape of a joint search over two coupled grilles.

14.6 Reliability Tuning

Two levers were explored to restore reliable convergence: slower cooling, and **multiple independent restarts** (a complete simulated annealing run repeated N times from a distinct random initial state, keeping only the best result). Each configuration's reliability was measured directly: the *exact* recovery rate of a known text encrypted with a random key, over 12 independent trials (Table 8); an initial evaluation limited to 3 trials had proved misleading (3 successes out of 3) and was discarded in favor of this larger measurement before drawing any conclusion.

Configuration	Cooling	Success (/12)	Time/combination
20 restarts × 250,000 it.	0.999988	58% (7)	≈30 s
30 restarts × 400,000 it.	0.999992	75% (9)	≈75 s
40 restarts × 300,000 it.	**0.99999**	**83% (10)**	≈76 s

Table 8: *Reliability measured by multi-restart simulated-annealing configuration.* At nearly identical compute volume (12 million iterations), 40 restarts of 300,000 iterations each clearly beat 30 restarts of 400,000, indicating that the *number* of independent random starts matters more than the convergence depth of each.

Two additional acceleration strategies, motivated by the high cost of the chosen configuration, were tested then rejected after measurement (Figure 25): *funnel filtering* (briefly probing the restarts, continuing only the most promising) caps out at 17–25% success, since under this cooling schedule temperature does not drop until roughly 570,000 iterations: a 15,000-iteration probe stays too "hot" to predict the final result; *internal reheating* (resuming from the best point found rather than a fresh random start, to save on warm-up cost) drops to 0% success on both configurations tested, confirming that the diversity provided by a genuine random restart outweighs deepening a single trajectory.

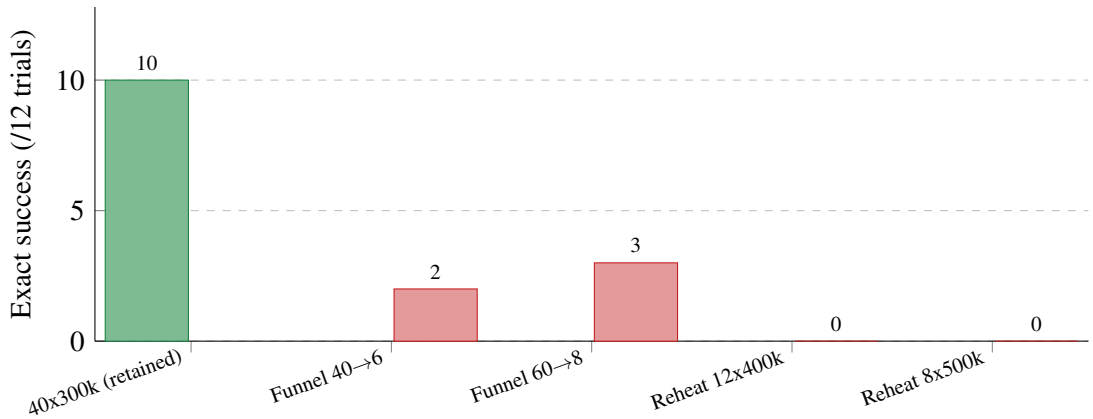

Figure 25: *Configuration retained (green) versus the four rejected acceleration attempts (red), at equal or lower compute volume.* None matches simply multiplying independent restarts.

No shortcut that leaves the score computation unaltered was thus found. Incremental score computation (recomputing only the quadgrams affected by a swap rather than the entire text) would remain the only unexplored avenue for gains, deemed riskier (since the score computation lies at the heart of the entire project methodology, an error there would be silent) and not attempted at this stage.

14.7 The Two-square Case, Set Aside

A second variant of the same family, *Two-square* (two grilles, both keyed, with no fixed grille to anchor the search), was implemented and validated with the same rigor, but proved unstable even at a much larger budget (30 to 40 restarts of 500 to 600 thousand iterations): the exact recovery rate varies from 7% to 97% of correct letters from one trial to another on the *same* known key, and further increasing the budget does not systematically improve the result. The hypothesis retained is the absence of a fixed grille to geometrically anchor the search, making the score landscape even more degenerate than Four-square's. The corresponding code is kept for future resumption under a different search strategy, rather than a simple increase in compute budget.

14.8 Deployment Architecture and Current Status

Four-square's cost per combination ($\approx$76 seconds) exceeds that of methods A through D ($\approx$50 milliseconds) by nearly 500-fold: direct integration into the same shared search loop would have slowed the latter down by several hundred-fold. An inline integration attempt, undertaken before this discovery, was entirely reverted before any production deployment. Four-square is deployed in a dedicated, independent script, nevertheless reusing the entire shared infrastructure (corpus, language model, intelligibility dictionary, generation of the 146,529 streams) and reproducing the same guaranteed two-phase systematic sweep as methods A through D (section 8.2), for the same exhaustive-coverage reasons.

Deployed in production with 12 parallel processes at the chosen configuration (83% measured reliability), the throughput observed under real conditions ($\approx$40 seconds per combination, cores fully dedicated) exceeds the throughput measured in isolated testing, bringing the estimate for the first complete sweep down to about 6 days rather than 11. The second sweep, once the first is complete, brings the cumulative risk of missing the right combination (if it exists) down from about 17% (a single pass at 83% reliability) to about 3% (two independent passes).

Monitoring continued over several computing sessions, with the number of parallel processes adjusted downward (12, then 6) to limit the thermal load on the machine used. When the search was stopped, coverage of the first systematic sweep had reached 30,562 combinations out of 146,529 (20.9%), with 3,500 results logged. The best score obtained, -981.97, remained stable across the last checkpoints and stays well below the A-D methods' record (-853.60); several lexical intelligibility scans run on new entries as progress continued (same method as section 9.5) revealed no coherent fragment beyond the records already reached by pure chance (maximum coverage 86.7%, longest run 22 words). As with methods A through D, no signal has therefore distinguished itself from noise over the fraction of the space covered to date.

15. What Remains to Be Explored

15.1 The Periodic-Nulls Hypothesis

On page 111 of his book, D'Agapeyeff himself describes the technique of inserting a dummy character every 3, 4, or 5 characters, making decryption "extremely difficult unless one is in on the secret." Nothing proves he applied this technique to his own cryptogram, but the hypothesis is plausible and has not been implemented: it would break the founding $196 = 14 \times 14$ assumption on which the current grille-dimension architecture entirely rests, and would require dynamic grille dimensions depending on the length after null removal.

15.2 Pelling's Discovery (2017)

Independent researcher Nick Pelling established that D'Agapeyeff describes the Kerckhoffs double-transposition method with numbering reversed relative to the standard convention, suggesting he may have mistakenly believed that this double transposition was self-inverse (encryption = decryption). In his own example from the book, this property holds by rare coincidence (an involutive permutation); it almost certainly would not for a 14×14 grille. If D'Agapeyeff applied the same flawed reasoning to his own cryptogram, his own attempt at verification would have failed, which would explain the forgetting anecdote. This discovery directly motivated the addition of method D (section 5): unlike D'Agapeyeff, our search assumes no symmetry and explores the full space of transposition pairs.

15.3 Crib-dragging

Actively searching for probable words (`CIPHER`, `SECRET`, `AGAPEYEFF`, `SCHUVALOW`) as anchors positioned within the candidate text, under the present project's calibrated quadgram model, has not been attempted. The most advanced external research project on this cryptogram tested it without success (116 cribs drawn from the book, 42 historical keywords), but under a substantially less rigorous scoring system (chi-squared and index of coincidence rather than a calibrated quadgram language model); a fresh attempt under our methodology was not judged a priority ahead of Four-square (section 14), but remains feasible at low cost.

15.4 Where Do Things Stand?

At the closure of this project, guaranteed coverage of the two systematic sweeps of the four methods A through D reaches 100%, on both the main run and the null baseline, with no signal emerging on any of the three metrics retained (quadgram score, lexical coverage, length of coherent fragments). The Four-square cipher, the only new structural avenue explored after this exhaustive coverage, was tracked to 20.9% of its first systematic sweep (section 14) with no signal appearing there either. The search

was halted at this stage, not because the hypothesis itself was exhausted, but because the computing resources available for this project, a personal machine run continuously over several weeks, no longer allowed continuing under reasonable time and thermal-load conditions. Any eventual discovery of a solution depends in any case on the validity of the structural premise underlying each modeled space, a hypothesis unconfirmed by decades of prior attempts, including an independent research project that tested more than 390 configurations over 30 hours of computation without success.

Several avenues identified over the course of the project remain unexplored. The periodic-nulls hypothesis and targeted crib-dragging, described above, have never been implemented. The Two-square cipher, whose mechanics are validated but whose search remains unstable with the current strategy (section 14), awaits a different search approach rather than a simple increase in compute budget. A null-baseline calibration dedicated to the French hypothesis, never built despite one existing for English and transliterated Hebrew (section 5.9), is likewise missing to judge that hypothesis with the same rigor as the other two. Four-square's own systematic sweep remains far short of its guaranteed coverage and can be resumed at any time with no loss: the code, corpora, frequency tables, and the exact coverage state at the time of the stop are published in open access on Zenodo (see References), allowing the search to resume exactly where it left off rather than starting from scratch.

16. Conclusion

Independently of solving the cryptogram itself, this project yields several methodological lessons transferable to other statistical cryptanalysis problems.

The first concerns calibration itself. A detection threshold fixed without empirical verification can turn out to be structurally unreachable, invalidating any negative conclusion drawn from failing to cross it (section 4). Once this threshold is replaced by a self-calibrated measure, a second trap remains: as soon as a large number of combinations is tested independently, the best score observed drifts upward even on pure chance. The resulting null-baseline methodology is applied systematically throughout this report whenever a result looks promising, rather than trusting an isolated score alone (section 9.2).

The hypothesis space was extended defensibly, double transposition, transcription variants, and multilingualism having been added only on the basis of concrete evidence or a concrete clue rather than arbitrary exploration, and this same discipline governed the acceleration of the search (section 8): the shift from probabilistic to guaranteed coverage, like that from a partially compiled score to a fully compiled search loop, was deployed only after explicit verification of the model's numerical fidelity and of the search quality obtained. This vigilance extended to the measurement infrastructure itself: the logging mechanism, never questioned since the first version, turned out to be the true bottleneck after the numba acceleration, and fixing it required a safety net (section 8.6) that was itself tested and partly rejected before arriving at a sustainable solution.

A high numerical score is not proof of linguistic coherence (section 9.5), and a second detection criterion,

based on lexical coverage rather than on the score, proved subject to the same calibration requirement as the first (section 10.3). Chance can moreover produce misleading lexical coherence at scale (section 12.3): an isolated result's apparent readability, however striking on direct reading, proves nothing by itself.

An observed structural asymmetry is not, by default, a signal. The difference in success rate noted between transcription hypotheses turned out to be confounded by the self-reinforcement bias of the adaptive sampling mechanism itself, motivating the instrumentation of phase tagging designed to settle the question on unbiased data (section 13). This tagging itself reached its limits: the volume of unbiased data it produced remained well below what was needed, and the attempt to widen the filter revealed contamination more severe than the original problem. Recognizing that a control mechanism is insufficient proved preferable to forcing conclusions out of it, leading to closing this thread for lack of data rather than publishing an unfounded result (section 13.4).

Finally, validating a new search mechanism must focus on its ability to converge, not merely on the correctness of its computation: the Four-square cipher's cryptographic mechanics were validated independently, yet standard simulated annealing systematically failed to recover a known key, until a dedicated tuning effort (section 14.5). A larger compute budget does not, moreover, necessarily improve a search mechanism: two acceleration strategies were tested then rejected after measurement, both less effective than simply multiplying independent random starts.

This document retraces the entire research process, from the first, invalidated version of the solver (section 4) to the project's closure. Successive revisions added the execution changes of section 8, the repair of the null baseline and its updated verdict (section 9), the introduction of a second lexical detection axis, the redesign of the refinement phase, the hunt for coherent fragments, and the second-order structural analysis (sections 10 through 13), the closure for lack of data of this last analysis (section 13.4), then the implementation, validation, reliability tuning, and monitoring of the Four-square cipher up to the search's halt (section 14). This version constitutes the project's final state; the code, data, and underlying raw results remain available for any future resumption (see References).

References

1. D'Agapeyeff, A. (1939). *Codes and Ciphers: A History of Cryptography*. Oxford University Press. https://archive.org/details/codesciphers0000daga

2. Wikipedia contributors. D'Agapeyeff cipher. *Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/wiki/D'Agapeyeff_cipher

3. Pelling, N. (2017). A new clue for the d'Agapeyeff Challenge Cipher...? *Cipher Mysteries*. https://ciphermysteries.com/2017/03/05/new-clue-dagapeyeff-challenge-cipher

4. Marland, T. (2026). D'Agapeyeff Research: computational cryptanalysis project. https://dagapeyeffresearch.com

5. Hyde & Rugg. (2013). A very British mystery, part 2: The D'Agapeyeff Cipher, and the first edition. https://hydeandrugg.wordpress.com/2013/07/17/

6. Gariazzo, R. (2026). Data and code: computational cryptanalysis of the D'Agapeyeff cryptogram (1939) [data set]. Zenodo. https://doi.org/10.5281/zenodo.21970478