VigenereSolver-ng~ How it works

VigenereSolver-ng

VigenereSolver-ng is an advanced toolkit for analyzing and breaking Vigenère ciphers, designed for cryptanalysis research and educational purposes. It combines classical statistical attacks with modern, language-model-based techniques to robustly estimate key length and recover the key, even on challenging ciphertexts.

Table of Contents

Key features and novel techniques

- **Language Model Integration:** Uses high-order n-gram language models (up to 5-gram) for scoring candidate plaintexts and keys, providing much greater accuracy than traditional frequency analysis.
- **Windowed Coincidence Periodogram:** Introduces a windowed version of the coincidence periodogram, which computes coincidence rates over sliding windows to localize and stabilize key-length signals, especially on heterogeneous or short texts.
- **Key-Length Voting and Non-Maximum Suppression:** Implements a voting mechanism across windows and applies non-maximum suppression to robustly select likely key lengths, reducing false positives from harmonics and noise.
- **Jensen-Shannon Divergence Scoring:** Uses JS divergence between observed and English letter distributions to weight key character votes, improving key recovery in the presence of uneven letter frequencies.
- **Kasiski Examination with Factor Analysis:** Augments classical Kasiski examination by aggregating factors of repeated-sequence spacings, then ranks candidate key lengths by their frequency as divisors.

- **Plaintext Generator for Testing:** Includes a generator for English-like plaintexts using real language data, enabling realistic benchmarking of attacks.

These innovations make VigenereSolver-ng more effective and reliable than standard Vigenère solvers, especially on real-world ciphertexts with non-uniform content or formatting.

How to use

> Requires Python 3.11+ and the project's `requirements.txt`.

Solve ciphertexts (the usual thing)

1. **Install & activate env**

```
python -m venv .venv
source .venv/bin/activate  # Windows: .venv\Scripts\activate
pip install -r requirements.txt
```

2. **Put your ciphertext in a file** using triple quotes (you can have multiple blocks):

```
"""
PXWZB ... ZQL
"""

"""
ANOTHER CIPHERTEXT BLOCK...
"""
```

3. **Run the solver**

```
python SolverSite/solver.py --input ciphertexts/tests.txt --passes 6 --decoder lm
```

- The solver auto-tunes the window/step, tests several key lengths in parallel, and prints the best key with IoC and score.
- Output includes a **readable plaintext** (with optional word segmentation when the original had no spaces).

Speed tip: Add `--workers <N>` to control parallelism across candidate key lengths (defaults to CPU count).

Determinism tip: For reproducible generation experiments, use `--seed <int>`; solving itself is mostly deterministic aside from small randomization in sweeps.

Generate test data

Create realistic test ciphertexts (plus sidecar keys JSON):

```
python SolverSite/solver.py --generate 5 --words 200 --out generated_ciphertexts.txt
# → ciphertexts in triple-quoted blocks
# → keys in generated_ciphertexts.txt.keys.json
```

Tweak key-length range with `--min-key` / `--max-key`.

Encrypt a plaintext file

Turn a plaintext into Vigenère ciphertext while **preserving original layout** (spacing, punctuation, case):

```
python SolverSite/solver.py --encrypt-file raw_text/kafka.txt --key SECRET --out ciphertexts/kafka_ct.txt
# If --key is omitted, a random key length [3..50] is chosen.
```

Practical tuning knobs

- **Decoder:** `--decoder lm` (default, KN 3–5-gram LM) or `--decoder legacy` (χ^2/JSD/ngram blend).
- **Optimization budget:** `--passes 4..8` — more passes = more key refinement (diminishing returns beyond ~6–8).
- **Auto window/step:** on by default; to **fix** them:
  `--no-auto-ws --window 600 --step 150`
- **Annealing (escape plateaus):** `--anneal 0.05` (small positive) occasionally accepts worse local moves to avoid shallow minima.
- **LM blend weight:** `--lm-weight 0.65` mixes LM NLL with legacy fitness for final ranking.
- **Segmentation off:** if you prefer raw, unsegmented output: `--no-seg`.

Troubleshooting quick refs

- **“No ciphertext found”** → Ensure blocks are wrapped in `""" ... """` or pass raw text as a single block.
- **Slow on huge inputs** → Lower `--passes`, use `--workers`, or temporarily `--no-auto-ws`.
- **Weird characters** → Save files as UTF-8. Only A–Z are analyzed; other chars are preserved in-place when formatting.

---

1) Model of the cipher and the text

Let the plaintext be a sequence of random variables $P_1, P_2, \ldots, P_N$ taking values in $\mathcal{A} = \{0, \ldots, 25\}$ (A=0,…,Z=25).
A fixed Vigenère key $K = (K_0, \ldots, K_{m-1}) \in \mathcal{A}^m$ produces ciphertext

$$C_i \equiv P_i + K_{i \bmod m} \pmod{26} \qquad (1 \le i \le N).$$

We assume the letter process $\{P_i\}$ is stationary and ergodic with marginal $p(a) = \mathbb{P}\{P_i = a\}$ and higher-order statistics captured by an n-gram distribution (language model).

All non-alphabetic characters of the original message are carried through as fixed tokens; only the projected A–Z stream is transformed algebraically. This projection/reinjection step is a bijection between $\mathcal{A}^N$ and the subset of strings with the same tokenization metadata, hence layout is exactly preserved.

---

2) Coincidence, the periodogram, and key-length detection

2.1 Coincidence probability at lag ℓ

Define the coincidence rate at lag ℓ:

$$R_\ell \triangleq \frac{1}{N-\ell}\sum_{i=1}^{N-\ell}\mathbf{1}\{C_i = C_{i+\ell}\}.$$

Under (1),

$$C_i = C_{i+\ell} \iff P_i + K_{i \bmod m} \equiv P_{i+\ell} + K_{(i+\ell) \bmod m} \pmod{26}.$$

- If $\ell \equiv 0 \pmod{m}$, the key offsets cancel, so

$$\mathbb{P}\{C_i = C_{i+\ell}\} = \mathbb{P}\{P_i = P_{i+\ell}\}.$$

Under an i.i.d. approximation $P_i \sim p$, the RHS equals the **index of coincidence**

$$I_c(p) \triangleq \sum_{a\in\mathcal{A}} p(a)^2.$$

For English, $I_c(p) \approx 0.066$.

- If $\ell \not\equiv 0 \pmod{m}$, let $d \equiv K_{i \bmod m} - K_{(i+\ell) \bmod m} \not\equiv 0$. Then

$$C_i = C_{i+\ell} \iff P_{i+\ell} \equiv P_i + d \pmod{26},$$

so (under i.i.d.)

$$\mathbb{P}\{C_i = C_{i+\ell}\} = \sum_a p(a)\,p(a+d) \;\triangleq\; S_d(p).$$

For typical language p, $S_d(p)$ is close to the random baseline $1/26 \approx 0.0385$ and strictly less than $I_c(p)$ unless p is uniform or pathologically symmetric.

Separation. For every ℓ with $\ell \equiv 0 \pmod{m}$,

$$\mathbb{E}[R_\ell] = I_c(p), \qquad \text{whereas for } \ell \not\equiv 0 \pmod{m}, \quad \mathbb{E}[R_\ell] = S_d(p) < I_c(p).$$

By the strong law of large numbers (SLLN), $R_\ell \to \mathbb{E}[R_\ell]$ almost surely as $N \to \infty$. Hence the periodogram $\ell \mapsto R_\ell$ exhibits peaks at $\ell \in m\mathbb{Z}_+$.

2.2 Windowed periodogram and variance reduction

Let $\{(s_j, s_j + w - 1)\}$ be sliding windows of length w and stride t. Define the windowed coincidence:

$$R_\ell^{(j)} \triangleq \frac{1}{w-\ell} \sum_{i=s_j}^{s_j+w-\ell-1} \mathbf{1}\{C_i = C_{i+\ell}\}.$$

The **window-averaged** periodogram is

$$\overline{R}_\ell \triangleq \frac{1}{J} \sum_{j=1}^{J} R_\ell^{(j)}.$$

Assuming stationarity within windows, $\mathbb{E}[\overline{R}_\ell] = \mathbb{E}[R_\ell]$, and

$$\mathrm{Var}(\overline{R}_\ell) = \frac{1}{J^2} \sum_{j=1}^{J} \mathrm{Var}\big(R_\ell^{(j)}\big) + \frac{2}{J^2} \sum_{j<k} \mathrm{Cov}\big(R_\ell^{(j)}, R_\ell^{(k)}\big).$$

For modest overlap (stride t not too small), the covariance terms are bounded and $\mathrm{Var}(\overline{R}_\ell) = O(1/J)$. Thus window averaging lowers estimator variance and stabilizes peak detection under topic/style drift.

2.3 Peakiness and stability objective for w, t

Let $\Phi(\overline{R})$ denote a peakiness statistic, e.g.

$$\Phi(\overline{R}) \triangleq \frac{\frac{1}{k}\sum_{r=1}^{k} \overline{R}_{\ell_{(r)}} - \mathrm{median}_\ell \overline{R}_\ell}{\sqrt{\mathrm{Var}_\ell(\overline{R}_\ell) + \varepsilon}},$$

where $\ell_{(1)}, \ldots, \ell_{(k)}$ are the top k lags.

Partition the ciphertext into thirds and compute the top-k lag sets T_1, T_2, T_3. Define stability

$$\Psi \triangleq \frac{1}{3} \sum_{1 \le u < v \le 3} \frac{|T_u \cap T_v|}{|T_u \cup T_v|}.$$

Add a weak prior Υ encouraging $\arg\max_\ell \overline{R}_\ell$ to be near a multiple of the Friedman guess $\tilde{m}$, e.g.

$$\Upsilon \triangleq \frac{1}{1 + \min_{k \in \mathbb{N}} |\hat{\ell} - k\tilde{m}|}, \qquad \hat{\ell} \in \arg\max_{\ell} \overline{R}_\ell,$$

and a utility term U penalizing degenerate windows (too short/long for N). The tuning objective is

$$J(w,t) \triangleq \alpha\,\Phi(\overline{R}) + \beta\,\Psi + \gamma\,\Upsilon + \eta\,U, \qquad \alpha, \beta, \gamma, \eta > 0.$$

Maximizing $J(w,t)$ increases the signal-to-noise of the key-period peaks and their reproducibility across segments. A tie-break among top (w,t) uses the language-model score defined in §4 to prefer pairs that make subsequent decryption “look” more like the target language.

---

3) Friedman estimate and candidate key lengths

Let $I_c(\mathrm{obs})$ be the observed IoC of the cleaned ciphertext and let $I_r = 1/26$ be the random baseline. With $I_e = I_c(p)$ the English IoC, the (classical) Friedman estimate is

$$\widehat{m}_{\mathrm{Friedman}} \approx \frac{I_e - I_r}{I_c(\mathrm{obs}) - I_r}.$$

Rounding and taking the neighborhood $\{\widehat{m}_{\mathrm{F}}-2,\dots,\widehat{m}_{\mathrm{F}}+2\}$, then uniting with the top lags of $\overline{R}_\ell$ and the small-factor set from Kasiski's test yields a finite candidate set $\mathcal{M}\subset\{2,\dots,50\}$.
By §2.1–2.2 and LLN, if N is large enough, the true m appears among the top lags with probability $\to 1$, and thus in $\mathcal{M}$.

---

4) Initial key by coset correlation (per m)

Fix $m\in\mathcal{M}$. Split the cleaned ciphertext into cosets

$$C^{(r)} \triangleq (C_r, C_{r+m}, C_{r+2m},\dots), \qquad r=0,\dots,m-1.$$

Within a sliding window W of length w, let $\hat{f}^{(r)}\in\Delta^{25}$ be the empirical histogram of $C^{(r)}$ letters (as $\mathcal{A}$ values). For a Caesar shift $s\in\mathcal{A}$, define the shifted histogram $\hat{f}^{(r)}_{-s}$ by $(\hat{f}^{(r)}_{-s})_a=\hat{f}^{(r)}_{a+s}$. With a reference English distribution $q\in\Delta^{25}$, consider

$$s^\star \in \arg\max_{s\in\mathcal{A}}\ \langle \hat{f}^{(r)}_{-s},\, q\rangle \;=\; \arg\min_{s\in\mathcal{A}}\ H(\hat{f}^{(r)}_{-s},\, q),$$

where $H(p,q)=-\sum_a p(a)\log q(a)$ is cross-entropy.

Claim (MLE for a Caesar coset). If the plaintext letters in coset r are i.i.d. $\sim q$, then $s^\star$ maximizes the log-likelihood of the observed coset under a Caesar model.

Proof. The log-likelihood for shift s is $\sum_a n^{(r)}_a \log q(a-s)$, which equals $n\,\langle \hat{f}^{(r)}_{-s},\,\log q\rangle$. Maximizing this is equivalent to minimizing $H(\hat{f}^{(r)}_{-s},q)$. $\square$

Windows whose letter distribution deviates from English are down-weighted using a divergence D (here Jensen–Shannon): with weights $w_W\propto 1/D(\hat{f}_W,q)$, the final per-coset shift is the weighted mode over windows. This reduces variance by emphasizing windows closer to the stationary regime.

---

5) Language model and the decryption objective

5.1 Interpolated Kneser–Ney (KN) probabilities

Let $c_n(\cdot)$ be counts for n-grams over $\mathcal{A}$, and $\mathrm{cont}_n(h)$ the number of unique continuations of a context h (order $n-1$). With absolute discount $D \in (0,1)$, the KN conditional probability for a next symbol w given context h of length $n-1$ is

$$p_{\mathrm{KN}}(w \mid h) = \frac{\max\{c_n(hw) - D, 0\}}{c_{n-1}(h\cdot)} + \lambda(h)\, p_{\mathrm{KN}}(w \mid h'), \qquad \lambda(h) \triangleq \frac{D\,\mathrm{cont}_n(h)}{c_{n-1}(h\cdot)},$$

with backoff to h' the suffix of h. For the base case $n=1$, the continuation probability can be taken as

$$p_{\mathrm{cont}}(w) \triangleq \frac{N_{1+}(*,w)}{N_{1+}(*,*)}, \qquad N_{1+}(*,w) = \big|\{\, x \in \mathcal{A} :\ c_2(xw) > 0 \,\}\big|, \quad N_{1+}(*,*) = \big|\{\,(x,y) \in \mathcal{A}^2 :\ c_2(xy) > 0\,\}\big|.$$

5.2 Per-character negative log-likelihood (NLL)

For a cleaned candidate plaintext $x_1^N \in \mathcal{A}^N$ and maximum order $n_{\max} = 5$,

$$\mathcal{L}(x_1^N) \triangleq \frac{1}{N} \sum_{i=1}^{N} -\log p_{\mathrm{KN}}\big(x_i \,\big|\, x_{i-k}^{i-1}\big), \qquad k = \min\{n_{\max} - 1,\ i-1\}.$$

Given a key $K \in \mathcal{A}^m$, define the decryption mapping $\mathrm{Dec}_K(C)_i \equiv C_i - K_{i \bmod m} \pmod{26}$ and the **LM objective**

$$F(K) \triangleq \mathcal{L}\big(\mathrm{Dec}_K(C)\big).$$

5.3 Legacy fitness and blended score

Let $\mathrm{fit}(x_1^N)$ be a convex combination of reduced χ^2, Jensen–Shannon divergence, and n-gram surprisals (3–4 grams) on x_1^N. The **selection score** is

$$S(K) \triangleq w\, F(K) + (1-w)\, \mathrm{fit}\big(\mathrm{Dec}_K(C)\big), \qquad w \in (0,1].$$

---

6) Coordinate-wise optimization over the key

6.1 Decomposition of local influence

Let $m = |K|$. For a residue class (coset) $r \in \{0, \dots, m-1\}$, write $\mathcal{I}_r = \{\, i : i \equiv r \pmod{m} \,\}$. Consider modifying only K_r to $s \in \mathcal{A}$. Let $x^{(s)}$ be the plaintext after this modification. For a fixed $n_{\max}$, the change in F obeys

$$F\big(K_{[r\leftarrow s]}\big) - F(K) = \frac{1}{N} \sum_{i \in \mathcal{N}_r} \Big[-\log p_{\mathrm{KN}}\big(x_i^{(s)} \mid x_{i-k}^{(s)}{}^{i-1}\big) + \log p_{\mathrm{KN}}\big(x_i \mid x_{i-k}^{i-1}\big) \Big],$$

where $\mathcal{N}_r$ is the set of indices whose $n_{\max}$-gram window **touches** any position in $\mathcal{I}_r$. Thus only those local windows need to be rescored when optimizing K_r.

6.2 Monotone descent and convergence

Define one **coordinate update** at residue r by

$$K_r \leftarrow \arg\min_{s \in \mathcal{A}} F\big(K_{[r\leftarrow s]}\big).$$

(This is the `anneal=0` case.) Since the minimizer is selected exactly over a finite set,

$$F\big(K^{(t+1)}\big) \;\le\; F\big(K^{(t)}\big) \quad \text{at every step,}$$

hence $\{F(K^{(t)})\}$ is a bounded, monotonically non-increasing sequence and converges. Because there are only 26^m keys, and strict decreases can occur only finitely many times, the process terminates at a **coordinate-wise minimum** $K^\star$ (a fixed point of all residue-wise updates).

When a small simulated annealing acceptance is used, monotonicity is relaxed to escape plateaus; nonetheless the state space is finite, and with a standard cooling schedule the process converges almost surely to a local minimum.

---

7) Effectiveness guarantees

We state assumptions explicitly:

- **A1 (Language).** The true plaintext P_1^N is generated by a stationary ergodic process whose n-gram statistics are well-approximated by the KN model used to score text.
- **A2 (Key).** The Vigenère key $K \in \mathcal{A}^m$ is fixed and unknown; m is bounded (e.g., ≤ 50).
- **A3 (Non-degenerate alphabetics).** The cleaned text length $N \to \infty$ and letter frequency vector is non-uniform (true for English).

7.1 Consistency of key-length detection

Theorem 1 (Periodogram consistency). Under A1–A3, the set of top lags of $\overline{R}_\ell$ contains m and its multiples with probability $\to 1$ as $N \to \infty$. Consequently, the candidate pool $\mathcal{M}$ includes m with probability $\to 1$.

Sketch. §2.1 showed $\mathbb{E}[\overline{R}_\ell] = I_c(p)$ for $\ell \in m\mathbb{Z}_+$ and $S_d(p)$ otherwise with strict separation. By SLLN, $\overline{R}_\ell \to \mathbb{E}[\overline{R}_\ell]$ uniformly over a finite set of lags. Therefore the top lags converge to the maximizers, which include multiples of m. □

7.2 Correctness of coset shifts (initial key)

Proposition 2 (Coset MLE). Within a coset r, the shift

$$\hat{s}_r \in \arg\min_s H\big(\hat{f}^{(r)}_{-s}, q\big)$$

is the maximum-likelihood estimate of the Caesar offset assuming the coset plaintext is i.i.d. $\sim q$.

Proof. Already given in §4. □

Variance reduction. If $D(\hat{f}_W, q)$ is a proper divergence and $\mathbb{E}[D(\hat{f}_W, q)]$ increases with departure from stationarity, then weighting windows by $w_W \propto 1/D(\hat{f}_W, q)$ yields a **minimum variance unbiased** estimator among the class of linear unbiased combinations of $\hat{s}_{r,W}$ under a heteroskedastic model of window quality. (This follows from generalized least squares: inverse-variance weights are optimal; here D is used as a monotone proxy for variance.)

7.3 Optimality of the LM objective at the true key

Let P denote the distribution of true plaintext strings, Q_K the distribution induced by decrypting with key K (wrong alignment mixes cosets and introduces Caesar-shifted contexts). Let M denote the KN model used for scoring. The per-character cross-entropy under M is

$$H(P;M) \triangleq \lim_{N\to\infty} \mathbb{E}\big[\mathcal{L}(P_1^N)\big], \qquad H(Q_K;M) \triangleq \lim_{N\to\infty} \mathbb{E}\big[\mathcal{L}(\tilde{P}_1^N)\big],$$

where $\tilde{P} = \mathrm{Dec}_K(C)$.

Theorem 3 (Asymptotic separation). If $M = P$ (oracle LM), then for every $K \neq K_{\text{true}}$,

$$H(Q_K;M) - H(P;M) = \mathrm{KL}(P\|Q_K) \;>\; 0.$$

Proof. By the information inequality,

$$\mathbb{E}_P[-\log P(X)] \;\leq\; \mathbb{E}_P[-\log Q_K(X)],$$

with equality iff $Q_K = P$. But $Q_K = P$ only when $K = K_{\text{true}}$ up to inconsequential Caesar offsets that are already absorbed in K; otherwise different cosets and contexts are mismatched and the equality fails. □

Corollary 4 (Consistency with approximate LM). If M is a consistent estimator of P in the sense that $H(P;M) \to H(P;P)$ and, uniformly over wrong keys K, $H(Q_K;M) \to H(Q_K;P)$, then there exists $\delta > 0$ such that for all large N,

$$\mathbb{E}\big[F(K_{\text{true}})\big] + o(1) \;\leq\; \mathbb{E}\big[F(K)\big] - \delta \quad \text{for all } K \neq K_{\text{true}}.$$

Hence the LM part of the selection prefers the true key with probability $\to 1$.

Blended score. Because $S(K) = wF(K) + (1-w)\mathrm{fit}(\cdot)$ with $w > 0$, the separation provided by F dominates for large N, so the blended score also correctly ranks the true key with probability $\to 1$.

7.4 Convergence of the key optimizer

Proposition 5 (Finite convergence to a coordinate-wise minimum). The coordinate-wise minimization of $F(K)$ over residues $r = 0, \ldots, m-1$ with exact per-residue minimizers terminates in finitely many steps at a key $K^\star$ such that for each residue r,

$$K_r^\star \in \arg\min_{s\in\mathcal{A}} F\big(K^\star_{[r\leftarrow s]}\big).$$

Proof. Each successful update strictly decreases F by at least a positive amount (finite score space due to finite data and finite alphabet), and there are finitely many keys 26^m. Hence only finitely many strict decreases occur; the process halts when no per-residue improvement exists. □

Locality correctness. The update uses exactly the set $\mathcal{N}_r$ of windows affected by residue r (see §6.1). Therefore each residue update is an **exact** coordinate minimization of F, not a heuristic.

---

8) Auto-tuned window/step with LM tie-break

Given the objective $J(w,t)$ from §2.3, select the top few (w,t) pairs by J. For each, take the best 1–2 candidate key lengths m by $\overline{R}_\ell$, form initial keys (§4), decrypt, and measure $-F(K)$ (i.e., LM likelihood). The final choice is

$$(w^\star,t^\star) \in \arg\max_{(w,t)\in\text{beam}} \Big(\alpha'\, J(w,t) + \beta' \max_{m\in\mathcal{M}(w,t)} \max_{K\in\mathcal{K}_{\text{init}}(m)} -F(K)\Big),$$

with $\alpha',\beta' > 0$. Under Theorem 3/Corollary 4, the LM tie-break favors (w,t) that sharpen the m proposals and produce plaintext closer to P, improving the probability that the downstream key optimization starts in the basin of attraction of K_{true}.

---

9) Readability segmentation (post-processing)

When the recovered clean plaintext x_1^N lacks spaces, a dictionary-rank cost $c(w) \propto \log \operatorname{rank}(w)$ and a length penalty $\lambda|w|$ define the segmentation objective

$$\min_{x_1^N = w_1\cdots w_T} \sum_{t=1}^{T} \big(c(w_t) + \lambda|w_t|\big),$$

solved by standard dynamic programming. This does not affect correctness; it only improves human readability.

---

10) Summary of guarantees

- **Key length:** $\overline{R}_\ell$ consistently peaks at multiples of m (§2), so $m \in \mathcal{M}$ w.h.p. for large N (Theorem 1).
- **Initial key:** per-coset Caesar shifts via cross-entropy minimization are MLEs (Proposition 2); divergence-weighted voting reduces variance.
- **Key optimization:** exact coordinate updates monotonically decrease F and converge to a coordinate-wise optimum (Proposition 5); only locally affected LM windows are rescored (§6.1).
- **Selection:** if the LM matches the plaintext process, the true decryption minimizes the expected NLL; any wrong key incurs a strictly larger cross-entropy (Theorem 3), and this separation persists (Corollary 4). With $w > 0$, the blended score inherits this separation.

Under these conditions, the pipeline is **asymptotically effective**: as $N \to \infty$, it recovers the correct key length with probability $\to 1$, and among candidate keys, the LM objective (and thus the blended score) is uniquely minimized at the true key.