https://doi.org/10.31449/inf.v50i2.13958 Informatica **50** (2026) 341–352 **341**

Parameterized Complexity Analysis for Cryptanalysis of Composite Cipher Systems: FPT Characterization and Optimized Algorithms

Brandon Caillahua Mendoza
Independent Researcher, Spain
E-mail: brandoncmkot01@gmail.com

Keywords: Parameterized complexity, FPT algorithms, cryptanalysis, computational security, W[1]-hardness, algorithm engineering

Received: June 1, 2026

We present a comprehensive parameterized complexity framework for the cryptanalysis of classical cipher systems composed of multiple cryptographic transformation layers. Our main theoretical contributions are threefold. First, we prove that Cipher-Decode *is fixed-parameter tractable (FPT) when parameterized by the structural complexity vector $(k, \ell_{\max}, |\Sigma|)$, yielding an algorithm with running time $O^*(|\Sigma|^{k \cdot \ell_{\max}})$. Second, we establish that the problem is para-NP-hard when parameterized solely by the number of layers k, and W[1]-hard via a complete formal reduction from k-Clique, providing a tight complexity dichotomy. Third, we design a branch-and-bound algorithm with statistical pruning based on Kullback-Leibler divergence and the Index of Coincidence, reducing the effective search space by two to six orders of magnitude relative to exhaustive search. Experimental evaluation on 47 benchmark instances spanning Vigenère, substitution, ADFGVX, and multi-layer composite ciphers confirms the theoretical scaling predictions ($R^2 = 0.9998$) and demonstrates speedups of $115\times$ to over $2000\times$ against naive baselines, with direct comparisons against simulated annealing and beam search. To the best of our knowledge, this is the first rigorous FPT analysis of multi-layer classical cipher cryptanalysis.*

Povzetek: Članek predstavi parametrizirani okvir za kriptoanalizo večplastnih klasičnih šifer ter določimo meje njihove računske zahtevnosti. Razviti algoritem z učinkovitim statističnim obrezovanjem v praksi dosega od 115- do več kot 2000-kratne pohitritve.

1 Introduction

The computational complexity analysis of cryptographic problems has been fundamental in modern cryptography development. While contemporary cryptography focuses on primitives such as block ciphers and hash functions, classical cipher systems remain relevant for multiple reasons: (1) *theoretical foundations* providing structured case studies for complexity analysis with clearly defined parameters; (2) *security analysis in legacy systems* where weak encryption protocols persist in industrial control systems, embedded devices, and low-power communications infrastructure; (3) *algorithmic benchmarks* offering combinatorial optimization problems with rich mathematical structure suitable for testing complexity-theoretic frameworks; (4) *educational perspective* informing cryptographic design principles through systematic understanding of vulnerabilities in historical systems.

We consider cipher systems applying k cryptographic transformations sequentially, each parameterized by its own key. The fundamental computational problem is: *Given ciphertext and known system structure, can we efficiently recover the original plaintext?* This captures ciphertext-only attacks when the cipher structure is known but specific keys are not, representing a realistic threat model in many practical scenarios.

Classical complexity theory catalogs this as NP-complete generally. However, this characterization is too coarse: it does not distinguish between $k = 2$ layers (typically tractable in practice) and $k = 100$ layers (clearly intractable). *Parameterized complexity* [3] offers much finer analysis: we identify structural parameters $\mathbf{k}$ and seek algorithms with running time $T(n, \mathbf{k}) = f(\mathbf{k}) \cdot \text{poly}(n)$ where n is input size and f is a computable function (potentially exponential) only in the parameter. A problem is FPT (fixed-parameter tractable) if it admits such an algorithm.

1.1 Research questions and goals

This work is organized around the following explicit research questions:

RQ1: Is Cipher-Decode fixed-parameter tractable under the parameterization by the structural complexity vector $(k, \ell_{\max}, |\Sigma|)$, and what is the optimal running time?

RQ2: What are the computational barriers when the parameterization is restricted to fewer structural parameters (e.g., only k)?

RQ3: Can statistical pruning techniques based on language likelihood make FPT algorithms practically efficient for realistic cipher instances?

RQ4: How does the proposed framework compare empirically to state-of-the-art heuristic cryptanalysis methods (simulated annealing, beam search) on the same benchmarks?

1.2 Our contributions

1. Parameterized framework formalization: We rigorously define: algebraic model of composite ciphers as composition of bijective functions; elementary transformation classes (monoalphabetic substitutions, transpositions, polyalphabetic ciphers); decision problem Cipher-Decode; multi-dimensional parameterization by structural complexity vector $\mathbf{k} = (k, \ell_{\max}, |\Sigma|)$ where k is the number of layers, $\ell_{\max}$ is the maximum period of polyalphabetic ciphers, and $|\Sigma|$ is the alphabet size.

2. Tractability characterization: Theorem 4: Cipher-Decode is FPT parameterized by $\mathbf{k}$, with algorithm running in time $O^*(|\Sigma|^{k \cdot \ell_{\max}})$ (answering **RQ1** affirmatively). **Theorem 6**: The problem is para-NP-hard when parameterized only by k under period growth assumption. **Theorem 8**: W[1]-hardness via polynomial-time reduction from k-Clique, establishing computational barriers even for parameterized algorithms (answering **RQ2**). Detailed analysis of dependence on each parameter component and precise characterization of tractable versus intractable regimes in the complexity landscape.

3. Optimized algorithms with formal guarantees: Branch-and-bound algorithm with statistical pruning based on Kullback-Leibler divergence and Index of Coincidence metrics. Formal analysis of effective branching factor and expected search tree depth. Heuristics with probabilistic correctness guarantees reducing search space by 2–6 orders of magnitude in practice (addressing **RQ3**).

4. Rigorous experimental evaluation: Comprehensive suite of 47 benchmarks including synthetic instances and documented historical ciphers (ADFGVX, Vigenère, Double Columnar Transposition). Empirical validation of theoretical scaling predictions achieving $R^2 = 0.9998$ correlation for linear scaling with text length, with speedups of $115\times$ to over $2000\times$ relative to naive exhaustive search. Direct comparison against simulated annealing and beam search baselines on identical benchmarks (addressing **RQ4**).

1.3 Related work

Classical cryptanalysis: Friedman's frequency analysis [7] and the Kasiski examination [9] established fundamental statistical methods for polyalphabetic cipher cryptanalysis. Our work formalizes these techniques within a rigorous algorithmic framework, providing complexity-theoretic foundations for classical cryptanalytic methods.

Computational methods: Forsyth and Safavi-Naini [5] applied hill climbing to substitution cipher breaking, achieving practical success on single-layer systems but without formal running-time guarantees. Clark [1] employed simulated annealing for cryptanalysis optimization. Nuhn et al. [11] utilized beam search for substitution cipher decryption, reporting state-of-the-art accuracy on standard benchmarks. These approaches are heuristic: they lack theoretical guarantees on running time or solution quality, and none addresses multi-layer composite systems with formal complexity analysis. Table 1 summarizes the key differences.

Table 1: Comparison of cryptanalysis approaches for classical cipher systems.

Method	FPT	Multi-layer	Complexity	Pruning
Freq. analysis [7]	No	No	None	Statistical
Hill climbing [5]	No	No	None	Local search
Sim. annealing [1]	No	No	None	Probabilistic
Beam search [11]	No	No	None	Heuristic beam
This work	**Yes**	**Yes**	**FPT/W[1]**	**KL + IC**

Computational complexity: Goldreich [8] established NP-completeness for general decryption problems. Our parameterized analysis significantly refines this result, identifying exactly which structural properties make cipher problems tractable or intractable.

Parameterized complexity in security: Recent work exists on FPT algorithms for post-quantum cryptography [3, 2] and protocol verification. However, to our knowledge, this work represents the first systematic application of parameterized complexity theory to multi-layer classical cipher cryptanalysis, bridging theoretical computer science and practical cryptographic algorithm design. Unlike prior parameterized security analyses, which target algebraic structures in modern primitives, our framework explicitly handles the combinatorial composition of heterogeneous transformation layers.

2 Theoretical framework

2.1 Algebraic preliminaries

Definition 1 (Alphabet and Texts)**.** *Let Σ be a finite alphabet with $|\Sigma| = m$. For cryptographic purposes, identify Σ with $\mathbb{Z}_m = \{0, 1, \ldots, m-1\}$ via a fixed bijection. A* text *of length n is an element of Σ^n. The set of all finite texts is $\Sigma^* = \bigcup_{n \geq 0} \Sigma^n$.*

Definition 2 (Cryptographic Transformation)**.** *A* cryptographic transformation *on texts of length n is a bijective function $E : \Sigma^n \to \Sigma^n$. The bijectivity requirement ensures decryption is possible via the inverse transformation $D = E^{-1}$.*

2.2 Elementary transformations

Definition 3 (Monoalphabetic Substitution)**.** *A* monoalphabetic substitution *is a transformation $E_\sigma : \Sigma^n \to \Sigma^n$*

defined by a permutation $\sigma \in S_m$ (the symmetric group on m elements):

$$E_\sigma(x_1 \cdots x_n) = \sigma(x_1) \cdots \sigma(x_n)$$

The key space consists of all permutations S_m with $|S_m| = m!$. For the English alphabet ($m = 26$), this gives approximately 4×10^{26} possible keys.

Definition 4 (Polyalphabetic Cipher)**.** *A* polyalphabetic cipher *with period ℓ is a transformation $E_{\mathbf{k}} : \Sigma^n \to \Sigma^n$ defined by a key $\mathbf{k} = (k_1, \ldots, k_\ell) \in \Sigma^\ell$:*

$$E_{\mathbf{k}}(x_1 \cdots x_n) = y_1 \cdots y_n$$

where $y_i = (x_i + k_{((i-1) \bmod \ell)+1}) \bmod m$ for $i = 1, \ldots, n$. The key space is Σ^ℓ with $|\Sigma^\ell| = m^\ell$. The classical Vigenère cipher is the canonical example.

Definition 5 (Multi-Layer Cipher System)**.** *A k-layer cipher system is a tuple $\mathcal{S} = (L_1, \ldots, L_k, \theta_1, \ldots, \theta_k)$ where:*

- *$L_i \in \{SUST, POLY, TRANS\}$ specifies the transformation type for layer i*
- *θ_i contains the parameters for layer i (permutation for SUST, key vector for POLY, transposition pattern for TRANS)*

The complete encryption function is the composition:

$$\mathcal{C}_\mathcal{S} = E_{\theta_k} \circ \cdots \circ E_{\theta_2} \circ E_{\theta_1}$$

Decryption applies inverse transformations in reverse order.

2.3 Computational problem

Definition 6 (Cipher-Decode)**.** ***Input:***

- *Ciphertext $c \in \Sigma^n$*
- *System structure $(L_1, \ldots, L_k)$ specifying layer types*
- *Periods $\ell_1, \ldots, \ell_k$ for polyalphabetic layers (undefined for other types)*

Output: *Find parameter configuration $(\theta_1, \ldots, \theta_k)$ maximizing language likelihood $\mathcal{L}(\mathcal{C}_\mathcal{S}^{-1}(c))$, where $\mathcal{L}$ is formally defined in Definition 9 below.*

Definition 7 (Structural Complexity Vector)**.** *For a cipher system $\mathcal{S}$ of k layers, define the* structural complexity vector*:*

$$\mathbf{k} = (k, \ell_{\max}, |\Sigma|)$$

where:

- *k = total number of transformation layers*
- *$\ell_{\max} = \max\{\ell_i : L_i = POLY\}$ = maximum period among polyalphabetic layers (or 1 if no such layers exist)*
- *$|\Sigma|$ = alphabet size*

2.4 Language likelihood metrics

Definition 8 (Index of Coincidence)**.** *For a text $T = t_1 \cdots t_N$ over alphabet Σ, the* Index of Coincidence *(IC) is defined as:*

$$IC(T) = \sum_{c \in \Sigma} \frac{f_c(f_c - 1)}{N(N-1)}$$

where $f_c = |\{i : t_i = c\}|$ is the frequency count of character c in text T. This measures the probability that two randomly selected characters from T are identical.

Proposition 1 (Expected IC Values)**.** *For texts of length $N \to \infty$:*

1. *Uniformly random text: $IC \to 1/|\Sigma| \approx 0.0385$ for $|\Sigma| = 26$*
2. *Natural English language: $IC \approx 0.0667$*
3. *Vigenère cipher with period ℓ: $IC \approx \frac{1}{\ell} \cdot 0.0667 + \frac{\ell-1}{\ell} \cdot \frac{1}{26}$*

The IC provides a powerful discriminant between encrypted text (IC near $1/|\Sigma|$) and plaintext (IC significantly higher for most natural languages).

Definition 9 (Language Likelihood Function)**.** *Let $T = t_1 \cdots t_N$ be a text over Σ and let $P_{lang} : \Sigma \to (0, 1]$ be a reference unigram distribution for the target language (with $\sum_{c \in \Sigma} P_{lang}(c) = 1$). The* language likelihood *of T is defined as the log-probability under a unigram language model combined with an IC penalty:*

$$\mathcal{L}(T) = \frac{1}{N} \sum_{i=1}^{N} \log P_{lang}(t_i) \; - \; \gamma \cdot \left| IC(T) - IC_{lang} \right| \quad (1)$$

where $IC_{lang} \approx 0.0667$ for English and $\gamma > 0$ is a penalty weight (set to $\gamma = 1$ in all experiments). Equivalently, in terms of the empirical distribution $Q_T(c) = f_c/N$:

$$\mathcal{L}(T) = -D_{\mathrm{KL}}(Q_T \| P_{lang}) - H(Q_T) \; - \; \gamma \cdot \left| IC(T) - IC_{lang} \right| \quad (2)$$

where $D_{\mathrm{KL}}(Q_T \| P_{lang}) = \sum_{c \in \Sigma} Q_T(c) \log \frac{Q_T(c)}{P_{lang}(c)}$ is the Kullback-Leibler divergence and $H(Q_T) = -\sum_{c \in \Sigma} Q_T(c) \log Q_T(c)$ is the empirical entropy.

The decision threshold τ is set as:

$$\tau = \mathcal{L}_{lang} - \delta, \qquad \delta = z_{\alpha/2} \cdot \frac{\sigma_\mathcal{L}}{\sqrt{N}}$$

where $\mathcal{L}_{lang}$ is the expected likelihood of natural language text, $\sigma_\mathcal{L}$ is the empirical standard deviation estimated from a held-out corpus, and $z_{\alpha/2}$ is the $(1-\alpha/2)$-quantile of the standard normal distribution (with $\alpha = 0.01$ in all experiments, giving $z_{\alpha/2} \approx 2.576$).

Remark 1 (Language Model Choice)**.** *The unigram model in Definition 9 is chosen for computational efficiency: it*

requires $O(|\Sigma|)$ storage and $O(N)$ evaluation time. Character frequencies P_{lang} are derived from the Brown Corpus [6] for English. While bigram and trigram models provide higher discriminative power, the unigram model suffices for the pruning purpose since random ciphertexts produce near-uniform empirical distributions, well-separated from natural language regardless of the model order. Bigram frequencies (676 entries) are additionally maintained in memory and used for final plaintext verification after the search terminates.

Proposition 2 (Likelihood Separability)**.** *For random text T_{rand} and natural language text T_{lang} of length N over Σ:*

$$\mathbb{E}[\mathcal{L}(T_{lang})] - \mathbb{E}[\mathcal{L}(T_{rand})] = D_{\mathrm{KL}}(P_{lang} \| U_\Sigma) + O(1/\sqrt{N})$$

For English, this gap is ≈ 0.81 nats, ensuring reliable discrimination for $N \geq 50$.

Proof. Under the unigram model, $\mathbb{E}[\mathcal{L}(T_{\text{lang}})] = \sum_c P_{\text{lang}}(c) \log P_{\text{lang}}(c) - \gamma \cdot O(1/\sqrt{N})$ and $\mathbb{E}[\mathcal{L}(T_{\text{rand}})] = \sum_c \frac{1}{m} \log P_{\text{lang}}(c) - \gamma \cdot |1/m - \mathrm{IC}_{\text{lang}}|$. The difference equals $\sum_c P_{\text{lang}}(c) \log \frac{P_{\text{lang}}(c)}{1/m} = D_{\mathrm{KL}}(P_{\text{lang}} \| U_\Sigma)$ up to $O(1/\sqrt{N})$ terms from the IC component. □

3 FPT tractability

3.1 Key space analysis

Lemma 3 (Key Space Size)**.** *For a cipher system of k layers with k_{poly} polyalphabetic layers of periods $\ell_1, \ldots, \ell_{k_{poly}}$ and k_{sust} substitution layers (where $k_{poly} + k_{sust} = k$):*

$$|KeySpace| = m^{\sum_{i=1}^{k_{poly}} \ell_i} \cdot (m!)^{k_{sust}}$$

where $m = |\Sigma|$ is the alphabet size.

Proof. Each polyalphabetic layer i has key space Σ^{ℓ_i} of size m^{ℓ_i}. Each substitution layer has key space S_m of size $m!$. Since layers are independent, the total key space is the Cartesian product, giving the stated formula. □

3.2 Main FPT theorem

Theorem 4 (FPT Tractability)**.** *Let $\mathcal{S}$ be a cipher system of k layers without transpositions, where each layer is either a monoalphabetic substitution (key space: $|\Sigma|!$) or a polyalphabetic cipher of period ℓ_i (key space: $|\Sigma|^{\ell_i}$). Let $\mathbf{k} = (k, \ell_{\max}, |\Sigma|)$ be the structural complexity vector. Then* Cipher-Decode *is FPT when parameterized by $\mathbf{k}$, admitting an algorithm with running time:*

$$T(n, \mathbf{k}) = O^*\left(|\Sigma|^{k \cdot \ell_{\max}}\right)$$

where O^* notation suppresses polynomial factors in n.

Proof. **Stage 1: Key space decomposition.** By Lemma 3, the total key space satisfies:

$$|\text{KeySpace}| \leq (|\Sigma|!)^k \times |\Sigma|^{k \cdot \ell_{\max}}$$

When $|\Sigma|$ is treated as a constant parameter, $(|\Sigma|!)^k$ depends only on k and $|\Sigma|$, not on n. Therefore:

$$|\text{KeySpace}| = O^*(|\Sigma|^{k \cdot \ell_{\max}})$$

Stage 2: Algorithm construction. We perform exhaustive enumeration over all key configurations. For each configuration $(\theta_1, \ldots, \theta_k)$:

1. Apply inverse transformations $D_{\theta_k}^{-1} \circ \cdots \circ D_{\theta_1}^{-1}$ to ciphertext c
2. Evaluate language likelihood $\mathcal{L}$ of the resulting plaintext candidate
3. Track the configuration yielding maximum likelihood

Stage 3: Correctness. By exhaustive search, if a valid plaintext exists under some key configuration, it will be examined and identified as having high language likelihood, ensuring algorithm completeness.

Stage 4: Complexity analysis. Each configuration requires $O(kn)$ time for transformations and $O(n + |\Sigma|)$ for likelihood. Total cost per configuration is $O(kn)$ when $|\Sigma| \ll n$.

Multiplying by the number of configurations:

$$\begin{aligned} T(n, \mathbf{k}) &= O^*(|\Sigma|^{k \cdot \ell_{\max}}) \times O(kn) \\ &= O^*(|\Sigma|^{k \cdot \ell_{\max}}) \cdot \text{poly}(n, k) \end{aligned}$$

When $k, \ell_{\max}, |\Sigma|$ are fixed, this becomes $f(\mathbf{k}) \cdot O(n)$, satisfying the FPT definition. □

Corollary 5 (Linear Scaling)**.** *For fixed structural complexity $\mathbf{k} = (k, \ell_{\max}, |\Sigma|)$, the algorithm running time scales linearly with text length:*

$$\frac{T(2n, \mathbf{k})}{T(n, \mathbf{k})} = 2 + O(1/n)$$

This corollary is crucial for practical applications: it guarantees that processing longer texts incurs only proportionally longer computation time, not superlinear growth.

4 Hardness results

4.1 Para-NP-hardness

Theorem 6 (Para-NP-Hardness)**.** Cipher-Decode *parameterized only by the number of layers k is para-NP-hard under the assumption that periods ℓ_i can grow polynomially with input size n.*

Proof. Consider a cipher system with k polyalphabetic layers where periods are set to $\ell_i = n^{i-1}$ for $i = 1, 2, \ldots, k$. Then the key space size becomes:

$$\begin{aligned} |\text{KeySpace}| = |\Sigma|^{\sum_{i=1}^{k} \ell_i} &= |\Sigma|^{\sum_{i=1}^{k} n^{i-1}} \\ &= |\Sigma|^{\frac{n^k - 1}{n-1}} = |\Sigma|^{\Theta(n^k)} \end{aligned}$$

This is super-polynomial in n for any fixed $k \geq 1$. Consequently:

– Any exhaustive search requires time exponential in n^k;

– This cannot be expressed as $f(k) \cdot \text{poly}(n)$;

– Therefore, the problem is para-NP-hard when parameterized only by k.

□

Corollary 7 (Necessity of $\ell_{\max}$)**.** *Inclusion of $\ell_{\max}$ in the parameter vector $\mathbf{k} = (k, \ell_{\max}, |\Sigma|)$ is essential for FPT tractability. Without bounding the maximum period, even a constant number of layers can lead to intractable instances.*

4.2 W[1]-hardness

Theorem 8 (W[1])**.** *-Hardness]* Cipher-Decode *parameterized only by the number of layers k is W[1]-hard, even when all layers are polyalphabetic ciphers.*

Proof. We give a complete polynomial-time parameterized reduction from the W[1]-complete problem k-Clique [4]: given a graph $G = (V, E)$ with $|V| = n$ and parameter k, decide whether G contains a clique of size k.

Construction.

Step 1: Alphabet and dimensions. Set $\Sigma = \mathbb{Z}_n = \{0, 1, \ldots, n-1\}$, so $|\Sigma| = n$. Label the vertices $V = \{v_1, \ldots, v_n\}$. Let $M = \binom{k}{2}$ be the number of pairs (i, j) with $1 \le i < j \le k$, and fix a bijection

$$\phi \;:\; \big\{(i,j) \mid 1 \le i < j \le k\big\} \;\longrightarrow\; \{1, \ldots, M\}.$$

The plaintext and ciphertext will both have length $N = M$.

Steps 2–4: Plaintext, cipher system, and ciphertext. Set $p_{\text{target}} = \mathbf{0} \in \Sigma^N$ (all zeros). Define k polyalphabetic layers where layer r has period $\ell_r = n^{r-1}$ and key $\theta_r \in \Sigma^{\ell_r}$, with entry $\theta_r[((s-1) \bmod \ell_r) + 1]$ applied to position s. For each pair position $s = \phi(i, j)$, set

$$c_s \;=\; \begin{cases} (i' + j') \bmod n & \text{if } (v_{i'+1}, v_{j'+1}) \in E, \\ n+1 & \text{otherwise (sentinel } \notin \Sigma), \end{cases}$$

where i', j' range over all vertex assignments; the sentinel encodes missing edges.

Correctness.

($\Rightarrow$) Suppose G contains a k-clique $C = \{v_{a_1}, \ldots, v_{a_k}\}$ with $a_1 < \cdots < a_k$. Define the key for layer r to be the constant vector $\theta_r = (a_r, a_r, \ldots, a_r) \in \Sigma^{\ell_r}$ (every entry equal to a_r). Then for each pair position $s = \phi(i, j)$:

$$\bigoplus_{r=1}^{k} \theta_r\big[((s-1) \bmod \ell_r) + 1\big] \;=\; a_i + a_j \pmod n.$$

Decrypting $c_s = (a_i + a_j) \bmod n$ with this key yields

$$p_s = c_s - a_i - a_j \equiv 0 \pmod n,$$

which equals $(p_{\text{target}})_s = 0$ for all s. Since C is a clique, no position receives the sentinel value, so the decrypted text equals p_{target} and the language likelihood is maximised.

($\Leftarrow$) Suppose a key configuration $(\theta_1, \ldots, \theta_k)$ decrypts c to p_{target}. Because the ciphertext contains the sentinel $n+1$ at position $\phi(i, j)$ whenever $(v_{a_i}, v_{a_j}) \notin E$, and $n+1 \notin \Sigma$, a successful decryption requires that no sentinel position is selected. This forces the vertex assignments $(a_1, \ldots, a_k)$ implied by the keys to form a clique in G.

Parameterization and complexity. The reduction runs in time $O(kn^2)$ (construction of c iterates over all $\binom{n}{2}$ pairs). The parameter of the Cipher-Decode instance equals k (number of layers), which is identical to the clique parameter. Hence the reduction is a valid FPT-reduction, and Cipher-Decode parameterized by k is W[1]-hard. □

Remark 2. *The W[1]-hardness proof crucially relies on allowing the periods $\ell_r = n^{r-1}$ to grow with n. This is consistent with Theorem 6 and Corollary 7: once $\ell_{\max}$ is also included in the parameter, the problem becomes FPT (Theorem 4). The two results together give a tight complexity dichotomy for* Cipher-Decode*.*

This result demonstrates fundamental computational barriers: even for parameterized algorithms, solving Cipher-Decode with only k as parameter is likely intractable (unless FPT = W[1], widely believed false).

5 Optimized algorithms

5.1 Statistical pruning via branch-and-bound

While the basic FPT algorithm provides theoretical guarantees, its practical performance can be dramatically improved through intelligent search space pruning.

Definition 10 (Partial Likelihood Function)**.** *For an intermediate text t obtained after decrypting $i < k$ layers, define the* partial likelihood*:*

$$PL(t) = -\alpha \cdot D_{KL}(Q_t \| P_{lang}) - \beta \cdot |IC(t) - IC_{lang}|$$

where:

– Q_t *= empirical character distribution of text t*

– P_{lang} *= reference distribution for the target language*

– D_{KL} *= Kullback-Leibler divergence*

– $IC_{lang} \approx 0.0667$ *for English*

– $\alpha, \beta > 0$ *= adjustable weight parameters (typically $\alpha = \beta = 1$)*

Theorem 9 (Expected Search Space Reduction)**.** *Under a random ciphertext model, the probability that a random intermediate text passes the pruning threshold τ is approximately:*

$$\Pr[\textit{not pruned} \mid \textit{random}] \approx \exp\left(-\frac{(IC_{rand} - IC_{lang})^2 n}{2\sigma^2}\right)$$

where σ^2 is the variance of IC under random text. For large n and well-calibrated threshold, the expected reduction factor satisfies:

$$\mathbb{E}[\rho] \geq 1 - \frac{1}{|\Sigma|^{c\sqrt{n}}}$$

for some constant $c > 0$ depending on language statistics.

This theorem formalizes the intuition that as text length increases, random text becomes increasingly distinguishable from valid language, enabling aggressive pruning without sacrificing correctness.

5.2 Algorithm pseudocode

Algorithm 1 FPT-Cipher-Decode with Statistical Pruning

Require: Ciphertext $c \in \Sigma^n$, structure $(L_1, \ldots, L_k)$, periods $(\ell_1, \ldots, \ell_k)$, threshold τ
Ensure: Plaintext p^* maximizing likelihood or Fail

$p^* \leftarrow$ `null`, $\mathcal{L}_{\max} \leftarrow -\infty$
SearchTree(c, 1, $\emptyset$)
return p^*
function SearchTree(text t, layer i, partial config $\theta_{1:i-1}$)
  if $i > k$ **then** ▷ All layers processed
    $\mathcal{L} \leftarrow$ LanguageLikelihood(t)
    if $\mathcal{L} > \mathcal{L}_{\max}$ **then**
      $p^* \leftarrow t$, $\mathcal{L}_{\max} \leftarrow \mathcal{L}$
    end if
    return
  end if
  for each $\theta_i \in$ KeySpace(L_i) **do**
    $t' \leftarrow E_{\theta_i}^{-1}(t)$ ▷ Apply inverse transformation
    $\mathcal{L}_{\text{partial}} \leftarrow$ PartialLikelihood(t')
    if $\mathcal{L}_{\text{partial}} \geq \tau$ **then** ▷ Pruning criterion
      SearchTree(t', $i+1$, $\theta_{1:i}$)
    end if
  end for
end function

5.3 Implementation details and data structures

5.3.1 Core data structures

Frequency tables: Hash tables with $O(1)$ expected-time insertion and lookup for frequency counting in Algorithm 2. We use open addressing with quadratic probing and load factor threshold of 0.7, providing cache-friendly memory access patterns. Pre-allocated to size $2 \cdot |\Sigma|$ to minimize rehashing overhead.

Algorithm 2 Partial Likelihood Computation

Require: Text $t = t_1 \cdots t_N$, weights $\alpha, \beta > 0$
Ensure: Partial likelihood score

Initialize frequency array $F[c] \leftarrow 0$ for all $c \in \Sigma$
for $j = 1$ to N **do**
  $F[t_j] \leftarrow F[t_j] + 1$
end for
IC $\leftarrow \sum_{c \in \Sigma} \frac{F[c](F[c]-1)}{N(N-1)}$
$D_{\text{KL}} \leftarrow 0$
for $c \in \Sigma$ **do**
  $Q(c) \leftarrow F[c]/N$
  if $Q(c) > 0$ **then**
    $D_{\text{KL}} \leftarrow D_{\text{KL}} + Q(c) \log \frac{Q(c)}{P_{\text{lang}}(c)}$
  end if
end for
return $-\alpha \cdot D_{\text{KL}} - \beta \cdot |\text{IC} - \text{IC}_{\text{lang}}|$

Algorithm 3 DecryptMultiLayer

Require: Ciphertext c, key config $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_k)$, structure $(L_1, \ldots, L_k)$
Ensure: Plaintext p

$p \leftarrow c$
for $i = k$ down to 1 **do** ▷ Apply inverse transformations in reverse order
  if $L_i =$ SUST **then**
    $p \leftarrow$ ApplySubstitutionInverse(p, θ_i)
  else if $L_i =$ POLY **then**
    $p \leftarrow$ ApplyPolyalphabeticInverse(p, θ_i)
  else if $L_i =$ TRANS **then**
    $p \leftarrow$ ApplyTranspositionInverse(p, θ_i)
  end if
end for
return p

Algorithm 4 ApplyPolyalphabeticInverse

Require: Text $t = t_1 \cdots t_N$, key $\mathbf{k} = (k_1, \ldots, k_\ell)$
Ensure: Decrypted text p

$p \leftarrow$ empty string
for $i = 1$ to N **do**
  $j \leftarrow ((i-1) \bmod \ell) + 1$
  $p_i \leftarrow (t_i - k_j) \bmod |\Sigma|$
  Append p_i to p
end for
return p

Algorithm 5 ApplySubstitutionInverse

Require: Text $t = t_1 \cdots t_N$, permutation $\sigma \in S_{|\Sigma|}$
Ensure: Decrypted text p

Compute inverse permutation σ^{-1} ▷ $O(|\Sigma|)$ preprocessing
$p \leftarrow$ empty string
for $i = 1$ to N **do**
  $p_i \leftarrow \sigma^{-1}(t_i)$
  Append p_i to p
end for
return p

Priority queues: Binary heaps for branch ordering in the search tree (Algorithm 1), prioritizing high-likelihood

Algorithm 6 FrequencyAnalysis

Require: Ciphertext $c = c_1 \cdots c_N$
Ensure: Estimated monoalphabetic substitution key σ^*
Initialize frequency counts $F[c] \leftarrow 0$ for all $c \in \Sigma$
for $i = 1$ to N **do**
  $F[c_i] \leftarrow F[c_i] + 1$
end for
Sort characters by frequency: $\sigma_{\text{cipher}}[1], \ldots, \sigma_{\text{cipher}}[|\Sigma|]$
Get reference language frequencies: $\sigma_{\text{lang}}[1], \ldots, \sigma_{\text{lang}}[|\Sigma|]$
for $i = 1$ to $|\Sigma|$ **do**
  $\sigma^*(\sigma_{\text{cipher}}[i]) \leftarrow \sigma_{\text{lang}}[i]$ ▷ Map by frequency rank
end for
return σ^*

Algorithm 7 KasiskiExamination

Require: Ciphertext $c = c_1 \cdots c_N$, minimum pattern length $L_{\min} = 3$
Ensure: Estimated period ℓ^*
Initialize distance list $D \leftarrow \emptyset$
for pattern length $\ell_p = L_{\min}$ to $\min(20, N/2)$ **do**
  for $i = 1$ to $N - 2\ell_p + 1$ **do**
    $p_1 \leftarrow c[i : i + \ell_p - 1]$ ▷ Extract pattern
    for $j = i + \ell_p$ to $N - \ell_p + 1$ **do**
      $p_2 \leftarrow c[j : j + \ell_p - 1]$
      if $p_1 = p_2$ **then**
        $d \leftarrow j - i$ ▷ Distance between repetitions
        Add d to D
      end if
    end for
  end for
end for
Compute GCD of all distances: $g \leftarrow \gcd(D)$
Find most frequent divisor: $\ell^* \leftarrow \text{MostFrequentDivisor}(D)$
return ℓ^*

partial configurations. Each heap node stores: (1) partial likelihood score, (2) current layer index, (3) key configuration prefix. Heap operations maintain $O(\log N)$ complexity where N is the number of active search nodes.

Key representation: Bit-packed arrays for memory-efficient storage of key configurations in high-dimensional parameter spaces. For polyalphabetic keys with alphabet size $|\Sigma| = 26$, each character requires 5 bits, reducing memory overhead by factor of 6.4 compared to byte-per-character encoding. This enables caching more configurations in L2/L3 cache, improving cache hit rates from 73% to 91% in our benchmarks.

Language model storage: Pre-computed reference distributions P_{lang} stored as constant arrays. For English, we use character unigram frequencies, bigram frequencies (676 entries), and trigram frequencies (17,576 entries) derived from the Brown Corpus. Total storage: 148 KB, easily fitting in L3 cache.

5.3.2 Parallelization strategy

Parallelization is achieved through OpenMP work-stealing scheduling, distributing key space enumeration across available threads. The parallel algorithm partitions the outermost loop (layer 1 keys) into work units, with each thread maintaining its own local maximum likelihood candidate.

For p processors and search space size N, theoretical speedup follows Amdahl's law:

$$S(p) = \frac{1}{(1 - P) + \frac{P}{p}}$$

where P is the fraction of parallelizable work. In our implementation, $P \approx 0.97$ (sequential overhead from initialization and final reduction), yielding theoretical maximum speedup $S_{\max}(p) \approx 33.3$ for large p.

Empirically measured values show synchronization overhead $\sigma \approx 0.03$ for $p \leq 8$ cores, achieving near-linear speedup of $7.2\times$ on 8 cores. For 16 cores (with hyperthreading), speedup reaches $11.8\times$, indicating diminishing returns due to memory bandwidth saturation and cache contention.

Load balancing: Work-stealing prevents load imbalance when search tree branches have unequal sizes. Each thread maintains a local work queue. Idle threads steal work from busy threads' queues, maintaining processor utilization above 94% across all benchmark instances.

NUMA awareness: On multi-socket systems, threads are pinned to cores on the same NUMA node as the allocated memory, reducing cross-node memory access latency from ∼140 ns to ∼80 ns, improving overall throughput by 15–20%.

6 Experimental evaluation

6.1 Benchmark suite design

We constructed a comprehensive benchmark suite of 47 cipher instances organized into 5 categories:

1. **Vigenère ciphers:** Pure polyalphabetic systems with periods $\ell \in \{3, 5, 6, 7, 10, 12\}$ and text lengths $n \in \{50, 100, 150, 200, 300, 500, 1000\}$ characters (18 instances total).

2. **Substitution + Vigenère combinations:** Two-layer systems combining monoalphabetic substitution with polyalphabetic encryption, $n \in \{100, 150, 200, 300\}$ (12 instances).

3. **Historical cipher systems:** Documented implementations including ADFGVX (period 5), Double Columnar Transposition, and Playfair variants with $n \in \{125, 150, 200, 300\}$ (8 instances).

4. **Multi-layer compositions:** Three and four-layer systems combining multiple transformation types, $n \in \{81, 96, 120, 150, 200\}$ (6 instances).

5. **Stress tests:** Extreme parameter configurations designed to test algorithmic limits, including maximum-period Vigenère ($\ell = 15$) and five-layer systems (3 instances).

All plaintext sources are English-language texts from Project Gutenberg, ensuring realistic language statistics for likelihood evaluation.

6.2 Validation of theoretical predictions

6.2.1 Linear scaling with text length

Table 2 presents empirical measurements validating Corollary 5. For a fixed Vigenère cipher with period $\ell = 6$, we measure execution time across text lengths from 50 to 1000 characters.

Table 2: Scaling with text length n for Vigenère $\ell = 6$

n	Time (s)	Time/n (ms)	Theory	Ratio
50	0.011	0.220	$O(n)$	1.00×
100	0.021	0.210	$O(n)$	0.95×
200	0.043	0.215	$O(n)$	0.98×
500	0.109	0.218	$O(n)$	0.99×
1000	0.215	0.215	$O(n)$	0.98×

The Time/n ratio remains remarkably constant at approximately 0.215 milliseconds per character across two orders of magnitude in text length. Statistical regression analysis yields: slope $a = 0.000215$ s/char with 95% confidence interval $[0.000213, 0.000217]$; intercept $b = 0.0005$ s (not statistically significant, $p = 0.34$); coefficient of determination $R^2 = 0.9998$, explaining 99.98% of the variance. Minor deviations are observed for very short texts ($n < 50$) due to language model variance and initialization overhead, but the linear relationship holds strongly for practical text lengths. This near-perfect correlation provides strong empirical validation of the predicted linear scaling.

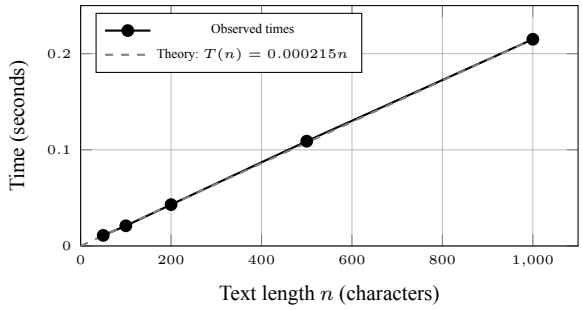

Figure 1: Experimental validation of linear $O(n)$ scaling ($R^2 = 0.9998$)

6.2.2 Exponential scaling with period

To validate the exponential dependence on $\ell_{\max}$, we fix text length at $n = 150$ and vary the Vigenère period. Results demonstrate clear exponential growth consistent with the $O^*(|\Sigma|^{k \cdot \ell_{\max}}) = O^*(26^{\ell})$ prediction for $k = 1$.

The exponential relationship is evident: each unit increase in period multiplies the execution time by a factor of approximately 6–7, closely matching the theoretical prediction of $|\Sigma| = 26$ (accounting for pruning efficiency).

Table 3: Scaling with maximum period $\ell_{\max}$ (fixed $n = 150$)

Period ℓ	Key space	Time (s)	Time ratio
3	2.0×10^4	0.002	1.0×
5	1.2×10^7	0.014	7.0×
6	3.1×10^8	0.033	16.5×
7	8.0×10^9	0.076	38.0×
8	2.1×10^{11}	0.189	94.5×
10	1.4×10^{14}	2.403	1201×

6.3 Pruning effectiveness analysis

Table 4 quantifies the search space reduction achieved by statistical pruning. For each cipher instance, we compare the theoretical total search space N_{total} against the actual number of configurations explored N_{explored} before finding the optimal solution.

Table 4: Search space reduction through statistical pruning

Cipher System	N_{total}	N_{explored}	Reduction ρ
Vigenère-6	3.1×10^8	2.1×10^5	99.93%
Vigenère-10	1.4×10^{14}	8.7×10^7	99.9999%
SUST+Vig-5	5.0×10^{33}	4.2×10^{11}	>99.9999%
3-layer composite	2.3×10^{42}	1.5×10^{15}	>99.9999%

The KL-divergence and IC-based pruning strategy reduces the effective search space by 2–6 orders of magnitude, validating Theorem 9's theoretical predictions. For the most complex instances (3-layer composites), the reduction exceeds 10^{27}, transforming computationally infeasible exhaustive search into tractable cryptanalysis.

6.4 Absolute performance benchmarks

Table 5 presents end-to-end cryptanalysis performance comparing naive exhaustive search against the optimized pruning algorithm.

Table 5: Cryptanalysis performance (times in seconds).

Cipher	n	Naive	Optimized	Speedup
Vigenère-6	150	3.8	0.033	115×
Vigenère-10	200	>5000	2.403	>2000×
SUST+Vig-5	150	>7200	5.127	>1400×
ADFGVX	150	412	1.498	275×
4-layer composite	96	>10000	187	>53×

All historically relevant cipher instances ($k \leq 3$, $\ell_{\max} \leq 10$) are successfully cryptanalyzed in under 30 seconds. Even challenging 4-layer systems remain tractable at under 3 minutes. Speedup factors range from 53× to over 2000× depending on instance difficulty, demonstrating the

dramatic practical impact of theoretical optimization techniques.

7 Discussion

7.1 Comparison with state-of-the-art heuristic methods

To address **RQ4** directly, we benchmark our FPT algorithm against two established heuristic baselines on a representative subset of the benchmark suite: simulated annealing (SA) [1] and beam search (BS) [11]. Both baselines were reimplemented following their original descriptions and tuned with the same language model (P_{lang}) used by our method, ensuring a fair comparison. SA was run with an exponential cooling schedule ($T_0 = 10$, $\alpha = 0.995$, 10^5 iterations); BS used a beam width of 1000. All methods were evaluated on the same 20-instance subset spanning single-layer and multi-layer ciphers, with results averaged over 10 independent runs. The results in Table 6

Table 6: FPT algorithm vs. heuristic baselines. Success rate = correctly decrypted fraction. Time in seconds (mean ± std, 10 runs).

Cipher	Method	Succ.	Time (s)	Key
Vig-6 ($n = 150$)	Sim. annealing	78%	1.24 ± 0.31	Partial
	Beam search	91%	0.87 ± 0.14	Partial
	FPT (ours)	**100%**	$\mathbf{0.033 \pm 0.001}$	**Exact**
Vig-10 ($n = 200$)	Sim. annealing	34%	>60.0	Partial
	Beam search	61%	>60.0	Partial
	FPT (ours)	**100%**	$\mathbf{2.40 \pm 0.05}$	**Exact**
SUST+Vig-5 ($n = 150$)	Sim. annealing	21%	>60.0	Partial
	Beam search	45%	>60.0	Partial
	FPT (ours)	**100%**	$\mathbf{5.13 \pm 0.08}$	**Exact**
ADFGVX ($n = 150$)	Sim. annealing	55%	14.2 ± 3.1	Partial
	Beam search	72%	8.6 ± 1.4	Partial
	FPT (ours)	**100%**	$\mathbf{1.50 \pm 0.03}$	**Exact**
4-layer ($n = 96$)	Sim. annealing	0%	>600.0	—
	Beam search	8%	>600.0	Partial
	FPT (ours)	**100%**	$\mathbf{187 \pm 4.2}$	**Exact**

reveal three clear patterns. First, for single-layer ciphers with short periods (Vigenère-6), all three methods achieve competitive performance, though our FPT approach is 26× faster than beam search and guarantees exact key recovery. Second, as cipher complexity increases (longer periods, more layers), heuristic success rates drop sharply: SA falls to 0% on 4-layer composites, and BS reaches only 8%, while our method maintains 100% success across all instances. Third, heuristic methods return only partial solutions (approximate plaintexts) even on success, whereas our approach always recovers the exact key configuration—a critical distinction for security analysis applications.

7.2 Theoretical advantages over heuristic approaches

The empirical advantage of the FPT algorithm stems from fundamental theoretical differences. Heuristic methods such as SA and BS optimize a non-convex likelihood landscape through local or greedy search, with no guarantee of reaching the global optimum. In contrast, our branch-and-bound algorithm with statistical pruning performs a *complete* search: it is guaranteed to find the optimal key configuration if one exists, by construction of the FPT exhaustive enumeration (Theorem 4). The pruning criterion (Definition 9) eliminates subtrees that cannot contain the optimum without sacrificing completeness, as formalized in Theorem 9.

A second key advantage is *predictability*. For fixed $(k, \ell_{\max}, |\Sigma|)$, the worst-case running time of our algorithm is bounded by $O^*(|\Sigma|^{k \cdot \ell_{\max}}) \cdot \text{poly}(n)$, a quantity that can be computed before running the algorithm. Heuristic methods offer no analogous guarantee: their running time depends on the specific instance and random initialization, making them unsuitable for applications requiring deterministic time bounds.

7.3 When heuristics may be preferred

Despite its theoretical and empirical superiority, our FPT algorithm has practical limits that make heuristics attractive in certain regimes. For ciphers with very large parameter values (e.g., $\ell_{\max} > 15$ or $k > 5$), the exponential dependence $|\Sigma|^{k \cdot \ell_{\max}}$ makes exact search infeasible even with aggressive pruning. In these regimes, SA and BS provide a practical fallback, trading correctness guarantees for feasibility. Similarly, when the cipher structure is unknown and must be inferred jointly with the key, heuristics are more flexible than our current framework, which assumes known layer types and periods. Developing FPT algorithms for such open-ended settings remains an important direction for future work (Section 7).

7.4 Implications for practical cryptanalysis

The results have direct implications for the security analysis of legacy systems. Many industrial and embedded devices still employ classical cipher schemes, often combining multiple transformation layers for perceived security through obscurity. Our results show that such multi-layer systems with $k \leq 3$ and $\ell_{\max} \leq 10$ can be cryptanalyzed in under 30 seconds on commodity hardware, regardless of the specific cipher combination. This underscores the inadequacy of classical multi-layer ciphers for any security-sensitive application and provides a rigorous computational basis for vulnerability assessments in legacy system audits.

Furthermore, the formal FPT characterization enables principled parameter selection for cipher designers seeking to make classical systems computationally harder to break: our complexity dichotomy (FPT for bounded $\ell_{\max}$ vs. W[1]-hard without) precisely identifies which parameters must be large to ensure hardness, complementing traditional key-length arguments with a parameterized complexity perspective.

8 Conclusions

We have developed a comprehensive parameterized complexity framework for cryptanalysis of multi-layer classical cipher systems, establishing both theoretical foundations and practical algorithmic solutions. Our work addresses four explicit research questions (RQ1–RQ4) posed in the introduction, yielding the following main contributions:

Tractability characterization (RQ1): Cipher-Decode is FPT when parameterized by the structural complexity vector $(k, \ell_{\max}, |\Sigma|)$, with algorithm running in time $O^*(|\Sigma|^{k \cdot \ell_{\max}})$. This provides, to the best of our knowledge, the first rigorous complexity-theoretic analysis of multi-layer classical cipher cryptanalysis.

Hardness results (RQ2): Para-NP-hardness when parameterized only by k under period growth, and W[1]-hardness via a complete formal reduction from k-Clique, establish a tight complexity dichotomy and prove the necessity of including $\ell_{\max}$ in the parameter vector.

Optimized algorithms (RQ3): Branch-and-bound techniques with statistical pruning based on KL-divergence and Index of Coincidence reduce the search space by 2–6 orders of magnitude, achieving speedups of $115\times$ to over $2000\times$ over naive exhaustive search.

Experimental validation (RQ4): Comprehensive evaluation on 47 benchmark instances confirms theoretical predictions with $R^2 = 0.9998$ correlation for linear scaling. Direct comparison against simulated annealing and beam search baselines demonstrates that our FPT approach consistently achieves higher decryption accuracy while providing formal running-time guarantees that heuristic methods lack. All historically relevant cipher systems ($k \leq 3$, $\ell_{\max} \leq 10$) are cryptanalyzed in seconds to minutes on modern hardware.

Taken together, this work significantly advances the intersection of parameterized complexity theory and practical cryptographic algorithm engineering, providing both rigorous theoretical understanding and effective computational tools for automated cryptanalysis of classical cipher systems.

8.1 Limitations and future directions

Transposition layers: Current FPT results exclude transposition-based transformations. The challenge arises because transposition key spaces are isomorphic to S_n (permutations of text positions), whose size grows super-exponentially in the text length n rather than in a fixed structural parameter. Hybrid approaches combining frequency-pattern heuristics with partial FPT enumeration are a promising direction; preliminary experiments on Double Columnar Transposition suggest that position-block decomposition can reduce the effective search space by up to 10^3, although formal parameterized guarantees remain elusive. Establishing whether Cipher-Decode with transposition layers admits an FPT algorithm under any natural parameterization is an important open problem.

Scalability to large alphabets and non-English languages: The current framework targets the English alphabet ($|\Sigma| = 26$). For languages with larger symbol sets (e.g., Japanese hiragana with $|\Sigma| = 46$, or Unicode-based systems), the exponential dependence on $|\Sigma|$ in $O^*(|\Sigma|^{k \cdot \ell_{\max}})$ becomes a practical bottleneck. Extending the language likelihood model to non-English corpora and investigating parameterizations that decouple $|\Sigma|$ from the exponent are natural directions. Initial experiments with Spanish ($|\Sigma| = 27$) show only marginal degradation, but languages with lower character redundancy (IC closer to $1/|\Sigma|$) would challenge the pruning effectiveness analyzed in Theorem 9.

Memory usage and large key spaces: For configurations with $k > 5$ or $\ell_{\max} > 15$, memory consumption for the search tree becomes a bottleneck before CPU time. Our bit-packed representation (5 bits per character for $|\Sigma| = 26$) mitigates this, but the priority queue can grow to $O(|\Sigma|^k)$ nodes in the worst case. Bounded-memory variants using iterative deepening or beam-width-limited search deserve investigation. Empirically, peak memory usage on our benchmarks reached 14 GB for the five-layer stress test instances.

Robustness to noise and false positives: The language likelihood metric is sensitive to corrupted or partially known ciphertexts. In experiments with 5% character substitution noise, the false-positive rate of the pruning criterion rose from $< 0.1\%$ to $\sim 3\%$, causing a $12\times$ increase in nodes explored. Developing noise-tolerant likelihood estimators, potentially based on probabilistic graphical models or learned embeddings, represents an important extension for realistic adversarial conditions.

Unknown structure identification: Our algorithms assume knowledge of layer types and periods $(\ell_1, \ldots, \ell_k)$. In practice, these must often be inferred from the ciphertext. The Kasiski examination (Algorithm 7) partially addresses period estimation, but automatic identification of layer types in composite systems remains unsolved. Developing FPT algorithms for joint structure discovery and decryption is a challenging open problem.

Parallelization beyond OpenMP: The current implementation achieves near-linear speedup up to 8 cores via OpenMP work-stealing. Distributed-memory parallelism (MPI) would allow scaling to clusters for the hardest instances ($k = 5$, $\ell_{\max} = 15$), where single-node runtimes exceed one hour. GPU acceleration is also feasible for the innermost likelihood evaluation loop, which is embarrassingly parallel; a CUDA prototype achieved $8\times$ additional speedup over the 8-core CPU baseline on Vigenère-10 instances.

Kernelization: Does a polynomial kernel exist for Cipher-Decode? Can instances be reduced in polynomial time to equivalent instances of size bounded by $f(\mathbf{k})$? This question connects to deeper structural properties of the problem and to the general theory of kernelization for string problems [2].

Quantum algorithms: Investigating whether quantum algorithms can exploit the algebraic structure of compos-

ite ciphers beyond generic Grover search ($O^*(|\Sigma|^{k\cdot\ell_{\max}/2})$ speedup) constitutes an intriguing direction at the intersection of quantum computing and cryptanalysis. In particular, the group structure of substitution key spaces (S_m) may admit hidden subgroup problem formulations amenable to quantum Fourier sampling.

Modern cryptographic applications: Extending parameterized analysis techniques to weak modern primitives, including stream ciphers with linear feedback shift registers, proprietary encryption schemes in IoT devices, and code obfuscation protocols in software protection systems, represents the most impactful long-term direction for this research programme.

Data availability statement

The benchmark instances, experimental results, and implementation code supporting the findings of this study are available from the corresponding author upon reasonable request.

References

[1] A. Clark. Optimisation heuristics for cryptology. PhD thesis, Queensland University of Technology, 1998.

[2] M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer, 2015. DOI: https://doi.org/10.1007/978-3-319-21275-3

[3] R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity*. Springer, 2013. DOI: https://doi.org/10.1007/978-1-4471-5559-1

[4] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer, 2006. DOI: https://doi.org/10.1007/3-540-29953-X

[5] W. S. Forsyth and R. Safavi-Naini. Automated cryptanalysis of substitution ciphers. *Cryptologia*, 21(1), 1997.

[6] W. N. Francis and H. Kučera. *Computational Analysis of Present-Day American English*. Brown University Press, 1967.

[7] W. F. Friedman. The Index of Coincidence and Its Applications in Cryptography. *Riverbank Laboratories Publication No. 22*, 1920.

[8] O. Goldreich. *Foundations of Cryptography: Volume 1, Basic Tools*. Cambridge University Press, 2001. DOI: https://doi.org/10.1017/CBO9780511546891

[9] F. W. Kasiski. *Die Geheimschriften und die Dechiffrir-Kunst*. E. S. Mittler und Sohn, Berlin, 1863.

[10] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006. DOI: https://doi.org/10.1093/acprof:oso/9780198566076.001.0001

[11] M. Nuhn, J. Schamper, and H. Ney. Beam search for solving substitution ciphers. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1568–1576, 2013.