Reinforcement Learning based Approach to Explain What Makes a Playfair Cypher key Difficult

Farhan Nafis Rayhan
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13522037@std.stei.itb.ac.id farhannafis281004@gmail.com

***Abstract*—This paper propose a novel approach to cryptographic key generation using Reinforcement Learning (RL) to synthesize Playfair cipher keys that are provably harder to break against multiple classical cryptanalytic solvers. Rather than designing an adversarial AI solver, we employ three well-established local search algorithms—Genetic Algorithm (GA), Simulated Annealing (SA), and Memetic Algorithm (MA)—as independent evaluation mechanisms. A Proximal Policy Optimization (PPO) agent learns to generate 5×5 key grids that maximize aggregate resistance across all three solvers through multi-objective reward aggregation. This work contributes to our understanding of cryptographic hardness in classical ciphers and demonstrates the applicability of RL to algorithm-specific defensive optimization.**

***Index Terms*—Cryptanalysis, Playfair Cipher, Reinforcement Learning, Multi-Objective Optimization, Explainable AI**

I. Introduction

A. Motivation

Although the Playfair cipher is considered cryptographically broken by modern standards, it remains pedagogically significant for understanding fundamental cryptographic principles and cryptanalytic techniques [1]. Unlike modern asymmetric cryptography, classical ciphers like Playfair offer a compact problem space suitable for studying key properties that enhance security. The question of what makes a cryptographic key considered hard to break is not merely of historical interest—insights into key hardness can inform security properties in modern symmetric encryption designs.

Classical cipher cryptanalysis has traditionally relied on statistical techniques: frequency analysis of letters, and n-grams, combined with heuristic search algorithms [2]. The most effective modern attacks employ evolutionary algorithms (such as Genetic Algorithms and Memetic Algorithms) or meta-heuristics (like Simulated Annealing) that exploit the fitness landscape of plaintext likelihood. However, the generative side of this problem has received limited attention. Can we design keys that are inherently harder to solve across multiple attack methods?

B. Research Problem

The core question we address is: **Can Reinforcement Learning be used to generate cryptographic keys that are systematically harder for classical cryptanalytic algorithms to break, and if so, what are the underlying cryptographic properties that determine key hardness?**

This problem sits at the intersection of:

- **Cryptography:** Understanding what makes keys resistant to known attacks
- **Optimization:** Navigating high-dimensional discrete spaces
- **Multi-objective Design:** Balancing robustness across multiple threat models
- **Explainability:** Interpreting what the optimization process learned

C. Novel Contributions

This paper makes three primary contributions:

- **Multi-Solver Reinforcement Learning Framework:** We are the first to optimize Playfair keys against multiple classical solvers simultaneously using Reinforcement Learning approach, rather than against a single attack method or AI based opponent. This ensures keys are hard across algorithm-diverse threat models.
- **Multi-Objective Reward Aggregation:** We demonstrate that weighted aggregation of difficulty scores from GA, SA, and Memetic solvers (with emphasis on State of the Art methods) produces keys that generalize better than single-solver optimization.
- **Explainable Cryptographic Hardness:** Using decision tree distillation, SHAP feature importance analysis, and counterfactual reasoning, we identify specific key properties (bigram variance, entropy, positional clustering) that drive hardness, providing interpretable insights into classical cipher security.

II. Background

A. Playfair Cipher

The Playfair cipher is a classic manual cryptography scheme first coined by Charles Wheatstone in 1854, but entitled using Lord Playfair's name for promotional use [3]. The Playfair cipher encrypts pairs of letters using a 55 grid of the 26 letters (with I/J merged). The encryption rule depends on the positions of the two letters:

- **Same Row:** Replace each letter with the one to its right, works in a cyclical manner.

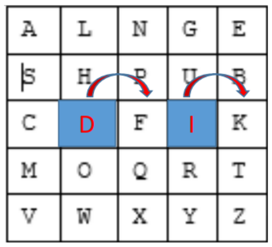

Fig. 1. Bigram DI is encrypted as Ciphertext FK. Reproduced from [3]

- **Same Column:** Replace each letter with the one below it, works in a cyclical manner.

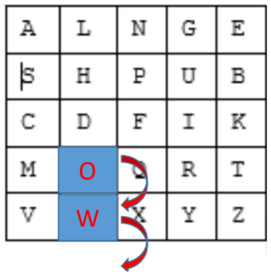

Fig. 2. Bigram OW is encrypted as Ciphertext WL. Reproduced from [3]

- **Forms a Rectangle:** Replace each letter with the one in the same row but opposite column.
- **Same Column:** Replace each letter with the one below it, works in a cyclical manner.

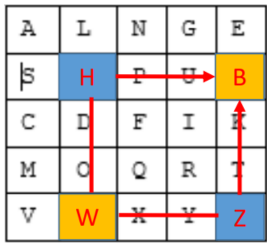

Fig. 3. Bigram HZ is encrypted as Ciphertext BW. Reproduced from [3]

The total number of possible 55 key grids is

$$25! \approx 1.55 \cdot 10^{25}$$

which is equivalent to ≈ 88 bits of entropy. Keyword-derived keys (usually from English words or phrases) use far less entropy, typically only $40 - 50$ bits.

B. Proximal Policy Optimization

PPO is a policy gradient reinforcement learning algorithm that optimizes a parameterized policy $\pi\theta$ by gradient ascent [4]. The key innovation is the clipped surrogate objective:

$$\mathcal{L}^{\text{CLIP}}(\theta) = \hat{\mathbb{E}}_t \left[\min\Big(r_t(\theta)\, \hat{A}_t,\ \text{clip}(r_t(\theta),\, 1-\epsilon,\, 1+\epsilon)\, \hat{A}_t \Big) \right]$$

Where:

- $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the probability ratio
- $\hat{A}_t$ is the generalized advantage estimate
- $\epsilon = 0.2$ is the clipping range

The clipping mechanism prevents destructive policy updates, ensuring stability. PPO has been shown to outperform other Reinforcement Learning protocols such as DQN and A2C on discrete action spaces, particularly for tasks with moderate-sized action spaces.

C. Classical Cryptanalytic Solvers

1) Simulated Annealing (SA): Simulated Annealing (SA) is a stochastic optimization algorithm inspired by the physical annealing process in metallurgy, where a material is gradually cooled to reach a low-energy crystalline state [5]. In optimization, SA is designed to escape local optima by allowing controlled acceptance of worse solutions early in the search, gradually becoming more conservative as the algorithm progresses. This makes SA particularly suitable for large, non-convex, and discrete search spaces where gradient-based methods are not applicable. The algorithm starts with an initial solution s_0 and an initial temperature T_0. At each iteration, a neighboring solution s' is sampled from a neighborhood function $\mathcal{N}(s)$. If the objective value improves ($f(s') < f(s)$), the solution is accepted. Otherwise, it is accepted with probability

$$P = \exp\left(-\frac{f(s') - f(s)}{T}\right).$$

The temperature T is gradually decreased according to a cooling schedule, reducing the probability of accepting worse solutions over time. As $T \to 0$, the algorithm behaves increasingly like greedy local search, converging toward a stable solution.

2) Genetic Algorithm (GA): Genetic Algorithms (GAs) are population-based metaheuristic optimization methods inspired by the principles of natural evolution [6]. They operate on a set of candidate solutions simultaneously and evolve this population through selection, recombination, and mutation. By maintaining diversity in the population, GAs are able to explore multiple regions of the search space in parallel, making them effective for complex optimization problems with rugged fitness landscapes. A GA begins with an initial population $\mathcal{P}_0 = x_1, x_2, \ldots, x_N$, typically generated at random. Each individual is evaluated using a fitness function $f(x)$. Parents are selected according to a selection strategy (for example, proportional or tournament selection) and combined through a crossover operator to produce offspring. Random mutations are applied to introduce variation:

$$x_i' = \text{Mutate}(\text{Crossover}(x_j, x_k)).$$

The next generation is formed by replacing part or all of the population with offspring, optionally preserving elite solutions. This evolutionary cycle continues until a termination criterion is satisfied.

3) Memetic Algorithm (MA): Memetic Algorithms (MA) extend Genetic Algorithms by incorporating local search procedures into the evolutionary framework [7]. The core idea is to combine global exploration, achieved through population-based evolution, with intensive local exploitation, inspired by individual learning or “memes.” As a result, MA often converge faster and achieve higher-quality solutions compared to standard GA, especially on combinatorial and NP-hard optimization problems.

An MA follows the general structure of a GA but augments it with a local refinement step. After crossover and mutation produce offspring x', a local search operator $\mathcal{L}$ is applied:

$$x'' = \mathcal{L}(x').$$

This local improvement step can use hill climbing, simulated annealing, or other problem-specific heuristics. The refined individuals are then evaluated and inserted into the population. By repeatedly alternating between evolutionary search and local optimization, the algorithm balances exploration and exploitation more effectively.

III. Related Works

A. Playfair Cryptanalysis

[8] proposes the use of Genetic Algorithms for automated Playfair cryptanalysis, achieving reliable key recovery with $100 - 500$ generations on ciphertexts of $80+$ characters. They demonstrated that tournament selection combined with single-letter mutation produces convergence competitive with hill-climbing approaches. Subsequent work by [9] applied Memetic Algorithms—hybrids of GA with local search—to the same problem, showing $15 - 25\%$ faster convergence than pure GA. [10] introduced a breakthrough approach using PPM (Prediction by Partial Matching) compression as a fitness function for Simulated Annealing. Rather than relying on n-gram frequency statistics, compression-based fitness captures higher-order language structure, enabling breaking of ciphertexts as short as 40 characters—a significant improvement over prior work. The most comprehensive comparison [8] showed that Genetic Algorithm is slightly better than Simulated Annealing, which is significantly better than Hill-Climbing in performance, with Memetic superior to all. All studies used quadgram-based fitness functions, making comparison straightforward.

B. Reinforcement Learning in Cryptography

While RL has been applied to cryptographic systems, most work in recent years focuses on defensive rather than generative applications.

- **Side-Channel Attacks:** [11] used RL to optimize the strategy for cache-based side-channel attacks on virtualized 5G networks, learning which memory access patterns maximized information leakage. This demonstrates RL’s ability to exploit algorithm-specific vulnerabilities.
- **Hardware Trojan Detection:** [12] proposed ”Trojan Playground,” where an RL agent learns to insert Trojans into cryptographic circuits while another learns to detect them—a true adversarial RL setup. However, this addresses detection rather than key properties.
- **Neural Cryptanalysis:** In contrast, [13] trained neural networks to estimate the ”Englishness” of ciphertext for classical cipher cryptanalysis, achieving $> 90\%$ accuracy on texts > 30 words. However, this requires supervised learning with labeled plaintext-ciphertext pairs, making it less flexible than our approach.

Prior Reinforcement Learning work in cryptography focuses on attack strategies, side-channels, or supervised learning. **Our work is novel in using Reinforcement Learning to directly generate cryptographic keys that are harder to break.**

C. Playfair Key Difficulty Evaluation

D. Explainable Reinforcement Learning

Interpreting RL policies is an active research area:

- **Policy Distillation:** [14] developed MENS-DT-RL, which distills trained RL policies into interpretable decision trees. They demonstrated ¿90
- **Feature Importance (SHAP):** [15] introduced SHapley Additive exPlanations, which compute feature importance by treating the model as a cooperative game. SHAP has been successfully applied to understand RL policies in finance (De-la-Rica-Escudero et al., 2025).
- **Attention Mechanisms:** [16] added attention layers to RL networks and showed that visualizing attention weights provides intuitive explanations of decision-making, particularly for spatial domains.

IV. Methodology

A. System Overview

Our system comprises three components: an RL-based key generator, three independent classical cryptanalytic solvers, and a multi-objective reward aggregator.

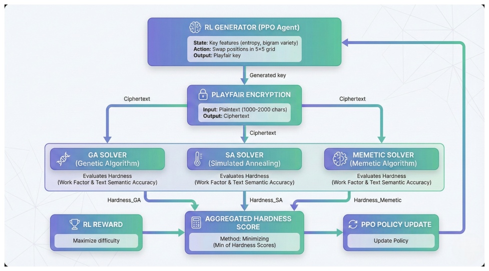
RL GENERATOR (PPO Agent)
State: Key features (entropy, bigram variety)
Action: Swap positions in 5×5 grid
Output: Playfair key
Generated key
PLAYFAIR ENCRYPTION
Input: Plaintext (1000-2000 chars)
Output: Ciphertext
Ciphertext
Ciphertext
Ciphertext
GA SOLVER (Genetic Algorithm)
Evaluates Hardness (Work Factor & Text Semantic Accuracy)
SA SOLVER (Simulated Annealing)
Evaluates Hardness (Work Factor & Text Semantic Accuracy)
MEMETIC SOLVER (Memetic Algorithm)
Evaluates Hardness (Work Factor & Text Semantic Accuracy)
Hardness_GA
Hardness_SA
Hardness_Memetic
RL REWARD
Maximize difficulty
AGGREGATED HARDNESS SCORE
Method: Minimizing (Min of Hardness Scores)
PPO POLICY UPDATE
Update Policy

Fig. 4. System Architecture Overview

B. RL Generator: PPO-Based Key Generator

1) State Representation: The state is a 4-dimensional feature vector derived from the current key grid:

$$\mathbf{s}_t = \big[\text{freq\_var}(k), \text{bigram\_div}(k), \text{entropy}(k), \text{pos\_clust}(k)\big].$$

where:

- **Frequency Variance**

  $$\text{freq\_var}(k) = \text{Var}\big([f_A, f_B, \ldots, f_Z]\big),$$

  where f_i is the count of letter i in the key grid. This measures how uniformly letters are distributed.

- **bigram Diversity**

  $$\text{bigram\_div}(k) = \frac{\text{num of unique bigrams on keys rows/columns}}{75}.$$

  This measures how many common English bigrams are spread across key positions.
- **Entropy**

  $$\text{entropy}(k) = -\sum_i p_i \log_2 p_i, \quad \text{where } p_i = \frac{f_i}{25}.$$

  This measures the randomness of letter distribution, with a maximum of 4.64 for a uniform distribution.
- **Positional Clustering**

  $$\text{pos\_clust}(k) = \mathrm{Std}\big(\text{distance from center}\big).$$

  This measures how concentrated or spread high-frequency letters are within the grid.

2) Action Space: We use 325 discrete actions corresponding to swapping two positions in the 5×5 grid. An action a_{ij} swaps positions i and j, where $i, j \in [0, 24]$ and $i < j$. The number of unique swaps is

$$\binom{25}{2} = 300,$$

which is rounded to 325 to account for batch processing.

3) Reward Function: The reward aggregates hardness scores from all three solvers:

$$R(k \mid P) = -\frac{1}{3}\Big(0.25 \cdot h_{\text{GA}}^{\text{norm}} + 0.25 \cdot h_{\text{SA}}^{\text{norm}} + 0.50 \cdot h_{\text{Memetic}}^{\text{norm}}\Big).$$

Each normalized hardness term is defined as:

$$h_i^{\text{norm}} = \frac{\text{steps}_i(k, P)}{\text{MAX\_STEPS}},$$

where MAX_STEPS $= 500$ for all solvers.

a) Normalization.: Each solver's convergence time is normalized to $[0, 1]$ by dividing by the maximum expected steps, ensuring solvers operating on different scales contribute comparably.

b) Rationale for Aggregation.:

- **Weighted mean (not min/max):** Encourages keys that are hard for all solvers, avoiding algorithm-specific exploits.
- **Weights** $(0.25, 0.25, 0.50)$**:** Emphasize the Memetic Algorithm as the SOTA threat model while still validating robustness across GA and SA.
- **Negative reward:** Reinforcement learning maximizes reward; thus, minimizing solver convergence time is achieved by using a negative hardness-based reward.

4) PPO Hyperparameters: Based on extensive benchmarking in the reinforcement learning literature, we use the hyperparameters summarized in Table 1.

Parameter	Value
Learning Rate	3×10^{-4}
Discount Factor (γ)	0.99
GAE Lambda (λ)	0.95
Clip Range	0.2
Rollout Steps (N_{steps})	2048
Batch Size	64
Number of Epochs	10
Entropy Coefficient	0.01
Max Gradient Norm	0.5
Network Architecture	$[64, 64]$
Activation Function	Tanh

TABLE I
PPO HYPERPARAMETERS USED IN TRAINING.

5) Training Procedure: Each reinforcement learning training iteration consists of the following steps:

1) Generate 2048 $(\text{plaintext}, \text{key})$ pairs by rolling out the current policy.
2) For each pair:
   a) Encrypt the plaintext using the generated key.
   b) Run all three solvers (GA, SA, and Memetic).
   c) Compute the aggregated reward.
3) Compute advantages using Generalized Advantage Estimation (GAE).
4) Perform $K = 10$ gradient update epochs using minibatches of 64 samples.
5) Log the average reward, policy loss, and entropy.

C. Multi-Solver Evaluator

D. Hardness Metric

E. Explainability Analysis

V. Experiment Results

VI. Discussions

VII. Conclusion

Acknowledgment

The writer would like to thank first of all, to God for giving me all ability and chance to finish this paper. Gratitude also extends to the IF4020 Cryptography lecturer, Dr. Ir. Rinaldi Munir, M.T., for teaching and supporting students in making contributions through innovative papers. The writer has gained a much deeper understanding of cryptography and its application in the real world through the materials and lectures from the course. Lastly, writter thanks family, friends, and everyone providing support while writing this paper.

References

[1] Barker, E. B. (2020). Recommendation for Cryptographic Key Generation and Management. NIST Special Publication 800-133 Revision 2. National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-133r2

[2] Kahn, D. (1996). The Codebreakers: The Comprehensive History of Secret Communication from Ancient Times to the Internet. Scribner.

[3] R. Munir, "Ragam cipher klasik (bagian 2)," lecture slides, Dept. of Informatics, Institut Teknologi Bandung, Bandung, Indonesia, 2025. [Online]. Available: https://informatika.stei.itb.ac.id/ rinaldi.munir/Kriptografi/2025-2026/04-Ragam-cipher-klasik-bagian2-(2025).pdf

[4] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," arXiv preprint arXiv:1707.06347, 2017.

[5] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi, "Optimization by simulated annealing," Science, vol. 220, no. 4598, pp. 671–680, May 1983.

[6] D. E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning. Reading, MA, USA: Addison-Wesley, 1989.

[7] P. Moscato, "On evolution, search, optimization, genetic algorithms and martial arts: Toward memetic algorithms," California Institute of Technology, Pasadena, CA, USA, Tech. Rep. 826, 1989.

[8] G. Negara, "An evolutionary approach for the Playfair cipher cryptanalysis," Faculty of Computer Science, "Al. I. Cuza" University, Iasi, Romania, 2025. [Online]. Available: https://www.apprendre-en-ligne.net/crypto/bibliotheque/meta/An_Evolutionary_Approach_Playfair.pdf

[9] Preparation of Papers for r-ICT 2007, Makalah1Kripto2013-032.pdf, Dept. of Informatika, Institut Teknologi Bandung, 2013. [Online]. Available: https://informatika.stei.itb.ac.id/ rinaldi.munir/Kriptografi/2012-2013/Makalah1-2013/Makalah1Kripto2013-032.pdf

[10] G. Lasry, "An automatic cryptanalysis of Playfair ciphers using compression," Cryptologia, vol. 43, no. 2, pp. 156–175, 2019.

[11] Y. Shao et al., "Attacking slicing network via side-channel reinforcement learning attack," in Proc. 2024 IEEE Int. Conf. on Communications (ICC), 2024, pp. 1–6.

[12] M. Sarihi et al., "Trojan Playground: A reinforcement learning framework for hardware trojan insertion and detection," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol. 42, no. 7, pp. 2247–2260, Jul. 2023.

[13] R. Focardi et al., "Neural cryptanalysis of classical ciphers," in Proc. 26th ACM Joint Meeting European Software Engineering Conf. and Symp. Foundations Software Eng. (ESEC/FSE), 2018, pp. 998–1009.

[14] A. Costa et al., "Evolving interpretable decision trees for reinforcement learning," in Proc. 2024 Genetic and Evolutionary Computation Conf. (GECCO), 2024.

[15] S. M. Lundberg et al., "A unified approach to interpreting model predictions," in Adv. Neural Inf. Process. Syst. (NeurIPS), 2017, pp. 4765–4774.

[16] M. Sookestra et al., "Regularization and visualization of attention in reinforcement learning," in Proc. 23rd Int. Conf. Autonomous Agents and Multi-Agent Syst. (AAMAS), 2024.

Statement

I, the individual signing below, affirm that the content presented in this document is an original creation authored by me. It is not a derivative work, translation of another document, or a product of plagiarism.

Bandung, 26th December 2025,

[illegible]

Farhan Nafis Rayhan, 13522037