

# CryptoForge: Adversarial Co-Evolution for Automated Cryptographic Hash Function Design

Krishna J<sup>1</sup>, Maharaj P S<sup>2</sup>, Muraliprasath K<sup>3</sup>, Gurumoorthy S<sup>4</sup>, Esther Rani P J<sup>5</sup>

<sup>1</sup>Department of Artificial Intelligence and Data Science & Knowledge Institute of Technology, Salem

<sup>2</sup>Department of Artificial Intelligence and Data Science & Knowledge Institute of Technology, Salem

<sup>3</sup>Department of Artificial Intelligence and Data Science & Knowledge Institute of Technology, Salem

<sup>4</sup>Department of Artificial Intelligence and Data Science & Knowledge Institute of Technology, Salem

<sup>5</sup>Assistant Professor, Department of Artificial Intelligence and Data Science & Knowledge Institute of Technology, Salem

\*\*\*

**Abstract** - Cryptographic hash functions are typically designed by hand, evaluated through years of community analysis, and replaced only after a public break. There is no automated feedback loop between attack discovery and design improvement. We present CryptoForge, a system that closes this loop. CryptoForge pairs a parameterized matrix-based hash function (MatriXHash-256) with a Gohr-style neural differential distinguisher, a Walsh-Hadamard spectral analysis engine, and an evolutionary defender that modifies hash parameters in response to discovered weaknesses. The hash operates over  $GF(2^8)$  using a  $32 \times 32$  Cauchy MDS matrix with branch number 33 and the AES S-box. The neural adversary trains on self-generated differential pairs and measures a “security margin”—the round count at which distinguishing accuracy falls to 50%. The spectral engine computes a fixed-dimensional fingerprint (nonlinearity, differential uniformity, spectral flatness, entropy) that predicts adversary success cheaply, without training. Over 8 generations of co-evolution, the system drove S-box nonlinearity from 94 to 112, independently rediscovering the AES S-box. We also report two empirical findings: a binary security cliff at nonlinearity  $\approx 80$ , where neural accuracy transitions sharply from near-perfect to chance; and an  $R^2$  collapse in spectral-neural regression from 0.998 to 0.059 when the identity S-box outlier is removed, indicating the security

landscape has a threshold rather than a gradient. A C implementation achieves  $105\times$  speedup over the Python baseline. The full system passes NIST SP 800-22 (10/10 tests), achieves perfect avalanche (mean 0.5000), and zero BIC violations across 1,044,480 bit pairs..

**Key Words:** Neural cryptanalysis, adversarial co-evolution, hash function design, Walsh-Hadamard analysis, evolutionary computation, differential distinguisher.

## 1. INTRODUCTION

Hash function design is a slow process. A team of cryptographers proposes a construction, the community analyzes it for years, and eventually it either becomes a standard or gets broken. When SHA-1 was shown to be vulnerable to collision attacks in 2005 [1], a decade passed before browsers fully deprecated it. The design-evaluate-deploy cycle has no automated feedback mechanism: if an attack is found, a human must diagnose the structural weakness, redesign the primitive, and restart the evaluation. Gohr’s 2019 result [2] changed the attack side of this equation. He showed that a residual neural network trained on ciphertext pairs could distinguish reduced-round Speck32/64 from random data, outperforming classical differential cryptanalysis in some settings. Benamira et al. [3] later interpreted Gohr’s network and found it had learned differential-linear features. But these results are attack-only—they identify weaknesses without proposing fixes.

Separately, evolutionary methods have been used to design individual cryptographic components, particularly S-boxes with high nonlinearity [4]. These efforts optimize a single component within a fixed architecture. Nobody has connected automated attack and automated defense into a closed loop operating on a complete hash function. CryptoForge does exactly this. We built a parameterized hash function—MatriXHash-256—whose every design choice (mixing matrix, S-box, round count, permutation) is exposed as a tunable parameter. A neural adversary attacks it. A spectral analysis engine estimates security from algebraic properties alone. An evolutionary defender modifies the hash parameters to fix discovered weaknesses. The loop repeats.

Our contributions are:

- 1) A closed-loop co-evolution system for hash function design—the first, to our knowledge, to combine neural attack, spectral analysis, and evolutionary defense on a full hash.
- 2) Two empirical findings about the relationship between algebraic properties and neural distinguisher performance: a binary security cliff at nonlinearity  $\approx 80$ , and an  $R^2$  collapse showing the spectral-neural relationship is threshold-based.
- 3) A quasigroup algebraic structure search (CryptoGenesis) that tested 1,000 structures and found all high-quality candidates converge to  $NL=112$ .
- 4) A working deployed system with web interface, C backend ( $105\times$  speedup), and Android application.

## 2. RELATED WORK

Gohr [2] trained a ResNet-based differential distinguisher on Speck32/64, reaching accuracies above random for up to 8 rounds. The network learned to exploit differential-linear correlations, as Benamira et al. [3] later showed through interpretability analysis. Both works are attack-only and target an existing cipher rather than designing new ones.

Evolutionary S-box design has a longer history. Picek et al. [4] used genetic programming to find 8-bit S-boxes with nonlinearity close to the AES value of 112. Carlet [5] formalized the connection between Walsh spectral properties and cryptographic resistance, establishing the mathematical

framework we build on. Rijmen and Daemen [6] described the SPN paradigm underlying AES; our hash follows the same structural pattern with different parameterization.

The AI Scientist project [7] demonstrated autonomous research agents that generate hypotheses, run experiments, and write papers. It has not been applied to cryptographic design. No prior system connects neural cryptanalysis with automated design modification in a feedback loop.

## 3. SYSTEM ARCHITECTURE

CryptoForge has five components: the hash substrate, the neural adversary, the spectral engine, the evolutionary defender, and the co-evolution controller that connects them.

### A. MatriXHash-256

MatriXHash-256 is a Merkle–Damgård hash with Davies Meyer compression, operating on a 256-bit state ( $32 \text{ bytes} \in \text{GF}(2^8)^{32}$ ) with 512-bit blocks. The round function applies four operations in sequence:

- 1) **Inject:** XOR the state with round-key material

derived from the message block and a round constant (SHA-256 of a domain-separation string).

- 2) **Substitute:** Apply the S-box byte-wise. Default:

the AES S-box ( $x-1$  in  $\text{GF}(28)$  followed by an affine transformation), which has nonlinearity 112, differential uniformity 4, and algebraic degree 7.

- 3) **Diffuse:** Multiply the state by a  $32\times 32$  matrix over  $\text{GF}(28)$ . Default: a Cauchy MDS matrix with branch number 33 (the theoretical maximum for this size).

- 3) **Permute:** Shuffle byte positions using a ShiftRows-style permutation on a  $4\times 8$  grid.

The field is  $\text{GF}(28)$  with the AES irreducible polynomial  $x^8 + x^4 + x^3 + x + 1$ . The IV is the first 32 bytes of the fractional part of  $\pi$ .

Every design choice is a tunable parameter: round count (4-16), matrix type (Cauchy MDS, circulant, Toeplitz, random), S-box type and parameters, and permutation pattern. This parameter space is what the evolutionary defender searches.

The Cauchy construction builds  $M[i][j] = (x_i + y_j)^{-1}$  in  $\text{GF}(2^8)$ , where the sets  $\{x_i\}$  and  $\{y_j\}$  are disjoint. Every square

submatrix of a Cauchy matrix is invertible, which guarantees the MDS property and maximum branch number.

#### B. Neural adversary

We follow Gohr’s methodology [2] adapted for our hash. The distinguisher receives pairs of 256-bit hash outputs and classifies them as “real differential pair” (both outputs derived from inputs differing by a fixed  $\Delta$ ) or “random pair” (independent random strings). The pair is concatenated into a 512-dimensional binary vector.

The architecture is a ResNet with an initial linear layer (512  $\rightarrow$  256, ReLU), followed by 4 residual blocks (BatchNorm  $\rightarrow$  Linear  $\rightarrow$  ReLU  $\rightarrow$  Linear  $\rightarrow$  skip connection, all 256-wide), and an output head (BatchNorm  $\rightarrow$  Linear 256  $\rightarrow$  64  $\rightarrow$  ReLU  $\rightarrow$  Linear 64  $\rightarrow$  1  $\rightarrow$  Sigmoid). Total: 677K parameters. We also train a smaller MLP variant (173K parameters) for faster evaluation during evolution.

Training uses 30,000 pairs per configuration, binary cross entropy loss, and Adam with learning rate  $10^{-3}$ . Differential pairs use single-bit input differences; we sweep over multiple bit positions to find the worst case.

The security margin is the round count at which distinguishing accuracy drops to  $\approx 50\%$ . For the default configuration (Cauchy MDS, AES S-box, 8 rounds), accuracy is  $\approx 50\%$  from round 1—the hash is indistinguishable from random even at a single round. This is not a bug; the  $32 \times 32$  MDS matrix with branch number 33 produces complete diffusion in one round.

#### C. Walsh spectral engine

For an S-box  $S : \text{GF}(2)^8 \rightarrow \text{GF}(2)^8$ , the Walsh-Hadamard transform gives:

$$W_S(a, b) = \sum_{x \in \{0,1\}^8} (-1)^{\langle b, S(x) \rangle \oplus \langle a, x \rangle} \quad (1)$$

for input mask  $a$  and output mask  $b$ . The result is a  $256 \times 256$  matrix of coefficients. From this we compute a spectral security fingerprint:

- Nonlinearity:  $NL = (2^n - \max_{b \neq 0} \max_a |W_S(a, b)|) / 2$
- Differential uniformity:  $\max_{\Delta \neq 0, \nabla} |\{x : S(x \oplus \Delta) \oplus S(x) = \nabla\}|$
- Algebraic degree of coordinate functions
- Spectral flatness: geometric mean of  $|W|^2$  divided by arithmetic mean
- Spectral entropy:  $H = \sum p(w) \log_2 p(w)$

These quantities are computed deterministically in milliseconds. No training is needed. The fingerprint is an 8

dimensional vector that characterizes resistance to linear and differential cryptanalysis.

#### D. Evolutionary defender

The defender uses a genetic algorithm (DEAP library) to search the hash parameter space. Each individual’s genome encodes: round count, matrix seed, S-box type, and permutation variant. Fitness is a weighted sum:

$$f = 0.4 \cdot S_{\text{neural}} + 0.3 \cdot S_{\text{avalanche}} + 0.2 \cdot S_{\text{stats}} + 0.1 \cdot S_{\text{speed}} \quad (2)$$

where  $S_{\text{neural}}$  is derived from the neural distinguisher (lower accuracy = higher score),  $S_{\text{avalanche}}$  measures deviation from the ideal 0.5 flip probability,  $S_{\text{stats}}$  is the NIST test pass rate, and  $S_{\text{speed}}$  is throughput. Population size is 20, selection is tournament with  $k = 3$ , crossover swaps matrix parameters between parents, and mutation perturbs one parameter at random. We ran 8 generations. Total runtime: 81 minutes on CPU (AMD Ryzen 5).

#### E. Co-evolution loop

The controller runs the following cycle per generation:

- 1) Instantiate each candidate hash configuration.
- 2) Run statistical tests (avalanche, uniformity, BIC) on 5,000 samples.
- 3) Train a neural distinguisher (reduced training : 3 epochs, 30K pairs).
- 4) Compute spectral fingerprints.
- 5) Evaluate fitness, select survivors, generate offspring. The loop tracks whether the security margin increases over generations.

## 4. EXPERIMENTS AND RESULTS

#### A. Statistical Evaluation

On the default configuration (Cauchy MDS, AES S-box, 8 rounds):

TABLE I: Statistical Test Results for MatrixHash-256

| Test                   | Result         | Threshold       |
|------------------------|----------------|-----------------|
| Avalanche Mean         | 0.5000         | $0.50 \pm 0.05$ |
| BIC Violations         | 0 / 1,044,840  | $< 0.05$ corr.  |
| NIST SP 800-22         | 10/10 pass     | All pass        |
| Chi-squared (per byte) | All $p > 0.01$ | $p > 0.01$      |

| Test                                   | Result | Threshold |
|----------------------------------------|--------|-----------|
| Full Collisions (256-bit, 220 samples) | 0      | 0         |

The avalanche result deserves comment. A mean flip probability of exactly 0.5000 across all input-output bit pairs is unusually good. It is a direct consequence of the MDS matrix: with branch number 33, every non-zero input difference activates all 32 output bytes after a single round. By round 8, the diffusion is thoroughly complete.

### B. Neural Distinguisher Results

For the default configuration, both the ResNet (677K) and MLP (173K) distinguishers achieve accuracy  $\approx 50\%$  at all round counts from 1 to 8. The hash is indistinguishable from random.

We then swept across S-box configurations with varying nonlinearity (NL), keeping the matrix and round count fixed. The results reveal a sharp threshold at  $NL \approx 80$ :

- Below  $NL = 80$ : Neural accuracy approaches 100%. The adversary easily detects differential structure.
- Above  $NL = 80$ : Neural accuracy drops to  $\approx 50\%$ . The adversary learns nothing.

The transition is not gradual. A binary classifier trained to predict “breakable vs. not” using only the  $NL = 80$  threshold achieves  $AUC = 1.0$ . We call this the “security cliff.”

### C. Spectral-Neural Regression

We fit a regression model predicting neural distinguisher accuracy from the spectral fingerprint across all tested configurations. The initial result is striking:  $R^2 = 0.998$ .

But this number is misleading. The identity S-box ( $NL = 0$ ) is a massive outlier—it produces near-perfect adversary accuracy and sits far from every other configuration in spectral space. Remove it, and  $R^2$  drops to 0.059.

This is not a failure of the spectral analysis; it is a genuine finding about the structure of the security landscape. The spectral fingerprint is a perfect predictor of catastrophic weakness ( $NL < 80$ ) but carries no discriminative power among secure configurations ( $NL \geq 80$ ). The relationship between algebraic properties and neural resistance is binary, not continuous.

Key takeaway: The spectral engine is a smoke detector, not a thermometer. It reliably catches fire but cannot tell you the room temperature.

### D. Co-Evolution Progress

Over 8 generations, the best S-box nonlinearity in the population rose from 94 to 112. The system converged on the **AES S-box**—the same construction designed by hand by Rijmen and Daemen. This took 81 minutes on a consumer CPU.

TABLE II: Co-Evolution Progress (8 Generations)

| Gen | BestNL | Avg. Avalanche | Neural Acc. |
|-----|--------|----------------|-------------|
| 0   | 94     | 0.4812         | 68.2%       |
| 1   | 98     | 0.4891         | 61.7%       |
| 2   | 102    | 0.4923         | 55.3%       |
| 3   | 104    | 0.4954         | 53.1%       |
| 4   | 108    | 0.4971         | 51.2%       |
| 5   | 110    | 0.4988         | 50.4%       |
| 6   | 112    | 0.4996         | 50.1%       |
| 7   | 112    | 0.4999         | 50.0%       |

The convergence to  $NL = 112$  is consistent with the cliff finding: evolutionary pressure from neural-in-loop fitness drives the S-box nonlinearity past the threshold at  $NL \approx 80$ , after which further improvement in neural resistance is impossible. However, avalanche and statistical fitness components continue to pull the design toward the AES S-box specifically.

### E. CryptoGenesis: Quasigroup Search

We searched 1,000 quasigroup structures of order 256 for cryptographic suitability, evaluating each for nonlinearity, differential uniformity, algebraic degree, and absence of zero-annihilation ( $a \circ 0 \neq a$  for all  $a$ ).

Every structure that passed the zero-annihilation filter achieved  $NL = 112$  (the AES value). No structure exceeded it. Within the family of affine quasigroups over  $GF(2^8)$ , the AES algebraic structure appears to be optimal rather than merely one good choice among many.

### F. RL Collision Finder

A **DQN-based agent** was trained on reduced-round configurations:

- **State:** Bit vectors of candidate pair.
- **Action:** Flip one bit in either input.
- **Reward:** Negative Hamming distance between hash outputs.

On the default 8-round hash, the agent could not beat the **birthday bound**, consistent with the statistical and neural evaluations. On 2-round configurations, the agent found **near-collisions** (Hamming distance  $< 10$  on 256-bit output) within 500 episodes.

#### G. Implementation Performance

A C implementation of the hash core (matrixhash.c), compiled with gcc -O2 and accessed via ctypes from Python, achieves a  $105\times$  speedup over the pure Python version. The C library is compiled and deployed on the Render backend.

## 5. DISCUSSION

### Is MatriXHash-256 a New Hash Function?

No. It follows the same SPN paradigm as AES, Grøstl, and Whirlpool. The mixing layer is matrix multiplication over a finite field; the nonlinear layer is a byte-wise S-box; the structure is Merkle-Damgård with Davies-Meyer. The linear algebra involved is  $GF(2^8)$  arithmetic and the Walsh-Hadamard transform (which is the Fourier transform on  $(\mathbb{Z}/2\mathbb{Z})^n$ , not eigenvalue decomposition over  $\mathbb{R}$ ). We do not claim novelty for the hash itself. The hash is a well-parameterized substrate; the contribution is the co-evolution system built around it.

### Are Self-Generated Differential Pairs Valid?

Yes. Gohr's original methodology [2] and all subsequent neural cryptanalysis work generate training data by running the cipher on chosen inputs. There is no external dataset to use—the hash function is the data source.

The differential pairs are methodologically correct, not a shortcut.

### Why Does Neural Accuracy Sit at 50% from Round 1?

The MDS matrix has branch number 33, meaning every non-zero input difference produces output differences spanning all 32 bytes. Combined with the AES S-box ( $NL = 112$ ), even a single round produces output that is statistically indistinguishable from random with respect to the differential patterns the neural network can detect.

This is correct behavior for a well-designed hash, not a pipeline bug.

**Limitations.** The co-evolution ran for only 8 generations with population 20 on consumer hardware. Larger populations and more generations might reveal finer-grained dynamics.

The spectral-neural regression result is limited by the threshold structure of the security landscape—it works as a binary classifier but not as a continuous predictor.

The hash itself should not be used in production; SHA-256 and BLAKE3 exist for that purpose.

## 6. CONCLUSIONS

CryptoForge demonstrates that adversarial co-evolution is a viable approach to automated hash function design. A neural attacker finds weaknesses; a spectral engine provides cheap algebraic estimates; an evolutionary defender modifies the design; the cycle repeats. Over 8 generations, the system independently converged on the AES S-box. The binary security cliff at  $NL \approx 80$  and the  $R^2$  collapse are empirical findings about the structure of the design space that, to our knowledge, have not been previously reported. The source code, trained models, web interface, and all 17 result files are publicly available.<sup>1</sup>

## ACKNOWLEDGEMENT

The authors would like to express their sincere gratitude to their project guide for continuous guidance, valuable suggestions, and support throughout the course of this work. The authors also thank the Department of Artificial Intelligence and Data Science and the institution for providing the necessary resources and facilities.

## REFERENCES

- [1] X. Wang, Y. L. Yin, and H. Yu, "Finding collisions in the full SHA-1," in *Advances in Cryptology—CRYPTO 2005*, LNCS 3621, Springer, 2005, pp. 17–36.
- [2] A. Gohr, "Improving attacks on round-reduced Speck32/64 using deep learning," in *Advances in Cryptology—CRYPTO 2019*, LNCS 11693, Springer, 2019, pp. 324–352.
- [3] A. Benamira, D. Gerault, T. Peyrin, and Q. Q. Tan, "A deeper look at machine learning-based cryptanalysis," in *Advances in*

*Cryptology—EUROCRYPT 2021*, LNCS 12696, Springer, 2021, pp. 805–835.

[4] S. Picek, K. Knezevic, D. Jakobovic, and J. F. Miller, “Evolving S-boxes with improved cryptographic properties,” in *Proc. Genetic and Evolutionary Computation Conf. (GECCO)*, ACM, 2016, pp. 409–410.

[5] C. Carlet, *Boolean Functions for Cryptography and Coding Theory*. Cambridge University Press, 2010.

[6] J. Daemen and V. Rijmen, *The Design of Rijndael: AES—The Advanced Encryption Standard*. Springer, 2002.

[7] C. Lu, C. Lu, R. T. Lange, J. Foerster, J. Clune, and D. Ha, “The AI Scientist: Towards fully automated open-ended scientific discovery,” arXiv:2408.06292, 2024.

[8] A. Rukhin et al., “A statistical test suite for random and pseudorandom number generators for cryptographic applications,” NIST Special Publication 800-22, Rev. 1a, 2010.

[9] M. Matsui, “Linear cryptanalysis method for DES cipher,” in *Advances in Cryptology—EUROCRYPT ’93*, LNCS 765, Springer, 1993, pp. 386–397.

[10] E. Biham and A. Shamir, “Differential cryptanalysis of DES-like cryptosystems,” *J. Cryptology*, vol. 4, no. 1, pp. 3–72, 1991.