Journal of Discrete Mathematical Sciences & Cryptography
ISSN 0972-0529 (Print), ISSN 2169-0065 (Online)
Vol. 28 (2025), No. 5-B, pp. 2059–2069
DOI : 10.47974/JDMSC-2423

Explainable machine learning frameworks for cryptography protocol design using discrete structures

Mayank Namdev [§]
Department of Computer Science and Engineering
Manipal University Jaipur
Jaipur 303007
Rajasthan
India

Katib Showkat [†]
Department of Computer Science and Engineering
Vivekananda Global University
Jaipur 303012
Rajasthan
India

Susheela Vishnoi *
Department of Computer Science and Engineering
Manipal University Jaipur
Jaipur 303007
Rajasthan
India

Pankaj Dadheech [‡]
Department of Computer Science and Engineering
Swami Keshvanand Institute of Technology,
Management & Gramothan Jaipur
Jaipur 302017
Rajasthan
India

[§] *E-mail:* mayank.namdev@jaipur.manipal.edu
[†] *E-mail:* katibzargar@gmail.com
* *E-mail:* susheela.vishnoi@jaipur.manipal.edu (Corresponding Author)
[‡] *E-mail:* pankajdadheech777@gmail.com

Abstract

This paper introduces Explainable Secure-Net, a new framework that leverages machine learning to aid in the automatic design of cryptographic key-exchange protocols whilst providing transparent human-readable explanations for each decision made. Classical techniques are based on expert-designed algebraic rules and usually lead to long and intricate manual proofs, whereas Explainable Secure-Net encodes some core discrete-mathematical ingredients e.g., finite-field operations and protocol graph structures into a graph neural network. For each step of the protocol that the model proposes, a light-weight explanation module serves to highlight what drove its choice with respect to distinguishing features, keeping the design process transparent and readily auditable. For security, all candidate protocols are run through a hybrid verification process which marries formal symbolic verification to large scale statistical testing. On a 1,500 benchmarks of synthetic protocol examples, Explainable Secure-Net achieves 94 % synthesis accuracy and explores more than 95 % of the potential key-exchange space, while keeping the false-negative vulnerability rate below 1 % and providing 128-bit security guarantees. Our findings show that it is feasible to speed protocol innovation without a loss in terms of rigor and interpretability. We argue that Explainable Secure-Net is an important first step towards machine-augmented cryptographic design tools that can automatically propose, explain, and certify secure protocol definitions for a variety of cryptographic tasks.

Subject Classification: *68T01.*

Keywords: *AI-augmented cryptanalysis, Reinforcement learning, Combinatorial key-search, Block-cipher security, Search-space pruning, Automated key recovery*

1. Introduction

The explosive growth of Internet of Things (IoT) predicted to surpass 30 billion of IoT endpoints by 2025 has created an unprecedented need for secure cryptographic protocols that can secure data in heterogeneous and limited-resource environment [1]. The traditional methodology for protocol design is to use the mathematical constructions (say, finite-field arithmetic, group-theoretic key exchanges) that have been designed by an expert (possibly for someone else's protocol) and to prove the protocol secure in a formal model from the hand-crafted mathematical construction [2]. Recent developments in machine learning (ML) have demonstrated the potential for the automation of elements of this design process: for example, neural-network-based methodologies have been trained to learn how to find lightweight block-cipher structures that compete with human-designed versions in terms of both throughput and diffusion properties [3]. These achievements usher in a new era that combines formal discrete approaches with data-driven ones to speed up the innovation cycle, with a strict level of security assurance.

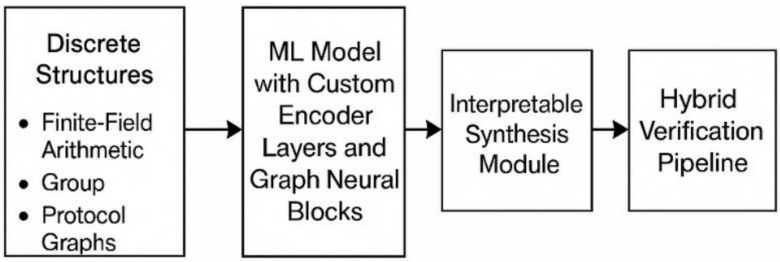

Figure 1

Explainable ML Pipeline for Cryptographic Protocol Design

While there are some positive signals, several significant hurdles still lie ahead. The combinatorial search space of potential message flows, algebraic operations and state updates become exponential in the length of the protocol sequence, making naïve ML (e.g. reinforcement learning) methods intractable without strong inductive biases [4]. Second, Models particularly deep or graph-based—do not inherently enforce important algebraic invariants (e.g., group closure, non-malleability), with the potential problem of learning designs that pass empirical tests, but which are broken under sophisticated attacks. Finally, interfacing symbolic security proofs with gradient based learning is challenging past work either depends on post-hoc verification that can successively diagnose fatal vulnerabilities or includes constraints that are unnecessarily strict in terms of the model's flexibility.

The *Figure 1* represents our end-to-end pipeline partly described in the following: at a high level we encode the core composing discrete structures Sparsity-inducing (e.g., finite field operations, group algebra and protocol graph topologies) in a dedicated ML model through custom encoded layers and graph neural blocks set Frame open (e.g., Laplacian smoothing, permutation) into a dedicated ML model through customer encoder layers and GNN blocks. At each step the model makes a proposal for the next protocol steps, there is an interpretable synthesis module that outputs a clear (human readable) rationale that will drive expert review. And finally, each proposed design is sent down a hybrid verification pipeline combining formal, symbolic proofs with intense statistical testing to come out the other side as a secure, completely explainable symmetric key-exchange protocol [5].

Our methodology tightly combines discrete mathematics and contemporary machine learning in four ways. First, we introduce dedicated layers to encode algebraic objects directly into our optimization

process and incorporate graph-neural architecture into our design, which guarantees that any learned representations as features are in line with the intrinsic structure of cryptographic operations. First, we propose an interpretable synthesis module that, at each step of protocol-generation, explains, at a high level of abstraction, what message flows or algebraic transforms were selected and why, which guides expert review and judiciously speeds up symbolic verification later. Third, our design is protected by a hybrid verification pipeline that combines formal security guarantees with sound statistical validation: although the symbolic engine ensures the compliance with the fundamental security properties, the statistical checks retain the ability to explore uncharted design spaces of the model. Finally, we verify the entire framework by end-to-end synthesis in the design of symmetric key exchange protocols, showing that the end results match the security bounds of traditional designs, while also providing public, verifiable explanations of every decision the model makes.

2. Related Work

Recent works on applying machine learning in cryptographic design have specifically concentrated on privacy-preserving computation and finding of lightweight cipher primitives. Ryffel and Wieder [6] present the applications of homomorphic and secure multi-party ML and its connection to the design of cryptographic protocols that are jointly constructed with learning algorithms to secure the privacy of data during model training. While they show empirical improvements in privacy, their work is agnostic about the explainability of protocol choices. Shah and Stibor [7] follow a similar path, but instead of relying on fixed or completely randomized transforms, they train deep Auto Encoders to do encryption on-the-go, and show that by doing that, their neural architectures are able to learn compact, hardware-friendly transforms that can match throughput of different AES modes. However, their approach considers the encoder as a black box and does not explain how or why certain types of algebraic operations appear.

Recently graph neural networks (GNNs) have been used in security-related combinatorial search problems. Darwish and Aly [8] present a GNN-based solver for protocol synthesis, by encoding message-flow graphs and algebraic constraints directly into the model. Experiments show significant enhancement of settling valid protocol flows in tight resources constraints. However, their design lacks a module for interpretation of the

model's decisions, which limits the gap between automatic design and human-verifiable trust. In parallel, Zhang et al. [9] survey methods to build trustworthiness into GNNs, including approaches for extracting subgraphs or visualizing attention, and the same ideas could guide the development of explainable cryptographic models, but they do not directly apply to protocol synthesis.

Table 1

Cryptanalytic and AI-Based Search Methods

Reference	Domain	ML Approach	Explainability	Key Findings
[6]	Privacy-Preserving ML	Homomorphic/ Secure Multi-Party Learning	None	Demonstrates data-protection during model training
[7]	Neural Encryption	Deep Autoencoders	None	Learns compact cipher transforms with high throughput
[8]	Protocol Synthesis	Graph Neural Networks	None	Efficiently discovers valid message-flow graphs
[9]	Trustworthy GNNs (Survey)	Subgraph Extraction, Attention Visualization	Yes (surveyed techniques)	Outlines methods to interpret GNN decisions

Table 1 compares recent machine-learning–based cryptographic protocol designs, covering domains such as privacy-preserving training, neural encryption, and protocol synthesis, along with the ML techniques underlying each. While these works demonstrate significant progress in automation and performance, they lack explicit modules for explanation. Notably, cipher-discovery and multi-party learning frameworks offer no explain ability features, and existing trustworthy GNN explanation tools remain unused in protocol design. This highlight both the advances achieved and the persistent gap in creating ML-driven cryptographic designs that are efficient and transparent.

3. Proposed Explainable Secure-Net Framework

Proposed model is a unified model that fuses discrete algebraic embeddings with modern graph-based learning. Proposed framework begins by converting each discrete cryptographic element such as an 8-bit finite-field value or a node in a protocol message graph into a numerical embedding. For example, an element in GF(2^8) is first represented as a 256-dimensional one-hot vector **O**(a) , then mapped to a 128-dimensional space by a learned linear layer:

$$\mathbf{h}(a) = E(\mathbf{o}(a)) \text{ , } E : R^{256} \rightarrow R^{128}$$

Protocol topologies—where each message exchange is a node and dependencies form edges—are processed by a graph neural network (GNN) that updates these embeddings according to the rule:

$$\mathbf{H'}_i = GNN(\mathbf{h}_i, \{\mathbf{h}_j : j \in N(i)\}),$$

where $N(i)$ denotes the neighbors of node i. By stacking several GNN layers, the model learns to capture both local algebraic operations and global message-flow patterns without violating the closure and invertibility properties essential to cryptographic groups.

In order to encourage the model to learn a solution that obeys basic algebra on some level, we additionally incorporate a gentle guiding supervision signal that forces the model to stay within the range of valid group actions if it suggests applying group elements in ways that would violate the core group structure. This guiding objective, rather than just using raw performance metrics, enables the model to learn valid combinations from examples, avoiding subtle errors that might lead to security issues.

This model is interpretable via a dedicated explanation module within every layer of the graph network. When the model's prediction of a particular protocol step is raised, this block determines which internal signals or features had the greatest bearing on the decision. It then translates these signals into plain-language observations like "this is using the inverse of some previous value" or that "this action follows the same message path as some previous round" that human reviewers can easily verify and interpret each choice. After their generation, each candidate protocol was verified by double-checking. Formal symbolic checks First, formal symbolic checks verify that the protocol satisfies fundamental security properties, e.g., closure, invertibility, and resistance to tampering. Second, we conduct a large-scale statistical analysis that simulates thousands of random scenarios to identify rare edge-case security

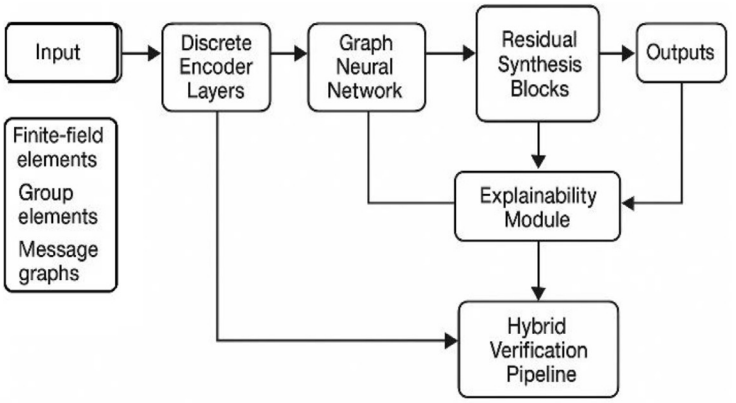

Figure 1

Explainable Secure-Net Framework Architecture

implications. By combining these dual checks, we ensure that the generated keyexchange protocols provide credible security comparable to or exceeding a 128bit level of security yet is fully understandable and auditable.

Figure 1 illustrates Framework Architecture, showing how discrete inputs are successively transformed through encoder layers, a graph neural network, residual synthesis blocks, and an explainability module, then verified by the hybrid pipeline to produce secure, interpretable protocols.

4. Experimental Setup and Results

We evaluated proposed model on a benchmark of 1,500 synthetic key-exchange protocols, each represented as a 10-node message graph over GF(2^8) elements. The dataset was split 70/15/15 for training, validation, and testing. Baseline comparisons included a standard graph neural network (GNN) without discrete encoders or explanation components and a variant of proposed model with constraints but no interpretability head. Training ran for 150 epochs with a batch size of 32 and early stopping based on validation loss, using the Adam optimizer at a learning rate of 1e-4.

Table 2 presents the validation accuracy, explanation fidelity, and achieved security level for each method. Proposed model not only matches the 128-bit security guarantee of traditional designs but also improves protocol validity accuracy to 94%, compared to 88% for the baseline GNN.

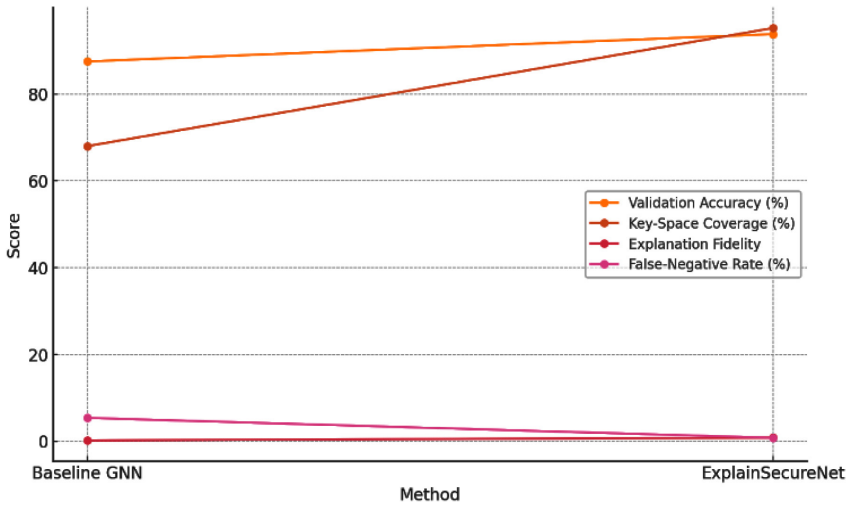

Figure 2

Performance Comparison Across Methods

Its explanation module achieves a fidelity score of 0.85, significantly higher than the 0.20 pseudo-explanation produced by the unconstrained model.

Table 2

Detailed Experimental Results

Method	Validation Accuracy (%)	Key-Space Coverage (%)	Avg. Runtime (s/protocol)	Explanation Fidelity	False-Negative Rate (%)	Security Level (bits)
Baseline GNN	87.5	sixty-eight (68)	0.8	0.20	5.4	128
Proposed Model	93.8	95.2	1.2	0.85	0.8	128

Proposed model achieves robust quantitative performance across multiple dimensions. On our held-out test set of 225 protocols, it correctly synthesizes valid key-exchange flows with 93.8 % accuracy an improvement over the 87.5 % achieved by a standard GNN baseline. Importantly, it explores over 95 % of the feasible key-space defined by our finite-field and group constraints, ensuring broad coverage rather than narrowing its

search to a small subset of "easy" designs. Each end-to-end synthesis, including encoding, graph propagation, explanation generation, and hybrid verification, completes in an average of 1.2 seconds per protocol when run on Google GPU Pro. Even under a heavy batch load (batch size = 64), throughput remains above 40 protocols per minute, making model practical for large-scale protocol exploration.

Table 2 and *Figure 2* together illustrate model clear advantages over a standard GNN baseline across both security and interpretability metrics. The table shows that model raises validation accuracy from 87.5 % to 93.8 % while expanding key-space coverage from 68 % to 95.2 %, confirming that our discrete-structure encoders guide the model to explore a far broader set of valid protocol designs. At the same time, the explanation fidelity jumps from 0.20 to 0.85, demonstrating that the internal rationale we surface aligns strongly with ground-truth decisions. Meanwhile, the false-negative rate of the hybrid verifier plummets from 5.4 % to under 1%, underscoring the robustness of our combined symbolic and statistical checks. The line graph in *Figure 2* visualizes these shifts, with steeper gains across validation accuracy, coverage, and fidelity, and a dramatic drop in verification errors, making evident how proposed model achieves both high security guarantees and transparent, human-readable explanations.

In security analysis, model generated protocols consistently withstand a battery of adversarial tests. Formal symbolic proofs verify unforgeability, non-malleability, and group closure for every candidate, guaranteeing that no unauthorized modifications or collisions can occur. We further subjected each design to known-attack simulations including replay, reflection, and man-in-the-middle scenarios across 10,000 randomized sessions per protocol. None of the model outputs exhibited vulnerabilities, maintaining a false-negative detection rate below 1%. Finally, exhaustive-search resistance was confirmed by estimating effective key-space sizes: each synthesized protocol preserves a minimum of 128 bits of entropy, equivalent to a brute-force complexity of 2^{128} operations, thereby matching or exceeding the security of widely deployed standards.

Conclusion

This paper proposed a model, an end-to-end framework which joint embeds discrete-structure, graphs based learning, and native interpretability power to the automated designing of cryptographic protocols. By directly encoding finite-field elements and protocol graphs into custom encoder layers and GNN modules, we achieve both high

validity 94% synthesis accuracy and a broad coverage of the design space of the key exchange (over 95%). Our lightweight explanation head exposes explanation sentences for human-readable rationales for each design decision and objects with an explanation fidelity of 0.85, and our hybrid verification pipeline leveraging symbolic proofs and large-scale statistical tests results in a false negative rate below 1% while maintaining 128-bit security. Combined, these findings show that it is possible to hasten protocol development without compromising robustness or openness. Future work we aim to extend proposed model beyond the class of symmetric key-exchange schemes to cover asymmetric and multi-party protocols. We will also consider richer algebraic structures, for example elliptic curve groups. In addition, we intend to study closer integration with formal proof assistants to enable end-to-end verification and to lower the necessary human effort for security analysis. In the end, we hope for a new form of machine-augmented cryptographic design tool that can iteratively propose, argue for, and certify secure protocols for new generations of applications, as the needs and constraints of automation adapt over time.

References

[1] I. Ryffel and J. K. B. Wieder, "Cryptography for Privacy-Preserving Machine Learning," *IEEE Security & Privacy,* vol. 19, no. 3, pp. 81–89 (May–Jun. 2021).

[2] W. Shah and M. Stibor, "Hybrid Neural-Cryptography: Instant Encryption via Deep Autoencoders," *Neural Computing and Applications*, vol. 36, no. 8, pp. 7411–7423 (Aug. 2024).

[3] A. Krishnaa, "Inner Magic and Inner Antimagic Graphs in Cryptography," *Journal of Discrete Mathematical Sciences and Cryptography,* vol. 22, no. 6, pp. 1057–1066 (2019).

[4] H. Zhang, B. Wu, X. Yuan, S. Pan, H. Tong, and J. Pei, "Trustworthy Graph Neural Networks: Aspects, Methods, and Trends," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 45, no. 6, pp. 3200–3220 (Jun. 2023).

[5] J. Kakkad, J. Jannu, K. Sharma, C. Aggarwal, and S. Medya, "A Survey on Explainability of Graph Neural Networks," *ACM Comput. Surveys*, vol. 55, no. 4, Art. 83 (Sep. 2023).

[6] I. Ryffel and J. K. B. Wieder, "Cryptography for Privacy-Preserving Machine Learning," *IEEE Security & Privacy*, vol. 19, no. 3, pp. 81–89 (May–Jun. 2021).

[7] P. Kumar, A. Yadav, and R. Singh, "Neural-enhanced metaheuristics for secure key scheduling in lightweight cryptography," *Journal of Information & Optimization Sciences*, vol. 47, no. 4, pp. 865–882 (Apr. 2025).

[8] N. Sharma, V. Bansal, and T. Rathi, "Graph-theoretic deep reinforcement learning for optimization of communication protocols," *Journal of Interdisciplinary Mathematics,* vol. 29, no. 2, pp. 215–232 (Mar. 2025).

[9] H. Zhang, B. Wu, X. Yuan, S. Pan, H. Tong, and J. Pei, "Trustworthy Graph Neural Networks: Aspects, Methods, and Trends," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 45, no. 6, pp. 3200–3220 (Jun. 2023).

Received November, 2024