CRYPTOSCOPE: UTILIZING LARGE LANGUAGE MODELS FOR AUTOMATED CRYPTOGRAPHIC LOGIC VULNERABILITY DETECTION

Zhihao Li[1], *Zimo Ji*[2], *Tao Zheng*[1], *Hao Ren*[1], *Xiao Lan*[1,*]

[1]Sichuan University, Chengdu, China
[2]The Hong Kong University of Science and Technology, Hong Kong SAR, China

ABSTRACT

Cryptographic algorithms are fundamental to modern security, yet their implementations frequently harbor subtle logic flaws that are hard to detect. We introduce CRYPTOSCOPE, a novel framework for automated cryptographic vulnerability detection powered by Large Language Models (LLMs). CRYPTOSCOPE combines Chain-of-Thought (CoT) prompting with Retrieval-Augmented Generation (RAG), guided by a curated cryptographic knowledge base containing over 12,000 entries. We evaluate CRYPTOSCOPE on LLM-CLVA, a benchmark of 92 cases primarily derived from real-world CVE vulnerabilities, complemented by cryptographic challenges from major Capture The Flag (CTF) competitions and synthetic examples across 11 programming languages. CRYPTOSCOPE consistently improves performance over strong LLM baselines, boosting DeepSeek-V3 by 11.62%, GPT-4o-mini by 20.28%, and GLM-4-Flash by 28.69%. Additionally, it identifies 9 previously undisclosed flaws in widely used open-source cryptographic projects.

Index Terms— Cryptographic logic vulnerabilities, Large language models, Retrieval-Augmented Generation, Chain-of-Thought

1. INTRODUCTION

Cryptographic algorithms and protocols are fundamental to securing computer systems, offering confidentiality, integrity, and authentication based on strong mathematical foundations. However, translating these principles into correct implementations remains challenging and error-prone [1]. Developers must implement algorithms accurately, handle inputs properly, select parameters carefully, and optimize performance, with mistakes potentially compromising entire systems. Furthermore, insufficient cryptographic expertise and the increasing use of large language models (LLMs) like GPT-3.5 [2] for coding assistance may introduce subtle vulnerabilities. Such flaws in widely used cryptographic libraries can propagate to numerous dependent projects, as exemplified by the critical ECDSA bypass vulnerability (CVE-2022-21449) [3] in Oracle Java SE and GraalVM, which allowed attackers to forge digital signatures and bypass authentication.

Existing automated detection efforts largely target cryptographic API misuse [4–8]. In contrast, only a few studies [9–12] have explored the automated detection of cryptographic logic flaws [1], which often suffer from limited automation, strong language dependencies, and restricted generalizability.

To address this gap, we propose CRYPTOSCOPE, the first LLM-based framework for detecting cryptographic logic vulnerabilities. It initiates with a pre-detection step verifying algorithm correctness and employs few-shot learning [13] combined with Chain-of-Thought (CoT) prompting [14] to guide the LLM in analyzing code parameters and logic. We further build a cryptographic knowledge base by extracting diverse, multi-source domain information and integrate relevant knowledge via Retrieval-Augmented Generation (RAG) [15] to enhance reasoning accuracy. Detection results are output in a structured, developer friendly format.

We evaluate CRYPTOSCOPE on LLM-CLVA, a 92-sample benchmark covering real-world vulnerabilities, Capture The Flag (CTF) [16] challenges, and synthetic cases in 11 programming languages, using the LLM-as-a-Judge [17] framework across six representative LLMs [18–23]. Deployed on 20 open-source projects, CRYPTOSCOPE discovered 9 previously unknown cryptographic flaws, demonstrating its practical effectiveness.

The contributions of this work can be summarized as follows.

- **Benchmark:** LLM-CLVA, comprising 92 multi-language cryptographic vulnerability samples with manual reports and comprehensive evaluation metrics.
- **Framework:** CRYPTOSCOPE, a language-agnostic LLM-based system leveraging CoT and RAG for cryptographic logic vulnerability detection without code execution.
- **Empirical validation:** Strong experimental gains across architectures, validated by ablations, real-world discoveries, and improved human analysis through knowledge augmentation.

*Corresponding Author: Xiao Lan

2. RELATED WORK

Automated detection of cryptographic logic vulnerabilities generally falls into two main categories: test vector–based validation and fuzzing. Project Wycheproof [9], developed by Google, provides a comprehensive suite of curated test vectors targeting known issues in cryptographic algorithms. It includes over 80 test cases and has helped uncover more than 40 implementation bugs. However, its use across different programming languages requires custom parsers and test harnesses, which can limit portability and scalability.

Fuzzing-based approaches have also gained traction. DifFuzz [11] identifies side-channel vulnerabilities by generating inputs that maximize differences in resource consumption between program variants. CDF [10] integrates fuzzing with stateless test vectors to explore known edge cases in cryptographic operations. Cryptofuzz [24] employs differential testing across cryptographic libraries by comparing algorithm outputs to detect inconsistencies, and, with the help of sanitizers, can also reveal memory-related issues. However, many fuzzing-based techniques depend on triggering specific failure conditions and require manual inspection of anomalous behaviors to assess the underlying flaw.

3. METHOD

3.1. Construction of the LLM-CLVA Benchmark

In our preliminary research, we developed a novel benchmark, LLM-CLVA (**LLM** for **C**ryptographic **L**ogic **V**ulnerability **A**nalysis), to address the absence of specialized benchmarks for evaluating LLMs in detecting cryptographic logic vulnerabilities. Our dataset comprises:

- Code samples of cryptographic logic vulnerabilities from CVE entries (57%).
- High-quality cryptographic challenges from major international CTF competitions (30%).
- Artificially constructed algorithm implementations violating cryptographic standards (13%).

For each code snippet in the dataset, we conducted manual auditing to summarize the cryptographic logic vulnerabilities present in the code, which served as the ground truth. To comprehensively assess model performance, we defined four evaluation metrics:

- *Credibility Score*: A composite metric assessing relevance, informativeness, and logical soundness of reasoning. It serves as the primary indicator of model performance.
- *Cosine Similarity*: Measures semantic similarity between generated and reference reasoning via sentence embeddings (MiniLM-L6-v2), with scores in [0,1].
- *Semantic Match Rate*: Assesses semantic consistency using LLM-as-a-Judge to determine alignment with the reference. Scores range from 0 to 1.
- *Coverage Score*: Estimates the proportion of informative and relevant content in the output, judged by LLM-as-a-Judge.

3.2. The Architecture of CRYPTOSCOPE

In this work, we present a novel LLM-based cryptographic vulnerability detection framework CRYPTOSCOPE, the main idea of which is to leverage the semantic comprehension ability and the reasoning ability of LLM to simulate the process of cryptanalysts analyzing vulnerabilities. We summarize the process of manually analyzing cryptographic logic vulnerabilities into the following steps: understanding the semantics of the code, verifying its compliance with cryptographic algorithm standards, examining the code for potential vulnerabilities by referencing established vulnerability categories, and leveraging knowledge from vulnerability databases and best practices for cryptographic algorithm implementation to identify issues in the code. Accordingly, we applied this paradigm to the vulnerability detection model. Figure 1 shows the overview of our approach, which includes the following three phases.

- *Phase-1 Diversified Cryptographic Knowledge Base Construction*: Cryptographic knowledge is extracted from various unstructured documents via LLMs to construct the diversified cryptographic knowledge base.
- *Phase-2 Pre-detection and Knowledge Retrieval*: CRYPTOSCOPE summarizes the input code to extract its algorithmic and mathematical structure, then conducts a preliminary security analysis by either comparing it with cryptographic algorithm specifications or using few-shot CoT prompting. Both summaries are independently used to retrieve the most relevant knowledge block.
- *Phase-3 Knowledge-Augmented Vulnerability Detection*: The LLM learns from the two retrieved knowledge blocks and conducts an in-depth analysis of code defects by integrating the pre-detection analytical process, ultimately deriving conclusions.

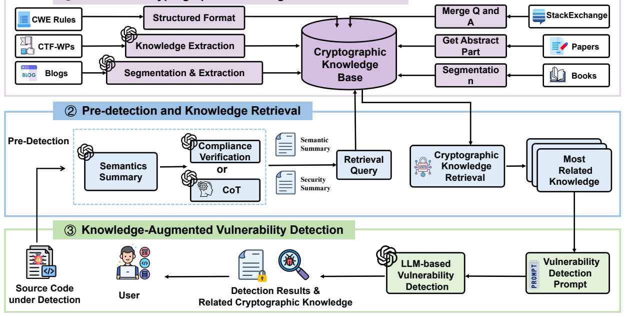
① Diversified Cryptographic Knowledge Base Construction
CWE Rules
CTF-WPs
Blogs
Structured Format
Knowledge Extraction
Segmentation & Extraction
Cryptographic Knowledge Base
Merge Q and A
Get Abstract Part
Segmentatio n
StackExchange
Papers
Books
② Pre-detection and Knowledge Retrieval
Pre-Detection
Semantics Summary
Compliance Verification
or
CoT
Semantic Summary
Security Summary
Retrieval Query
Cryptographic Knowledge Retrieval
Most Related Knowledge
③ Knowledge-Augmented Vulnerability Detection
Source Code under Detection
User
Detection Results & Related Cryptographic Knowledge
LLM-based Vulnerability Detection
PROMPT
Vulnerability Detection Prompt

Fig. 1: Overview of CRYPTOSCOPE

3.3. Diversified Cryptographic Knowledge Base Construction

We construct a high-quality knowledge base of over 12,000 cryptography-related chunks, termed the *diversified cryptographic knowledge base*, by crawling a wide range of open-source unstructured materials and applying large-model-assisted extraction and segmentation. Vectorized for efficient retrieval, this base supports downstream vulnerability detection.

Data Sources. The corpus integrates diverse cryptographic resources, summarized in Table 1.

Table 1: Sources of the Cryptographic Knowledge Base

Source Type	Description
298 CTF Writeups	CTF crypto challenge writeups from top competitions.
11 Cryptographic Blogs	Expert blogs on common cryptographic flaws.
15 CWE Rules	CWE rules related to cryptographic vulnerabilities.
3 Books [25–27]	Books on cryptographic implementation and security flaws.
738 Research Abstracts	Abstracts of cutting-edge cryptanalysis research.
3909 StackExchange [28] Posts	Practical cryptography Q&A from StackExchange.

Knowledge Extraction. For CTF writeups, we use an LLM to extract fine-grained knowledge units per challenge. Blogs in Markdown format are manually segmented by third-level headers, then parsed into structured units by the LLM. Other sources are preprocessed through heuristic or fixed-size chunking.

Embedding. Knowledge units are stored in JSON Lines format. During vector index construction, all sources are embedded using cosine similarity. StackExchange questions serve as retrieval keys; question-answer pairs are returned. For other sources, each unit is used as both the key and content.

3.4. Pre-detection

The pre-detection phase has three components: *Semantic Summary*, *Compliance Verification*, and *CoT-Based Reasoning*.

First, the LLM generates a semantic summary of the target code, emphasizing cryptographic logic, parameter sizes, and algebraic structures to aid understanding and support retrieval.

Next, compliance with standards is verified. We manually prepare reference documents for 42 common algorithms based on FIPS [29], covering logic flow, parameter limits, and security assumptions. The LLM checks conformity by analyzing parameter generation and encryption/decryption, simulating manual audits. Results form a retrieval index.

For non-standard algorithm code, few-shot CoT prompting guides the LLM to detect potential flaws by breaking down security goals—confidentiality, integrity, authentication—into concrete checks. Prompts focus on typical issues like input validation, primitive misuse, and error handling. Representative cases enable expert-level reasoning and generalization, enhancing accuracy and interpretability.

Figure 2 shows the prompt format, consisting of three parts: Instruction (principle and reasoning steps), Example (code walkthrough), and Notice (output format and key reminders).

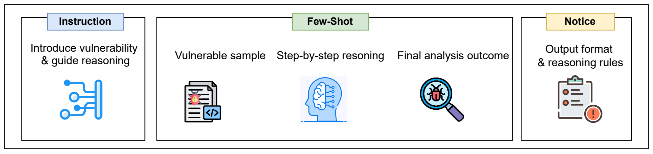

Fig. 2: CoT prompt structure.

To detect weak elliptic curves [30], we integrate a remote SageMath [31] execution environment. The LLM extracts curve parameters from the code, converts them into Sage-compatible syntax, and submits the code for remote execution. It then analyzes the returned results to assess potential vulnerabilities.

3.5. RAG-based Vulnerability Analysis

Although CoT-based reasoning helps the LLM identify potential vulnerabilities, it is still prone to false positives, false negatives, and imprecise analyses. To mitigate this, we integrate a RAG mechanism that leverages external cryptographic knowledge to enhance and support the reasoning process.

Our retrieval strategy uses two signals: the semantic summary of the code and intermediate outputs from CoT-based reasoning. The former captures core cryptographic constructs and parameter attributes for matching structurally or semantically similar code fragments, while the latter highlights vulnerability logic paths that often correspond to known flaw patterns. These signals are embedded into vectors, and the top-k relevant entries are retrieved using cosine similarity.

To avoid semantically irrelevant or misleading results, we apply a similarity threshold τ and retain only entries with $\cos_s\text{im} \geq \tau$. This process is detailed in Algorithm 1. In subsequent experiments, we set the similarity threshold $\tau = 0.75$, as empirical results indicate this value achieves the best trade-off between relevance and precision.

In the generation phase, the retrieved knowledge is combined with the original semantic and reasoning features and provided to the LLM. This design ensures that the model benefits from historical vulnerability patterns and domain-specific knowledge, while still retaining its internal reasoning capacity.

Algorithm 1 Threshold-Based Cryptographic Knowledge Retrieval

```
Input: Query $q$, number $k$, and threshold $\tau$
Output: Top-$k$ similar results $R$
 1: $docs\_with\_scores \leftarrow \texttt{similarity\_search}(q, k)$
 2: Initialize result text $R \leftarrow \emptyset$; counter $c \leftarrow 1$
 3: for each $(d, s)$ in $docs\_with\_scores$ do
 4:     $cos\_sim \leftarrow 1 - s$
 5:     if $cos\_sim \geq \tau$ then
 6:         Append content of $d$ to $R$ with index $c$
 7:         $c \leftarrow c + 1$
 8:         if $d$ contains an ID then
 9:             Extract ID and lookup full Q&A from Crypto StackExchange
10:             Append question and answer to $R$
11:         end if
12:     end if
13: end for
14: return $R$
```

4. EVALUATION SETUP

We evaluate the performance and practical value of CRYPTOSCOPE by addressing the following four research questions:

- **RQ1: Compared to the baseline:** How does CRYPTOSCOPE perform compared to the baseline?
- **RQ2: Ablation study:** To what extent does each component of our framework contribute to the overall performance of vulnerability detection?
- **RQ3: Real-world evaluation:** How well does CRYPTOSCOPE detect cryptographic vulnerabilities in real-world cryptographic libraries?

5. RESULTS AND ANALYSES

5.1. RQ1: Effectiveness Compared to Baselines

To evaluate the effectiveness of CRYPTOSCOPE, we compare it against vanilla LLMs on the LLM-CLVA benchmark across six representative LLMs using four metrics. The results in Table 2 demonstrate consistent improvements across all models.

CRYPTOSCOPE significantly enhances the performance of strong baselines such as DeepSeek-V3 and GPT-4o-mini. DeepSeek-V3 improves from 80.73 to 90.11 in Credibility Score and from 76.14% to 83.04% in Semantic Match Rate. GPT-4o-mini exhibits substantial gains in both credibility (+13.33) and coverage (+18.32%). While the gains for Qwen-Plus and Gemini 1.5 Flash are more moderate, improvements are still observed across key metrics.

These findings indicate the generalizability of CRYPTOSCOPE: it systematically boosts cryptographic reasoning capabilities across diverse model backbones.

5.2. RQ2: Ablation Study

We conduct an ablation study to evaluate the impact of two core components in CRYPTOSCOPE: the Pre-detection module (which employs Chain-of-Thought-based reasoning) and the Knowledge-Augmented Analysis module based on Retrieval-Augmented Generation (RAG). The experiments are performed on two representative large language models, DeepSeek-V3 and GLM-4-Flash, which serve as proxies for distinct categories of LLM architectures—DeepSeek-V3 representing open-source models optimized for multi-round reasoning tasks, and GLM-4-Flash exemplifying high-throughput models designed for efficient short-context inference. Table 3 reports the resulting Credibility Scores under various configurations, highlighting the contribution of each component across different model capabilities.

Table 2: Comparison of baseline LLMs and CRYPTOSCOPE on the LLM-CLVA benchmark.

Model	**Credibility**	**Cosine Sim. (%)**	**Semantic Match (%)**	**Coverage (%)**
DeepSeek-V3 (Base)	80.73	69.78	76.14	50.61
DeepSeek-V3 (CRYPTOSCOPE)	**90.11**	**72.71**	**83.04**	**56.15**
Qwen-Plus (Base)	72.39	67.18	69.57	47.12
Qwen-Plus (CRYPTOSCOPE)	**75.76**	67.84	71.41	49.79
GPT-4o-mini (Base)	65.74	61.45	61.96	36.38
GPT-4o-mini (CRYPTOSCOPE)	**79.07**	**65.05**	**68.04**	**54.70**
Gemini 1.5 Flash (Base)	64.92	62.76	62.17	53.35
Gemini 1.5 Flash (CRYPTOSCOPE)	**71.34**	**68.78**	66.88	49.04
GLM-4-Flash (Base)	53.93	60.40	48.21	28.62
GLM-4-Flash (CRYPTOSCOPE)	**69.40**	**65.19**	**60.27**	**43.20**
Claude 3 Haiku (Base)	53.34	60.71	48.37	39.46
Claude 3 Haiku (CRYPTOSCOPE)	**59.51**	**67.44**	49.24	37.48

Table 3: Ablation study results (Credibility Score).

Model	Baseline	Full	**w/o CoT**	**w/o RAG**
DeepSeek-V3	80.73	**90.11**	83.02	85.45
GLM-4-Flash	53.93	**69.40**	65.32	56.16

5.3. RQ3: Real-world Evaluation

To assess practicality, we applied CRYPTOSCOPE (with DeepSeek-V3) to 20 real-world cryptographic codebases. The tool uncovered various logic-level flaws, such as improper ECDSA signature range checks, insecure padding in RSA, ECB-mode misuse, and weak key derivation practices.

Table 4 presents representative cases. Notably, many issues had not been previously reported, confirming CRYPTOSCOPE's potential in real-world auditing.

Table 4: Vulnerabilities discovered in open-source cryptographic projects.

Project	**Commit**	**File**	**Vulnerability**
goEncrypt	be7042	rsacrypt.go	PKCS#1 v1.5 misuse
cryptography	5dc3c3	controllers-ck.js	ECB mode, weak KDF
crypto-random-string	25f893	core.js	Modulo bias
nimcrypto	4a0633	pbkdf2.nim	Weak iteration count
generate-password	d11ddd	generate.js	Modulo bias
simple-crypto	13559f	publickeysystem.py	Insecure RSA padding
ecurve	ee8a22	curve.js	Incorrect square root algorithm
fastecdsa	4617ef	_ecdsa.c	Missing r/s range check allows signature bypass
crypto	7112a2	diffiehellman.py	Weak prime generation

6. CONCLUSION

We introduced CRYPTOSCOPE, a novel framework for automated cryptographic logic vulnerability detection using LLMs. By combining CoT prompting and RAG with a curated knowledge base, CRYPTOSCOPE identifies complex flaws without code execution. On our LLM-CLVA benchmark, it consistently and significantly boosted the performance of various baseline models and discovered 9 undisclosed vulnerabilities in real-world projects. This work demonstrates that knowledge-augmented LLMs are a powerful, scalable, and language-agnostic tool for security auditing. Future work will enhance the knowledge base and reasoning capabilities.

References

[1] Ross Anderson, "Why cryptosystems fail," in *Proceedings of the 1st ACM Conference on Computer and Communications Security*, 1993, pp. 215–227.

[2] OpenAI, "GPT-3.5 Turbo," `https://platform.openai.com/docs/models/gpt-3.5-turbo`, 2023, Accessed: 2025-05-03.

[3] NIST, "Cve-2022-21449 detail," `https://nvd.nist.gov/vuln/detail/cve-2022-21449`, 2022, Accessed: 2025-05-02.

[4] Manuel Egele, David Brumley, Yanick Fratantonio, and Christopher Kruegel, "An empirical study of cryptographic misuse in android applications," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, 2013, pp. 73–84.

[5] Stefan Krüger, Johannes Späth, Karim Ali, Eric Bodden, and Mira Mezini, "Crysl: An extensible approach to validating the correct usage of cryptographic apis," *IEEE Transactions on Software Engineering*, vol. 47, no. 11, pp. 2382–2400, 2019.

[6] Sazzadur Rahaman, Ya Xiao, Sharmin Afrose, Fahad Shaon, Ke Tian, Miles Frantz, Murat Kantarcioglu, and Danfeng Yao, "Cryptoguard: High precision detection of cryptographic vulnerabilities in massive-sized java projects," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 2455–2472.

[7] Yifan Xia, Zichen Xie, Peiyu Liu, Kangjie Lu, Yan Liu, Wenhai Wang, and Shouling Ji, "Exploring automatic cryptographic api misuse detection in the era of llms," *arXiv preprint arXiv:2407.16576*, 2024.

[8] Heewon Baek, Minwook Lee, and Hyoungshick Kim, "Cryptollm: Harnessing the power of llms to detect cryptographic api misuse," in *European Symposium on Research in Computer Security*. Springer, 2024, pp. 353–373.

[9] C2SP, "Wycheproof," `https://github.com/C2SP/wycheproof`, 2022, Accessed: 2 May 2025.

[10] Jean-Philippe Aumasson and Yolan Romailler, "Automated testing of crypto software using differential fuzzing," *Black Hat USA*, vol. 7, pp. 2017, 2017.

[11] Shirin Nilizadeh, Yannic Noller, and Corina S Pasareanu, "Diffuzz: differential fuzzing for side-channel analysis," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 2019, pp. 176–187.

[12] Guido Vranken, "cryptofuzz," `https://github.com/guidovranken/cryptofuzz`, 2022, Accessed: 2 May 2025.

[13] Ben Mann, N Ryder, M Subbiah, J Kaplan, P Dhariwal, A Neelakantan, P Shyam, G Sastry, A Askell, S Agarwal, et al., "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165*, vol. 1, pp. 3, 2020.

[14] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al., "Chain-of-thought prompting elicits reasoning in large language models," *Advances in neural information processing systems*, vol. 35, pp. 24824–24837, 2022.

[15] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al., "Retrieval-augmented generation for knowledge-intensive nlp tasks," *Ad-*

vances in neural information processing systems, vol. 33, pp. 9459–9474, 2020.

[16] Lucas McDaniel, Erik Talvi, and Brian Hay, "Capture the flag as cyber security introduction," in *2016 49th hawaii international conference on system sciences (hicss)*. IEEE, 2016, pp. 5479–5486.

[17] Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba, "Large language models are human-level prompt engineers," in *The Eleventh International Conference on Learning Representations*, 2022.

[18] DeepSeek, "Deepseek-v3," `https://github.com/deepseek-ai/DeepSeek-V3`, 2025, Accessed: 2025-05-02.

[19] OpenAI, "Gpt-4o-mini," `https://platform.openai.com/docs/models/gpt-4o-mini`, 2024, Accessed: 2025-05-02.

[20] Zhipu AI, "Glm-4-flash," `https://bigmodel.cn/dev/activities/free/glm-4-flash`, 2025, Accessed: 2025-05-02.

[21] Alibaba Cloud, "Qwen-plus," `https://help.aliyun.com/zh/model-studio/what-is-qwen-llm?userCode=okjhlpr5#6c45e49509gtr`, 2025, Accessed: 2025-05-02.

[22] Google DeepMind, "Gemini 1.5 flash documentation," `https://ai.google.dev/gemini-api/docs/models#gemini-1.5-flash`, 2024, Accessed: 2025-05-02.

[23] Anthropic, "Claude 3.5 haiku," `https://www.anthropic.com/claude/haiku`, 2024, Accessed: 2025-05-02.

[24] Yuanhang Zhou, Fuchen Ma, Yuanliang Chen, Meng Ren, and Yu Jiang, "Clfuzz: Vulnerability detection of cryptographic algorithm implementation via semantic-aware fuzzing," *ACM Transactions on Software Engineering and Methodology*, vol. 33, no. 2, pp. 1–28, 2023.

[25] Mark Stamp and Richard M Low, *Applied cryptanalysis: breaking ciphers in the real world*, John Wiley & Sons, 2007.

[26] Niels Ferguson, Bruce Schneier, and Tadayoshi Kohno, *Cryptography engineering: design principles and practical applications*, John Wiley & Sons, 2011.

[27] Alfred J Menezes, Paul C Van Oorschot, and Scott A Vanstone, *Handbook of applied cryptography*, CRC press, 2018.

[28] Crypto Stack Exchange, "Crypto Stack Exchange," `https://crypto.stackexchange.com/`, 2025, Accessed: 2025-05-02.

[29] National Institute of Standards and Technology, "Federal information processing standards (fips)," `https://www.nist.gov/federal-information-processing-standards-fips`, 2022, Accessed: 2025-05-02.

[30] Peter Novotney, "Weak curves in elliptic curve cryptography," *modular. math. washington. edu/edu/2010/414/projects/novotney. pdf*, 2010.

[31] SageMath, "Sage," `https://github.com/sagemath/sage`, 2024, Accessed: 2025-05-02.