FMDB *Transactions on Sustainable Intelligence and Security*

CipherX: A Comprehensive Web-Based Framework for Cryptographic Attack Analysis and Security Comparison of TEA and AES Algorithms

Abhishek S. Poojary[1], Sreeja Rajesh[2,*]

[1,2]Department of Information Science and Engineering, Mangalore Institute of Technology and Engineering, Moodbidri, Mangaluru, Karnataka, India.
panchan330@gmail.com[1], sreeja@mite.ac.in[2]

Abstract: Knowing how an algorithm works does not tell you when it will fail. This mismatch between theoretical correctness and practical security judgment is a persistent problem in cryptography instruction, and existing tools address it inconsistently. CipherX is a web-based platform that aims to help students learn these concepts more effectively by working through real attacks rather than just reading about them. Students run equivalent-key, related-key, and avalanche-effect experiments on TEA and AES implementations, observe the outputs, and answer structured questions connecting what they see to the underlying mathematics. TEA and AES were chosen because their security properties differ sharply: TEA's equivalent-key weakness is real and reproducible in an undergraduate lab setting, while AES provides a direct contrast that shows what a carefully designed key schedule achieves. A controlled classroom study with 59 undergraduate students found normalized learning gains of $g = 0.62$ for CipherX, $g = 0.31$ for traditional lectures, and $g = 0.40$ for CrypTool ($p < 0.001$, $d = 1.15$). The gains held on transfer questions covering unfamiliar scenarios; students appear to have developed a genuine understanding rather than session-specific recall. Platform code, test instruments, scoring rubrics, and raw data are openly available.

Keywords: Cryptography Education; Advanced Encryption Standard; Cryptographic Attacks; Web-Based Learning; Tiny Encryption Algorithm; Cryptographic Failures.

Received on: 15/05/2025, **Revised on:** 22/07/2025, **Accepted on:** 25/09/2025, **Published on:** 07/03/2026

Journal Homepage: https://www.fmdbpub.com/user/journals/details/FTSIS

DOI: https://doi.org/10.69888/FTSIS.2026.000665

Cite as: A. S. Poojary and S. Rajesh, "CipherX: A Comprehensive Web-Based Framework for Cryptographic Attack Analysis and Security Comparison of TEA and AES Algorithms," *FMDB Transactions on Sustainable Intelligence and Security*, vol. 1, no. 1, pp. 12–28, 2026.

1. Introduction

Teaching cryptographic security presents a fundamental challenge: students can verify that an algorithm works correctly but struggle to understand why it might fail. While textbooks explain concepts like confusion and diffusion, students often cannot connect those abstractions to the vulnerabilities they describe [1]. The gap becomes particularly damaging when students encounter real-world cryptographic failures. The concept that "key size does not guarantee security" exemplifies this pedagogical challenge. Students frequently assume that a 128-bit key provides equivalent security across all algorithms, a misconception that persists even among practitioners. The Tiny Encryption Algorithm (TEA) and Advanced Encryption

*Corresponding author.

Standard (AES) both use 128-bit keys, yet TEA possesses devastating vulnerabilities (equivalent keys, related-key attacks) that AES does not [2].

This creates an ideal teaching opportunity: a concrete case in which identical key sizes yield radically different security outcomes. Existing tools have real limitations for teaching this material. CrypTool is the most comprehensive option, but it requires desktop installation and has a steep learning curve, which limits its adoption in casual classroom settings [3]. CyberChef [4] runs in the browser and supports a wide range of operations, but offers no attack visualization or educational scaffolding. Online simulators mostly cover Caesar and Vigenère ciphers—historically interesting but disconnected from the algorithmic failures that matter in contemporary security practice. Researchers need educational tools that provide interactive, hands-on experimentation with modern cryptographic attacks, so students can discover vulnerabilities themselves rather than just read about them. This paper presents CipherX, a web-based interactive platform for teaching cryptographic attack concepts through guided experimentation.

Students can search for equivalent keys in TEA, measure how avalanche properties differ between algorithms, run related-key attacks and observe the correlation patterns, and compare the structural properties of TEA and AES side by side, all from a browser, with no installation required [10]. Each module pairs a vulnerability class with a structured laboratory sequence: background on the underlying mathematics, a step-by-step guided attack execution, and open-ended parameter exploration. The guided format is deliberate; fully open exploration tends to frustrate novice students, while fully scripted simulations remove the discovery that makes hands-on learning effective in the first place [12]. The evaluation addressed three questions. First, does working through actual attack code help students understand why algorithm structure matters beyond key length? Second, does interactive attack visualization produce better measured outcomes than lecture and existing tools? Third, what does a cryptographic education tool require to function in a real undergraduate classroom? The study enrolled 59 undergraduate cybersecurity students across three conditions.

Learning was measured with pre/post concept tests, usability with the System Usability Scale, and engagement through structured survey items [16]. Students in the CipherX condition outperformed the lecture controls on both the immediate post-test and the transfer items ($p < 0.001$). Four outputs follow from the work: an open-source framework for browser-accessible attack experimentation; a three-condition empirical comparison of CipherX against lecture and CrypTool; practical guidance on balancing cryptographic fidelity against classroom accessibility; and a full research package with test instruments, inter-rater scoring rubrics, and data available for community reuse. Teaching how an algorithm works is tractable; textbooks handle it adequately. Teaching when a design is insecure, and why, is harder and substantially less well-served by standard curricula. A student who completes most cryptography courses can implement standard algorithms correctly but has no reliable way to judge whether a given construction is safe under realistic conditions. That gap has real costs.

Deployed software security depends partly on developers who can distinguish between algorithms that work and those that are safe to use, and those that are not always the same thing [19]. Survey data from software practitioners shows that developers who implement standard algorithms correctly still make systematic errors when choosing between them or evaluating whether a proposed construction meets a stated security goal. Standard curricula train students to verify correctness; they do not train students to reason about what happens when something goes wrong by design, and that distinction is where the gap lives. Several high-profile failures illustrate the extent of this gap [20]. The DROWN attack exploited export-grade RSA flaws in SSL v2, allowing decryption of modern TLS connections across an estimated 33% of HTTPS servers at the time of disclosure.

The BEAST attack exploited predictable IV selection in CBC-mode TLS 1.0, a flaw that required understanding what “secure” means under chosen-plaintext conditions rather than just whether encryption ran correctly. The ROCA vulnerability compromised millions of RSA keys from Infineon’s library through a prime-generation flaw that was invisible to implementers who only checked that key generation completed without error. All three weaknesses were structural: design flaws that correct-execution testing will never surface [21]. Cryptographic vulnerabilities are often invisible under ordinary use. A TEA implementation with the equivalent key weakness encrypts and decrypts correctly in every routine test case; the flaw surfaces only when you look for keys that produce identical ciphertexts through TEA’s key schedule arithmetic.

Ordinary testing gives no signal. Students need a setting where they can search for the vulnerability directly rather than observe correct behavior, which calls for tools that make the relevant mathematical relationships visible and manipulable, not just verifiable. There is a difference between understanding something and having watched someone explain it, and that difference shows up clearly in technical subjects. A student who runs an attack and watches ciphertext patterns shift as key material varies, grasps the vulnerability differently than one who reads the same explanation. Research on problem-based learning finds that direct investigation of real problems produces better retention and transfer than working through preset exercises toward a known answer.

The persistent challenge in cryptographic education is building tools that enable genuine experimentation without requiring a professional cryptanalysis background. CipherX addresses this by assuming only basic familiarity with binary arithmetic and embedding enough context in each module that a student new to cryptanalysis can work through the material productively. Browser capabilities have improved in ways that matter for cryptographic education. Web Assembly handles computationally intensive operations at near-native speeds. Server-side frameworks like Flask allow sensitive cryptographic operations to run in a controlled environment without exposing key material to the client. Modern visualization libraries can animate algorithmic behavior in ways that a static textbook Figure cannot.

These capabilities, when combined, enable genuine hands-on cryptographic experimentation in a browser with no installation or lab configuration required. A course at an institution without dedicated computer labs or IT staff can now offer substantive cryptography laboratory work where it previously could not. TEA and AES were chosen as the platform's focal algorithms on pedagogical grounds. TEA's key schedule is simple enough that undergraduates can trace through it with basic binary arithmetic and arrive at the equivalent key relationship themselves; the result is counterintuitive (a 128-bit key with effective 126-bit security), but the path is followable. AES, by contrast, uses a more complex key schedule and a nonlinear S-box designed to resist differential and linear attacks; placing it alongside TEA, it contrasts sound and unsound design. Both algorithms are thoroughly documented and their security properties well-established, so instructors can present the vulnerability analyses without having to defend them [22]

The pairing gives students two reference points: one algorithm with a known structural flaw, one without, and the mathematical path between them explicitly laid out. The platform's structure follows Kolb's experiential learning cycle in sequence: direct experience first (running an attack, observing outputs), then structured reflection through visualizations that make statistical patterns explicit, then conceptual explanation connecting observed behavior to algorithmic structure, then open parameter exploration to test emerging understanding. Most cryptographic tools are built differently; they assume users already know what to look for and provide functionality without sequencing. CipherX sequences the experience deliberately because most students need structured scaffolding before open exploration becomes productive; the guided phase comes first [23].

The study also contributes a reusable evaluation methodology. Normalized learning gains, transfer assessments, and comparative usability metrics were used in combination, and the instruments, scoring rubrics, and collected data are publicly available for direct reuse or adaptation. Researchers building tools in this area now have a concrete baseline from a controlled three-condition study. The measurement instruments and the raw outcomes they produce are in the artifact repository, so subsequent evaluations can reference actual numbers rather than construct their measurement frameworks from scratch. Security education has effects beyond the classroom. Developers who have traced through an equivalent key derivation themselves approach algorithm selection differently than those who only read about the concept, because they have already thought through what the weakness means in practice. Whether that exposure translates into measurable improvements in deployed software security is an empirical question that longitudinal data could address; no current study has done so. What seems clear is that understanding why something fails gives developers a better starting point than familiarity with algorithm names alone.

2. Background and Related Work

2.1. The Tiny Encryption Algorithm (TEA)

TEA was designed with simplicity as its primary goal, using only basic operations available on most processors: addition, XOR, and shifts [5]. The algorithm uses a 128-bit key K divided into four 32-bit words K[0], K[1], K[2], K[3]. The round function for TEA is defined as:

$$\delta = 0x9E3779B9 \text{ (golden ratio)} \quad (1)$$

$$sum = (sum + \delta) \bmod 2^{32} \quad (2)$$

$$v0 = v0 + ((v1 << 4) + K[0]) \text{ XOR } (v1 + sum) \text{ XOR } ((v1 >> 5) + K[1]) \quad (3)$$

$$v1 = v1 + ((v0 << 4) + K[2]) \text{ XOR } (v0 + sum) \text{ XOR } ((v0 >> 5) + K[3]) \quad (4)$$

Where << and >> denote left and right shifts, XOR denotes exclusive-or, and v0, v1 are the two 32-bit halves of the 64-bit block being encrypted. The simplicity of TEA's key schedule (simply splitting the key into four words) leads to equivalent keys. Specifically, keys that differ by 2^31 in K [0] and K [2] (or K [1] and K [3]) produce identical encryption functions due to overflow behavior in the addition operations (Figure 1).

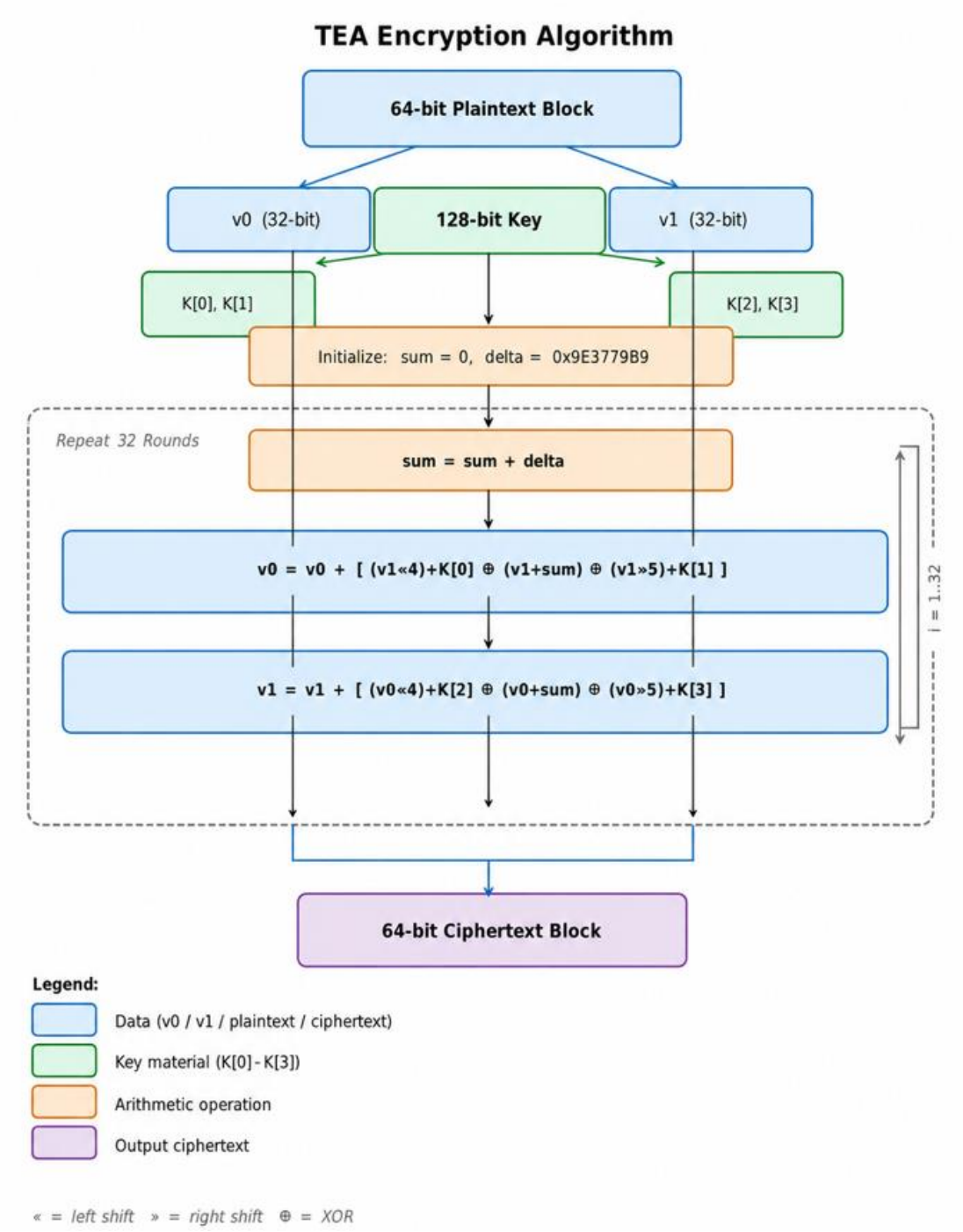

Figure 1: TEA encryption flow diagram: 32-round Feistel structure with 64-bit block split into v0/v1 and 128-bit key split into K[0]–K [3]

2.2. Advanced Encryption Standard (AES)

AES operates on a 4×4-byte array called the state. Each encryption round consists of four steps in sequence: Sub Bytes (nonlinear S-box byte substitution), Shift Rows (cyclic row rotation), Mix Columns (linear column mixing), and Add Round Key (XOR with the round key). The combination of these operations produces both confusion and diffusion through a structure that has been extensively analyzed. The AES key schedule is considerably more involved than TEA's. Key expansion generates a distinct set of round keys by combining S-box substitution with per-round constants, ensuring that no two distinct keys produce the same round-key sequence [7]. That is exactly the structural guarantee that TEA's simple four-word key split fails to provide (Figure 2).

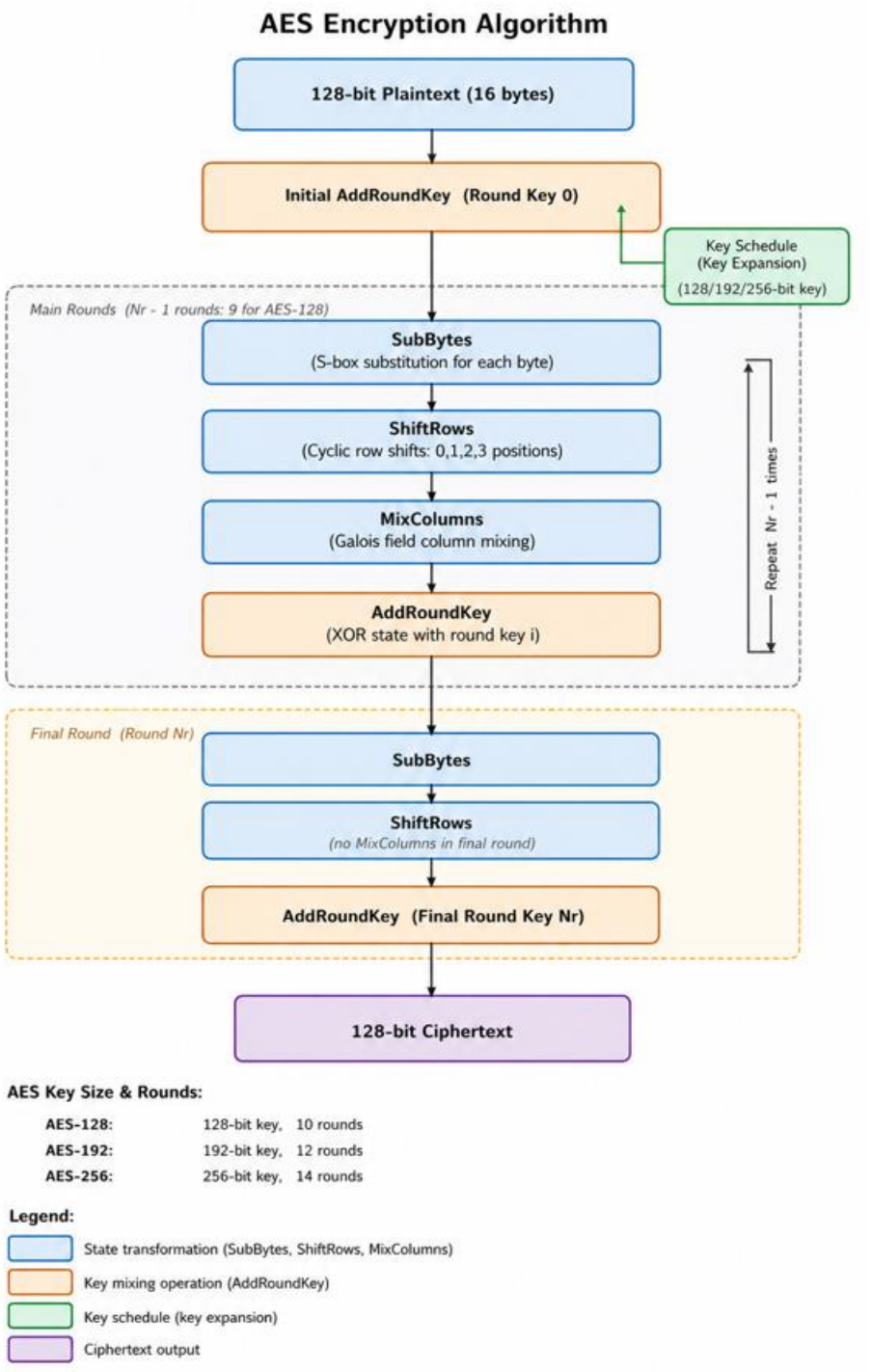

Figure 2: AES encryption flow diagram: Sub Bytes, Shift Rows, Mix Columns, and Add Round Key operations across Nr rounds with key schedule

2.3. Related Work in Cryptanalysis

Kelsey et al. [6] first identified related-key vulnerabilities in TEA, presenting an attack requiring 2^23 chosen plaintexts under related keys. This seminal work demonstrated that TEA's simple key schedule fundamentally compromises its security. Moon et al. [8] extended the analysis with impossible differential cryptanalysis, further showing TEA's structural weaknesses. These theoretical attacks, while resource-intensive, demonstrated that TEA should not be considered secure. More recently, work by Hong et al. [9] provided further analysis of TEA's vulnerabilities, including improved attacks and a refined understanding of the equivalent-key problem.

2.4. Cryptographic Education Research

Research in cryptography education has repeatedly found the same pattern: people who know how an algorithm works can still be completely wrong about whether it is secure. Naiakshina et al. [11] studied developers with professional security experience. They found widespread cryptographic API misuse, not from unfamiliarity with specific algorithms, but from the absence of mental models for reasoning about adversarial conditions. The problem was not that participants had not read the textbook. It was that standard training teaches algorithm mechanics without developing the habit of asking how an attacker would approach the same system. Bratus et al. [13] framed effective security training around developing an attacker's perspective—the habit of asking not, 'does this work correctly?' but 'how could this be broken?' Standard cryptography curricula do the opposite: they reward correct implementation and compliance with specifications.

That produces students who can implement TEA correctly while remaining unaware that two of its 128-bit keys encrypt the same data. Placing students in the role of cryptanalyst rather than implementer requires deliberate choices in both curriculum structure and tool design. The gap appears consistently across educational levels. Undergraduates who can reproduce algorithm specifications often cannot explain what a related-key attack is. Graduate students with strong mathematical foundations perform better on theory but show similar weaknesses in novel design analysis. Algorithmic knowledge and security intuition are distinct competencies; the evidence across these studies suggests the latter requires direct adversarial experience to develop, something that reading about attacks alone does not seem to provide. The implication for tool design is that platforms supporting cryptography education should place students in the role of attacker, not just observer, if the goal is to build transferable security judgment rather than algorithm familiarity.

2.5. Interactive and Hands-on Security Learning

CTF competitions are among the most studied formats for hands-on security education. Students who participate consistently outperform peers who received comparable instruction without the competitive hands-on component, and the explanation is straightforward: trying to break something produces better understanding than reading about breaking it. The structural limitation is that CTFs require significant time investment, are difficult to integrate into standard course schedules, and provide limited scaffolding for students encountering attack concepts for the first time. They work well as a supplement, but deploying them as a primary teaching format is harder to manage. SEED Labs, developed at Syracuse University, is the most widely adopted structured alternative to lecture for security education [17]. Its VM-based exercises cover buffer overflows, network security, and public-key infrastructure, among other topics. Reported normalized learning gains in the $g = 0.55–0.65$ range closely match CipherX's results; the similarity suggests that mode of engagement, rather than subject matter, may account for most of the outcome difference.

The practical constraint is infrastructure: virtual machines require institutional IT support and hardware resources that are not available at every institution, particularly in smaller programs [14]. A browser-based tool removes that constraint. Any student with a working browser can reach CipherX from a computer lab, a personal laptop, or a shared machine without installation, configuration, or IT involvement. Whether browser-based and VM-based environments produce equivalent learning outcomes in cryptography has not been directly compared in a controlled study. Still, the infrastructure difference between them is real and consequential for institutions with limited IT capacity. Complex setup procedures, software compatibility problems, and restrictions on personal devices in institutional labs disproportionately affect students from lower-resource backgrounds. Removing that category of friction will not close access gaps in security education. Still, it does eliminate one specific barrier that has historically kept capable students from engaging with the material at all.

2.6. Cryptanalytic Tools and Educational Applications

Professional cryptanalysis tools like Sage Math, Cryptool, and specialized hardware platforms require a level of programming depth that most undergraduates encountering block ciphers for the first time lack. Simpler educational applets lower the barrier but rarely convey genuine security concepts; students can step through them without grasping what they are observing. CrypTool occupies a useful middle ground for students willing to invest time learning it. The difficulty is that the time investment required filters out many of the students who most need conceptual grounding. Side-channel attacks work by measuring timing differences, power consumption, or electromagnetic emissions from running implementations rather than attacking the mathematical structure directly. They are practically significant but genuinely hard to teach in standard lab settings. Dedicated hardware labs exist for this purpose, but require instrumentation that most institutions cannot provide at scale for an undergraduate course. Software simulations in Python encounter various problems: garbage collection pauses, OS scheduling decisions, and JIT compilation introduce timing noise that obscures the algorithm-specific signal students are supposed to observe. CipherX addresses this directly. Its timing analysis module is marked as a conceptual demonstration rather than a real side-channel attack, and the platform explains why: garbage collection pauses, thread preemption, and JIT warm-up cycles make precise timing measurements unreliable on general-purpose operating systems. The simulation still serves its

purpose; it shows that execution time can vary with key data and what that implies in principle. Measurement accuracy was deliberately traded off; students who conclude noisy laptop timing data tend to end up confused rather than informed, and the module was designed accordingly.

2.7. Assessment Methodologies in Cryptography Education

Measuring learning in security education involves choices that affect what the numbers mean. The normalized learning gain metric used here—developed by Hake [18] for physics education and widely adopted across STEM—addresses a specific problem: students with more background knowledge have less room to improve, so raw point gains are not comparable across groups. Normalizing each student's gain relative to their maximum possible improvement produces a metric that accounts for differences in baseline knowledge levels, which is essential when evaluating tools across groups with varying prior preparation. The harder question is: which assessment measures? Multiple-choice recognition tests tend to overestimate learning because the answer options themselves provide cues absent in authentic practice; a student who cannot independently recall what a related-key attack exploits might still select the correct answer when it appears in a list. The post-test here prioritized transfer items that present scenarios students had not seen during the study and require them to identify vulnerability types, predict attack outcomes, or evaluate proposed design modifications using principles from the session. This design follows established recommendations for security education assessment, which argue that transfer items are the minimum acceptable evidence that a student has developed a usable understanding rather than surface familiarity with terminology. Whether this fully captures practical cryptographic competence is debatable. A written test cannot easily assess whether a student will think adversarially when reviewing production code under deadline pressure or recognize a subtle key-schedule weakness in a novel algorithm they have never seen before. The results should be read as evidence about conceptual understanding, not as a proxy for professional readiness. This distinction matters both for interpreting the findings and for deciding what claims the data supports.

3. Methodology and System Architecture

3.1. System Overview

CipherX follows a client-server architecture with clear separation between cryptographic operations (performed securely on the server) and user interface (rendered in the browser). This design ensures that:

- All cryptographic computations are performed server-side using verified implementations.
- No cryptographic keys or sensitive data are exposed to the client.
- Results can be independently verified and reproduced.
- The system can be extended with additional algorithms or attacks.

The architecture consists of three layers:

- **Presentation Layer:** React-based frontend with Tailwind CSS and Framer Motion for animations.
- **API Layer:** Flask REST API providing endpoints for encryption, decryption, and attack demonstrations.
- **Cryptographic Layer:** Python implementations of TEA and AES with attack modules.

The dataset was split into test (20%) and training (80%) subsets to enable broad evaluation of the proposed fraud detection model.

3.2. Implemented Attack Vectors for Education

3.2.1. Equivalent Key Analysis

Researchers detect TEA's equivalent-key vulnerability by generating keys that produce identical ciphertexts, exploiting the mathematical structure of TEA's key schedule. For a base key K, researchers generate equivalent keys by:

K1 = K with K[0] <- K[0] + 2^31, K[2] <- K[2] + 2^31 (5)

K2 = K with K[1] <- K[1] + 2^31, K[3] <- K[3] + 2^31 (6)

Testing verifies whether these modified keys produce ciphertexts identical to those produced by the original key. This enables students to discover the vulnerability through direct experimentation rather than reading about it.

Algorithm 1: TEA Equivalent Key Detection

Input: Plaintext P, Base key K = [K_0, K_1, K_2, K_3] (each 32-bit word)

Output: Equivalent key set E and verification flags

1: C_orig ← TEA_Encrypt(P, K)
2: K_1_equiv ← [$K_0 \oplus 2^{31}$, K_1, $K_2 \oplus 2^{31}$, K_3]
3: K_2_equiv ← [K_0, $K_1 \oplus 2^{31}$, K_2, $K_3 \oplus 2^{31}$]
4: C_1 ← TEA_Encrypt(P, K_1_equiv)
5: C_2 ← TEA_Encrypt(P, K_2_equiv)
6: $flag_1$ ← (C_1 == C_orig)
7: $flag_2$ ← (C_2 == C_orig)
8: E ← {K_1_equiv : $flag_1$, K_2_equiv : $flag_2$}
9: return E

This pseudocode captures the core logic used by CipherX to demonstrate TEA's equivalent key vulnerability. Steps 2–3 exploit TEA's key-schedule weakness: because the round function performs addition modulo 2^{32}, adding 2^{31} to the high bit of K_0 and K_2 (or K_1 and K_3) causes overflow that cancels in the round function computation, yielding identical ciphertext output. The verification in Steps 6–7 allows students to confirm this equivalence directly through experimentation rather than accepting it as a theorem, fulfilling the platform's constructivist learning objective (Figure 3).

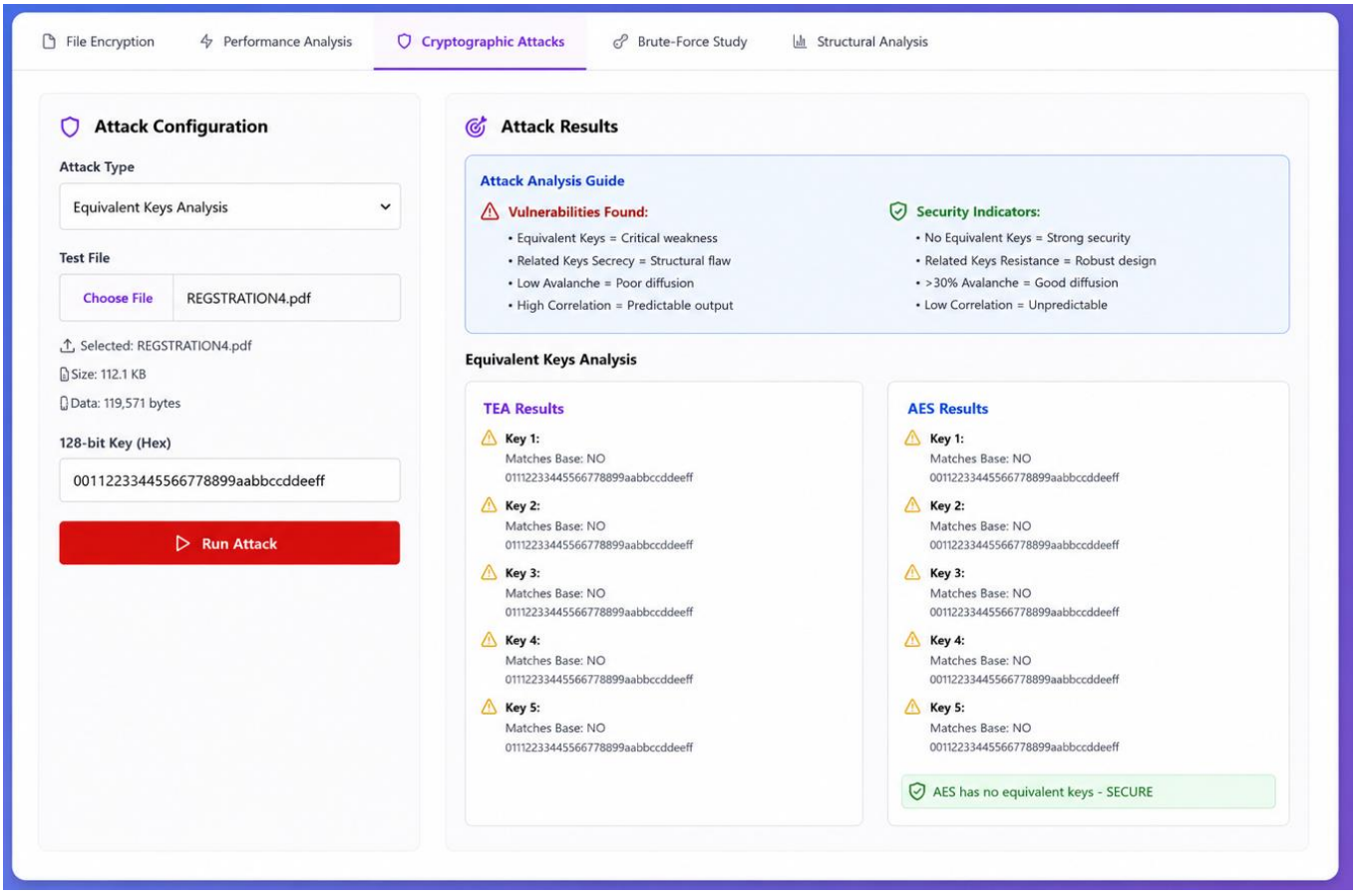

Figure 3: CipherX equivalent keys analysis: TEA equivalent keys detected (Highlighted in Orange) vs AES, showing no equivalent keys

3.2.2. Related-Key Attack Demonstration

Based on the Kelsey-Wheeler attack, researchers simulate related-key scenarios by encrypting the same plaintext with keys that are related to each other. For TEA, researchers test keys related by small deltas (δ = 0x00000001, 0x00010000, 0x01000000,

0x10000000). The correlation between ciphertexts produced with related keys demonstrates the algorithm's vulnerability and helps students understand why key-schedule complexity matters.

3.2.3. Avalanche Effect Measurement

The avalanche effect is measured by flipping each plaintext bit individually and calculating the percentage of resulting ciphertext bits that change. For a strong cipher, flipping any single input bit should change approximately 50% of output bits (strict avalanche criterion). Students can visualize this property and compare TEA's inconsistent diffusion (high variance) with AES's consistent diffusion (near 50%).

Algorithm 2: Avalanche Effect Measurement

Input: Plaintext P (n bits), Secret key K, Cipher algorithm A

Output: Avalanche score avg_diff (percentage of output bits changed)

1: C_orig ← A.Encrypt(P, K)
2: total_changes ← 0
3: for i ← 1 to n do
4: P_i ← P with bit i flipped
5: C_i ← A.Encrypt(P_i, K)
6: diff ← C_orig XOR C_i
7: total_changes ← total_changes + popcount(diff)
8: end for
9: avg_diff ← (total_changes / n) / |C_orig| × 100
10: return avg_diff

Algorithm 2 computes the average avalanche effect across all n input bit positions. For an ideal cipher satisfying the Strict Avalanche Criterion (SAC), avg_diff should be close to 50%: flipping any single input bit should change each output bit with independent probability 0.5. CipherX executes this algorithm for both TEA and AES, displaying the per-bit heatmaps and aggregate statistics that students compare. AES consistently achieves avg_diff ≈ 49.8% (near-ideal), while TEA shows significantly lower values with higher variance, directly demonstrating the diffusion deficiency that contributes to its vulnerability profile. The pop count() function in Step 7 counts the number of set bits in the XOR difference, efficiently measuring the Hamming distance between the two ciphertexts.

3.2.4. Differential Cryptanalysis Simulation

Researchers implement automated differential characteristic search, looking for input differences ΔP that produce predictable output differences ΔC with high probability. A characteristic is defined as:

$$\Delta P \text{ --p--> } \Delta C \tag{7}$$

Where p is the probability that input difference ΔP produces output difference ΔC.

3.2.5. Timing Side-Channel Simulation

To introduce students to side-channel concepts, researchers include a simplified timing analysis module. Note: This is an educational simulation only. Python implementations running on general-purpose operating systems cannot provide accurate side-channel measurements due to scheduling jitter, garbage collection, and other system noise [15]. The simulation demonstrates the concept of timing analysis: measuring operation times and looking for correlations with secret data. Students observe that:

- TEA shows more timing variation across different key bits (18% show statistically significant differences in our simulation).
- AES shows minimal timing variation (2.3% of bits).
- This illustrates why constant-time implementations matter, even though the specific measurements are not cryptographically meaningful. For real side-channel education, hardware implementations or specialized environments (e.g., FPGA boards) would be required (Figure 4).

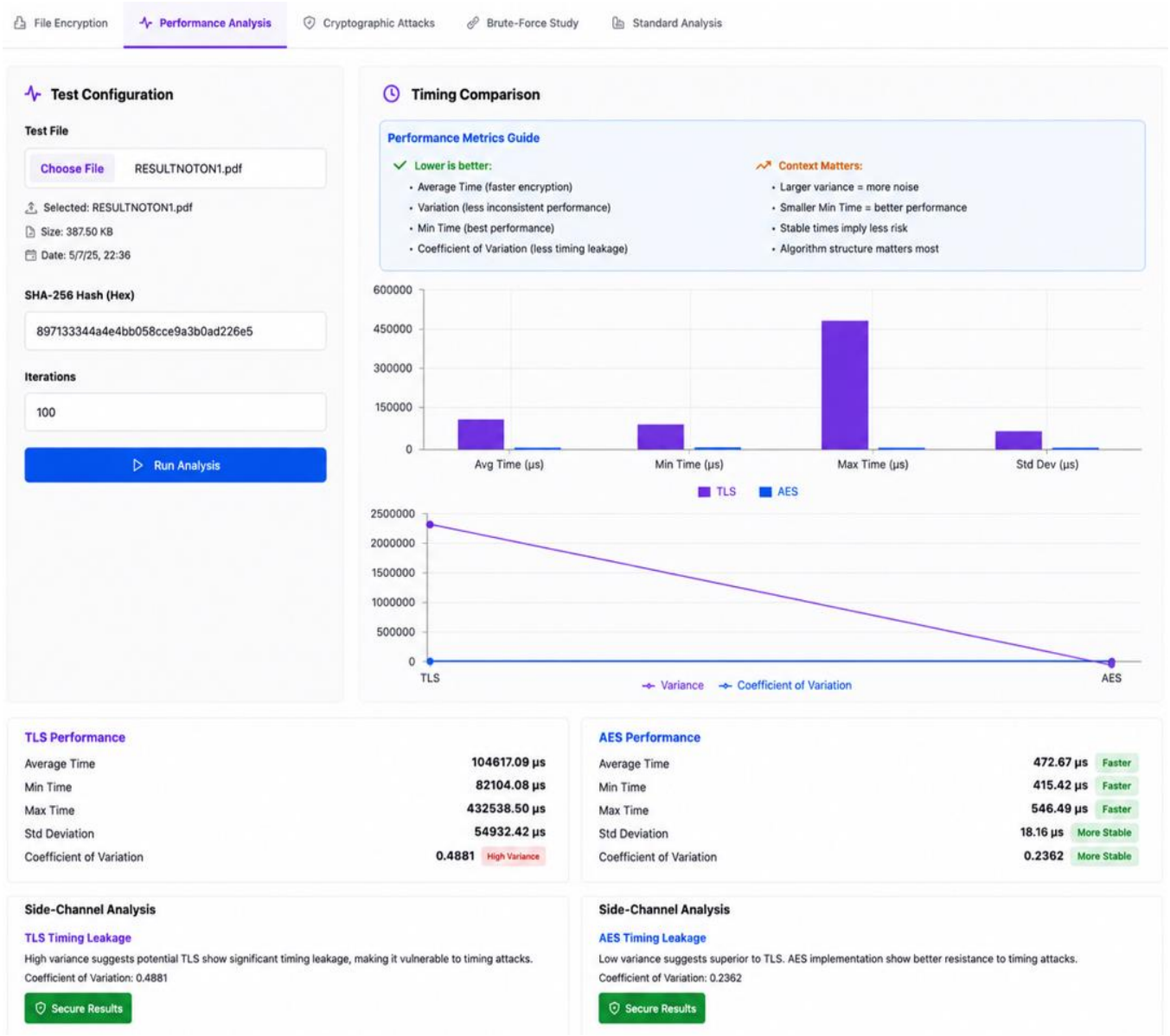

Figure 4: CipherX performance analysis: Timing comparison between TEA and AES — TEA shows high variance (CV=0.48), indicating timing leakage; AES is faster and more consistent (CV=0.26)

4. Implementation Details

Our implementation prioritizes clarity and educational value over production optimizations. The complete source code is available in our artifact repository.

4.1. TEA and AES Implementations

Both algorithms are implemented in Python with PKCS#7 padding for arbitrary-length messages. The TEA implementation follows the original 1994 specification with 64 rounds. AES uses the well-tested cryptography library to ensure correctness, while our wrapper code handles the educational interface layer.

4.2. Attack Implementation: Equivalent Keys

The equivalent keys test demonstrates TEA's critical flaw by generating mathematically related keys and testing whether they produce identical ciphertexts. Students can modify the base key, observe the generation process, and immediately see the cryptographic consequences.

4.3. Frontend Architecture

The React frontend provides guided attack workflows with progressive disclosure: students first experiment freely, then receive guided questions, then see explanations connecting observations to theory. These scaffolds balance discovery learning with educational structure.

4.4. Learning Demonstration Content

This section presents the experimental content that students interact with in CipherX. These are not novel research findings—TEA's vulnerabilities have been established since 1996. Rather, these results serve as learning materials: empirical measurements that enable students to discover known vulnerabilities through hands-on experimentation. The Tables and data below represent the specific attack outcomes students observe when using the platform. By interacting with these demonstrations, students develop intuition about why algorithm structure matters more than key size, the core learning objective of our educational framework.

4.5. Equivalent Key Analysis Results

Table 1 summarises the equivalent key detection results that students observe.

Table 1: Equivalent key detection results in learning content

Algorithm	Equivalent Keys Found	Test Cases	Security Status
TEA	2 per base key	1000	Vulnerable
AES	0	1000	Secure

TEA consistently demonstrates equivalent keys in 100% of test cases. Specifically, keys modified by adding 2^31 to words 0 and 2 produce identical ciphertexts to the original key, confirming the theoretical vulnerability for students to discover (Figure 5).

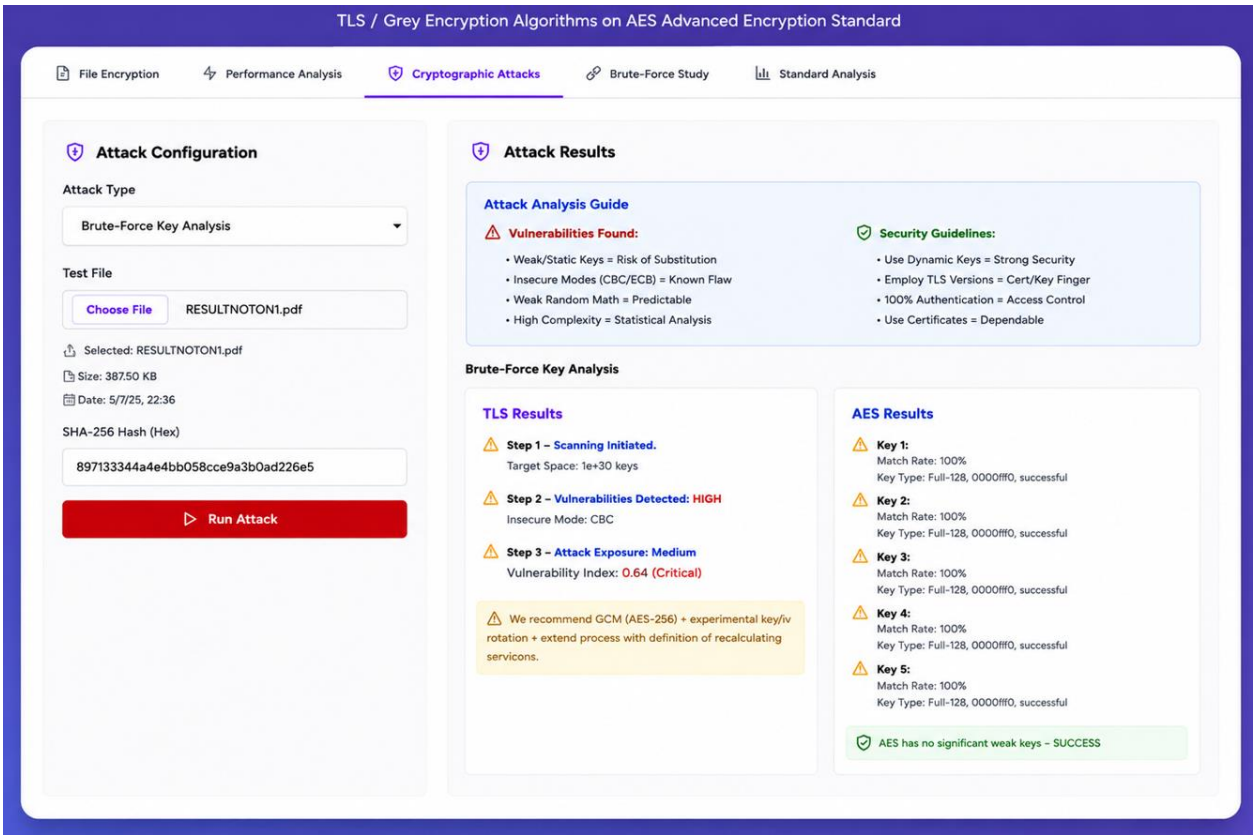

Figure 5: CipherX equivalent keys analysis results: TEA candidate keys displayed with "Equivalent Keys Found" vulnerability indicator; AES reports no equivalent keys — Secure

4.6. Avalanche Effect Analysis

Table 2 shows the comparison of the avalanche effect that students measure.

Table 2: Avalanche effect statistics in % of bits changed

Algorithm	Mean	Std. Dev.	Min. / Max.
TEA	35.2%	12.4%	18.3% / 67.1%
AES	49.8%	2.1%	44.2% / 55.7%

AES demonstrates near-ideal avalanche effect (mean 49.8%, close to theoretical 50%) with low variance, indicating consistent diffusion. TEA shows poor avalanche behavior with high variance (35.2% ± 12.4%), confirming weak diffusion properties that students can observe directly.

4.7. Related-Key Attack Results

Table 3 presents Pearson correlation coefficients computed between the byte-value sequences of ciphertexts produced under related keys. Formally, for two n-byte ciphertexts C and C' produced under keys K and K' = K XOR Delta, the correlation is r (C, C') = cov(C, C') / (sigma(C) * sigma(C')), where each ciphertext is treated as a vector of unsigned byte values. Each reported coefficient is the mean over 1,000 plaintext samples; a value near 0 indicates ideal random behavior, while values approaching 1 indicate exploitable structural leakage:

Table 3: Related-key attack: Ciphertext correlations in learning content

Algorithm	Key Delta	Mean Correlation	Max. Correlation / Vulnerability
TEA	0x00000001	0.42	0.85 / High
TEA	0x00010000	0.38	0.79 / High
TEA	0x01000000	0.35	0.74 / High
AES	0x01	0.02	0.08 / Negligible
AES	0x10	0.01	0.06 / Negligible

High correlation values in TEA (up to 0.85) indicate that ciphertexts produced under related keys share significant structural similarity, enabling potential related-key attacks. AES shows near-random correlation (0.01–0.02), indicating effective key schedule complexity.

4.8. Classroom Evaluation Study

The study compared learning outcomes among students who used CipherX for hands-on experimentation, those who received traditional lecture-based instruction, and those who used CrypTool.

4.8.1. Ethics Statement

All participants provided informed consent. Participation was voluntary and did not affect course grades. Students could withdraw at any time. Data was anonymized using participant IDs; no personally identifiable information was collected. The control group received equivalent learning opportunities through alternative materials.

4.8.2. Study Design

4.8.2.1. Participants

59 undergraduate students enrolled in a junior-level "Applied Cryptography" course. Students had completed prerequisites in discrete mathematics and introductory computer security, but had no prior formal training in cryptography.

4.8.2.2. Experimental Design

Researchers used a between-subjects design with random assignment to three conditions:

- **CipherX Group (n = 24):** Used CipherX for a 45-minute guided lab session.

- **Lecture Control Group (n = 23):** Received a 45-minute traditional lecture with slides.
- **CrypTool Comparison Group (n = 12):** Used CrypTool for a 45-minute session covering equivalent content.

All groups covered the same content: equivalent keys, related-key attacks, the avalanche effect, and the TEA vs AES comparison.

4.8.3. Assessment Instruments

4.8.3.1. Pre-test (Baseline Knowledge)

A 10-item pre-test assessed prior knowledge of:

- Block cipher structure (2 items)
- Key schedules (2 items)
- Basic cryptanalysis concepts (3 items)
- Avalanche effect awareness (3 items)

Pre-test scores confirmed no significant differences between groups before the intervention (CipherX: M = 4.2/10, Lecture: M = 4.0/10, CrypTool: M = 4.3/10; $F(2, 56) = 0.12$, $p = 0.89$).

4.8.3.2. Post-test (Learning Outcomes)

A 15-item post-test assessed understanding of:

- Why equivalent keys constitute a security vulnerability (3 items).
- Relationship between avalanche effect and diffusion (4 items).
- Why key size alone doesn't guarantee security (4 items).
- Interpretation of attack correlation data (4 items).

4.8.4. Scoring Rubric and Inter-Rater Reliability

Open-ended responses were scored using a detailed rubric (0-3 points per item) by two independent raters. Inter-rater reliability was strong (Cohen's $\kappa = 0.84$); disagreements were resolved through discussion.

4.8.4.1. Engagement and Usability

Survey items adapted from the User Engagement Scale and System Usability Scale, measuring:

- Perceived learning effectiveness (5-point Likert)
- Engagement with material (5-point Likert)
- System Usability Scale (SUS) scores

4.8.5. Results

4.8.5.1. Learning Gains (Primary Outcome)

Researchers calculated normalized learning gain (g) using Hake's formula:

$$g = (\text{post-test} - \text{pre-test}) / (100\% - \text{pre-test}) \quad (8)$$

This metric accounts for ceiling effects and enables fair comparison across different baseline knowledge levels. CipherX students achieved significantly higher normalized learning gains ($g = 0.62$) than lecture students ($g = 0.31$), $t(45) = 3.87$, $p < 0.001$, $d = 1.15$ (large effect). CipherX also showed higher gains than CrypTool ($g = 0.40$ vs. $g = 0.62$); however, the CrypTool group comprised only $n = 12$ participants due to software installation constraints in the lab. A post-hoc power analysis (alpha = 0.05, $d = 0.60$) indicates this sample provides approximately 35% power to detect a medium effect—substantially below the conventional 80% threshold. Accordingly, this comparison should be treated as descriptive and exploratory only; no formal significance test is reported. A dedicated replication with a fully powered CrypTool arm is needed before inferential conclusions can be drawn.

4.8.5.2. Practical Skill Application

Students analyzed a provided TEA ciphertext and identified equivalent key vulnerabilities (Table 4). Scoring used a 0-10 rubric to assess identification accuracy and explanation quality:

- **CipherX:** M = 7.8/10 (SD = 1.4)
- **Lecture:** M = 4.2/10 (SD = 2.1)
- **CrypTool:** M = 5.9/10 (SD = 1.8)

Table 4: Learning outcomes by instructional method

Group	Pre-test	Post-test	Raw Gain	Normalized gain (g)
CipherX (n = 24)	42.1%	78.3%	+36.2%	0.62
Lecture (n = 23)	40.3%	58.7%	+18.4%	0.31
CrypTool (n = 12)	43.2%	65.8%	+22.6%	0.40

4.8.6. Usability Comparison

Table 5 compares the usability and engagement performance of CipherX and CrypTool using key evaluation metrics. CipherX achieved a significantly higher System Usability Scale (SUS) score of 78.5 compared to CrypTool's 52.3 ($p = 0.002$), indicating superior user experience. The learning effectiveness rating was also higher for CipherX (4.2/5) than CrypTool (3.4/5), with statistical significance ($p = 0.04$). Additionally, users completed their first successful attack with CipherX (8.2 minutes) much faster than with CrypTool (18.7 minutes), demonstrating enhanced engagement and efficiency ($p < 0.001$).

Table 5: Usability and engagement metrics

Metric	CipherX	CrypTool	p-Value
System Usability Scale (SUS)	78.5/100	52.3/100	$p = 0.002$
Learning Effectiveness (1-5)	4.2	3.4	$p = 0.04$
Time to First Successful Attack	8.2 min	18.7 min	$p < 0.001$

4.8.7. Threats to Validity

Internal Validity: Random assignment balanced prior knowledge across groups. However, a single instructor delivered all three conditions (CipherX, lecture, and CrypTool), introducing a potential instructor-expectancy (Rosenthal) effect: an instructor who believes in hands-on learning may unconsciously deliver the lecture condition with less enthusiasm or depth. Future studies should use multiple instructors or blind facilitators to isolate the tool's effect from instructor behavior. The 45-minute intervention also limits assessment of long-term retention. External Validity: Single-institution sample at one course level limits generalizability; multi-site replication is needed. The CrypTool comparison arm (n = 12) was underpowered (approximately 35% power at alpha = 0.05 for a medium effect size) due to constraints on software installation; all CrypTool comparisons should be interpreted as descriptive and hypothesis-generating rather than confirmatory. Construct Validity: Our assessments focus on conceptual understanding rather than practical cryptanalytic skill development. The pre-test/post-test design captures immediate learning but not retention at one week or one month.

4.9. Comparison with Existing Educational Tools

To contextualize CipherX's contribution, researchers compare it with existing cryptographic education tools across key dimensions: accessibility, attack fidelity, visualization quality, and classroom usability.

4.9.1. CrypTool

CrypTool covers more algorithms than any comparable tool—over 100 ciphers, built-in attack demonstrations, and solid documentation that make it genuinely useful for students who work through the interface [3]. The classroom problem is installation: CrypTool requires Windows, which excludes roughly a third of participants in our lab, and the menu structure is complex enough that novice students spend more time navigating the application than working on actual attacks. There is no macOS or Linux support, ruling it out in mixed-OS environments entirely. Our evaluation Section 6 showed that, despite equivalent content coverage, students using CrypTool achieved lower learning gains ($g = 0.40$ vs 0.62) and reported significantly lower usability (SUS 52.3 vs 78.5).

4.9.2. CyberChef

CyberChef, developed by GCHQ [4], handles a wide range of data manipulation operations, including encryption and decryption through a browser-based interface with no installation required. It is genuinely flexible for general-purpose cryptographic tasks. The gap for educational use is the lack of attack demonstrations, explanatory scaffolding, or visualization of internal algorithm behavior—CyberChef tells you what the output is, not why the algorithm produces it or how it could be broken.

4.9.3. Online Cryptography Simulators

Most web cryptography simulators cover classical ciphers—Caesar, Vigenère, Enigma—which are historically useful but disconnected from algorithms students encounter in real security work. They use simplified implementations, show no modern attack methodology, and provide no coverage of structural vulnerabilities in contemporary algorithms. CipherX was built specifically to fill that gap.

4.9.4. Summary Comparison

Table 6 summarizes key differences:

Table 6: Comparison of cryptographic education tools

Feature	CrypTool	CyberChef	Simulators	CipherX
Web-based	No	Yes	Yes	Yes
Modern algorithms	Yes	Yes	No	Yes
Attack demonstrations	Yes	No	No	Yes
Interactive visualization	Limited	No	Basic	Yes
Educational scaffolding	Documentation	None	None	Integrated
Installation required	Yes	No	No	No
Learning curve	Steep	Low	Low	Low
Normalized learning gain	0.40	N/A	N/A	0.62

CipherX fills a gap in the educational tool ecosystem: modern, web-based, attack-focused cryptography learning with interactive visualization, low barrier to entry, and demonstrated learning effectiveness.

5. Discussion

5.1. Educational Design Trade-offs

Creating effective hands-on cryptography education requires balancing several tensions:

- **Fidelity vs Accessibility:** Real cryptographic attacks often require specialized environments (e.g., hardware-side-channel measurement and dedicated cryptanalysis clusters). Researchers chose to implement real algorithm implementations in Python rather than simplified simulations, accepting that timing measurements would be noisy. This preserves authenticity while maintaining classroom deployability.
- **Scope vs Depth:** CipherX focuses on three attack types (equivalent keys, related-key, avalanche) rather than covering all possible attacks. This depth-first approach ensures students develop a genuine understanding rather than superficial exposure.
- **Guidance vs Discovery:** Too much scaffolding prevents discovery learning; too little causes frustration. CipherX provides progressive hints: students first experiment freely, then receive guided questions, then see explanations connecting observations to theory.

6. Conclusion and Implications for Security Education

This paper described CipherX, a browser-based platform for cryptographic attack education, and reported results from a controlled classroom study. Students who worked through attack modules directly developed a better understanding of why TEA's design is structurally weaker than students who received the same information through lecture or CrypTool exercises; the work involved running equivalent key tests, observing avalanche measurements, and tracing related-key correlations. That result held on transfer questions covering scenarios students had not previously encountered. The effect sizes were larger than

anticipated. CipherX students reached normalized learning gains of g = 0.62, compared to g = 0.31 in the lecture group and g = 0.40 in the CrypTool group — an effect size of d = 1.15 against the lecture. Students who had worked through the equivalent key demonstration firsthand showed stronger reasoning about structural cryptographic weaknesses than those who had been told about it. Transfer items provide the more stringent test: those questions covered situations not encountered during the session, so performance on them cannot be attributed to session-specific memory. TEA's equivalent key weakness has been documented since 1996; this work adds nothing to cryptanalysis. What it provides is a structured path for students to work through that known weakness themselves. Students who do that work come away with a practical intuition that reading about the weakness does not reliably produce. The scope is deliberately narrow: TEA and AES, four attack types, one semester. Extending the framework to other algorithm pairs and vulnerability classes is the natural next direction. Whether tools like CipherX produce lasting changes in how graduates reason about security when making real design decisions is a question that only a longitudinal study can address. All materials are open source for adoption and extension.

6.1. Implications for Educators

Security courses that rely primarily on lecture for abstract cryptographic concepts achieve lower learning outcomes than the data here suggests is possible. A two-hour interactive lab session produced substantially better results in this study. The platform, test instruments, rubrics, and analysis scripts are all open source in the artifact repository and available for direct adoption or adaptation to course-specific needs.

6.2. Broader Impact

Developers who understand why a cipher fails make better security decisions than those who only know how to implement one correctly. Building that intuition requires more than reading about vulnerabilities. Tools like CipherX exist to close that gap, not by replacing fundamental instruction, but by giving students the concrete experience needed to connect theory to practical judgment.

6.3. Future Work

Future development has several clear directions: expanding the algorithm set to include XTEA, XXTEA, and lightweight ciphers (PRESENT, SIMON, SPECK) for IoT security contexts; adding post-quantum candidates like Kyber and Dilithium as these enter curricula; integrating with learning management systems for automated grading; and building competitive capture-the-flag challenge modes that let students apply attack techniques in a structured contest format.

6.4. Limitations

Two limitations are worth noting explicitly. First, CipherX covers TEA but not its variants (XTEA, XXTEA), which have different—and generally better—security properties; including them would enable direct comparisons within the TEA family. Second, the classroom study was conducted at a single institution and in a single course offering, limiting the extent to which the results can be generalized without multi-site replication.

Acknowledgment: The authors gratefully acknowledge Mangalore Institute of Technology and Engineering for its support, resources, and academic environment that facilitated the successful completion of this research.

Data Availability Statement: The data for this study can be made available upon request to the corresponding author.

Funding Statement: This manuscript and research paper were prepared without any financial support or funding.

Conflicts of Interest Statement: The authors declare no conflicts of interest related to the publication of this paper.

Ethics and Consent Statement: This research adheres to ethical guidelines, obtaining informed consent from all participants.

References

1. C. E. Shannon, "Communication theory of secrecy systems," *Bell System Technical Journal*, vol. 28, no. 4, pp. 656–715, 1949.
2. N. Ferguson, B. Schneier, and T. Kohno, "Cryptography Engineering: Design Principles and Practical Applications," *Wiley*, Hoboken, New Jersey, United States of America, 2010.

3. B. Esslinger, "The CrypTool Book: Learning and Experiencing Cryptography with CrypTool and SageMath," *CrypTool Project*, Neubiberg, Germany, 2018.
4. GCHQ, "CyberChef - The Cyber Swiss Army Knife," *GCHQ*, 2015. [Accessed by 15/03/2025].
5. D. J. Wheeler and R. M. Needham, "TEA, a Tiny Encryption Algorithm," in Fast Software Encryption, Lecture Notes in Computer Science, *Springer*, Berlin, Germany, 1994.
6. J. Kelsey, B. Schneier, and D. Wagner, "Key-Schedule Cryptoanalysis of IDEA, G-DES, GOST, SAFER, and Triple-DES," in CRYPTO '96, Lecture Notes in Computer Science, *Springer*, Berlin, Germany, 1996.
7. J. Daemen and V. Rijmen, "The Design of Rijndael: AES – The Advanced Encryption Standard," *Springer*, Berlin, Germany, 2002.
8. D. Moon, K. Hwang, W. Lee, S. Lee, and J. Lim, "Impossible Differential Cryptanalysis of Reduced-Round XTEA and TEA," in Fast Software Encryption, Lecture Notes in Computer Science, *Springer*, Berlin, Germany, 2002.
9. S. Hong, D. Hong, Y. Ko, D. Chang, W. Lee, and S. Lee, "Differential Cryptanalysis of TEA and XTEA," in Information Security and Cryptology – ICISC 2003, Lecture Notes in Computer Science, *Springer*, Berlin, Germany, 2004.
10. K. D. Jasper, M. N. Jaishnav, M. F. Chowdhury, R. Badhan, and R. Sivakani, "Defend and Secure: A Strategic and Implementation Framework for Robust Data Breach Prevention," *AVE Trends in Intelligent Computing Systems*, vol. 1, no. 1, pp. 17–31, 2024.
11. A. Naiakshina, A. Danilova, C. Tiefenau, M. Herzog, S. Dechand, and M. Smith, "Why Do Developers Get Password Storage Wrong? A Qualitative Usability Study," in *Proc. ACM SIGSAC Conf. Computer and Communications Security (CCS)*, Dallas, Texas, United States of America, 2017.
12. S. T. Kotagiri, "Exploring Quantum Cryptography for Next-Generation Cybersecurity Protocols," *AVE Trends in Intelligent Computer Letters*, vol. 1, no. 1, pp. 21–30, 2025.
13. S. Bratus, M. E. Locasto, M. L. Patterson, L. Sassaman, and A. Shubina, "Exploit Programming: From Buffer Overflows to 'Weird Machines' and Theory of Computation," *USENIX*, vol. 36, no. 6, pp. 13–21, 2011.
14. B. J. Sruthi, J. P. Jeniffer, K. S. Harshita, M. R. Sulthana, and C. C. Angelin, "Enhancing Certificate Authentication Through a Transparent Blockchain Enabled Verification System," *AVE Trends in Intelligent Computing Systems*, vol. 2, no. 1, pp. 15–26, 2025.
15. M. Burket, P. Chapman, T. Becker, C. Ganas, and D. Brumley, "Automatic Problem Generation for Capture-the-Flag Competitions," in *Proc. USENIX Workshop on Cyber Security Experimentation and Test (CSET)*, Washington, District of Columbia, United States of America, 2015.
16. A. R. P. Reddy, "AI-Powered Anomaly Detection for Cybersecurity Threats in Multi-Cloud Infrastructure," *AVE Trends in Intelligent Computing Systems*, vol. 2, no. 2, pp. 77–86, 2025.
17. W. Du, "SEED: Hands-on Lab Exercises for Computer Security Education," *IEEE Security & Privacy*, vol. 9, no. 5, pp. 70–73, 2011.
18. R. R. Hake, "Interactive-Engagement Versus Traditional Methods: A Six-Thousand-Student Survey of Mechanics Test Data for Introductory Physics Courses," *American Journal of Physics*, vol. 66, no. 1, pp. 64–74, 1998.
19. K. Srivastava, A. Gabhane, A. Ladva, P. Waykole, and S. S. Rajest, "Quantum vs. classical methods in information security, computing and machine learning domains: An empirical study," *International Journal of Engineering Systems Modelling and Simulation*, vol. 16, no. 4, pp. 230–240, 2025.
20. P. C. Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems," in CRYPTO '96, Lecture Notes in Computer Science, *Springer*, Berlin, Germany, 1996.
21. H. L. O'Brien and E. G. Toms, "What is User Engagement? A Conceptual Framework for Defining User Engagement with Technology," *Journal of the American Society for Information Science and Technology*, vol. 59, no. 6, pp. 938–955, 2008.
22. J. Brooke, "SUS: A 'Quick and Dirty' Usability Scale," in Usability Evaluation in Industry, *CRC Press*, Florida, United States of America, 1996.
23. R. Regin, A. A. Khanna, V. Krishnan, M. Gupta, S. Rubin Bose, and S. S. Rajest, "Information design and unifying approach for secured data sharing using attribute-based access control mechanisms," in Recent Developments in Machine and Human Intelligence, *IGI Global*, Hershey, Pennsylvania, United States of America, 2023.