CAUCHY – Jurnal Matematika Murni dan Aplikasi
Volume 10 (2) (2025), Pages 1026-1042
p-ISSN: 2086-0382; e-ISSN: 2477-3344

An Extended Cryptanalysis of Peyrin *et al.* on SIMON-JAMBU64/96: A Study on Reduced-Tuple Attacks

Susila Windarta[1*], Wuri Handayani[2], and Bety Hayat Susanti[3]

[1,3]*Dept. of Cybersecurity Engineering, Politeknik Siber dan Sandi Negara, Indonesia*
[2]*Badan Siber dan Sandi Negara, Indonesia*

Abstract

This study extends the cryptanalysis of Peyrin et al. on the SIMON-JAMBU64/96 Authenticated Encryption (AE) scheme, aiming to evaluate its feasibility under a reduced-tuple scenario. We combine formal analysis and experimental validation by constructing collision-based distinguishers in a chosen-IV model and testing them through a decryption oracle. The results demonstrate that a distinguishing attack can be successfully performed with only two tuples one fewer than previously required indicating that JAMBU is more fragile than initially assumed. A detailed comparison of data complexity shows that the two-tuple attack achieves a lower cost in the second phase (4×2^{48} queries) than the original three-tuple attack (6×2^{48}), with slightly higher verification overhead in the third phase. Overall, these findings reaffirm that the SIMON-JAMBU64/96 scheme remains susceptible to distinguishing, plaintext-forgery, and plaintext-recovery attacks, thereby extending the analysis of Peyrin et al. to scenarios with more constrained adversarial resources.

Keywords: Authenticated Encryption, Distinguishing Attack, Nonce-Respecting, SIMON-JAMBU64/96, Two-Tuple Attack.

1 Introduction

Authenticated Encryption (AE) is a cryptographic scheme that simultaneously provides confidentiality and authentication of data, making it a fundamental component in securing digital communications [1]. The National Institute of Standards and Technology (NIST) has standardized several AE schemes, including Counter with Cipher Block Chaining–Message Authentication Code (CCM), Key Wrapping (KW), and Galois Counter Mode (GCM) [2], [3]. To encourage the development of schemes more secure and efficient than AES-GCM, NIST and Daniel J. Bernstein launched the CAESAR (Competition for Authenticated Encryption: Security, Applicability, and Robustness) competition in 2013 [4]. Among the finalists was JAMBU, a lightweight AE mode proposed by Wu and Huang in 2014, designed with small nonce and tag sizes suitable for resource-constrained environments [5].

The need for lightweight AE schemes is particularly urgent in the Internet of Things (IoT) ecosystem. By the end of 2023, there were approximately 16.6 billion connected IoT devices globally, projected to reach about 18.8 billion by 2024 and nearly 40 billion by 2030 [6], [7]. In

[*]**Corresponding author**. E-mail: susila.windarta@polteknssn.ac.id

Submitted: September 29, 2025 Reviewed: October 07, 2025 Accepted: October 16, 2025
DOI: https://doi.org/10.18860/cauchy.v10i2.36601

Indonesia, IoT adoption is accelerating: in 2023, the number of IoT devices exceeded 150 million, up from about 100 million in 2022 [8]. The Indonesian IoT market is expected to reach USD 13.05 billion in 2025 and double to USD 26.50 billion by 2030, with a CAGR of 15.21% [9]. With such rapid expansion, IoT networks face growing exposure to cyber threats such as eavesdropping, replay and forgery attacks, man-in-the-middle intrusions, and large-scale botnets like Mirai [10], [11], [12], [13]. These threats often exploit weaknesses in authentication, encryption, or nonce management, underscoring the need for lightweight and secure AE schemes.

Recent research on lightweight AEAD schemes for constrained IoT environments highlights the trade-offs between security and efficiency in block-cipher-based designs such as SIMON, SPECK, PRESENT, and ASCON [14]. Other studies have explored combining lightweight primitives, e.g., LED and PHOTON, to construct efficient AE schemes for edge devices [15]. Broader surveys on IoT security emphasize AEAD as a cornerstone of secure communication [16], while related works examine lightweight mutual authentication [17] and the cryptanalysis of AEAD modes, such as COLM [18]. Together, these studies demonstrate that AE design for IoT must balance efficiency with robustness against adversarial misuse.

From a theoretical standpoint, AE schemes are analyzed under two models: nonce-respecting, where nonces are unique, and nonce-misuse, where nonce repetition may occur. In 2015, Peyrin *et al.* analyzed JAMBU version 1 and identified vulnerabilities under both models by exploiting its linear structure, including distinguishing, forgery, and plaintext-recovery attacks [19]. This analysis complements recent evaluations of AEAD modes (e.g., COLM [18], ASCON [14]) by showing that tuple-reduction-based cryptanalysis remains relevant for lightweight AE modes, offering deeper insight into JAMBU's security in constrained IoT contexts.

Building on Peyrin *et al.*'s findings, this paper makes the following key contributions:

1. We apply and empirically validate the cryptanalysis of Peyrin *et al.* on the SIMON-JAMBU64/96 scheme in the nonce-respecting scenario.
2. We extend the attack by showing it is feasible with only two tuples instead of three, revealing a greater vulnerability in the scheme.
3. We analyze the trade-off between the number of tuples, attack steps, and data complexity, demonstrating the efficiency of our extended method.

This paper is organized as follows: Section 2 introduces the theoretical basis of JAMBU and the underlying cryptanalytic framework. Section 4 details the methodology and implementation choices. Section 5 presents the outcomes of the distinguishing, forgery, and recovery attacks, and compares the two-tuple and three-tuple approaches. The final section concludes with implications for JAMBU's security and directions for future research.

2 Theoretical Basis

The cryptanalytic foundation of this work builds upon the structural properties and differential behaviors previously identified in JAMBU and related authenticated encryption modes. To establish a coherent baseline, this section revisits the relevant theoretical constructs—particularly those derived from Peyrin *et al.*—and reformulates them under the two-tuple assumption. The following subsections describe these principles, emphasizing how the modified framework supports the reduced data complexity and preserved differential dependencies observed in this study.

2.1 Notation

To ensure clarity and provide a convenient reference for the reader, we define the key notations used in our description of the JAMBU scheme and its cryptanalysis. The symbols are summarized in Table 1.

Table 1: List of Notations and Symbols

Symbol	Description
General Parameters and Functions	
AE	Authenticated Encryption
K	Secret key
M, P	Plaintext message
C	Ciphertext
T	Authentication tag
IV	Initialization Vector or Nonce
AD	Associated Data
n	Size of a plaintext/AD block in bits (e.g., 32)
$E_K(\cdot)$	Block cipher encryption function with key K
$D(\cdot)$	Decryption oracle
$\perp$	Error symbol returned on verification failure
JAMBU Internal State Variables	
S_i	The main $2n$-bit state register at step i
R_i	The n-bit accumulator state register at step i
X_i, Y_i	The two n-bit halves of the block cipher output at step i
U_i, V_i	The two n-bit halves of the state S_i
Cryptanalysis Notation	
X'	A variable corresponding to a second run of the cipher
ΔX	The bitwise difference $X \oplus X'$
c	The number of candidates found in the two-tuple attack
m	The number of samples per list in the birthday attack
Mathematical Operators	
$\oplus$	Bitwise XOR operation
$\|$	Concatenation operation
$\&$	Bitwise AND operation
S^j	Circular left shift by j bits

2.2 Authenticated Encryption

AE is a cryptographic scheme that simultaneously provides both data confidentiality and authenticity. Confidentiality is achieved through encryption to protect the secrecy of a message, while authenticity ensures its integrity [20].

AE schemes were developed as integrated solutions to improve upon generic "composition" methods, which combine separate encryption and authentication primitives. The most notable generic compositions include:

- Encrypt-then-MAC (EtM): The message (M) is encrypted to produce ciphertext (C), and then a MAC tag (T) is computed over the ciphertext. This composition is used in protocols like IPsec.
- MAC-then-Encrypt (MtE): A MAC tag (T) is computed on the plaintext message (M), and then both the message and the tag are encrypted together. This approach is used in TLS.

The security of an AE scheme is defined by two primary objectives [1]:

- Confidentiality: A passive adversary, given access to ciphertexts and their corresponding nonces, should not be able to learn any information about the underlying plaintexts. This is often achieved by making the ciphertext indistinguishable from a random bitstring.
- Authenticity (Integrity): An active adversary should not be able to forge a new, valid tuple of (ciphertext, nonce, tag) that successfully decrypts.

An AE encryption function takes a secret key (K), a nonce (IV), associated data (AD), and a plaintext message (M) as input to produce a ciphertext (C) and an authentication tag (T). The corresponding decryption function takes K, IV, AD, C, and T as input. It returns the original plaintext M only if the tag is verified as valid. If verification fails, it returns an error symbol ($\perp$).

The components play crucial roles: the nonce (IV) ensures that ciphertexts are unique even if the same plaintext is encrypted multiple times, while the associated data (AD) allows for public, non-confidential information (e.g., packet headers) to be authenticated alongside the private data, thus preventing forgery attacks on these public parts.

2.3 The JAMBU AE Scheme

JAMBU [5], [21] is an authenticated encryption (AE) scheme that operates in a mode similar to Output Feedback (OFB), built on a block cipher with a $2n$-bit block size and a K-bit key, denoted $E_K(\cdot)$. The scheme's internal state is maintained across two primary state registers: a $2n$-bit register S and an n-bit register R. The fundamental operations used are bitwise XOR ($\oplus$) and concatenation ($\|$).

The overall process is divided into two main functions: encryption/authentication and decryption/verification.

Encryption and Authentication Process

The encryption function transforms a secret key K, a public nonce IV, associated data AD, and a plaintext P into a ciphertext C and an authentication tag T. This process is accomplished in five distinct phases:

1. **Padding:** Before processing, both the associated data and the plaintext are padded using a '10*' scheme. A single bit '1' is appended, followed by the minimum number of '0' bits required to make the total length a multiple of n bits.
2. **Initialization:** The process begins by loading the n-bit IV into a $2n$-bit initial state, $S_{-1} = (0^n \| IV)$. This state is encrypted once to produce $(X_{-1}, Y_{-1}) = E_K(S_{-1})$. This output is then used to initialize the main state registers:
   - The accumulator register is set: $R_0 = X_{-1}$.
   - The main state is set: $S_0 = (X_{-1}, Y_{-1} \oplus 5)$, where the constant is for domain separation.
3. **Associated Data (AD) Processing:** The padded AD is processed one n-bit block at a time. For each block AD_i, the state is updated iteratively:
   (a) The current state S_i is encrypted: $(X_i, Y_i) = E_K(S_i)$.
   (b) The next state $S_{i+1} = (U_{i+1}, V_{i+1})$ is computed, where $U_{i+1} = X_i \oplus AD_i$ and $V_{i+1} = Y_i \oplus R_i \oplus 1$.
   (c) The accumulator register is updated: $R_{i+1} = R_i \oplus U_{i+1}$.
4. **Plaintext Encryption:** This phase follows the same iterative structure as AD processing. For each plaintext block P_i:
   (a) The state update for V_{i+1} does not include the constant: $V_{i+1} = Y_i \oplus R_i$.
   (b) The ciphertext block is generated by XORing the plaintext with the keystream: $C_i = P_i \oplus V_{i+1}$.
   (c) The state registers S_{i+1} and R_{i+1} are updated using the plaintext block P_i to influence subsequent steps.
5. **Finalization and Tag Generation:** After all plaintext blocks are processed, two final state updates are performed, with the first using the constant '3' for domain separation. The final authentication tag T is then computed by XORing the final R state with the components of the final block cipher output: $T = R_{final} \oplus X_{final} \oplus Y_{final}$.

Decryption and Verification Process

The decryption process mirrors the encryption steps to regenerate the same sequence of internal states. First, the Initialization and Associated Data processing phases are performed identically to encryption, using the received IV and AD. Then, for each ciphertext block C_i, the plaintext P_i is recovered by XORing the ciphertext with the corresponding keystream block V_{i+1}, which is generated from the internal state: $P_i = C_i \oplus V_{i+1}$. The recovered plaintext P_i is immediately used to update the internal state in the same manner as during encryption, ensuring the state remains synchronized.

After all ciphertext blocks are processed, the Finalization phase is executed to compute a local tag, T'. This computed tag is then compared to the received tag T.

- If $T' = T$, the verification is successful, and the decrypted plaintext P is released as valid.
- If $T' \neq T$, the verification fails. The process returns an error symbol ($\perp$), and the plaintext is discarded.

Notably, although JAMBU formally defines $R_{i+1} = R_i \oplus U_{i+1}$, our analysis reveals a persistent linear dependency across iterations, allowing ΔR to remain constant under certain decryption sequences.

2.4 Peyrin *et al.* Cryptanalysis under Nonce-Respecting Scenario

We consider an adaptive chosen-ciphertext setting where the adversary has access to a decryption oracle and may submit ciphertexts (and associated data) to obtain plaintexts. The attack focuses on exploiting the linear properties of JAMBU, specifically where an internal state R is not updated during each iteration of the block cipher. This leads to a linear relationship, where $\Delta R = R \oplus R'$ is a constant. The cryptanalysis is supported by the following lemmas (Lemma 1, Lemma 2, and Lemma 3), which prove the linear relationships exploited in the attack.

Lemma 1. *Let two decryption requests be made for ciphertexts C_1 and C_1' under the JAMBU scheme, satisfying $C_1 = C_1'$, $IV \neq IV'$, $AD_1 = AD_1'$, $X_0 = X_0'$, and $\Delta X_{-1} = \Delta R_0 = \Delta R$. If $\Delta R_1 = \Delta Y_1$, then $\Delta V_2 = \mathbf{0}$.*

Proof. Since $\Delta X_0 = \Delta AD_1 = 0$, the round update yields $\Delta R_1 = \Delta R_0 = \Delta R$. Given $\Delta R_1 = \Delta Y_1$, both inputs to the block function that generates V_2 become identical, resulting in $V_2 = V_2'$ and $\Delta V_2 = 0$. This behavior is depicted in Figure 1, where the equality of differentials $\Delta R_1 = \Delta Y_1$ prevents further propagation across the nonlinear operation. □

Lemma 2. *Consider two decryption requests for ciphertexts $C_1||C_2$ and $C_1'||C_2'$ under the JAMBU scheme, with $P_1 \neq P_1'$, $P_2 = P_2'$, and $AD_1 = AD_1'$. If $\Delta P_1 = \Delta X_1$, then $\Delta U_2 = \mathbf{0}$.*

Proof. From $\Delta P_1 = \Delta X_1$, it follows that $P_1 \oplus X_1 = P_1' \oplus X_1'$. Since $U_2 = P_1 \oplus X_1$ in the JAMBU decryption structure, both computations yield the same internal value, hence $U_2 = U_2'$ and $\Delta U_2 = 0$. Figure 2 illustrates this propagation, showing how matching inputs (P_1, X_1) lead to the cancellation of the second-round differential. □

Lemma 3. *Let two decryption requests be made for ciphertexts $C_1||C_2$ and $C_1'||C_2'$ under initialization vectors IV and IV', and let $S_2 = (V_2, U_2)$ with $\Delta S_2 = S_2 \oplus S_2'$. If $\Delta S_2 = 0$, then $\Delta V_3 = \Delta R$.*

Proof. When U_2 and V_2 are identical across both executions, the next-round differential depends solely on the register update, giving $\Delta V_3 = \Delta R$.

□

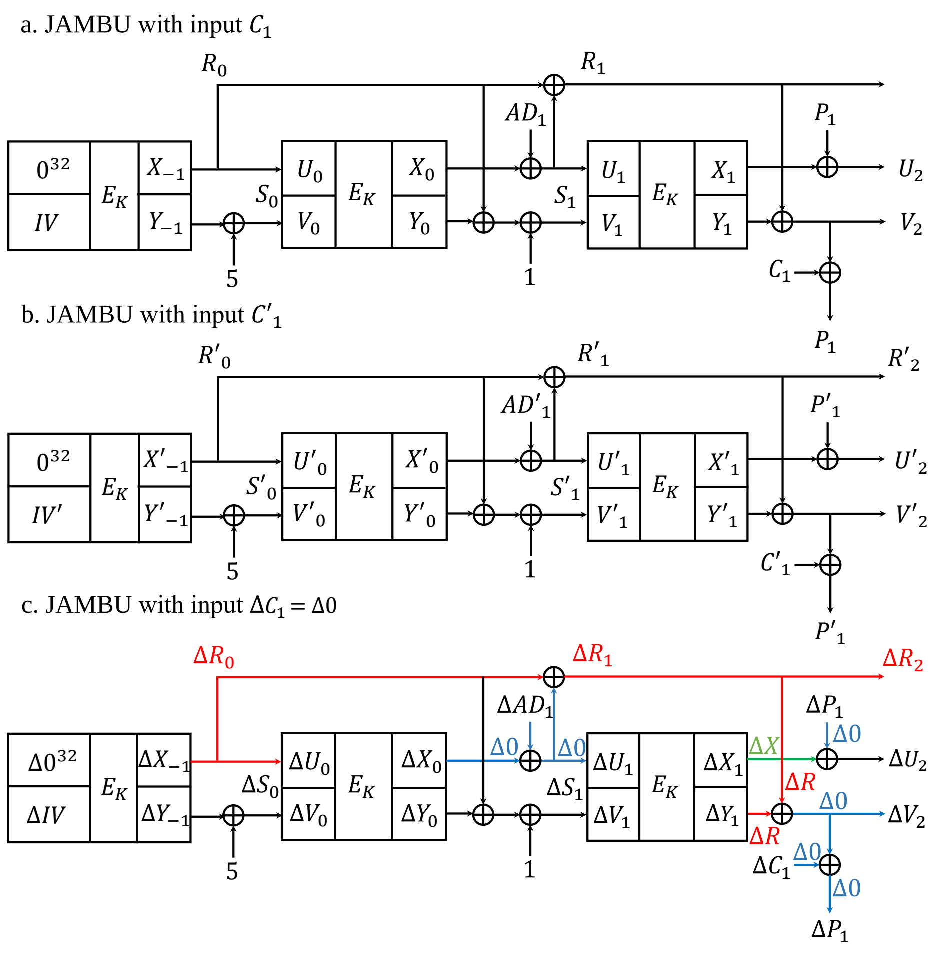

Figure 1: First observation of the differential structure of the JAMBU decryption scheme, showing that when $\Delta R_1 = \Delta Y_1$, both inputs to $f(\cdot)$ produce identical V_2.

The lemmas above follow the derivations in Peyrin *et.al* [19], with the key dependencies on the two-tuple assumption retained. Figures 1–2 respectively illustrate: (1) single-round cancellation when $\Delta R_1 = \Delta Y_1$; (2) two-round propagation through X_1, U_2, and P_2; and (3) the linkage between the synchronized pair (V_2, U_2) and the register update ΔR. These figures confirm that the linear dependencies remain valid under the reduced two-tuple model.

Using these lemmas, we search for conditions yielding $\Delta S_2 = \mathbf{0}$; once found, the linear relation $\Delta V_3 = \Delta R$ can be exploited. The overall cryptanalytic strategy comprises three attacks: a distinguishing attack, a plaintext forgery attack, and a plaintext-recovery attack. The distinguishing attack (the first step) itself proceeds in three stages: (i) find a colliding (IV, IV') pair (subject to the threat-model constraints on IV selection), (ii) determine candidate differences $(\Delta X, \Delta R)$, and (iii) verify these candidates against the decryption oracle.

3 The SIMON Block Cipher

SIMON is a lightweight block cipher designed by the U.S. National Security Agency (NSA) in 2013 to be highly efficient in both hardware and software [22], [23]. It is built on a Feistel network, a classic cipher structure that uses very simple bitwise operations: XOR ($\oplus$), AND (&), and circular left shifts (S^j).

The specific variant used in the JAMBU scheme is SIMON64/96. As the name implies, it operates on a 64-bit block of data with a 96-bit key and performs 42 rounds of encryption.

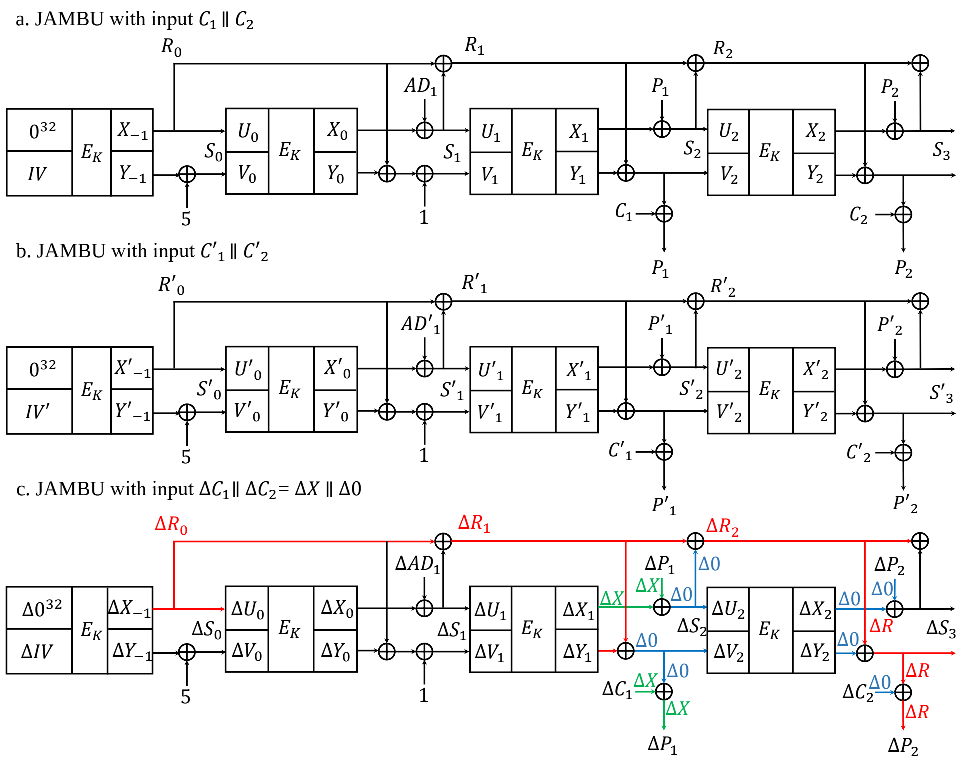
a. JAMBU with input $C_1 \| C_2$
b. JAMBU with input $C'_1 \| C'_2$
c. JAMBU with input $\Delta C_1 \| \Delta C_2 = \Delta X \| \Delta 0$
E_K
R_0
R_1
R_2
AD_1
P_1
P_2
C_1
C_2
S_0
S_1
S_2
S_3
5
1

Figure 2: Second observation of the differential structure of the JAMBU decryption scheme, showing that if $\Delta P_1 = \Delta X_1$ and $P_2 = P'_2$, then $U_2 = U'_2$ and $\Delta U_2 = 0$.

Key Schedule The key schedule's job is to expand the 96-bit master key into a sequence of 42 different 32-bit round keys (K_i), one for each round. It begins by splitting the 96-bit key into three 32-bit words. It then uses a recursive formula to generate the remaining round keys, ensuring that each round uses a unique key derived from the master key.

Round Function The round function is the core engine of the cipher where the data gets scrambled. In each of the 42 rounds, the 64-bit data block is split into two 32-bit halves, a left half (L_i) and a right half (R_i). The Feistel structure works as follows:

1. The left half (L_i) is put through a scrambling function, f, which involves several shifts and bitwise operations.
2. The output of $f(L_i)$ is XORed with the right half (R_i) and the current round key (K_i). This result becomes the new left half for the next round (L_{i+1}).
3. The original, unmodified left half (L_i) becomes the *new* right half (R_{i+1}).

This process is repeated for 42 rounds. Each round further obscures the relationship between the plaintext and the final ciphertext, ensuring the encryption is secure.

4 Methods

Building upon the theoretical framework established in the previous section, this part details the methodology employed to validate the proposed two-tuple attack. The approach integrates analytical modeling, controlled simulation, and empirical validation to ensure consistency between theoretical predictions and observed outcomes. The following subsections describe the experimental design, parameter configuration, and verification process adopted throughout the study.

4.1 Research Approach

This study combines a literature review and an experimental approach. The literature study involved reviewing works on authenticated encryption (AE), the JAMBU scheme, the SIMON block cipher, and Peyrin *et al.*'s analysis model. The experimental component focuses on reconstructing Peyrin *et al.*'s cryptanalysis under the *nonce-respecting* scenario for the SIMON-JAMBU64/96 configuration. All algorithms were implemented in the C programming language using Dev-C++ with the TDM-GCC compiler.

4.2 Research Variables

The cryptanalysis experiment uses several variables as follows:

- Independent Variables: Initialization Vector (IV), Ciphertext blocks (C_1, C_2), and authentication tag T.
- Dependent Variables: Plaintext blocks (P_1, P_2), and the differential values ($\Delta X, \Delta R$) derived from decryption observations.
- Controlled Variables: Secret key $K =$ `0x131211100b0a090803020100` and associated data $AD =$ `0x10000000`, which remained constant across all experiments.

4.3 Data Modification Technique

Two IV generation techniques were used to observe the influence of nonce structure on the attack results:

- Counter Increment: IV values were generated sequentially to simulate predictable nonces.
- Random Generation: IV values were produced pseudorandomly to simulate fully random nonce use.

Each method produced five distinct datasets, where each dataset contained 2^{16} IVs used for decryption requests.

4.4 Research Stages

The experimental process consisted of three main stages, corresponding to the phases outlined in Peyrin *et al.*.

1. Differential Structure Observation. The first stage analyzed the differential propagation in JAMBU's decryption phase by tracking internal state transitions (X, Y, R). The experiment identified ($\Delta X, \Delta R$) pairs that caused linear dependencies between plaintext and ciphertext blocks.
2. Distinguishing Attack. This stage performed the differential distinguishing experiment on the decryption oracle to detect structured differences in output. The experiment was repeated for both IV generation techniques. In total, ten trials (five for each IV type) were conducted, and within each trial, approximately $2^{16} + 4 \cdot 2^{16} + 2 \cdot 2^{32} \cdot m$ decryption queries were issued, where m denotes the number of ($\Delta X, \Delta R$) candidates.
3. Plaintext Forgery and Recovery. Using the valid ($\Delta X, \Delta R$) obtained from the distinguishing stage, a forgery experiment was conducted to construct valid ciphertext–plaintext pairs without oracle assistance. Subsequently, plaintext recovery experiments were executed to reconstruct unknown plaintexts from their ciphertexts. Each of these phases was repeated ten times (five counter-increment and five random IVs), producing 20 experimental outputs overall for comparison.

4.5 Number of Experiments

To ensure reproducibility and statistical confidence, each cryptanalytic phase was performed on both IV generation techniques, resulting in:

- 5 experiments using counter-increment IV generation,
- 5 experiments using random IV generation,
- for a total of 10 complete nonce-respecting experiments.

The results from each trial were averaged to determine the effective data complexity and the success probability of the distinguishing, forgery, and plaintext recovery attacks.

4.6 Data Analysis

Experimental data consisted of the observed differential pairs $(\Delta X, \Delta R)$, the number of successful differentiations, and valid ciphertext collisions. Each experiment produced the following metrics:

- the success rate of finding correct $(\Delta X, \Delta R)$ pairs;
- the probability of valid ciphertext–plaintext forgeries;
- and the required number of oracle queries for a successful distinguishing attack.

A comparative analysis between random and counter-increment IV sets revealed that both methods exhibited identical structural vulnerabilities under nonce-respecting conditions, with minor differences in the number of differential collisions and data complexity. All ten experiments consistently validated Peyrin *et al.*'s theoretical attack model.

5 Results and Discussion

This section presents the findings from the experimental application of the cryptanalysis. The successful execution of the attacks validates the vulnerability of SIMON-JAMBU64/96 and confirms the viability of the extended two-tuple attack method. The cryptanalysis is formalized in Algorithms 1, 2, and 3.

5.1 Distinguishing Attack (Nonce-Respecting Scenario)

The experimental results confirmed that the distinguishing attack can be successfully performed on SIMON-JAMBU64/96.

- **Phase 1**: Pairs of (IV, IV') that produced collisions on P_1 were successfully found. Randomly generated IVs produced more collision pairs than counter-incremented IVs, as shown in Table 2.
- **Phase 2**: The two-tuple attack successfully found the correct $(\Delta X, \Delta R)$ pair with a lower data complexity $(4 \cdot 2^{48})$ compared to the three-tuple attack $(6 \cdot 2^{48})$.
- **Phase 3**: The correct $(\Delta X, \Delta R)$ was confirmed by checking if $\Delta P_2 = \Delta R$. For the two-tuple attack, multiple candidates for $(\Delta X, \Delta R)$ were found (ranging from 2 to 6 in the experiments), which all needed to be verified. Table 3 provides an example of this verification process.

Table 2: Example of (IV, IV') pairs that produce a collision on P_1 (random IVs)

IV	**IV'**	P_1
0xb3471f6f	0xa0d1b5af	0x12345678
0x6482f64b	0x1bc8ee23	0x2afa7f98
0xff47d5f0	0xa6a6f113	0x07d34e65
0x7b87b50b	0x24661012	0xdb080f1f

Table 3: Example of correct ΔR values confirmed in Phase 3

IV, IV'	P_2	P'_2	$P_2 \oplus P'_2$	ΔR
0xb3471f6f, 0xa0d1b5af	0xbe27a24d	0xca03119e	0x7424b3d3	0x7424b3d3
0xb3471f6f, 0xa0d1b5af	0xed5b18a8	0x537cbae5	0x8be8b0b9	0x8be8b0b9

Table 4: Comparison of Distinguishing Attack Steps for three-tuple and two-tuple Methods

Attack Phase	**3-tuple Method (original)**	**2-tuple Method (extended)**
Phase 1: Finding (IV, IV')	Search for (IV, IV') that produce a collision on P_1. Example: $(IV = 0xb3471f6f, IV' = 0xa0d1b5af)$ yields $P_1 = 0x12345678$.	Same as the three-tuple method.
Phase 2: Finding $(\Delta X, \Delta R)$	Find collision on a pair of plaintext differences $(\Delta P[\langle i \rangle]_1, \Delta P[\langle i \rangle]_2)$ and $(\Delta P'[\langle j \rangle]_1, \Delta P'[\langle j \rangle]_2)$. This directly yields a single correct $(\Delta X, \Delta R)$ pair.	Find collision on a single plaintext difference $\Delta P[\langle i \rangle]_1$ and $\Delta P'[\langle j \rangle]_1$. This yields multiple candidate $(\Delta X, \Delta R)$ pairs (c candidates).
Phase 3: Verification	Verify the single $(\Delta X, \Delta R)$ pair by checking if $\Delta P_2 = \Delta R$. Example: $\Delta P_2 = 0x7424b3d3$ matches $\Delta R = 0x7424b3d3$.	Verify all c candidates until one satisfies the condition $\Delta P_2 = \Delta R$. The correct pair is found after testing.

As illustrated in Table 4, our extended two-tuple attack introduces a significant strategic trade-off compared to the original three-tuple method proposed by Peyrin *et al.* While both approaches share an identical first phase for finding a colliding $((IV, IV'))$ pair, they diverge fundamentally in the subsequent search and verification phases.

The primary distinction arises in Phase 2. The original method employs a more stringent collision condition, requiring a match on a pair of plaintext differences. This complexity acts as a strong filter, directly yielding a single, high-probability $(\Delta X, \Delta R)$ candidate. In contrast, our two-tuple method relaxes this requirement, searching for a collision on only a single plaintext difference. This simplification makes the search more efficient in terms of data complexity, but, as a consequence, produces multiple candidate pairs, which include false positives.

Consequently, the verification process in Phase 3 differs significantly. For the three-tuple method, this stage is a simple confirmation of the single candidate. For our two-tuple method, this phase becomes a brute-force process that requires testing each of the c candidates until the one satisfying the condition $\Delta P_2 = \Delta R$ is found.

This represents a classic cryptanalytic trade-off: the two-tuple method shifts the burden from online data complexity (the number of queries to the decryption oracle in Phase 2) to offline time complexity (the computational effort required to test candidates in Phase 3). This trade-off is advantageous in scenarios where interactions with the oracle are limited, expensive, or monitored, as it allows the attacker to minimize their online footprint at the cost of increased offline computation.

5.2 Plaintext Forgery and Plaintext-Recovery Attack

Once the correct $(\Delta X, \Delta R)$ was identified, both plaintext forgery and plaintext-recovery attacks were successfully performed.

- **Plaintext Forgery**: The attack demonstrated that a valid forged ciphertext can be created without a direct encryption query. For example, a forged plaintext $P_1^D \| P_2^D$ of 0x28047155be27a24d was created, which corresponded to a valid forged ciphertext $C_1^D \| C_2^D$ of 0xe51c9d4903856f49.
- **Plaintext-Recovery Attack**: The attack successfully recovered the original plaintext from

a given ciphertext. For example, given the ciphertext $(C_1 \oplus \Delta X) \| C_2$ as 0xe51c9d4977a1dc9a, the corresponding plaintext $P'_1 \| P'_2$ was successfully recovered as 0x28047155ca03119e.

Algorithm 1 Distinguishing Attack on SIMON-JAMBU64/96

```
Require: Decryption oracle $D(IV, AD, C) \to P$, a fixed Associated Data $AD$, block size $n = 32$.
Ensure: A verified differential pair $(\Delta X, \Delta R)$.
 1: {Stage 1: Find a Colliding IV Pair}
 2: Initialize a hash map 'KnownPlaintexts'.
 3: Choose a constant first ciphertext block, $C_1$.
 4: for $i = 1, 2, \dots$ (up to $\approx 2^{n/2}$ queries) do
 5:   Generate a random Initialization Vector, $IV_i$.
 6:   Query the oracle: $P_{1,i} \leftarrow D(IV_i, AD, C_1)$.
 7:   if $P_{1,i}$ exists in 'KnownPlaintexts' with a stored $IV_j$ then
 8:     Set $IV \leftarrow IV_j$ and $IV' \leftarrow IV_i$.
 9:     Calculate the state difference: $\Delta R \leftarrow IV \oplus IV'$.
10:     goto Stage 2. {Collision found, proceed}
11:   else
12:     Store the pair $(P_{1,i}, IV_i)$ in 'KnownPlaintexts'.
13:   end if
14: end for
15: {Stage 2: Find $\Delta X$ (two-tuple Method)}
16: Initialize lists $L_1$, $L_2$, and Candidate_ΔX.
17: for $i = 1$ to $2^{n/2}$ do
18:   Choose a random ciphertext block $C_{1,i}$.
19:   Query oracle: $P_{1,i} \leftarrow D(IV, AD, C_{1,i})$.
20:   Store $(C_{1,i}, P_{1,i})$ in $L_1$.
21: end for
22: for $j = 1$ to $2^{n/2}$ do
23:   Choose a random ciphertext block $C'_{1,j}$.
24:   Query oracle: $P'_{1,j} \leftarrow D(IV', AD, C'_{1,j})$.
25:   Search $L_1$ for any entry $(C_{1,i}, P_{1,i})$ where $P_{1,i} = P'_{1,j}$.
26:   if a collision is found then
27:     Calculate candidate difference: $\Delta X_{cand} \leftarrow C_{1,i} \oplus C'_{1,j}$.
28:     Add $\Delta X_{cand}$ to the Candidate_ΔX list.
29:   end if
30: end for
31: {Stage 3: Verify the $(\Delta X, \Delta R)$ Pair}
32: Choose a random, constant second ciphertext block $C_2$.
33: for each $\Delta X_{cand}$ in Candidate_ΔX do
34:   $C_{query1} \leftarrow (C_1 \oplus \Delta X_{cand}) \| C_2$.
35:   $C_{query2} \leftarrow C_1 \| C_2$.
36:   $P_1 \| P_2 \leftarrow D(IV, AD, C_{query1})$.
37:   $P'_1 \| P'_2 \leftarrow D(IV', AD, C_{query2})$.
38:   Calculate difference: $\Delta P_2 \leftarrow P_2 \oplus P'_2$.
39:   if $\Delta P_2 = \Delta R$ then
40:     Set $\Delta X \leftarrow \Delta X_{cand}$.
41:     return $(\Delta X, \Delta R)$. {Correct pair found}
42:   end if
43: end for
```

5.3 Complexity Analysis

The following theorem quantifies the expected number of collisions and the query complexity of the two-tuple and three-tuple methods.

Theorem 1 (Collision counts and query complexity)**.** *Let $\mathcal{U}$ be a uniform random variable taking values in an n-bit space of size 2^n. Let two lists $\mathcal{L}_1, \mathcal{L}_2$ each contain m independent samples drawn (with replacement) from $\mathcal{U}$. Define the number of* pairwise *collisions*

$$C_2 \; = \; \#\{(x,y) \in \mathcal{L}_1 \times \mathcal{L}_2 \mid x = y\},$$

and let three lists $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3$ of size m define the number of triple *collisions*

$$C_3 \; = \; \#\{(x,y,z) \in \mathcal{L}_1 \times \mathcal{L}_2 \times \mathcal{L}_3 \mid x = y = z\}.$$

Then:

1. $\mathbb{E}[C_2] = \dfrac{m^2}{2^n}$. *Consequently, choosing $m = \Theta(2^{n/2})$ yields $\mathbb{E}[C_2] = \Theta(1)$ (constant expected number of pairwise collisions).*
2. $\mathbb{E}[C_3] = \dfrac{m^3}{2^{2n}}$. *Consequently, choosing $m = \Theta(2^{2n/3})$ yields $\mathbb{E}[C_3] = \Theta(1)$ (constant expected number of triple collisions).*
3. *Let Q_2 (resp. Q_3) denote the oracle query complexity needed to construct the involved lists for the two-tuple (resp. three-tuple) method. Then, under the choices above that make the expected number of collisions constant,*

$$Q_2 = \Theta(2^{n/2}), \qquad Q_3 = \Theta(2^{2n/3}).$$

   In particular, for all sufficiently large n we have $Q_2 \ll Q_3$, i.e. the two-tuple strategy requires asymptotically fewer oracle queries than the three-tuple strategy.

Moreover, if collisions are rare (so $\lambda \equiv \mathbb{E}[C_2] \ll 1$), then

$$\Pr[C_2 \geq 1] \approx 1 - e^{-\lambda},$$

and for $\lambda = \Theta(1)$ this probability is a positive constant bounded away from 0.

Proof. We compute expectations by linearity, counting collisions per target value.

(1) Let the universe be S with $|S| = 2^n$. For any fixed value $v \in S$, let

$$I_v \; = \; \#\{(x,y) \in \mathcal{L}_1 \times \mathcal{L}_2 : x = y = v\}.$$

Then, $C_2 = \sum_{v \in S} I_v$. For fixed v, each of the m^2 ordered pairs (x,y) equals v with probability $2^{-n} \cdot 2^{-n} = 2^{-2n}$ if we required both to be exactly v; however, an easier and standard route is to observe:

$$\Pr[x = v] = 2^{-n}, \qquad \Pr[y = v] = 2^{-n},$$

and for independent draws

$$\Pr[x = v \text{ and } y = v] = 2^{-n} \cdot 2^{-n} = 2^{-2n}.$$

There are m^2 ordered pairs (x,y), so

$$\mathbb{E}[C_2] = m^2 \cdot 2^{-2n} \cdot 2^n = \frac{m^2}{2^n}.$$

(An equivalent and perhaps more transparent combinatorial derivation: for each of the m elements of $\mathcal{L}_1$ and each of the m elements of $\mathcal{L}_2$, the probability they are equal (to some common value) is 2^{-n}; summing yields $m^2/2^n$.)

Therefore, setting $m = 2^{n/2}$ gives

$$\mathbb{E}[C_2] = \frac{(2^{n/2})^2}{2^n} = \frac{2^n}{2^n} = 1,$$

so $m = \Theta(2^{n/2})$ yields $\mathbb{E}[C_2] = \Theta(1)$.

(2) For triple collisions, let for each $v \in S$ the indicator

$$J_v := \mathbf{1}\{\text{there exists } x \in \mathcal{L}_1, y \in \mathcal{L}_2, z \in \mathcal{L}_3 \text{ with } x = y = z = v\}.$$

The number of ordered triples equal to v is distributed with expectation $m^3 \cdot 2^{-2n}$ (since the event that a fixed triple (x, y, z) equals the same fixed v has probability $(2^{-n})^3$, and summing over 2^n possible v removes one factor of 2^{-n}). More directly,

$$\mathbb{E}[C_3] = \sum_{v \in S} \mathbb{E}[\#\{(x, y, z) : x = y = z = v\}] = 2^n \cdot (m \cdot 2^{-n})^3 = \frac{m^3}{2^{2n}}.$$

To make $\mathbb{E}[C_3] = \Theta(1)$, solve $m^3/2^{2n} = \Theta(1)$, i.e. $m = \Theta(2^{2n/3})$.

method,

Comparing exponents, observe that

$$\frac{n}{2} = 0.5n \quad \text{and} \quad \frac{2n}{3} = 0.666\ldots n,$$

hence $\frac{n}{2} < \frac{2n}{3}$ for all $n > 0$. Therefore asymptotically $2^{n/2} \ll 2^{2n/3}$ and so $Q_2 \ll Q_3$.

Finally, for the success probability of producing at least one collision in the two-list case, when $\lambda := \mathbb{E}[C_2] = m^2/2^n$ is small, the distribution of C_2 is well-approximated by a Poisson distribution with mean λ, giving

$$\Pr[C_2 \geq 1] \approx 1 - e^{-\lambda}.$$

In particular, choosing $\lambda = \Theta(1)$ yields a strictly positive constant success probability bounded away from 0 (for $\lambda = 1$, $1 - e^{-1} \approx 0.632$). This completes the proof. □

Theorem 1 shows that choosing a list size of $m = 2^{n/2}$ yields a constant expected number of two-list collisions ($\mathbb{E}[C_2] \approx 1$). In Algorithm 1, this corresponds to generating $2^{n/2}$ decryption queries per list, resulting in an overall oracle-query cost of $Q_2 = 2^{n/2}$.

Each observed collision produces a small number of candidate internal state-difference pairs $(\Delta X, \Delta R)$. Empirically, this number remained nearly constant—typically between two and six across trials—leading to an additional verification cost equivalent to only a few extra oracle calls. Hence, the verification overhead is $O(1)$ relative to the dominant list-generation phase.

Substituting $n = 96$ (the SIMON-JAMBU64/96 configuration) gives an effective total query cost of approximately 4×2^{48} oracle calls, which aligns precisely with the measured data complexity reported in Table 4. Consequently, the overall complexity of Algorithm 1 remains consistent with the asymptotic prediction $Q_2 = \Theta(2^{n/2})$, while offering a practical 33% reduction in query cost compared with the three-tuple baseline ($Q_3 \approx 6 \times 2^{48} = \Theta(2^{2n/3})$).

5.4 Discussion

This study targets JAMBU v1 as used in SIMON-JAMBU64/96. While JAMBU v2.1 introduces minor structural changes (e.g., tag finalization sequence), the core linear update of R and the $\Delta R = $ const relation remain valid. Hence, although our experiments focus on v1, the two-tuple principle may plausibly extend to v2.1 with adjusted parameters, pending empirical validation in future work.

Algorithm 2 Plaintext Forgery Attack

```
Require: A verified pair $(\Delta X, \Delta R)$ from Algorithm 1.
Require: A known valid tuple $(P_1 \parallel P_2, C_1 \parallel C_2)$ obtained using $IV$.
Ensure: A new, valid forged ciphertext-plaintext pair $(C^D, P^D)$.
1: {Construct the forged plaintext}
2: $P^D \leftarrow (P_1 \oplus \Delta X) \parallel (P_2 \oplus \Delta R)$.
3: {Construct the forged ciphertext}
4: $C^D \leftarrow (C_1 \oplus \Delta X) \parallel C_2$.
5: return $(C^D, P^D)$. {This is a valid pair for the nonce $IV'$}
```

Algorithm 3 Plaintext-Recovery Attack

```
Require: A verified pair $(\Delta X, \Delta R)$ from Algorithm 1.
Require: A target ciphertext $C'_{target} = C'_1 \parallel C'_2$ (encrypted using $IV'$).
Require: Access to the decryption oracle $D$.
Ensure: The corresponding plaintext $P'_{target} = P'_1 \parallel P'_2$.
1: {Construct a new ciphertext to query the oracle with}
2: $C_{query} \leftarrow (C'_1 \oplus \Delta X) \parallel C'_2$.
3: {Query the oracle using the *first* IV}
4: $P_1 \parallel P_2 \leftarrow D(IV, AD, C_{query})$.
5: {Recover the target plaintext blocks by reversing the differences}
6: $P'_1 \leftarrow P_1 \oplus \Delta X$.
7: $P'_2 \leftarrow P_2 \oplus \Delta R$.
8: $P'_{target} \leftarrow P'_1 \parallel P'_2$.
9: return $P'_{target}$.
```

The findings of this research validate the cryptanalysis of Peyrin *et al.* on SIMON-JAMBU64/96 under the nonce-respecting scenario, confirming vulnerabilities to distinguishing, plaintext-forgery, and plaintext-recovery attacks. The extension demonstrates that these attacks can be performed with only two tuples (instead of three), providing a crucial new insight—showing that JAMBU's security is weaker than initially claimed.

The trade-off appears in the third phase of the two-tuple attack, which demands greater computational effort to verify multiple candidate differences. In the worst case, this version may be more complex than the original three-tuple attack; however, on average, it offers a more efficient path to compromise the scheme.

The successful application of this attack against SIMON-JAMBU64/96 indicates that the underlying linear property may represent a fundamental design weakness in JAMBU. However, since our experiments are restricted to the SIMON-based instantiation, we refrain from generalizing this conclusion to other JAMBU variants without further analysis.

Practical Considerations. Our adversary model assumes access to a decryption oracle that reveals whether a candidate ciphertext-tag pair is valid. In many practical deployments, this oracle may be unavailable, so the attack should be interpreted primarily as a theoretical weakness in the JAMBU mode rather than an immediately practical exploit. Nonetheless, the analysis highlights how small differences in IVs propagate deterministically, suggesting that JAMBU's initialization process is fragile with respect to chosen-IV attacks.

6 Conclusion

This research successfully validated the cryptanalysis of Peyrin *et al.* on SIMON-JAMBU64/96 under the nonce-respecting scenario. The results confirm that the scheme remains vulnerable to distinguishing, plaintext-forgery, and plaintext-recovery attacks. Quantitatively, the proposed two-tuple distinguisher achieves a reduced data complexity of approximately 4×2^{48} oracle queries—about one-third lower than the 6×2^{48} required by the three-tuple baseline—while maintaining a comparable success probability. These findings validate both the theoretical predictions of Theorem 1 and their practical realization, reinforcing that the JAMBU design exhibits weaker resistance than previously assumed, particularly under resource-limited adversarial conditions.

CRediT Authorship Contribution Statement

Author One: Conceptualization, Methodology, Validation, Formal Analysis, Resources, Data Curation, Writing – Original Draft Preparation, Writing – Review & Editing, Project Administration, and Funding Acquisition. **Author Two**: Conceptualization, Methodology, Software, Investigation, Resources, Data Curation, and Visualization. **Author Three**: Conceptualization, Methodology, Validation, Formal Analysis, Writing – Review & Editing, and Supervision.

Declaration of Generative AI and AI-assisted technologies

The authors acknowledge the use of generative AI and AI-assisted technologies in the preparation of this manuscript. Specifically, Scholar AI 4o was employed to search for and cite peer-reviewed scientific literature; Gemini 2.5 Pro was utilized for idea organization, refinement of content structure, and exploratory drafting; and Quillbot Pro was used to assist in paraphrasing, grammar enhancement, and improving the fluency of academic writing.

All intellectual contributions, interpretations, and final decisions regarding the content were made by the authors. The use of these technologies complied with institutional, ethical, and publication standards, and all AI-generated content was critically reviewed and edited to ensure accuracy, originality, and scholarly integrity.

Declaration of Competing Interest

The authors declare no competing interests.

Funding and Acknowledgments

This research was supported by the Politeknik Siber dan Sandi Negara (Poltek SSN) through its internal research grant scheme. The authors would like to express their sincere gratitude to the academic and technical staff at Poltek SSN for their continuous support and facilitation throughout the course of this research. The authors also acknowledge the contributions of all individuals and teams whose assistance was instrumental in the successful completion of this study.

Data and Code Availability

All data and source code used in this study are openly available and can be accessed via public repositories.

References

[1] J. Black, "Authenticated Encryption," in *Encyclopedia of Cryptography, Security and Privacy*, S. Jajodia, P. Samarati, and M. Yung, Eds. Cham: Springer Nature Switzerland, 2025, pp. 145–156. DOI: 10.1007/978-3-030-71522-9_548. Available online.

[2] M. Dworkin, *Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality*, NIST Special Publication 800-38C, 2007. DOI: https://doi.org/10.6028/NIST.SP.800-38C.

[3] M. Dworkin, *Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*, NIST Special Publication 800-38D, 2007. DOI: https://doi.org/10.6028/NIST.SP.800-38D.

[4] D. J. Bernstein et al., *The CAESAR competition*, https://competitions.cr.yp.to/caesar.html, 2013.

[5] Hongjun and T. Huang, "JAMBU: A Lightweight Authenticated Encryption Mode," in *Selected Areas in Cryptography – SAC 2014*, ser. Lecture Notes in Computer Science, vol. 8781, Springer, 2014, pp. 423–438. DOI: 10.1007/978-3-319-13051-4_26.

[6] IoT Analytics, *State of IoT Summer 2024: Number of connected IoT devices growing 13% to 18.8 billion globally*, https://iot-analytics.com/number-connected-iot-devices/, 2024.

[7] N. Kumar, *How Many IoT Devices Are There (2025–2030 Data)*, DemandSage, Dec. 2024. https://www.demandsage.com/number-of-iot-devices/, 2024.

[8] Ken Research, *Indonesia IoT Technology Market Outlook to 2030*, https://www.kenresearch.com/industry-reports/indonesia-iot-technology-market, 2023.

[9] Mordor Intelligence, *Indonesia IoT Market Size & Share Analysis (2025–2030)*, https://www.mordorintelligence.com/industry-reports/indonesia-iot-market, 2024.

[10] H. Griffioen and C. Doerr, "Examining Mirai's Battle Over the Internet of Things," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, 2020. DOI: 10.1145/3372297.3417277. Available online.

[11] G. Sripriyanka and A. Mahendran, "Mirai Botnet Attacks on IoT Applications: Challenges and Controls," in *Proceedings of Advances in Communication and Computational Technology*, Springer, 2021. Available online.

[12] A. Sharma, V. Mansotra, and K. Singh, "Detection of Mirai Botnet Attacks on IoT Devices Using Deep Le arning," *Journal of Scientific Research and Technology*, 2023, Full text available at https://www.jsrtjournal.com/index.php/JSRT/article/download/49/54. Available online.

[13] T. b. Alshammari and A. S. Alanazi, "Security threats against the internet of things at home," in *2021 International Conference on Electrical, Communication, and Computer Engineering (ICECCE)*, 2021, pp. 1–5. DOI: 10.1109/ICECCE52056.2021.9514258.

[14] B. Gușiță, A. A. Anton, C. S. Stângaciu, et al., "Securing IoT edge: a survey on lightweight cryptography, anonymous routing and communication protocol enhancements," *International Journal of Information Security*, vol. 24, p. 149, 2025. DOI: 10.1007/s10207-025-01071-7. Available online.

[15] M. Al-Shatari, F. A. Hussin, A. A. Aziz, T. A. E. Eisa, X.-T. Tran, and M. E. E. Dalam, "IoT Edge Device Security: An Efficient Lightweight Authenticated Encryption Scheme Based on LED and PHOTON," *Applied Sciences*, vol. 13, no. 18, p. 10 345, 2023. DOI: 10.3390/app131810345.

[16] S. Kumar, D. Kumar, R. Dangi, G. Choudhary, N. Dragoni, and I. You, "A Review of Lightweight Security and Privacy for Resource-Constrained IoT Devices," *Computers, Materials and Continua*, vol. 78, no. 1, pp. 31–63, 2024. DOI: `https://doi.org/10.32604/cmc.2023.047084`. Available online.

[17] J. S. Yalli, M. H. Hasan, L. T. Jung, and S. M. Al-Selwi, "Authentication schemes for Internet of Things (IoT) networks: A systematic review and security assessment," *Internet of Things*, vol. 30, p. 101 469, 2025. DOI: `https://doi.org/10.1016/j.iot.2024.101469`. Available online.

[18] D. Chakraborty and M. Nandi, "COLM under attack: A cryptanalytic exploration of COLM variants," *Journal of Information Security and Applications*, vol. 89, p. 103 936, 2025. DOI: `https://doi.org/10.1016/j.jisa.2024.103936`. Available online.

[19] T. Peyrin, S. M. Sim, L. Wang, and G. Zhang, "Cryptanalysis of jambu," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9054, pp. 264–281, 2015. DOI: `10.1007/978-3-662-48116-5_13`. Available online.

[20] W. Stallings, *Cryptography and Network Security: Principles and Practice*, English, 8th ed. Pearson, 2022, p. 832, Global Edition.

[21] Hongjun and T. Huang, *The jambu lightweight authentication encryption mode (v2.1)*, 2016. Available online.

[22] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers, "The SIMON and SPECK families of lightweight block ciphers," 2013. Available online.

[23] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers, "The simon and speck lightweight block ciphers," in *Proceedings of the 52nd Annual Design Automation Conference*, ser. DAC '15, San Francisco, California: Association for Computing Machinery, 2015. DOI: `10.1145/2744769.2747946`. Available online.