WBSLT: A Framework for White-Box Encryption Based on Substitution-Linear Transformation Ciphers

Yang Shi, Tianchen Gao, Yimin Li, Jiayao Gao* and Kaifeng Huang
School of Computer Science and Technology, Tongji University, Shanghai, China
{shiyang, gaotianchen, hidayat, gaojiayao, kaifengh}@tongji.edu.cn

***Abstract*—With the development of Internet of Things (IoT) networks, data transmitted via such networks has become extremely valuable. Thus, the Advanced Encryption Standard (AES) is widely used for data encryption in IoT networks. Unfortunately, the unattended deployment scenarios pose a significant threat to IoT devices, where attackers have full access to and control over the cryptographic implementation and its execution environment. Such an environment can be modeled as a white-box threat model. And white-box encryption designs are proposed to protect the encryption scheme (such as AES) in this model. However, prior white-box encryption designs primarily protected a single key-dependent table, enabling white-box and side-channel attacks to recover the key. Based on our observation, fuzzing the boundaries of these tables can make attacks ineffective. Thus, we proposed WBSLT[1], a novel design framework for tabulated white-box implementations of substitution-linear transformation (SLT) ciphers. WBSLT protects key-embedded tables with linear and nonlinear transformations and partially leaves each component's computation to the next component to mitigate single key-dependent table breach. To further defend against differential computation analysis and differential fault analysis, the framework integrates masking, shuffling, and external encoding. Theoretical analysis indicates its immunity to various attacks. Experimental results validate the practicality of WBSLT across multiple computing platforms, showing efficient encryption performance and reasonable memory usage.**

I. Introduction

The past decades have witnessed an enormous increase in the Internet of Things (IoT) networks. As a result, data transmitted via IoT networks has increased simultaneously and become an invaluable resource [1], making its protection a critical concern. As one of the most effective encryption methods, the Advanced Encryption Standard (AES) [2] has been widely adopted as an IoT protocol solution for handling large volumes of valuable data efficiently, which ensures the confidentiality and integrity of IoT data. Typical scenario that utilizes AES for data protection in IoT network [3], [4] includes smart city solutions from Samsara [5], smart home systems from companies like Google [6], Samsung [7] and Apple [8], Teladoc Health's smart healthcare and health monitoring products [9], as well as industrial IoT deployments [10] and connected vehicles [11]. The common underlying communication protocols, including LoRaWAN [12], Zigbee [13], and Bluetooth Low Energy (BLE) [14], all leverage AES as their core encryption mechanism to ensure secure data transmission and storage in data centers.

Unlike Wi-Fi or cellular networks, which are typically equipped with high-performance devices having sufficient computing power to support complex encryption algorithms and frequent key renewal, typical IoT devices are resource-constrained. So, they usually do not support frequent key renewal and rely on pre-shared keys for encryption. A significant vulnerability then arises when IoT devices are deployed in potentially insecure environments where attackers have full control over the device. In such scenarios, attackers can extract encryption keys, thereby compromising all data encrypted with the same key. For example, Butun et al. [15] indicate that an attacker with full access to a device running LoRaWAN v1.1 can extract AES keys due to the explicit exposure of key-related information during the Over-the-Air Activation (OTAA) key distribution process. And Camurati et al. [16] demonstrate that AES keys used in BLE can be extracted using Simple Power Analysis (SPA), exploiting the direct exposure of key material through physical access. In both cases, the key is compromised due to its direct exposure to the attacker with full control. Tournier et al. [17] also note that gateways control the network and handle all data transmission in common IoT topologies. Therefore, preventing key exposure in the gateways is more important. White-box cryptography addresses this issue by transforming cryptographic operations into protected lookup tables, preventing direct exposure of secret keys, thereby enhancing security in these vulnerable IoT ecosystems.

A. Threat Model

The *white-box threat model* was first introduced by Chow et al. [18], [19] to model severe threat scenarios (e.g., unattended IoT deployment scenarios). In this model, attackers completely control the execution environments of encryption algorithms, so they can access or even modify the source code,

*Jiayao Gao is the corresponding author.
[1]WBSLT is available at https://github.com/mmt200088/NDSS2026-WBSLT.git.

Network and Distributed System Security (NDSS) Symposium 2026
23-27 February 2026, San Diego, CA, USA
ISBN 979-8-9919276-8-0
https://dx.doi.org/10.14722/ndss.2026.242492
www.ndss-symposium.org

internal states, and execution behavior, which aligns with the vulnerable IoT scenarios. In terms of terminology, encryption algorithms that are secure in the white-box threat model are called *white-box encryption algorithms* (WBEAs), the implementations of WBEAs are called *white-box implementations*, and the study of how to construct WBEAs is called *white-box cryptography*.

It is worth noting that the security goal of WBEAs is not simply the key-extraction security, or more formally, *unbreakability* [20]. Instead, it is the basic goal that all WBEAs should achieve. However, *unbreakability* alone is insufficient for practical use, as it does not protect against attacks such as code-lifting or preimage attacks. In these attacks, an adversary can either copy the entire program code and use it as a larger effective key in another device or reverse the encryption functionality to obtain the decryption functionality. Therefore, WBEAs should achieve additional security goals for practical meanings in different scenarios. In addition, the security goal for white-box decryption algorithms under the standard white-box threat model focuses primarily on unbreakability [20]. While a fully-privileged white-box attacker can directly decrypt ciphertexts by code-lifting without extracting the secret key, the goal of white-box decryption algorithms is to make this process more difficult by increasing the workload for the attacker.

These additional goals are divided into two branches:

- **Security goals in standard white-box cryptography** strictly follow the original assumptions of the white-box threat model [19]. The most widely used is the one-wayness [20] of encryption, which requires that adversaries with complete access to the implementations of encryption algorithms cannot decrypt ciphertexts. Hardware-binding is another security goal that some recently proposed WBEAs [21] tried to satisfy. It requires that WBEAs can only be executed on the intended devices. Traceability [20] is a specific goal for digital rights management (DRM) systems. It requires that decryption modules in the players of digital contents can be traced if they are illegally redistributed.
- **Security goals in weak white-box cryptography** does not strictly follow the original assumptions of white-box attack contexts in [19] and suppose that there are some restrictions on the adversaries' access privilege, such as the number of queries [22] or accessible storage [23]. For example, space hardness [23], [24] requires that adversaries with partial access to the implementations of encryption/decryption algorithms cannot obtain encryption/decryption functionalities. Incompressibility [20], [25] is similar to space hardness, and it requires that adversaries given implementations of encryption/decryption algorithms cannot find functional equivalent but smaller implementations.

The security goals mentioned above are summarized in Figure 1.

Our study considers unbreakability and one-wayness and excludes other security goals for the following reasons: (1) Incompressibility [25], [20] and space-hardness [23], [24] are defined in weaker assumptions and they should be treated as security goals in the gray-box threat model, which is beyond the scope of this paper. (2) Achieving hardware-binding [21] requires hardware modification, which deviates from the traditional objective of white-box cryptography to provide software-only solutions. (3) Traceability [20] lacks generality since it only captures DRM scenarios. The formal definitions of unbreakability and one-wayness are presented below. Unlike black-box encryption schemes, there is currently no formal security proof for WBEAs. The security of WBEAs is mainly based on analysis of their resistance to existing attacks. A detailed security analysis of WBSLT is presented in Section IV.

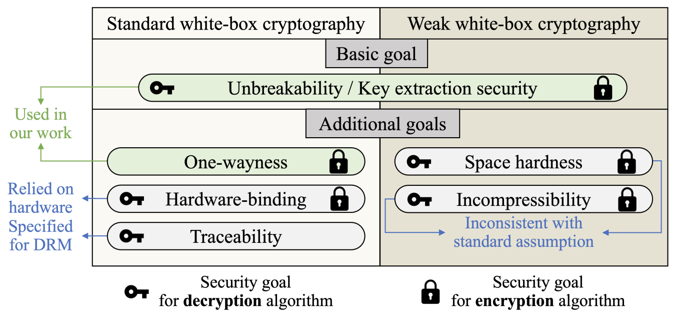

Standard white-box cryptography
Weak white-box cryptography
Basic goal
Unbreakability / Key extraction security
Used in our work
Additional goals
One-wayness
Space hardness
Relied on hardware
Hardware-binding
Incompressibility
Specified for DRM
Traceability
Inconsistent with standard assumption
Security goal for **decryption** algorithm
Security goal for **encryption** algorithm

Fig. 1. Security goals for white-box cryptography

Typically, a white-box encryption scheme is a tuple of four algorithms (KG, Enc, Dec and $WBIG$). KG is the key generation algorithm; Enc is the encryption algorithm; Dec is the decryption algorithm; and $WBIG$ is the white-box implementation generation algorithm. The last algorithm takes a key and a random value as input and outputs a WBEA. Moreover, $\mathcal{R}_{SP}$ is the space of random values for $WBIG$, and $\mathcal{M}_{SP}$ is the message space for Enc.

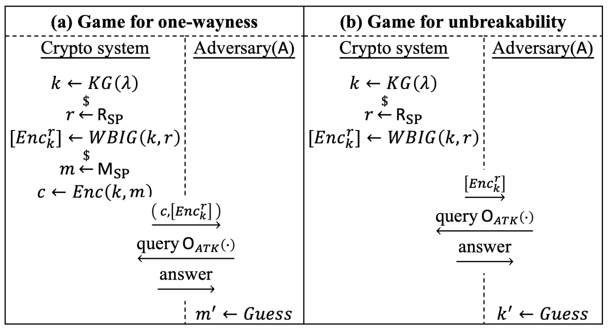

(a) Game for one-wayness
Crypto system
Adversary(A)
$k \leftarrow KG(\lambda)$
$r \xleftarrow{\$} R_{SP}$
$[Enc_k^r] \leftarrow WBIG(k,r)$
$m \xleftarrow{\$} M_{SP}$
$c \leftarrow Enc(k,m)$
$(c,[Enc_k^r])$
query $O_{ATK}(\cdot)$
answer
$m' \leftarrow Guess$
(b) Game for unbreakability
Crypto system
Adversary(A)
$k \leftarrow KG(\lambda)$
$r \xleftarrow{\$} R_{SP}$
$[Enc_k^r] \leftarrow WBIG(k,r)$
$[Enc_k^r]$
query $O_{ATK}(\cdot)$
answer
$k' \leftarrow Guess$

Fig. 2. Security models for white-box cryptography

A number of prior works [26], [27], [20], [28] dealt with security models for white-box encryption. For conciseness, the security models and the corresponding security definitions in [20] are used in this paper. Two scenarios are investigated here, reflecting two different security objectives. The first objective we explored is to decrypt the ciphertexts, using a given WBEA (in Figure 2(a)), and to extract the secret key from a given WBEA (in Figure 2 (b)). To evaluate these objectives, four types of attacks are considered, collectively referred to as ATK, including chosen plaintext attack (CPA), chosen ciphertext attack (CCA), re-compilation attack (RCA), and a combined CCA + RCA attack. We note that RCA allows the adversary to have oracle access to the white-box implementation generation algorithm $WBIG(k, r')$, while $r' \xleftarrow{\$} \mathcal{R}_{SP}$ is kept secret; here, the symbol $\xleftarrow{\$}$ denotes selecting an element from a set at random. Let $\mathcal{O}_{ATK}(\cdot)$ be the oracle for the adversary, then algorithms $Enc(\cdot)$, $Dec(k, \cdot)$, $WBIG(k, \mathcal{R}_{SP})$ and $\{Dec(k, \cdot), WBIG(k, \mathcal{R}_{SP})\}$ are queried, when CPA, CCA, RCA, and CCA + RCA are carried out, respectively.

Security definitions for one-wayness and unbreakability are given below. The former security is stronger than the latter, because a successful extraction of the secret key implies successful decryption of ciphertexts, while the reverse does not hold.

Definition I.1 (One-Wayness)**.** In the security game shown in Figure 2(a), given $Succ_{\mathcal{A},WBIG}^{OW-ATK} = \Pr[m' = m]$, the scheme is (τ, σ)-secure with respect to $OW - ATK$, if and only if for any adversary $\mathcal{A}$ running in time of at most τ, the probability $Succ_{\mathcal{A},WBIG}^{OW-ATK} \leq \sigma$.

Definition I.2 (Unbreakability)**.** In the security game shown in Figure 2(b), given $Succ_{\mathcal{A},WBIG}^{UBK-ATK} = \Pr[k' = k]$, the scheme is (τ, σ)-secure with respect to $UBK - ATK$, if and only if for any adversary $\mathcal{A}$ running in time of at most τ, the probability $Succ_{\mathcal{A},WBIG}^{UBK-ATK} \leq \sigma$.

Definitions I.1 and I.2 strictly follow the original assumptions of the white-box threat model [18], which assume an adversary can do anything to the given white-box implementation (i.e., $[Enc_k^r]$), at their will. This assumption is used in most security definitions [26], [27], [20], [28], WBEA designs [18], [29], [30], [31], [32], [33], and cryptanalysis [34], [35], [36], [37], [38], [39], [40], [41].

B. Existing White-Box Encryption Algorithms

After Chow et al. [18] presented the first two WBEAs, namely, the white-box AES (WB-AES) [18] and white-box DES (WB-DES) [19], white-box cryptography has attracted great interest from both industry and academia.

In industry, white-box cryptography is reported to be widely used in practical security solutions. For example, the white paper on EMV Payment Tokenization published by the U.S. Payments Forum claimed that white-box cryptography is used to obfuscate tokenized payment keys [42]. The Irdeto ActivaCloak for Media, an SDK for digital rights management, uses white-box cryptography to mitigate content key extraction [43]. The Verimatrix XTD, a security solution for mobile applications, uses white-box cryptography to dissolve key information and protect sensitive data [44].

In academia, numerous WBEAs have been proposed, which are divided into two categories according to the security assumptions: (i) **Standard white-box cryptography**, such as white-box AES [18], [45], [31], [29], [30], [32], [46], white-box SHARK [47], that aim to construct implementations with unbreakability and one-wayness for some standardized block ciphers, and (ii) **Weak white-box cryptography**, such as SPACE [23], SPNbox [24], and WhiteBlock [25], that can achieve provable unbreakability and quantifiable code-lifting security (incompressibility/space hardness) by restricting the amount of the data that attackers can obtain. As weak white-box cryptography poses restrictions on attackers' ability, it only reflects limited real-world scenarios. Therefore, this paper focuses on protecting encryption algorithms in the standard white-box threat model. Unless otherwise stated, WBEAs refer to algorithms constructed in the standard white-box threat model.

SLT ciphers are a category of standard white-box cryptography block ciphers that alternately perform substitutions and linear transformations on the cipher state for several rounds. Unlike SPN ciphers, which use permutation operations (P-boxes) to rearrange bits instead of linear transformations, SLT ciphers focus on linear mappings to achieve diffusion. The formal definition is presented as follows:

Definition I.3 (Substitution-Linear Transformation Cipher)**.** A cipher is called an SLT cipher if it contains $R(\geq 1)$ rounds of bijective function $\mathcal{F}(x_1, x_2, \cdots, x_s)$ on $\mathbb{F}_2^n$, where $x_i \in \mathbb{F}_2^m$ and n is the block length that satisfies $n = sm$. For $1 \leq r \leq R$, the round function $\mathcal{F}^{[r]}(x_1, x_2, \cdots, x_s)$ operates as follows:

(i) *Round key addition*: Add an n-bit round key $k^{[r]}$ =($k_1^{[r]}$, $k_2^{[r]}$, $\cdots$, $k_s^{[r]}$) to the state to compute $y_i = k_i^{[r]} \oplus x_i$ for $1 \leq i \leq s$.

(ii) *Substitution*: Compute $z_i = S_i^{[r]}(y_i)$ for $1 \leq i \leq s$, where the (nonlinear) invertible substitution boxes (S-boxes) $S_1^{[r]}$, $S_2^{[r]}$, $\cdots$, $S_s^{[r]}$ are part of the cipher specification and independent of the key.

(iii) *Linear transformation*: Left-multiplying an $n \times n$ invertible matrix $M^{[r]}$ over $\mathbb{F}_2$ with the vector $z = (z_1, z_2, \cdots, z_s) \in \mathbb{F}_2^n$.

Despite many SLT-based WBEAs being proposed in the last two decades, they have all been demonstrated to lack sufficient unbreakability. We conducted a comprehensive review of white-box cryptography by searching Google Scholar and Semantic Scholar using the keywords "white-box cryptography," "white-box implementation," "white-box cryptanalysis," and "white-box attack" for the period between 2002 and 2025. This search yielded 1330, 607, 41, and 8750 papers, respectively. By analyzing the titles of these papers, we identified 80 papers focused on white-box defensive schemes and 164 papers about attacks on white-box defenses (244 in total). Additionally, a search of papers citing these identified papers led to 2267 distinct citations, but no new papers on white-box defensive schemes or attacks were found. After a careful manual screening on the 244 papers, we identified 11 papers that can produce concrete white-box SLT (WB-SLT) encryption schemes, and they were broken by key-extraction attacks soon after their proposal as shown in Figure 3.

Their drawbacks can be classified into two categories: (i) **Existing designs focused on protecting individual parts that hold fragments of the secret key, but failed to prevent**

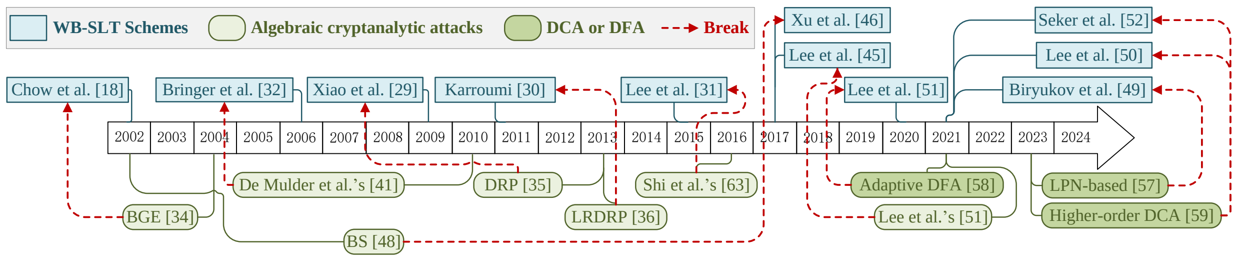


Fig. 3. WB-SLT schemes and related cryptanalysis

attackers from analyzing multiple components together to uncover the broader computation logic. Thus, their internal structure can be identified by algebraic cryptanalytic attacks to progressively remove protections. For example, the BGE attack [34] successfully broke Chow et al.'s WB-AES [18]; the MGH attack [38] targeted the white-box SLT cipher following Chow et al.'s design; the LRDRP attack [36] compromised Karroumi's WB-AES [30] which improved Chow et al.'s design by introducing dual AES specifications; the BS attack [48] was effective against white-box implementations that followed a certain structural pattern, including the aforementioned designs as well as Xu et al.'s WB-AES [46] which introduced dummy rounds within Chow et al.'s WB-AES, and Bringer et al.'s WB-AES [32] which introduced perturbations. (ii) **Existing works [49], [50], [51], [52] require extra memory usage and execution time in order to defend against or slow down differential computation analysis (DCA) [53], and differential fault analysis (DFA) [54] that require little knowledge of the design or internal structure of the implementation.** Nevertheless, these countermeasures have proven ineffective against more advanced versions. Higher-order and adaptive variants of DCA [55], [56], [57], [58], [59] and DFA [58], [60] continue to break these implementations. The aforementioned drawbacks lead to the following WBEA design challenges:

C1: How to obscure the structure of SLT-based components to resist algebraic cryptanalytic attacks, DCA and DFA?

C2: How to maintain the efficiency of the WBEAs when achieving higher security level?

C. Our Design

In this paper, we focus on substitution-linear transformation (SLT) ciphers and propose a novel design framework WBSLT to secure SLT ciphers in the white-box threat model and address the design challenges above.

SLT ciphers, which include AES [2], SERPENT [61], SHARK [62], etc., are a famous category of block ciphers that alternately perform substitution and linear transformation. Our framework WBSLT takes arbitrary SLT ciphers as input and generates table-based implementations that can achieve unbreakability and one-wayness. Due to the inherent symmetry between encryption and decryption processes in SLT ciphers, WBSLT can be adapted to protect decryption keys with a few modifications.

To address **C1**, we observe that algebraic cryptanalytic [63], [34], [36], [48], [38], [35], [41] attacks commonly involve isolating and analyzing groups of interconnected components that collectively realize key-dependent functionalities. Rather than protecting individual components in isolation, which has proven insufficient, WBSLT strategically fuzzes the interactions among neighboring components. This is achieved by partially leaving each component's computation to the next component and protecting each table with randomized transformations. By doing so, WBSLT mitigates the effectiveness of attacks that rely on tracing or isolating distinct transformation layers or encoding patterns. Additionally, lightweight and widely adopted side-channel defenses masking and shuffling [64], [52], [49], and external encoding [18] that recommended by Chow et al., are incorporated to resist DCA and DFA. These measures are integrated into the protected structure and are compatible with the fuzzing strategy.

To address **C2**, we apply masking, shuffling and external encoding strategically and selectively as they introduce performance overhead. We combine the input external encoding with the protection transformations for the first round, and the output external encoding with the protection transformations in the last round to reduce additional operations and memory usage. Furthermore, the masking scheme is integrated with the shuffled output of key-dependent and protection tables, thereby avoiding extra masking operations.

Our key contributions are summarized as follows.

- **Design of WBSLT**: We proposed WBSLT, a novel framework for designing secure white-box implementations of arbitrary SLT ciphers. WBSLT fuzzes the boundaries of components to defend algebraic cryptanalytic attacks, and is compatible with existing countermeasures against side-channel leakage.
- **Security Analysis of WBSLT**. We conducted a comprehensive security analysis on WBSLT's white-box security, i.e., unbreakability and one-wayness. The analysis indicates that WBSLT is resistant to various key-extraction attacks, including algebraic cryptanalytic attacks, DCA and DFA, and prevents attackers from decrypting a ciphertext even if the encryption program is compromised. To the best of our knowledge, no existing WBEAs can achieve unbreakability.
- **A Novel Metric of Structure Security**. We propose a novel security metric, distinguishable SA-structure number (DSASN), to measure fuzziness between rounds. It indicates

the minimum number of layers in a secret-key embedded indivisible SA structure, where S denotes a bijective substitution layer and A denotes a bijective affine transformation layer. However, existing metrics such as white-box diversity (WBD) and white-box ambiguity (WBA) [18] cannot achieve this goal.

- **Application of WBSLT on AES and Optimization**. We present optimization strategies to decrease the storage and improve the efficiency of WB-AES generated by WBSLT. Our optimized WB-AES is expected to replace existing insecure WB-AES schemes.
- **Efficiency Analysis, Evaluation, and Comparison**. Our analysis and evaluation indicate that WBSLT is efficient with reasonable memory usage. The implementations of SLT ciphers with different round numbers and block sizes achieve encryption speeds from 0.01 to 7.90 MB/s across various platforms. Our optimized and general WB-AES-128 implementation achieves encryption speeds from 0.18 to 3.29 and 0.06 to 0.97 MB/s with average power from 1.11 to 3.77 and 1.12 to 3.79 W, and the tables with size 9.78 and 38.37 MB can be generated from 178.32 to 2504.97 and 589.39 to 8012.94 ms consuming 534.12 to 2947.27 and 1972.51 to 9301.01 J energy. Moreover, our optimized and general WB-AES-128 can be integrated into 86.96% and 82.61% of popular IoT gateways running several wireless communication protocols with no efficiency reduction and negligible power increase.

The remainder of this paper is organized as follows: Section II presents our proposed design framework; Section III describes the design of our optimized WB-AES; Section IV analyzes the security and attack resistance of our design; Section VI provides experimental evaluations and comparisons, and our conclusions are given in Section VII.

II. WBSLT: THE DESIGN FRAMEWORK

In this section, we first present two key threats that motivate the design of our framework and then introduce the core strategies in our design. Finally, we summarize our design by illustrating the entire construction and discussing its computational consumption and compatibility with current white-box implementations. The frequently used symbols are listed in Table I.

The core strategy of WBSLT is to fuzz the boundaries of rounds in a fully-tabulated white-box implementation. It can be adapted to a number of white-box implementations with slight modifications, especially the implementation based on an SLT cipher. An example is presented in Section III.

A. Design Rationale

As mentioned in Section I-B, the design of WBSLT is motivated by the following two drawbacks of existing white-box cryptography. The **first drawback** is structural vulnerabilities that exist in current WBEAs. An adversary could still pick up a group of tables, composite them into an object for specialized cryptanalysis, then extract the secret key; and tables corresponding to an encryption round could be easily collected to form a group of tables having a mathematical representation as SAS or SASAS structure, where 'S' stands for a (layer of) bijective substitution and 'A' stands for a bijective affine transformation. The **second drawback** is masking, shuffling and external encoding, which are used to defeat DCA and DFA, requiring extra encryption time and computational resources.

To mitigate the two drawbacks, we transform the cryptographic algorithm into pre-generated look-up tables, secured through linear and nonlinear transformations. To further obscure sensitive variables, we integrate normal and partial adders, making the boundary between (a group of) look-up tables fuzzy to protect the secret key against white-box attacks. Technically, the computational functionality of one component is partially performed by consecutive components. As long as the adversaries are unable to determine the boundary of certain computational functionalities, they can not execute further attacks. This approach is applicable to various concrete white-box implementations of SLT ciphers and is compatible with countermeasures against DCA and DFA. We enhance security by applying masking, shuffling and external encoding to the algorithm. Figure 4 is an overview of our design.

TABLE I
FREQUENTLY USED SYMBOLS

Symbol	Description
n	Block length in bits, $n = sm$.
s	Number of sub-blocks.
m	Sub-block length in bits.
t, t_1, t_2	Number of bits a normal adder outputs. If m is odd, $t_1 = \lceil \frac{m}{2} \rceil$, $t_2 = \lfloor \frac{m}{2} \rfloor$ and t is not considered. If m is even, $t = t_1 = t_2 = \frac{m}{2}$.
q	Number of bits in each input halves of a partial adder that is not summed up.
h_1, h_2	Functions that sum up two q bits and concatenate with bits that are already summed up.
r, R	Round index and the total number of rounds.
$\mathcal{L}_n$	The set of all $n \times n$ invertible binary matries.
$\mathcal{S}_n$	The set of all n-bit permutations.
$M^{[r]}, M_i^{[r]}$	The $n \times n$ linear transformation matrix $M^{[r]}$ in SLT cipher, and the i-th block of the matrix $M^{[r]} = [M_1^{[r]}, M_2^{[r]}, \cdots, M_s^{[r]}]$ that are partitioned by group of columns.
$A^{[r]}, A_i^{[r]}$	The protection matrix $A^{[r]} \in \mathcal{L}_n$, and the i-th block of the matrix $A^{[r]} = [A_1^{[r]}, A_2^{[r]}, \cdots, A_s^{[r]}]$ that are partitioned by group of columns.
$B_i^{[r]}$	The protection matrix $B_i^{[r]} \in \mathcal{L}_m$.
$T_i^{[r]}$	Function that performs a round operations of an SLT cipher protected by linear and nonlinear transformations.
$F_i^{[r]}$	Function serving to enhance protection after $T_i^{[r]}$.
u, v, c	Permutation transformations.
$\mathbf{x}, \mathbf{x_0}, \mathbf{x_1}$	The unmasked value $\mathbf{x}$ and mask shares $\mathbf{x_0}, \mathbf{x_1}$, satisfying $\mathbf{x} = \mathbf{x_0} \oplus \mathbf{x_1}$.
σ	Shuffling transformation.
G_i	External input encoding before the first round.

B. Protected Round Transformations

First, each round of the SLT cipher is protected by linear and nonlinear transformations as well as masking and shuffling, which are implemented as look-up tables.

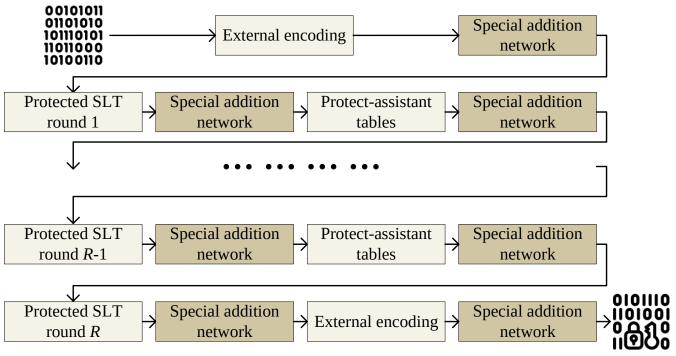


Fig. 4. An overview of the proposed design

Let $t_1 = \lceil \frac{m}{2} \rceil$ and $t_2 = \lfloor \frac{m}{2} \rfloor$. For the r-th round, where $r \in \{1, 2, \cdots, R\}$, and $i, j \in \{1, 2, \cdots, s\}$, let matrices $A^{[r]} \in_\$ \mathcal{L}_n$, $B_i^{[r]} \in_\$ \mathcal{L}_m$, $M_i^{[r]}$ and $A_i^{[r]}$ be the i-th block of the matrix $M^{[r]}$ and $A^{[r]}$ that are partitioned by group of columns, and $B^{[r]} = diag(B_1^{[r]}, \cdots, B_s^{[r]})$, which serve as linear protections. Let permutations $f_{2i-1}^{[r]}, g_{2i-1}^{[r]} \in_\$ \mathcal{S}_{t_1}$, $f_{2i}^{[r]}, g_{2i}^{[r]} \in_\$ \mathcal{S}_{t_2}$, $u_{i,2j-1}^{[r]}, v_{i,2j-1}^{[r]} \in_\$ \mathcal{S}_{t_1}$ and $u_{i,2j}^{[r]}, v_{i,2j}^{[r]} \in_\$ \mathcal{S}_{t_2}$, which serve as nonlinear protections.

The function $T_i^{[r]} : \mathbb{F}_2^m \to \mathbb{F}_2^n$ $(1 \le r \le R)$ is given by

$$\begin{aligned} T_i^{[r]} =& (u_{i,1}^{[r]} || \cdots || u_{i,2s}^{[r]}) \circ \left(((A^{[r]})^{-1} \cdot M^{[r]})_i \cdot \right) \\ & \circ S_i^{[r]} \circ \oplus_{k_i^{[r]}} \circ (B_i^{[r]} \cdot) \circ (f_{2i-1}^{[r]} || f_{2i}^{[r]}), \end{aligned} \tag{1}$$

and the function $F_i^{[r]} : \mathbb{F}_2^m \to \mathbb{F}_2^n$ $(1 \le r < R)$ is given by

$$F_i^{[r]} = (v_{i,1}^{[r]} || \cdots || v_{i,2s}^{[r]}) \circ \left((B^{[r]})^{-1} \cdot A_i^{[r]} \cdot \right) \circ (g_{2i-1}^{[r]} || g_{2i}^{[r]}), \tag{2}$$

where the symbol $||$ stands for working in parallel when it is used between functions.

The output of each function is protected by masking and shuffling. Specifically, for an output $\mathbf{x} = (x_1, x_2, \cdots, x_s) \in (\mathbb{F}_2^m)^s$ produced by $T_i^{[r]}$, before it is protected by the permutation $(u_{i,1}^{[r]} || \cdots || u_{i,2s}^{[r]})$, a random mask $\mathbf{x_0} = (x_{0,1}, x_{0,2}, \cdots, x_{0,s}) \in (\mathbb{F}_2^m)^s$ is generated. A masked output $\mathbf{x_1} = (x_{1,1}, x_{1,2}, \cdots, x_{1,s}) \in (\mathbb{F}_2^m)^s$ is then computed such that $\mathbf{x_1} = \mathbf{x} \oplus \mathbf{x_0}$. The tuple $((x_{0,1}, x_{1,1}), (x_{0,2}, x_{1,2}), \cdots, (x_{0,s}, x_{1,s})) \in (\mathbb{F}_2^m \times \mathbb{F}_2^m)^s$ replaces the original output $(x_0, x_1, \cdots, x_s)$, and the number of permutations is accordingly expanded to protect the masked output. Subsequently, element-wise shuffling $\sigma^{[r]} : \{1, 2, \cdots, s\} \to \{1, 2, \cdots, s\}$ is applied to the masked tuple, and the final output is $((x_{0,\sigma^{[r]}(1)}, x_{1,\sigma^{[r]}(1)}), (x_{0,\sigma^{[r]}(2)}, x_{1,\sigma^{[r]}(2)}), \cdots, (x_{0,\sigma^{[r]}(s)}, x_{1,\sigma^{[r]}(s)}))$. To preserve the correctness of the subsequent addition phase, all $T_i^{[r]}$ functions must be shuffled in the same way. The outputs of the $F_i^{[r]}$ functions undergo a similar masking and shuffling process.

The look-up tables of functions $T_i^{[r]}$ and $F_i^{[r]}$ are implemented by precomputing all possible input-output pairs and storing them in tables $\mathrm{T}_i^{[r]}$ and $\mathrm{F}_i^{[r]}$, which is equivalent to $T_i^{[r]}(x) = \mathrm{T}_i^{[r]}[x]$ and $F_i^{[r]}(x) = \mathrm{F}_i^{[r]}[x]$.

C. Special Addition Enabling Fuzzy Boundary

Second, the addition operations are modified to eliminate the nonlinear protections on the input and add new nonlinear protections on the output. And certain addition operations are partially executed to fuzz the boundaries between each round, thus protecting sensitive variables from being attacked.

Since the output is protected by a permutation in the final step, adding two outputs correctly requires first applying the inverse of their respective permutations. After performing the addition, the result is then protected again using a new permutation. Here we assume $t = t_1 = t_2$ (i.e. m is even) for simplicity, and let $1 \le q \le t$. The output of $T_i^{[r]}$ and $F_i^{[r]}$ can be treated as $4s$ t-bit vectors $\alpha_{i,l,j}^{[r]}$, $i \in \{1, 2, \cdots, s\}, l \in \{1, 2, \cdots, 2s\}$, $j \in \{1, 2\}$. For each l and j, the s outputs are added first, resulting in the masked output of the round transformation. This process is done by $s-1$ normal adders. The normal adder is a mapping from $\mathbb{F}_2^t \times \mathbb{F}_2^t$ to $\mathbb{F}_2^t$ that first applying the inverse of the two inputs' respective permutations and then adds them up, and finally outputs the protected sum. Figure 5(a) illustrates the structure of a normal adder with $t = 4$, where $c_1^{in}, c_2^{in}, c^{out} \in_\$ \mathcal{S}_t$.

Then for each l, the two masked values are added by a partial adder. It is a mapping from $\mathbb{F}_2^t \times \mathbb{F}_2^t$ to $\mathbb{F}_2^{t+q}$ and only $t - q$ bits are added up, which is the key point to fuzz the boundary. Figure 5(b) illustrates the structure of a partial adder with $t = 4, q = 2$, where $c_1^{in}, c_2^{in} \in_\$ \mathcal{S}_t$, $\tilde{c}^{out} \in_\$ \mathcal{S}_{t+q}$.

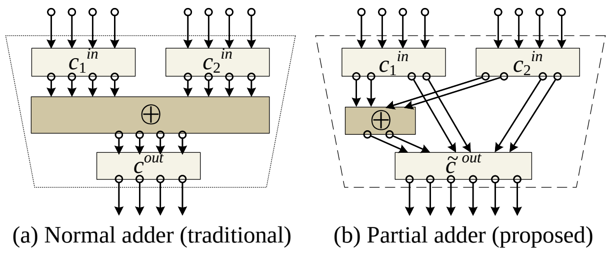


Fig. 5. Structures of adders (t = 4, q = 2)

An addition network consists of $2s-1$ adders (trapeziums), is shown in Figure 6 with $s = 4$. Each pair of directly connected c^{in} and c^{out} is mutually inverse. All adders are implemented as look-up tables.

Since the final output is partially added, the function $T_i^{[r]}$ and $F_i^{[r]}$ are corresponding modified to compelete the addition operations. The function $T_i^{[r]} : \mathbb{F}_2^{t_1+t_2+2q} \to \mathbb{F}_2^n$ is given by

$$\begin{aligned} T_i^{[r]} =& \left(u_{i,1}^{[r]} || \cdots || u_{i,2s}^{[r]} \right) \circ \left(((A^{[r]})^{-1} \cdot M^{[r]})_i \cdot \right) \circ S_i^{[r]} \\ & \circ \oplus_{k_i^{[r]}} \circ (B_i^{[r]} \cdot) \circ \left(h_1 \circ f_{2i-1}^{[r]} || h_2 \circ f_{2i}^{[r]} \right), \end{aligned} \tag{3}$$

and the function $F_i^{[r]} : \mathbb{F}_2^{t_1+t_2+2q} \to \mathbb{F}_2^n$ $(1 \le r < R)$ is given by

$$\begin{aligned} F_i^{[r]} =& (v_{i,1}^{[r]} || \cdots || v_{i,2s}^{[r]}) \circ \left((B^{[r]})^{-1} \cdot A_i^{[r]} \cdot \right) \\ & \circ \left(h_1 \circ g_{2i-1}^{[r]} \parallel h_2 \circ g_{2i}^{[r]} \right), \end{aligned} \tag{4}$$

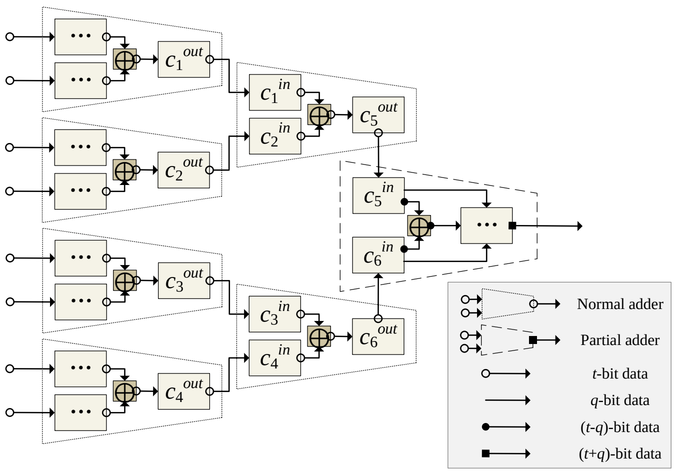


Fig. 6. An addition network consisting of seven adders

where the functions $h_1 : \mathbb{F}_2^{t_1-q} \times \mathbb{F}_2^q \times \mathbb{F}_2^q \rightarrow \mathbb{F}_2^{t_1}$ and $h_2 : \mathbb{F}_2^{t_2-q} \times \mathbb{F}_2^q \times \mathbb{F}_2^q \rightarrow \mathbb{F}_2^{t_2}$ are given by

$$h_1(\alpha_1,\beta,\gamma) = \alpha_1 \,\|\, (\beta \oplus \gamma), \quad h_2(\alpha_2,\beta,\gamma) = \alpha_2 \,\|\, (\beta \oplus \gamma), \tag{5}$$

with α_1 a $(t_1 - q)$-bit value, α_2 a $(t_2 - q)$-bit value, β and γ q-bit values, and the symbol $||$ standing for concatenation, when it is used between binary strings.

D. External Encoding

Third, we apply external encoding to the input and the output of the algorithm, respectively.

The external encoding consists of input and output encoding. The output encoding has two options. The first option uses $2s$ addition networks, each addition network consists of $s-2$ normal adders and one partial adder to process the output. This fuzzes the boundary of the whole encryption and is expected to have high security, resulting in a ciphertext expansion rate of $\frac{t+q}{t}$. The second option uses $2s$ traditional addition networks completely constructed by normal adders. The advantage of the second option is that the ciphertext retains the size of the plaintext (we exclude initial vectors and paddings). Since the input boundary of the last round is still fuzzy, we have not found any existing attacks that can threaten this choice. As for the input encoding, which is used before the first round, it is implemented by using s look-up tables given by G_i ($i \in \{1, 2, \cdots , s\}$), as well as a layer of $2s$ addition networks, each addition network consists of $s-2$ normal adders and one partial adder. Each G_i is a mapping from $\mathbb{F}_2^m$ to $\mathbb{F}_2^n$, as given by

$$G_i =(v_{i,1}^{[0]}|| \cdots ||v_{i,2s}^{[0]}) \circ (B^{[1]} \cdot A_i^{[0]} \cdot) \circ g_i, \tag{6}$$

where $v_{i,2j-1}^{[0]} \in_\$ \mathcal{S}_{t_1}$, $v_{i,2j}^{[0]} \in_\$ \mathcal{S}_{t_2}$ and $g_i \in_\$ \mathcal{S}_m$ for each $j \in \{1, 2, \cdots , s\}$, $A^{[0]} \in_\$ \mathcal{L}_n$ and $A_i^{[0]}$ is the i-th block of the matrix $A^{[0]}$ that is partitioned by group of columns .

E. The Entire Construction and Computational Consumption Analysis

The entire construction involving the first, intermediate, and final rounds is illustrated in Figure 7.

Denote look-up tables corresponding to $T_i^{[r]}, F_i^{[r]}, G_i$, normal adder, partial adder and addition network as T-table, F-table, G-table, NA-table, PA-table and AN-table, respectively. For a tuple of cipher parameters (R, n, m, q), the sizes of T-table, F-table, G-table and AN-table are $R \cdot \frac{n^2}{m} \cdot 2^{m+2q-2}$, $(R-1) \cdot \frac{n^2}{m} \cdot 2^{m+2q-2}$, $\frac{n^2}{m} \cdot 2^{m-2}$, $2^{m-1} \cdot R \cdot \frac{n}{m} \cdot (n - \frac{m}{2} + q)$ bytes, respectively.

The overall storage consumption (in byte) is

$$\frac{2^{m-2}}{m} \cdot \left((2R-1) \cdot n^2 \cdot 2^{2q} + n^2 + n(2n-m+2q)R\right). \tag{7}$$

If the addition networks in the last round consist of completely normal adders, then the overall storage consumption (in byte) can be calculated similarly as

$$\frac{2^{m-2}}{m} \cdot \left((2R-1) \cdot n^2 \cdot 2^{2q} + n^2 + n(2n-m+2q)R - n + q\right). \tag{8}$$

Regarding efficiency, in a single encryption, look-up operations on T-tables, F-tables, G-tables, NA-tables, and PA-tables are executed $R \cdot \frac{n}{m}$, $(R-1) \cdot \frac{n}{m}$, $\frac{n}{m}$, $8R \cdot \frac{n}{m} \cdot (\frac{n}{m} - 1)$, and $4\mathrm{R} \cdot \frac{n}{m}$ times, respectively.

III. Our Optimized WB-AES and a Comparison with Chow et al.'s WB-AES

In this section, we describe the design of our optimized WB-AES. Our optimized WB-AES benefits from the feature of AES, in which the diffusion transformation is two 32-bit to 32-bit operations (i.e., MixColumns and ShiftRows). Thus, our optimized WB-AES should achieve a lower storage footprint of 9.78 MB and increased operational efficiency than directly applying our approach to Chow et al.'s WB-AES (i.e., the general WB-AES) with 38.37 MB storage footprint. Although there is storage overhead compared to traditional AES, our optimized and general WB-AES can be deployed on 20 (86.96%) and 19 (82.61%) out of 23 popular IoT gateways [65], [66], respectively. In a more resource-constrained scenario, the minimum storage footprint is 2.35 MB with q set to 1, which can be deployed on 22 (95.65%) of 23 popular IoT gateways. Although it remains secure, we recommend setting q to 2 for a larger security margin, resulting in unacceptable attack costs.

There are five types of tables used in Chow et al.'s WB-AES, namely type Ia, Ib, II, III, and IV. In our optimized WB-AES, type Ia, Ib, II, and III tables are changed to type Ia', Ib', II' and III', correspondingly. Type IV tables (normal adders) are still used in our WB-AES, and type IV' tables are used to denote partial adders.

The differences between our optimized WB-AES and Chow et al.'s WB-AES are briefed as follows: (i) the normal adder at the end of each addition network is replaced by a partial adder, (ii) in each round, type II and III tables of Chow et al.'s WB-AES are modified to a 12-bit to 128-bit tables, and these tables' mathematical representation is changed to fit two connected partial adders; and (iii) the input of type Ib tables is also changed to 12-bit.

In the description of our optimized WB-AES and the comparison with Chow et al.'s WB-AES, we denote $\mathrm{ID}, \mathrm{OE} \in_\$ \mathcal{S}_b$

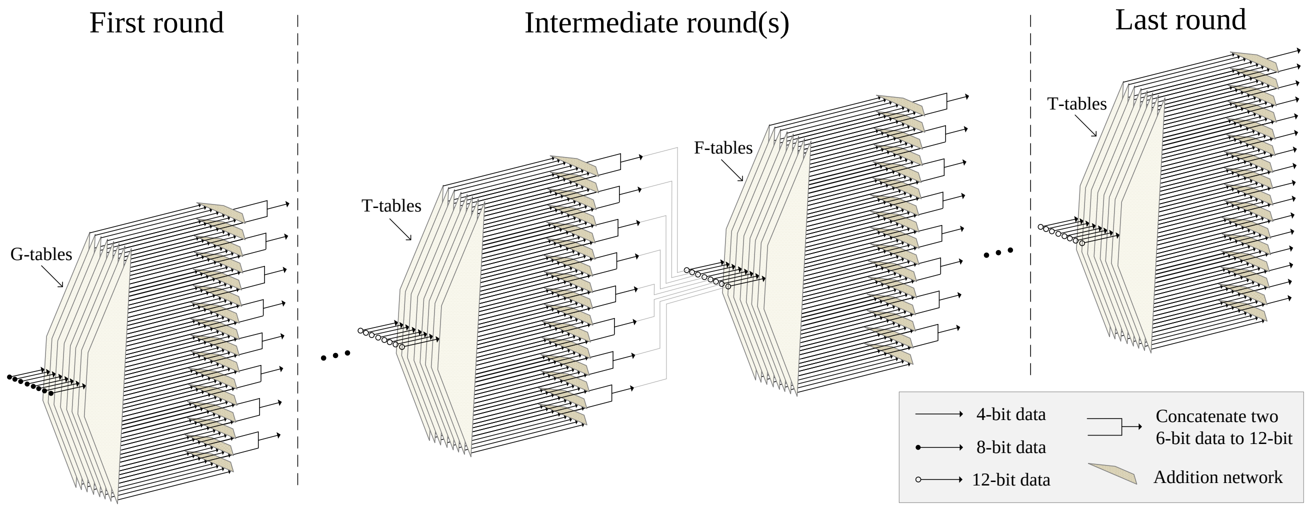


Fig. 7. The first, intermediate, and last rounds of the entire construction (n = 64, m = 8)

the input decoding and output encoding, Mask the masking scheme and $\sigma^{[r]}$ the shuffling technique as described in Section II-B. For $r \in \{1, 2, \cdots, 10\}, i, j \in \{1, 2, 3, 4\}$, denote $L_{i,j}^{[r]} \in_\$ \mathcal{L}_8$, $L_i^{[r]} = diag\{L_{i,1}^{[r]}, L_{i,2}^{[r]}, L_{i,3}^{[r]}, L_{i,4}^{[r]}\}$, $\mathrm{MB}_i^{[r]} = [\mathrm{MB}_{i,1}^{[r]}, \mathrm{MB}_{i,2}^{[r]}, \mathrm{MB}_{i,3}^{[r]}, \mathrm{MB}_{i,4}^{[r]}] \in_\$ \mathcal{L}_{32}$, $k_{i,j}^{[r]}$ be one byte of the round key and $k_{sr(i,j)}^{[r]}$ be one byte of the round key after ShiftRows operation, S be the S-box and $\mathrm{MC} = [\mathrm{MC}_1, \mathrm{MC}_2, \mathrm{MC}_3, \mathrm{MC}_4]$ be the MixColumns matrix.

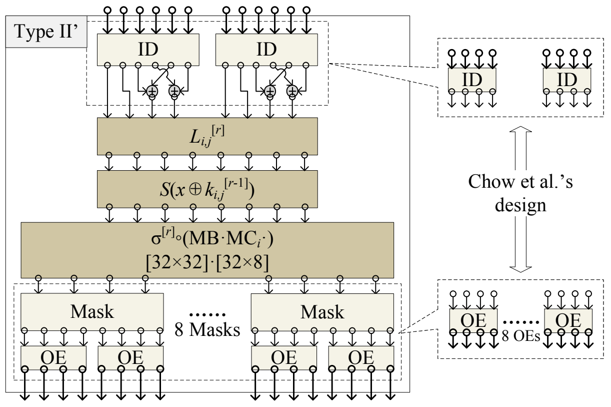


Fig. 8. Our type II' table compared with Chow et al.'s type II table

Type II' tables are the most important in terms of security, because each type II' table implicitly contains an 8-bit segment of the round key. Therefore, we first describe our changes to the structure of type II tables. We set $q = 2$, so that each type II table is modified to a 12-bit to 128-bit table; thus, the table's mathematical representation is changed to fit two connected partial adders. The structure of a type II' table is shown in Figure 8, as well as its comparison with Chow et al.'s type II table.

Type III' tables serve to annihilate inserted mixing bijections of the type II' tables in two consecutive rounds. Similarly, we modified a type III table of Chow et al.'s WB-AES to a 12-

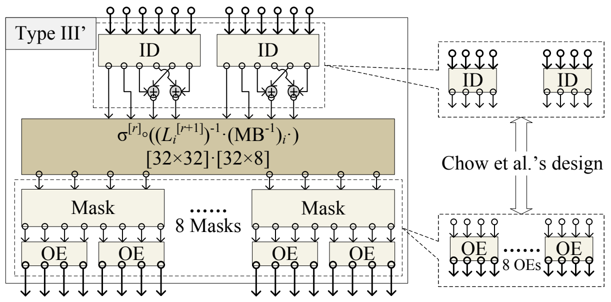


Fig. 9. Our type III' table compared with Chow et al.'s type III table

bit to 128-bit type III' table in our optimized WB-AES; thus, the type III' table's mathematical representation was changed to fit two connected partial adders. The structure of type III' table and a comparison with Chow et al.'s type III table is illustrated in Figure 9.

The structures of a type IV table (normal adder) and a type IV' table (partial adder) are shown in Figure 10.

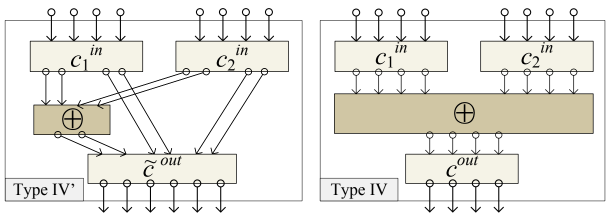


Fig. 10. The structure of type IV and IV' tables

Type Ia' and Ib' tables, as shown in Figures 11 and 12, are used to protect the input and output of the algorithm. Denote $U, V \in_\$ \mathcal{L}_{128}$, U^{-1} is left-multiplied with the input, and it is then left-multiplied by the diagonal matrix that consists of the inverse of $L_{i,j}^{[1]}$ to construct a type Ia' table. Similarly, V is inserted before the output encoding of the last round to construct a type Ib' table.

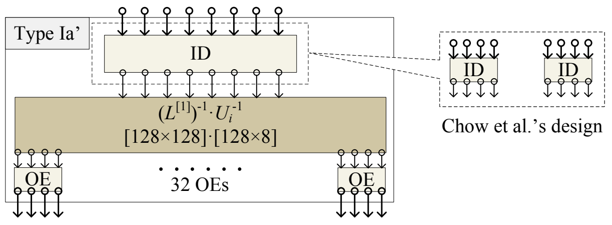


Fig. 11. Our type Ia' table compared with Chow et al.'s type Ia table

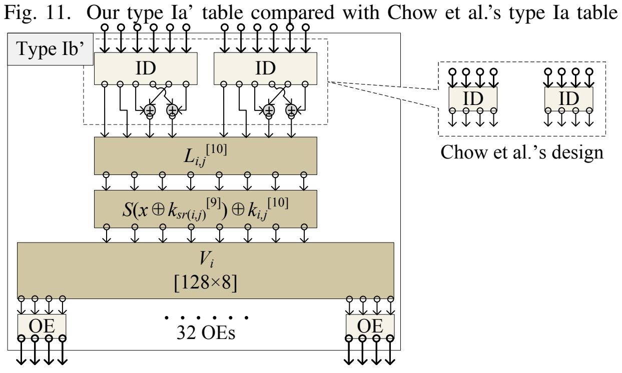


Fig. 12. Our type Ib' table compared with Chow et al.'s type Ib table

IV. Security Analysis and Attack Resistance

This section introduces the new distinguishable SA-structure number (DSASN) metric alongside existing metrics WBD and WBA for evaluating the safety level of WBEAs. We present the metric values for our optimized WB-AES design and compare them with existing schemes, showing improved security. We also analyze resistance against several known attacks, demonstrating strong protection offered by our approach.

A. White-Box Cipher Metrics

In this section, we first introduce a new white-box cipher metric distinguishable SA-structure number (DSASN), and two existing metrics white-box diversity (WBD) and white-box ambiguity (WBA), which measure the safety level of WBEAs. The metrics value of our optimized WB-AES, as well as the comparisons with other WB-AES schemes, are also presented. The comparison results show that our approach is safer than other WB-AES schemes.

1) DSASN: This metric measures the minimum number of layers in the indivisible SA-structure related to the key, where SA-structure begins with one bijective substitution (S) layer, followed by a bijective affine transformation (A) layer, and continues to alternate between S and A layers sequentially. The reason for proposing this metric is that existing metrics are table or component oriented, lacking the measurement for the whole structure of white-box designs. While structural analysis [48] poses a significant threat to white-box designs with DSASN less than or equal to 5.

Since our approach fuzzes the boundaries between every round by partial adders, the adversaries cannot find a small group of look-up tables that form an SA-structure. The only SA-structure they can get is the entire white-box algorithm, whose DSASN value is $5R$, where R is the number of rounds. In contrast, other WB-AES schemes [18], [32], [30], [51], [50] are facing the risk by the structural analysis, because they all have the SA-structure within a single round, which has the DSASN value 5.

TABLE II
Comparison of DSASN of our optimized WB-AES and other WB-AES schemes

WB-AES scheme	ours	[18]	[30]	[51]	[32]	[50]
DSASN value	50	5	5	5	5	5

2) WBD and WBA: WBD measures how many distinct constructions exist for a table of that type, and WBA measures the number of distinct constructions that produce exactly the same table of that type. We provide the following proposition about formulas to compute WBD and WBA of tables in SLT cipher with our design. The type IV table is omitted because it is the same as Chow et al.'s design [18]. We set m even for simplicity, and denote $t = t_1 = t_2$. The number of $n \times m$ full rank matrices over $\mathbb{F}_2$ is denoted by $\Psi(n,m) = 2^n \cdot \prod_{k=0}^{m-1}(1 - 2^{k-n})$.

Proposition 1. For type I, type II, type III and type IV' tables in SLT cipher with our approach, formulas to compute WBD and WBA are shown as the third column in Table III.

Proof.

1) Type I (an instance of G-table)
WBD: There are one m-bit input decoding and $4s$ t-bit output encoding, with an $n \times m$ nonsingular matrix. The number of different encoding is $2^m!$, and $(2^t!)^{4s}$ for decoding. The number of possible matrices is $\Psi(n,m)$. Therefore, the WBD for type I table is $2^m! \times \Psi(n,m) \times (2^t!)^{4s}$.
WBA: Given the input decoding is known, for a certain $n \times m$ matrix that produces the output, there are $m!$ possible matrices that can produce the same output set. And the output encoding are determined by the input decoding and matrix. Therefore the WBA of type I table is at least $2^m! \times m!$.
2) Type II (an instance of T-table)
WBD: There are 2 $(t+q)$-bit input decoding, an $m \times m$ nonsingular matrix, an m-bit round key, followed by an $n \times m$ nonsingular matrix, 4 Mask functions and 16 t-bit output encoding. Therefore the WBD of type II table is $(2^{t+q}!)^2 \times m! \times 2^m \times \Psi(n,m) \times 2^{3n} \times (2^t!)^{4s}$.
WBA: Given the two input decoding are known, for a certain $n \times m$ matrix that produces the output, there are $m!$ possible matrices that can produce the same output. And $4s$ out of $6s$ output encoding can be randomly chosen because of the Mask functions, and the rest of the $2s$ output encoding are determined by the other values. Therefore the WBA of type II table is at least $(2^{t+q}!)^2 \times m! \times 2^m \times \Psi(n,m) \times (2^t!)^{6s}$.
3) Type III (an instance of F-table)
WBD: There are 2 $(t+q)$-bit input decoding, an $n \times m$ matrix, 8 Mask functions and 32 t-bit output encoding. Therefore the WBD of type III table is $(2^{t+q}!)^2 \times \Psi(n,m) \times 2^{3n} \times (2^t!)^{8s}$.
WBA: Given the two input decoding are known, for a

TABLE III

WBD and WBA formulas of SLT cipher with our approach and comparison of our optimized WB-AES and Chow et al.'s WB-AES (N/A stands for not applicable)

Metric	Table	SLT cipher with our approach	Our optimized WB-AES	Chow et al.'s WB-AES
WBD	Type I	$2^m! \times \Psi(n,m) \times (2^t!)^{2s}$	$2^{2923.2}$	$2^{2419.7}$
	Type II	$(2^{t+q}!)^2 \times \Psi(m,m) \times \Psi(n,m) \times 2^{3n+m} \times (2^t!)^{4s}$	$2^{1,272.2}$	$2^{768.7}$
	Type III	$(2^{t+q}!)^2 \times \Psi(n,m) \times 2^{3n} \times (2^t!)^{4s}$	2^{954}	$2^{698.5}$
	Type IV	$(2^t!)^3$	$2^{132.8}$	$2^{132.8}$
	Type IV'	$(2^t!)^2 \times 2^{t+q}!$	$2^{384.5}$	N/A
WBA	Type I	$2^m! \times m!$	$2^{1,049.6}$	$2^{546.1}$
	Type II	$(2^{t+q}!)^2 \times \Psi(m,m) \times 2^m \times \Psi(n,m) \times (2^t!)^{6s}$	$2^{632.25}$	$2^{128.75}$
	Type III	$(2^{t+q}!)^2 \times \Psi(n,m) \times (2^t!)^{6s}$	$2^{620.6}$	$2^{117.1}$
	Type IV	$2^t! \times 2^t$	$2^{48.2}$	$2^{48.2}$
	Type IV'	$2^t! \times \binom{2^t}{2^{t-q}} \times 2^{t-q}!$	$2^{52.84}$	N/A

certain $n \times m$ matrix that produces the output, there are $m!$ possible matrices that can produce the same output set. And $4s$ out of $6s$ output encoding can be randomly chosen because of the Mask functions, and the $2s$ remaining output encoding are determined by the other values. Therefore the WBA of type III table is at least $(2^{t+q}!)^2 \times \Psi(n,m) \times (2^t!)^{6s}$.

4) Type IV' (an instance of PA-table)
WBD: There are 2 t-bit input decoding and 1 $(t+q)$-bit output encoding. Therefore the WBD of type IV' table is $(2^t!)^2 \times 2^{t+q}!$.
WBA: Given one input decoding is known, by enumerating all the XORed bits that are decoded to 0 by the other input decoding, we can uniquely determine the output encoding. Therefore, the WBA of type IV' table is $2^t! \times \binom{2^t}{2^{t-q}} \times 2^{t-q}!$.

This concludes the proof.□

B. Analysis of Algebraic Cryptanalytic Attacks Resistance

In this section, we analyze the resistance of our approach against algebraic cryptanalytic attacks, including the BGE attack [34], LRDRP [36] attack, MGH attack [38], and BS attack [48]. These attacks combine a series of lookup tables, and some of these tables contain portions of secret keys. By performing table-lookup operations on these tables in order, the data transformation can be equivalently represented as a function F. The function is a composition of several (possibly concatenated) constituent functions. Let $f_1, f_2, \ldots, f_n$ be constituent functions such that the composite function is given by $F = f_1 \circ f_2 \circ \cdots \circ f_n$. For any input x, the value of $F(x)$ is given by $F(x) = f_1(f_2(\cdots f_n(x)\cdots))$. And the concatenation of these functions results in a new function $f = (f_1, f_2, \cdots, f_n)$ formed by applying each function to its respective input $f(\mathbf{x}) = (f_1(x_1), f_2(x_2), \ldots, f_n(x_n))$ where $\mathbf{x} = (x_1, x_2, \ldots, x_n)$.

Definition IV.1 (Attack Boundary). Let F be a function composed of a sequence of constituent functions $f_1, f_2, \ldots, f_n$, and let X be its input domain. The attack boundary is defined as the tuple $(X, F(X))$, representing the domain and codomain of F. This boundary represents the interface accessible to the attacker.

Definition IV.2 ($\boldsymbol{P_{\mathrm{ATK}}(X, F(X))}$). Given an attack strategy $\mathrm{ATK} \in \{\mathrm{BGE}, \mathrm{MGH}, \mathrm{LRDRP}, \mathrm{BS}\}$, we define $P_{\mathrm{ATK}}(X, F(X))$ as a logical predicate that must hold over the attack boundary for the attack to succeed.

To evaluate the resistance of our white-box implementation against these attacks, we analyze whether the required properties P_{ATK} are preserved or disrupted by our design. Let F' denote the corresponding function synthesized by WBSLT, with domain X' and codomain $F'(X')$. We compare $P_{\mathrm{ATK}}(X, F(X))$ with $P_{\mathrm{ATK}}(X', F'(X'))$.

Our analysis demonstrates that for each of the aforementioned attacks, the critical properties required for a successful key-recovery attempt do not hold in the case of the white-box implementation generated by WBSLT. In particular, WBSLT disrupts function separability and obscures structural patterns necessary for the reconstruction of key-dependent tables. This effectively prevents the adversary from isolating or recovering any functional boundaries, thereby thwarting the cryptanalytic process.We further empirically validate our analysis. Specifically, we adapted the open-source BGE [67] and BS [68] attacks to target our optimized WB-AES implementation. And we developed the LRDRP and MGH attacks according to the paper and evaluated our resistance since there are no open-source implementations available. Each of the four attacks was conducted on 10,000 independent white-box instances of WBSLT. Experimental results show that all these 40,000 attempts failed, and indicate that WBSLT is resistant to all these attacks.

1) BGE Attack: The BGE attack [34] is the first successful attack against Chow et al.'s WB-AES, which was improved further by Tolhuizen [40] and Lepoint et al. [36] for higher efficiency. The main intermediate structure used in the BGE attack is shown in Figure 13, where the details of bijections $P_{i,j}^r$, $T_{i,j}^r$ and $Q_{i,j}^r$ $(i, j \in \{0,1,2,3\}, r \in \{1,2,\cdots,10\})$ are defined as follows.

$$P_{i,j}^r : \mathbb{F}_2^4 \times \mathbb{F}_2^4 \to \mathbb{F}_2^8, P_{i,j}^r = (L_{i,j}^{r-1})^{-1} \cdot \circ(D_{i,j,0}^r, D_{i,j,1}^r), \tag{9}$$

where $D^r_{i,j,0}, D^r_{i,j,1} : \mathbb{F}_2^4 \rightarrow \mathbb{F}_2^4$ are bijective permutations, $(D^r_{i,j,0}, D^r_{i,j,1})$ denotes the function concatenated by $D^r_{i,j,0}$ and $D^r_{i,j,1}$, and $(L^{r-1}_{i,j})^{-1}$ is the inverse of a random 8×8 nonsingular binary matrix $L^{r-1}_{i,j}$.

$$\begin{aligned} &T^r_{i,j} : \mathbb{F}_2^8 \rightarrow \mathbb{F}_2^8, \\ &T^r_{i,j} = \begin{cases} S \circ \oplus_{k^r_{i,j}}, & 1 \leq r \leq 9, \\ \oplus_{k^{11}_{i,j-i}} \circ S \circ \oplus_{k^{10}_{i,j}}, & r = 10, \end{cases} \end{aligned} \tag{10}$$

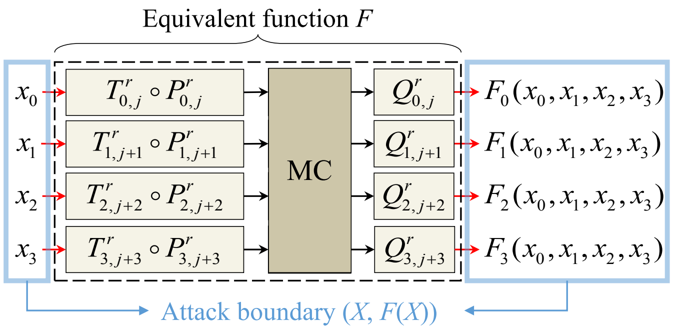
where $k^r_{i,j} \in \mathbb{F}_2^8$ denotes a byte of round key, $\oplus_{k^r_{i,j}}$ denotes bitwise XOR operation with key byte $k^r_{i,j}$, and S denotes the AES SubBytes operation.

$$Q^r_{i,j} : \mathbb{F}_2^8 \rightarrow \mathbb{F}_2^4 \times \mathbb{F}_2^4, Q^r_{i,j} = (E^r_{i,j,0}, E^r_{i,j,1}) \circ L^r_{i,j-i} \cdot, \tag{11}$$

where $E^r_{i,j,0}, E^r_{i,j,1} : \mathbb{F}_2^4 \rightarrow \mathbb{F}_2^4$ are bijective permutations, $(E^r_{i,j,0}, E^r_{i,j,1})$ denotes the function concatenated by $E^r_{i,j,0}$ and $E^r_{i,j,1}$, $L^r_{i,j-i}$ is a random 8×8 nonsingular binary matrix. The equivalent function F required by BGE attack is defined as

$$F : (\mathbb{F}_2^4)^8 \rightarrow (\mathbb{F}_2^4)^8, \quad F = Q^r \circ \mathrm{MC} \cdot \circ T^r \circ P^r, \tag{12}$$

where $Q^r = (Q^r_{0,j}, Q^r_{1,j+1}, Q^r_{2,j+2}, Q^r_{3,j+3})$, $T^r = (T^r_{0,j}, T^r_{1,j+1}, T^r_{2,j+2}, T^r_{3,j+3})$ and $P^r = (P^r_{0,j}, P^r_{1,j+1}, P^r_{2,j+2}, P^r_{3,j+3})$ are concatenated functions, MC is the matrix corresponding to the AES MixColumns operation. And we denote $F_i(x_0, x_1, x_2, x_3)$ as the i-th output element y_i when evaluating $(y_0, y_1, y_2, y_3) = F(x_0, x_1, x_2, x_3)$, where $i \in \{0, 1, 2, 3\}, x_i, y_i \in \mathbb{F}_2^4 \times \mathbb{F}_2^4$.



Equivalent function F
x_0
x_1
x_2
x_3
$T^r_{0,j} \circ P^r_{0,j}$
$T^r_{1,j+1} \circ P^r_{1,j+1}$
$T^r_{2,j+2} \circ P^r_{2,j+2}$
$T^r_{3,j+3} \circ P^r_{3,j+3}$
MC
$Q^r_{0,j}$
$Q^r_{1,j+1}$
$Q^r_{2,j+2}$
$Q^r_{3,j+3}$
$F_0(x_0, x_1, x_2, x_3)$
$F_1(x_0, x_1, x_2, x_3)$
$F_2(x_0, x_1, x_2, x_3)$
$F_3(x_0, x_1, x_2, x_3)$
Attack boundary (X, $F(X)$)

Fig. 13. The crux of the BGE attack and the boundaries required

The attack predicate $P_{\mathrm{BGE}}((\mathbb{F}_2^4)^8, F((\mathbb{F}_2^4)^8))$ required by the BGE attack is

$$\begin{aligned} &\forall x_0, x_1, x_2, x_3 \in \mathbb{F}_2^4 \times \mathbb{F}_2^4, \exists x_0' \in \mathbb{F}_2^4 \times \mathbb{F}_2^4, x_0' \neq x_0, \\ &F_0(x_0, x_1, x_2, x_3) = F_0^{-1}(x_0', x_1, x_2, x_3). \end{aligned} \tag{13}$$

It is worth noting that $F_0^{-1}(x_0', x_1, x_2, x_3)$ is not explicitly available, but can be inferred by exhaustively evaluating F with the first input fixed as $F_0(x_0, x_1, x_2, x_3)$, and then selecting the first element of output whose last three elements match (x_1, x_2, x_3). This is because knowledge of the equivalent function F relies on performing table lookups rather than the underlying transformations, meaning that only forward computation is feasible. The process of inferring F^{-1} from F is illustrated in Figure 14. However, the actual observed behavior reveals an expanded and property-changed attack boundary: instead of matching outputs from different inputs, we observe a persistent mismatch for all alternative inputs relative to a fixed one. This indicates a fundamentally altered relationship and satisfies the modified attack predicate $P_{\mathrm{BGE}}((\mathbb{F}_2^6)^8, F'((\mathbb{F}_2^6)^8))$ which is

$$\begin{aligned} &\exists x_0', x_1, x_2, x_3 \in \mathbb{F}_2^6 \times \mathbb{F}_2^6, \forall x_0 \in \mathbb{F}_2^6 \times \mathbb{F}_2^6, x_0 \neq x_0' \\ &F_0'(x_0', x_1, x_2, x_3) \neq F_0'^{-1}(x_0, x_1, x_2, x_3), \end{aligned} \tag{14}$$

where F' denotes the equivalent function implemented by WBSLT using a modified structure with an addition network over outputs of Type II' tables as illustrated in Figures 6 and 8.

In conclusion, the BGE attack fails against white-box implementations generated by WBSLT, because protection mechanisms in WBSLT eliminate exploitable algebraic relationships, thereby preventing key extraction.

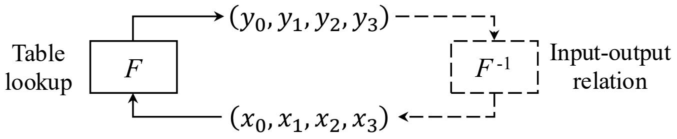


Fig. 14. Deriving the inverse function by forward input-output relations.

2) MGH Attack: The MGH attack [38] was inspired by the BGE attack, and designed to extract the secret key from white-box implementations of a generic class of substitution-permutation network (SPN) block ciphers (i.e., SLT ciphers). The main intermediate structure used in the MGH attack is shown in Figure 15, where the details of bijections f_i^r, $\oplus k_i^r$, S_i^r and f_i^{r+1} ($i \in \{0, 1, \cdots, s\}, r \in R$) are defined as follows.

$$f_i^r : \mathbb{F}_2^{\frac{m}{2}} \times \mathbb{F}_2^{\frac{m}{2}} \rightarrow \mathbb{F}_2^m, f_i^r = (L_i^{r-1})^{-1} \cdot \circ (D^r_{i,0}, D^r_{i,1}), \tag{15}$$

where $D^r_{i,0}, D^r_{i,1} : \mathbb{F}_2^{\frac{m}{2}} \rightarrow \mathbb{F}_2^{\frac{m}{2}}$ are bijective permutations, $(D^r_{i,0}, D^r_{i,1})$ denotes the function concatenated by $D^r_{i,0}$ and $D^r_{i,1}$, $(L_i^{r-1})^{-1}$ is the inverse of a random $m \times m$ nonsingular binary matrix L_i^{r-1}.

$$\oplus_{k_i^r} : \mathbb{F}_2^m \rightarrow \mathbb{F}_2^m, \quad \oplus_{k_i^r}(x) = x \oplus k_i^r, \tag{16}$$

where $k_i^r \in \mathbb{F}_2^m$ denotes a byte of round key.

$$S_i^r : \mathbb{F}_2^m \rightarrow \mathbb{F}_2^m, \quad S_i^r(x) = \mathrm{S}[x], \tag{17}$$

where S denotes a substitution box.

$$f_i^{r+1} : \mathbb{F}_2^m \rightarrow \mathbb{F}_2^{\frac{m}{2}} \times \mathbb{F}_2^{\frac{m}{2}}, f_i^{r+1} = (E^r_{i,0}, E^r_{i,1}) \circ L_i^r \cdot, \tag{18}$$

where $E^r_{i,0}, E^r_{i,1} : \mathbb{F}_2^{\frac{m}{2}} \rightarrow \mathbb{F}_2^{\frac{m}{2}}$ are bijective permutations, $(E^r_{i,0}, E^r_{i,1})$ denotes the function concatenated by $E^r_{i,0}$ and $E^r_{i,1}$, L_i^r is a random $m \times m$ nonsingular binary matrix. The equivalent function F required by MGH attack is defined as

$$F : (\mathbb{F}_2^m)^s \rightarrow (\mathbb{F}_2^m)^s, F = f^{r+1} \circ M^r \cdot \circ S^r \circ \oplus_{k^r} \circ (f^r)^{-1}, \tag{19}$$

where $(f^r)^{-1} = ((f_1^r)^{-1}, (f_2^r)^{-1}, \cdots, (f_s^r)^{-1})$, $\oplus_{k^r} = (\oplus_{k_1^r}, \oplus_{k_2^r}, \cdots, \oplus_{k_s^r})$, $S^r = (S_1^r, S_2^r, \cdots, S_s^r)$ and $f^{r+1} = (f_1^{r+1}, f_2^{r+1}, \cdots, f_s^{r+1})$ are concatenated functions, and M^r is the diffusion matrix. And we denote $F_i(x_0, x_1, x_2, x_3)$ as the i-th output element y_i when evaluating $(y_0, y_1, \cdots, y_s) = F(x_0, x_1, \cdots, x_s)$, where $i \in \{0, 1, \cdots, s\}, x_i, y_i \in \mathbb{F}_2^m$.

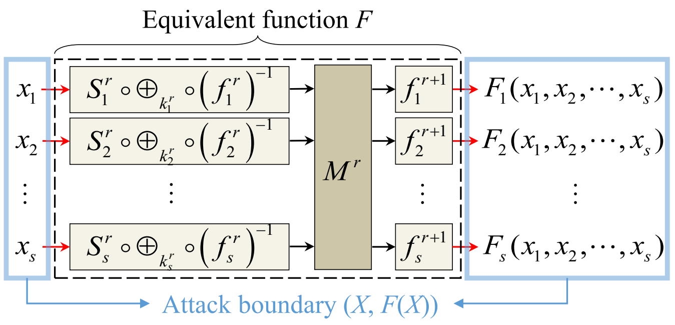

Equivalent function F
x_1
x_2
x_s
$S_1^r \circ \oplus_{k_1^r} \circ (f_1^r)^{-1}$
$S_2^r \circ \oplus_{k_2^r} \circ (f_2^r)^{-1}$
$S_s^r \circ \oplus_{k_s^r} \circ (f_s^r)^{-1}$
M^r
f_1^{r+1}
f_2^{r+1}
f_s^{r+1}
$F_1(x_1, x_2, \cdots, x_s)$
$F_2(x_1, x_2, \cdots, x_s)$
$F_s(x_1, x_2, \cdots, x_s)$
Attack boundary (X, $F(X)$)

Fig. 15. The crux of the MGH attack and the boundaries required

Specifically, the attack predicate $P_{\mathrm{MGH}}((\mathbb{F}_2^m)^s,\, F((\mathbb{F}_2^m)^s))$ required by the MGH attack is

$$\forall c \in (\mathbb{F}_2^m)^{s-1}, \quad \left[\begin{array}{l} \forall x_1, x_2 \in \mathbb{F}_2^m, F_i(x_1, c) = F_i(x_2, c) \implies x_1 = x_2 \\ \wedge \quad \forall y \in \mathbb{F}_2^m, \exists x \in \mathbb{F}_2^m, y = F_i(x, c) \end{array}\right]. \tag{20}$$

However, the actual observed behavior deviates significantly, showing an expanded and property-changed attack boundary. In this case, we observe the existence of outputs that cannot be reached by any input under fixed conditions, violating the expected surjectivity. This altered behavior satisfies the modified attack predicate $P_{\mathrm{MGH}}((\mathbb{F}_2^{m+2q})^s, F'((\mathbb{F}_2^{m+2q})^s))$ which is

$$\forall c \in (\mathbb{F}_2^{m+2q})^{s-1}, \exists y \in \mathbb{F}_2^{m+2q}, \forall x \in \mathbb{F}_2^{m+2q}, F_i'(x, c) \neq y, \tag{21}$$

and F' is the equivalent function implemented by WBSLT using a modified structure with an addition network over outputs of Type II' tables as illustrated in Figures 6 and 8.

In conclusion, the MGH attack fails against white-box implementations generated by WBSLT because no component of the input can be isolated or algebraically reversed.

3) LRDRP Attack: The LRDRP attack [36] is another specialized attack against Chow et al.'s WB-AES. It starts from finding collisions in the output of the coordinate functions to recover fragments of the round keys, forming the crux of the attack in Figure 16. The details of bijections $P_i^{(r,j)}$, $AES^{(r,j)}$ and $Q_i^{(r,j)}$ $(i, j \in \{0, 1, \cdots, s\}, r \in \{1, 2, \cdots, 10\})$ are defined as follows.

$$\begin{aligned} &P_i^{(r,j)} : \mathbb{F}_2^4 \times \mathbb{F}_2^4 \to \mathbb{F}_2^8, \\ &P_i^{(r,j)} = (L_{i,j}^{r-1})^{-1} \cdot \circ (D_{i,j,0}^r, D_{i,j,1}^r), \end{aligned} \tag{22}$$

where $D_{i,j,0}^r, D_{i,j,1}^r : \mathbb{F}_2^4 \to \mathbb{F}_2^4$ are bijective permutations, $(D_{i,j,0}^r, D_{i,j,1}^r)$ denotes the function concatenated by $D_{i,j,0}^r$ and $D_{i,j,1}^r$, $(L_{i,j}^{r-1})^{-1}$ is the inverse of a random 8×8 nonsingular binary matrix $L_{i,j}^{r-1}$.

$$\begin{aligned} &AES^{(r,j)} : \mathbb{F}_{256}^4 \to (\mathbb{F}_2^8)^4, \\ &AES^{(r,j)} = \begin{cases} \mathrm{MC}^{(r,j)} \cdot \circ S \circ \oplus_{k^{(r,j)}}, & 1 \leq r \leq 9, \\ \oplus_{\hat{k}^{(11,j)}} \circ \mathrm{MC}^{(r,j)} \cdot \circ S \circ \oplus_{k^{(10,j)}}, & r = 10, \end{cases} \end{aligned} \tag{23}$$

where $k^{(r,j)} \in (\mathbb{F}_2^8)^4$ denotes four bytes of round keys, $\oplus_{k^{(r,j)}}$ denotes XOR operation with the keys, S denotes the AES SubBytes operation, $\mathrm{MC}^{(r,j)}$ denotes the matrix corresponding to AES MixColumns operation.

$$Q_i^{(r,j)} : \mathbb{F}_2^8 \to \mathbb{F}_2^4 \times \mathbb{F}_2^4, Q_i^{(r,j)} = (E_{i,j,0}^r, E_{i,j,1}^r) \circ L_{i,j-i}^r \cdot, \tag{24}$$

where $E_{i,j,0}^r, E_{i,j,1}^r : \mathbb{F}_2^4 \to \mathbb{F}_2^4$ are bijective permutations, $(E_{i,j,0}^r, E_{i,j,1}^r)$ denotes the function concatenated by $E_{i,j,0}^r$ and $E_{i,j,1}^r$, $L_{i,j-i}^r$ is a random 8×8 nonsingular binary matrix. The equivalent function F required by LRDRP attack is defined as

$$F : (\mathbb{F}_2^4)^8 \to (\mathbb{F}_2^4)^8, F = Q^{(r,j)} \circ AES^{(r,j)} \circ P^{(r,j)}, \tag{25}$$

where $Q^{(r,j)} = (Q_0^{(r,j)}, Q_1^{(r,j)}, Q_2^{(r,j)}, Q_3^{(r,j)})$ and $P^{(r,j)} = (P_0^{(r,j)}, P_1^{(r,j)}, P_2^{(r,j)}, P_3^{(r,j)})$ are concatenated functions. And we denote $F_i(x_0, x_1, x_2, x_3)$ as the i-th output element y_i when evaluating $(y_0, y_1, y_2, y_3) = F(x_0, x_1, x_2, x_3)$, where $i \in \{0, 1, 2, 3\}, x_i, y_i \in \mathbb{F}_2^4 \times \mathbb{F}_2^4$.

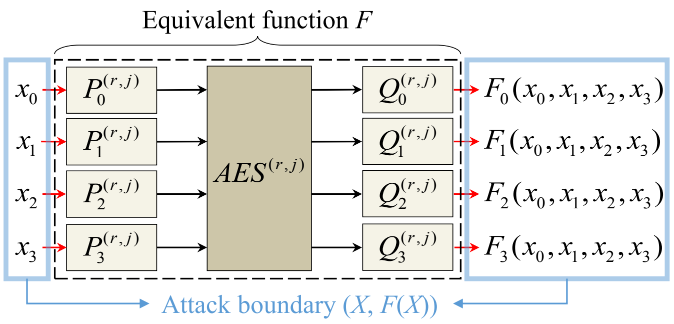


Fig. 16. The crux of the LRDRP attack and the boundaries required

Specifically, the attack predicate $P_{\mathrm{LRDRP}}((\mathbb{F}_2^4)^8,\, F((\mathbb{F}_2^4)^8))$ required by the LRDRP attack is

$$\begin{aligned} &\forall \alpha \in \mathbb{F}_2^4 \times \mathbb{F}_2^4, \exists \beta \in \mathbb{F}_2^4 \times \mathbb{F}_2^4, \\ &F_0(\alpha, 0, 0, 0) = F_0(0, \beta, 0, 0). \end{aligned} \tag{26}$$

In contrast, the observed behavior reveals a fuzzed and expanded attack boundary with fundamentally altered properties. Rather than preserving this symmetry, we find that certain inputs now lead to outputs that cannot be matched by any counterpart, breaking the expected equivalence. This modified behavior satisfies the altered attack predicate $P_{\mathrm{LRDRP}}((\mathbb{F}_2^6)^8, F'((\mathbb{F}_2^6)^8))$ which is

$$\begin{aligned} &\exists \alpha \in \mathbb{F}_2^6 \times \mathbb{F}_2^6, \forall \beta \in \mathbb{F}_2^6 \times \mathbb{F}_2^6, \\ &F_0'(\alpha, 0, 0, 0) \neq F_0'(0, \beta, 0, 0). \end{aligned} \tag{27}$$

In conclusion, WBSLT invalidates the algebraic assumptions underlying LRDRP, preventing attackers from isolating exploitable collisions, and thus effectively neutralizing this attack.

4) BS Attack: The BS attack [48] is a form of structural cryptanalysis targeting substitution-affine transformation networks (SANs). It is capable of recovering an equivalent representation of certain cipher constructions, such as a 2.5-round SAN. Based on the description in [48], a classical 2.5-round SAN is shown in Figure 17. The BS attack is a prominent threat to white-box implementations of SLT ciphers. Regardless of how well the details of a round transformation are hidden in computational components (e.g., look-up tables), if adversaries can identify the boundaries of a round, they can easily acquire an equivalent representation of the round transformation. This enables adversaries to decrypt ciphertext without the secret key. The bijections $S_{i,j}$ and A_h $(i \in \{0,1,2\}, j \in \{0,1,\cdots,s\}, h \in \{0,1\})$ are defined as follows.

$$S_{i,j} : \mathbb{F}_2^m \to \mathbb{F}_2^m, \quad S_{i,j}(x) = \mathrm{S_{i,j}}[x], \tag{28}$$

where $\mathrm{S_{i,j}}$ denotes an invertible S-box.

$$A_h : \mathbb{F}_2^n \to \mathbb{F}_2^n, \quad A_h = \oplus_{B_h} \circ L_h \cdot, \tag{29}$$

where L_h is a random nonsibgular $n \times n$ binary matrix, $B_h \in \mathbb{F}_2^n$ is a random value. And the equivalent function F required by BS attack is defined as

$$F : (\mathbb{F}_2^m)^s \to (\mathbb{F}_2^m)^s, \quad F = S_2 \circ A_1 \circ S_1 \circ A_0 \circ S_0. \tag{30}$$

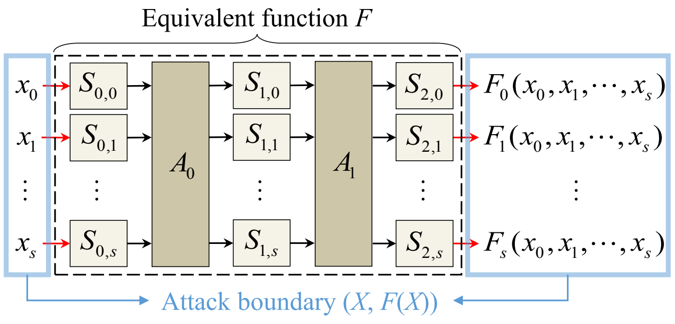


Fig. 17. The crux of the BS attack and the boundaries required

Specifically, the attack predicate $P_{\mathrm{BS}}((\mathbb{F}_2^m)^s, F((\mathbb{F}_2^m)^s))$ required by the BS attack is

$$\begin{aligned}(\forall x_1, x_2 \in (\mathbb{F}_2^m)^s, F(x_1) = F(x_2) \implies x_1 = x_2) \\ \wedge (\forall y \in F((\mathbb{F}_2^m)^s), \exists x \in (\mathbb{F}_2^m)^s, y = F(x)).\end{aligned} \tag{31}$$

However, the observed behavior reveals a significantly expanded and property-changed attack boundary. Rather than preserving bijectivity, the function now exhibits collisions: two distinct inputs map to the same output, violating the injective requirement. This altered behavior satisfies the modified attack predicate $P_{\mathrm{BS}}((\mathbb{F}_2^{m+2q})^s, F'((\mathbb{F}_2^{m+2q})^s))$ which is

$$\exists x_1, x_2 \in (\mathbb{F}_2^{m+2q})^s, x_1 \neq x_2, F'(x_1) = F'(x_2). \tag{32}$$

In conclusion, WBSLT obstructs an adversary's ability to uniquely characterize the affine or substitutional structure, thereby thwarting the BS attack.

C. Analysis of DCA and DFA Resistance

This section illustrates the resistance of our approach against DCA and DFA. The analysis and experiment demonstrate that these attacks cannot break the white-box implementation based on WBSLT.

1) Differential Computation Analysis: DCA [53] is the software counterpart of the differential power analysis, which can help adversaries extract the secret key embedded in the look-up tables without much knowledge of the detailed construction. Adversaries collect a bunch of *traces* that are produced while the program is running, and guess all possible keys until a guessed key gains a high enough correlation score with the collected *traces*. This key will be selected as the correct key.

Specifically, denote $I(p_i, k_i^*)$ as an intermediate state of the program with input p_i and the corresponding secret key k_i^*, and $\mathbb{L}(I(p_i, k_i^*)) + \delta$ as the leaked information (e.g. the value of a certain bit) with noise δ when program runs into state $I(p_i, k_i^*)$. The adversary first collects t *traces* for n inputs $(p_1, p_2, \cdots, p_n)$

$$\mathbf{v}_i = (v_1, v_2, \cdots, v_t), i \in \{1, 2, \cdots, n\}, \tag{33}$$

and then uses a distinguish function $\mathbb{D}$ to calculate the correlation score γ_{k_j} of the j-th guessed key k_j

$$\gamma_{k_j} = \mathbb{D}((\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_n); (p_1, p_2, \cdots, p_n)). \tag{34}$$

If the guessed key is correct, then it will get a relatively high correlation score, or vise versa.

There are several improvements to the original DCA [53]. Bogdanov et al. proposed the higher-order DCA (HO-DCA) [69], breaking linear masking without external encoding. And Goubin et al. proposed data-dependency HO-DCA (DDHO-DCA) [56], breaking the linear and nonlinear masking and avoiding the exponential explosion of the window size at the same time. Moreover, Tang et al. further improved DDHO-DCA (IDDHO-DCA) [59] that can break the dummy shuffling and SEL masking without external encoding.

TABLE IV
SUCCESS RATE OF THE IDDHO-DCA [59] ATTACK TARGETING A BYTE OF THE ROUND KEY IN WB-AES, WITH AND WITHOUT WBSLT

Scheme	Success Rate
WB-AES without WBSLT	1
WB-AES with WBSLT	$\frac{103}{25600}$ = 0.004023 (≈ $\frac{1}{256}$, close to random guessing)

We conduct the latest IDDHO-DCA [59] on Chow et al.'s WB-AES with and without WBSLT, repeating 25600 times for each case, to evaluate the resistance of WBSLT to DCA. In the absence of WBSLT, the actual key consistently ranks first in every trial, yielding an average rank of 1. Consequently, the success rate of the IDDHO-DCA attack in this scenario is 100%, as depicted in Table IV. In contrast, when WBSLT is applied, the rank of the actual key becomes approximately uniformly distributed, with an observed average rank of 128.13—close to the theoretical median of 128 for a uniform distribution over 256 candidates. Under this condition, the success rate of the IDDHO-DCA attack drops significantly to just 0.40% as shown in Table IV, which is statistically close to random guessing (i.e., $\frac{1}{256}$). Since complete recovery of the AES key requires correctly identifying all 16 bytes of a round key, the probability of a successful attack Pr_{succ} is only

$$Pr_{succ} = (\frac{1}{256})^{16} = 2^{-128}. \tag{35}$$

In summary, our approach is able to resist the DCA [53], [69], [56], [59]. Our approach can resist DCA for two reasons.

First, we use external encoding to protect the input before the first round and output after the final round, while DCA is infeasible on implementations with external encoding. Second, the protected tabulated implementation adds extra randomness to the intermediate state of the program, which adds further protection against DCA.

2) Differential Fault Analysis: Since the use of external encoding can effectively protect secret keys from being extracted by DCA, Amadori et al. proposed the DFA [54] to break white-box implementations with bijective external encoding. This attack assumes that the external encoding E is defined as

$$E : (\mathbb{F}_2^8)^{16} \to (\mathbb{F}_2^8)^{16}, E(x) \mapsto (E_0(x_0)||E_1(x_1)|| \cdots ||E_{15}(x_{15})), \tag{36}$$

where $E_i : \mathbb{F}_2^8 \to \mathbb{F}_2^8 (i = 0, 1, \cdots, 15)$ are 8-bit bijective encoding. It uses a technique similar to BGE attack [34] to remove the external encoding, and the standard DFA to extract secret keys. The original DFA was improved by Amadori et al. themselves in [60], in which the attack was simplified and more efficient using heuristic analysis. Tang et al. proposed an adaptive version in [58] that introduced a new adaptive DFA model which assumes that an adversary can adaptively collect the intermediate values of a specific function and launch DFA with chosen inputs.

However, DFA only applies to 8-bit encodings, while our design uses a 128-bit external encoding $G : \mathbb{F}_2^{128} \to \mathbb{F}_2^{128}$. We conduct the DFA [54] on our approach using Dark-Phoenix [70]. After 1,000 attack attempts, not a single byte of the secret key was recovered, thereby supporting the conclusion that our approach is resistant to DFA.

D. Analysis of One-Wayness

In this section we analyze why WBSLT satisfies the one-wayness property, that is, why it is infeasible to get an equivalent decryption algorithm from the white-box encryption implementation.

From a brute-force perspective, recovering an equivalent decryption algorithm would require reconstructing both the external encoding and each protected round transformation. For a typical SLT cipher with input size n and R rounds, recovering the input-output relationship of a single round transformation or external encoding would require evaluating 2^n input-output pairs. So the overall effort to recover all rounds is $R \times 2^n$, which is infeasible for typical ciphers.

From a cryptanalysis perspective, all the attacks discussed in Section IV-B involve a step to get equivalent representations of the protected lookup tables. However, WBSLT fundamentally obstructs this process in each case, rendering these attacks ineffective. Nevertheless, a potential concern arises for the AES case: due to the small domain of certain intermediate computations, it is feasible to collect all input-output pairs of one round using 4×2^{48} queries. This raises the question of whether the full cipher can be recovered by assembling these round-by-round equivalents. However, the protection introduced by WBSLT does not stop at round transformations. The external encoding layer remains a robust barrier. Recovering its equivalent representation requires observing the full input-output behavior of the cipher. For AES, this entails 2^{128} input-output pairs. Moreover, the best known algorithm [71] for recovering a linear equivalent of the external encoding has time complexity $128^3 \times 2^{128}$, which is beyond practical feasibility.

In conclusion, it is infeasible for an attacker to get an equivalent decryption algorithm from the white-box implementation, neither by analyzing the entire input and output nor round by round.

V. Analytic Comparison with Related Studies

In this section, we compare some well-known WB-AES schemes with our general and optimized WB-AES schemes.

As shown in Table V, we compare our general and optimized WB-AES schemes with Chow et al.'s [18], Bringer et al.'s [32], Xiao et al.'s [29], Karroumi's [30], Bai et al.'s [72], as well as three Lee et al.'s WB-AES schemes based on re-encoding [31], masking [51], and table redundancy [50] techniques, which are denoted as Lee et al.'s (Re-encoding), Lee et al.'s (Mask) and Lee et al.'s (Table), respectively. The data presented in the second and third columns (storage and efficiency) are based on AES-128. Note that our two WB-AES schemes both use normal adders in the final layer of the addition network. The column that lists the decryption efficiency only considers the operations for eliminating external encoding (i.e., it does not include the standard AES decryption). The existing WB-AES schemes' efficiencies are similar to or worse than our proposed WB-AES schemes. Although the storage consumption of our approach is relatively large, they are still affordable in most application scenarios.

Table V clearly shows that existing WB-AES schemes[18], [29], [30], [31], [32], [72], [51], [50] are not sufficiently secure. Most of them are even classified as insecure under the weaker security definition. Only our general and optimized WB-AES schemes are secure under the stronger security definition.

VI. Experiments

In this section, we conduct experiments to answer the following three research questions (RQs):

- **RQ1**: How do different SLT configurations, as well as the SHARK and PRESENT cipher specifications, impact the encryption performance of WBSLT?
- **RQ2**: How do our optimized and general WB-AES compare with other WB-AES schemes in terms of execution and energy efficiency during table generation and encryption, across different platforms?
- **RQ3**: How do our optimized and general WB-AES affect execution and energy efficiency when implemented with wireless communication protocols, compared with traditional AES?

A. Experiment Setup

The performance evaluations of SLT and WB-AES were carried out on four high-performance platforms and four resource-limited platforms as listed in Table VI. Protocol

TABLE V
COMPARISON OF STORAGE, EFFICIENCY, AND SECURITY OF WB-AES SCHEMES[1]

WB-AES	**Storage (Calculated)**		**Efficiency (Calculated)**		**Security**	
	Encryption (MB)	Decryption (KB)	Encryption	Decryption	UBK	OW
Chow et al.'s [18]	0.73	5	$3104L$	$64L + 2^{16}B$	No	No
Bringer et al.'s [32]	568	0	N/A	0	No	No
Xiao et al.'s [29]	20.02	4	$80L + 11 \cdot 2^{21}B$	$2^{16}B$	No	No
Karroumi's [30]	0.73	5	$3104L$	$64L + 2^{16}B$	No	No
Lee et al.'s [31] (Re-encoding)	1.16	5	$3104L$	$64L + 2^{16}B$	No	No
Lee et al.'s [51] (Mask)	17.54	5	$2512L$	$64L + 2^{16}B$	No	No
Lee et al.'s [50] (Table)	0.99	5	$3024L$	$64L + 2^{16}B$	No	No
Our general WB-AES	38.37	8.75	$20160L$	$48L + 2^{16}B$	Yes	Yes
Our optimized WB-AES	9.78	8.75	$5280L$	$48L + 2^{16}B$	Yes	Yes

[1] **Storage**: calculated storage of static data, such as look-up tables and matrices; **B/L**: bitwise/table lookup operation; **OW**: the stronger security definition one-wayness (Definition I.1; **UBK**: the weaker security definition unbreakability (Definition I.2.

TABLE VI
SPECIFICATIONS OF EVALUATION PLATFORMS

Category	**Device**	**Processor**
high-performance	PC 1	Intel® Core™ i7-11800H@2.3GHz
	PC 2	AMD Ryzen 7 5700X3D@3.0GHz
	Cloud VM 1	2vCPU: Intel Ice Lake@2.70GHz
	Cloud VM 2	4vCPU: AMD EPYC Bergamo@3.10GHz
resource-limited	UP Nezha	Intel Alder Lake-N@2.0GHz
	Raspberry Pi 5	ARM Cortex-A76@2.4GHz
	Smartphone 1	Qualcomm Snapdragon 8 Elite@4.47GHz
	Smartphone 2	Qualcomm Snapdragon 778G@2.40GHz

performance assessments were performed on two Raspberry Pi units, with LoRaWAN and Zigbee supported by external modules, and BLE enabled through the onboard chipsets. The performance results are the average of 1,000 executions. The development of our table generation and encryption programs benefits from WBMatrix library by Tang et al. [73]. The compared WB-AES algorithms are based on Tang et al.'s implementations [74]. All programs and state-of-the-art algorithms are developed using the C programming language.

B. SLT Encryption Performance Evaluation

To evaluate the encryption performance of WBSLT with different SLT configurations, we generated tabulated implementations, corresponding to 6-to-32-round 64-bit and 128-bit SLT ciphers, with the parameter m sets to 8 and q sets to 2 to be compatible with most currently used block ciphers [2], [75], [76], [77]. Many well-known block ciphers, such as the SHARK (64-bit, 6-round), KHAZAD (64-bit, 7-round), AES-128 (128-bit, 10-round), SERPENT (128-bit, 32-round), and PRESENT (64-bit, 30-round), fall into this range. We also generated and evaluated WBSLT-based SHARK and PRESENT implementations.

The results of encryption efficiency are presented in Figure 18. The results show that the encryption efficiency changes almost linearly with block size and rounds. Additionally, the performance of WBSLT when applied to SHARK and PRESENT is consistent with our general evaluation of SLT configurations, suggesting that the white-box implementations generated by our design framework work reasonably well on various platforms.

Answer to RQ1: The encryption efficiency of WBSLT on SLT ciphers shows linear scalability with block size and rounds across various cipher configurations, as well as the SHARK and PRESENT cipher specifications.

C. Our Optimized and General WB-AES Execution Efficiency and Energy Consumption Evaluation and Comparison

Our optimized and general WB-AES, corresponding to AES-128 that consists of 16 T-tables and 16 F-tables in each round, were generated. Our optimized WB-AES follows Chow et al.'s design, and benefits from the 32-bit MixColumns and ShiftRows operations.

In terms of execution efficiency of our optimized and general WB-AES, the table generation time varies from 178.32-2504.97 and 589.39-8012.94 ms as shown in Figure 19 (a), and the encryption speed varies from 0.18-3.29 and 0.06-0.97 MB/s as shown in Figure 19 (c), respectively. In terms of energy efficiency of our optimized and general WB-AES, the table generation consumes 534.12-2947.27 and 1972.51-9301.01 J as shown in Figure 19 (b), and the encryption power varies from 1.11-3.77 and 1.12-3.79 W as shown in Figure 19 (d), respectively.

The performance of Chow et al.'s WB-AES [18] (expected to have the same performance as Karroumi's [30] and Lee et al.'s [31]), Xiao et al.'s WB-AES [29] and Lee et al.'s two WB-AES schemes [51], [50] have also been evaluated on various platforms for the efficiency comparison, despite these white-box implementations being insecure.

The comparison results in Figure 19 show that the encryption and table generation performance of our optimized WB-AES is much better than that of Xiao et al.'s WB-AES and Lee et al's WB-AES (Mask), and is similar to that of Chow et al.'s WB-AES and Lee et al's WB-AES (Table), while our general WB-AES presents slight performance overhead. Moreover, our optimized WB-AES is approximately three times more efficient than our general WB-AES. This improvement can be attributed to the smaller table size and fewer operations required by our optimized WB-AES. The experimental results

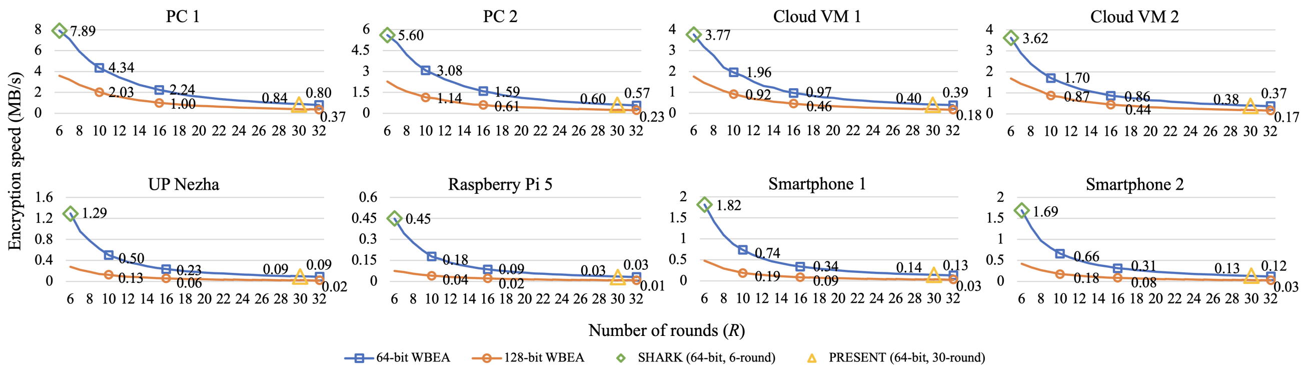


Fig. 18. Our SLT's encryption performance under various configurations

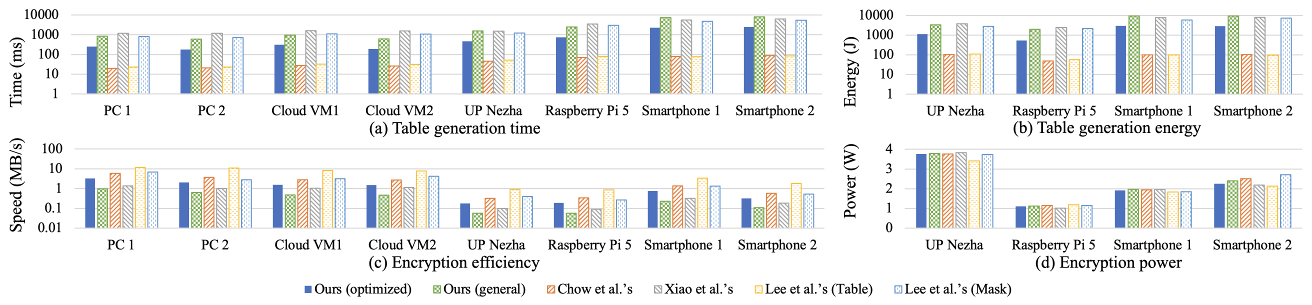


Fig. 19. The performance evaluation of different WB-AES schemes on various platforms

are consistent with the analytical comparison presented in Section V.

Answer to RQ2: Our optimized WB-AES performs reasonably well. It outperforms Xiao et al.'s and Lee et al.'s (Mask) WB-AES schemes in both encryption speed and table generation time and energy, and have similar performance compared with existing high-performance WB-AES schemes, while our general WB-AES presents a little performance overhead. Additionally, our optimized WB-AES is approximately three times more efficient than our general WB-AES, indicating the effectiveness and necessity of our optimization.

D. Real-World Applicability Evaluation

To evaluate real-world applicability, we integrated our optimized WB-AES into three widely used wireless communication protocols: BLE, LoRaWAN, and Zigbee. Each protocol was tested using the traditional AES, our optimized WB-AES, and our general WB-AES under the same hardware and environmental conditions listed in Table VII.

TABLE VII
SPECIFICATIONS OF PROTOCOLS AND CHIPSETS

Protocol	Chipset
BLE 5.0	Cypress CYW43455
BLE 5.3	Infineon CYW55572
LoRaWAN	Semtech SX1262
Zigbee 3.0	Silicon Labs EFR32MG21

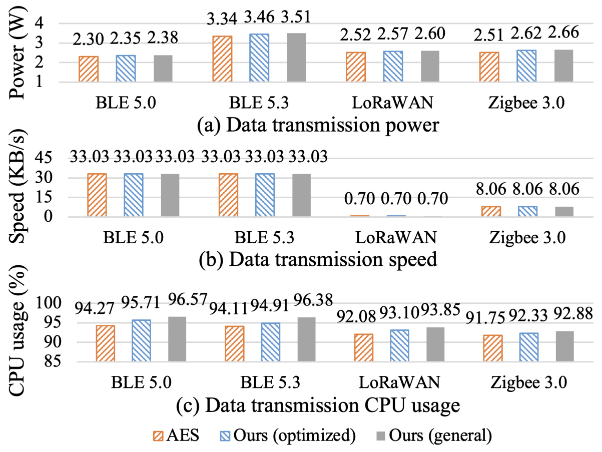


Fig. 20. The wireless protocol communication performance evaluation of using AES, our optimized WB-AES, and our general WB-AES

As shown in Figure 20, the integration of our optimized and general WB-AES resulted in an approximately 2.6%–4.0% and 4.0%-5.5% increase in average power consumption, and 0.6%-1.1% and 1.2%-2.4% increase in average CPU usage across all the protocols, respectively, while data transmission speed remained unchanged. When integrated into wireless communication protocols, our optimized WB-AES is up to 1.45% more efficient in data transmission power and 1.55% more efficient in CPU usage compared with our general WB-AES, while the data transmission speed is the same. This behavior is attributed to the fact that AES-based encryption occupies only a small portion of the total transmission time, with the majority consumed by protocol-specific operations. Therefore, encryption is not a critical performance bottleneck. **Answer to RQ3: Our optimized and general WB-AES introduce a negligible impact on data transmission speed, power efficiency and CPU usage. Moreover, our optimized WB-AES is slightly more efficient than the general version in computation efficiency and requires less storage than our general WB-AES. These results indicate that both our optimized and general WB-AES are suitable for secure integration into wireless protocols such as BLE, LoRaWAN, and Zigbee, especially when robust security is a priority.**

VII. Conclusions

Based on the analysis of common vulnerabilities in existing WBEAs, we proposed a new design framework for tabulated white-box implementations. The framework fuzzes the boundaries of computational functionalities in look-up tables corresponding to an arbitrary SLT cipher, thus deterring an adversary from analyzing certain groups of tables required by most cryptanalysis of WBEAs. Security analysis indicates that our proposed approach is resistant to known attacks, while computational consumption analysis shows that its efficiency and memory footprint are reasonable. Experimental results demonstrate that these white-box implementations perform well on various computing platforms. Furthermore, the proposed approach and associated techniques can be adapted to other ciphers. We aim to explore their adoption beyond SLT ciphers in future work.

Acknowledgement

We would like to thank the anonymous reviewers for their valuable feedback and constructive comments. This work was supported by the National Natural Science Foundation of China (Nos. 62472318, 62172301, and 62402342), and Shanghai Sailing Program (No. 24YF2749500).

References

[1] World Bank, “World development report 2021: Data for better lives,” World Bank, Washington, DC, Tech. Rep., 2021, accessed: 2025-07-20. [Online]. Available: https://www.worldbank.org/en/publication/wdr2021

[2] V. Rijmen and J. Daemen, “Advanced encryption standard,” *Proceedings of federal information processing standards publications, national institute of standards and technology*, vol. 19, p. 22, 2001.

[3] IoT Analytics, “State of iot 2025: Number of connected iot devices growing 13% to 18.8 billion globally,” 2025, accessed: 2025-07-20. [Online]. Available: https://iot-analytics.com/number-connected-iot-devices

[4] Statista, “Number of internet of things (iot) connected devices worldwide from 2019 to 2030,” 2025, accessed: 2025-07-20. [Online]. Available: https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/

[5] Samsara, “Samsara: The connected operations cloud,” 2025, accessed: 2025-07-20. [Online]. Available: https://www.samsara.com

[6] Google, “Connected home: Smart home devices,” 2025, accessed: 2025-07-20. [Online]. Available: https://store.google.com/category/connected_home

[7] Samsung, “Smartthings app support,” 2025, accessed: 2025-07-20. [Online]. Available: https://www.samsung.com/us/support/owners/app/smartthings

[8] Apple, “Home app accessories,” 2025, accessed: 2025-07-20. [Online]. Available: https://www.apple.com/home-app/accessories/

[9] Teladoc Health, “Teladoc health: Virtual care and telehealth solutions,” 2025, accessed: 2025-07-20. [Online]. Available: https://www.teladochealth.com/

[10] Bosch, “Bosch iot suite: Iot platform for connected solutions,” 2025, accessed: 2025-07-20. [Online]. Available: https://bosch-iot-suite.com/

[11] Verizon Connect, “Verizon connect: Fleet management and telematics solutions,” 2025, accessed: 2025-07-20. [Online]. Available: https://www.verizonconnect.com/

[12] LoRa Alliance, “Lorawan specification v1.0,” LoRa Alliance, Tech. Rep., 2025, accessed: 2025-07-20. [Online]. Available: https://resources.lora-alliance.org/technical-specifications/lorawan-specification-v1-0

[13] Connectivity Standards Alliance, “Zigbee: Iot connectivity solutions,” 2025, accessed: 2025-07-20. [Online]. Available: https://csa-iot.org/all-solutions/zigbee/

[14] Bluetooth SIG, “Bluetooth specifications,” 2025, accessed: 2025-07-20. [Online]. Available: https://www.bluetooth.com/specifications/specs

[15] I. Butun, N. Pereira, and M. Gidlund, “Security risk analysis of lorawan and future directions,” *Future Internet*, vol. 11, no. 1, 2019.

[16] G. Camurati, A. Francillon, and F.-X. Standaert, “Understanding screaming channels: From a detailed analysis to improved attacks,” *IACR Transactions on CHES*, vol. 2020, no. 3, p. 358–401, Jun. 2020.

[17] J. Tournier, F. Lesueur, F. Le Mouël, L. Guyon, and H. Ben-Hassine, “A survey of iot protocols and their security issues through the lens of a generic iot stack,” *Internet of Things*, vol. 16, p. 100264, 07 2020.

[18] S. Chow, P. A. Eisen, H. Johnson, and P. C. v. Oorschot, “White-box cryptography and an aes implementation,” in *Revised Papers from the 9th Annual International Workshop on Selected Areas in Cryptography*, 2002, p. 250–270.

[19] S. Chow, P. Eisen, H. Johnson, and P. C. Van Oorschot, “A white-box des implementation for drm applications,” in *Digital Rights Management*, 2003, pp. 1–15.

[20] C. Delerablée, T. Lepoint, P. Paillier, and M. Rivain, “White-box security notions for symmetric encryption schemes,” in *Selected Areas in Cryptography*, 2014, pp. 247–264.

[21] E. A. Bock, A. Amadori, C. Brzuska, and W. Michiels, “On the security goals of white-box cryptography,” *IACR Transactions on CHES*, pp. 327–357, 2020.

[22] A. Biryukov, C. Bouillaguet, and D. Khovratovich, “Cryptographic schemes based on the asasa structure: Black-box, white-box, and public-key,” in *ASIACRYPT*, vol. 8873, 2014, pp. 63–84.

[23] A. Bogdanov and T. Isobe, “White-box cryptography revisited: Space-hard ciphers,” in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015, p. 1058–1069.

[24] A. Bogdanov, T. Isobe, and E. Tischhauser, “Towards practical white-box cryptography: Optimizing efficiency and space hardness,” in *ASIACRYPT*, 2016, pp. 126–158.

[25] P.-A. Fouque, P. Karpman, P. Kirchner, and B. Minaud, “Efficient and provable white-box primitives,” in *International Conference on the Theory and Application of Cryptology and Information Security*, 2016, pp. 159–188.

[26] A. Herzberg, H. Shulman, A. Saxena, and B. Crispo, “Towards a theory of white-box security,” in *Emerging Challenges for Security, Privacy and Trust*, 2009, pp. 342–352.

[27] A. Saxena, B. Wyseur, and B. Preneel, “Towards security notions for white-box cryptography,” in *Information Security*, 2009, pp. 49–58.

[28] T. Lin, X. Lai, W. Xue, and G. Huang, “Discussion on the theoretical results of white-box cryptography,” *Science China Information Sciences*, pp. 1–11, 2016.

[29] Y. Xiao and X. Lai, “A secure implementation of white-box aes,” in *2nd International Conference on Computer Science and its Applications*, 2009, pp. 1–6.

[30] M. Karroumi, “Protecting white-box aes with dual ciphers,” in *International Conference on Information Security and Cryptology*, 2011, pp. 278–291.

[31] S. Lee, D. Choi, and Y.-J. Choi, “Conditional re-encoding method for cryptanalysis-resistant white-box aes,” *ETRI Journal*, vol. 37, no. 5, pp. 1012–1022, 2015.

[32] J. Bringer, H. Chabanne, and E. Dottax, “White box cryptography: Another attempt,” Cryptology ePrint Archive, Paper 2006/468, 2006.

[33] Y. Shi, W. Wei, Z. He, and H. Fan, “An ultra-lightweight white-box encryption scheme for securing resource-constrained iot devices,” in

Proceedings of the 32nd Annual Conference on Computer Security Applications, 2016, pp. 16–29.

[34] O. Billet, H. Gilbert, and C. Ech-Chatbi, "Cryptanalysis of a white box aes implementation," in *Selected Areas in Cryptography*, vol. 3357, 2005, pp. 227–240.

[35] Y. De Mulder, P. Roelse, and B. Preneel, "Cryptanalysis of the xiao – lai white-box aes implementation," in *Selected Areas in Cryptography*, vol. 7707, 2013, pp. 34–49.

[36] T. Lepoint, M. Rivain, Y. De Mulder, P. Roelse, and B. Preneel, "Two attacks on a white-box aes implementation," in *Selected Areas in Cryptography*, 2014, pp. 265–285.

[37] Y. Shi and H. Fan, "On security of a white-box implementation of shark," in *Information Security*, 2015, pp. 455–471.

[38] W. Michiels, P. Gorissen, and H. D. L. Hollmann, "Cryptanalysis of a generic class of white-box implementations," in *Selected Areas in Cryptography*, 2008, pp. 414–428.

[39] L. Goubin, J.-m. Masereel, and M. Quisquater, "Cryptanalysis of white box des implementations," in *Selected Areas in Cryptography*, 2007, pp. 278–295.

[40] L. Tolhuizen, "Improved cryptanalysis of an aes implementation," in *Proceedings of the 33rd WIC Symposium on Information Theory*, 2012.

[41] Y. De Mulder, B. Wyseur, and B. Preneel, "Cryptanalysis of a perturbated white-box aes implementation," in *INDOCRYPT*, 2010, pp. 292–310.

[42] U.S. Payments Forum, "Emv payment tokenization primer: Lessons learned," U.S. Payments Forum, Tech. Rep., June 2019, accessed: 2025-04-14. [Online]. Available: https://www.uspaymentsforum.org/wp-content/uploads/2019/06/EMV-Payment-Tokenization-Primer-Lessons-Learned-FINAL-June-2019.pdf

[43] Irdeto, "Irdeto launches renewed irdeto activecloak for media," February 2023, accessed: 2025-04-14. [Online]. Available: https://irdeto.com/news/irdeto-launches-renewed-irdeto-activecloak-for-media

[44] Verimatrix, "Whitebox cryptography," 2023, accessed: 2025-04-14. [Online]. Available: https://www.verimatrix.com/cybersecurity/whitebox-cryptography/

[45] S. Lee, "A masked white-box cryptographic implementation for protecting against differential computation analysis," *IACR Cryptology ePrint Archive*, vol. 2017, p. 267, 2017.

[46] T. Xu, C. Wu, F. Liu, and R. Zhao, "Protecting white-box cryptographic implementations with obfuscated round boundaries," *Science China Information Sciences*, vol. 61, no. 3, p. 039103, 2017.

[47] Y. Shi, Q. Liu, and Q. Zhao, "A secure implementation of a symmetric encryption algorithm in white-box attack contexts," *Journal of Applied Mathematics*, 2013.

[48] A. Biryukov and A. Shamir, "Structural cryptanalysis of sasas," in *EUROCRYPT*, 2001, pp. 395–405.

[49] A. Biryukov and A. Udovenko, "Dummy shuffling against algebraic attacks in white-box implementations," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 2021, pp. 219–248.

[50] S. Lee, N.-S. Jho, and M. Kim, "Table redundancy method for protecting against fault attacks," *IEEE Access*, vol. 9, pp. 92 214–92 223, 2021.

[51] S. Lee and M. Kim, "Improvement on a masked white-box cryptographic implementation," *IEEE Access*, vol. 8, pp. 90 992–91 004, 2020.

[52] O. Seker, T. Eisenbarth, and M. Liskiewicz, "A white-box masking scheme resisting computational and algebraic attacks," *IACR Transactions on CHES*, pp. 61–105, 2021.

[53] J. W. Bos, C. Hubain, W. Michiels, and P. Teuwen, "Differential computation analysis: Hiding your white-box designs is not enough," in *IACR Transactions on CHES*, 2016.

[54] A. Amadori, W. Michiels, and P. Roelse, "A dfa attack on white-box implementations of aes with external encodings," in *Selected Areas in Cryptography*, 2019, pp. 591–617.

[55] M. Rivain and J. Wang, "Analysis and improvement of differential computation attacks against internally-encoded white-box implementations," *IACR Transactions on CHES*, pp. 225–255, 2019.

[56] L. Goubin, M. Rivain, and J. Wang, "Defeating state-of-the-art white-box countermeasures with advanced gray-box attacks," *IACR Transactions on CHES*, vol. 2020, no. 3, pp. 454–482, 2020.

[57] A. Charlès and A. Udovenko, "Lpn-based attacks in the white-box setting," *IACR Transactions on CHES*, vol. 2023, no. 4, 2023.

[58] Y. Tang, Z. Gong, T. Sun, J. Chen, and F. Zhang, "Adaptive side-channel analysis model and its applications to white-box block cipher implementations," in *Information Security and Cryptology: 17th International Conference, Inscrypt*, 2021, pp. 399–417.

[59] Y. Tang, Z. Gong, J. Chen, and N. Xie, "Higher-order dca attacks on white-box implementations with masking and shuffling countermeasures," *IACR Transactions on CHES*, pp. 369–400, 2023.

[60] A. Amadori, W. Michiels, and P. Roelse, "Automating the bge attack on white-box implementations of aes with external encodings," in *International Conference on Consumer Electronics*, 2020, pp. 1–6.

[61] E. Biham, R. Anderson, and L. Knudsen, "Serpent: A new block cipher proposal," in *Fast Software Encryption*, 1998, pp. 222–238.

[62] V. Rijmen, J. Daemen, B. Preneel, A. Bosselaers, and E. De Win, "The cipher shark," in *Fast Software Encryption*, 1996, pp. 99–111.

[63] Y. Shi and H. Fan, "On security of lee et al.'s white-box aes," 2016.

[64] S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi, "Towards sound approaches to counteract power-analysis attacks," in *CRYPTO*, 1999, pp. 398–412.

[65] Ubidots. (2025) Top industrial iot gateways. Accessed: Nov. 18, 2025. [Online]. Available: https://ubidots.com/blog/top-industrial-iot-gateways/

[66] S. H. Abdelwahed, I. M. Hefny, M. Hegazy, L. A. Said, and A. Soltan, "Survey of iot multi-protocol gateways: Architectures, protocols and cybersecurity," *Internet of Things*, vol. 33, p. 101703, 2025.

[67] SideChannelMarvels, "Bluegalaxyenergy," 2025, accessed: 2025-11-25. [Online]. Available: https://github.com/SideChannelMarvels/BlueGalaxyEnergy

[68] OpenWhiteBox, "Aes cryptanalysis," 2025, accessed: 2025-11-25. [Online]. Available: https://github.com/OpenWhiteBox/AES/tree/master/cryptanalysis

[69] A. Bogdanov, M. Rivain, P. S. Vejre, and J. Wang, "Higher-order dca against standard side-channel countermeasures," in *International Workshop on Constructive Side-Channel Analysis and Secure Design*, 2019, pp. 118–141.

[70] SideChannelMarvels, "Darkphoenix: A tool for side-channel reverse engineering of white-box ciphers," https://github.com/SideChannelMarvels/DarkPhoenix, 2021, accessed April 13, 2025.

[71] A. Biryukov, C. D. Cannière, A. Braeken, and B. Preneel, *A Toolbox for Cryptanalysis: Linear and Affine Equivalence Algorithms*, ser. EuroCrypt, 2003.

[72] K. Bai and C. Wu, "An aes-like cipher and its white-box implementation," *The Computer Journal*, p. bxv119, 2016.

[73] Y. Tang, Z. Gong, T. Sun, J. Chen, and Z. Liu, "Wbmatrix: An optimized matrix library for white-box block cipher implementations," *IEEE Transactions on Computers*, vol. 71, no. 12, pp. 3375–3388, 2022.

[74] SCNUCrypto, "Wbmatrix: White-box matrix encoding tool," https://github.com/scnucrypto/WBMatrix, 2022, accessed April 13, 2025.

[75] B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, and N. Ferguson, "Twofish: A 128-bit block cipher," *NIST AES Proposal*, vol. 15, no. 1, pp. 23–91, 1998.

[76] B. Schneier, "Description of a new variable-length key, 64-bit block cipher (blowfish)," in *Fast Software Encryption, Cambridge Security Workshop*, vol. 809, 1993, pp. 191–204.

[77] T. K. Mondal, *Camellia*. Springer Berlin Heidelberg, 2011, pp. 15–39.

Appendix A
Artifact Appendix

The proposed WBSLT is a novel design framework for tabulated white-box implementations of substitution-linear transformation (SLT) ciphers. This artifact appendix presents setup, configuration, and evaluation details.

A. *Description & Requirements*

This section provides the necessary information for accessing the artifact, hardware and software requirements, and benchmarks.

1) How to access: The source code is available on Zenodo with DOI: https://doi.org/10.5281/zenodo.17543341. There are two alternative ways to access the artifact.

- Access through Github: https://github.com/mmt200088/NDSS2026-WBSLT.git.
- Access through Docker: https://hub.docker.com/r/gaotianchen/ndss2026-wbslt.

2) Hardware dependencies:

- Hardware dependencies 1 (HD1):
  - Raspberry Pi 4b.
  - Raspberry Pi 5.
  - SX1262 LoRa Node with Semtech SX1262 chipsets.
  - Sonoff Zigbee 3.0 USB Dongle Plus with Silicon Labs EFR32MG32 chipset.

3) Software dependencies:

- Software dependencies 1 (SD1):
  - GCC $\geq$ 8.1.
  - Python $\geq$ 3.10.
  - CMake $\geq$ 2.8.
- Software dependencies 2 (SD2):
  - GCC $\geq$ 8.1.
  - Python $\geq$ 3.10.
  - CMake $\geq$ 2.8.
  - Golang $\geq$ 1.13.
  - pip $\geq$ 24.0.
  - libgmp-dev $\geq$ 6.3.0.
  - libntl-dev $\geq$ 11.5.1.

4) Benchmarks:

- Experiments E1, E2 and E4 can be run on a commodity desktop machine, where E1 and E2 require SD1 and E4 requires SD2. Experiment E3 requires access to IoT gateways supporting communication protocols Bluetooth Low Energy (BLE), LoRaWAN and Zigbee with requirements HD1 and SD1, and we have provided guides of using SSH to our own infrastructure in README in the artifact.
- Experiment E2 involves comparison with other white-box AES (WB-AES) implementations in https://github.com/scnucrypto/WBMatrix, and we have included these implementations in the artifact.

B. Artifact Installation & Configuration

This section provides the steps to install and configure the artifact using either Github or Docker.

- Access through Github:

```
$ git clone https://github.com/mmt200088/
    NDSS2026-WBSLT.git
$ cd NDSS2026-WBSLT
$ sh setup.sh
```

- Access through Docker:

```
$ docker pull gaotianchen/ndss2026-wbslt
$ docker run -it ndss2026-wbslt /bin/bash
```

C. Major Claims

The artifacts support the following major claims presented in our paper:

- (C1): WBSLT is a novel framework to design secure white-box implementations of arbitrary SLT ciphers with efficient encryption speed. Experiment (E1) proves this claim by:
  - (RQ1 in initial submission) Generating white-box implementations under various SLT cipher configurations and evaluating the encryption speed, for which the results are illustrated in Figure 15.
  - (Major revision) Generating white-box implementations of WBSLT-based white-box SHARK and PRESENT for encryption speed evaluation.
- (C2): WBSLT can be applied on AES-128 and generate the general WB-AES implementation. We also present optimization strategies to decrease the storage and improve the efficiency of the general WB-AES. The generated WB-AES implementation performs reasonably well. Experiment (E2) proves this claim by:
  - (RQ2 in initial submission) Generating our optimized WB-AES implementation and compare the table generation time and encryption speed with other WB-AES implementations, for which the results are illstrated in Figure 16.
  - (Major revision) Generating our general WB-AES implementation and compare the table generation time and encryption speed with other WB-AES implementations and our optimized WB-AES.
- (C3): Our optimized and general WB-AES can be integrated into wireless communication protocols BLE, LoRaWAN and Zigbee with no efficiency degradation. Experiment (E3) proves this claim by:
  - (RQ3 in initial submission) Integrating our optimized WB-AES implementation into these protocols to compare the data transmission speed with the original AES implementation, for which the results are illustrated in Figure 17.
  - (Major revision) Integrating our optimized WB-AES implementation into these protocols to compare the data transmission speed with the original AES implementation.
- (C4): WBSLT fuzzes the boundaries of components to defend against various white-box attacks. Experiment (E4) proves this claim by:
  - (Initial submission) Conducting DCA and DFA attacks on WBSLT, for which the analysis and results are illustrated in Section 4.3.
  - (Major revision) Conducting algebraic and structural attacks (i.e., BGE, MGH, LRDRP and BS attacks) on WBSLT, for which the analysis are illustrated in Section 4.2.

D. Evaluation

The following experiments prove the major claims above and validate the functionality of the artifacts.

1) Experiment (E1): [SLT Application Experiment] [5 human-minutes + 6 compute-hour]: This experiment generates white-box implementations under various SLT configurations, and evaluates their encryption speed using the generated implementations.

[Preparation] None beyond Appendix A-B.

[Execution] Run

```
$ cd NDSS2026-WBSLT/E1-SLT-Application
$ sh run-e1.sh
```

After completion, the results will be stored in `NDSS2026-WBSLT/E1-SLT-Application/results/`.

[Results] Run

```
$ sh run-get-results.sh
```

After completion, the data will be stored in `NDSS2026-WBSLT/E1-SLT-Application/results/data.csv`, and an illustration figure will be stored in `NDSS2026-WBSLT/E1-SLT-Application/results/figure.png`. The results will follow similar trends in Figure 15.

2) Experiment (E2): [AES Performance Experiment] [5 human-minutes + 4 compute-hour]: This experiment generates our general and optimized white-box implementations, and evaluates the table generation and encryption efficiency.

[Preparation] None beyond Appendix A-B.

[Execution] Run

```
$ cd NDSS2026-WBSLT/E2-AES-Performance
$ sh run-e2.sh
```

After completion, the results will be stored in `NDSS2026-WBSLT/E2-AES-Performance/results/`.

[Results]

```
$ sh run-get-results.sh
```

After completion, the data will be stored in `NDSS2026-WBSLT/E2-AES-Performance/results/data.csv`, and an illustration figure will be stored in `NDSS2026-WBSLT/E2-AES-Performance/results/figure.png`. The results will follow similar trends in Figure 16.

3) Experiment (E3): [IoT Integration Experiment] [20 human-minutes + 1 compute-hour]: This experiment tests IoT protocol communication efficiency using traditional AES implementation and our general and optimized WB-AES implementations.

[Preparation] Please refer to Section 5.3.1 in the README in the artifact to install and configure the VPN.

[Execution] Please refer to Section 5.3.2 in the README in the artifact to run this experiment on our machines.

[Results] Please refer to Section 5.3.3 in the README in the artifact to get the results.

4) Experiment (E4): [Attack Resistance Experiment] [5 human-minutes + 7 compute-hour]: This experiment conducts various white-box attacks on our WBSLT to test the resistance.

[Preparation] None beyond Appendix A-B.

[Execution] Run

```
$ cd NDSS2026-WBSLT/E4-Attack-Resistance
$ sh run-e4.sh
```

After completion, the results will be stored in `NDSS2026-WBSLT/E4-Attack-Resistance/results/`.

[Results]

```
$ sh run-get-results.sh
```

After completion, the result will be displayed on the terminal. The attack results of DCA attack should follow similar trends in Table 4, and the results of other attacks should have zero success time on WBSLT.