IACR Transactions on Symmetric Cryptology
ISSN XXXX-XXXX, Vol. 0, No. 0, pp. 1–27. DOI:XXXXXXXX

Key Guidance Invocation: A White-box Mode Enables Strong Space Hardness under Adaptively Chosen-Space Attacks

Yipeng Shi, Xiaolin Zhang, Boshi Yuan, Chenghao Chen, Jintong Yu, Yuxuan Wang, Chi Zhang and Dawu Gu

Shanghai Jiao Tong University, Shanghai, China,

Abstract. The notion of space hardness serves as a quantitative measure to characterize the resilience of dedicated white-box schemes against code-lifting attacks, making it a widely utilized metric in the field. However, achieving strong space hardness (SSH) under the adaptively chosen-space attack model (ACSAM) remains an unresolved challenge, as no existing white-box scheme has given SSH guarantees under ACSAM.
To address the problem, we introduce a novel mode of operation tailored for white-box cryptography, termed the Key Guidance Invocation (KGI) mode. Our security analysis reveals that the KGI mode not only significantly strengthens the resistance to adaptively chosen-space attacks, but also ensures SSH under ACSAM. Moreover, we propose a dedicated white-box construction, RubikStone-(n,n_{in},R,s), which directly leverages the concept of the lookup table pool. RubikStone offers enhanced flexibility in lookup table utilization compared to existing white-box constructions and is particularly well-suited to the KGI mode.
Additionally, we instantiate RubikStone-(256,8,12,2^{16}) with the KGI mode, resulting in $\mathsf{RS}_{\mathsf{KGI}}$-256, which delivers $(T/4, 127.99)$-SSH security guarantees under ACSAM. Remarkably, $\mathsf{RS}_{\mathsf{KGI}}$-256 also shows superior performance, surpassing the efficiency of white-box AES based on the CEJO framework by 27.1% in real-world settings. Besides, we conduct a comprehensive statistical analysis of the operations in all existing white-box ciphers. Our findings indicate that $\mathsf{RS}_{\mathsf{KGI}}$-256 remains highly competitive in computational efficiency despite offering unprecedented security.

Keywords: White-box cryptography · Space hardness · Lookup table pool · Key guidance invocation · Mode of operation.

1 Introduction

1.1 White-box Cryptography

In untrusted environments, especially devices lacking sufficient hardware support, the security of the execution of cryptographic algorithms is a topic of widespread discussion. In 2002, Chow *et al.* [CEJvO02a, CEJvO02b] differentiated this scenario and traditional ones by white- and black-box contexts and pioneered a solution known as white-box cryptography. The main idea of white-box cryptography is pre-storing intermediate values that keys may participate in within lookup tables. These table entries are then utilized to substitute the keys during cryptographic operations, effectively ensuring that they do not appear directly in the implementation of cryptographic algorithms. Additional internal and external encoding and masking methods are also widely employed to increase the security of lookup table entries. Compared to computationally expensive fully homomorphic encryption [MOO+14] and frequently vulnerable secure enclaves [BPS17, MIE17, BMW+18], white-box cryptography has been widely welcomed by industry and is even required for use in some standards [Pay, BBF+20].

Black-box and White-box Contexts. According to Chow *et al.*'s initial perspective, in the white-box context, the adversary is assumed to have "full access" to the implementation of cryptographic algorithms, enabling them to observe the dynamic execution process of the algorithm and modify details at will. While in the traditional black-box context, the adversary can only observe the input-output behavior of the algorithm as a whole and conduct known-plaintext, chosen-plaintext, chosen-ciphertext, or even adaptively chosen-ciphertext attacks based on that. In subsequent works, the notion of white-box context has been further refined (see Section 2.2).

Key Extraction and Code Lifting [DLPR13]. Undoubtedly, the white-box adversary possesses unprecedentedly powerful capabilities in the assumptions made by Chow *et al.*. With such abilities, he can execute cryptographic algorithms and possess the same privileges as a legitimate user. Sometimes, the white-box adversary is actually a legitimate user, referred to as a malicious user. However, the "privileges" are often tied to a specific device. The white-box adversary aims to transplant these privileges to other unauthorized devices and profit from them. To achieve the goal, the white-box adversary typically has two approaches:

- **Key Extraction.** The adversary analyses the information observed during the algorithm execution process and recovers the key, which is the core secret for legitimate users to obtain privileges. For an implementation that uses the key in plaintext form, the white-box adversary can even directly obtain this key by observing the intermediate processes of the algorithm execution without additional analysis.
- **Code Lifting.** In some algorithm implementations, such as the white-box implementation proposed by Chow *et al.*, recovering keys remains a very difficult task for the white-box adversary. In this case, the white-box adversary can simply isolate the cryptographic code in the implementation and lift it as a whole to other devices. In this process, the lifted cryptographic code can be considered as an inflated variant of the original key.

How to effectively defend the two types of attacks in a white-box context has been the focus of white-box cryptography research.

1.2 Related Works and Motivation

White-box Implementation of Existing Block Ciphers. In the early stages of white-box cryptography research, the primary focus was on improving the implementation of some existing block ciphers, especially AES [CEJvO02a, BCD06, Kar10, XL09, LLY14] and DES [CEJvO02b, LN05, WP05]. However, most of them have been explicitly broken [BGE04, GMQ07, JBF02, LRM+13, MGH08, MRP12, MWP10, WMGP07]. Due to the significant challenge of designing a white-box implementation for an existing cipher, these works and even some recent ones [RP20, RVP22] only aimed at preventing key extraction attacks.

Dedicated White-box Ciphers. In order to better prevent both key extraction and code lifting attacks, several dedicated white-box ciphers [BI15, BIT16, FKKM16, CCD+17, KSHI20, KLLM20, KI21, YZDZ23] were proposed later, which are more suitable for white-box application scenarios. By generating lookup tables based on a well-studied block cipher, these ciphers reduce the security against key extraction in a white-box context to that against key recovery in a black-box context. At the same time, by using lookup tables ranging from several hundred kilobytes to tens of gigabytes, they mitigate code lifting attacks to some extent.

Space Hardness. Biryukov *et al.* proposed the notion of weak white-box security in [BBK14], which was also called incompressibility by De Mulder [Mul14]. The property uses the minimum size of code that the white-box adversary needs to extract from the software for an equivalent key to evaluate the security of white-box ciphers. Based on the property of weak white-box security, Bogdanov and Isobe proposed (M, Z)-space hardness to evaluate the difficulty of code lifting attacks in a more quantitative way[BI15]. Which was widely used in subsequent works [BIT16, FKKM16, CCD$^+$17, KSHI20, KLLM20, KI21, YZDZ23]. Subsequently, they introduced a more fine-grained method to capture the resistance towards code lifting at ASIACRYPT 2016: weak and strong (M, Z)-space hardness with respect to various abilities of the adversaries (see Section 2.2). The method was widely used in those works [BIT16, FKKM16, CCD$^+$17, KSHI20, KLLM20, KI21, YZDZ23].

Motivation. To the best of our knowledge, although new white-box schemes continue to be proposed, none of them has been able to provide strong space hardness (SSH) under the strongest attack model, namely the adaptively chosen-space attack model (ACSAM) (see Section 2.2). Additionally, there has been no discussion on the modes of operation for white-box ciphers. Therefore, we propose a mode of operation specifically tailored for white-box cryptography, aiming to achieve SSH under ACSAM. To effectively utilize the mode, we introduce a dedicated white-box construction. Additionally, we emphasize the importance of scenario-driven customization in the design of white-box schemes, advocating for solutions that are finely tuned to the unique requirements of specific application contexts. The primary focus of this paper is on cloud-based Digital Rights Management (DRM) systems, a critical scenario for white-box cryptography. To rigorously evaluate the security of our scheme, we formalize the advantage of ACSAM adversaries within this context through a two-stage game framework. This formal analysis serves to validate the effectiveness and robustness of our proposals in resisting adaptively chosen-space attacks.

1.3 Our Contribution

(1) Novel Mode of Operation for White-box Schemes. We introduce a mode of operation for white-box schemes, named the Key Guidance Invocation (KGI) mode. The mode is built upon two novel concepts introduced in this work: the guidance key and the lookup table pool. The guidance key is a randomized sequence designed to govern the invocation process of lookup tables, ensuring that the manner in which lookup tables are accessed during each execution of the algorithm is non-deterministic and unpredictable. The lookup table pool consists of multiple lookup tables of identical specifications, and the pool should be sufficiently large to provide a broad selection space for the guidance key. Furthermore, we provide a detailed description of the generation algorithm for the lookup table pool, the scheduling rules for the guidance key, and an application protocol to facilitate the practical deployment of the KGI mode. These elements synergistically augment the adaptability and security of the KGI mode in practical applications.

(2) Dedicated White-box Construction with Enhanced Flexibility in Lookup Table Utilization. We propose a dedicated white-box construction, termed RubikStone-(n, n_{in}, R, s). it leverages a balanced Feistel network to guarantee the scheme's reversibility irrespective of the specifications of the lookup tables. Furthermore, the design introduces an additional parameter that defines the size of the lookup table pool. This inclusion endows RubikStone with enhanced flexibility in the utilization of lookup tables, thereby facilitating a more effective application of the KGI mode.

(3) Complete Application Case and Performance Evaluation. We present a complete application case, denoted as $\mathsf{RS}_{\mathsf{KGI}}$-256, by integrating the KGI mode with RubikStone, and evaluate its performance. Real-world experimental results demonstrate that $\mathsf{RS}_{\mathsf{KGI}}$-256 achieves a 27.1% higher efficiency compared to the white-box AES implementation under the CEJO framework. Furthermore, we conduct a comprehensive statistical analysis of the operations in all existing white-box schemes, revealing that $\mathsf{RS}_{\mathsf{KGI}}$-256 employs significantly fewer operations, underscoring its strong potential for practical applications.

(4) Precise Adversary Models and Comprehensive Security Analysis. We employ a two-stage game framework to precisely define the attack objectives and adversarial advantages of an ACSAM adversary. Based on this framework, we rigorously prove that our proposal effectively provides resistance against adaptively chosen space attacks. As a representative, $\mathsf{RS}_{\mathsf{KGI}}$-256 achieves $(T/4, 127.99)$-SSH security guarantees under ACSAM. Additionally, we conduct a comprehensive security analysis of our scheme using multiple cryptanalysis and derive explicit security bounds. The results indicate that our scheme exhibits strong resilience against these attacks, further validating its robustness.

1.4 Organization

In Section 2, we introduce the models and security notions used in the paper. Then we formally introduce the KGI mode along with its associated terminology and constructions in Section 3. In Section 4, we present the construction of RubikStone and its specifications. Subsequently, in Section 5, we provide a complete application case based on the KGI mode and the RubikStone construction, accompanied by a corresponding performance analysis. In Section 6, we conduct a comprehensive security analysis of our proposals and derive explicit security bounds using $\mathsf{RS}_{\mathsf{KGI}}$-256 as an example. Finally, we offer concluding remarks on our research in Section 7.

2 Preliminaries

2.1 Application Scenarios of White-box Cryptography

2.1.1 Cloud-based DRM

Digital Rights Management (DRM) is extensively employed to regulate authorized access to digital content, representing the primary and most critical application scenario for white-box cryptography. To minimize development and operational expenses for DRM developers and content service providers, while delivering more flexible and diversified services to content consumers, contemporary DRM systems have evolved into cloud-based content distribution frameworks [LPSS16, Inc14].

Figure 1 illustrates the architecture of a cloud-based DRM service. Digital content stored on the cloud server is encoded and encrypted before being distributed to consumers' devices. To decode the encrypted content, consumers must interact with the cloud server for rights verification. During this process, the client sends a verification request along with identifying information, such as an ID and a signature. Upon successful verification of legitimate users, the cloud server transmits the corresponding authorization key to the client via a secure channel. Legitimate consumers can then use this key to decode the encrypted content. However, if authorization keys are stored and utilized using conventional methods, a white-box adversary within the client device could easily extract and

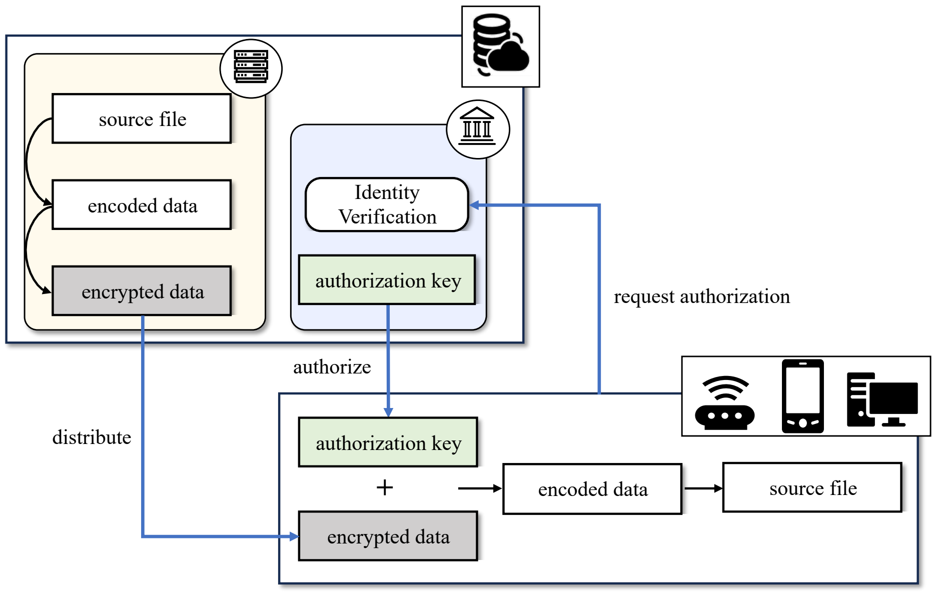


Figure 1: Overview of cloud-based DRM

distribute these keys to unauthorized users. Therefore, white-box cryptography plays a crucial role in enhancing the security of cloud-based DRM systems.

2.1.2 White-box Cryptography Varies across Different Scenarios

In addition to cloud-based DRM, white-box cryptography is also regarded as an effective software security solution in various scenarios, including Host Card Emulation (HCE) for mobile payment services and memory-leakage resilient software [BIT16]. Previous research has often sought to address these scenarios collectively, striving to design a universal white-box cryptography algorithm capable of providing security guarantees across all these contexts.

However, we argue that the resources and constraints in different white-box application scenarios vary significantly, and the application methods of white-box cryptography can differ substantially. For example, a critical distinction between cloud-based DRM and mobile payments is the security context of the encryptor. In cloud-based DRM, the encryptor typically operates in a secure environment shielded from white-box attacks, often leveraging cloud servers with hardware-based protection mechanisms. Conversely, in mobile payment scenarios, the encryptor is also exposed to a white-box environment. On the other hand, in cloud-based DRM, the interests of adversaries and legitimate users may occasionally align, leading to potential collaboration in compromising protected digital content. However, in mobile payment scenarios, the interests of adversaries are typically opposed to those of legitimate users, as no legitimate user would desire an adversary to spend their funds. These differences result in distinct roles for white-box cryptography: in cloud-based DRM, it primarily focuses on privacy preservation, whereas in mobile payment scenarios, its role extends to providing authenticated encryption within a white-box context.

As a result, white-box cryptography, as a software security mechanism that balances security assurance with available resources, should be studied in a specialized and scenario-specific manner. This paper primarily concentrates on application scenarios like cloud-based DRM, operating under the assumption that the encryptor and decryptor are located on distinct terminals, with the encryptor being immune to white-box attacks.

2.2 Security Notions

2.2.1 Space Hardness and Adversary Models

Bogdanov and Isobe [BI15] have defined the following weak and strong $(\boldsymbol{M}, \boldsymbol{Z})$-space hardness based on different security objectives, which now serve as the most critical and widely adopted security evaluation metrics for dedicated white-box ciphers.

Definition 1. (Weak $(\boldsymbol{M}, \boldsymbol{Z})$-Space Hardness (WSH)[BI15].) An implementation of a block cipher E_K is weakly (M, Z)-space hard if it is computationally difficult to encrypt (decrypt) any randomly drawn plaintext (ciphertext) with probability of more than 2^{-Z} given any code (table) of size less than M bits.

Definition 2. (Strong $(\boldsymbol{M}, \boldsymbol{Z})$-space hardness (SSH)[BI15].) An implementation of a block cipher E_K is strongly (M, Z)-space hard if it is computationally difficult to obtain a valid plaintext and ciphertext pair with probability of more than 2^{-Z} given any code (table) of size less than M bits.

Three models were proposed in [BIT16] to characterize the white-box adversary's ability to access lookup tables in a more refined way. Each model constrains the adversary to access only a subset of the input-output pairs of tables, differing in the selection methodology.

Definition 3. (Known-Space Attack Model (KSAM) [BIT16].) The adversary obtains a certain number of input-output pairs of tables, where the inputs are randomly chosen.

Definition 4. (Chosen-Space Attack Model (CSAM) [BIT16].) The adversary obtains a certain number of input-output pairs of tables, where the inputs are preselected according to the adversary's will.

Definition 5. (Adaptively Chosen-Space Attack Model (ACSAM) [BIT16].) The adversary obtains a certain number of input-output pairs of tables, where each input is chosen according to the adversary's will, and he can choose the next input after obtaining the outputs corresponding to the previous inputs.

By combining the two security metrics with the three adversary models, we establish six security notions for dedicated white-box ciphers: WSH-KSAM, WSH-CSAM, WSH-ACSAM, SSH-KSAM, SSH-CSAM, and SSH-ACSAM.

2.2.2 ACSAM Games

In real-world settings, the white-box adversary gains control over the decryption device and possesses the ability to lift code from it. The adversary's goal is to leverage the lifted code to replicate all the functionality of the legitimate decryptor. Building on this, we utilize a two-stage game framework between the challenger $\mathcal{C}$ and the adversary $\mathcal{A}$ to more precisely demonstrate the adversary's advantage in compromising WSH and SSH within the ACSAM context.

Formally, let $\Pi = (\mathcal{E}, \mathcal{D})$ be a white-box scheme, and $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be a white-box adversary. The adversary $\mathcal{A}$ attacks the scheme $\diamond$ under ACSAM, running in two stages, as follows:

- **First stage: lifting the codes.** As illustrated in the Figure 2, the first-stage is run by the sub-adversary $\mathcal{A}_1$. We assume that for each entry t_i in Π's lookup tables(LUTs), there exists an index $index_i$ that allows precise access to it. $\mathcal{A}_1$ is allowed to perform adaptively LUT queries on a per-entry basis using the indexes, meaning that he can determine the index of the next query based on the response from the previous query. After b accesses, $\mathcal{A}_1$ generates a lifter $\mathcal{L}$ and give it to $\mathcal{A}_2$. $\mathcal{L}$ comprises all the LUT query results, serving as a substitute for the functionality of the decryptor.

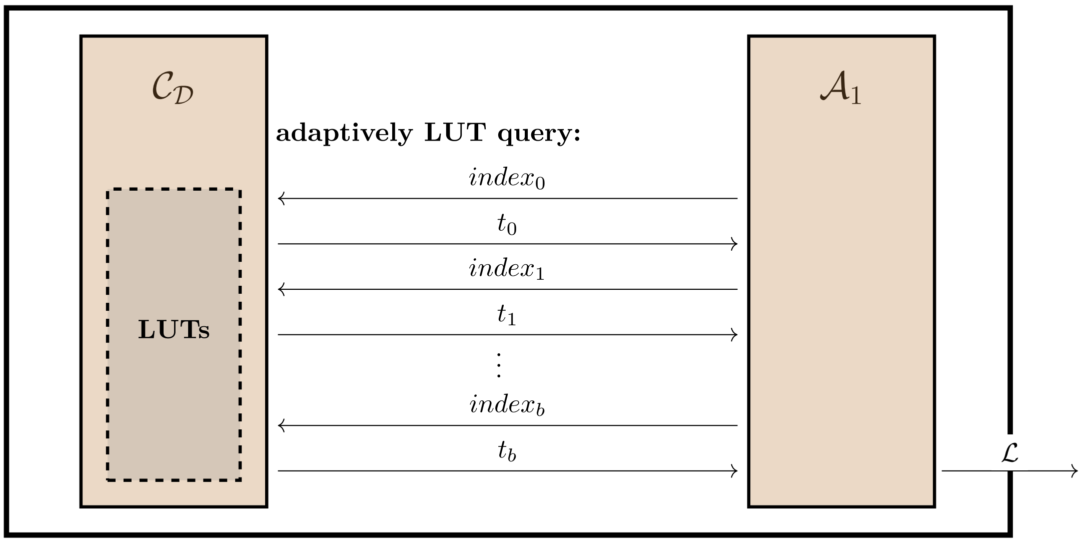
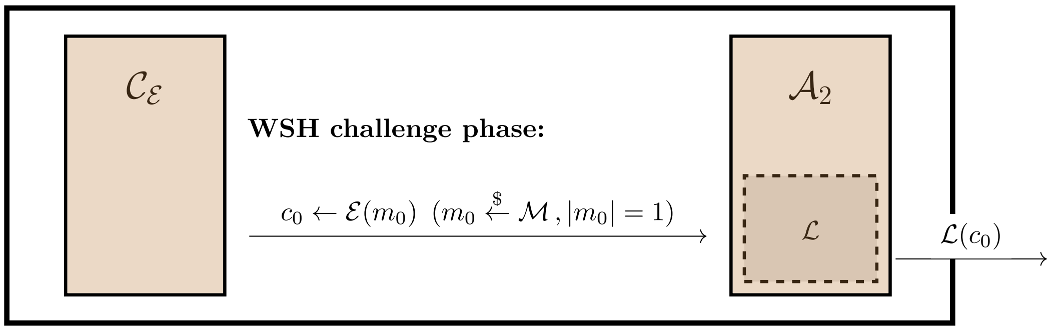
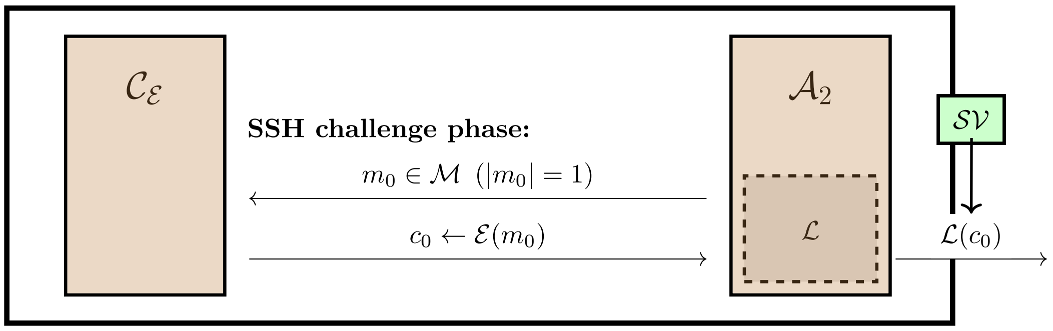


Figure 2: The first stage of the game $\mathsf{G}_{\Pi,\mathcal{A}}^{\mathsf{WSH-ACSAM}}$ and $\mathsf{G}_{\Pi,\mathcal{A}}^{\mathsf{SSH-ACSAM}}$


$\mathcal{C}_{\mathcal{E}}$
WSH challenge phase:
$c_0 \leftarrow \mathcal{E}(m_0)\ \ (m_0 \xleftarrow{\$} \mathcal{M}, |m_0| = 1)$
$\mathcal{A}_2$
$\mathcal{L}$
$\mathcal{L}(c_0)$

(a) The second stage of the game $\mathsf{G}_{\Pi,\mathcal{A}}^{\mathsf{WSH-ACSAM}}$


$\mathcal{C}_{\mathcal{E}}$
SSH challenge phase:
$m_0 \in \mathcal{M}\ \ (|m_0| = 1)$
$c_0 \leftarrow \mathcal{E}(m_0)$
$\mathcal{A}_2$
$\mathcal{SV}$
$\mathcal{L}$
$\mathcal{L}(c_0)$

(b) The second stage of the game $\mathsf{G}_{\Pi,\mathcal{A}}^{\mathsf{SSH-ACSAM}}$

Figure 3: challenge phase

• **Second stage: challenge phase.** Owing to the distinct adversarial objectives inherent in the WSH-ACSAM and SSH-ACSAM security notions, the second phases of Game $\mathsf{G}_{\Pi,\mathcal{A}}^{\mathsf{WSH-ACSAM}}$ and $\mathsf{G}_{\Pi,\mathcal{A}}^{\mathsf{SSH-ACSAM}}$ diverge accordingly, as illustrated in Figures 3a and Figure 3b, respectively.

- To compromise WSH, the adversary must possess the capability to decrypt a randomly selected ciphertext. During the challenge phase of $\mathsf{G}_{\Pi,\mathcal{A}}^{\mathsf{WSH-ACSAM}}$, $\mathcal{A}_2$ is required to utilize the lifter $\mathcal{L}$ to decrypt a ciphertext $c_0 \leftarrow \mathcal{E}(m_0)$ provided by $\mathcal{C}_{\mathcal{E}}$, where m_0 is a one-block ciphertext randomly drawn from the message space $\mathcal{M}$.
- To compromise SSH, the adversary merely requires a valid plaintext-ciphertext pair, effectively enabling them to select a plaintext that the corresponding ciphertext they are most confident in decrypting. Consequently, in the challenge phase of $\mathsf{G}_{\Pi,\mathcal{A}}^{\mathsf{SSH-ACSAM}}$, $\mathcal{A}_2$ initially selects the one-block plaintext m_0 from $\mathcal{M}$, which is subsequently encrypted by $\mathcal{C}_{\mathcal{E}}$ to generate the corresponding ciphertext c_0. $\mathcal{A}_2$ then employ $\mathcal{L}$ to decrypt this ciphertext. We posit the existence of a supervisor $\mathcal{SV}$ that ensures $\mathcal{A}_2$ does not directly output the result from the selection phase to cheat, implying that if $\mathcal{L}$ produces an output $\mathcal{L}(c_0) = m_0$, it is derived through a formal decryption process.

The advantages of the adversary in breaking WSH or SSH under ACSAM are as follows.

$$\mathsf{Adv}_{\Pi}^{\mathsf{WSH-ACSAM}}(\mathcal{A}) := \mathsf{Pr}[\mathcal{L}(c_0) = m_0]$$
$$\mathsf{Adv}_{\Pi}^{\mathsf{SSH-ACSAM}}(\mathcal{A}) := \mathsf{Pr}[\mathcal{L}(c_0) = m_0]$$

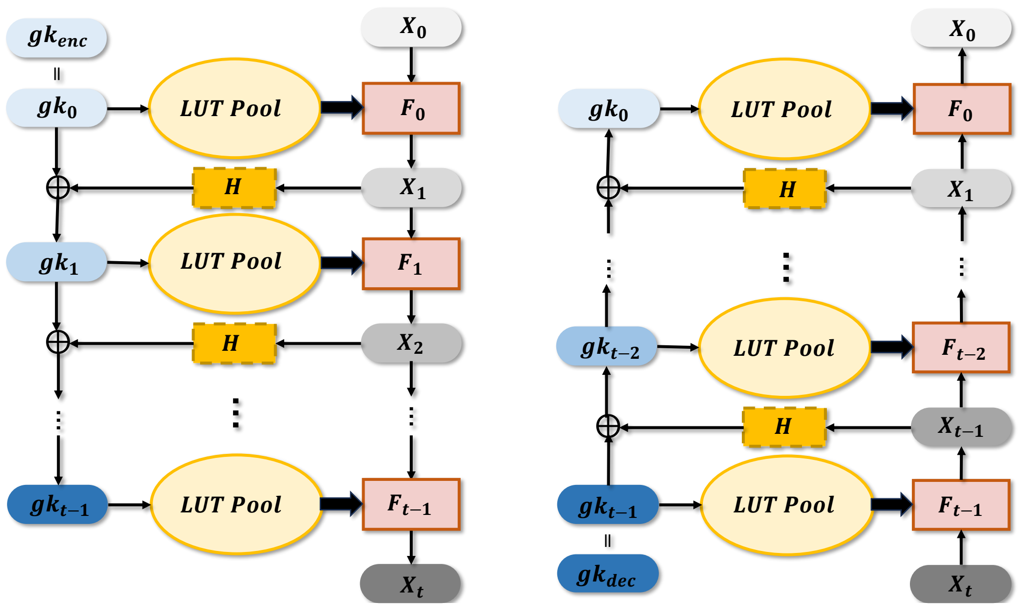
In accordance with Kerckhoffs' Principle, the entirety of a cryptographic system's specifications may be disclosed without compromising security. Conventional cryptographic mechanisms derive their security from the confidentiality of the key, while white-box cryptography's resilience is contingent upon the constraints imposed on an adversary's ability for code lifting.

However, an ACSAM adversary possesses complete freedom to invoke lookup tables, subject only to the number of table entries. Moreover, the number of lookup tables required to encrypt a single plaintext block is typically far fewer than the number of entries he is permitted. This means that, given a deterministic white-box scheme and a randomly selected plaintext block, the ACSAM adversary is fully aware of all the details required to encrypt that block. In $\mathsf{G}_{\Pi,\mathcal{A}}^{\mathsf{SSH-ACSAM}}$, $\mathcal{A}_1$ can pre-select a plaintext m^* and, leveraging the advantages of adaptive selection, acquire all the necessary LUT entries required to generate the corresponding ciphertext c^*. Based on this, $\mathcal{A}_1$ constructs a lifter $\mathcal{L}^*$ that ensures encryption on m^*. Then, in the challenge phase, $\mathcal{A}_2$ once again selects m^* and sends it to $\mathcal{C}_{\mathcal{E}}$, ultimately succeeding with probability 1 in obtaining $\mathcal{L}^*(c^*) = m^*$. Consequently, for a deterministic white-box scheme, the ACSAM adversary can always derive a valid plaintext-ciphertext pair, thereby no strong space hardness can be guaranteed.

3 Mode of Operation for White-box Ciphers

3.1 Key Guidance Invocation Mode

As previously discussed, all deterministic white-box schemes have been demonstrated to lack strong space hardness when subjected to adaptively chosen-space attacks. It is necessary to introduce some randomness to ensure that how a message will be encrypted is unpredictable to the white-box adversary. Therefore, in Figure 4a, we define the first mode of operation tailored for white-box ciphers, i.e. the Key Guidance Invocation (KGI) mode. It employs a random sequence, termed the guidance key, to direct each invocation of the lookup tables.



(a) Encryption mode (b) Decryption mode

Figure 4: Mode of operation for white-box ciphers

Algorithm 1: LUT Pool Generation

```
Input: Input length of tables $n_{in}$, output length of tables $n_{out}$,
       number of tables in the LUT pool $s$.
Output: A LUT pool $\mathcal{P}$.
1 $\mathcal{P} = []$;
2 for $j \leftarrow 0$ to $s - 1$ do
3     $k \leftarrow \text{Gen}(1^\kappa)$;            // $\kappa$ is the security parameter of $E()$
4     for $i \leftarrow 0$ to $2^{n_{in}} - 1$ do
5         $T_j[i] = Truncate(E_k(i\|0^*), n_{out})$;
6                  // $Truncate(x, m)$ means truncating the highest $m$-bit of $x$
7     $\mathcal{P}.append(T_j)$;
8 return $\mathcal{P}$
```

Guidance Key and Lookup Table Pool. A novel concept-the lookup table pool (represented by LUT pool for brevity)-is introduced simultaneously, which is a collection composed of multiple lookup tables with identical specifications. Each lookup table within the LUT pool is assigned a unique index. Essentially, the guidance key represents an ordered subset of these index numbers. We want as many lookup tables as possible within the LUT pool, even significantly more than the number required to encrypt a single plaintext block. It ensures a sufficiently large space for the guidance key. In fact, all existing white-box ciphers can be regarded as special cases of the KGI mode when the LUT pool is very small and the guidance keys are fixed.

Round Function. Then, in each round, we can obtain a particular round function by inputting a specific guidance key into the LUT pool. The function sequentially invokes some lookup tables in an order predefined by the guidance key, thereby performing a permutation on the input. A plaintext block undergoes multiple rounds of such operations to be encrypted into the final ciphertext.

LUT Pool Generation. Let $T : \{0,1\}^{n_{in}} \rightarrow \{0,1\}^{n_{out}}$ be a lookup table that outputs n_{out}-bit $T(x)$ based on an n_{in}-bit input x. Lookup tables are the primary source of confidentiality in white-box cryptography, therefore the adversary should not be able to infer x from $T(x)$. As given in Algorithm 1, E_k is a well-studied block cipher with a randomly selected key. We can yield the desired lookup table with the specific input and output lengths by padding the input with an all-zero binary value to achieve length extension and truncating the output of E_k, which is also the method used in [BI15].

Guidance Key Schedule. We employ an efficient method to schedule the guidance key, ensuring that the invocation of lookup tables in each round is unpredictable. The guidance key gk_i for the i-th round is derived by performing an XOR operation between the guidance key gk_{i-1} from the $(i-1)$-th round and the output X_i of the $(i-1)$-th round function. The hash function depicted in the figure is optional. Its role is to match the length of the state block with that of the guidance key. Specifically, it takes the output of the previous round function as input and outputs a sequence of the same length as the guidance key. Assuming the encryption process requires t rounds, then t guidance keys from gk_0 to gk_{t1} are needed. We refer to the first guidance key gk_0 as the encryption guidance key gk_{enc} and the last guidance key gk_{t-1} as the decryption guidance key gk_{dec}. When the message length exceeds one block, the encryption guidance key for each block (except the first one) is derived from the bit-wise XOR operation applied to the ciphertext and the decryption guidance key of the previous block. Moreover, we use the decryption guidance key of the last block as the decryption guidance key for the entire message.

Decryption Mode. When the round function is invertible, we trivially obtain the decryption mode as shown in Figure 4b, where the guidance keys are scheduled and applied in the opposite order to the encryption mode. Following the bottom-up order as shown in the figure, the round functions from F_{t-1} to F_0 are sequentially generated and utilized, transforming X_t back to X_0 and thereby completing a decryption process.

3.2 Application Protocol based on the KGI Mode

Figure 5 presents an application protocol based on the KGI mode, tailored to the typical scenario of white-box cryptography. In this context, we assume the presence of an encryptor $\mathcal{E}$ and a decryptor $\mathcal{D}$, and the objective of $\mathcal{E}$ is to securely transmit a message m to $\mathcal{D}$. The protocol is divided into two phases as follows: initialization and message transmission.

- ***Initialization.*** Prior to message transmission, an initialization phase is imperative. During this phase, $\mathcal{E}$ and $\mathcal{D}$ need to engage in negotiation and deployment to establish a shared LUT pool $\mathcal{P}$.

- ***Message Transmission.*** During this phase, $\mathcal{E}$ and $\mathcal{D}$ utilize the KGI mode to securely transmit the message m. The steps are as follows.

  1. $\mathcal{E}$ randomly generates the encryption guidance key $gk_{enc} \overset{\$}{\leftarrow} \{0,1\}^{|gk_{enc}|}$, where $\{0,1\}^{|gk_{enc}|}$ denotes the set of all binary strings of length $|gk_{enc}|$, i.e. the value space of the encryption guidance key.
  2. $\mathcal{E}$ evaluates $(gk_{dec}, c) \leftarrow \mathsf{Enc_{KGI}}(gk_{enc}, m, \mathcal{P})$ to generate the ciphertext c and the decryption guidance key gk_{dec} and sends them to $\mathcal{D}$. $\mathsf{Enc_{KGI}}$ represents a

white-box encryption program utilizing the KGI mode. It takes an encryption guidance key, a message, and a LUT pool as inputs.

3. Upon receiving (gk_{dec}, c), $\mathcal{D}$ executes the corresponding white-box decryption program $m \leftarrow \mathsf{Dec}_{\mathsf{KGI}}(gk_{dec}, c, \mathcal{P})$ to retrieve the original message m.

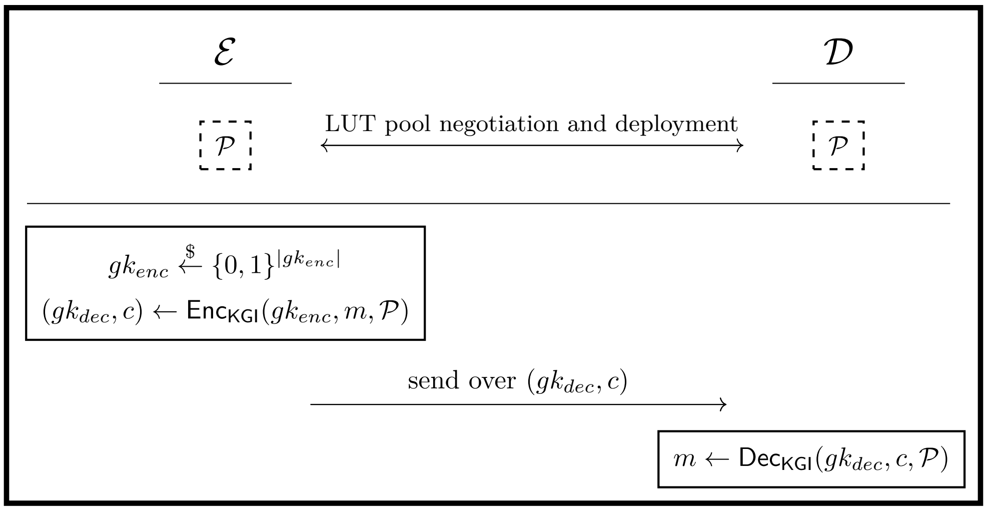

$\mathcal{E}$
$\mathcal{D}$
$\mathcal{P}$
LUT pool negotiation and deployment
$\mathcal{P}$
$gk_{enc} \xleftarrow{\$} \{0,1\}^{|gk_{enc}|}$
$(gk_{dec}, c) \leftarrow \mathsf{Enc}_{\mathsf{KGI}}(gk_{enc}, m, \mathcal{P})$
send over (gk_{dec}, c)
$m \leftarrow \mathsf{Dec}_{\mathsf{KGI}}(gk_{dec}, c, \mathcal{P})$

Figure 5: Application protocol based on the KGI mode

4 A Novel White-box Construction: RubikStone

Theoretically, all existing white-box ciphers can be adapted to the KGI mode after expanding the LUT pool. To better adapt to the KGI mode, we propose a novel white-box scheme, named RubikStone, which exhibits superior flexibility in the expansion of the LUT pool compared to existing white-box schemes.

4.1 Balanced Feistel Network

The Feistel [Fei73] network is a structure widely used in block ciphers [S+99, RRSY98, SKW+98]. According to whether the size of the two parts split from each round's input is equal, it can be divided into two categories, i.e. balanced Feistel network and unbalanced Feistel network. As shown in Figure 6, the n-bit state in r-th round X^r is split into two equally sized parts X_a^r and X_b^r. Let K_r be a k-bit key of the r-th round, and $F : \{0,1\}^{n/2} \times \{0,1\}^k \to \{0,1\}^{n/2}$ be an F function. A K_r-based F function is represented as F_{K_r}. The round function of the balanced Feistel network can be formalized as the following equation:

$$X_a^{r+1} \| X_b^{r+1} = X_b^r \| (F_{K_r}(X_b^r) \oplus X_a^r)$$

4.2 Design of RubikStone

RubikStone just employs a balanced Feistel network where the key-based F function is replaced by several table lookups. As shown in Figure 7, a n-bit plaintext X is encrypted to a ciphertext C by applying R-round permutations.

Let X^r denote the r-th round state. Specifically, the most significant $n/2$ bits of X^r is expressed as $l(= (n/2)/n_{in})$ elements of n_{in} bits, i.e. $X_a^r = \{x_0^r, x_1^r, \ldots, x_{l-1}^r\}$,

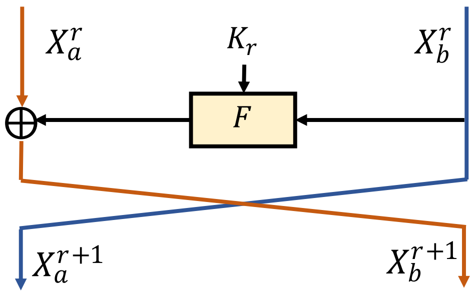


Figure 6: The r-th round of the balanced Feistel network

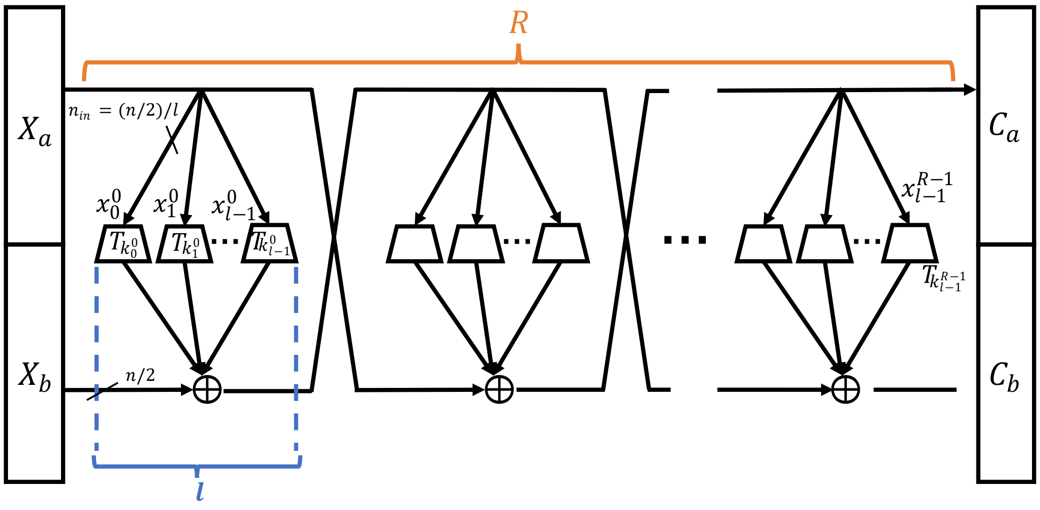


Figure 7: The RubikStone construction

$x_i^r \in \{0,1\}^{n_{in}}$ for $0 \le r \le R-1$, where R is the number of total rounds of RubikStone. All the elements $x_i^r(1 \le r \le R, 0 \le i \le l-1)$ are then transformed into $n/2$-bit strings through table lookups and ultimately XORed with X_b^r. The lookup table $T_{k_i^r}$ corresponding to each element x_i^r is selected from the LUT pool based on the index k_i^r, $0 \le k_i^r \le s-1$, where s is the number of total tables in the LUT pool. Trivially, we utilize $gk_r = k_0^r||k_1^r|| \cdot ||k_{l-1}^r$ as the guidance key for the r-th round.

Based on the construction of Rubikstone, we can obtain different white-box ciphers by adjusting the four parameters n, n_{in}, R and s. Thus, we use RubikStone-(n,n_{in},R,s) to uniquely represent different variants of RubikStone.

On one hand, the balanced Feistel network guarantees the reversibility of RubikStone irrespective of the specifications of the lookup tables utilized. On the other hand, the design of RubikStone incorporates additional parameters, providing it with a significantly expanded configuration space and enhanced flexibility in terms of LUT pool size. As a result, RubikStone demonstrates superior suitability for the KGI mode compared to existing dedicated white-box schemes.

5 Application Case and Performance

5.1 The Application Case

As a complete application case, we apply the KGI mode to a specific instantiation of Rubikstone, with $n = 256$, $n_{in} = 8$, $R = 12$, and $s = 2^{16}$. We refer to the case as

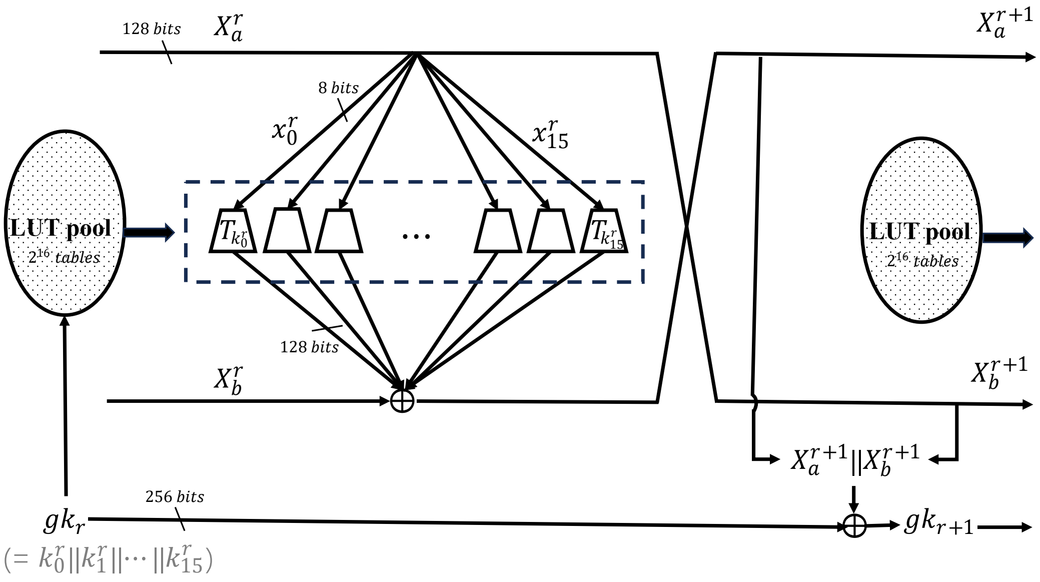


Figure 8: Round process of $\mathsf{RS}_{\mathsf{KGI}}$-256

RubikStone-(256,8,12,2^{16})-KGI and denote it as $\mathsf{RS}_{\mathsf{KGI}}$-256 for brevity. We generate the LUT pool using AES-128 as the underlying cipher according to Algorithm 1 and its size is $2^{16} \times 2^8 \times 256/2(bits) = 256(MB)$. Figure 8 illustrates the computation process of each round in $\mathsf{RS}_{\mathsf{KGI}}$-256. Algorithm 2 describes the whole process of encrypting a random message.

Algorithm 2: Encryption in $\mathsf{RS}_{\mathsf{KGI}}$-256

```
    Input: The message to be encrypted $M$.
    Output: The ciphertext $C$ and the decryption guidance key $gk_{dec}$.
 1  $\{M_0, M_1, \dots, M_s\} \leftarrow M$;        // $M$ is divided into several 256-bit blocks.
 2  $gk \xleftarrow{\$} \{0,1\}^{256}$;
 3  $\{M_a, M_b\} \leftarrow M_0$;
 4  for $i \leftarrow 0$ to $s$ do
 5      $\{M_a, M_b\} \leftarrow M_i$;
 6      for $j \leftarrow 0$ to 11 do
 7          $\{m_0, m_1, \dots, m_{15}\} \leftarrow M_a$;
 8          $\{g_0, g_1, \dots, g_{15}\} \leftarrow gk$;
 9          $M_c \leftarrow M_a$;
10          $M_a \leftarrow M_b \oplus T_{g_0}[m_0] \oplus T_{g_1}[m_1] \oplus \cdots \oplus T_{g_{15}}[m_{15}]$;
11          $M_b \leftarrow M_c$;
12          $g_k \leftarrow g_k \oplus (M_a||M_b)$;
13      $C_i \leftarrow (M_a||M_b)$;
14  $C \leftarrow \{C_0, C_1, \dots, C_s\}$;
15  $gk_{dec} \leftarrow g_k \oplus (M_a||M_b)$;
16  return $(gk_{dec}, C)$
```

5.2 Performance

To evaluate the performance of $\mathsf{RS}_{\mathsf{KGI}}$-256 in a real-world setting, all experiments were conducted on a system equipped with an Intel(R) Core(TM) i5-9500 CPU operating at 3.00 GHz and 16 GB of DDR4 RAM. The processor features a 384 KB L1 cache, a 1.5

MB L2 cache, and a 9 MB L3 cache, respectively. The implementation was carried out using the C programming language. To precisely measure the number of CPU clock cycles, GCC's inline assembly syntax was employed to invoke the "rdpmc" instruction as follows, which directly accesses the Performance Monitoring Counters (PMCs).

```
  __asm__ volatile ("rdpmc; shlq $32,%%rdx; orq %%rdx,%%rax"
  : "=a" (result) : "c" (ecx) : "rdx");
```

Table 1: Evaluation of encryption efficiency which is given in cycle per byte

Algorithm	**Table Size**	**Efficiency (cycle/byte)**
AES-128(Black-box)	-	539
AES-128(CEJO)[CEJvO02a]	752 KB	4027
$\mathsf{RS}_{\mathsf{KGI}}$-256	256 MB	2935

The experimental results are presented in Table 1. For comparative analysis, the efficiency of AES-128 and its white-box implementation based on the CEJO architecture [CEJvO02a] is measured under identical conditions. The results demonstrate that $\mathsf{RS}_{\mathsf{KGI}}$-256 achieves superior efficiency compared to the white-box implementation of AES-128, highlighting its significant potential in terms of performance.

It is important to note that our measured results appear substantially larger than those reported in [KSHI20, KI21]. However, despite using the same efficiency evaluation metrics, a direct comparison is not feasible. This discrepancy arises not only from differences in the performance of the hardware platforms used but also because the aforementioned studies prioritize implementation efficiency and incorporate numerous optimizations, such as leveraging instruction sets like AES-NI and SSE. In contrast, this work focuses on analyzing the algorithm itself and its security guarantees during operation, rather than exploring the limits of implementation efficiency. Consequently, our measurements were performed without optimization. However, by comparing the implementation efficiency of $\mathsf{RS}_{\mathsf{KGI}}$-256 with black-box and white-box AES implementations under the same conditions, the potential for efficient implementation of RubikStone is clearly demonstrated.

Due to the lack of open-source implementations for other dedicated white-box ciphers, a direct comparison with our implementation was not conducted. However, we have performed a comprehensive statistical analysis of the operations used in all existing dedicated white-box schemes, as shown in Table 2. In contrast, $\mathsf{RS}_{\mathsf{KGI}}$-256 utilizes fewer operations, comprising exclusively table lookups and bitwise XORs - both are computationally lightweight. This characteristic suggests that $\mathsf{RS}_{\mathsf{KGI}}$-256 holds substantial potential for optimization in both software and hardware implementations, which could be a future work.

6 Security Analysis

In this section, we present a thorough security analysis of the KGI mode and the RubikStone scheme. In most cases, we provide explicit security bounds of $\mathsf{RS}_{\mathsf{KGI}}$-256 as an illustrative example. Should readers utilize distinct instances of RubikStone, they are encouraged to consult our analytical methodology to determine the associated security boundaries relevant to their particular instantiation.

Specifically, we examine the diverse threats that a white-box scheme may encounter within black-box, gray-box, and white-box contexts, leading to the following conclusions.

Table 2: Operations of all white-box ciphers

Ciphers	**Calculations(per byte)**				
	L[i]	XOR[ii]	M[iii]	A[iv]	F[v]
SPACE-(8,300) [BI15]	18.75	2250	-	-	-
SPACE-(16,128) [BI15]	8	896	-	-	-
SPACE-(24,128) [BI15]	8	832	-	-	-
SPACE-(32,128) [BI15]	8	768	-	-	-
SPNbox-8 [BIT16]	10	80	$0.63(M_{16\times16})$	-	-
SPNbox-16 [BIT16]	5	80	$0.63(M_{8\times8})$	-	-
SPNbox-24 [BIT16]	3.33	80	$0.67(M_{5\times5})$	-	-
SPNbox-32 [BIT16]	2.50	80	$0.63(M_{4\times4})$	-	-
WhiteBlock 16[FKKM16]	4.50	288	-	1.13	-
WhiteBlock 20 [FKKM16]	4.31	276	-	1.44	-
WhiteBlock 24 [FKKM16]	4.25	272	-	2.13	-
WhiteBlock 28 [FKKM16]	4.25	272	-	2.13	-
WhiteBlock 32 [FKKM16]	4.25	272	-	2.13	-
WEM-128 [CCD+17]	7.5	-	-	0.38	-
Galaxy-8 [KSHI20]	12.50	100	-	-	-
Galaxy-16 [KSHI20]	5	80	-	-	-
Galaxy-32 [KSHI20]	4	128	-	-	-
FPL-(128,12,20,17) [KLLM20]	21.25	1360	-	-	1.06 $(F_{64\rightarrow240})$
FPL-(128,12,20,33) [KLLM20]	41.25	2640	-	-	2.06 $(F_{64\rightarrow240})$
FPL-(128,16,16,17) [KLLM20]	17	1088	-	-	1.06 $(F_{64\rightarrow256})$
FPL-(128,16,16,33) [KLLM20]	33	2112	-	-	2.06 $(F_{64\rightarrow256})$
FPL-(128,20,12,17) [KLLM20]	12.75	816	-	-	1.06 $(F_{64\rightarrow240})$
FPL-(128,20,12,33) [KLLM20]	24.75	1584	-		2.06 $(F_{64\rightarrow240})$
FPL-(64,8,16,9) [KLLM20]	18	576	-	-	1.13 $(F_{32\rightarrow128})$
FPL-(64,8,16,17) [KLLM20]	34	1088	-	-	2.13 $(F_{32\rightarrow128})$
FPL-(64,8,16,33) [KLLM20]	66	2112	-	-	4.13 $(F_{32\rightarrow128})$
FPL-(64,16,8,17) [KLLM20]	34	544	-	-	2.13 $(F_{32\rightarrow128})$
FPL-(64,16,8,33) [KLLM20]	66	1056	-	-	4.13 $(F_{32\rightarrow128})$
FPL-(64,16,16,17) [KLLM20]	34	1088	-	-	2.13 $(F_{32\rightarrow256})$
Yoroi-16 [KI21]	4	14	$1(M_{8\times8})$	0.06	-
Yoroi-32 [KI21]	4	30	$1(M_{4\times4})$	0.06	-
WAS [YZDZ23]	5	80	$0.63(M_{8\times8})$	-	-
$\mathsf{RS}_{\mathsf{KGI}}$-256	6	864	-	-	-

[i] L represents a table lookup.
[ii] XOR represents a bit XOR.
[iii] $M_{m\times m}$ represents a multiplication operation with a $m\times m$ MDS matrix.
[iv] A represents a 10-round AES.
[v] $F_{n_{in}\rightarrow n_{out}}$ represents a probe function.

1. $\mathsf{RS}_{\mathsf{KGI}}$-256 is secure against differential and linear cryptanalysis in the black-box context (see Section 6.1);
2. The underlying ciphers' keys utilized for generating the lookup tables are secure against key extraction (see Section 6.2);
3. The RubikStone construction is secure against advanced side-channel attacks, including differential computation analysis, linear decoding analysis, and differential fault attacks (see Section 6.3);
4. The RubikStone construction is capable of delivering robust security guarantees across the notions of WSH-KSAM, WSH-CSAM, WSH-ACSAM, SSH-KSAM, and SSH-CSAM. By integrating the KGI mode, the security guarantees for WSH-ACSAM are further strengthened, while simultaneously achieving, for the first time, security guarantees in SSH-ACSAM.(see Section 6.4).

6.1 Security in the Black-box Context

6.1.1 Differential Cryptanalysis

For a function $f(x) : \{0,1\}^{n_{in}} \rightarrow \{0,1\}^{n_{out}}$, the cardinality of a differential pair (a, b) is defined as the number of input pairs (x_1, x_2) that satisfy the input difference equation $x_1 \oplus x_2 = a$ and the output difference equation $f(x_1) \oplus f(x_2) = b$, denoted by $N(a, b)$. It has been proven in the Theorem 1 of [BI15] that for all non-trivial values of a and b, the probability q_B that $N(a, b)$ is at most B can be lower-bounded by the inequality:

$$q_B > (1 - 2 \cdot \frac{(2^{n_{in}-n_{out}-1})^{B+1}}{(B+1)!})^{2^{n_{in}+n_{out}}}$$

Based on the LUT pool generation algorithm in 1, each table used in $\mathsf{RS}_{\mathsf{KGI}}$-256 is essentially a black-box instance of AES-128. All the lookup tables in RubikStone-(256,8,12,2^{16}) can be treated as functions conforming to the specification $F_{8\rightarrow128} : \{0,1\}^8 \rightarrow \{0,1\}^{128}$. Table 3 shows the lower bounds on several q_Bs for $F_{8\rightarrow128}$. Given that q_2 is exceedingly close to 1, it follows that the maximum repetition count of differential pairs for $F_{8\rightarrow128}$ can be reasonably assumed to be 2. Consequently, we posit that the maximum differential probability of $F_{8\rightarrow128}$ is $2^{-7}(= 2/2^{-8})$. Our experimental results demonstrate that the 12-round $\mathsf{RS}_{\mathsf{KGI}}$-256 has at least 137 differentially active lookup tables, indicating an upper bound for its maximum differential probability of 2^{-959}.

Table 3: Lower-bound on q_B for $F_{8\rightarrow128}$

q_B	**Lower-bound on q_B**
q_1	$1 - 2^{-106}$
q_2	$1 - 2^{-228.58}$
q_3	$1 - 2^{-351.58}$
q_4	$1 - 2^{-474.9}$

6.1.2 Linear Cryptanalysis

For a function $f(x) : \{0,1\}^{n_{in}} \rightarrow \{0,1\}^{n_{out}}$, the correlation of a linear approximation (α, β) is defined by the equation 1, where $\alpha \in \{0,1\}^{n_{in}}$ is an input mask and $\beta \in \{0,1\}^{n_{out}}$ is an output mask.

$$Cor = 2^{-n_{in}} \cdot (|\{x \in \{0,1\}^{n_{in}} | \alpha \cdot x \oplus \beta \cdot f(x) = 0\}| - |\{x \in \{0,1\}^{n_{in}} | \alpha \cdot x \oplus \beta \cdot f(x) = 1\}|) \quad (1)$$

According to the Corollary 4.4 in [DR07], it can be assumed that the linear probability LP of a non-trivial linear approximation over n_{in}-bit to n_{out}-bit functions has mean $\mu(LP) = 2^{-n_{in}}$ and variance $\sigma^2(LP) \approx 2 \times 2^{-2n_{in}}$ when $n_{in} > 5$. Therefore, the linear probability LP of function $f(x)$ is lower than $2^{-n_{in}} + 10\sigma(\approx (10\sqrt{2}+1) \times 2^{-n_{in}})$ with probability $1 - 2^{-148}$. Then we assume the maximum linear probability of the tables used in RubikStone-(256,8,12,2^{16}) to be $2^{-4.08}$. Our experimental results demonstrate that the 12-round $\mathsf{RS}_{\mathsf{KGI}}$-256 has at least 119 linearly active lookup tables, indicating an upper bound for its maximum linear probability of $2^{-485.52}$.

6.2 Security against Key Extraction

As mentioned earlier, lookup tables are the primary source of condentiality in white-box cryptography. Since our algorithm for generating the LUT pool is public, if the adversary can extract the keys from the lookup tables, then the adversary no longer needs to exert effort to obtain the lookup tables. They can simply use the underlying ciphers to achieve the functionality of a white-box implementation on any device. Therefore, it is necessary to analyze the feasibility of the adversary successfully extracting these keys.

In the white-box context, the adversary can freely observe and intervene the execution process in the computation unit. With such ability, the adversary can easily obtain a large number of pairs of inputs and the corresponding outputs in lookup tables. This implies that adversaries can conduct any form of black-box attack on the lookup tables. Therefore, the adversary can extract the secret keys from lookup tables in white-box context as long as he can recover the secret keys from the underlying ciphers for the tables in black-box context. As a corollary, we reduce the security of lookup tables against the key extraction attack in the white-box context to the key recovery problem for the underlying ciphers in the black-box context, which is also the reduction method used in some existing dedicated white-box ciphers [BI15, BIT16, FKKM16, CCD+17, KLLM20, KI21].

In the application case $\mathsf{RS}_{\mathsf{KGI}}$-256, we utilize AES-128 as an instantiation of the underlying cipher in our LUT pool generation algorithm, for which no efficient key recovery attack has been proposed so far. Furthermore, in our design of the LUT pool generation algorithm, a different random key is used for each lookup table generation. This means that the adversary needs to crack at least $2^{23}(= 2^{16} \times 128)$-bit AES keys in the black-box context if he attempts to gain an advantage in transplanting the functionality of the decryption program by extracting keys from the lookup tables, which is not easier than accumulating all lookup table entries through computation leakage in the white-box context.

6.3 Security against Advanced Side Channel Attacks

6.3.1 Differential Computation Analysis

Differential Computation Analysis (DCA) attack was proposed by Bos *et al.* at CHES 2016 [BHMT16], serving as the software counterpart to differential power analysis (DPA) [KJJ99] attacks employed by the cryptographic hardware community. The main idea of DCA involves utilizing Dynamic Binary Instrumentation (DBI) frameworks such as Pin [LCM+05] and Valgrind [NS07] to acquire software traces. These traces encompass information like the physical addresses corresponding to memory read/write operations, and stack or register values during program execution, aiding attackers in determining the approximate location of the encryption algorithm within the software implementation and in conducting statistical analyses to extract the secret key. Employing this method, Bos *et al.* successfully extracted keys from various public white-box AES and DES implementations [CEJvO02a, CEJvO02b, LN05, Kar10, XL09] without knowledge of the encodings applied to intermediate results or which cipher operations are implemented by which lookup tables, and without resorting to reverse engineering of the binary files. This establishes DCA as a significant threat to white-box cryptographic implementations.

However, DCA is fundamentally an attack aimed at recovering a specific key. It may be highly effective against the white-box implementations of some existing block ciphers [CEJvO02a, BCD06, Kar10, XL09, LLY14, CEJvO02b, LN05, WP05], but is still ineffective against many dedicated white-box ciphers [BI15, BIT16, FKKM16, CCD+17, KSHI20, KLLM20, KI21, YZDZ23]. This is because these dedicated white-box ciphers are largely based on lookup tables generated from the overall inputs and outputs of a well-studied block cipher. Since the lookup tables are pre-generated, attackers cannot access any side-channel information produced during their creation, limiting them to black-box analysis of the tables. Therefore, when applying DCA to RubikStone, an attacker might recover the guidance key, which could assist in determining the encryption method corresponding to a particular ciphertext. However, without the ability to recover the underlying block cipher keys on which the lookup tables rely, the attacker gains no additional advantage over lifting the lookup tables for decrypting a specific ciphertext.

6.3.2 Algebraic Differential Computation Analysis

Linear decoding analysis (LDA) was first proposed by Goubin *et al.* in [GPRW20]. It was also called algebraic DCA by Biryukov and Udovenko in [BU18], which gradually evolved into an attack method that includes higher-order algebraic structures. This attack is designed to breach masking protection schemes by identifying algebraic combinations of some functions, thereby constructing a predictable sensitive function. With a sufficient number of computational traces, it can effectively pinpoint the location of shares after masking, thus circumventing the combinatorial explosion in complexity. In practical cases, algebraic DCA achieved remarkable success in the WhibOx contest 2017/2019 [PCY+17, GRW20]. On the basis of DCA, Algebraic DCA has improved its attack capability against some mask protection schemes. But algebraic DCA is still an attack method aimed at recovering a specific key, and therefore it cannot pose an effective threat to RubikStone.

6.3.3 Differential Fault Attack

Differential fault attack (DFA) targeting white-box ciphers was proposed by Sanfelix *et al.* in [SMdH15]. It modifies some specific bits by injecting faults into the white-box implementations and then conducts a differential analysis. This attack is also ineffective against RubikStone. Attackers cannot inject faults into the pre-generated lookup tables, and the internals of the underlying ciphers are inaccessible.

As mentioned in [YZDZ23], side-channel analysis exploits the fact that each lookup table relies only on a small portion of the key, which allows it to exhaustively enumerate all possibilities in segments, compute the correlation of traces, and thus guess the key. However, in RubikStone, all the lookup tables contain the full 128-bit of the keys. Therefore, even if an attacker can fully monitor the memory access patterns of the target key-related lookup tables, the amount of information the attacker must guess is 2^{128}.

Therefore, all forms of attacks targeting the lookup tables eventually reduce to black-box attacks against the underlying block cipher AES-128. If the adversary wishes to achieve the ultimate goal of decrypting a randomly drawn ciphertext on any device, the best strategy would be to lift all the lookup tables. The security of RubikStone against code lifting will be analyzed in Section 6.4.

6.4 Security against Code Lifting

In the white-box context, the adversary has full observational and control abilities over the computation unit. The only limitation hindering the adversary from transplanting the decryption functionality is their lack of the entire LUT pool. In fact, under the condition of applying a LUT pool, the number of lookup tables significantly exceeds the quantity likely to be utilized in a single encryption (or decryption) operation. The adversary must precisely determine the index of a lookup table where an input-output pair resides to effectively apply a specific table entry for attack. This presents an additional difficulty for the adversary in gathering lookup table entries compared to existing white-box schemes.

However, the adversary can still gather a significant number of lookup table entries through multiple analyses of the computation unit's execution process, which is just the process known as code lifting. To quantify the resilience against code lifting attacks, we evaluate the space hardness of the scheme under known-, chosen- and adaptively chosen-space attacks, to delineate the feasibility for the adversary to transplant decryption functionality under the condition of acquiring partial lookup table entries.

Furthermore, we separately evaluate the space hardness of RubikStone-(n,n_{in},R,s) in both the deterministic mode[1](see Section 6.4.1 and Section 6.4.2) and the KGI mode (see Section 6.4.3 and Section 6.4.4).

6.4.1 WSH in the Deterministic Mode

According to the discussion in [BIT16], we can obtain the following two theorems.

Theorem 1. *The probability that a randomly drawn plaintext (ciphertext) can be encrypted (decrypted) is upper bounded by* $(\frac{M}{s\cdot 2^{n_{in}}\cdot\frac{n}{2}})^{\frac{n}{2\cdot n_{in}}\cdot R}$ *given known or chosen space of size M from RubikStone-(n,n_{in},R,s).*

Proof. For a randomly drawn plaintext, it can be encrypted to the final ciphertext through $\frac{n}{2\cdot n_{in}}\cdot R$ table lookups. Because the inputs of tables are unpredictable in advance in both known- and chosen-space attacks, the probability for the adversary with a space of size M successfully locating the corresponding lookup table entry during each table lookup is given by $\frac{M}{s\cdot 2^{n_{in}}\cdot\frac{n}{2}}$, where $s\cdot 2^{n_{in}}\cdot\frac{n}{2}$ is the total size of all the lookup tables. To calculate the correct ciphertext, the adversary needs to possess exactly all the $\frac{n}{2\cdot n_{in}}\cdot R$ relevant table entries, so the probability is upper bounded by $(\frac{M}{s\cdot 2^{n_{in}}\cdot\frac{n}{2}})^{\frac{n}{2\cdot n_{in}}\cdot R}$. □

Theorem 2. *Given adaptively chosen space of size M from RubikStone-(n,n_{in},R,s), the probability that a randomly drawn plaintext (ciphertext) can be encrypted (decrypted) is upper bounded by* $\frac{N}{2^n}+(1-\frac{N}{2^n})\cdot(\frac{M}{s\cdot 2^{n_{in}}\cdot\frac{n}{2}})^{\frac{n}{2\cdot n_{in}}\cdot R}$*, where N satisfies the equation* $N=\lceil(log_{(\frac{2^{n_{in}\cdot s-1}}{2^{n_{in}\cdot s}})}(1-\frac{M}{s\cdot 2^{n_{in}}\cdot\frac{n}{2}}))/(\frac{n}{2\cdot n_{in}}\cdot R)\rceil$.

Proof. In the process of obtaining input-output pairs, the ACSAM adversary can choose an input after obtaining the outputs corresponding to the previous inputs. Moreover, in the deterministic mode, the manner in which lookup tables are invoked during the encryption process of a plaintext is also predictable. Exploiting the advantage, the adversary can ensure that a number of plaintexts' corresponding ciphertexts can be obtained with a probability of 100%. In other words, each lookup table entry required in the process of encrypting these plaintexts into the final ciphertexts is present in the part controlled by the adversary. Assuming the number of these plaintexts is N, the following equation can

[1]As in all existing dedicated white-box schemes, each lookup table invocation is predetermined in the deterministic mode. It can also be interpreted as the KGI mode with fixed guidance keys.

be obtained, where $2^{n_{in}} \cdot s$ is the number of all the entries in the LUT pool and $\frac{n}{2}$ is the size of each entry.

$$(1 - (\frac{2^{n_{in}} \cdot s - 1}{2^{n_{in}} \cdot s})^{N \cdot \frac{n}{2 \cdot n_{in}} \cdot R}) \cdot 2^{n_{in}} \cdot \frac{n}{2} \cdot s = M$$

Namely, the adversary can confidently know the ciphertexts corresponding to $N(= \lceil (log_{(\frac{2^{n_{in}} \cdot s-1}{2^{n_{in}} \cdot s})}(1 - \frac{M}{s \cdot 2^{n_{in}} \cdot \frac{n}{2}}))/(\frac{n}{2 \cdot n_{in}} \cdot R) \rceil)$ plaintexts with adaptively chosen space of size M. For a randomly drawn plaintext and a randomly drawn guidance key, the probability that it is included in these N plaintexts is $\frac{N}{2^n}$, where 2^n is the number of all the plaintexts. Otherwise, the probability that the adversary can successfully calculate the corresponding ciphertexts given space of size M is upper bounded by $(\frac{M}{s \cdot 2^{n_{in}} \cdot \frac{n}{2}})^{\frac{n}{2 \cdot n_{in}} \cdot R}$ from Theorem 1. The result in Theorem 2 can be obtained by adding the probabilities of the two parts. □

As a corollary, we finally obtain $(M, -log_2((\frac{M}{s \cdot 2^{n_{in}} \cdot \frac{n}{2}})^{\frac{n}{2 \cdot n_{in}} \cdot R}))$-WSH-KSAM/CSAM and $(M, -log_2(\frac{N}{2^n} + (1 - \frac{N}{2^n}) \cdot (\frac{M}{s \cdot 2^{n_{in}} \cdot \frac{n}{2}})^{\frac{n}{2 \cdot n_{in}} \cdot R}))$-WSH-ACSAM for Rubik-Stone-(n,n_{in},R,s) in the deterministic mode from Theorem 1 and Theorem 2 respectively.

6.4.2 SSH in the Deterministic Mode

The notion of SSH requires that the adversary cannot obtain a valid plaintext-ciphertext pair, which is obviously more strict than WSH where the adversary is not allowed to encrypt(decrypt) a randomly drawn plaintext(ciphertext).

According to Theorem 1, a randomly-drawn plaintext or ciphertext can be computed with the probability $(\frac{M}{s \cdot 2^{n_{in}} \cdot \frac{n}{2}})^{\frac{n}{2 \cdot n_{in}} \cdot R}$ or less given known or chosen space of size M. Then for 2^n plaintexts, the probability of at least one computable pair is upper bounded by $1 - (1 - (\frac{M}{s \cdot 2^{n_{in}} \cdot \frac{n}{2}})^{\frac{n}{2 \cdot n_{in}} \cdot R})^{2^n} \approx 2^n \cdot (\frac{M}{s \cdot 2^{n_{in}} \cdot \frac{n}{2}})^{\frac{n}{2 \cdot n_{in}} \cdot R}$.

Based on the analytical findings presented in Section 2.2.2, an ACSAM attacker can compromise SSH with a 100% success probability in the deterministic mode, as long as the number of the table entries that the attacker is allowed to access is more than the number required to encrypt a single block.

As a corollary, we obtain $(M, -log_2(2^n \cdot (\frac{M}{s \cdot 2^{n_{in}} \cdot \frac{n}{2}})^{\frac{n}{2 \cdot n_{in}} \cdot R}))$-SSH-KSAM/CSAM and $(M, 0)$-SSH-ACSAM for RubikStone-(n,n_{in},R,s) in the deterministic mode.

6.4.3 WSH in the KGI Mode

In contrast to the deterministic mode, the KGI mode dynamically determines the invocation pattern of lookup tables for encrypting a message. That is, the adversary gains knowledge of the lookup table invocation sequence for decrypting a ciphertext only during the challenge phase. Consequently, adversaries of the KSAM, CSAM, or ACSAM models are effectively restricted to performing random accesses during the query phase. Since KSAM/CSAM adversaries in the deterministic mode are also limited to performing random lookup table queries, the advantages of KSAM/CSAM adversaries are identical under both modes.

In the KGI mode, the ACSAM adversary cannot predict the guidance key value that will be used during the challenge phase. Consequently, the adversary lacks the ability to deterministically decrypt specific ciphertexts, as is possible for an ACSAM adversary in the deterministic mode. To maximize its advantage, the ACSAM adversary may attempt to guess potential guidance key values, thereby gaining the capability to deterministically decrypt certain ciphertexts under specific guidance key values. This leads us to the following theorem.

Theorem 3. *Given adaptively chosen space of size M from RubikStone-(n,n_{in},R,s) in the KGI mode, the probability that a randomly drawn ciphertext can be decrypted is upper* *bounded by* $\frac{N}{2^n}\cdot s^{-\frac{n}{2\cdot n_{in}}}+(1-\frac{N}{2^n}\cdot s^{-\frac{n}{2\cdot n_{in}}})\cdot(\frac{M}{s\cdot 2^{n_{in}}\cdot\frac{n}{2}})^{\frac{n}{2\cdot n_{in}}\cdot R}$*, where* N *satisfies the equation* $N=\lceil(log_{(\frac{2^{n_{in}}\cdot s-1}{2^{n_{in}}\cdot s})}(1-\frac{M}{s\cdot 2^{n_{in}}\cdot\frac{n}{2}}))/(\frac{n}{2\cdot n_{in}}\cdot R)\rceil$.

Proof. By guessing the decryption guidance key, the ACSAM adversary can deterministically decrypt a specific subset of ciphertexts. The size of this subset is precisely equal to N as defined in Theorem 2. During the challenge phase, if the ciphertext provided by the challenger falls within the subset, the adversary has a $2^{-log_2 s\cdot\frac{n}{2\cdot n_{in}}}=s^{-\frac{n}{2\cdot n_{in}}}$ probability of accurately guessing the corresponding decryption guidance key, thereby achieving decryption success with absolute certainty. Here, $log_2 s\cdot\frac{n}{2\cdot n_{in}}$ is just the length of the guidance key. As for ciphertexts outside this subset, the adversary's success probability in decryption is identical to that of a KSAM or CSAM adversary. Aggregating the probabilities from these two distinct cases yields the total success probability as $\frac{N}{2^n}\cdot s^{-\frac{n}{2\cdot n_{in}}}+(1-\frac{N}{2^n}\cdot s^{-\frac{n}{2\cdot n_{in}}})\cdot(\frac{M}{s\cdot 2^{n_{in}}\cdot\frac{n}{2}})^{\frac{n}{2\cdot n_{in}}\cdot R}$. □

As a corollary, we finally obtain $(M,-log_2((\frac{M}{s\cdot 2^{n_{in}}\cdot\frac{n}{2}})^{\frac{n}{2\cdot n_{in}}\cdot R}))$-WSH-KSAM/CSAM and $(M,-log_2(\frac{N}{2^n}\cdot s^{-\frac{n}{2\cdot n_{in}}}+(1-\frac{N}{2^n}\cdot s^{-\frac{n}{2\cdot n_{in}}})\cdot(\frac{M}{s\cdot 2^{n_{in}}\cdot\frac{n}{2}})^{\frac{n}{2\cdot n_{in}}\cdot R}))$-WSH-ACSAM for RubikStone-(n,n_{in},R,s) in the KGI mode.

6.4.4 SSH in the KGI Mode

AS mentioned in Section 6.4.3, the advantages of KSAM/CSAM adversaries are identical under both modes. Therefore, the analytical conclusions drawn in Section 6.4.2 can also be applied to the KSAM/CSAM adversaries in the KGI model. Specifically, given a known or chosen space of size M, the probability of an adversary obtaining a pair of plaintext-ciphertext is upper bounded by $2^n\cdot(\frac{M}{s\cdot 2^{n_{in}}\cdot\frac{n}{2}})^{\frac{n}{2\cdot n_{in}}\cdot R}$.

As delineated by Theorem 3, given adaptively chosen space of size M from RubikStone-(n,n_{in},R,s) in the KGI mode, the probability that a randomly drawn ciphertext can be decrypted is upper bounded by $\frac{N}{2^n}\cdot s^{-\frac{n}{2\cdot n_{in}}}+(1-\frac{N}{2^n}\cdot s^{-\frac{n}{2\cdot n_{in}}})\cdot(\frac{M}{s\cdot 2^{n_{in}}\cdot\frac{n}{2}})^{\frac{n}{2\cdot n_{in}}\cdot R}$. Then for 2^n plaintexts, the expected number of the decryptable ciphertexts is upper bounded by $2^n\cdot(\frac{N}{2^n\cdot s^{\frac{n}{2\cdot n_{in}}}}+(1-\frac{N}{2^n\cdot s^{\frac{n}{2\cdot n_{in}}}})\cdot(\frac{M}{s\cdot 2^{n_{in}}\cdot\frac{n}{2}})^{\frac{n}{2\cdot n_{in}}\cdot R})=\frac{N}{s^{\frac{n}{2\cdot n_{in}}}}+(2^n-\frac{N}{s^{\frac{n}{2\cdot n_{in}}}})\cdot(\frac{M}{s\cdot 2^{n_{in}}\cdot\frac{n}{2}})^{\frac{n}{2\cdot n_{in}}\cdot R}$.

As a corollary, we obtain $(M,-log_2(2^n\cdot(\frac{M}{s\cdot 2^{n_{in}}\cdot\frac{n}{2}})^{\frac{n}{2\cdot n_{in}}\cdot R}))$-SSH-KSAM/CSAM and $(M,-log_2(\frac{N}{s^{\frac{n}{2\cdot n_{in}}}}+(2^n-\frac{N}{s^{\frac{n}{2\cdot n_{in}}}})\cdot(\frac{M}{s\cdot 2^{n_{in}}\cdot\frac{n}{2}})^{\frac{n}{2\cdot n_{in}}\cdot R}))$-SSH-ACSAM for RubikStone-(n,n_{in},R,s) in the KGI mode.

6.4.5 Explicit Bounds on Examples

Let T denote the total size of all the lookup tables, i.e. $T=s\cdot 2^{n_{in}}\cdot\frac{n}{2}$. The space hardness of $M=T/4$ has received considerable attention from previous works [BI15, BIT16, CCD+17, KLLM20, KI21]. Based on the above evaluation of RubikStone-(n,n_{in},R,s), we derive the results shown in Table 4.

Table 4: Space hardness of RubikStone instantiations

Instantiations	Weak Space Hardness		Strong Space Hardness	
	KSAM/CSAM	ACSAM	KSAM/CSAM	ACSAM
RS-256 *	$(T/4,384)$	$(T/4,241.38)$	$(T/4,128)$	$(T/4,0)$
RS$_{\text{KGI}}$-256	$(T/4,384)$	$(T/4,383.99)$	$(T/4,128)$	$(T/4,127.99)$

* RS-256 denotes the instantiation of RubikStone-(256,8,12,2^{16}) in the deterministic mode.

The data presented in the table demonstrate that the application of the KGI model leads to significant enhancements in both the WSH-ACSAM and SSH-ACSAM notions, rendering them nearly equivalent to WSH-KSAM/CSAM and SSH-KSAM/CSAM. Notably, $\mathsf{RS}_{\mathsf{KGI}}$-256, for the first time, achieves security guarantees in SSH-ACSAM, surpassing all existing dedicated white-box schemes. This highlights the significant contribution of the KGI model to enhancing the security of white-box schemes. By incorporating guidance keys, the KGI mode strengthens the randomness of lookup table invocations, thereby substantially diminishing the advantages of ACSAM adversaries.

7 Conclusion

This paper proposes the KGI mode of operation, tailored for white-box cryptography, and introduces RubikStone, a dedicated white-box construction optimized for the KGI mode. A thorough security evaluation of the scheme was conducted, alongside a performance analysis using a complete application case $\mathsf{RS}_{\mathsf{KGI}}$-256. By employing the KGI mode, a unique encryption algorithm is assigned to each message, significantly enhancing security. The incorporation of a LUT pool further increases the adaptability of white-box implementations, allowing for tailored customization across diverse platforms with varying resources. Experimental findings indicate that our scheme not only delivers enhanced security but also maintains excellent performance, even outperforming white-box AES implementation, highlighting its practicality.

Nonetheless, the scheme has some limitations. Like other existing solutions, it is confined to privacy-only white-box applications, leaving the challenge of secure authenticated encryption in white-box contexts unaddressed. Moreover, while our scheme is well-suited for environments like cloud-based DRM, where encryption terminals are shielded from white-box attacks, it falls short in scenarios requiring authenticated encryption, such as mobile payments, necessitating further innovation. We look forward to future advancements that will address these challenges and yield superior results.

References

[BBF$^+$20] Estuardo Alpirez Bock, Chris Brzuska, Marc Fischlin, Christian Janson, and Wil Michiels. Security reductions for white-box key-storage in mobile payments. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part I*, volume 12491 of *Lecture Notes in Computer Science*, pages 221–252. Springer, 2020.

[BBK14] Alex Biryukov, Charles Bouillaguet, and Dmitry Khovratovich. Cryptographic schemes based on the ASASA structure: Black-box, white-box, and public-key (extended abstract). In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, volume 8873 of *Lecture Notes in Computer Science*, pages 63–84. Springer, 2014.

[BCD06] Julien Bringer, Hervé Chabanne, and Emmanuelle Dottax. White box cryptography: Another attempt. *IACR Cryptol. ePrint Arch.*, page 468, 2006.

[BGE04] Olivier Billet, Henri Gilbert, and Charaf Ech-Chatbi. Cryptanalysis of a white box AES implementation. In Helena Handschuh and M. Anwar Hasan, editors, *Selected Areas in Cryptography, 11th International Workshop, SAC 2004, Waterloo, Canada, August 9-10, 2004, Revised Selected Papers*, volume 3357 of *Lecture Notes in Computer Science*, pages 227–240. Springer, 2004.

[BHMT16] Joppe W. Bos, Charles Hubain, Wil Michiels, and Philippe Teuwen. Differential computation analysis: Hiding your white-box designs is not enough. In Benedikt Gierlichs and Axel Y. Poschmann, editors, *Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings*, volume 9813 of *Lecture Notes in Computer Science*, pages 215–236. Springer, 2016.

[BI15] Andrey Bogdanov and Takanori Isobe. White-box cryptography revisited: Space-hard ciphers. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*, pages 1058–1069. ACM, 2015.

[BIT16] Andrey Bogdanov, Takanori Isobe, and Elmar Tischhauser. Towards practical whitebox cryptography: Optimizing efficiency and space hardness. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I*, volume 10031 of *Lecture Notes in Computer Science*, pages 126–158, 2016.

[BMW+18] Jo Van Bulck, Marina Minkin, Ofir Weisse, Daniel Genkin, Baris Kasikci, Frank Piessens, Mark Silberstein, Thomas F. Wenisch, Yuval Yarom, and Raoul Strackx. Foreshadow: Extracting the keys to the intel SGX kingdom with transient out-of-order execution. In William Enck and Adrienne Porter Felt, editors, *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018*, pages 991–1008. USENIX Association, 2018.

[BPS17] Jo Van Bulck, Frank Piessens, and Raoul Strackx. Sgx-step: A practical attack framework for precise enclave execution control. In *Proceedings of the 2nd Workshop on System Software for Trusted Execution, SysTEX@SOSP 2017, Shanghai, China, October 28, 2017*, pages 4:1–4:6. ACM, 2017.

[BU18] Alex Biryukov and Aleksei Udovenko. Attacks and countermeasures for white-box designs. In Thomas Peyrin and Steven D. Galbraith, editors, *Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part II*, volume 11273 of *Lecture Notes in Computer Science*, pages 373–402. Springer, 2018.

[CCD+17] Jihoon Cho, Kyu Young Choi, Itai Dinur, Orr Dunkelman, Nathan Keller, Dukjae Moon, and Aviya Veidberg. WEM: A new family of white-box block ciphers based on the even-mansour construction. In Helena Handschuh, editor, *Topics in Cryptology - CT-RSA 2017 - The Cryptographers' Track at the RSA Conference 2017, San Francisco, CA, USA, February 14-17, 2017, Proceedings*, volume 10159 of *Lecture Notes in Computer Science*, pages 293–308. Springer, 2017.

[CEJvO02a] Stanley Chow, Philip A. Eisen, Harold Johnson, and Paul C. van Oorschot. White-box cryptography and an AES implementation. In Kaisa Nyberg and

Howard M. Heys, editors, *Selected Areas in Cryptography, 9th Annual International Workshop, SAC 2002, St. John's, Newfoundland, Canada, August 15-16, 2002. Revised Papers*, volume 2595 of *Lecture Notes in Computer Science*, pages 250–270. Springer, 2002.

[CEJvO02b] Stanley Chow, Philip A. Eisen, Harold Johnson, and Paul C. van Oorschot. A white-box DES implementation for DRM applications. In Joan Feigenbaum, editor, *Security and Privacy in Digital Rights Management, ACM CCS-9 Workshop, DRM 2002, Washington, DC, USA, November 18, 2002, Revised Papers*, volume 2696 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2002.

[DLPR13] Cécile Delerablée, Tancrède Lepoint, Pascal Paillier, and Matthieu Rivain. White-box security notions for symmetric encryption schemes. In Tanja Lange, Kristin E. Lauter, and Petr Lisonek, editors, *Selected Areas in Cryptography - SAC 2013 - 20th International Conference, Burnaby, BC, Canada, August 14-16, 2013, Revised Selected Papers*, volume 8282 of *Lecture Notes in Computer Science*, pages 247–264. Springer, 2013.

[DR07] Joan Daemen and Vincent Rijmen. Probability distributions of correlation and differentials in block ciphers. *J. Math. Cryptol.*, 1(3):221–242, 2007.

[Fei73] Horst Feistel. Cryptography and computer privacy. *Scientific American*, 228(5):15–23, 1973.

[FKKM16] Pierre-Alain Fouque, Pierre Karpman, Paul Kirchner, and Brice Minaud. Efficient and provable white-box primitives. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I*, volume 10031 of *Lecture Notes in Computer Science*, pages 159–188, 2016.

[GMQ07] Louis Goubin, Jean-Michel Masereel, and Michaël Quisquater. Cryptanalysis of white box DES implementations. In Carlisle M. Adams, Ali Miri, and Michael J. Wiener, editors, *Selected Areas in Cryptography, 14th International Workshop, SAC 2007, Ottawa, Canada, August 16-17, 2007, Revised Selected Papers*, volume 4876 of *Lecture Notes in Computer Science*, pages 278–295. Springer, 2007.

[GPRW20] Louis Goubin, Pascal Paillier, Matthieu Rivain, and Junwei Wang. How to reveal the secrets of an obscure white-box implementation. *J. Cryptogr. Eng.*, 10(1):49–66, 2020.

[GRW20] Louis Goubin, Matthieu Rivain, and Junwei Wang. Defeating state-of-the-art white-box countermeasures with advanced gray-box attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 454–482, 2020.

[Inc14] Adobe Systems Incorporated. *Adobe Primetime Technical Primer for Operators.* Adobe Systems Incorporated, 2014.

[JBF02] Matthias Jacob, Dan Boneh, and Edward W. Felten. Attacking an obfuscated cipher by injecting faults. In Joan Feigenbaum, editor, *Security and Privacy in Digital Rights Management, ACM CCS-9 Workshop, DRM 2002, Washington, DC, USA, November 18, 2002, Revised Papers*, volume 2696 of *Lecture Notes in Computer Science*, pages 16–31. Springer, 2002.

[Kar10] Mohamed Karroumi. Protecting white-box AES with dual ciphers. In Kyung Hyune Rhee and DaeHun Nyang, editors, *Information Security and Cryptology - ICISC 2010 - 13th International Conference, Seoul, Korea, December 1-3, 2010, Revised Selected Papers*, volume 6829 of *Lecture Notes in Computer Science*, pages 278–291. Springer, 2010.

[KI21] Yuji Koike and Takanori Isobe. Yoroi: Updatable whitebox cryptography. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(4):587–617, 2021.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

[KLLM20] Jihoon Kwon, ByeongHak Lee, Jooyoung Lee, and Dukjae Moon. FPL: white-box secure block cipher using parallel table look-ups. In Stanislaw Jarecki, editor, *Topics in Cryptology - CT-RSA 2020 - The Cryptographers' Track at the RSA Conference 2020, San Francisco, CA, USA, February 24-28, 2020, Proceedings*, volume 12006 of *Lecture Notes in Computer Science*, pages 106–128. Springer, 2020.

[KSHI20] Yuji Koike, Kosei Sakamoto, Takuya Hayashi, and Takanori Isobe. Galaxy: A family of stream-cipher-based space-hard ciphers. In Joseph K. Liu and Hui Cui, editors, *Information Security and Privacy - 25th Australasian Conference, ACISP 2020, Perth, WA, Australia, November 30 - December 2, 2020, Proceedings*, volume 12248 of *Lecture Notes in Computer Science*, pages 142–159. Springer, 2020.

[LCM$^+$05] Chi-Keung Luk, Robert S. Cohn, Robert Muth, Harish Patil, Artur Klauser, P. Geoffrey Lowney, Steven Wallace, Vijay Janapa Reddi, and Kim M. Hazelwood. Pin: building customized program analysis tools with dynamic instrumentation. In Vivek Sarkar and Mary W. Hall, editors, *Proceedings of the ACM SIGPLAN 2005 Conference on Programming Language Design and Implementation, Chicago, IL, USA, June 12-15, 2005*, pages 190–200. ACM, 2005.

[LLY14] Rui Luo, Xuejia Lai, and Rong You. A new attempt of white-box AES implementation. In *Proceedings IEEE International Conference on Security, Pattern Analysis, and Cybernetics, SPAC 2014, Wuhan, China, October 18-19, 2014*, pages 423–429. IEEE, 2014.

[LN05] Hamilton E. Link and William D. Neumann. Clarifying obfuscation: Improving the security of white-box DES. In *International Symposium on Information Technology: Coding and Computing (ITCC 2005), Volume 1, 4-6 April 2005, Las Vegas, Nevada, USA*, pages 679–684. IEEE Computer Society, 2005.

[LPSS16] Hyejoo Lee, Suwan Park, Changho Seo, and Sang-Uk Shin. DRM cloud framework to support heterogeneous digital rights management systems. *Multim. Tools Appl.*, 75(22):14089–14109, 2016.

[LRM$^+$13] Tancrède Lepoint, Matthieu Rivain, Yoni De Mulder, Peter Roelse, and Bart Preneel. Two attacks on a white-box AES implementation. In Tanja Lange, Kristin E. Lauter, and Petr Lisonek, editors, *Selected Areas in Cryptography*

- SAC 2013 - 20th International Conference, Burnaby, BC, Canada, August 14-16, 2013, Revised Selected Papers, volume 8282 of *Lecture Notes in Computer Science*, pages 265–285. Springer, 2013.

[MGH08] Wil Michiels, Paul Gorissen, and Henk D. L. Hollmann. Cryptanalysis of a generic class of white-box implementations. In Roberto Maria Avanzi, Liam Keliher, and Francesco Sica, editors, *Selected Areas in Cryptography, 15th International Workshop, SAC 2008, Sackville, New Brunswick, Canada, August 14-15, Revised Selected Papers*, volume 5381 of *Lecture Notes in Computer Science*, pages 414–428. Springer, 2008.

[MIE17] Ahmad Moghimi, Gorka Irazoqui, and Thomas Eisenbarth. Cachezoom: How SGX amplifies the power of cache attacks. In Wieland Fischer and Naofumi Homma, editors, *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, volume 10529 of *Lecture Notes in Computer Science*, pages 69–90. Springer, 2017.

[MOO$^+$14] Ciara Moore, Máire O'Neill, Elizabeth O'Sullivan, Yarkin Doröz, and Berk Sunar. Practical homomorphic encryption: A survey. In *IEEE International Symposium on Circuits and Systemss, ISCAS 2014, Melbourne, Victoria, Australia, June 1-5, 2014*, pages 2792–2795. IEEE, 2014.

[MRP12] Yoni De Mulder, Peter Roelse, and Bart Preneel. Cryptanalysis of the xiao - lai white-box AES implementation. In Lars R. Knudsen and Huapeng Wu, editors, *Selected Areas in Cryptography, 19th International Conference, SAC 2012, Windsor, ON, Canada, August 15-16, 2012, Revised Selected Papers*, volume 7707 of *Lecture Notes in Computer Science*, pages 34–49. Springer, 2012.

[Mul14] Yoni De Mulder. *White-Box Cryptography: Analysis of White-Box AES Implementations (White-Box Cryptografie: Analyse van White-Box AES implementaties)*. PhD thesis, Katholieke Universiteit Leuven, Belgium, 2014.

[MWP10] Yoni De Mulder, Brecht Wyseur, and Bart Preneel. Cryptanalysis of a perturbated white-box AES implementation. In Guang Gong and Kishan Chand Gupta, editors, *Progress in Cryptology - INDOCRYPT 2010 - 11th International Conference on Cryptology in India, Hyderabad, India, December 12-15, 2010. Proceedings*, volume 6498 of *Lecture Notes in Computer Science*, pages 292–310. Springer, 2010.

[NS07] Nicholas Nethercote and Julian Seward. Valgrind: a framework for heavyweight dynamic binary instrumentation. In Jeanne Ferrante and Kathryn S. McKinley, editors, *Proceedings of the ACM SIGPLAN 2007 Conference on Programming Language Design and Implementation, San Diego, California, USA, June 10-13, 2007*, pages 89–100. ACM, 2007.

[Pay] EMV-Mobile Payment. Software-based mobile payment security requirements.

[PCY$^+$17] Emmanuel Prouff, Chen-Mou Cheng, Bo-Yin Yang, Thomas Baignères, Matthieu Finiasz, Pascal Paillier, and Matthieu Rivain. Ches 2017 capture the flag challenge-the whibox contest, an ecrypt white-box cryptography competition, 2017.

[RP20] Adrián Ranea and Bart Preneel. On self-equivalence encodings in white-box implementations. In Orr Dunkelman, Michael J. Jacobson Jr., and Colin O'Flynn, editors, *Selected Areas in Cryptography - SAC 2020 - 27th International Conference, Halifax, NS, Canada (Virtual Event), October 21-23, 2020, Revised Selected Papers*, volume 12804 of *Lecture Notes in Computer Science*, pages 639–669. Springer, 2020.

[RRSY98] Ronald L Rivest, Matthew JB Robshaw, Ray Sidney, and Yiqun Lisa Yin. The rc6tm block cipher. In *First advanced encryption standard (AES) conference*, page 16, 1998.

[RVP22] Adrián Ranea, Joachim Vandersmissen, and Bart Preneel. Implicit white-box implementations: White-boxing ARX ciphers. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part I*, volume 13507 of *Lecture Notes in Computer Science*, pages 33–63. Springer, 2022.

[S$^+$99] Data Encryption Standard et al. Data encryption standard. *Federal Information Processing Standards Publication*, 112:3, 1999.

[SKW$^+$98] Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, and Niels Ferguson. Twofish: A 128-bit block cipher. *NIST AES Proposal*, 15(1):23–91, 1998.

[SMdH15] Eloi Sanfelix, Cristofaro Mune, and Job de Haas. Unboxing the white-box practical attacks against obfuscated ciphers. *Black Hat Europe*, 2015, 2015.

[WMGP07] Brecht Wyseur, Wil Michiels, Paul Gorissen, and Bart Preneel. Cryptanalysis of white-box DES implementations with arbitrary external encodings. In Carlisle M. Adams, Ali Miri, and Michael J. Wiener, editors, *Selected Areas in Cryptography, 14th International Workshop, SAC 2007, Ottawa, Canada, August 16-17, 2007, Revised Selected Papers*, volume 4876 of *Lecture Notes in Computer Science*, pages 264–277. Springer, 2007.

[WP05] Brecht Wyseur and Bart Preneel. Condensed white-box implementations. In *Proceedings of the 26th Symposium on Information Theory in the Benelux*, pages 296–301. Werkgemeenschap voor Informatie-en Communicatietheorie, 2005.

[XL09] Yaying Xiao and Xuejia Lai. A secure implementation of white-box aes. In *2009 2nd International Conference on Computer Science and its Applications*, pages 1–6, 2009.

[YZDZ23] Yatao Yang, Yuying Zhai, Hui Dong, and Yanshuo Zhang. WAS: improved white-box cryptographic algorithm over AS iteration. *Cybersecur.*, 6(1):56, 2023.