


Article

A White-Box Implementation of IDEA

Siyu Pang ¹ , Tingting Lin ^{2,*}, Xuejia Lai ^{1,3,4,*} and Zheng Gong ⁵

¹ Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China; p_cloudy@sjtu.edu.cn

² School of Cyber Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China

³ State Key Laboratory of Cryptology, P.O. Box 5159, Beijing 100878, China

⁴ Westone Cryptologic Research Center, Beijing 100070, China

⁵ School of Computer Science, South China Normal University, Guangzhou 510631, China; gongzheng@scnu.edu.cn

* Correspondence: lintingting@sjtu.edu.cn (T.L.); laix@sjtu.edu.cn (X.L.)

Abstract: IDEA is a classic symmetric encryption algorithm proposed in 1991 and widely used in many applications. However, there is little research into white-box IDEA. In traditional white-box implementations of existing block ciphers, S-boxes are always converted into encoded lookup tables. However, the algebraic operations of IDEA without S-boxes, make the implementation not straight forward and challenging. We propose a white-box implementation of IDEA by applying a splitting symmetric encryption method, and verify its security against algebraic analysis and BGE-like attacks. Our white-box implementation requires an average of about 2800 ms to encrypt a 64-bit plaintext, about 60 times more than the original algorithm would take, which is acceptable for practical applications. Its storage requirements are only about 10 MB. To our knowledge, this is the first public white-box IDEA solution, and its design by splitting can be applied to similar algebraic encryption structures.

Keywords: cryptanalysis; IDEA; white-box cryptography; obfuscation



Citation: Pang, S.; Lin, T.; Lai, X.; Gong, Z. A White-Box Implementation of IDEA. *Symmetry* **2021**, *13*, 1066. <https://doi.org/10.3390/sym13061066>

Academic Editor: Kuo-Hui Yeh

Received: 26 May 2021
Accepted: 12 June 2021
Published: 15 June 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Among many classic symmetric encryption algorithms, the International Data Encryption Algorithm (IDEA) was jointly developed and proposed [1–3] in 1991. The main encryption process of IDEA includes group operations, a multiplication and addition (MA) structure, and output transformations. Its core operations are modular multiplication, modular addition, and XOR. In addition, the Feistel or SPN structure always uses the S-box as the core structure, while IDEA incorporates three different algebraic operations to replace the traditional S-box non-linear function. Due to its special structure, IDEA differs from these symmetric ciphers and is often selected as an encryption primitive in products such as VLSI and SMS messages [4–6].

In 2002, the concepts of a black-box model, gray-box model, and white-box model were proposed by Chow et al. in conjunction with the white-box implementation of AES [7] and DES [8]. As the color lightens, the attacker is given more powerful capabilities. In the white-box model, an attacker can fully monitor and track the contents of memory and cache during dynamic and static execution [9] (for fixed keys), modify the internal operations and memory contents of the algorithm at any time point [10], and so on. Traditional implementations of block ciphers are insecure in these scenarios, because the key simply added (XOR-ed) in the round function will be exposed directly.

More seriously, if an attacker can perform the white-box attacks on legitimate terminal devices securing information such as SMS messages, all of the high-value messages will be obtainable by the attacker. This is a critically serious security risk to both symmetric encryption and asymmetric encryption. White-box cryptography is a solution for preserving the same security level as in the black-box model, with practical significance in applications

such as wireless sensor networks, mobile agents, and digital rights management [11–16]. Many companies have been working to use white-box encryption solutions, such as the secure chip of a downloadable conditional access system (DCAS) terminal [17] and white-Cryption secure key box (SKB). The latter is the first and only enterprise-ready white-box cryptography solution for web applications, launched by Intertrust company at the RSA Conference, on 24 February 2020.

In the field of white-box research, especially based on symmetric encryption, much work has been done in the design of white-box implementations [18–22], mostly concentrating on DES, AES, SM4, and their variants, which provide a certain degree of protection, but are still breakable [23–28]. Since the only non-linear component of the above block ciphers is S-box, a common approach of these white-box schemes is to hide the S-box in lookup tables and use random bijection to confuse inputs and outputs of the lookup tables to protect secret information (the key). As we mentioned before, IDEA has a variety of non-linear operational modes that replace the S-box component. Therefore, the key will be included in multiple non-linear operations. This special behavior brings new challenges to the design of white-box implementations.

In 2019, Lu et al. [29] proposed white-box KMAC, converting modular operations into lookup tables. Since the size of its lookup tables are relatively large and the algebraic operations are different from IDEA, it is not directly usable for implementing a white-box IDEA while preserving IDEA's throughput. There are no other solutions for transforming algebraic operations into lookup tables, and, as far as we know, no white-box IDEA implementation has been proposed. The research into this topic is scant.

In this paper, we observe the algebraic structure of IDEA's group operations and reduce the storage cost of the lookup tables by splitting the plaintext blocks into reasonably sized chunks. We split every arithmetic operation, multiplication modulo $(2^{16} + 1)$ and addition modulo 2^{16} , into two parts and use an affine transformation to obscure them. A Type I lookup table is created for each part, such that effective analysis is impossible. The outputs of the Type I lookup tables will be added with corresponding module. A Type II lookup table is generated by encoding the additional module results with a randomly chosen mask to eliminate leakage of key related information from unbalanced encodings. We reorganize the MA structure as four arithmetic operations, with each operation processed similarly as before. We then create a working white-box IDEA solution with adequate performance. The total storage is 10.06 MB with an average encryption time of 2786 ms (based on 50,000 encryptions of 64-bit plaintext using Java 8.0), which is about 60 times as much time as the original IDEA. Compared with other white-box implementations in Table 1, our solution offers satisfactory efficiency. The implementation methods that split the blocks with encoding and masks are also usable with other cryptographic primitives using algebraic operations.

Table 1. Efficiency comparison.

Scheme	Operations ¹	Delay Cost ²	Storage
white-box KMAC	32,920	375	107.7 MB
white-box AES	3104	55	580 KB
white-box IDEA	264	60	10.06 MB

¹ The operations include: lookups, XORs, and other algebraic operations. ² Times slower, compared with the cost in the black-box model (the plain algorithm).

The structure of the rest of the paper is as follows. In Section 2, we review the IDEA cipher along with encoding and masking techniques. We present our white-box implementation in Section 3. In Section 4, we present a performance analysis. We measure and analyze the security of the white-box IDEA in Section 5. Section 6 contains an additional discussion and our conclusions.

2. Preliminaries

2.1. Notation

We use the following notations in this paper.

- \odot multiplication mod $(2^{16} + 1)$ of 16-bit integers
- \boxplus addition mod 2^{16} of 16-bit integers. During arithmetic operations of IDEA, the all-zero value is never used and is replaced by $(2^{16} + 1)$ or 2^{16} .
- \otimes multiplication mod 2^{16} of 16-bit integers
- \oplus bitwise exclusive OR (XOR of 16-bit strings)
- $\times, -, +$ arithmetic operation
- \parallel bit string concatenation
- \circ functional composition

2.2. Description of IDEA

We now describe the encryption process of IDEA. The fixed size 64-bit plaintext P is divided into four 16-bit blocks $P = (P_1 \parallel P_2 \parallel P_3 \parallel P_4)$. The entire algorithm includes eight rounds of encryption operations plus the output transformations, with each encryption round selecting six sub-keys $Z_i^{(r)}, i = 1, 2, \dots, 6, r = 1, 2, \dots, 9$. The 128-bit initial master key is divided into eight 16-bit sub-keys from left to right, for 52 total sub-keys (eight rounds \times six sub-keys + four sub-keys), generated by left shifting by 25 positions. $X_j^{(r)}$ and $Y_j^{(r)}, j = 1, 2, 3, \dots$, denote the intermediate values, with the final ciphertext being $C = (C_1 \parallel C_2 \parallel C_3 \parallel C_4)$.

The eight encryption rounds are

$$\left\{ \begin{array}{l} \text{group operations} \left\{ \begin{array}{l} Y_1^{(r)} = P_1^{(r)} \odot Z_1^{(r)} \\ Y_2^{(r)} = P_2^{(r)} \boxplus Z_2^{(r)} \\ Y_3^{(r)} = P_3^{(r)} \boxplus Z_3^{(r)} \\ Y_4^{(r)} = P_4^{(r)} \odot Z_4^{(r)} \end{array} \right. \\ \text{XOR operations} \left\{ \begin{array}{l} Y_5^{(r)} = Y_1^{(r)} \oplus Y_3^{(r)} \\ Y_6^{(r)} = Y_2^{(r)} \oplus Y_4^{(r)} \end{array} \right. \\ \text{MA structure} \left\{ \begin{array}{l} Y_7^{(r)} = Y_5^{(r)} \odot Z_5^{(r)} \\ Y_8^{(r)} = Y_7^{(r)} \boxplus Y_6^{(r)} \\ Y_9^{(r)} = Y_8^{(r)} \odot Z_6^{(r)} \\ Y_{10}^{(r)} = Y_9^{(r)} \boxplus Y_7^{(r)} \end{array} \right. \\ \text{XOR operations} \left\{ \begin{array}{l} Y_{11}^{(r)} = Y_1^{(r)} \oplus Y_9^{(r)} \\ Y_{12}^{(r)} = Y_3^{(r)} \oplus Y_9^{(r)} \\ Y_{13}^{(r)} = Y_2^{(r)} \oplus Y_{10}^{(r)} \\ Y_{14}^{(r)} = Y_4^{(r)} \oplus Y_{10}^{(r)} \end{array} \right. \end{array} \right. .$$

The output transformations are

$$\left\{ \begin{array}{l} C_1 = Y_{11}^{(8)} \odot Z_1^{(9)}, C_2 = Y_{13}^{(8)} \boxplus Z_2^{(9)} \\ C_3 = Y_{12}^{(8)} \boxplus Z_3^{(9)}, C_4 = Y_{14}^{(8)} \odot Z_4^{(9)} \end{array} \right. .$$

In the group operations, the arithmetic operations, integer multiplication modulo $2^{16} + 1$ (1) and integer addition modulo 2^{16} (2), are the core operations in IDEA:

$$X_j^{(r)} (or Y_j^{(r)}) \odot Z_i^{(r)} = Y_j^{(r)}, \quad (1)$$

$$X_j^{(r)} (or Y_j^{(r)}) \boxplus Z_i^{(r)} (or Y_j^{(r)}) = Y_j^{(r)}. \quad (2)$$

For integer multiplication modulo $2^{16} + 1$, direct calculation is expensive. We can use a low-high algorithm to compute integer multiplication, for example:

$$X_j^{(r)} \odot Z_i^{(r)} = \begin{cases} (X_j^{(r)} \times Z_i^{(r)} \bmod 2^{16}) - (X_j^{(r)} \times Z_i^{(r)} \div 2^{16}), \\ \text{if } (X_j^{(r)} \times Z_i^{(r)} \bmod 2^{16}) \geq (X_j^{(r)} \times Z_i^{(r)} \div 2^{16}); \\ (X_j^{(r)} \times Z_i^{(r)} \bmod 2^{16}) - (X_j^{(r)} \times Z_i^{(r)} \div 2^{16}) + (2^{16} + 1), \\ \text{if } (X_j^{(r)} \times Z_i^{(r)} \bmod 2^{16}) \leq (X_j^{(r)} \times Z_i^{(r)} \div 2^{16}). \end{cases}$$

Thus, the low-high algorithm is available for most CPU, and invertible for the Fermat primes. The two arithmetic operations with XOR operations were added for both confusion and diffusion. The three operations on 16-bit blocks are incompatible, and they are non-associative, non-distributive, or non-isotopic.

2.3. White-Box Cryptography Techniques and Terminology

White-box cryptography is an obfuscation technique that hides the information about the key in a cryptography system. In recent developments, multi-linear mapping technology [30] and masking technology [31] have provided meaningful security guarantees.

2.3.1. Internal Encodings

A popular method for handling a fixed key is to embed it in lookup tables encoded with random bijection [7,8]. Randomness of the bijection makes it difficult for the adversary to recover the key from the encoding lookup tables.

1. Encoding: X is the transformation, from m -bit to n -bit:

$$E(X) = G \circ X \circ F,$$

where F is a randomly selected m -bit to m -bit bijection called the input encoding, and G is a randomly selected n -bit to n -bit bijection called the output encoding. $E(X)$ is called an encoded transformation. The main purpose of constructing $E(X)$ is to obfuscate the input and output of X .

2. Networked encoding: a networked encoding of the compound transformation $Y \circ X$ (Y transformation after X transformation) is

$$Y' \circ X' = (N \circ Y \circ M^{-1}) \circ (M \circ X \circ H^{-1}) = N \circ (Y \circ X) \circ H^{-1},$$

where N , M , and H are all bijections. The networked encoding confuses the input and output of X and Y and ensures that all the transformations are combined to be functionally equivalent to the original transformation.

2.3.2. External Encodings

An adversary's extraction of the entire solution to another device is equivalent to possessing the encryption function of the white-box implementation [25]. One possible solution to this problem is the use of external encodings that assumes the encryption function E_k is part of a more powerful environmental system E'_k :

$$E'_k = A \circ E_k \circ B^{-1},$$

where A and B are randomly selected bijective encodings that make it impossible for the attacker to calculate E'_k directly by extracting E_k .

2.3.3. Masking Technology

When generating the lookup table Y , for a plaintext P and secret key K , before encoding the output of E , one idea [31] is to use a mask α that is selected randomly to perform the

XOR operation followed by use of a linear encoding L and a different non-linear encoding N to encode the output:

$$N \circ L \circ [E(P, K) \oplus \alpha] \rightarrow Y.$$

Using the same linear encoding can protect the encryption operations and cancel the mask via a simple XOR operation.

3. White-Box Implementation of IDEA

The strategy of our solution is to split the IDEA into five parts. Each part is obfuscated and represented as a number of lookup tables. Specifically, we use randomly chosen group members and the additive inverses of $\mathbb{F}_{2^{16}+1}$ and $\mathbb{F}_{2^{16}}$ to transform the group operations to Type I lookup tables, add different masks to generate the Type II-V tables, and eliminate the masks via XOR operations with the Type II-E tables. Finally, external encodings are used to protect the initial inputs and final outputs.

3.1. Generating the Lookup Tables for Group Operations

A 64-bit initial plaintext P is split into four blocks $P = (P_1 \| P_2 \| P_3 \| P_4)$. Prior to the encryption operations, encoding blocks $L_{(0,1)}$, $L_{(0,2)}$, $L_{(0,3)}$, and $L_{(0,4)}$ are used to encode the plaintext blocks. The encoding blocks are randomly chosen group members from $\mathbb{F}_{2^{16}+1}$ or $\mathbb{F}_{2^{16}}$:

$$\begin{aligned} X_1 &= L_{(0,1)} \odot P_1, X_2 = L_{(0,2)} \otimes P_2, \\ X_3 &= L_{(0,3)} \otimes P_3, X_4 = L_{(0,4)} \odot P_4. \end{aligned}$$

We denote the four blocks of the encryption ROUND-1 as:

$$\begin{aligned} &(L_{(0,1)} \odot P_1 \| L_{(0,2)} \otimes P_2 \| L_{(0,3)} \otimes P_3 \| L_{(0,4)} \odot P_4) \\ &= (X_1 \| X_2 \| X_3 \| X_4) = X. \end{aligned}$$

The encoding blocks $L_{(0,1)}$, $L_{(0,2)}$, $L_{(0,3)}$, and $L_{(0,4)}$ are the external encodings to protect the initial blocks. The inputs of each encryption round are denoted as

$$X^{(r)} = (X_1^{(r)} \| X_2^{(r)} \| X_3^{(r)} \| X_4^{(r)}).$$

We select four group members $Q_{(r,a)}$, $a = 1, 2, 3, 4$ from $\mathbb{F}_{2^{16}+1}$ and four group members $P_{(r,a)}$ from $\mathbb{F}_{2^{16}}$ in each encryption round. We now describe the computations more concretely using the examples $X_1^{(r)}$ and $X_2^{(r)}$.

We implement the multiplication modulo (1) by splitting the 16-bit $X_1^{(r)}$ into two 8-bit parts (see Figure 1):

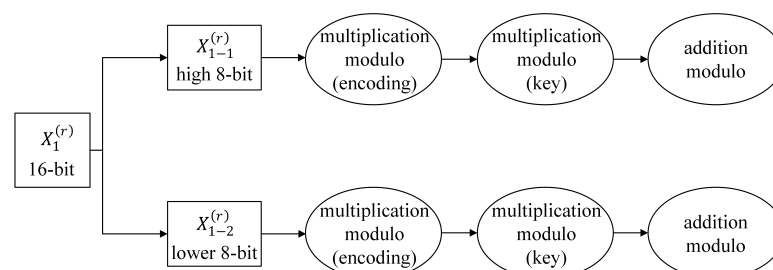


Figure 1. Splitting the input of group operations.

$$X_1^{(r)} = 2^8 \times X_{1-1}^{(r)} + X_{1-2}^{(r)}.$$

The two parts of $X_1^{(r)}$ are encoded by $L_{(r-1,1)}^{-1}$, which is the multiplicative inverse of $L_{(r-1,1)}$, and $Q_{(r,1)}$, resulting in: $X_{1-1}^{(r)'} and $X_{1-2}^{(r)'}$. Thus, the multiplication modulo (1) can be split into two calculations:$

$$\begin{aligned} X_1^{(r)} \odot Z_1^{(r)} &= [(X_{1-1}^{(r)'} + X_{1-2}^{(r)'}) \bmod (2^{16} + 1)] \odot Z_1^{(r)} \\ &= [X_{1-1}^{(r)'} \odot Z_1^{(r)} + X_{1-2}^{(r)'} \odot Z_1^{(r)}] \bmod (2^{16} + 1). \end{aligned}$$

We let

$$Y_{1-1}^{(r)} = X_{1-1}^{(r)'} \odot Z_1^{(r)}, Y_{1-2}^{(r)} = X_{1-2}^{(r)'} \odot Z_1^{(r)},$$

and then we randomly select the additive inverses (a_1, a_1') from $\mathbb{F}_{2^{16}+1}$, adding them to $Y_{1-1}^{(r)}$ and $Y_{1-2}^{(r)}$, respectively:

$$\begin{aligned} MY_{1-1}^{(r)} &= (Y_{1-1}^{(r)} + a_1) \bmod (2^{16} + 1), \\ MY_{1-2}^{(r)} &= (Y_{1-2}^{(r)} + a_1') \bmod (2^{16} + 1). \end{aligned}$$

The operations $X_{1-1}^{(r)} (\rightarrow X_{1-1}^{(r)'} \rightarrow Y_{1-1}^{(r)}) \rightarrow MY_{1-1}^{(r)}$, $X_{1-2}^{(r)} (\rightarrow X_{1-2}^{(r)'} \rightarrow Y_{1-2}^{(r)}) \rightarrow MY_{1-2}^{(r)}$ are implemented as lookup tables Type I-HB1 and Type I-LB1. We can compute the final result $Y_1^{(r)}$ using $MY_{1-1}^{(r)} + MY_{1-2}^{(r)} \bmod (2^{16} + 1)$.

We perform the addition modulo (2) by splitting $X_2^{(r)}$ into $X_{2-1}^{(r)}$ and $X_{2-2}^{(r)}$, but the difference is that the sub-key $Z_2^{(r)}$ is also encoded by $P_{(r,1)}$, and as a result, $Z_2^{(r)'}$. The multiplicative inverse of $L_{(r-1,2)}$ along with $P_{(r,1)}$ are used to encode $X_{2-1}^{(r)}$ and $X_{2-2}^{(r)}$, resulting in $X_{2-1}^{(r)'}$ and $X_{2-2}^{(r)'}$. This leads to

$$\begin{aligned} X_2^{(r)} \boxplus Z_2^{(r)} &= (X_{2-1}^{(r)'} \boxplus X_{2-2}^{(r)'}) \boxplus (2^8 \times Z_{2-1}^{(r)'} + Z_{2-2}^{(r)'}) \\ &= [X_{2-1}^{(r)'} \boxplus (2^8 \times Z_{2-1}^{(r)'})] \boxplus (X_{2-2}^{(r)'} \boxplus Z_{2-2}^{(r)'}), \end{aligned}$$

where $Z_{2-1}^{(r)'}$ and $Z_{2-2}^{(r)'}$ are generated by splitting $Z_2^{(r)'}$. We then define

$$Y_{2-1}^{(r)} = X_{2-1}^{(r)'} \boxplus (2^8 \times Z_{2-1}^{(r)'}), Y_{2-2}^{(r)} = X_{2-2}^{(r)'} \boxplus Z_{2-2}^{(r)'}$$

with further additive inverses (a_2, a_2') from $\mathbb{F}_{2^{16}}$ added into $Y_{2-1}^{(r)}$ and $Y_{2-2}^{(r)}$, respectively:

$$MY_{2-1}^{(r)} = Y_{2-1}^{(r)} \boxplus a_2, MY_{2-2}^{(r)} = Y_{2-2}^{(r)} \boxplus a_2'.$$

The transformations $X_{2-1}^{(r)} (\rightarrow X_{2-1}^{(r)'} \rightarrow Y_{2-1}^{(r)}) \rightarrow MY_{2-1}^{(r)}$ and $X_{2-2}^{(r)} (\rightarrow X_{2-2}^{(r)'} \rightarrow Y_{2-2}^{(r)}) \rightarrow MY_{2-2}^{(r)}$ are denoted as tables Type I-HB2 and Type I-LB2. The final result of the addition modulo (2) is $Y_2^{(r)}$, the result of computing $MY_{2-1}^{(r)} \boxplus MY_{2-2}^{(r)}$.

In the same way, we obtain the results $Y_3^{(r)}$ and $Y_4^{(r)}$, and have other four Type I lookup tables $X_{3-1}^{(r)} \rightarrow MY_{3-1}^{(r)}$, $X_{3-2}^{(r)} \rightarrow MY_{3-2}^{(r)}$, $X_{4-1}^{(r)} \rightarrow MY_{4-1}^{(r)}$, and $X_{4-2}^{(r)} \rightarrow MY_{4-2}^{(r)}$. Following the strategy, we transform the group operations for IDEA into eight Type I lookup tables. Figure 2 depicts them.

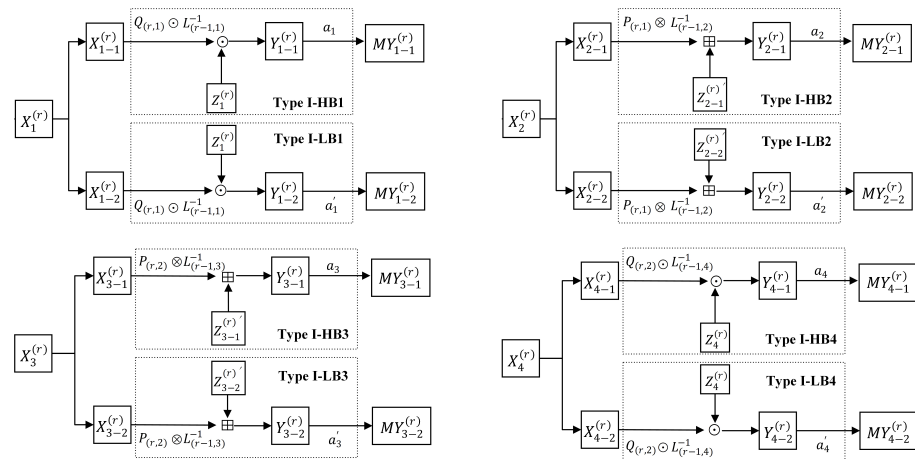


Figure 2. Generating the eight Type I lookup tables for group operations (the two corresponding 8-bit in and 16-bit out lookup tables can be calculated to obtain the encoded 16-bit original block).

3.2. Incorporating Masks

In this step, we mask four outputs of group operations by randomly selecting $MASK_{N1}$, $MASK_{N2} \in \{0, 1\}^{16}$. First, we offset the influences of the four group members. To do so, we design the *Calculation* boxes, the $CA_{(r,b)}$ box, $b = 1, 2, 3, 4$, to calculate the additive inverses of $[Q_{(r,a)} - 1] \odot \Gamma_{(r,b)}$ or $[P_{(r,a)} - 1] \otimes \Gamma_{(r,b)}$, denoted $\Psi_{(r,b)}$, where $\Gamma_{(r,b)}$ represents the four group operations (1) and (2). For example, In $CA_{(r,1)}$, the additive inverse of $[Q_{(r,1)} - 1] \odot \Gamma_{(r,1)}$ is $\Psi_{(r,1)}$, which is calculated and applied to the $Q_{(r,1)} \odot \Gamma_{(r,1)}$ by performing the addition modulo:

$$[\Psi_{(r,1)} + (Q_{(r,1)} - 1) \odot \Gamma_{(r,1)}] \bmod (2^{16} + 1) = 0,$$

$$(\Psi_{(r,1)} + Q_{(r,1)} \odot \Gamma_{(r,1)}) \bmod (2^{16} + 1) = \Gamma_{(r,1)}.$$

Second, we randomly select $MASK_{N1}$, $MASK_{N2} \in \{0, 1\}^{16}$ to perform the XOR operations and encode the outputs using S_r^{-1} , T_r^{-1} , where $S_r^{-1} = M_r^{-1} \circ A_{(r,1)}$, $T_r^{-1} = M_r^{-1} \circ A_{(r,2)}$, M_r , $A_{(r,1)}$, and $A_{(r,2)}$ are all 16×16 randomly selected reversible affine mappings. The overall process is shown in Figure 3.

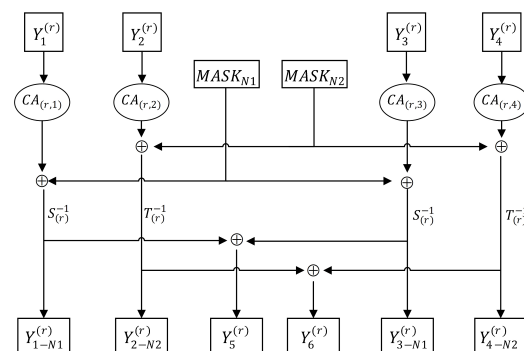


Figure 3. Application of masks.

Because the calculation of $\Psi_{(r,b)}$ is related to the keys, it must be invisible to users, and the application of masks is similar to adding round-keys. Thus, we use the Type II lookup table to include these two transformations.

We use $Y_1^{(r)} \rightarrow Y_{1-N1}^{(r)}$ as an example. In $CA(r, 1)$, we perform the calculation $[Q_{(r,1)} - 1] \odot \Gamma_{(r,1)}$ and then find $\Psi_{(r,1)}$ from $\mathbb{F}_{2^{16}+1}$. $\Psi_{(r,1)}$ is used to offset a part of $Q_{(r,1)} \odot \Gamma_{(r,1)}$. This calculation is

$$(Y_1^{(r)} + \Psi_{(r,1)}) \bmod (2^{16} + 1) = \Gamma_{(r,1)} = Y_1^{(r)'}$$

The mask $MASK_{N1} \in \{0, 1\}^{16}$ is selected randomly, and XORed with $Y_1^{(r)'}$. The composite affine mapping $S_{(r)}^{-1}$ encodes the output to obtain the result:

$$Y_{1-N1}^{(r)} = S_{(r)}^{-1} \circ \{[(Y_1^{(r)} + \Psi_{(r,1)}) \bmod (2^{16} + 1)] \oplus MASK_{N1}\}.$$

The Type II-V1 lookup table includes the operation $Y_1^{(r)} (\rightarrow Y_1^{(r)'}) \rightarrow Y_{1-N1}^{(r)}$, which uses a 16-bit input and produces a 16-bit output. $Y_{1-N1}^{(r)}$ is an intermediate value, which is key-sensitive and masked as well (see Figure 4). $MASK_{N1}$ is also encoded by $S_{(r)}^{-1}$, $MASK_{N1} \rightarrow E - MASK_{N1}$, and denoted as Type II-M1. Note that the encode masks are stored in other security components not in our solution.

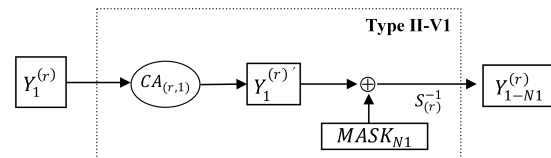


Figure 4. Generating the Type II-V1 lookup table (the application of $MASK_{N1}$).

Using the same method, we use the Type II-V lookup tables to obtain $Y_1^{(r)} \rightarrow Y_{1-N1}^{(r)}$, $Y_2^{(r)} \rightarrow Y_{2-N2}^{(r)}$, $Y_3^{(r)} \rightarrow Y_{3-N1}^{(r)}$, and $Y_4^{(r)} \rightarrow Y_{4-N2}^{(r)}$. The corresponding two blocks perform XOR operations to obtain the inputs of the MA structure:

$$Y_5^{(r)} = Y_{1-N1}^{(r)} \oplus Y_{3-N1}^{(r)}, Y_6^{(r)} = Y_{2-N2}^{(r)} \oplus Y_{4-N2}^{(r)}.$$

Since these XOR operations of IDEA eliminate $MASK_{N1}$ and $MASK_{N2}$ added in this step, the inputs of the MA structure have no masks.

3.3. Generating the Lookup Tables of MA Structure and Adding Another Masks

Although operations in MA structure are different, the four arithmetic calculations are similar to the group operations except for the number of the sub-keys (the group operations have four sub-keys, and the MA structure has two). The table type is also Type I. To encode the outputs of the MA structure, we use $CA_{(r,c)}$, $c = 5, 6$ to offset the confusion caused by the random group members and add the other two masks to generate the Type II lookup tables. The whole process of the MA structure is shown in Figure 5.

The inputs of the MA structure are $Y_5^{(r)}$ and $Y_6^{(r)}$. $Y_5^{(r)}$ is encoded by the composite affine mapping $R_{(r)}$ to obtain $Y_5^{(r)'}$, where $R_{(r)} = A_{(r,1)}^{-1} \circ M_{(r)}$, with $R_{(r)}$ stored in the encryption system. $Y_5^{(r)'}$ is split into two parts, and $Q_{(r,3)}$ is used to encode them. The multiplication modulo can be performed with the sub-key $Z_5^{(r)}$, and the addition inverses (a_5, a_5') from $\mathbb{F}_{2^{16}+1}$ are added, yielding the operations $Y_{5-1}^{(r)' \rightarrow MY_{5-1}^{(r)}}$ and $Y_{5-2}^{(r)' \rightarrow MY_{5-2}^{(r)}}$, which are then recorded as lookup tables Type I-HB5 and Type I-LB5. The final result $Y_7^{(r)}$ is obtained by the calculation $MY_{5-1}^{(r)} + MY_{5-2}^{(r)} \bmod (2^{16} + 1)$.

We use the composite affine mapping $T_{(r)}$ to encode $Y_6^{(r)}$, where $T_{(r)} = A_{(r,2)}^{-1} \circ M_{(r)}$. $T_{(r)}$ is also stored in the system. In particular, the addition modulo operation will be done with $Y_7^{(r)}$ rather than a sub-key. We split $Y_6^{(r)'}$ into two parts and use $P_{(r,3)}$ to encode $Y_{6-1}^{(r)'}$

and $Y_{6-2}^{(r)'} \cdot \Phi_{(r,1)}$ is the multiplicative inverse of $Q_{(r,3)}$ from $\mathbb{F}_{2^{16}+1}$ and is used to offset $Q_{(r,3)}$ on $Y_7^{(r)}$. $P_{(r,3)}$ handles it, yielding $Y_7^{(r)'}$. This leads to the following calculations:

$$\begin{aligned} & P_{(r,3)} \otimes (2^8 \times Y_{6-1}^{(r)'} + Y_{6-2}^{(r)'}) \boxplus [P_{(r,3)} \otimes (\Phi_{(r,1)} \odot Y_7^{(r)})] \\ &= [P_{(r,3)} \otimes (2^8 \times Y_{6-1}^{(r)'})] \boxplus [(P_{(r,3)} \otimes Y_{6-2}^{(r)'}) \boxplus Y_7^{(r)'}]. \end{aligned}$$

We then let

$$CY_{6-1}^{(r)} = P_{(r,3)} \otimes (2^8 \times Y_{6-1}^{(r)'}) , CY_{6-2}^{(r)} = (P_{(r,3)} \otimes Y_{6-2}^{(r)'}) \boxplus Y_7^{(r)'}$$

The additive inverses (a_6, a_6') from $\mathbb{F}_{2^{16}}$ should be incorporated to obtain

$$MY_{6-1}^{(r)} = CY_{6-1}^{(r)} \boxplus a_6, MY_{6-2}^{(r)} = CY_{6-2}^{(r)} \boxplus a_6'.$$

We implement the calculations $Y_{6-1}^{(r)'} \rightarrow MY_{6-1}^{(r)}$ and $Y_{6-2}^{(r)'} \rightarrow MY_{6-2}^{(r)}$ using tables Type I-HB6 and Type I-LB6. The final result $Y_8^{(r)}$ can then be calculated using $MY_{6-1}^{(r)} \boxplus MY_{6-2}^{(r)}$ (see Figure 6).

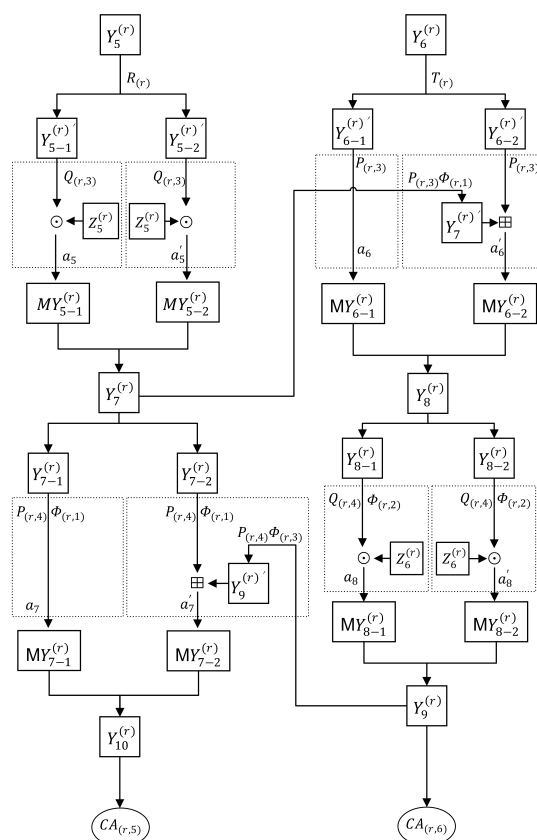


Figure 5. Process of the MA structure.

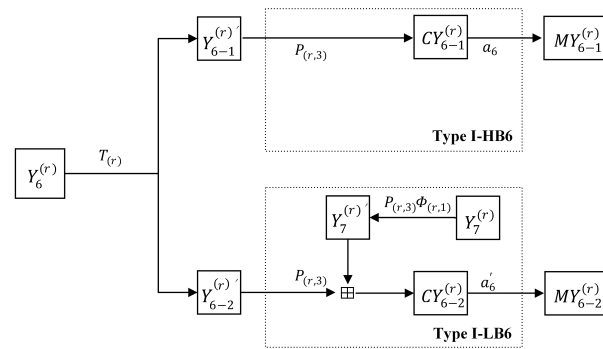


Figure 6. Generating the Type I lookup tables of the MA structure.

Using the same method, the second multiplication modulo in the MA structure uses $\Phi_{(r,2)}$ to eliminate $P_{(r,3)}$, with the output encoded by $Q_{(r,4)}$ and incorporating the additive inverses (a_8, a'_8) . From this, we obtain $Y_{8-1}^{(r)} \rightarrow MY_{8-1}^{(r)}$ and $Y_{8-2}^{(r)} \rightarrow MY_{8-2}^{(r)}$, implemented as the tables Type I-HB8 and Type I-LB8. The result of the calculations is $Y_9^{(r)}$. The two operations of the second addition modulo are $Y_{7-1}^{(r)} \rightarrow MY_{7-1}^{(r)}$ and $Y_{7-2}^{(r)} \rightarrow MY_{7-2}^{(r)}$, implemented as tables Type I-HB7 and Type I-LB7, with the calculation result $Y_{10}^{(r)}$. Finally, the outputs of the MA structure are $Y_9^{(r)}$ and $Y_{10}^{(r)}$.

After the MA structure, $MASK_{K1}$ and $MASK_{K2} \in \{0, 1\}^{16}$ are selected randomly, and they differ from $MASK_{N1}$ and $MASK_{N2}$. After calculations of $CA_{(r,c)}$, the XOR operations are performed via $Y_9^{(r)'} and $Y_{10}^{(r)'}$. Note that $CA_{(r,c)}$ uses the same method as $CA_{(r,b)}$. The composite affine mappings $T_{(r)}^{-1}$ and $S_{(r)}^{-1}$ are used to encode the outputs. We then obtain the Type II lookup tables providing the masked intermediate values $Y_{10-K1}^{(r)}$ and $Y_{9-K1}^{(r)}$. The transformation of $Y_{10}^{(r)}$ can be seen in Figure 7.$

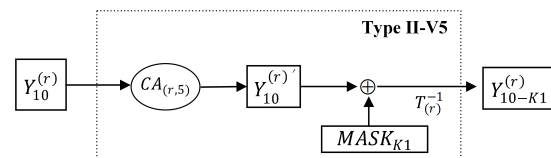


Figure 7. Generating the Type II-V5 lookup tables (the application of $MASK_{K1}$).

3.4. Eliminating the Masks

At this point, we have six blocks: four containing the outputs of group operations and two containing the outputs of the MA structure. We perform four XOR operations to complete the original IDEA design:

$$\begin{aligned} Y_{11}^{(r)} &= Y_{1-N1} \oplus Y_{9-K2}, Y_{12}^{(r)} = Y_{2-N2} \oplus Y_{10-K1}, \\ Y_{13}^{(r)} &= Y_{3-N1} \oplus Y_{9-K2}, Y_{14}^{(r)} = Y_{4-N2} \oplus Y_{10-K1}. \end{aligned}$$

There are two different masks for each of the four results $Y_{11}^{(r)}$, $Y_{12}^{(r)}$, $Y_{13}^{(r)}$, and $Y_{14}^{(r)}$. We remove the masks with the XOR operations and then exchange $Y_{16}^{(r)}$ and $Y_{17}^{(r)}$ (see Figure 8).

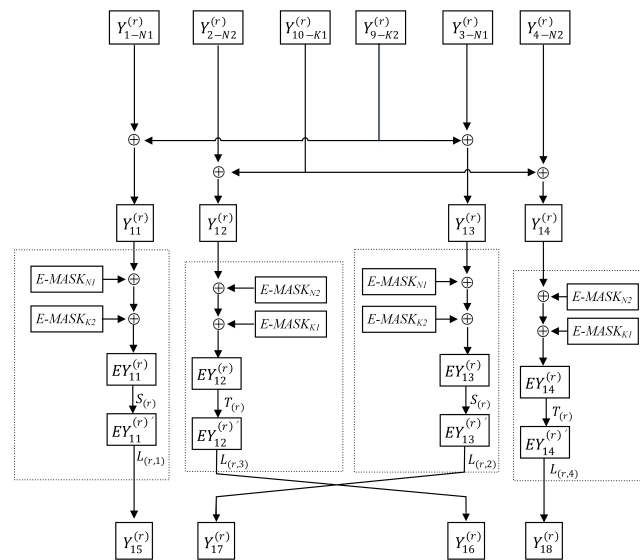


Figure 8. The whole mask elimination process (use the coded masks in Type II-M tables and affine transformations to perform operations in order).

The same affine layers make it easy to eliminate the masks with table Type II-M. The outputs should be encoded by $L_{(r,1)}$, $L_{(r,3)}$, $L_{(r,2)}$, and $L_{(r,4)}$, with the intermediate values encoded by $T_{(r)}$ and $S_{(r)}$. We now show $Y_{11}^{(r)}$ as an example (see Figure 9). After two masks have been eliminated, $S_{(r)}$ is used to encode the output, producing $EY_{11}^{(r)'}$, and $L_{(r,1)}$ is used to process $EY_{11}^{(r)'}$ from $\mathbb{F}_{2^{16}+1}$, with a final value of $Y_{15}^{(r)}$.

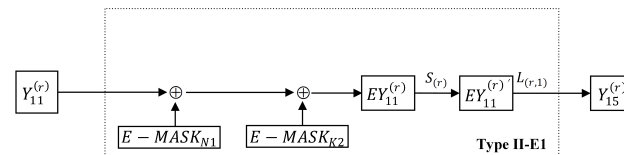


Figure 9. Eliminating the masks of $Y_{11}^{(r)}$ (the operations will generate a Type II-E1 lookup table).

Thus far, we have finished one encryption round. The other seven rounds have the same sequence of operations.

3.5. Generating the Lookup Tables for Output Transformation

The output transformation phase performs only group operations, so we also convert these computations into eight Type I lookup tables. Two corresponding tables can be used to compute EC_1 , EC_2 , EC_3 , and EC_4 , respectively. Finally, the ciphertext is

$$EC = (EC_1 \| EC_2 \| EC_3 \| EC_4).$$

In particular, we regard the multiplicative inverses of $Q_{(9,1)}$, $P_{(9,1)}$, $P_{(9,2)}$, and $Q_{(9,2)}$ as external encodings embedded in other components of the computer. Using the external encodings, we obtain the same outputs as the original IDEA implementation:

$$\begin{aligned} &(\Phi_{(9,1)} \odot EC_1 \| \Phi_{(9,2)} \otimes EC_2 \| \Phi_{(9,3)} \otimes EC_3 \| \Phi_{(9,4)} \odot EC_4) \\ &= (C_1 \| C_2 \| C_3 \| C_4) = C. \end{aligned}$$

4. Performance Analysis

The white-box IDEA requires 68 modular additions, 48 XORs, 216 lookups, and eight composite affine mappings.

All of the arithmetic operations are transformed into Type I lookup tables, requiring $4 \times (2 \times 2^8 \times 16) \times 8 + 4 \times (2 \times 2^8 \times 16) \times 8 + 4 \times (2 \times 2^8 \times 16) = 65 \times 2^{13}$ bits of storage. The two corresponding Type I lookup tables are synthesized (modular addition) and confused by 16×16 affine transformations and 16-bit random masks. The Type II lookup tables require $4 \times (2^{16} \times 16) \times 8 + 2 \times (2^{16} \times 16) \times 8 + 4 \times (2^{16} \times 16) \times 8 = 10240 \times 2^{13}$ bits of storage. The storage required for eight composite affine mappings is $8 \times 2 \times (16 \times 16 + 16) = 4115$ bits. The total storage cost is about 10.06 MB.

We used a common laptop with an Intel® Core(TM) i7-3630QM CPU at 2.40 GHz to test our white-box IDEA implementation written in Java 8.0. Encrypting a 64-bit plaintext 50,000 times with the original IDEA requires about 47 ms on average. Our white-box IDEA required 2786 ms on average, about 60 times slower than the plain algorithm, which is acceptable. Since our solution is the first white-box IDEA implementation, we can only compare efficiency with white-box implementations of other algorithms, such as KMAC [29] and AES [7]. As shown in Table 1, our solution offers competitive computing efficiency.

5. Security Analysis

Since the function of the S-box is replaced by algebraic operations in IDEA, the core idea of our solution is the transformation of the two arithmetic operations into lookup tables. Therefore, we conduct our security analysis on the lookup table structure of our white-box IDEA against algebraic attacks and BGE-like attacks.

5.1. Analysis of the Lookup Tables

We first analyze the Type I lookup tables using $X_1^{(r)}$ as the example. The adversary can obtain the input $X_{1-1}^{(r)}$ and the output $MY_{1-1}^{(r)}$ of a single Type I lookup table. $MZ_1^{(r)}$ can be computed by inverse multiplication:

$$MZ_1^{(r)} = MY_{1-1}^{(r)} \odot [X_{1-1}^{(r)}]^{-1}. \quad (3)$$

On the other hand, by combining the two Type I lookup tables to eliminate the influence of (a_1, a'_1) , we can find:

$$MZ_1^{(r)'} = Y_1^{(r)} \odot [X_1^{(r)}]^{-1}. \quad (4)$$

Due to the randomness of (a_1, a'_1) , $Q_{(r,1)}$, and $L_{(0,1)}^{-1}$, where (a_1, a'_1) are additive inverses and $Q_{(r,1)}, L_{(0,1)}^{-1}$ are group members from $\mathbb{F}_{2^{16}+1}$. They provide 2^{48} different values in (3) and 2^{32} probabilities in (4). Thus, there is no effective calculation or analysis of the secret key.

Another approach is the 2003 proposal from Biryukov et al. [24] that describes an algorithm for any two substitutions (S-boxes) to solve the affine equivalence problem. The S_1 and S_2 affine equivalents should satisfy the equation

$$\begin{aligned} S_2(X) &= \Lambda_2 \circ S_1(\Lambda_1 \circ X \oplus \alpha) \oplus \beta \quad \text{or} \\ S_2(X) \oplus \beta &= \Lambda_2 \circ S_1(\Lambda_1 \circ X \oplus \alpha), \end{aligned}$$

where Λ_1 and Λ_2 are $n \times n$ invertible matrices, and α and β are n -bit columns. The linearity of Λ_1, Λ_2 means we can check all α, β that satisfy the above equation to the recovered key with time complexity $O(n^3 2^{2n})$. The core idea of this attack is to construct the affine equivalence problem.

Since the algebraic operations replace the S-boxes, it is not possible to find the S-boxes and the linear relationship directly. However, we can regard the multiplication modulo (or addition modulo) operation as a T-box to try to find a structure equivalent to the S-box.

Then, using the idea of the affine equivalence problem to attack the solution, and using the example $X_1^{(r)}$, we have

$$\begin{aligned} Y_{1-N1}^{(r)} &= S_{(r)}^{-1} \circ \{CA_{(r,1)} \circ [(Q_{(r,1)} \odot L_{(r,1)}^{-1} \odot X_1^{(r)}) \odot Z_1^{(r)}] \oplus MASK_{N1}\} \\ &= S_{(r)}^{-1} \circ T_1[\Lambda_1 \otimes X_1^{(r)}] \oplus [S_{(r)}^{-1} \circ MASK_{N1}]. \end{aligned}$$

Here, we regard Λ_1 as a random group member from $\mathbb{F}_{2^{16}+1}$, α as a 16-bit column, and the T_1 box as a type of S-box representing the multiplication modulo and the operation of the $CA_{(r,1)}$ box. We regard the added masks as a T_2 box. This leads to

$$T_2(X_1^{(r)}) = Y_{1-N1}^{(r)} = S_{(r)}^{-1} \circ T_1[\Lambda_1 \otimes X_1^{(r)}] \oplus [S_{(r)}^{-1} \circ MASK_{N1}].$$

We then rewrite the preceding equation as

$$T_2(X_1^{(r)}) = \Lambda_2 \circ T_1(\Lambda_1 \otimes X_1^{(r)} \oplus \alpha) \oplus \beta,$$

where $S_{(r)}^{-1}$ is represented by Λ_2 , which is a randomly selected reversible composite affine mapping, and $S_{(r)}^{-1} \circ MASK_{N1}$ is represented by β , which is actually the Type II-M1 lookup tables. Thus, we have constructed the affine equivalence problem.

In our scheme, the above equation is only similar in form to the structure of the affine equivalence problem because the operations of the T-boxes and S-boxes are different. Even if the adversary combines the Type I and Type II tables to obtain the encoded masks' lookup tables, they also need to determine all possible Λ_1 , Λ_2 , and α to obtain the key information hidden in the T_1 box from the above formula.

5.2. BGE-Like Attacks

Billet et al. [23] proposed a method called BGE to attack the white-box AES designed by Chow et al. [7]. The core idea of the BGE attack is the transformation of the non-linear structure into a linear structure, construction of a linear relationship, and then use the linear relationship to determine the affine or linear encoding. The time complexity of this algorithm is 2^{22} at present.

The group operations of the IDEA, multiplication modulo, and addition modulo can be performed separately as non-linear T_1 - and T_2 -boxes, which are treated as non-linear S-boxes. We consider the group operations of one encryption round as follows:

$$\begin{cases} MY_{1-1}^{(r)} = Q_{(r,1)} \odot L_{(r-1,1)}^{-1} \odot (2^8 \times X_{1-1}^{(r)}) \odot Z_1^{(r)} \boxplus a_1, \\ MY_{1-2}^{(r)} = Q_{(r,1)} \odot L_{(r-1,1)}^{-1} \odot X_{1-2}^{(r)} \odot Z_1^{(r)} \boxplus a'_1, \\ MY_{2-1}^{(r)} = P_{(r,1)} \otimes L_{(r-1,2)}^{-1} \otimes (2^8 \times X_{2-1}^{(r)}) \boxplus (2^8 \times Z_{2-1}^{(r)'}) \boxplus a_2, \\ MY_{2-2}^{(r)} = P_{(r,1)} \otimes L_{(r-1,2)}^{-1} \otimes X_{2-2}^{(r)} \boxplus Z_{2-2}^{(r)'} \boxplus a'_2, \\ MY_{3-1}^{(r)} = P_{(r,2)} \otimes L_{(r-1,3)}^{-1} \otimes (2^8 \times X_{3-1}^{(r)}) \boxplus (2^8 \times Z_{3-1}^{(r)'}) \boxplus a_3, \\ MY_{3-2}^{(r)} = P_{(r,2)} \otimes L_{(r-1,3)}^{-1} \otimes X_{3-2}^{(r)} \boxplus Z_{3-2}^{(r)'} \boxplus a'_3, \\ MY_{4-1}^{(r)} = Q_{(r,2)} \odot L_{(r-1,4)}^{-1} \odot (2^8 \times X_{4-1}^{(r)}) \odot Z_4^{(r)} \boxplus a_4, \\ MY_{4-2}^{(r)} = Q_{(r,2)} \odot L_{(r-1,4)}^{-1} \odot X_{4-2}^{(r)} \odot Z_4^{(r)} \boxplus a'_4. \end{cases}$$

We can also consider the MA structure, obtaining the equations:

$$\begin{cases} MY_{5-1}^{(r)} = Q_{(r,3)} \odot (2^8 \times Y_{5-1}^{(r)'}) \odot Z_5^{(r)} \boxplus a_5, \\ MY_{5-2}^{(r)} = Q_{(r,3)} \odot Y_{5-2}^{(r)'}) \odot Z_5^{(r)} \boxplus a'_5. \\ MY_{6-1}^{(r)} = P_{(r,3)} \otimes (2^8 \times Y_{6-1}^{(r)'}) \boxplus a_6, \\ MY_{6-2}^{(r)} = P_{(r,3)} \otimes Y_{6-2}^{(r)'}) \boxplus Y_7^{(r)'}) \boxplus a'_6. \\ MY_{7-1}^{(r)} = [P_{(r,4)} \otimes \Phi_{(r,1)} \odot (2^8 \times Y_{7-1}^{(r)'})] \boxplus a_7, \\ MY_{7-2}^{(r)} = (P_{(r,4)} \otimes \Phi_{(r,1)} \odot Y_{7-2}^{(r)'}) \boxplus Y_9^{(r)'}) \boxplus a'_7. \\ MY_{8-1}^{(r)} = [Q_{(r,4)} \odot \Phi_{(r,2)} \otimes (2^8 \times Y_{8-1}^{(r)'})] \odot Z_6^{(r)} \boxplus a_8, \\ MY_{8-2}^{(r)} = (Q_{(r,4)} \odot \Phi_{(r,2)} \otimes Y_{8-2}^{(r)'}) \odot Z_6^{(r)} \boxplus a'_8. \end{cases}$$

From these eight sets of equations, we can see that the calculation methods of the T_1 and T_2 boxes are not exactly the same, meaning that a linear relationship $MY_i^{(r)} = \mathcal{K} \circ Y_j^{(r)} \circ \sigma$, where \mathcal{K} denotes random group members from $\mathbb{F}_{2^{16}+1}$ or $\mathbb{F}_{2^{16}}$, and σ is a constant, which does not necessarily exist. If the linear relationship is not established, the BGE-like method cannot recover the keys.

In addition, the number of invertible matrices of order 16 is

$$(2^{16} - 1) \times \prod_{j=1}^{15} (2^{16} - 1 - \sum_{k=1}^j \binom{j}{k}) = 2^{255}.$$

For white-box diversity and white-box ambiguity [7], our scheme has enough randomness to counter brute-force attacks.

In summary, although our scheme is a completely new implementation of a cryptography primitive, it is still sufficiently secure to resist ordinary algebraic analysis and BGE-like attacks.

6. Conclusions and Discussion

In this paper, we proposed a scheme to implement the white-box IDEA. We focused on the computation of two arithmetic operations in IDEA and developed a method to transform these arithmetic operations into lookup tables (Type I) and embedded four different masks to increase the resistance against white-box attacks (Type II). Our implementation presents a new approach for transforming different algebraic operations into lookup tables, and can be applied to other encryption systems with similar algebraic structures with resistance against algebraic attacks, including BGE attacks.

Since the security evaluation method of the white-box model is not as complete as in the black-box model, our future work will have two avenues. One effort will be to test other new white-box implementation methods. For example, we want to design a completely new white-box cipher with much smaller storage costs. The other will be to optimize security and robustness of this white-box IDEA implementation with the expectation that it will provide long-term protection. Since the structures of the three operations are completely different from other block ciphers, we used the relatively large lookup tables (Type II) to convert these calculations. Without the traditional S-box, there are still great difficulties and challenges in optimizing the storage costs and security of the lookup tables.

Author Contributions: Conceptualization, S.P.; methodology, S.P. and T.L.; software, S.P.; formal analysis, S.P. and T.L.; writing—original draft preparation, S.P.; Writing review and editing, T.L., X.L. and Z.G. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Natural Science Foundation of China (61702331, 61972248, 61972249, 62072192, U1536101), National Cryptography Development Fund (No. MMJJ20180206,

No. MMJJ20170105), Science and Technology on Communication Security Laboratory and State Key Laboratory of Cryptology No.MMKFKT201905, and China Postdoctoral Science Foundation No.2017M621471.

Informed Consent Statement: Not applicable.

Institutional Review Board Statement: Not applicable.

Data Availability Statement: The study did not report any data.

Conflicts of Interest: The authors declare no conflict of interest.

References

- Lai, X. On the Design and Security of Block Ciphers. Ph.D. Thesis, ETH Zurich, Zurich, Swiss, 1992.
- Lai, X.; Massey, J.L. A Proposal for a New Block Encryption Standard. In *Advances in Cryptology, Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques, Aarhus, Denmark, 21–24 May 1990*; Springer: Berlin/Heidelberg, Germany, 1990.
- Hoffman, N. A simplified IDEA algorithm. *Cryptologia* **2007**, *31*, 143–151. [CrossRef]
- Singh, H.P.; Verma, S.; Mishra, S. Secure-International Data Encryption Algorithm. *Int. J. Adv. Res. Electr. Electron. Instrum. Eng.* **2013**, *2*, 780–792.
- Rahim, R.; Mesran, M.; Siahaan, A.P.U. Data Security with International Data Encryption Algorithm. 2017. Available online: <https://doi.org/10.31227/osf.io/r98e5> (accessed on 3 October 2017).
- Zimmermann, R.; Curiger, A.; Bonnenberg, H.; Kaeslin, H.; Felber, N.; Fichtner, W. A 177 Mb/s VLSI implementation of the international data encryption algorithm. *IEEE J. Solid State Circuits* **1994**, *29*, 303–307. [CrossRef]
- Chow, S.; Eisen, P.; Johnson, H.; Oorschot, P.C.V. White-box cryptography and an AES implementation. In *Proceedings of the International Workshop on Selected Areas in Cryptography, St. John's, NF, Canada, 15–16 August 2002*; Springer: Berlin/Heidelberg, Germany, 2002; pp. 250–270.
- Chow, S.; Eisen, P.; Johnson, H.; van Oorschot, P.C. A white-box DES implementation for DRM applications. In *Proceedings of the ACM Workshop on Digital Rights Management, Washington, DC, USA, 18 November 2002*; Springer: Berlin/Heidelberg, Germany, 2002; pp. 1–15.
- Shamir, A.; Van Someren, N. Playing ‘hide and seek’ with stored keys. In *Proceedings of the International Conference on Financial Cryptography, Anguilla, British West Indies, 22–25 February 1999*; Springer: Berlin/Heidelberg, Germany, 1999; pp. 118–124.
- Halderman, J.A.; Schoen, S.D.; Heninger, N.; Clarkson, W.; Paul, W.; Calandrino, J.A.; Feldman, A.J.; Appelbaum, J.; Felten, E.W. Lest we remember: Cold-boot attacks on encryption keys. *Commun. ACM* **2009**, *52*, 91–98. [CrossRef]
- Liu, Q.; Safavi-Naini, R.; Sheppard, N.P. Digital rights management for content distribution. In *Proceedings of the Australasian Information Security Workshop Conference on ACSW Frontiers 2003—Volume 21*; Australian Computer Society, Inc.: Darlinghurst, Australia, 2003; pp. 49–58.
- Mulligan, D.K.; Han, J.; Burstein, A.J. How DRM-based content delivery systems disrupt expectations of “personal use”. In *Proceedings of the 3rd ACM Workshop on Digital Rights Management, Washington, DC, USA, 27 October 2003*; pp. 77–89.
- Chess, D.; Harrison, C.; Kershenbaum, A. Mobile agents: Are they a good idea? In *Proceedings of the International Workshop on Mobile Object Systems, Linz, Austria, 8–9 July 1996*; Springer: Berlin/Heidelberg, Germany, 1996; pp. 25–45.
- D’Anna, L.; Matt, B.; Risse, A.; Vleck, T.V.; Schwab, S.; LeBlanc, P. *Self-Protecting Mobile Agents Obfuscation Report*; Network Associates Laboratories Report; 2003. Available online: <http://profs.sci.univr.it/~giaco/download/Watermarking-Obfuscation/obfreport.pdf> (accessed on 3 October 2017).
- Yih-Chun, H.; Perrig, A. A survey of secure wireless ad hoc routing. *IEEE Secur. Priv.* **2004**, *2*, 28–39. [CrossRef]
- Ng, H.S.; Sim, M.L.; Tan, C.M. Security issues of wireless sensor networks in healthcare applications. *BT Technol. J.* **2006**, *24*, 138–144. [CrossRef]
- Tao, X.; Chuankun, W.; Weiming, Z. A White-Box-Cryptography-Based Scheme for the Secure Chip of DCAS Terminal. *J. Comput. Res. Dev.* **2016**, *53*, 2465.
- Karroumi, M. Protecting White-Box AES with Dual Ciphers. In *International Conference on Information Security and Cryptology*; Springer: Berlin/Heidelberg, Germany, 2010.
- Link, H.E.; Neumann, W.D. Clarifying Obfuscation: Improving the Security of White-Box Encoding. *IACR Cryptol. ePrint Arch.* **2004**, *2004*, 25.
- Shi, Y.; Wei, W.; He, Z. A lightweight white-box symmetric encryption algorithm against node capture for WSNs. *Sensors* **2015**, *15*, 11928–11952. [CrossRef] [PubMed]
- Xiao, Y. White-Box Cryptography and a White-Box Implementation of the SMS4 Algorithm. Master’s Thesis, Shanghai Jiaotong University, Shanghai, China, 2009.
- Xiao, Y.; Lai, X. A Secure Implementation of White-Box AES. In *Proceedings of the International Conference on Computer Science & Its Applications, Jeju, Korea, 10–12 December 2009*; pp. 1–6.
- Billet, O.; Gilbert, H.; Ech-Chatbi, C. Cryptanalysis of a white box AES implementation. In *Proceedings of the International Workshop on Selected Areas in Cryptography, Waterloo, ON, Canada, 9–10 August 2004*; Springer: Berlin/Heidelberg, Germany, 2004; pp. 227–240.

24. Biryukov, A.; De Canniere, C.; Braeken, A.; Preneel, B. A toolbox for cryptanalysis: Linear and affine equivalence algorithms. In Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, 4–8 May 2003; Springer: Berlin/Heidelberg, Germany, 2003; pp. 33–50.
25. Bos, J.W.; Hubain, C.; Michiels, W.; Teuwen, P. Differential computation analysis: Hiding your white-box designs is not enough. In Proceedings of the International Conference on Cryptographic Hardware and Embedded Systems, Santa Barbara, CA, USA, 17–19 August 2016; Springer: Berlin/Heidelberg, Germany, 2016; pp. 215–236.
26. De Mulder, Y.; Roelse, P.; Preneel, B. Cryptanalysis of the Xiao-Lai white-box AES implementation. In Proceedings of the International Conference on Selected Areas in Cryptography, Windsor, ON, Canada, 15–16 August 2002; Springer: Berlin/Heidelberg, Germany, 2012; pp. 34–49.
27. Jacob, M.; Boneh, D.; Felten, E. Attacking an obfuscated cipher by injecting faults. In Proceedings of the ACM Workshop on Digital Rights Management, Washington, DC, USA, 18 November 2002; Springer: Berlin/Heidelberg, Germany, 2002; pp. 16–31.
28. Wyseur, B.; Michiels, W.; Gorissen, P.; Preneel, B. Cryptanalysis of white-box DES implementations with arbitrary external encodings. In Proceedings of the International Workshop on Selected Areas in Cryptography, Ottawa, ON, Canada, 16–17 August 2007; Springer: Berlin/Heidelberg, Germany, 2007; pp. 264–277.
29. Lu, J.; Zhao, Z.; Guo, H. White-Box Implementation of the KMAC Message Authentication Code. In Proceedings of the International Conference on Information Security Practice and Experience, Kuala Lumpur, Malaysia, 26–28 November 2019; Springer: Cham, Switzerland, 2019; pp. 248–270.
30. Bock, E.A.; Bos, J.W.; Brzuska, C.; Hubain, C.; Michiels, W.; Mune, C.; Gonzalez, E.S.; Teuwen, P.; Treff, A. White-box cryptography: Don't forget about grey-box attacks. *J. Cryptol.* **2019**, *32*, 1095–1143. [[CrossRef](#)]
31. Lee, S.; Kim, T.; Kang, Y. A masked white-box cryptographic implementation for protecting against differential computation analysis. *IEEE Trans. Inf. Forensics Secur.* **2018**, *13*, 2602–2615. [[CrossRef](#)]