Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2013, Article ID 431794, 9 pages
http://dx.doi.org/10.1155/2013/431794

Research Article

A Secure Implementation of a Symmetric Encryption Algorithm in White-Box Attack Contexts

Yang Shi, Qin Liu, and Qinpei Zhao

School of Software Engineering, Tongji University, Shanghai 200184, China

Correspondence should be addressed to Qin Liu; qin.liu@tongji.edu.cn

Received 21 July 2013; Accepted 17 September 2013

Academic Editor: Sabri Arik

Copyright © 2013 Yang Shi et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

In a white-box context, an adversary has total visibility of the implementation of the cryptosystem and full control over its execution platform. As a countermeasure against the threat of key compromise in this context, a new secure implementation of the symmetric encryption algorithm SHARK is proposed. The general approach is to merge several steps of the round function of SHARK into table lookups, blended by randomly generated mixing bijections. We prove the soundness of the implementation of the algorithm and analyze its security and efficiency. The implementation can be used in web hosts, digital right management devices, and mobile devices such as tablets and smart phones. We explain how the design approach can be adapted to other symmetric encryption algorithms with a slight modification.

1. Introduction

There are three main models of the capability of an adversary to attack a cryptosystem [1]. First is the black-box model. It is a traditional attack model where an adversary only has access to the input and corresponding output of a cryptosystem. The limited information available means that an attack is usually difficult and time consuming. The second model is the grey-box model, where a leakage function is present. In such an attack model, the adversary can deploy side-channel cryptanalysis techniques. Several grey-box models can be defined because of the large variety of leakage functions. Third is the white-box model where the adversary has total visibility of the cryptographic software implementation and full control over its execution. One could refer to the white-box model as the worst-case model. The white-box model is used to analyze algorithms that are running in an untrustworthy environment, that is, an environment in which applications are subject to attacks from the execution platform.

Typical white-box attack contexts include

(1) a server or PC that an attacker has got the "root" or "admin" privilege of it,

(2) a mobile agent that is running on a malicious host,

(3) an attacker has control of an outdoor wireless sensor network node,

(4) digital right management (DRM) components in cable television applications.

Secure computing in a white-box attack context (WABC) is a challenge because, as discussed in [2, 3], (1) fully-privileged attack software shares a host with cryptographic software and has complete access to the implementation of algorithms, (2) dynamic execution (with instantiated cryptographic keys) can be observed, and (3) internal details of cryptographic algorithms are both completely visible and alterable.

Standard design and implementation of symmetric encryption algorithms were not intended to operate in a white-box attack context where their execution could be observed. In fact, cryptographic models usually assume that endpoints, hosts, and hardware protection tokens are to be trusted. This is not the case in a white-box attack. By actively monitoring standard cryptographic functions or memory dumps, an attacker can even extract the cryptographic keys. This is extremely dangerous when using a symmetric encryption scheme because the decryption algorithm uses the same key as the encryption algorithm.

In response to this security challenge, we propose a new, secure, and white-box implementation of a symmetric encryption algorithm that reduces the risk of keys being compromised. Note that the terms "white-box encryption algorithm" and "white-box implementation of an encryption algorithm" are used interchangeably throughout the paper.

The remainder of this article is organized as follows. Section 2 describes recent advances in white-box cryptography. A new white-box symmetric encryption algorithm is proposed in Section 3, followed by a security analysis in Section 4. Section 5 analyzes the complexity and performance of the new algorithm and includes a suggested implementation approach and some experimental results. In Section 6 we conclude with a discussion of our findings and ideas for future research.

2. Recent Advances in White-Box Cryptography

White-box cryptography provides protection to software implementations of encryption algorithms that may be executed on an untrustworthy host or other white-box attack contexts. The main constraint is that the result must be directly executable. Chow et al. introduced this idea and proposed a white-box implementation of DES by interleaving affine transformations and using delinearization techniques [2]. Chow et al. also introduced a white-box implementation of AES, representing it with a set of key-dependent look-up tables [3]. Their original proposal is that these two algorithms could be used in digital rights management (DRM) applications to satisfy the need to protect digital information content from unauthorized access, use, and dissemination.

In [4], Jacob et al. proposed that a fault injection attack, where an attacker injects errors into the program environment during execution, could defeat some obfuscation methods. They presented a cryptanalysis of a variant of the algorithm in [2] that does not have external encodings. Link and Neumann implemented white-box DES and triple-DES algorithms along the lines of Chow et al., with alterations that improved the security of the key [5]. Their system is secure against the previously published attacks on the implementation of Chow et al. and their own adaptation of a statistical bucketing attack. In 2007, Wyseur et al. [6] and Goubin et al. [7] independently cryptanalyzed all existing obfuscation methods of DES. Both attacks were based on a truncated differential cryptanalysis. Goubin et al. presented an attack that analyzed the first rounds of the white-box DES implementations, while Wyseur et al. presented an attack that works on the internal information.

In [8], Billet et al. presented an efficient and practical attack against the obfuscated AES implementation proposed by Chow et al. in [3]. It used negligible memory and had the worst time complexity of 2^{30}. In 2009, Michiels et al. improved the attack so that it could be deployed on a generic class of white-box implementations [9]. In 2011, Karroumi proposed a new white-box implementation that uses dual representations of AES [10]. Karroumi claimed that the time complexity of Billet et al.'s attack against his white-box AES is 2^{91}. Furthermore, even with the more powerful attack tool [11] proposed by Tolhuizen last year, the expected time complexity of Billet et al.'s attack remains 2^{81}.

In [12], Xiao and Lai proposed a secure implementation of white-box AES after a detailed analysis of the attack technique in [8] on the AES implementation proposed in [3]. In their scheme, the obfuscation works on at least two cells of an AES state, which the attacker cannot divide them into small ones and remove them using the attack technique proposed in [8]. The time complexity of Xiao and Lai's white-box AES implementation is 2^{24}. It is slower than Chow et al.'s implementation, which has a time complexity of 2^{20} [3]. Furthermore, the size of Xiao and Lai's white-box AES implementation is 20502 KB. In 2012, Mulder et al. [13] presented a cryptanalysis of a white-box AES implementation, based on Xiao and Lai's idea. They applied the linear equivalence algorithm presented by Biryukov et al. in [14] as a building block. The cryptanalysis efficiently extracts the AES key with a work factor of approximately 2^{32}. Furthermore, the size of Xiao and Lai's implementation still has potential to be improved.

3. A Novel White-Box Symmetric Encryption Algorithm

In this section, we propose a new white-box symmetric encryption algorithm based on SHARK [15]. Our general approach is to merge several steps of each round function of SHARK into table lookups, blending by randomly generated mixing bijections. We use techniques from [10, 12] to obtain the obfuscated implementation.

3.1. The Symmetric Encryption Algorithm, SHARK. SHARK is a six round substitution permutation-network that alternates a key mixing stage with linear and nonlinear transformation layers. We can split each round of the SHARK algorithm into three distinct layers: a nonlinear layer of substitution boxes, a diffusion layer, and a key addition layer. An interpolation attack can break the five rounds of a modified version of SHARK [16], but the security of the six round SHARK cipher is acceptable for many applications.

Let $S : GF(2^8) \to GF(2^8),\ x \mapsto S[x]$ denote the mapping of S-boxes. Then the nonlinear layer can be defined as $\gamma : GF(2^8)^8 \to GF(2^8)^8,\ \gamma(a) = b \Leftrightarrow b_i = S[a_i],\ 0 \le i \le 7$.

Let $\lambda : GF(2^8)^8 \to GF(2^8)^8$ be the linear transformation corresponding to the diffusion layer. Then there exists a matrix H such that $\lambda(a) = b \Leftrightarrow b = a \cdot H$.

Furthermore, let K^r be the round key of the rth round and let $\sigma[K^r] : GF(2^8)^8 \to GF(2^8)^8$ be the key addition mapping.

Now, the symmetric encryption algorithm SHARK with encryption key K is defined as follows:

$$\text{SHARK}[K] = \lambda^{-1} \circ \left(\mathop{\circ}_{r=1}^{6} \sigma[K^r] \circ \lambda \circ \gamma \right) \circ \sigma[K^0]. \quad (1)$$

3.2. Components of the White-Box Encryption Algorithm. To hide the encryption key, we must merge several steps of each

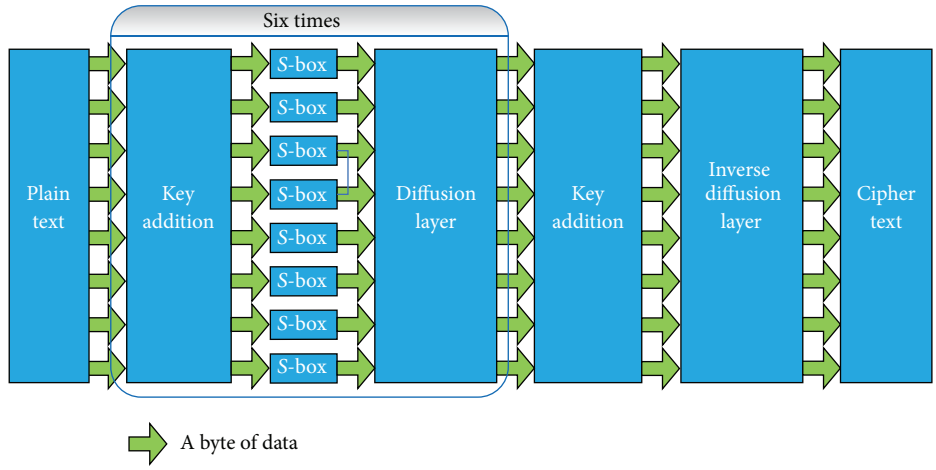


Figure 1: Flow of the SHARK algorithm.

round function of SHARK into table lookups blended by randomly generated mixing bijections. In this section, we investigate how to design such tables and how randomly generated mixing bijections can be counteracted.

Because $\mathrm{SHARK}[K] = \lambda^{-1} \circ \left(\mathop{\circ}_{r=1}^{6} \sigma[K^r] \circ \lambda \circ \gamma \right) \circ \sigma[K^0] \lambda^{-1} \circ \sigma[K^6] \circ \left(\mathop{\circ}_{r=0}^{5} \lambda \circ \gamma \circ \sigma[K^r] \right)$, we can also define the algorithm as

$$\mathrm{SHARK}[K] = \lambda^{-1} \circ \sigma\left[K^6\right] \circ \left(\mathop{\circ}_{r=0}^{5} \rho\left[K^r\right] \right), \tag{2}$$

where $\rho[K^r]$ is the round function of the rth round with round key K^r defined as

$$\rho\left[K^r\right] \equiv \lambda \circ \gamma \circ \sigma\left[K^r\right]. \tag{3}$$

The flow of SHARK depicted in (2) and (3) is shown in Figure 1.

Let M^r be a 64×64 nonsingular matrix over $GF(2)$, defined for $r = 1, \ldots, 6$ as

$$M^r = \left(N^{r-1}\right)^{-1} \begin{bmatrix} \left(Q^{r,0}\right)^{-1} & & & \\ & \left(Q^{r,1}\right)^{-1} & & \\ & & \left(Q^{r,2}\right)^{-1} & \\ & & & \left(Q^{r,3}\right)^{-1} \end{bmatrix}, \tag{4}$$

where N^r, $r = 0, \ldots, 6$ are randomly generated 64×64 nonsingular matrices over $GF(2)$.

The external input encoding, U, is a 64×64 nonsingular matrix over $GF(2)$ defined as

$$U = \begin{bmatrix} \left(Q^{0,0}\right)^{-1} & & & \\ & \left(Q^{0,1}\right)^{-1} & & \\ & & \left(Q^{0,2}\right)^{-1} & \\ & & & \left(Q^{0,3}\right)^{-1} \end{bmatrix}, \tag{5}$$

where $Q^{r,i}$, $r = 1, \ldots, 6$, $i = 0, \ldots, 3$ are randomly generated 16×16 nonsingular matrices over $GF(2)$. The external output encoding $V = \left(N^6\right)^{-1}$ is also a 64×64 nonsingular matrix over $GF(2)$.

In a white-box encryption algorithm, round functions should be obfuscated to protect the round keys against attacks from an adversary. Using the definitions above, we can define the obfuscated round functions, which we will implement using a set of tables (T-Boxes). For each round, r, let the obfuscated subround function be $\rho_W[r, i, k] : GF(2^8)^2 \to GF(2^8)^8$, $i = 0, 1, 2, 3$.

The number of possible different representations of $GF(2^{16})$ is 8160. The isomorphic transformation Δ that takes the description of the cipher under the standard irreducible polynomial to another description with a different irreducible polynomial is linear. For each round r, Δ_r is chosen randomly from these isomorphic transformations.

Let

$$\begin{aligned} L_i^0 &= \Delta_0 \circ \left(\cdot Q^{0,i}\right); \quad i = 0, 1, 2, 3, \\ L_i^r &= \Delta_r \circ \Delta_{r-1}^{-1} \circ \left(\cdot Q^{r,i}\right), \quad r = 1, \ldots, 6;\ i = 0, \ldots, 3 \end{aligned} \tag{6}$$

be preround mixing bijections.

Let

$$P^r = \left(\Delta_r(H)\right) \cdot N^r \overset{\text{def}}{=} \begin{bmatrix} P_0^r \\ P_1^r \\ P_2^r \\ P_3^r \end{bmatrix}, \quad r = 0, \ldots, 5, \tag{7}$$

$$P^6 = \left(\Delta_6\left(H^{-1}\right)\right) \cdot N^6 \overset{\text{def}}{=} \begin{bmatrix} P_0^6 \\ P_1^6 \\ P_2^6 \\ P_3^6 \end{bmatrix}$$

be postround diffusion-mixing bijections.

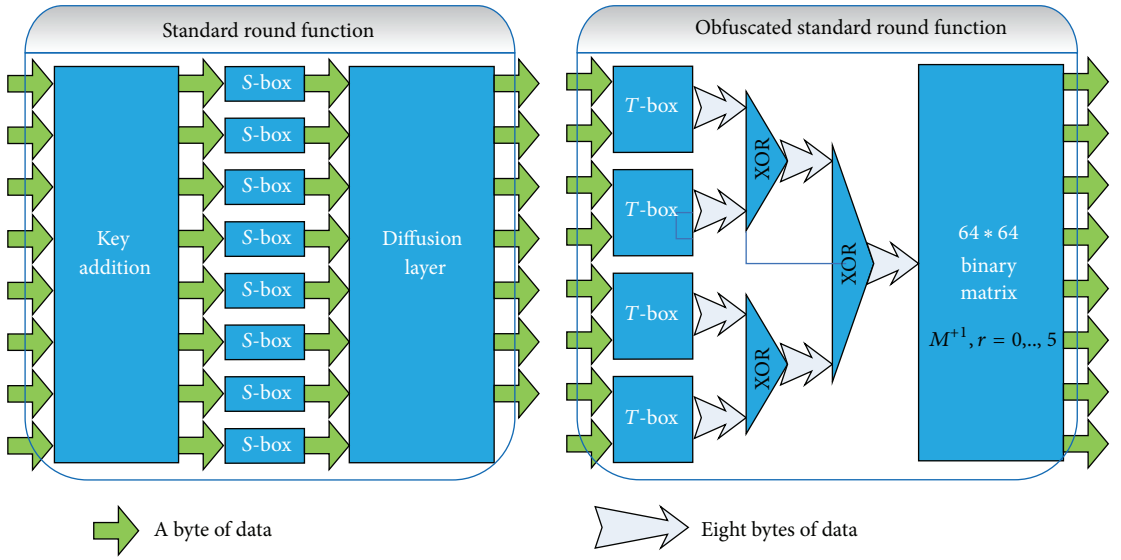


Figure 2: Data flow of the standard and the obfuscated implementation of the round functions for rounds 0 to 5.

Then, we can write the obfuscated subround functions as

$$\rho_W[r,i,K](x) = \left((S\|S)_{\Delta r}\left(\begin{matrix}(L_i^r(x))\\ \oplus(\Delta_r(k_{2i}^r\|k_{2i+1}^r))\end{matrix}\right)\right)P_i^r \quad r=0,\ldots,5;\ i=0,\ldots,3,$$

$$\rho_W[6,i,K](x) = \left(\left(L_i^6(x)\right)\oplus\left(\Delta_6\left(k_{2i}^6\left\|k_{2i+1}^6\right.\right)\right)\right)P_i^6, \quad i=0,\ldots,3, \tag{8}$$

where $S\|S$ refers to two S-boxes operating in parallel and $(S\|S)_{\Delta r} : GF(2^8)^2 \to GF(2^8)^2,\ x \mapsto \Delta_r((S\|S)(\Delta_r^{-1}(x)))$.

To invert the effect of the postround mixing matrices $P^{r-1,i}$, $r = 1,\ldots,6$, $i = 0,\ldots,3$ and the preround mixing matrices $Q^{r,i}$, $r = 0,\ldots,5$, $i = 0,\ldots,3$, a left multiplication of the matrix M^{r+1} is added at the end of each of rounds 0 to 5.

The data flows of our obfuscated implementation of round functions are shown in Figures 2 and 3.

As shown in Figure 3, the T-Boxes of the last round are lookup tables corresponding to the subround functions $\rho_W[6,i,K](x) = ((L_i^6(x)) \oplus (\Delta_6(k_{2i}^6\|k_{2i+1}^6)))P_i^6$, $i = 0,\ldots,3$. The nonlinear S-Boxes of other rounds have been removed. In fact, the T-Boxes of the last round are affine transformations $\rho_W[6,i,K](x) = (L_i^6(x)P_i^6) \oplus ((\Delta_6(k_{2i}^6\|k_{2i+1}^6))P_i^6)$, $i = 0,\ldots,3$. Clearly, this is dangerous, and so we modify the last round of SHARK as illustrated in Figure 4. Consequently, the T-Boxes of the last round should be $\rho_W[6,i,K](x) = ((S\|S)_{\Delta_6}((L_i^6(x)) \oplus (\Delta_6(k_{2i}^6\|k_{2i+1}^6))))P_i^6$, $i = 0,\ldots,3$. We call this modified version SHARK$'[K]$. Now, SHARK$[K]$ $= (\cdot H^{-1}) \circ (S_8^{-1}) \circ (\cdot H) \circ$ SHARK$'[K]$ where

$$S_8 = \mathop{\|}_{k=0}^{7} S_{\Delta_6}. \tag{9}$$

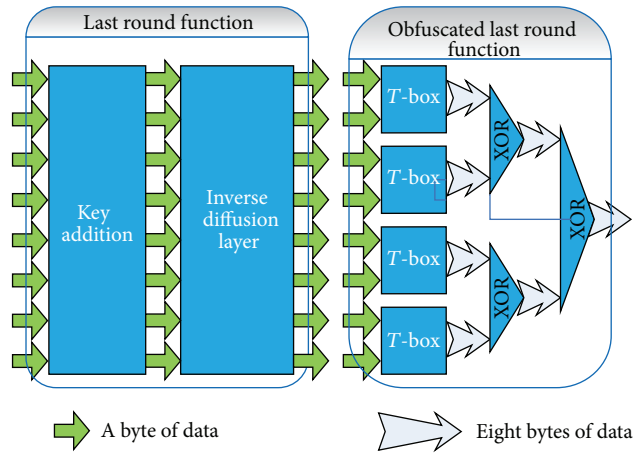


Figure 3: Data flow of the standard and the obfuscated implementation of round 6.

3.3. The Complete White-Box Encryption Algorithm. Using the components described in the previous section, the encryption process is shown in Algorithm 1.

We will now prove the soundness of our algorithm.

Proposition 1. *The encryption algorithm* SHARK$_W[K]$ *is such that*

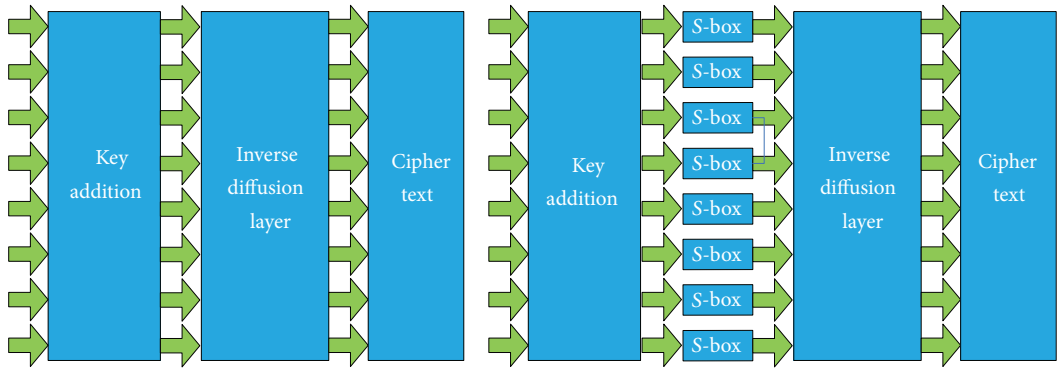
$$G \circ \text{SHARK}_W[K] \circ F = \text{SHARK}[K], \tag{10}$$

where

$$\text{SHARK}[K] = \lambda^{-1} \circ \sigma\left[K^6\right] \circ \left(\mathop{\circ}_{r=0}^{5} \rho\left[K^r\right]\right),$$

$$F = (\cdot U) \circ \left(\Delta_0^{-1}\left\|\Delta_0^{-1}\right\|\Delta_0^{-1}\left\|\Delta_0^{-1}\right.\right),$$

$$G = \left(\cdot H^{-1}\right) \circ \left(S_8\right)^{-1} \circ (\cdot H) \circ \left(\Delta_6^{-1}\left\|\Delta_6^{-1}\right\|\Delta_6^{-1}\left\|\Delta_6^{-1}\right.\right) \circ (\cdot V). \tag{11}$$



(a) The last round of SHARK (b) The modified last round of SHARK

Figure 4: Modification of the last round of SHARK.

(1) $i \leftarrow 0$
(2) $(x_0, x_1, x_2, x_3) \leftarrow x$
(3) $j \leftarrow 0$
(4) $y_j \leftarrow TBox_{i,j}(x_j)$ //*Lookup in a TBox*
(5) $j \leftarrow j + 1$
(6) if $(j < 4)$ goto (4); else goto (7)
(7) $x \leftarrow y_0 \oplus y_1 \oplus y_2 \oplus y_3$
(8) if $(i < 7)$ goto (9); else goto (11)
(9) $x \leftarrow x \cdot M_i$
(10) $i \leftarrow i + 1$; goto (2)
(11) output x

Algorithm 1: Algorithm $\mathrm{SHARK}_W[K]$ (on input x).

Proof. Let $x = (x_0, x_1, x_2, x_3)$, $x_i \in GF(2)^{16}$, $i = 0, 1, 2, 3$ be the input to the first round of $G \circ \mathrm{SHARK}_W[K] \circ F$. Then

$$\left(\sum_{i=0}^{3} \rho_W[0, i, K]\left(\left(\Delta_0^{-1}(x_i)\right) \cdot \left(Q^{0,i}\right)^{-1}\right)\right) \cdot M^1$$

$$= \left(\sum_{i=0}^{3} \left((S \| S)_{\Delta_0}\left(\begin{pmatrix} L_i^0\left(\left(\Delta_0^{-1}(x_i)\right) \cdot \left(Q^{0,i}\right)^{-1}\right) \\ \oplus \left(\Delta_0\left(k_{2i}^0 \| k_{2i+1}^0\right)\right) \end{pmatrix}\right)\right) P_i^0\right)$$

$$\cdot M^1 = \left(\sum_{i=0}^{3} \left((S \| S)_{\Delta_0} \times \left(\begin{pmatrix} \left(\Delta_0\left(\left(\Delta_0^{-1}(x_i)\right) \cdot \left(Q^{0,i}\right)^{-1} \cdot Q^{0,i}\right)\right) \\ \oplus \left(\Delta_0\left(k_{2i}^0 \| k_{2i+1}^0\right)\right) \end{pmatrix}\right)\right) \times P_i^0\right) \cdot M^1$$

$$= \left(\sum_{i=0}^{3} \Delta_0\left((S \| S)\left(x_i \oplus \left(k_{2i}^0 \| k_{2i+1}^0\right)\right)\right)\right) \cdot \left(\Delta_0(H)\right) \cdot N^0 \cdot M^1$$

$$= \left(\left(\Delta_0 \| \Delta_0 \| \Delta_0 \| \Delta_0\right)\left(\rho\left[K^0\right](x)\right)\right) \cdot N^0 \cdot \left(N^0\right)^{-1} \cdot \begin{bmatrix} \left(Q^{r,0}\right)^{-1} & & & \\ & \left(Q^{r,1}\right)^{-1} & & \\ & & \left(Q^{r,2}\right)^{-1} & \\ & & & \left(Q^{r,3}\right)^{-1} \end{bmatrix} \quad (12)$$

$$= \left(\left(\Delta_0 \| \Delta_0 \| \Delta_0 \| \Delta_0\right)\left(\rho\left[K^0\right](x)\right)\right) \cdot \begin{bmatrix} \left(Q^{r,0}\right)^{-1} & & & \\ & \left(Q^{r,1}\right)^{-1} & & \\ & & \left(Q^{r,2}\right)^{-1} & \\ & & & \left(Q^{r,3}\right)^{-1} \end{bmatrix},$$

where the round transformation ρ is defined in (3). We arrive at the last round by similar deductions on the previous rounds.

Let $y = \left(\overset{5}{\underset{r=0}{\circ}} \rho[K^r]\right)(x) = (y_0, y_1, y_2, y_3)$, $y_i \in GF(2)^{16}$, $i = 0, 1, 2, 3$ and define z to be the output of the fifth round of $G \circ \mathrm{SHARK}_W[K] \circ F$; that is,

$$z = \left(\left(\left(\Delta_5 \| \Delta_5 \| \Delta_5 \| \Delta_5\right) \circ \overset{5}{\underset{r=0}{\circ}} \rho\left[K^r\right]\right)(x)\right) \cdot \begin{bmatrix} \left(Q^{5,0}\right)^{-1} & & & \\ & \left(Q^{5,1}\right)^{-1} & & \\ & & \left(Q^{5,2}\right)^{-1} & \\ & & & \left(Q^{5,3}\right)^{-1} \end{bmatrix}$$

$$
\begin{aligned}
&= \left(\left(\Delta_5 \| \Delta_5 \| \Delta_5 \| \Delta_5\right)(y)\right) \\
&\quad \cdot \begin{bmatrix} \left(Q^{5,0}\right)^{-1} & & & \\ & \left(Q^{5,1}\right)^{-1} & & \\ & & \left(Q^{5,2}\right)^{-1} & \\ & & & \left(Q^{5,3}\right)^{-1} \end{bmatrix} \\
&= \Big(\left(\Delta_5(y_0)\right) \cdot \left(Q^{5,0}\right)^{-1}, \left(\Delta_5(y_1)\right) \cdot \left(Q^{5,1}\right)^{-1}, \\
&\qquad \left(\Delta_5(y_2)\right) \cdot \left(Q^{5,2}\right)^{-1}, \left(\Delta_5(y_3)\right) \cdot \left(Q^{5,3}\right)^{-1}\Big) \\
&= \left(z_0, z_1, z_2, z_3\right), \quad z_i \in GF(2)^{16}, \ i = 0, 1, 2, 3.
\end{aligned} \tag{13}
$$

Let $\rho[K^6] = \lambda^{-1} \circ \gamma \circ \sigma[K^6]$. The last round of $G \circ \mathrm{SHARK}_W[K] \circ F$ works on the output of previous round as follows:

$$
\begin{aligned}
&G\left(\sum_{i=0}^{3} \rho_W[6, i, K](z_i)\right) \\
&= G\left(\sum_{i=0}^{3} \left((S \| S)_{\Delta_6} \left(\begin{matrix} \left(L_i^6\left(\left(\Delta_5(y_i)\right) \cdot \left(Q^{5,i}\right)^{-1}\right)\right) \\ \oplus \left(\Delta_0\left(k_{2i}^6 \| k_{2i+1}^6\right)\right) \end{matrix}\right)\right) P_i^6\right) \\
&= G\left(\sum_{i=0}^{3} \left((S \| S)_{\Delta_6}\left(\left(\Delta_6(y_i)\right) \oplus \left(\Delta_6\left(k_{2i}^6 \| k_{2i+1}^6\right)\right)\right)\right) P_i^6\right) \\
&= G\left(\left(\sum_{i=0}^{3} \Delta_6\left((S \| S)\left(y_i \oplus \left(k_{2i}^0 \| k_{2i+1}^0\right)\right)\right)\right)\right. \\
&\qquad \left. \cdot \left(\Delta_6\left(H^{-1}\right)\right) \cdot N^6\right) \\
&= G\left(\left(\left(\Delta_6 \| \Delta_6 \| \Delta_6 \| \Delta_6\right)\left(\rho\left[K^6\right](y)\right)\right) \cdot N^6\right) \\
&= \left(\left(\cdot H^{-1}\right) \circ \left(S_8\right)^{-1} \circ (\cdot H) \circ \left(\Delta_6^{-1} \| \Delta_6^{-1} \| \Delta_6^{-1} \| \Delta_6^{-1}\right) \circ (\cdot V)\right) \\
&\quad \times \left(\left(\left(\Delta_6 \| \Delta_6 \| \Delta_6 \| \Delta_6\right)\left(\rho\left[K^6\right](y)\right)\right) \cdot N^6\right) \\
&= \left(\left(\cdot H^{-1}\right) \circ \left(S_8\right)^{-1} \circ (\cdot H) \circ \left(\Delta_6^{-1} \| \Delta_6^{-1} \| \Delta_6^{-1} \| \Delta_6^{-1}\right)\right) \\
&\quad \times \left(\left(\Delta_6 \| \Delta_6 \| \Delta_6 \| \Delta_6\right)\left(\rho\left[K^6\right](y)\right)\right) \\
&= \left(\left(\cdot H^{-1}\right) \circ \left(S_8\right)^{-1} \circ (\cdot H)\right)\left(\rho\left[K^6\right](y)\right) \\
&= \left(\lambda^{-1} \circ \sigma\left[K^6\right]\right)(y) \\
&= \left(\lambda^{-1} \circ \sigma\left[K^6\right] \circ \left(\mathop{\circ}_{r=0}^{5} \rho\left[K^r\right]\right)\right)(x) = \mathrm{SHARK}[K](x).
\end{aligned} \tag{14}
$$

This ends the proof. □

The following corollary shows how to decrypt the output of $\mathrm{SHARK}_W[K]$ by modifying, the decryption process of SHARK, that is, $\mathrm{SHARK}^{-1}[K]$.

Corollary 2. *The previously described encryption algorithm,* $\mathrm{SHARK}_W[K]$*, can be decrypted using*

$$
\mathrm{SHARK}_W[K]^{-1} = F \circ \mathrm{SHARK}^{-1}[K] \circ G. \tag{15}
$$

Proof. By Proposition 1, $G \circ \mathrm{SHARK}_W[K] \circ F = \mathrm{SHARK}[K]$. Hence,

$$
\begin{aligned}
\mathrm{SHARK}_W[K] &= G^{-1} \circ \mathrm{SHARK}[K] \circ F^{-1}, \\
\mathrm{SHARK}_W[K]^{-1} &= \left(G^{-1} \circ \mathrm{SHARK}[K] \circ F^{-1}\right)^{-1} \\
&= F \circ \mathrm{SHARK}[K]^{-1} \circ G \\
&= F \circ \mathrm{SHARK}^{-1}[K] \circ G.
\end{aligned} \tag{16}
$$

This ends the proof. □

4. Security Measurements and Analysis

4.1. Security Measurements. In [2, 3], Chow et al. used white-box diversity and white-box ambiguity to measure the security of a white-box encryption algorithm. The white-box diversity of a given table type counts the number of distinct constructions that exist in a table of the same type. It measures variability among implementations and is useful in foiling prepackaged attacks. White-box ambiguity of a table is a more important metric because it counts the number of distinct constructions that produce exactly the same type of table. It measures the number of alternative interpretations or meanings of a specific table, which an attacker must investigate in order to determine one of the obfuscated cipher's instances.

The number of nonsingular matrices of order n is $(2^n - 1) \times \prod_{j=1}^{n-1}\left(2^n - 1 - \sum_{k=1}^{j}\binom{j}{k}\right)$. The number of possible Δ is $8160 \approx 2^{13}$. For each table (T-Box), the white-box diversity is approximately $2^{255} \times 2^{13} \times 2^{16} \times 2^{13} \times 2^{255} = 2^{552}$, and the white-box ambiguity is approximately $2^{255} \times 2^{16} = 2^{271}$.

4.2. Against Billet et al.'s and Michiels et al.'s Attack. Billet et al. [8] described a very efficient attack against the white-box AES implementation proposed in [3]. Recovering information about the key by a local inspection of the lookup tables seems difficult, as the tables are designed to satisfy diversity and ambiguity criteria. In the Billet et al. attack, the authors take advantage of the fact that it is easier to recover information by analyzing compositions of lookup tables corresponding to one encoded AES round.

In this paper, the proposed implementation means that some attack techniques aimed at the simplicity of AES S-boxes are not valid. Furthermore, we have also used isomorphic transformations to increase the white-box diversity. For these reasons, the Billet et al. attack will not work.

The ideas presented in [3] can be used to derive a white-box implementation for any substitution linear-transformation network cipher [17]. Michiels et al. [9] presented an algorithm for extracting the round keys of such a cipher when all block rows of the diffusion matrices have disjoint spanning block sets. This condition on the diffusion matrices is, for example, satisfied by all maximum distance separable matrices [18, 19]. In our algorithm, we have implemented reverse

operations of linear mixing bijections in a different way. This ensures that our technique is immune from the attack of Michiels et al.

4.3. Against Mulder et al.'s Attack. Mulder et al. [13] presented a cryptanalysis of Xiao-Lai white-box AES implementation by using Biryukov et al.'s highly efficient linear equivalence algorithm [14]. The linear equivalence algorithm checks linear equivalence between two permutations (S-boxes), S_1 and S_2, and finds two invertible linear mappings, L_1 and L_2, such that $L_2 \circ S_1 \circ L_1 = S_2$. This is an important problem in symmetric cryptography.

Biryukov et al.'s linear equivalence algorithm exploits the following two ideas. The first is that we can guess portions of L_1, which will provide us with knowledge of the values of L_2. These new values from L_2 allow the algorithm to extract new information about L_1. The linear (affine) structure of the mappings causes another process, which they refer to as the exponential amplification of guesses. Their second idea is that if we know k vectors from the mapping L_1, we also know 2^k linear combinations of these vectors.

Mulder et al. proposed a modified version of the linear equivalence algorithm in [13]. The time complexity of solving the linear equivalence problem of a building block decreases from 2^{44} to 2^{29}. It follows that the attack efficiently extracts the AES key from Xiao-Lai white-box AES implementation with a time complexity of approximately 2^{32}. In the case of our white-box SHARK implementation, we have not found any technique that can reduce the time complexity in the same manner because of the following reasons.

(1) As shown in (17) and (18), the diffusion matrices of SHARK and AES are different.

(a) The diffusion matrix of SHARK is

$$\begin{bmatrix} \text{CE} & 95 & 57 & 82 & 8\text{A} & 19 & \text{B0} & 01 \\ \text{E7} & \text{FE} & 05 & \text{D2} & 52 & \text{C1} & 88 & \text{F1} \\ \text{B9} & \text{DA} & 4\text{D} & \text{D1} & 9\text{E} & 17 & 83 & 86 \\ \text{D0} & 9\text{D} & 26 & 2\text{C} & 5\text{D} & 9\text{F} & 6\text{D} & 75 \\ 52 & \text{A9} & 07 & 6\text{C} & \text{B9} & 8\text{F} & 70 & 17 \\ 87 & 28 & 3\text{A} & 5\text{A} & \text{F4} & 33 & 0\text{B} & 6\text{C} \\ 74 & 51 & 15 & \text{CF} & 09 & \text{A4} & 62 & 09 \\ 0\text{B} & 31 & 7\text{F} & 86 & \text{BE} & 05 & 83 & 34 \end{bmatrix}. \quad (17)$$

(b) The diffusion matrix of AES is

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix}. \quad (18)$$

(2) We use a different approach to compute M^r.

Furthermore, the Δ transformation that we use in this paper can provide a higher work factor. The overall work factor of Mulder, Roelse, and Preneel's attack against our white-box SHARK implementation is the product of following three factors:

(1) 2^{44} ($= n^3 2^{2n}$, $n = 16$) to solve the linear equivalence problem of a building block,

(2) 2^{13} (≈ 8160) to guess all the dual components,

(3) 2^2 because there are four building blocks in each round.

Thus, our white-box SHARK implementation remains with a security level higher than $2^{44+13+2} = 2^{59}$ against Mulder et al's attack.

Table 1: Number of operations in the algorithm SHARK$_W[K]$.

Operation	Number of operations	Formula
Bit multiplication	3×2^{13}	$= 6 \times 64 \times 64$
Bit addition	3×2^{13}	$\approx 6 \times 64 \times 64$
T-Box table lookup	28	$= 7 \times 4$

Table 2: Number of operations in the fast software implementation of SHARK$_W[K]$.

Operation	Number of operations	Formula
Multiplication table lookup	3×2^{10}	$= 6 \times 8 \times 64$
Bit addition	3×2^{10}	$\approx 6 \times 8 \times 64$
T-Box table lookup	28	$= 7 \times 4$

5. Size and Performance

In this section, we first analyze the size of static data that the algorithm requires. We then make some suggestions regarding the implementation and provide some experimental results. Finally, we discuss a highly efficient work mode for encrypting data.

Each round of our algorithm requires four T-Box tables. As the size of each table is $2^{16} \times 64$ bits $= 2^{19}$ bytes, the size of the 28 tables is 14 MB. The size of each matrix is 64×64 bits $= 2^9$ bytes. Thus, the size of these matrices is 3 KB. Combining these values, we determine that the size of all lookup tables and matrices is 14339 KB.

Three operations are needed to run the SHARK$_W[K]$ algorithm: bit multiplication, bit addition, and T-Box table lookup. We list the number of required operations in Table 1.

Of course, this is a "naïve" implementation as we can speed up the algorithm by using the memory-speed trade-off technique. A multiplication table can map two input bytes ($a_0, \ldots, a_7$ and $b_0, \ldots, b_7$) into a single bit $(a_0 \times b_0) \oplus (a_1 \times b_1) \oplus \cdots \oplus (a_7 \times b_7)$. With the help of such multiplication table, we can optimize the complexity of matrix multiplications and obtain a fast software implementation. The extra cost of memory is only 8 KB. This implementation requires three operations: multiplication table lookup, bit addition, and T-Box table lookup. Table 2 lists the required number of each operation.

We have investigated the time taken to encrypt 1 MB of data in the electronic codebook (ECB) mode on a ThinkPad notebook. The average time of the naïve implementation is 23.3 seconds and the average time of the fast implementation is only 1.2 seconds. Table 3 shows the details of the testing environment.

Clearly, the proposed algorithm is much slower than the standard algorithm because of the additional time taken when multiplying by $M^r, r = 1, \ldots, 6$. This is true even when

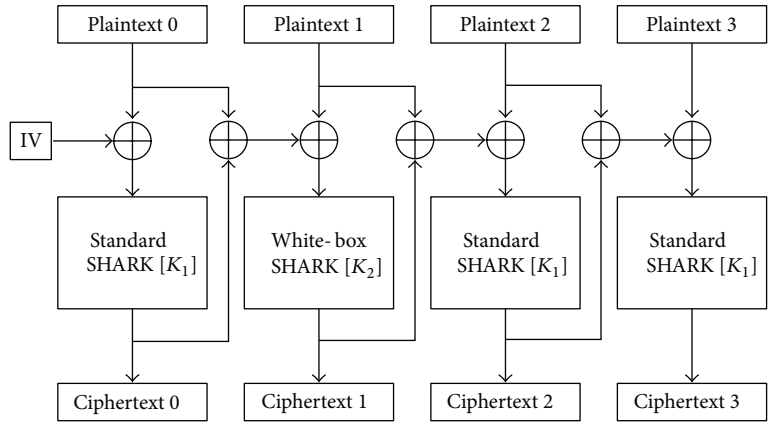


Figure 5: Flow of the white-box SHARK algorithm in composite PCBC mode.

Table 3: Details of testing environment.

Machine	Lenovo ThinkPad Carbon X1
CPU	Intel core i5-3317U 1.7 GHz/2.6 GHz
RAM	4 GB
OS	Window7 64 bit
JDK Version	7

using the fast implementation. But the proposed algorithm running in the composite propagating cipher-block chaining (PCBC) mode, as suggested by [20], is much faster than ECB mode. In the composite PCBC mode, the speed of encryption is almost the same as the standard implementation. Figure 5 shows the flow chart of the white-box SHARK algorithm running in the composite PCBC mode.

6. Conclusions and Discussion

In this paper, we propose a new white-box encryption algorithm that obfuscates the cipher SHARK. Our general approach is to merge several steps of the round function of SHARK into table lookups blended by randomly generated mixing bijections. Techniques used in [10, 12] are used in this paper to obtain the obfuscated cipher. Hence, this algorithm is secure against the attacks of Billet et al. [8], Michiels et al. [9], and Mulder et al. [13]. Thus, the algorithm is a countermeasure against the threat of key compromise in white-box attack context.

This design of white-box SHARK can also be used to obtain a white-box AES with a slight modification. The outcome of adapting our design to use AES will be a white-box AES implementation with the size of lookup tables and matrices being 20502 MB and with a security level of 2^{92}. We have chosen SHARK because it results in smaller tables and matrices and has a simpler description.

Future work should be focused on the size of the implementation. If we can significantly decrease the size, white-box encryption algorithms may be applied to lightweight applications such as the Internet of Things or wireless sensor networks.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (no. 61202382, no. 61103006) and the Fundamental Research Funds for the Central Universities.

References

[1] B. Wyseur, *White-box cryptography [Ph.D. thesis]*, Katholieke University, Leuven, Belgium, 2009.

[2] S. Chow, P. Eisen, H. Johnson, and P. C. Van Oorschot, "A white-box DES implementation for DRM applications," in *Proceedings of the 2nd ACM Workshop on Digital Rights Management*, vol. 2696 of *Lecture Notes in Computer Science*, pp. 1–15, Washington, DC, USA, November 2002.

[3] S. Chow, P. Eisen, H. Johnson, and P. C. Van Oorschot, "White-box cryptography and an AES implementation," in *Proceedings of the 9th Workshop on Selected Areas in Cryptography*, vol. 2595 of *Lecture Notes in Computer Science*, pp. 250–270, St. John's, Canada, 2003.

[4] M. Jacob, D. Boneh, and E. Felten, "Attacking an obfuscated cipher by injecting faults," in *Proceedings of the ACM Digital Rights Management Workshop*, vol. 2696 of *Lecture Notes in Computer Science*, pp. 16–31, Washington, DC, USA, November 2002.

[5] H. E. Link and W. D. Neumann, "Clarifying obfuscation: improving the security of white-box des," in *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC '05)*, pp. 679–684, Albuquerque, NM, USA, April 2005.

[6] B. Wyseur, W. Michiels, P. Gorisseii, and B. Preneel, "Cryptanalysis of white-box des implementations with arbitrary external encodings," in *Proceedings of the 14th Annual Workshop on Selected Areas in Cryptography*, vol. 4876 of *Lecture Notes in Computer Science*, pp. 264–277, Ottawa, Canada, August 2007.

[7] L. Goubin, J.-M. Masereel, and M. Quisquater, "Cryptanalysis of white box DES implementations," in *Proceedings of the 14th*

Annual Workshop on Selected Areas in Cryptography, vol. 4876 of *Lecture Notes in Computer Science*, pp. 278–295, Ottawa, Canada, August 2007.

[8] O. Billet, H. Gilbert, and C. Ech-Chatbi, "Cryptanalysis of a white box AES implementation," in *Proceedings of the 11th International Workshop on Selected Areas in Cryptography*, vol. 3357 of *Lecture Notes in Computer Science*, pp. 227–240, Waterloo, Canada, August 2005.

[9] W. Michiels, P. Gorissen, and H. D. L. Hollmann, "Cryptanalysis of a generic class of white-box implementations," in *Proceedings of the 15th International Workshop on Selected Areas in Cryptography*, vol. 5381 of *Lecture Notes in Computer Science*, pp. 414–428, Sackville, Canada, August 2008.

[10] M. Karroumi, "Protecting white-box AES with dual ciphers," in *Proceedings of the 13th International Conference on Information Security and Cryptology (ICISC '11)*, K. H. Rhee and D. Nyang, Eds., vol. 6829 of *Lecture Notes in Computer Science*, pp. 278–291, Seoul, Korea, 2011.

[11] L. Tolhuizen, "Improved cryptanalysis of an AES implementation," in *Proceedings of the 33rd WIC Symposium on Information Theory in the Benelux*, pp. 68–71, Boekelo, The Netherlands, 2012.

[12] Y. Xiao and X. Lai, "A secure implementation of white-box AES," in *Proceedings of the 2nd International Conference on Computer Science and Its Applications (CSA '09)*, pp. 410–415, eXpress Conference Publishing, Jeju, Korea, December 2009.

[13] Y. De Mulder, P. Roelse, and B. Preneel, "Cryptanalysis of the Xiao-Lai white-box AES implementation," in *Proceedings of the 19th Annual International Workshop on Selected Areas in Cryptography (SAC '13)*, vol. 7707 of *Lecture Notes in Computer Science*, pp. 34–49, Springer, 2013.

[14] A. Biryukov, C. De Cannière, A. Braeken, and B. Preneel, "A toolbox for cryptanalysis: linear and affine equivalence algorithms," in *Advances in Cryptology—EUROCRYPT*, E. Biham, Ed., vol. 2656 of *Lecture Notes in Computer Science*, pp. 33–50, Springer, Berlin, Germany, 2003.

[15] V. Rijmen, J. Daemen, B. Preneel, A. Bosselaers, and E. De Win, "The cipher SHARK," in *Proceedings of the 3rd International Workshop on Fast Software Encryption*, pp. 99–111, Cambridge, UK, February 1996.

[16] T. Jakobsen and L. Knudsen, "The interpolation attack on block ciphers," in *Proceedings of the 4th International Workshop on Fast Software Encryption*, pp. 28–40, Haifa, Israel, January 1997.

[17] J. Daemen, L. R. Knudsen, and V. Rijmen, "Linear frameworks for block ciphers," *Designs, Codes and Cryptography*, vol. 22, no. 1, pp. 65–87, 2001.

[18] M. F. Ezerman, M. Grassl, and P. Solé, "The weights in MDS codes," *IEEE Transactions on Information Theory*, vol. 57, no. 1, pp. 392–396, 2011.

[19] I. S. Kotsireas, C. Koukouvinos, and D. E. Simos, "MDS and near-MDS self-dual codes over large prime fields," *Advances in Mathematics of Communications*, vol. 3, no. 4, pp. 349–361, 2009.

[20] J.-Y. Park, O. Yi, and J.-S. Choi, "Methods for practical whitebox cryptography: a way to use dynamic key updates and high performance white box cryptography with certain mode of operations," in *Proceedings of the International Conference on Information and Communication Technology Convergence (ICTC '10)*, pp. 474–479, November 2010.