On the Key Leakage from Linear Transformations

Seungkwang Lee, Doyoung Chung, and Nam-su Jho

Information Security Research Division, ETRI
skwang@etri.re.kr

Abstract. Linear and non-linear transformations are often applied to the table-based cryptographic implementation in order to prevent key-dependent intermediate values from being analyzed. However, it has been shown that there is a correlation before and after the linear and non-linear transformations so that even a gray-box attacker can reveal secret keys hidden in a white-box cryptographic implementation. In this paper, we focus on the problem of linear transformations including the characteristics of block invertible binary matrices and the distribution of intermediate values. Our experimental results and proof show that the balanced distribution of the key-dependent intermediate value is the main cause of key leakage. Based on this observation, we find out that a random byte insertion in the intermediate values before linear transformations can eliminate a problematic correlation to the key.

Keywords: Power analysis, linear transformations, key leakage.

1 Introduction

From a secret key point of view, a block cipher can be seen as a secret bijective function between a plaintext set and a ciphertext set. One possible implementation of this function is a lookup table of the ciphertext for each of its corresponding plaintext. Since implementing a cipher as one lookup table is impractical because of its huge size, it is usually implemented as a series of lookup tables. This table-based implementation is also used in the white-box cryptographic implementation so that a secret key is not revealed by a white-box attacker observing internal memory and computing resources. The important thing to remember over here is that white-box cryptography generates key-instantiated lookup tables and protects each table with linear and non-linear transformations in order to prevent a key leakage from lookup values. Due to these transformations, a white-box attacker had been supposed to be unable to recognize a secret key via static or dynamic analysis.

So far, this white-box cryptography has been exploited by highly skilled attacks except for code lifting attacks [39] that steal the entire lookup table. For example, after the first two white-box DES (WB-DES) [10] and AES (WB-AES) [9] implementations were published, a number of practical cryptanalysis techniques [14][40] [3][21][26] have been introduced to extract the secret key from the white-box lookup tables. Many variants of WB-DES and WB-AES implementations [7][42] [17][20][22] were proposed and many were known to be practically broken [28][29]. Not only these white-box implementations of well-known standard block ciphers, dedicated white-box ciphers [4][5][27] have been also studied. One of dedicated cipher's shortcomings, however, is the lack of interoperability with the straightforward implementation of block ciphers. Recently, Differential Fault Analysis (DFA) [33] was introduced, where an attacker is able to inject a fault at a desired location in memory. Here, those white-box attacks rely on an in-depth understanding of a target implementation so that an attacker is able to gain read/write access to precise internal states during the execution. Thus commercial white-box cryptography [2][12][15][38] focuses on making a barrier to the full control of an attacker and is often combined with additional protection techniques including obfuscation, enveloping, hardware ID binding, and anti-debug protections. An explanation of these protections, primarily taking into account an untrusted environment, could be that the white-box implementation was without a doubt considered to defend against gray-box attacks, also known as side-channel attacks.

In contrast to white-box attacks, gray-box attacks are dependent on non-invasive information such as power consumption obtained while a target device performs cryptographic operations. For example, Differential Power Analysis (DPA) [19] is based on power consumption and this is one of the most well-known techniques to reveal the secret key imbedded in IC cards. Specifically, DPA is generally based on the fact that power consumption of a device is proportional or inversely proportional to the Hamming weight (HW) of data it processes. Thus a power analysis attacker collects a number of power traces with random plaintexts and finds a correct key that computes hypothetical values most highly correlated to the collected traces at a particular point. Currently, white-box cryptography can be easily broken by power analysis [6][34] without detailed knowledge of the target implementation. This means linear and non-linear transformations applied to lookup tables have no effect on hiding key-sensitive intermediate values.

In this study, we find out that the key leakage after linear transformations is largely due to the balanced distribution of intermediate values, and offer a simple proof of it. We also find out that a random byte insertion in the intermediate value before linear transformations prevents the key leakage. Furthermore, our experimental results show that inserting position does not make difference and inserting more than one byte provides no additional effect.

1.1 Organization of the Paper

The rest of this paper is organized as follows. Section 2 reviews some basic concepts including power analysis, and the key leakage issue in the linear transformation. In Section 3, we analyze the invertible linear transforms in detail to see why the key-dependent intermediate values are still correlated to the key even after linear transformations. Based on this analysis, we demonstrate that a random byte insertion before the linear transformation can eliminate the correlation to the key in Section 4. Finally, Section 5 concludes this paper.

2 Background

In this section, we introduce the basic concept of power analysis and and demonstrate the key leakage from the linear transformation with the Walsh transforms.

2.1 Power Analysis

Gray-box attacks, which were recently successful on white-box cryptography, are precisely power analysis such as DPA and CPA. An explanation of successful DPA and CPA on the white-box cryptographic implementation could be that attacker's correct hypothetical value will correlate to the target table lookup value. Note that DCA improves the efficiency of DPA and CPA attack since there is no measurement noise in the software execution traces, unlike the power consumption traces.

After collecting the traces with random plaintexts, DPA and CPA perform statistical analysis in different ways. DPA uses the selection function D to split the collected traces into two sets based on the attacker's hypothetical values. If the attacker's hypothetical key is correct (and therefore the hypothetical value is correct), then the trace separation by D is also accurate and there will be a peak in the differential trace.

In contrast, CPA uses a leakage model including the HW and the Hamming distance instead of the selection function D. When attacking a white-box implementation, the bit (mono-bit) model is appropriate because HW-based CPA attacks are unlikely to be successful due to the disturbed HW by linear and non-linear transformations. Given N power traces $V_{1..N}[1..\kappa]$ containing κ samples each, CPA will estimate the power consumption at each point of each trace using attacker's hypothetical intermediate value. For K different key candidates, let $\mathcal{E}_{n,k^*}$ ($1 \leq n \leq N$, $0 \leq k^* < K$) denote the power estimate in the n^{th} trace with the hypothetical key k^*. To measure a correlation between hypothetical power consumption and measured power traces, the estimator r is defined as follows [23]:

$$r_{k^*,j} = \frac{\sum_{n=1}^{N}(\mathcal{E}_{n,k^*} - \overline{\mathcal{E}_k^*}) \cdot (V_n[j] - \overline{V[j]})}{\sqrt{\sum_{n=1}^{N}(\mathcal{E}_{n,k^*} - \overline{\mathcal{E}_k^*})^2 \cdot \sum_{n=1}^{N}(V_n[j] - \overline{V[j]})^2}},$$

where $\overline{\mathcal{E}_k^*}$ and $\overline{V[j]}$ are sample means of $\mathcal{E}_k^*$ and $V[j]$, respectively. If there exists a correlation, a noticeable peak will be found in the correlation plot for the correct key.

Power analysis countermeasures can be categorized into masking and hiding, where masking breaks the correlation between power signals and the processed data while hiding reduces the signal to noise ratio. Maksing [1][11][13] [24][30][36] randomizes every key-dependent intermediate value by precomputing a new masked lookup table for each execution of encryption. To protect against higher-order DPA attacks [16][25][37], where an attacker exploits the joint key leakage from several intermediate values, higher-order DPA countermeasures have been studied [35][32] [18][8][31]. One of the most used hiding techniques, on the other hand, is introducing random delay. When the target cryptographic operation occurs uniformly distributed across n time instants due to random delay, the number of power traces for a successful DPA grows in n^2 only if DPA is performed straightforwardly. Here we can see these countermeasures are strongly dependent on expensive run-time random source, and also result in slow execution of cryptographic algorithm.

2.2 Detecting Key Leakage by the Walsh Transforms

Give a table-based implementation of a block cipher which is protected by table encoding, we can quantify or visualize a correlation using the Walsh transforms if a target lookup table is given. To understand how the Walsh transform can be used to quantify a correlation between the input and output of a target lookup table, we use the following definitions from [34].

Definition 1. *Let $x = \langle x_1, \ldots, x_n \rangle$, $\omega = \langle \omega_1, \ldots, \omega_n \rangle$ be elements of $\{0,1\}^n$ and $x \cdot \omega = x_1\omega_1 \oplus \ldots \oplus x_n\omega_n$. Let $f(x)$ be a Boolean function of n variables. Then the Walsh transform of the function $f(x)$ is a real valued function over $\{0,1\}^n$ that can be defined as $W_f(\omega) = \Sigma_{x \in \{0,1\}^n} (-1)^{f(x) \oplus x \cdot \omega}$.*

Definition 2. *Iff the Walsh transform W_f of a Boolean function $f(x_1, \ldots, x_n)$ satisfies $W_f(\omega) = 0$, for $0 \leq HW(\omega) \leq m$, it is called a balanced m^{th} order correlation immune function or an m-resilient function.*

Then we know that $W_f(\omega)$ quantifies the imbalances in the encoding, and the large absolute value of $W_f(\omega)$ means the strong correlation between $f(x)$ and $x \cdot \omega$. Using this property, we calculate the correlation between the table lookup values and hypothetical values.

Let's demonstrate the key leakage from the encoded lookup table generated by the composition of S-box and AddRoundKey in the first round of AES. Given a subkey $k^0_{0,2} = 0x88$, and for every input value $p \in \mathrm{GF}(2^8)$, we know that

$$\begin{aligned} x &= S(p \oplus k^0_{0,j}) \\ y_0(x) &= \left[2 \cdot x \; x \; x \; 3 \cdot x\right]^T \end{aligned}$$

where S represents the AES SubBytes and the multiplication by 2 is implemented as a 1-bit left shift followed by a conditional ($\oplus$ 0x1B) if the MSB of the operand was 1. Then $f(x)$ here denotes the lookup values computed from linear and non-linear transformations on $y(x)$, and we have 32 Boolean functions $f_{i \in \{1,\ldots,32\}}(x)$: $\{0,1\}^8 \rightarrow \{0,1\}$. To find a correct key, we calculate the Walsh transforms W_{f_i}

and sum all the imbalances for each key candidate and ω such that $\mathrm{HW}(\omega) = 1$ as follows:

$$\Delta^f_{k\in\{0,1\}^8} = \sum_{\omega=1,2,4,\ldots,128} \sum_{i=1,\ldots,32} |W_{f_i}(\omega)|.$$

The reason why we only select ω of $\mathrm{HW}(\omega) = 1$ is that the HW-based key leakage model is not effective to detect the correlation between the in/output of the encoding.

The Walsh transforms and their sum of all imbalances are given in Fig. 1. As we can see in Fig. 1a, the Walsh transforms with $\omega = 4$ of the correct key ($\mathit{0x}88$) produce 0 except two points; the W_{f14} and W_{f16} of the correct key are -128, and their absolute value (128) is the most highest value. In contrast the maximum and the average values of $|W_{f_i}(\omega)|$ of wrong key candidates are 56 and about 13.13 (the standard deviation is about 9.35), respectively. This gives us that $f_{14}(\cdot)$ and $f_{16}(\cdot)$ cause key leakages and thus power analysis using the 3^{rd} bit (when the LSB is the 1^{st} bit) of attacker's hypothetical SubBytes outputs is able to recover this subkey. $\Delta^f_{k=\mathit{0x}88}$ is 256 ($= |-128| + |-128|$) which is obviously distinguishable from that of other key candidates as shown in Fig. 1b while $\Delta^f_{k\neq \mathit{0x}88}$ are about 2900-3700. This simply shows us how to use the sum of all imbalances for recovering the correct key.

In the similar way, we can detect a key leakage from the final round input of a table-based implementation of AES. In this scenario, we give the first subkey of the ninth round key ($\mathit{0x}54$) and let the attacker guess the first subkey ($\mathit{0x}13$) of the final round. Given a subbyte of the ciphertext, x becomes the attacker's hypothetical input value and 8 Boolean functions $f_{i\in\{1,\ldots,8\}}(x)$: $\{0,1\}^8 \to \{0,1\}$ indicate the encoded input to the final round lookup table. On the condition that there is no external encoding on the ciphertext and the attacker knows the first subkey of the ninth and final round keys, $\Delta^f_{k=\mathit{0x}13}$ in the final round is 4096; this is much smaller than $\Delta^f_{k\neq \mathit{0x}13}$ in the range 25900 - 26500 as shown in Fig. 2. In the following section, we analyze the linear transformations used in the lookup table generation, and provide a clue for a secure implementation.

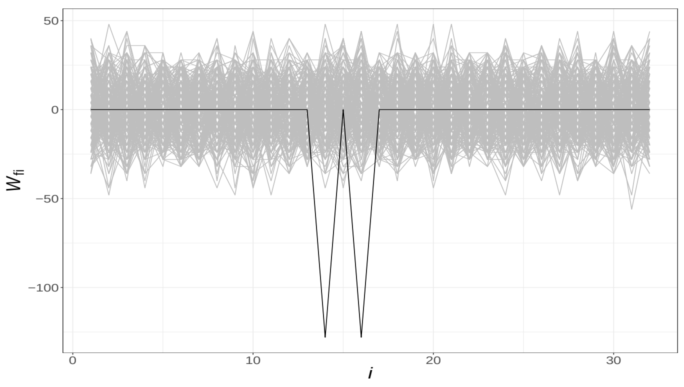
3 Analysis of Linear Transformations

3.1 Key Leakage Statistics after Linear Transformations

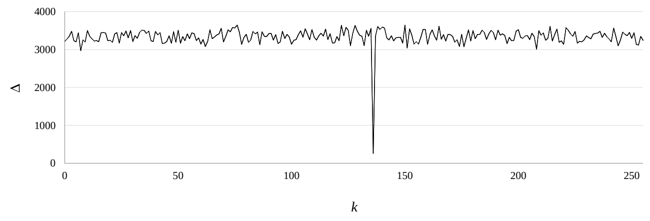
We begin with an experimental result of a key leakage at the linear transformation demonstrated by the sum of imbalance depicted in Fig. 3, where the Walsh transforms use

$$f(x) = M \cdot y_{i\in\{0,1,2,3\}}(x),$$

for x of each y_i computed from four subkeys, the 9^{th} to 12^{th} subkeys ($\mathit{0x}88$, $\mathit{0x}99$, $\mathit{0x}\mathrm{AA}$, $\mathit{0x}\mathrm{BB}$) of the first round in this experiment, and a 32×32 binary invertible matrix M. Unlike in the case of Fig. 1b of a key leakage from the linear and non-linear transformations, this shows a key leakage from linear transformations without non-linear transformations. We can see that linear transformations



(a) Walsh transforms for $f_{i\in\{1,\cdots,32\}}(\cdot)$ with $\omega = 4$ for all key candidates. Gray: wrong key candidates; Black: correct key.



(b) Sum of all imbalances for all key candidates.

Fig. 1: Key leakage detection using the Walsh transforms.

with M can hide three subkeys $\mathit{0x}88$, $\mathit{0x}\mathrm{AA}$, and $\mathit{0x}\mathrm{BB}$, but expose one subkey $\mathit{0x}99$ from $y_1(x)$. This gives us two facts. First, as we will show later in this section, linear transformations produce well-balanced output with an overwhelming probability, but this is not always guarantee a reliable protection on secret keys. Second, the correct key can be recovered by the Walsh transforms even if its sum of imbalances is 0 indicating no correlation because it is distinguishable.

Table 1 and Fig. 4 show our experimental results of linear transformations on $y_{i\in\{0,1,2,3\}}(x)$ using 1000 randomly generated invertible matrices. For HW(ω) = 1, $Wf_i(\omega) = 0$ with approximately 99.7% and 0.3 % of $Wf_i(\omega) = 256$; the average of $|W_{f_i}(\omega)|$ is approximately 0.7. We will proof later there is no other $Wf_i(\omega)$ values. Here, both cases (0 and 256) will lead to two different types of key leakages due to the distinguishable Walsh transform value and the noticeably high correlation coefficient, as pointed out previously. For there are 8 values of $\omega \in \mathrm{GF}(2^8)$ such that HW(ω) = 1 and y_0 - y_3 output 32-bit values, 1024 Wf_i will be tested to see if there exists a key leakage from the linear transformation using a given matrix M. Consequently, there probably exist about 3 peaks of the correct key distinguishable from wrong key candidates, and the 3 peaks can reveal 1 to 3 subkeys. Each of y_0, y_1, y_2, and y_3 shows around 1/2 probability of $\Delta^f_{k^c} = 0$, and only about 5% of matrices do not leak any subkeys after linear

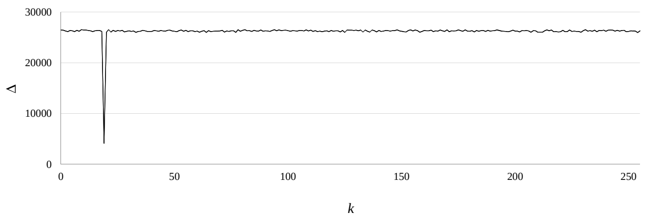


Fig. 2: Sum of the imbalance for all key candidates in the final round input.

Table 1: Experimental results of linear transformations with 1000 randomly generated block invertible matrices. k^c: correct key.

	Vectors to be transformed			
Number of	y_0	y_1	y_2	y_3
$Wf_i(\omega) = 0$	255,206	255,205	255,309	255,203
$Wf_i(\omega) = 256$	794	795	691	797
$\Delta^f_{k^c} = 0$	475	489	520	464
$\Delta^f_{k^c} = 256$	333	307	316	343
$\Delta^f_{k^c} = 512$	132	144	122	146
$\Delta^f_{k^c} > 512$	60	60	42	47

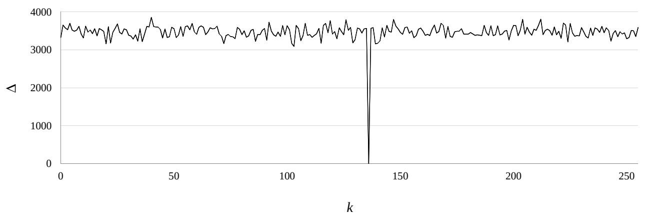
transformations, where k^c means the correct key. In most cases, 1 to 3 out of four subkeys are shown to be exposed.

From now on, we are going to analyze this problematic characteristic of the linear transformations that produce extreme Wf_i values of 0 or 256. The first thing we want to investigate is whether the invertible matrix is responsible for this matter.

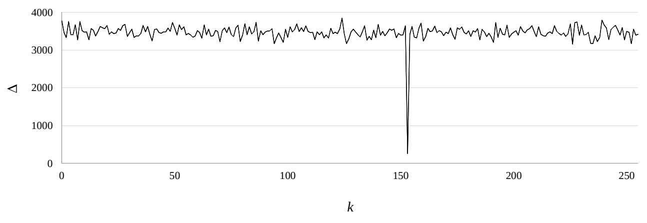
3.2 Analysis of Block Invertible Square Matrix

In [9], the authors choose M as a non-singular matrix with submatrices of full rank with a reference to [41] for maximizing information diffusion. To begin with, we briefly review the definition of a block invertible square matrix.

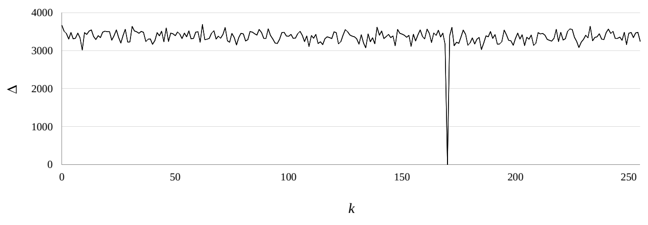
Definition 3. *If all the blocks $B_{i,j}$ in a block matrix ${}^{n}_{m}M[{}^{p}B]$ are invertible, matrix M is called an (m, n, p) block invertible matrix. Furthermore, if $m = n$, and M is invertible then M is called an (m, p) block invertible square matrix, where ${}^{n}_{m}M[{}^{p}B]$ denotes an $n \times m$ matrix M with nm/p^2 blocks (submatrices), and $B_{i,j}$ denotes the block in row i and column j of blocks [41].*



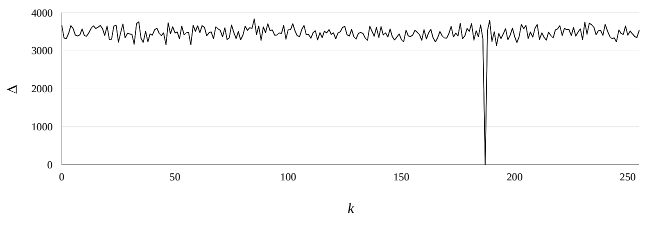
(a) On $M \cdot y_0(x)$



(b) On $M \cdot y_1(x)$



(c) On $M \cdot y_2(x)$



(d) On $M \cdot y_3(x)$

Fig. 3: Sum of the imbalance of $W_{f_i}(\omega)$ for all key candidates on each $y_{i\in\{0,1,2,3\}}(x)$ with only linear transformations.

Generating $(n,\ 2)$ block invertible square matrices begins with a $(2,\ 2)$ block invertible square matrix and extends by $(4,\ 2)$, $(6,\ 2)$, $\ldots$, and repeats it $(n-2)/2$ times. The important point over here is that every 2×2 submatrix in a $(n,\ 2)$ block invertible square matrix should be invertible by the definition and all 2×2

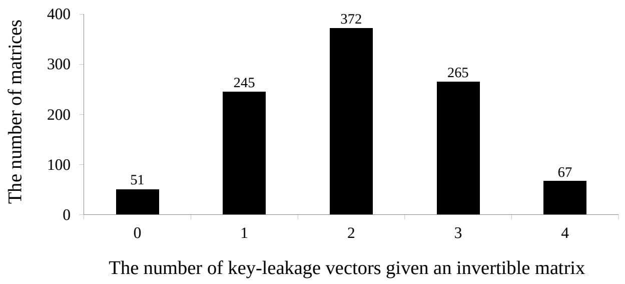


Fig. 4: The number of block invertible matrices (y-axis) vs. the number of key-leakage vectors among $y_{i\in\{0,1,2,3\}}$ given a block invertible matrix (x-axis).

invertible matrices in GF(2) are as follows:

$$\begin{vmatrix}1&0\\0&1\end{vmatrix}\quad\begin{vmatrix}1&1\\1&0\end{vmatrix}\quad\begin{vmatrix}0&1\\1&1\end{vmatrix}\quad\begin{vmatrix}0&1\\1&0\end{vmatrix}\quad\begin{vmatrix}1&1\\0&1\end{vmatrix}\quad\begin{vmatrix}1&0\\1&1\end{vmatrix}$$

At a glance, the number of 1s in the 4 out of 6 matrices is greater than 0s. By the principle of constructing a block invertible square matrix, the HW of each row and column in an $(n,\,2)$ block invertible matrix will be greater than $n/2$. For example, let's assume that a (4, 2) matrix is initialized with

$$\begin{vmatrix}1&0\\0&1\end{vmatrix},$$

then its resulting matrix will be

$$\begin{vmatrix}1&0&1&0\\0&1&0&1\\1&0&1&1\\0&1&1&0\end{vmatrix}.$$

In the case of an initialization with

$$\begin{vmatrix}0&1\\1&0\end{vmatrix},$$

we will have

$$\begin{vmatrix}0&1&0&1\\1&0&1&0\\0&1&1&0\\1&0&1&1\end{vmatrix}.$$

During the generation of a (32, 2) matrix through this process, 1s appear more frequently. We have performed the following experiment to check if this over-weight HW of the invertible block square matrix is the main reason for key leakage. We randomly generated a balanced *non-invertible* 32×32 matrix M^b, such that $f(x) = M^b \cdot y_{i\in\{0,1,2,3\}}(x)$, where M^b has the HW of 16 for each row and column, and used it to compute the sum of imbalances. As shown in Fig. 5, there still exist key leakages from y_1 and y_2 with $\Delta^f_{k^c} = 256$. For this reason, we can conclude that the matrix HW itself is not the cause of key leakages from linear transformations.

3.3 Analysis of Key-dependent Intermediate Values

The next key-leakage point to be analyzed is y. From Definition 1 and 2, we know that a balanced correlation immune function is strongly dependent on the distribution of $f_i(x) \oplus x \cdot \omega$. Since a matrix characteristic is not responsible for the key leakage as we analyzed previously, the distribution of y is convinced to mainly decide the distribution of $f_i(x) \oplus x \cdot \omega$. Here recall that given a key-dependent value $x \in \mathrm{GF}(2^8)$ and 1000 randomly generated invertible matrices M, $W_{f_i}(\omega) = 0$ with approximately 99.7% while only 0.3% of $W_{f_i}(\omega) = 256$, where $\mathrm{HW}(\omega) = 1$. The following proof explains the reason behind.

Lemma 1. *Assume that a 256×8 binary matrix* $\mathbf{H}$ *is defined as*

$$\mathbf{H} = \begin{bmatrix} h_{1,1} & h_{1,2} & \cdots \\ \vdots & \ddots & \\ h_{256,1} & & h_{256,8} \end{bmatrix}$$

where i^{th} *row vector* $\mathbf{h}_{i,*} = \langle h_{i,1}, h_{i,2}, \ldots, h_{i,8} \rangle$ *is an element of* $GF(2^8)$ *and* $\mathbf{h}_{i,*} \neq \mathbf{h}_{j,*}$ *for all* $i \neq j$. *Then the HW of XORs of arbitrary chosen column vectors from* H *is 0 or 128. In other words,* $HW(\mathbf{h}_{*,j_1} \oplus \mathbf{h}_{*,j_2} \oplus \cdots \oplus \mathbf{h}_{*,j_n}) =$ *0 or 128, where* n *is a random positive integer and* $j_i \in \{1, 2, \ldots, 8\}$.

Proof : Let $\mathcal{J}$ be a set of randomly chosen indices from $\{1, 2, \ldots, 8\}$. Note that for any duplicated indices α and α' in $\mathcal{J}$, i.e. $\alpha = \alpha'$, removing the duplicated indices from $\mathcal{J}$ makes no change to the result HW.

$$\oplus_{j\in\mathcal{J}}\mathbf{h}_{*,j} = \big(\oplus_{j\in\mathcal{J}-\{\alpha,\alpha'\}}\,\mathbf{h}_{*,j}\big) \oplus \mathbf{h}_{*,\alpha} \oplus \mathbf{h}_{*,\alpha'}$$

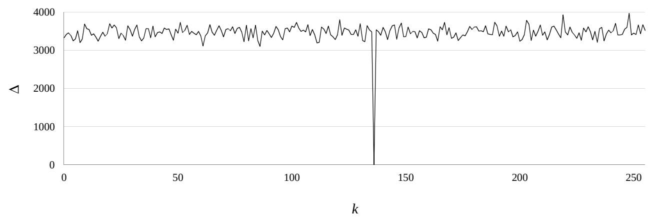
$$= \big(\oplus_{j\in\mathcal{J}-\{\alpha,\alpha'\}}\,\mathbf{h}_{*,j}\big) \oplus \mathbf{0} = \oplus_{j\in\mathcal{J}-\{\alpha,\alpha'\}}\mathbf{h}_{*,j}.$$

Therefore without loss of generality we can assume that $\mathcal{J}$ contains no duplicated indices and moreover $\big|\mathcal{J}\big| = n \leq 8$.

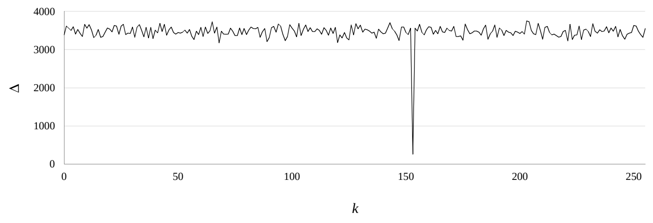
Now we can define following partitions of indices:

$$\mathcal{I}_{b_1,b_2,\ldots,b_n} = \{\ell \in \mathcal{I} | h_{\ell,j_i} = b_i \text{ for all } j_i \in \mathcal{J}\},$$

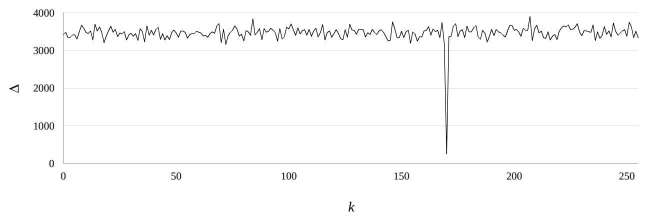
where $\mathcal{I} = \{1, 2, \ldots, 256\}$, and $b_i \in \{0, 1\}$. Here all $\mathcal{I}_{b_1,b_2,\ldots,b_n}$ are disjoint to the others and $\cup\mathcal{I}_{b_1,b_2,\ldots,b_n} = \mathcal{I}$. To complete the proof, we need that for any choice



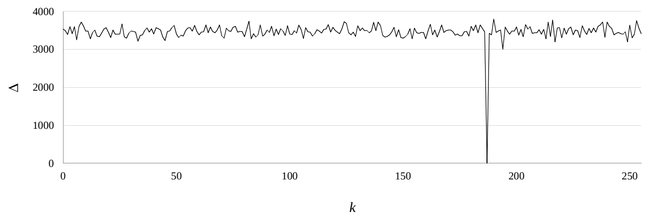
(a) On $M^b \cdot y_0(x)$



(b) On $M^b \cdot y_1(x)$



(c) On $M^b \cdot y_2(x)$



(d) On $M^b \cdot y_3(x)$

Fig. 5: Sum of the imbalance for all key candidates on each $y_{i\in\{0,1,2,3\}}(x)$ multiplied with a balanced matrix M^b.

of b_i's, $\left|\mathcal{I}_{b_1,b_2,\ldots,b_n}\right| = 256/2^n = 2^{8-n}$. This can be shown easily as followings. Suppose that $\left|\mathcal{I}_{b_1,b_2,\ldots,b_n}\right| = t > 2^{8-n}$. It means that there are t row vectors in $\mathbf{H}$ satisfying the condition j_i-th bit of the vector equals to b_i. In other words, n bits are determined by choice of b_i's and only $8-n$ bits are remained free. From the condition of t is larger than 2^{8-n} and the pigeon hole principle in mathematics, there must exist at least two indices ℓ and ℓ' in $\mathcal{I}_{b_1,b_2,...,b_n}$, where all bits of $\mathbf{h}_{\ell,*}$ are completely same to the bits of $\mathbf{h}_{\ell',*}$. It contradicts to the assumption $\mathbf{h}_{i,*} \neq \mathbf{h}_{j,*}$ for any $i \neq j$.

From the definition of HW, we can deduce $HW(\oplus_{j\in\mathcal{J}}\mathbf{h}_{*,j})$ is summation of $\left|\mathcal{I}_{b_1,b_2,...,b_n}\right|$ where $\oplus_{i=1,...,n}b_i = 1$.

$$HW(\oplus_{j\in\mathcal{J}}\mathbf{h}_{*,j}) = \Sigma_{\oplus_{i=1,\dots,n}b_i=1}\left|\mathcal{I}_{b_1,b_2,...,b_n}\right|$$

$$= \Sigma_{\oplus_{i=1,\dots,n}b_i=1}2^{8-n} = \Sigma_{2^{n-1}}2^{8-n}$$

$$= 2^{n-1}\cdot 2^{8-n} = 2^7 = 128.$$

Note that if $\mathcal{J}$ is empty after de-duplication then the final HW becomes 0. It concludes the proof of lemma.

Note that $W_{f_i}(w)$ is defined as $\sum_{x\in GF(2^8)}(-1)^{f_i(x)\oplus w\cdot x} = \sum_{x\in\{0,1\}^8}(-1)^{M_{i,*}\cdot y(x)\oplus w\cdot x}$, where $M_{i,*}$ is i^{th} row of the matrix M and $y(x)$ is one of $y_0(x)$ - $y_3(x)$ depending on the target subkey. For convenience, let $y(x) = y_0(x)$, a 32×1 matrix $[2\cdot x \;\; x \;\; x \;\; 3\cdot x]^T$. If we define $\mathbf{Y}(x)$ as a 32×256 matrix $[2\cdot\mathbf{H} \;\; \mathbf{H} \;\; \mathbf{H} \;\; 3\cdot\mathbf{H}]^T$, where the $\mathbf{H}$ is the matrix defined at the lemma 1, it is easy to show that each column vector of $2\cdot\mathbf{H}$ or $3\cdot\mathbf{H}$ can be defined with XORs of some column vectors of $\mathbf{H}$ based on the property of $GF(2^8)$. Then the above equation can be re-written as

$$\sum_{j=\{1,2,\dots 256\}}(-1)^{B_j(M_{i,*}\cdot\mathbf{Y}(x)\oplus(w\cdot\mathbf{H}^T))},$$

where $B_j(v)$ means the j^{th} bit of the vector v. Since the exponents of the equation can have only two values 0 or 1, the summation over $\{1,2,\dots,256\}$ can be re-written with the number of exponents which are 1.

$$W_{f_i}(w) = 256 - (2\times HW(M_{i,*}\cdot\mathbf{Y}(x)\oplus(w\cdot\mathbf{H}^T)))$$

Note that all row vectors of the matrix $\mathbf{Y}(x)$ is represented by XORing of column vectors of $\mathbf{H}$. Therefore $M_{i,*}\cdot\mathbf{Y}(x)\oplus(w\cdot\mathbf{H}^T)$ can be also represented by XORing of column vectors of $\mathbf{H}$. From the lemma 1, it deduces that the HW of $M_{i,*}\cdot\mathbf{Y}(x)\oplus(w\cdot\mathbf{H}^T)$ is 0 or 128. Finally, $W_{f_i}(w) = 256 - (2\times HW(M_{i,*}\cdot\mathbf{Y}(x)\oplus(w\cdot\mathbf{H}^T)))$ becomes 256 or 0. What is remarkable point over here is that the probability of $W_{f_i}(w) = 256$ is very small but not zero. Specifically, it happens when all column indices of $\mathbf{H}$ are canceled each other when the summation is computed with the randomly chosen matrix M.

As mentioned already, our experiment showed that $W_{f_i}(w) = 256$ with 0.3% in the calculation with the correct key, while the wrong key candidates produced $|W_{f_i}(\omega)| = 56$ at maximum and 13.13 in average. For this reason, 1024 tests of $W_{f_i}(w)$ given a matrix M are likely to cause key leakages with overwhelming probability. Based on these findings, we propose a novel approach for a secure implementation in the following section.

4 Inserting A Random Byte in the Intermediate Values

Our analysis in the previous section shows that a balanced distribution of the intermediate values is the main reason behind the key leakage. In order to make it unbalanced, our key idea is to insert random bytes in the intermediate values before linear transformations. From now on, we answer to the following questions.

– Where is the appropriate position for random bytes to be inserted?
– How many random bytes must be inserted?

We begin with an analysis of the inserting position and the required number of random bytes to be inserted.

First, we will insert a random byte at a particular position in the 4-byte intermediate value $y_{i\in\{0,1,2,3\}}(x)$ and then perform a linear transformation with a 40×40 binary block invertible matrix M^* to check if any key leakage occurs. Among the five inserting positions ρ_1 - ρ_5 of y_0, for example,

$$\left[\rho_1 \; 2\cdot x \; \rho_2 \; x \; \rho_3 \; x \; \rho_4 \; 3\cdot x \; \rho_5\right]^T$$

we select ρ_i, where $i \in$ [1, 5], and then insert different $\gamma \in_R$ GF(2^8) at ρ_i for each $x \in$ GF(2^8). Let $y_0^*(x)$ denote $y_0(x)$ after the random byte insertion, and $f^*(x)$ denote $y_0^*(x) \cdot M^*$. Then we can define the Walsh transforms with respect to f^*:

$$W_{f_i^*}(\omega) = \Sigma_{x\in\{0,1\}^8}(-1)^{f_i^*(x)\oplus x\cdot\omega}$$

for 40 Boolean functions

$$f^*_{i\in\{1,\ldots,40\}}(x) : \{0,1\}^8 \to \{0,1\}.$$

With 1000 randomly generated M^*, we computed $W_{f_i^*}(\omega)$ with respect to y_0 - $y_3(x)$ for each position ρ_i. As a result, Table 2 gives us that the correct key results in $W_{f_i^*}(\omega) = 0$ with approximately 5% and the average $|W_{f_i^*}(\omega)|$ is about 12.7. Recall that, without the random byte insertion, $W_{f_i}(\omega) = 0$ with approximately 99.7% and the average of $|W_{f_i}(\omega)|$ is approximately 0.7.

	ρ_1	ρ_2	ρ_3	ρ_4	ρ_5
% of $W_{f_i^*}(\omega) = 0$	5.05 (0.03)	5.06 (0.07)	4.93 (0.05)	5.0 (0.05)	5.04 (0.04)
Average of $\lvert W_{f_i^*}(\omega)\rvert$	12.73 (0.02)	12.75 (0.01)	12.76 (0.01)	12.73 (0.01)	12.76 (0.01)
Similarity with $W_{f_i^\gamma}$	> 0.999				

Table 2: $W_{f_i^*}$ after inserting a random byte at each inserting position (the standard deviation in parenthesis), and the cosine similarity of the distributions between $W_{f_i^*}$ and $W_{f_i^\gamma}$.

To see the effect of the random byte insertion, we conducted an additional experiment as follows.

1. Let $y^\gamma(x) = [\gamma 1\ \gamma 2\ \gamma 3\ \gamma 4\ \gamma 5]^T$ for each $x \in \mathrm{GF}(2^8)$. In other words, replace all the key-dependent intermediate values with random bytes.
2. $f^\gamma(x) = M^* \cdot y^\gamma(x)$.
3. Repeat step (1) - (2) with 1000 random M^* matrices, and accumulate the number of occurrences of each value of $W_{f_i^\gamma}(\omega)$.
4. Compute % of $W_{f_i^\gamma}(\omega) = 0$ and the average $|W_{f_i^\gamma}(\omega)|$.
5. Compute the cosine similarity between the distributions of $W_{f_i^\gamma}(\omega)$ and $W_{f_i^*}(\omega)$ for each of y_0 - y_3 and for each ρ_i.

As a result, we have $W_{f_i^\gamma}(\omega) = 0$ with approximately 5%, the average $|W_{f_i^\gamma}(\omega)|$ is approximately 12.74, and the cosine similarity between their distributions is always larger than 0.999. The cosine similarity larger than 0.99 means they show very similar distribution. We note that the cosine similarity between the distributions of $W_{f_i^\gamma}(\omega)$ and $W_{f_i}(\omega)$ is about 0.25.

In order to visualize this effect of inserting a random byte, we select ρ_5 and calcalculate the sum of the imbalances of $W_{f_i^*}(\omega)$ for each key candidate with ω such that $\mathrm{HW}(\omega) = 1$ as follows:

$$\Delta^{f^*}_{k\in\{0,1\}^8} = \sum_{\omega=1,2,\ldots,128}\ \sum_{i=1,\ldots,40} |W_{f_i^*}(\omega)|,$$

Fig. 6 shows $\Delta^{f^*}_{k\in\{0,1\}^8}$ and we can see that the correct subkeys *0x*88 - *0x*BB are no longer distinguishable from other candidates.

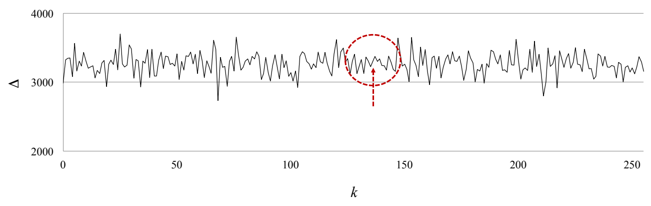
In addition, it is noticeable that inserting more than one random byte in the intermediate values does not increase the imbalance; they show a similar level of the imbalance of the one-byte insertion. Thus, we can conclude that inserting a random byte into anywhere among ρ_1-ρ_5 can effectively prevent the key leakage after the linear transformation.

5 Conclusion

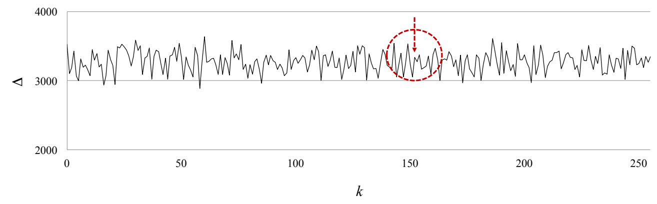
In this paper, our analysis shows the well distributed intermediate values cause the key leakage after linear transformation. Based on this analysis, we introduce a novel approach of inserting a random byte into intermediate values before linear transformations to prevent key leakages. We can apply this insertion of a random byte to design a power analysis countermeasure without using nonlinear transformations. In this case, the XOR lookup table is not needed and a normal XOR operation can be performed due to the distributive property of multiplication over addition. However, a 40×40 linear transformation must be replaced with a 16×16 transformation which is also including a random byte. The main disadvantage of this case is that the input size of two bytes for the next round input can significantly increase the lookup table size.

References

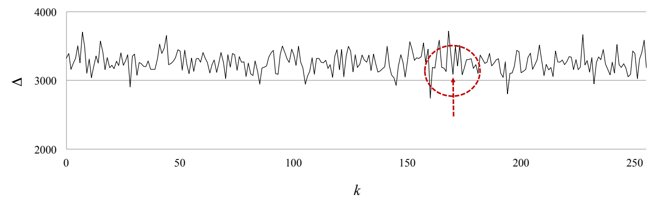
1. Akkar, M.L., Giraud, C.: An Implementation of DES and AES, Secure against Some Attacks. In: Proceedings of the Third International Workshop on Cryp-



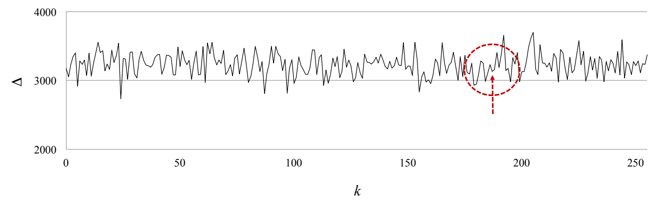
(a) On $M^* \cdot y_0(x)$



(b) On $M^* \cdot y_1(x)$



(c) On $M^* \cdot y_2(x)$



(d) On $M^* \cdot y_3(x)$

Fig. 6: Sum of the imbalance of $W_{f_i^*}(\omega)$ for all key candidates. Red arrow: the correct key.

tographic Hardware and Embedded Systems. pp. 309–318. CHES '01, Springer-Verlag, London, UK, UK (2001)

2. Axsan white-box cryptographic solution.: https://www.arxan.com/technology/white-box-cryptography/

3. Billet, O., Gilbert, H., Ech-Chatbi, C.: Cryptanalysis of a White Box AES Implementation. In: Selected Areas in Cryptography, 11th International Workshop, SAC 2004, Waterloo, Canada, August 9-10, 2004, Revised Selected Papers. pp. 227–240 (2004)

4. Biryukov, A., Bouillaguet, C., Khovratovich, D.: Cryptographic Schemes Based on the ASASA Structure: Black-Box, White-Box, and Public-Key (Extended Abstract). In: 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. pp. 63–84 (2014)
5. Bogdanov, A., Isobe, T.: White-Box Cryptography Revisited: Space-Hard Ciphers. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-6, 2015. pp. 1058–1069 (2015)
6. Bos, J.W., Hubain, C., Michiels, W., Teuwen, P.: Differential Computation Analysis: Hiding Your White-Box Designs is Not Enough. In: Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings. pp. 215–236 (2016), https://doi.org/10.1007/978-3-662-53140-2_11
7. Bringer, J., Chabanne, H., Dottax, E.: White Box Cryptography: Another Attempt. IACR Cryptology ePrint Archive 2006, 468 (2006)
8. Carlet, C., Goubin, L., Prouff, E., Quisquater, M., Rivain, M.: Higher-Order Masking Schemes for S-Boxes. In: Fast Software Encryption - 19th International Workshop, FSE 2012, Washington, DC, USA, March 19-21, 2012. Revised Selected Papers. pp. 366–384 (2012), https://doi.org/10.1007/978-3-642-34047-5_21
9. Chow, S., Eisen, P., Johnson, H., Oorschot, P.C.V.: White-Box Cryptography and an AES Implementation. In: Proceedings of the Ninth Workshop on Selected Areas in Cryptography (SAC 2002). pp. 250–270. Springer-Verlag (2002)
10. Chow, S., Eisen, P.A., Johnson, H., van Oorschot, P.C.: A White-Box DES Implementation for DRM Applications. In: Security and Privacy in Digital Rights Management, ACM CCS-9 Workshop, DRM 2002, Washington, DC, USA, November 18, 2002, Revised Papers. pp. 1–15 (2002)
11. Coron, J., Goubin, L.: On Boolean and Arithmetic Masking against Differential Power Analysis. In: Cryptographic Hardware and Embedded Systems - CHES 2000, Second International Workshop, Worcester, MA, USA, August 17-18, 2000, Proceedings. pp. 231–237 (2000), https://doi.org/10.1007/3-540-44499-8_18
12. Gemalto white-box cryptographic solution: https://sentinel.gemalto.com/software-monetization/white-box-cryptography/
13. Golic, J.D., Tymen, C.: Multiplicative Masking and Power Analysis of AES. In: Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers. pp. 198–212 (2002), https://doi.org/10.1007/3-540-36400-5_16
14. Goubin, L., Masereel, J., Quisquater, M.: Cryptanalysis of White Box DES Implementations. In: Selected Areas in Cryptography, 14th International Workshop, SAC 2007, Ottawa, Canada, August 16-17, 2007, Revised Selected Papers. pp. 278–295 (2007)
15. InsideSecure white-box cryptographic solution: https://www.insidesecure.com/Products/Application-Protection/Software-Protection/WhiteBox
16. Joye, M., Paillier, P., Schoenmakers, B.: On Second-Order Differential Power Analysis. In: Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings. pp. 293–308 (2005), https://doi.org/10.1007/11545262_22
17. Karroumi, M.: Protecting White-Box AES with Dual Ciphers. In: Information Security and Cryptology - ICISC 2010 - 13th International Conference, Seoul, Korea, December 1-3, 2010, Revised Selected Papers. pp. 278–291 (2010)

18. Kim, H., Hong, S., Lim, J.: A Fast and Provably Secure Higher-order Masking of AES S-box. In: Proceedings of the 13th international conference on Cryptographic hardware and embedded systems. pp. 95–107. CHES'11, Springer-Verlag, Berlin, Heidelberg (2011)
19. Kocher, P.C., Jaffe, J., Jun, B.: Differential Power Analysis. In: Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings. pp. 388–397 (1999)
20. Lee, S., Choi, D., Choi, Y.J.: Conditional Re-encoding Method for Cryptanalysis-Resistant White-Box AES. vol. 5. Electronics and Telecommunications Research Institute (Oct 2015), `http://dx.doi.org/10.4218/etrij.15.0114.0025`
21. Lepoint, T., Rivain, M., Mulder, Y.D., Roelse, P., Preneel, B.: Two Attacks on a White-Box AES Implementation. In: Selected Areas in Cryptography - SAC 2013 - 20th International Conference, Burnaby, BC, Canada, August 14-16, 2013, Revised Selected Papers. pp. 265–285 (2013)
22. Link, H.E., Neumann, W.D.: Clarifying Obfuscation: Improving the Security of White-box DES. In: International Conference on Information Technology: Coding and Computing (ITCC'05) - Volume II. vol. 1, pp. 679–684 Vol. 1 (2005)
23. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks: Revealing the Secrets of Smart Cards (Advances in Information Security) (2007)
24. Messerges, T.S.: Securing the AES Finalists Against Power Analysis Attacks. In: Fast Software Encryption, 7th International Workshop, FSE 2000, New York, NY, USA, April 10-12, 2000, Proceedings. pp. 150–164 (2000), `https://doi.org/10.1007/3-540-44706-7\_11`
25. Messerges, T.S.: Using Second-Order Power Analysis to Attack DPA Resistant Software. In: Cryptographic Hardware and Embedded Systems - CHES 2000, Second International Workshop, Worcester, MA, USA, August 17-18, 2000, Proceedings. pp. 238–251 (2000), `https://doi.org/10.1007/3-540-44499-8\_19`
26. Michiels, W., Gorissen, P., Hollmann, H.D.L.: Cryptanalysis of a Generic Class of White-Box Implementations. In: Selected Areas in Cryptography, 15th International Workshop, SAC 2008, Sackville, New Brunswick, Canada, August 14-15, Revised Selected Papers. pp. 414–428 (2008)
27. Minaud, B., Derbez, P., Fouque, P., Karpman, P.: Key-Recovery Attacks on ASASA. In: Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II. pp. 3–27 (2015)
28. Mulder, Y.D., Roelse, P., Preneel, B.: Cryptanalysis of the Xiao - Lai White-Box AES Implementation. In: Selected Areas in Cryptography, 19th International Conference, SAC 2012, Windsor, ON, Canada, August 15-16, 2012, Revised Selected Papers. pp. 34–49 (2012)
29. Mulder, Y.D., Wyseur, B., Preneel, B.: Cryptanalysis of a Perturbated White-Box AES Implementation. In: Progress in Cryptology - INDOCRYPT 2010 - 11th International Conference on Cryptology in India, Hyderabad, India, December 12-15, 2010. Proceedings. pp. 292–310 (2010)
30. Oswald, E., Mangard, S., Pramstaller, N., Rijmen, V.: A Side-channel Analysis Resistant Description of the AES S-box. In: Proceedings of the 12th international conference on Fast Software Encryption. pp. 413–423. FSE'05, Springer-Verlag, Berlin, Heidelberg (2005)
31. Prouff, E., Rivain, M.: Masking against Side-Channel Attacks: A Formal Security Proof. In: Advances in Cryptology - EUROCRYPT 2013, 32nd Annual In-

ternational Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings. pp. 142–159 (2013), https://doi.org/10.1007/978-3-642-38348-9_9

32. Rivain, M., Prouff, E.: Provably Secure Higher-order Masking of AES. In: Proceedings of the 12th International Conference on Cryptographic Hardware and Embedded Systems. pp. 413–427. CHES'10, Springer-Verlag, Berlin, Heidelberg (2010)
33. Sanfelix, E., Mune, C., de Haas, J.: Unboxing the White-Box: Practical Attacks against Obfuscated Ciphers. In: Presented at BlackHat Europe 2015 (2015), https://www.blackhat.com/eu-15/briefings.html
34. Sasdrich, P., Moradi, A., Güneysu, T.: White-Box Cryptography in the Gray Box - - A Hardware Implementation and its Side Channels -. In: Fast Software Encryption - 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers. pp. 185–203 (2016)
35. Schramm, K., Paar, C.: Higher Order Masking of the AES. In: Proceedings of the 2006 The Cryptographers' Track at the RSA conference on Topics in Cryptology. pp. 208–225. CT-RSA'06, Springer-Verlag, Berlin, Heidelberg (2006)
36. Trichina, E., Seta, D.D., Germani, L.: Simplified Adaptive Multiplicative Masking for AES. In: Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers. pp. 187–197 (2002), https://doi.org/10.1007/3-540-36400-5_15
37. Waddle, J., Wagner, D.A.: Towards Efficient Second-Order Power Analysis. In: Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings. pp. 1–15 (2004), https://doi.org/10.1007/978-3-540-28632-5_1
38. WhiteboxCRYPTO: https://www.microsemi.com/document-portal/doc_view/135631-whiteboxcrypto-product-overview-rev4
39. Wyseur, B.: White-Box Cryptography. In: Encyclopedia of Cryptography and Security, 2nd Ed. pp. 1386–1387 (2011), http://dx.doi.org/10.1007/978-1-4419-5906-5_627
40. Wyseur, B., Michiels, W., Gorissen, P., Preneel, B.: Cryptanalysis of White-Box DES Implementations with Arbitrary External Encodings. In: Selected Areas in Cryptography, 14th International Workshop, SAC 2007, Ottawa, Canada, August 16-17, 2007, Revised Selected Papers. pp. 264–277 (2007)
41. Xiao, J., Zhou, Y.: Generating Large Non-Singular Matrices over an Arbitrary Field with Blocks of Full Rank (2002), http://eprint.iacr.org/2002/096
42. Xiao, Y., Lai, X.: A Secure Implementation of White-box AES. In: The Second Internationial Conference on Computer Science and Its Applications - CSA 2009. vol. 2009, pp. 1–6 (2009)