On the Linear Transformation in White-box Cryptography

Seungkwang Lee, Nam-su Jho and Myungchul Kim

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

Abstract. Linear transformations are applied to the white-box cryptographic implementation for the diffusion effect to prevent key-dependent intermediate values from being analyzed. However, it has been shown that there still exists a correlation before and after the linear transformation, and thus this is not enough to protect the key against statistical analysis. So far, the Hamming weight of rows in the invertible matrix has been considered the main cause of the key leakage from the linear transformation. In this study, we present an in-depth analysis of the distribution of intermediate values and the characteristics of block invertible binary matrices. Our mathematical analysis and experimental results show that the balanced distribution of the key-dependent intermediate value is the main cause of the key leakage.

Keywords: White-box cryptography, linear transformation, key leakage.

1 Introduction

From a secret key point of view, a block cipher can be seen as a secret bijection between a plaintext set and a ciphertext set. One of the easy ways to implement this bijection is a lookup table mapping a plaintext to its corresponding ciphertext. Since implementing a block cipher as one lookup table is impractical because of its huge size, it is usually implemented as a series of lookup tables. White-box cryptography adapts this table-based implementation on a cryptographic algorithm and applies linear and nonlinear transformations to obfuscate inputs and outputs of each table thereby protecting the key.

There are various techniques to extract the key hidden in white-box cryptographic implementations of standard block ciphers such as DES and AES. Above all, a number of practical cryptanalysis techniques [3, 9, 13, 16–18, 23] on the white-box DES (WB-DES), AES (WB-AES) and their variants [6, 11, 12, 14, 25] have been introduced. In addition, Differential Fault Analysis (DFA) [20] on white-box cryptography has been demonstrated, where an attacker is able to inject faults at a precise location in memory. Note that these white-box attacks rely on an in-depth understanding of a target implementation so that an attacker is able to access precise internal states during the execution. Thus commercial white-box cryptography [2, 8, 10, 22] focuses on making a barrier to the full

control of an attacker by adapting additional techniques such as obfuscation, enveloping, hardware ID binding, and anti-debug protections.

In contrast, Differential Computation Analysis (DCA) [4] adapts a statistical technique of Correlation Power Analysis (CPA) [5] but uses computation traces (a.k.a software execution traces) consisting of internal information such as noise-free intermediate values and memory accesses, instead of classical power traces. Our work is motivated by the fact that white-box cryptography can be easily broken by simple statistical analysis [4, 21] without having to perform cryptanalysis. Along this paper, we use the term *key leakage* to mean that the key is recovered by some technique of the attack. Then we can say that the linear and nonlinear transformations of white-box cryptography cannot prevent the key leakage from statistical analysis. In case of linear transformations, it was recently analyzed that if the invertible matrix used for the linear transformation has rows of HW 1, then the key leakage will happen with overwhelming probability; otherwise, the correct key is supposed to be indistinguishable from the wrong key hypothesis [1, 19].

Importantly, it was recommended in white-box cryptography to choose a block invertible binary matrix consisting of full-rank submatrices for carrying maximum information and maximizing information diffusion [7]. If the previous analysis is true, the linear transformation using block invertible binary matrices is supposed to prevent the key leakage because there is no such matrix containing a row of HW 1 by the definition of a block invertible matrix. In addition, a white-box cryptographic implementation made up of tables generated using block invertible matrices should not cause a key leakage.

In this paper, we demonstrate that the linear transformation still causes the key leakage even in the case of block invertible matrices. Additionally, we find out that the key leakage is largely due to the balanced distribution of intermediate values, and we provide a simple proof and demonstrations using the Walsh transform. To enhance our analysis, we insert a random byte in the intermediate value before the linear transformation making an unbalanced distribution and show a reduced correlation to the key. The rest of this paper is organized as follows. Section 2 reviews DCA and the Walsh transform used to compute a correlation, and revisits the key leakage issue in white-box cryptography. In Section 3, we provide our analysis of the main reason behind the key leakage in the presence of linear transformations. Section 4 concludes this paper.

2 Background

In this section, we explain the basic concepts of DCA and the Walsh transform used to test the key leakage. Additionally, we demonstrate the key leakage using the Walsh transform in the presence of linear and nonlinear transformations.

2.1 DCA

An explanation of successful DCA on white-box cryptography could be that the correct hypothetical value is correlated to the target lookup value. After col-

lecting the computation traces with random plaintexts, a DCA attacker adapts CPA as a subroutine for calculating Pearson's correlation coefficient between each sample in the computation trace and the hypothetical value.

Given N power traces $V_{1..N}[1..\kappa]$ containing κ samples each, CPA estimates the value 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 estimate in the n-th trace with the hypothetical key k^*. To measure a correlation between computation traces and hypothetical values, the estimator r at the sample point j is defined as follows [15]:

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

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. Provided that one can know the intermediate value directly, the computation traces are not necessary. Instead, we can apply the Walsh transform consisting of simple operations to the intermediate value and the hypothetical value.

2.2 Walsh Transform

Given a white-box implementation protected by linear and nonlinear transformations (often we use the term *encoding*), we can detect the existence of a problematic correlation for the key leakage using the Walsh transform. In order to explain how the Walsh transform can be used to compute a correlation we use the following definitions from [21].

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 $0 \leq HW(\omega) \leq d$, it is called a balanced d-th order correlation immune function or an d-resilient function.*

By Definition 1 and 2, $W_f(\omega)$ quantifies the imbalances in the encoding, and the large absolute value of $W_f(\omega)$ means the strong correlation between $f(x)$ and $x \cdot \omega$. By utilizing this property, we calculate the correlation between the table lookup values $f(x)$ and hypothetical values $x \cdot \omega$, where ω selects a particular bit of x. Here it is important to notice that $f(x)$ plays the similar role as the computation trace of DCA because it represents the real intermediate value. For

this reason, when the lookup value can be exactly accessed, the Walsh transform can be used to detect the key leakage like CPA in the DCA attack as follows.

Here is a demonstration of the key leakage in Chow's WB-AES implementation with a 128-bit key [7]. To do so, we assume that the attacker's hypothetical value is the SubBytes output in the first round and the target lookup table is generated by the composition of SubBytes, AddRoundKey and MixColumns. We denote the initial round key and the plaintext by K $(= k_0k_1 \dots k_{15})$ and P $(= p_0p_1 \dots p_{15})$, respectively. Then we let $x_i = S(p_i \oplus k_i)$, where S means the S-box of AES. To decompose the MixColumns operation with a column vector $[x_0\ x_1\ x_2\ x_3]^T$, let us denote MC_i the i-th column vector of the MixColumns matrix. Then we have:

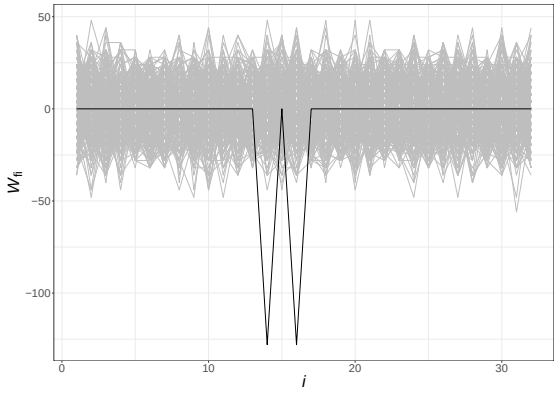
$$\begin{aligned}
&\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \\
&\quad = x_0 \begin{bmatrix} 02 \\ 01 \\ 01 \\ 03 \end{bmatrix} \oplus x_1 \begin{bmatrix} 03 \\ 02 \\ 01 \\ 01 \end{bmatrix} \oplus x_2 \begin{bmatrix} 01 \\ 03 \\ 02 \\ 01 \end{bmatrix} \oplus x_3 \begin{bmatrix} 01 \\ 01 \\ 03 \\ 02 \end{bmatrix} \\
&\quad = x_0 \cdot MC_0 \oplus x_1 \cdot MC_1 \oplus x_2 \cdot MC_2 \oplus x_3 \cdot MC_3,
\end{aligned}$$

Now let $y_i(x_i) = x_i \cdot MC_i$ on the right-hand side. For simplicity, we abuse the notation by skipping the subscript 0 to x and y by letting $y(x) = y_0(x_0)$. If we let $f(x) = \epsilon(y(x))$, where ϵ implies the linear and nonlinear transformations, $f(x)$ is then the encoded lookup values for the MixColumns multiplication when the input is the first subbyte of the plaintext. For 32 Boolean functions $f_{i\in\{1,\dots,32\}}(x)$: $\{0,1\}^8 \to \{0,1\}$, we calculate the Walsh transforms W_{f_i} and sum all the imbalances for each key candidate and ω such that $\mathrm{HW}(\omega) = 1$ in order to recover the target subkey $k_0 = 0x88$ as follows:

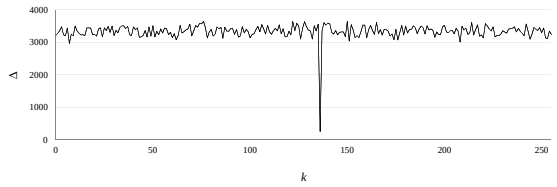
$$\Delta^f_{k\in\{0,1\}^8} = \sum_{\omega=1,2,4,\dots,128} \sum_{i=1,\dots,32} |W_{f_i}(\omega)|.$$

Here, we only select ω of $\mathrm{HW}(\omega) = 1$ because the HW-based key leakage model is not effective to detect the correlation before and after the both transformations.

The Walsh transforms and their sum of all imbalances are plotted in Fig. 1. As shown in Fig. 1a, the Walsh transforms with $\omega = 4$ of the correct subkey ($0x88$) produce 0 except two points; the W_{f14} and W_{f16} of the correct subkey 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 the key leakages and thus the statistical analysis using the third bit (the LSB is the first bit) can be successful.



(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. $\Delta^f_{k=0x88} = 256$ ($|-128| + |-128|$).

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

3 Analysis of Linear Transformations

As mentioned, the previous work [1, 19] on linear transformations analyzed that rows of HW 1 in the invertible matrix cause the key leakage. In addition, it was reportedly possible to recover the key in the presence of a matrix without identity row by calculating all of the 2^8 linear combinations of the bits in the target intermediate value [1]. Before going on, we note that a 32×32 linear transformation is applied to the SubBytes output multiplied with MC_i in the typical WB-AES implementation [7], instead of applying an 8×8 linear transformation to the SubBytes output (an 8×8 linear transformation is usually applied to the round output). In this case, it becomes very complex, unlike their analysis, to carry out an attack on all possible combinations. In the following, we present our mathematical analysis and experimental results showing that the main cause of the key leakage is related to the distribution of the intermediate values rather than some characteristic of the matrix.

3.1 Analysis of Key-dependent Intermediate Values

The following proof explains why the distribution of key-dependent intermediate values leads to $W_{f_i}(\omega) = 256$ after the linear transformation. By Definition 1, this is the maximum value of $W_{f_i}(\omega)$ for $x \in GF(2^8)$ and thus certainly causes the key leakage.

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 the 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.

$$\begin{aligned} \oplus_{j \in \mathcal{J}} \mathbf{h}_{*,j} &= \left(\oplus_{j \in \mathcal{J} - \{\alpha, \alpha'\}} \mathbf{h}_{*,j} \right) \oplus \mathbf{h}_{*,\alpha} \oplus \mathbf{h}_{*,\alpha'} \\ &= \left(\oplus_{j \in \mathcal{J} - \{\alpha, \alpha'\}} \mathbf{h}_{*,j} \right) \oplus \mathbf{0} = \oplus_{j \in \mathcal{J} - \{\alpha, \alpha'\}} \mathbf{h}_{*,j}. \end{aligned}$$

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

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 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 $\mathcal{I}_{b_1, b_2, \ldots, b_n}$, 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 $\left|\mathcal{I}_{b_1, b_2, \ldots, b_n}\right|$ where $\oplus_{i=1,\ldots,n} b_i = 1$.

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

$$= \Sigma_{\oplus_{i=1,\ldots,n} b_i=1} 2^{8-n} = \Sigma_{2^{n-1}} 2^{8-n}$$
$$= 2^{n-1} \cdot 2^{8-n} = 2^7 = 128.$$

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

Note that $W_{f_i}(\omega)$ is defined as $\sum_{x \in GF(2^8)} (-1)^{f_i(x) \oplus x \cdot \omega} = \sum_{x \in GF(2^8)} (-1)^{M_{i,*} \cdot y(x) \oplus x \cdot \omega}$, where $M_{i,*}$ is the i-th row of the matrix M. 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 in 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:

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

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 $\{1, 2, \ldots, 256\}$ can be re-written with the number of exponents which are 1.

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

All row vectors of the matrix $\mathbf{Y}(x)$ can be 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. Whenever this happens, there will definitely be the key leakage because $f_i(x)$ is most correlated to $x \cdot \omega$.

To demonstrate the experimental results for the lemma above for all $y_{i \in \{0,1,2,3\}}$, we let

$$f^i(x) = M \cdot y_i(x_i)_{i \in \{0,1,2,3\}},$$

where M is a (32, 2) block invertible square binary matrix defined in [24] as follows.

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.*

Importantly, it is recommended by the author of [7] to choose a non-singular matrix consisting of full-rank submatrices for the following reasons. First, this ensures that the encoded components will carry maximum information and maximizing information diffusion. Second, a large block invertible matrix can be efficiently generated by using the technique explained in [24]. Here we note that block invertible matrices have no row of HW 1. Thus, the frequent key leakage from the linear transformations using block invertible matrices is contradict to the previous analysis of the leakage cause [1, 19].

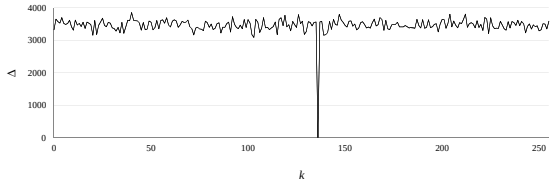
For f_j^i, $0 \leq i \leq 3$ and $1 \leq j \leq 32$, the statistical analysis with the SubBytes output in the first round can be performed by the Walsh transforms. By computing 1,024 ($= 4 \times 32 \times 8$) Walsh transforms with $f^i(\cdot)$ one can analyze the key leakage with respect to the four subkeys involved in the linear transformations by M. In this experiment, the four subkeys are $k_0 = 0x88$, $k_1 = 0x99$, $k_2 = 0x\text{AA}$ and $k_3 = 0x\text{BB}$.

The crucial observation of the Walsh transforms plotted in Fig. 2 is that there still exists a problematic probability of key leakage from linear transformations using the block invertible matrix. Unlike the case of the key leakage from the linear and nonlinear transformations shown in Fig. 1, this result shows the key leakage from only linear transformations without nonlinear transformations. Hereafter, we abuse the notation by skipping the superscript of f^i by letting $f = f^i$. We can see that linear transformations with M hide three subkeys k_0, k_2 and k_3 with the Walsh transforms score 0, but expose $k_1(0x99)$ from $y_1(x)$ with the Walsh transforms score 256. According to Lemma 1, 256 is the only nonzero value that the Walsh transform can take from the correct subkey and the maximum value that the Walsh transform can take. This gives us that linear transformations produce well-balanced outputs with an overwhelming probability, but this is not always guarantee a reliable protection on the key.

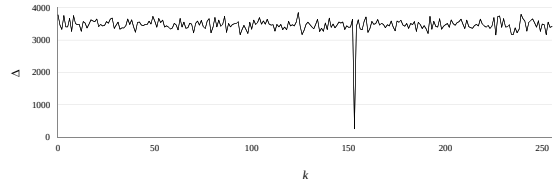
We have repeated the above experiment using 1,000 randomly generated (32, 2) matrices. For $\text{HW}(\omega) = 1$, the correct subkey gives us that $\Pr[W_{f_i}(\omega) = 0] \approx 0.997$ and $\Pr[W_{f_i}(\omega) = 256] \approx 0.003$; the average of $|W_{f_i}(\omega)|$ is approximately 0.7 as shown in Table 1. Although the probability of $W_{f_i}(\omega) = 256$ is small, 1,024 Walsh transforms probably produce three peaks of the correct subkey distinguishable from wrong key candidates, and the three peaks can reveal at most three subkeys. Fig. 3 depicts our experimental result that 1 to 3 out of four subkeys are exposed in most cases. Only 51 of 1,000 matrices did not leak any subkey. Consequently, this demonstrates Lemma 1 and explains why the linear transformation cannot guarantee the protection of key in white-box cryptography.

3.2 Analysis of Block Invertible Square Matrix

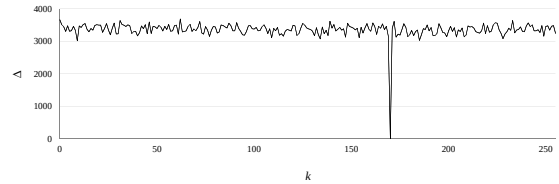
To enhance our analysis on the key leakage, we have performed additional experiments to check if the HW of (32, 2) matrices causes the key leakage. Generating (n, 2) block invertible square matrices begins with a (2, 2) block invertible square matrix and extends by (4, 2), (6, 2), ..., and repeats it $(n-2)/2$ times [24]. Note that every 2×2 submatrix in a (n, 2) block invertible square matrix should be



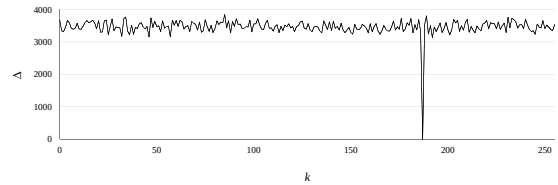
(a) On $M \cdot y_0(x)$



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



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



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

Fig. 2: Sum of the imbalance of $W_{f_i}(\omega)$ for all subkey candidates on linearly transformed $y_{i\in\{0,1,2,3\}}(x)$.

invertible by the definition and all 2×2 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 $n/2$.

		To be linearly transformed			
		y_0	y_1	y_2	y_3
Number of	$W_{f_i}(\omega) = 0$	255,206	255,205	255,309	255,203
	$W_{f_i}(\omega) = 256$	794	795	691	797

Table 1: Statistic of W_{f_i} scores calculated with 1,000 randomly generated (32, 2) matrices.

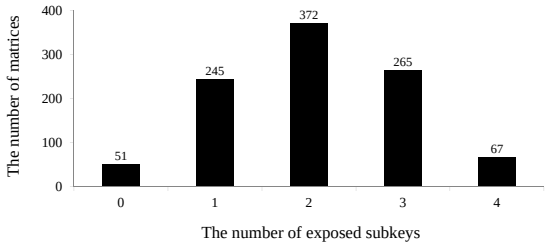


Fig. 3: The number of block invertible matrices (y-axis) vs. the number of exposed subkeys from $M \cdot y_{i\in\{0,1,2,3\}}$ for each block invertible matrix M (x-axis).

For example, consider a (4, 2) matrix initialized with

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

then its resulting matrix will be

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

If it is initialized with

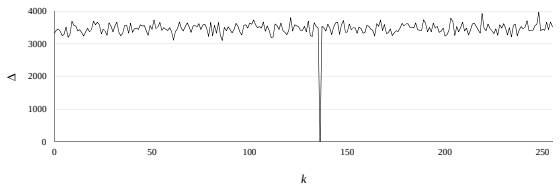
$$\begin{vmatrix} 0 & 1 \\ 1 & 0 \end{vmatrix},$$

we have

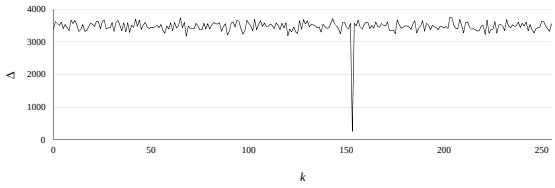
$$\begin{vmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{vmatrix}.$$

When generating a (32, 2) matrix through this process, 1s appear more frequently. Based on this fact, we have performed the following experiment to test whether or not this overweight HW of the block invertible matrix is one of the causes of the key leakage. We have randomly generated a balanced *non-invertible* (singular) 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

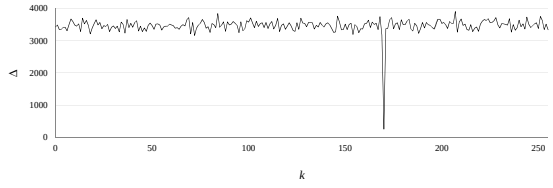
imbalances. As shown in Fig. 4, there still exist key leakages from y_1 and y_2 with the Walsh transform score 256. This shows us that the heavy HW of matrices is not the cause of the key leakage from linear transformations.



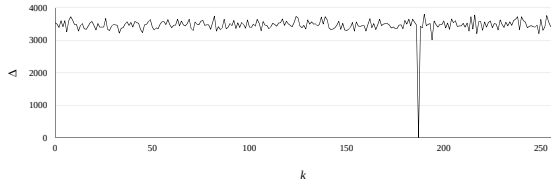
(a) On $M^b \cdot y_0(x)$



(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. 4: 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.

3.3 Effect of Unbalanced Intermediate Values

So far, we have analyzed the balanced distribution of the key-dependent intermediate values as the main cause of the key leakage. In the connection with this,

we demonstrate the effect of unbalanced distribution of intermediate values by inserting random bytes in the intermediate values before linear transformations.

Let's begin with an analysis of the inserting position. We have inserted a random byte at a particular position in the four-byte intermediate value $y_{i\in\{0,1,2,3\}}(x)$ and then performed a linear transformation with a (40, 2) 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 have selected ρ_i, where $i \in [1, 5]$, and then inserted different $\gamma \in_R \mathrm{GF}(2^8)$ at ρ_i for each $x \in \mathrm{GF}(2^8)$. Let $y_0^*(x)$ denote $y_0(x)$ after the random byte insertion, and $f^*(x) = M^* \cdot y_0^*(x)$. 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 1,000 randomly generated M^*, we have computed $W_{f_i^*}(\omega)$. As a result, Table 2 gives us that the correct subkey has $\Pr[W_{f_i^*}(\omega) = 0] \approx 0.05$ (The max and average $|W_{f_i^*}(\omega)|$ are about 72 and 12.7, respectively). This is in contrast to $\Pr[W_{f_i}(\omega) = 0] \approx 0.997$ and $|W_{f_i}(\omega)| \approx 0.7$ without the random byte insertion. This implies that the encoding imbalance increases in the linear transformation with an unbalanced intermediate value by inserting a random byte.

To compare an effect of a random byte, we have conducted an additional experiment with random vectors 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, these are five-byte random vectors.
2. $f^\gamma(x) = M^* \cdot y^\gamma(x)$.
3. Repeat step (1) - (2) with 1,000 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 ρ_i.

As a result, we have $\Pr[W_{f_i^\gamma}(\omega) = 0] \approx 0.05$ (The max and average $|W_{f_i^\gamma}(\omega)|$ are about 76 and 12.74, respectively) and the cosine similarity between their distributions is always larger than 0.999. Note that a cosine similarity greater than 0.99 indicates a very similar distribution, and the cosine similarity between the distributions of $W_{f_i^\gamma}(\omega)$ and $W_{f_i}(\omega)$ was just 0.25, approximately.

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

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

	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 $\lvert W_{f_i^*}(\omega)\rvert$	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 $W_{f_i^\gamma}$.

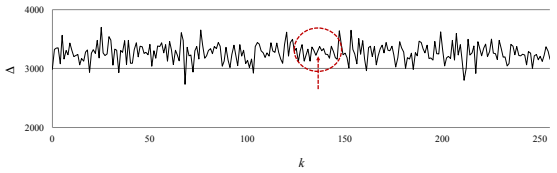
Fig. 5 shows $\Delta^{f^*}_{k\in\{0,1\}^8}$ that the correct subkeys *0x*88 - *0x*BB are no longer distinguishable from other candidates. In addition, it is noticeable that inserting more than one random byte in the intermediate values does not increase the imbalance; they show a similar level of the imbalance with the one-byte insertion.

4 Conclusion

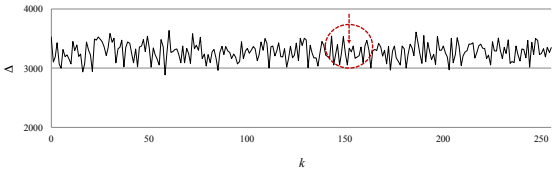
Previous analysis has shown that rows of HW 1 in the invertible matrix are the main cause of the key leakage from the linear transformation. Also, it has been suggested to recover the key in the presence of such a matrix without identity row by calculating all possible linear combinations of the bits in the target intermediate value. In this paper, we pointed out that there is no such row of HW 1 if we choose a block invertible matrix with submatrices of full rank for maximizing information diffusion. Nevertheless, the key leakage is likely to happen from the linear transformation regardless of the HW of block invertible matrices. In addition, we pointed out that a typical WB-AES implementation uses a 32×32 linear transformation on the SubBytes output multiplied with the decomposed MixColumns rather than an 8×8 linear transformation on the SubBytes output. Thus, it is complicated for an attacker to analyze all possible linear combinations. Our analysis and experimental results explained that the balanced distribution of intermediate values causes the key leakage. In the connection with this, it was demonstrated that the unbalanced distribution of the intermediate values can be effective to reduce the probability of key leakage.

Acknowledgment

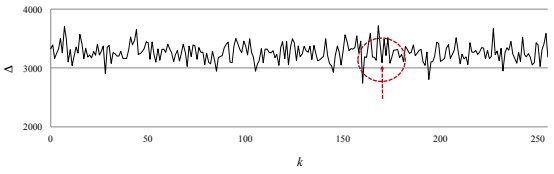
This work was supported by Institute of Information & communications Technology Planning & Evalution (IITP) grant funded by the Korea government (MSIT) (No.2018-0-00230, Development on Autonomous Trust Enhancement Technology of IoT Device and Study on Adaptive IoT Security Open Architecture based on Global Standardization [TrusThingz Project]).



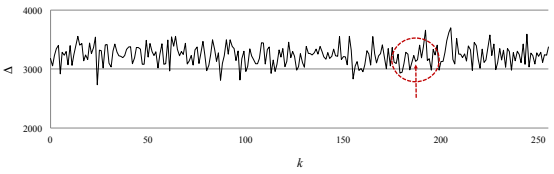
(a) On $M^* \cdot y_0^*(x)$



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



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



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

Fig. 5: Sum of the imbalance of $W_{f_i^*}(\omega)$ for all key candidates. Red arrow: the correct key.

References

1. Alpirez Bock, E., Brzuska, C., Michiels, W., Treff, A.: On the Ineffectiveness of Internal Encodings - Revisiting the DCA attack on White-box Cryptography. In: Applied Cryptography and Network Security - 16th International Conference, ACNS 2018, Proceedings. pp. 103–120. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Springer, Germany (1 2018)
2. Axsan white-box cryptographic solution.: (accessed Oct 7, 2019), `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. 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
5. Brier, E., Clavier, C., Olivier, F.: Correlation Power Analysis with a Leakage Model. In: Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings. Lecture Notes in Computer Science, vol. 3156, pp. 16–29. Springer (2004)
6. Bringer, J., Chabanne, H., Dottax, E.: White Box Cryptography: Another Attempt. IACR Cryptology ePrint Archive 2006, 468 (2006)
7. 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)
8. Gemalto white-box cryptographic solution: (accessed Oct 7, 2019), https://sentinel.gemalto.com/software-monetization/white-box-cryptography/
9. 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)
10. InsideSecure white-box cryptographic solution: (accessed Oct 7, 2019), https://www.insidesecure.com/Products/Application-Protection/Software-Protection/WhiteBox
11. 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)
12. 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
13. 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)
14. 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 (April 2005)
15. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks: Revealing the Secrets of Smart Cards (Advances in Information Security) (2007)
16. 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)
17. 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)
18. 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)
19. Rivain, M., Wang, J.: Analysis and Improvement of Differential Computation Attacks against Internally-Encoded White-Box Implementations. IACR Transactions on Cryptographic Hardware and Embedded Systems 2019(2), 225–255 (Feb 2019), https://tches.iacr.org/index.php/TCHES/article/view/7391
20. 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
21. 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)
22. WhiteboxCRYPTO: (accessed Oct 7, 2019), https://www.microsemi.com/document-portal/doc_view/135631-whiteboxcrypto-product-overview-rev4
23. 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)
24. 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
25. 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)