On the Key Leakage from Linear Transformations

Seungkwang Lee, Nam-su Jho and Myungchul Kim

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

Abstract. Linear transformations are often applied to the table-based cryptographic implementation including white-box cryptography in order 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 transformations, and thus this is not enough to protect the key against gray-box attacks such as power 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, power analysis, 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 like in the case of white-box cryptography. The important point here is that white-box cryptography generates key-instantiated lookup tables and protects each table with linear and nonlinear transformations in order to prevent a key leakage from lookup values.

There are various techniques to extract the key hidden in white-box cryptographic implementations of standard block ciphers such as DES and AES. First, a number of practical cryptanalysis techniques [4, 13, 21, 26–28, 39] on the white-box DES (WB-DES) and AES (WB-AES) and their variants [7, 17, 20, 22, 41] have been introduced. Second, Differential Fault Analysis (DFA) [33] on white-box cryptography was also demonstrated, 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 access precise internal states during the execution. Thus commercial white-box cryptography [3, 11, 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.

In contrast to above white-box attacks, gray-box attacks using non-invasive information such as power consumption of a target device can be mounted. Differential Power Analysis (DPA) [19], one of the most well-known techniques to reveal the secret key imbedded in IC cards, is based on the fact that power consumption of a device is proportional or inversely proportional to the Hamming weight (HW) of data it processes. In detail, 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. Here, we focus on the fact that white-box cryptography can be easily broken by power analysis [5, 34] without having to perform cryptanalysis. This means that linear and nonlinear transformations applied to lookup tables have no effect on hiding key-sensitive intermediate values. In case of linear transformations, it was recently reported in [2, 32] that if the invertible matrix used for the linear transformation has rows of HW 1, then power analysis will succeed with overwhelming probability. Otherwise, the correct key is expected to be indistinguishable from the wrong key hypothesis correlation and power analysis fails with high probability.

Here, we note that it is recommended in white-box cryptography to choose a block invertible binary matrix with submatrices of full rank for carrying maximum information and maximizing information diffusion [9]. According to the conclusion in [2, 32], linear transformations using block invertible binary matrices are unlikely to leak the key because there will be no such matrix containing any row of HW 1 by the definition of a block invertible matrix. In this paper, we demonstrate that the key leakage from the linear transformation still takes place even in the case of block invertible matrices. Importantly, we find out that the key leakage after linear transformations is largely due to the balanced distribution of intermediate values, and we offer a simple proof and demonstrations using the Walsh transforms. To enhance our finding, we insert a random byte in the intermediate value before linear transformations making an unbalanced distribution and show a reduced correlation to the key.

The rest of this paper is organized as follows. Section 2 reviews some basic concepts including power analysis and revisits the key leakage issue in white-box cryptography with the Walsh transforms. In Section 3, we provide our analysis of the main reason why the key is still revealed in the presence of linear transformations. Section 4 concludes this paper.
2 Background

In this section, we introduce the basic concept of power analysis and we use the Walsh transforms for demonstrating the key leakage in the presence of linear and nonlinear transformations.

2.1 Power Analysis

An explanation of successful power analysis on the white-box cryptographic implementation could be that attacker’s correct hypothetical value will correlate to the target lookup value. Here, DPA or Correlation Power Analysis (CPA) [6] can be used as power analysis techniques. Note that Differential Computation Analysis [5] improves the efficiency of DPA and CPA attacks 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 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 nonlinear transformations. Given \( N \) power traces \( \{V_{1..N}\} \) containing \( \kappa \) 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 \( r \) at the sample point \( j \) 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, 10, 12, 24, 29, 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 [8, 18, 30, 31, 35]. One of the most used hiding techniques, on the other hand, is to induce 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 [14]. 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

Given a table-based implementation of a block cipher which is protected by linear and nonlinear transformations (often we use the term \textit{encoding}), we can detect the key leakage using the Walsh transforms. To understand how the Walsh transform can be used to quantify a correlation we use the following definitions from [34].

\textbf{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 \( x \cdot \omega = x_1 \omega_1 \oplus \cdots \oplus x_n \omega_n \). 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

\[ W_f(\omega) = \sum_{x \in \{0, 1\}^n} (-1)^{f(x) \oplus x \cdot \omega}. \]

\textbf{Definition 2.} If 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 \( \omega \) plays a role in bit selection for \( x \) in mono-bit power analysis.

Let’s demonstrate the key leakage in the WB-AES implementation with a 128-bit key [9]. To do so, we assume that an 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 by \( k = k_0 k_1 \ldots k_{15} \), and decompose the Mixcolumns operation
with a column vector \( [x_0 \ x_1 \ x_2 \ x_3]^T \) of the state matrix as follows:

\[
\begin{bmatrix}
  02 & 03 & 01 & 01 \\
  01 & 02 & 03 & 01 \\
  01 & 01 & 02 & 03 \\
  03 & 01 & 01 & 02
\end{bmatrix}
\times
\begin{bmatrix}
  x_0 \\
  x_1 \\
  x_2 \\
  x_3
\end{bmatrix}
= x_0 \times MC_0 \oplus x_1 \times MC_1 \oplus x_2 \times MC_2 \oplus x_3 \times MC_3,
\]

where \( MC_i \) is the \( i \)th column vector of the MixColumns matrix, and \( y_i(x_i) = x_i \times MC_i \). Now we let

\[
x = S(p \oplus k_0)
y_0(x) = [2 \times x \times x \times 3 \times x]^T
\]

where \( p \in GF(2^8) \) means the first subbyte of the plaintext, and \( S \) represents SubBytes. Let \( f(x) \) denote the lookup values of \( y(x) \) encoded by linear and non-linear transformations. We also denote 32 Boolean functions by \( f_i \in \{1,...,32\}(x): \{0,1\}^8 \rightarrow \{0,1\} \). To recover the target subkey \( k_0 = 0\times88 \), we calculate the Walsh transforms \( W_{f_i} \) and sum all the imbalances for each key candidate and \( \omega \) such that \( HW(\omega) = 1 \) as follows:

\[
\Delta_{k \in \{0,1\}^8} = \sum_{\omega = 1,2,4,...,128} \sum_{i = 1,...,32} |W_{f_i}(\omega)|.
\]

The reason why we only select \( \omega \) of \( HW(\omega) = 1 \) is that 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 we can see in Fig. 1a, the Walsh transforms with \( \omega = 4 \) of the correct key \((0\times88)\) produce 0 except two points; the \( W_{f_{14}} \) and \( W_{f_{16}} \) 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 \( f_{14}(-) \) and \( f_{16}(-) \) cause key leakages and thus power analysis using the 3rd bit (the LSB is the 1st bit) of attacker’s hypothetical SubBytes output can be successful. Hereafter, we will utilize the Walsh transforms for various purposes including the calculation of correlation and our proof regarding the cause of key leakage.

### 3 Analysis of Linear Transformations

As mentioned, previous studies [2, 32] on linear transformations point out that rows of HW 1 in the invertible matrix cause the key leakage. In addition, it is
(a) Walsh transforms for $f_{i \in \{1, \ldots, 32\}}(\cdot)$ with $\omega = 4$ for all key candidates. Gray: wrong key candidates; Black: correct key.

(b) Sum of all imbalances for all key candidates. $\Delta_k^{f_i} = 256 (|−128| + |−128| )$.

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

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 [2]. However, we note that a $32 \times 32$ linear transformation is applied to the SubBytes output multiplied with $MC_i$ in the typical WB-AES implementation [9], instead of applying an $8 \times 8$ linear transformation to the SubBytes output (an $8 \times 8$ 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 lies in 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,
Lemma 1. Assume that a $256 \times 8$ binary matrix $H$ is defined as

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

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

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

$$\bigoplus_{j \in J} h_{*,j} = \left( \bigoplus_{j \in J - \{\alpha, \alpha'\}} h_{*,j} \right) \oplus h_{*,\alpha} \oplus h_{*,\alpha'} = \left( \bigoplus_{j \in J - \{\alpha, \alpha'\}} h_{*,j} \right) \oplus 0 = \bigoplus_{j \in J - \{\alpha, \alpha'\}} h_{*,j}.$$

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

Now we can define following partitions of indices:

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

where $\mathcal{I} = \{1, 2, \ldots, 256\}$, and $b_i \in \{0, 1\}$. Here all $I_{b_1, b_2, \ldots, b_n}$ are disjoint to the others and $\bigcup_{b_1, b_2, \ldots, b_n} I_{b_1, b_2, \ldots, b_n} = \mathcal{I}$. To complete the proof, we need that for any choice of $b_i$'s, $|I_{b_1, b_2, \ldots, b_n}| = 256/2^n = 2^{8-n}$. This can be shown easily as followings. Suppose that $|I_{b_1, b_2, \ldots, b_n}| = t > 2^{8-n}$. It means that there are $t$ row vectors in $H$ satisfying the condition $j_1^{th}$ bit of the vector equals to $b_1$. 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 $\ell$ and $\ell'$ in $I_{b_1, b_2, \ldots, b_n}$, where all bits of $h_{\ell,*}$ are completely same to the bits of $h_{\ell',*}$. It contradicts to the assumption $h_{i,*} \neq h_{j,*}$ for any $i \neq j$.

From the definition of HW, we can deduce $HW(\bigoplus_{j \in J} h_{*,j})$ is summation of $|I_{b_1, b_2, \ldots, b_n}|$ where $\oplus_{b_1=1, \ldots, n} b_i = 1$.

$$HW(\bigoplus_{j \in J} h_{*,j}) = \sum_{\oplus_{b_1=1, \ldots, n} b_i = 1} |I_{b_1, b_2, \ldots, b_n}|$$

$$= \sum_{\oplus_{b_1=1, \ldots, n} b_i = 1} 2^{8-n} = \sum_{2^n-1} 2^{8-n}$$

$$= 2^{n-1} \cdot 2^{8-n} = 2^7 = 128.$$
Note that if $J$ 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 $\sum_{x \in GF(2^8)} (-1)^{f_i(x)} \cdot x \cdot \omega = \sum_{x \in \{0,1\}} (-1)^{M_{i,*} \cdot y(x)} \cdot x \cdot \omega$, where $M_{i,*}$ is the $i^{th}$ 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 $32 \times 1$ matrix $[2 \cdot x \ x \ x \ 3 \cdot x]^T$. If we define $Y(x)$ as a $32 \times 256$ matrix $[2 \cdot H \ H \ 3 \cdot H]^T$, where the $H$ is the matrix defined in the Lemma 1, it is easy to show that each column vector of $2 \cdot H$ or $3 \cdot H$ can be defined with XORs of some column vectors of $H$ based on the property of $GF(2^8)$. Then the above equation can be re-written as

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

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.

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

Note that all row vectors of the matrix $Y(x)$ is represented by XORing of column vectors of $H$. Therefore $M_{i,*} \cdot Y(x) \oplus (w \cdot H^T)$ can be also represented by XORing of column vectors of $H$. From the Lemma 1, it deduces that the HW of $M_{i,*} \cdot Y(x) \oplus (w \cdot H^T)$ is 0 or 128. Finally, $W_{f_i}(w) = 256 - (2 \times HW(M_{i,*} \cdot Y(x) \oplus (w \cdot H^T)))$ 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 $H$ 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 with $x \cdot \omega$.

To demonstrate the experimental results for the lemma above, let denote

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

where $p \in GF(2^8)$, $M$ is a $(32, 2)$ block invertible square binary matrix which is defined in [40] as follows.

**Definition 3.** If all the blocks $B_{i,j}$ in a block matrix $^n_m M^{[pB]}$ 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 $^n_m M^{[pB]}$ 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 [9] to choose a non-singular matrix with submatrices of full rank 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 [40]. 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 will be counter examples of [2, 32].

Note that \( f^i \) linearly transforms the SubBytes output \( x \) multiplied with \( MC_i \), where \( x \) is connected to \( k_i \). For \( f^i \) given to an attacker, \( 0 \leq i \leq 3 \) and \( 1 \leq j \leq 32 \), mono-bit power analysis based on the SubBytes output in the first round can be simulated by the Walsh transforms. By computing \( 1,024 (= 4 \times 32 \times 8) \) Walsh transforms with \( f^i(\cdot) \) we can observe how the key leakage appears with respect to the four subkeys in the linear transformations by \( M \). In this experiment, we used \( k_0 = 0x88, k_1 = 0x99, k_2 = 0xAA \) and \( k_3 = 0xBB \).

The crucial observation over 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 in the case of Fig. 1b of a key leakage from the linear and nonlinear transformations, note that this shows the key leakage from linear transformations without nonlinear transformations. For simplicity, we abuse the notation by skipping the superscript of \( f^i \) by letting \( f = f^i \) and use the subscript to \( f \) indicating its Boolean functions. We can see that linear transformations with \( M \) hide three subkeys \( k_0, k_2 \) and \( k_3 \) (the Walsh transforms score 0), but expose \( k_1 (0x99) \) from \( y_1(x) \) (the Walsh transforms score 256 in Fig. 2b). This gives us that linear transformations produce well-balanced outputs with an overwhelming probability, but this is not always guarantee a reliable protection on secret keys.

We repeated the above experiment using 1,000 randomly generated \((32, 2)\) matrices. For \( \text{HW}(\omega) = 1 \), the correct key gives us that \( W_{f^i}(\omega) = 0 \) with approximately 99.7% and \( W_{f^i}(\omega) = 256 \) with 0.3%; the average of \( |W_{f^i}(\omega)| \) is approximately 0.7 as shown in Table 1. Although this probability of \( W_{f^i}(\omega) = 256 \) is small, 1,024 Walsh transforms probably produce three peaks of the correct key 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 of the four subkeys. Consequently, this demonstrates Lemma 1 and explains why a linear transformation cannot guarantee the protection of key in white-box cryptography.

\[ \begin{array}{cccc}
\text{To be linearly transformed} & y_0 & y_1 & y_2 & y_3 \\
\hline
\text{Number of } W_{f^i}(\omega) = 0 & 255,206 & 255,205 & 255,209 & 255,203 \\
W_{f^i}(\omega) = 256 & 794 & 795 & 691 & 797 \\
\end{array} \]

Table 1: Statistic of \( W_{f^i} \) scores calculated with 1,000 randomly generated \((32, 2)\) matrices.
3.2 Analysis of Block Invertible Square Matrix

To enhance our analysis on the cause of the key leakage, we perform 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 $(4, 2), (6, 2), \ldots,$ and repeats it $(n - 2)/2$ times [40]. Note that every $2 \times 2$ submatrix in a $(n, 2)$ block invertible square
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).

matrix should be invertible by the definition and all $2 \times 2$ invertible matrices in GF(2) are as follows:

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

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

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

then its resulting matrix will be

$$
\begin{bmatrix}
1 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 \\
1 & 0 & 1 & 1 \\
0 & 1 & 1 & 0 \\
\end{bmatrix}
$$

if the technique in [40] is used. Another case of an initialization with

$$
\begin{bmatrix}
0 & 1 \\
1 & 0 \\
\end{bmatrix}
$$

will produce

$$
\begin{bmatrix}
0 & 1 & 0 & 1 \\
1 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 \\
1 & 0 & 1 & 1 \\
\end{bmatrix}
$$

When generating a $(32, 2)$ matrix through this process, 1s appear more frequently. To do so, we performed the following experiment to test whether or
not this overweight HW of the block invertible matrix is one of the reasons for key leakage. We randomly generated a balanced non-invertible (singular) $32 \times 32$ 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.

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 insert a random byte at a particular position in the four-byte intermediate value $y_i \in \{0,1,2,3\}(x)$ and then perform a linear transformation with a $(40, 2)$ block invertible matrix $M^*$ to check if any key leakage occurs. Among the five inserting positions $\rho_1 - \rho_5$ of $y_0$, for example,

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

we select $\rho_i$, where $i \in [1, 5]$, and then insert different $\gamma \in_R \text{GF}(2^8)$ at $\rho_i$ for each $x \in \text{GF}(2^8)$. Let $y^*_0(x)$ denote $y_0(x)$ after the random byte insertion, and let $f^*(x) = M^* \cdot y^*_0(x)$. Then we can define the Walsh transforms with respect to $f^*$:

$$W_{f^*}^i(\omega) = \sum_{x \in \{0,1\}^8} (-1)^{f^*_i(x) \oplus x \cdot \omega}$$

for 40 Boolean functions

$$f^*_i \in \{1, \ldots, 40\}(x) : \{0,1\}^8 \rightarrow \{0,1\}.$$ 

With 1,000 randomly generated $M^*$, we computed $W_{f^*}^i(\omega)$. As a result, Table 2 gives us that the correct key results in $W_{f^*}^i(\omega) = 0$ with approximately 5% (The max and average $|W_{f^*}^i(\omega)|$ are about 72 and 12.7, respectively). 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. This implies that the encoding imbalance increases in the linear transformation with an unbalanced intermediate value by inserting a random byte.

For comparison, 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 $x \in \text{GF}(2^8)$. 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 $W_{f^\gamma}^i(\omega)$.
4. Compute % of $W_{f^\gamma}^i(\omega) = 0$ and the average $|W_{f^\gamma}^i(\omega)|$. 

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 $M^{b}$.

5. Compute the cosine similarity between the distributions of $W_{f}^{\gamma}(\omega)$ and $W_{f'}^{\gamma}(\omega)$ for each $\rho_{i}$.

As a result, we have $W_{f}^{\gamma}(\omega) = 0$ with approximately 5% (The max and average $|W_{f}^{\gamma}(\omega)|$ are about 76 and 12.74, respectively) 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 $W^f_1(\omega)$ and $W^f_1(\omega)$ is about 0.25.

<table>
<thead>
<tr>
<th>% of $W^f_1(\omega) = 0$</th>
<th>$\rho_1$</th>
<th>$\rho_2$</th>
<th>$\rho_3$</th>
<th>$\rho_4$</th>
<th>$\rho_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5.05</td>
<td>5.06</td>
<td>4.93</td>
<td>5.0</td>
<td>5.04</td>
</tr>
<tr>
<td></td>
<td>(0.03)</td>
<td>(0.07)</td>
<td>(0.05)</td>
<td>(0.05)</td>
<td>(0.04)</td>
</tr>
<tr>
<td>Average $</td>
<td>W^f_1(\omega)</td>
<td>$</td>
<td>12.73</td>
<td>12.75</td>
<td>12.76</td>
</tr>
<tr>
<td></td>
<td>(0.02)</td>
<td>(0.01)</td>
<td>(0.01)</td>
<td>(0.01)</td>
<td>(0.01)</td>
</tr>
<tr>
<td>Similarity with $W^f_1$</td>
<td>&gt; 0.999</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2: $W^f_1$ after inserting a random byte at each inserting position (the standard deviation in parenthesis), and the cosine similarity of the distributions between $W^f_1$ and $W^f_{1\gamma}$.

In order to visualize this effect of inserting a random byte, we select $\rho_5$ and calculate the sum of the imbalances of $W^f_1(\omega)$ for each key candidate with $\omega$ such that $HW(\omega) = 1$ as follows:

$$\Delta^f_\kappa \in \{0,1\}^8 = \sum_{\omega=1,2,\ldots,128} \sum_{i=1,\ldots,40} |W^f_1(\omega)|.$$

Fig. 5 shows $\Delta^f_\kappa \in \{0,1\}^8$ and we can see that the correct subkeys 0x88 - 0xBB 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 studies have shown that rows of HW 1 in the 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 $32 \times 32$ linear transformation on the SubBytes output multiplied with the decomposed MixColumns rather than an $8 \times 8$ linear transformation on the SubBytes output. Thus, it is complicated for an attacker to analyze all possible linear combinations. Our analysis 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.
Fig. 5: Sum of the imbalance of $W_{f^i}(\omega)$ for all key candidates. Red arrow: the correct key.

**Acknowledgment**

This work was supported by Institute of Information & communications Technology Planning & Evaluation (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]).
References


