Side-Channel Power Analysis of XTS-AES

Chao Luo*, Yunsi Fei* and A. Adam Ding†
*Department of Electrical & Computer Engineering, Northeastern University, Boston, MA 02115, USA
†Department of Mathematics, Northeastern University, Boston, MA 02115, USA
Email: luochao@ece.neu.edu, yfei@ece.neu.edu, a.ding@neu.edu

***Abstract*—XTS-AES is an advanced mode of AES for data protection of sector-based devices. Compared to other AES modes, it features two secret keys instead of one, and an additional tweak for each data block. These characteristics make the mode not only resistant against cryptoanalysis attacks, but also more challenging for side-channel attack. In this paper, we propose two attack methods on XTS-AES overcoming these challenges. In the first attack, we analyze side-channel leakage of the particular modular multiplication in XTS-AES mode. In the second one, we utilize the relationship between two consecutive block tweaks and propose a method to work around the masking of ciphertext by the tweak. These attacks are verified on an FPGA implementation of XTS-AES. The results show that XTS-AES is susceptible to side-channel power analysis attacks, and therefore dedicated protections are required for security of XTS-AES in storage devices.**

I. Introduction

XTS-AES [1], a mode designed specifically for the data protection on block-oriented storage devices, has been widely used on hard disk drives (HDD), solid-state disks (SSD) and flash cards. It protects the confidentiality of sensitive data even when the adversaries have physical access to the device. The US National Instituted of Standards and Technology (NIST) has approved its usage in 2010 [2]. It adopts AES as its block cipher, but involves two phases of AES encryption with different keys. In the first phase, block tweaks (128-bit data blocks) are generated through one AES encryption with a tweak key followed by a multiplication-by-2 in finite field GF(2^{128}). The second phase is for data encryption, where every input and output block of AES is XORed with a distinctive block tweak generated from the first phase. In order to reveal the plaintext of the data stored on a protected device, the attacker has to infer the block tweaks in addition to the data encryption key, all unknown to adversaries.

The security of different modes of AES under side-channel analysis has been evaluated by many researchers. Jaffe described a differential power analysis (DPA) against the counter mode (CTR) without knowing the initial counter [3]. In [4], Jayasinghe *et al.* studied the common advanced modes, including Cipher Block Chaining (CBC), Cipher Feedback (CFB), Output Feedback (OFB) and CTR, under side-channel analysis in a quantitative way. Recently the security of the new XTS-AES mode has also attracted a lot of research attention. Unterluggauer *et al.* pointed out the data encryption key can be extracted by one more attack on the second-last round [5] in addition to the last round DPA. However, the attack on the second-last round needs to deal with MixColumn operation [6], which increases the attack complexity with four subkey bytes. Side-channel information may also be leaked out through modular multiplication in finite field, and be used to launch attacks. In [7], [8], Belaïd *et al.* described a side-channel analysis of finite field multiplication. Their attack works when the secret part of multiplication is constant and the known part changes. However, their attack is not applicable to XTS-AES, where the secret part keeps changing by multiplication of a constant of 2 in XTS-AES's modular multiplication.

In this paper, we first propose a side-channel power analysis on the finite field multiplication in XTS-AES, aiming at extracting the block tweaks. The bit recovery errors under noisy environment can be detected and corrected. In addition, we also design a correlation power attack (CPA) [9] of the data encryption phase without attacking the second-last round at the cost of only doubling the power traces.

The rest of the paper is organized as follows. We give an introduction of the XTS-AES algorithm and the attack and leakage models in Section II. In Section III, we present the first side-channel power analysis attack on the modular multiplication. We describe how to eliminate errors, and give experimental results on an FPGA implementation. In Section IV, an improved CPA is designed as an alternative attack on XTS-AES, taking advantage of the relationship between consecutive block tweaks. Finally, conclusions are presented in Section V.

II. Preliminaries

In this section, we first give a brief introduction of XTS-AES and its modular multiplication. We then describe our attack model and the leakage model of XTS-AES on hard disk drives.

A. XTS-AES Algorithm

XTS-AES mode provides stronger security than Electronic Codebook Mode (ECB), and can be parallelized for better performance [10] compared with CBC Mode. It protects data against ciphertext manipulation and cut-and-paste attacks [11]. XTS-AES mode is an instantiation of Rogaway's XEX (XOR Encrypt XOR) tweakable block cipher [12]. The data to be encrypted is divided into equal-size data units containing multiple data blocks (*e.g.*, each at the size of 128 bits for AES-128). Usually the size of data unit agrees with the sector size of the storage devices, *e.g.*, 512 bytes before and 4K bytes after 2011 [13]. In this paper, we consider the sector size of 4K, which consists of 256 128-bit blocks. Each data unit is assigned an initial 128-bit unit tweak, which is normally

978-3-9815370-9-3/DATE17/©2017 EDAA

the logical sector address where the data unit is stored. The encryption of one AES data unit is shown in Fig. 1. The unit tweak i is encrypted by a tweak key Key_T, generating the first block tweak T_0. T_0 then goes through a modular multiplication α, a primitive element of $GF(2^{128})$ which is 2 here, to produce the next block tweak T_1, and so on for further block tweaks. Each block tweak is applied to XOR both the plaintext before the encryption, and the output after the encryption to generate the final ciphertext.

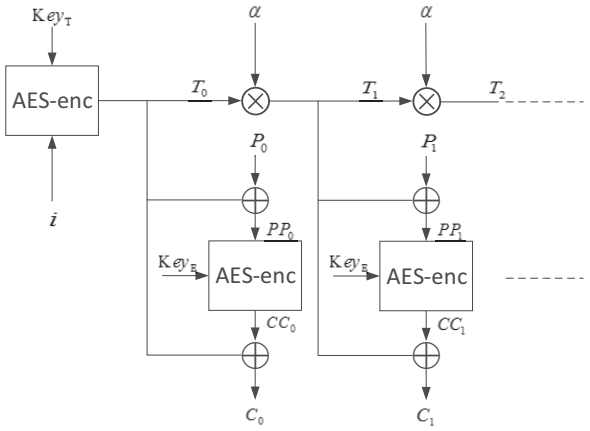

Fig. 1: Diagram of XTS-AES block encryption

The operation of modular multiplication of α can be described as below. With each block tweak T_j represented as a vector of 128 bits $(T_j[127], T_j[126], \ \ldots \ , T_j[0])$, the relationship between T_j and T_{j+1} is:

$$T_{j+1}[i] = \begin{cases} T_j[(i-1)\%128] \oplus T_j[127] & \text{for } i \in \{1,2,7\} \\ T_j[(i-1)\%128] & \text{otherwise} \end{cases} \tag{1}$$

where $\oplus$ is bit XOR, and $\%$ is integer modular operation. It means that when $T_j[127] = 0$, the modular multiplication is a round left shifting; otherwise, it is a round left shifting plus inverting the 1^{st}, 2^{nd} and 7^{th} bits.

B. Attack and Leakage Model

In the attack model, we assume that the adversary only has the knowledge of the stored ciphertext, C_j, which can be obtained from the memory storage device physically. It is also the threat model considered in the design of XTS-AES [1]. The plaintext and the block tweaks are never revealed. Although the initial unit tweak i, the logical sector address, can be observed by probing the memory bus [5], it can be easily hidden by adding an unknown offset onto it or other obfuscation methods. We consider it unknown for general attacks. The output of the data encryption, CC_j, is unknown because of the unknown block tweak T_j. For the first-phase AES encryption with Key_T, both its plaintext and ciphertext are unknown.

For the leakage model, we assume the Hamming weight or Hamming distance of intermediate data is leaked with some Gaussian noise. The Hamming weight model is commonly used for attacking microprocessor implementations, and the Hamming distance model is more suitable for hardware implementations such as FPGA and ASIC.

The goal of our side-channel power analysis is to recover both the encryption and tweak keys. The block tweak is a key element here. By knowing the T_0 of every data unit associated with different i, the tweak key Key_T can be recovered by a straightforward CPA considering T_0 as the ciphertext. For the encryption key Key_E recovery, the ciphertext on the disk is first demasked by the block tweak, then CPA can be applied with the knowledge of the encryption output CC.

III. Horizontal Attack: Analysis of Modular Multiplication

We first attack the modular multiplication in XTS-AES to recover the block tweaks, where the operations go horizontally within a data unit encryption, as shown in Fig. 1, to generate block tweaks consecutively. We call such attack *horizontal* attack. We first ignore the effect of noise, and demonstrate how to get the value of T_0 bit by bit. We then show how to make correction in case of errors induced by noise.

A. Block Tweak Leakage Analysis without Noise

We assume the result of modular multiplication, T_j, is stored in a register. Its Hamming weight and Hamming distance are $\text{HW}(T_j)$ and $\text{HW}(T_{j+1} \oplus T_j)$ for $j \geq 0$.

For the Hamming weight leakage model, we can obtain the Hamming weight from simple-power analysis (SPA) of the side-channel power information. If $T_j[127]$ is 0, according to (1), T_{j+1} is T_j round left shifted by one, resulting in the same Hamming weight $\text{HW}(T_{j+1}) = \text{HW}(T_j)$. Otherwise, there are three bits inverted in T_{j+1}, each of which will change $\text{HW}(T_{j+1})$ by either $+1$ or -1. The total effect is that $\text{HW}(T_{j+1})$ changes by either ± 1 or ± 3. We denote the Hamming weight difference between two consecutive block tweaks in a data unit as:

$$\Delta HW_{j+1} = \text{HW}(T_{j+1}) - \text{HW}(T_j) \text{ for } j \geq 0 \tag{2}$$

It can be summarized as below,

$$\Delta HW_{j+1} = \begin{cases} 0 & \text{for } T_j[127] = 0 \\ \Delta & \text{for } T_j[127] = 1, \Delta \in \{\pm 1, \pm 3\} \end{cases} \tag{3}$$

By observing ΔHW_{j+1}, we can tell what the value $T_j[127]$ is. After having $\{T_0[127], \ T_1[127], \ \ldots \ , \ T_{127}[127]\}$, the highest bits of 128 consecutive blocks tweaks within one data unit, we can reconstruct T_0 based on (1):

$$\begin{cases} T_0[n] = T_{127-n}[127] \text{ for } 127 \geq n \geq 7 \\ T_0[n] = T_{127-n}[127] \oplus T_0[121+n] \text{ for } 6 \geq n \geq 2 \\ T_0[1] = T_{126}[127] \oplus T_0[122] \oplus T_0[127] \\ T_0[0] = T_{127}[127] \oplus T_0[121] \oplus T_0[127] \oplus T_0[126] \end{cases} \tag{4}$$

For the Hamming distance leakage model, we define the differential of two consecutive Hamming distances in the register as:

$$\begin{aligned} \Delta HD_{j+2} &= HDT_{j+2} - HDT_{j+1} \\ &= \text{HW}(T_{j+2} \oplus T_{j+1}) - \text{HW}(T_{j+1} \oplus T_j) \end{aligned} \tag{5}$$

We consider all the 4 cases of $(T_j[127],\ T_{j+1}[127])$, and illustrate the composition of T_j, T_{j+1}, and T_{j+2} in Fig. 2. We represent each bit of T_j as a box with an index number, and also T_{j+1} and T_{j+2} using bits of T_j. A number with a bar over it means its inverse, and two bars means the inverse of its inverse, which is itself. We denote a set $\mathcal{I} = \{0, 1, 2, ...127\}$, and $\mathcal{I}\backslash\mathcal{A}$ as a subset in $\mathcal{I}$ complementing the subset $\mathcal{A}$.

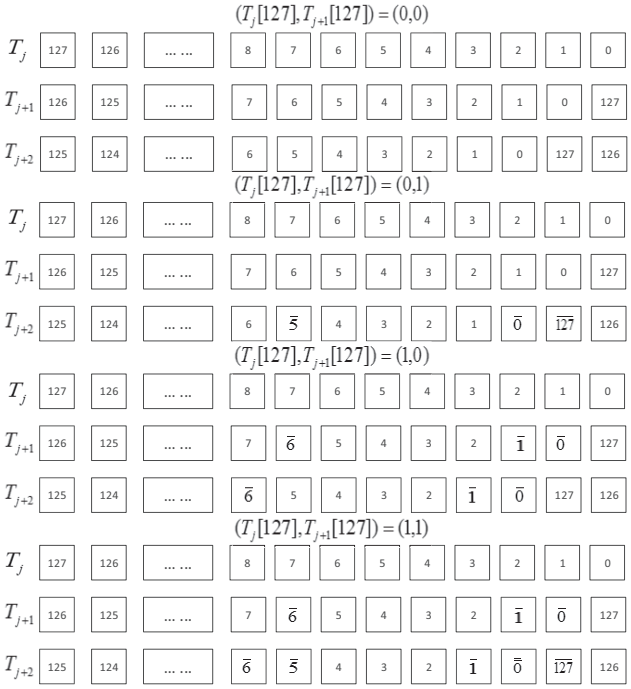

Fig. 2: Operation of modular multiplication for different cases of $\{T_j[127], T_{j+1}[127]\}$

Under the case of $(T_j[127],\ T_{j+1}[127]) = (0, 0)$, the bit values of all the three tweaks do not change, only the bit positions in T_{j+1} and T_{j+2} change. The Hamming distances HDT_{j+1} and HDT_{j+2}, are both equal to the sum of XOR results of every bit with its previous bit. That's

$$HDT^{00}_{j+1} = HDT^{00}_{j+2} = \sum_{i\in\mathcal{I}} T_j[i] \oplus T_j[(i+1)\%128] \quad (6)$$

$$\Delta HD^{00}_{j+2} = HDT^{00}_{j+2} - HDT^{00}_{j+1} = 0 \quad (7)$$

The superscription is used for different cases of $(T_j[127], T_{j+1}[127])$.

For the case of (0, 1), there are three bits, $T_j[127]$, $T_j[0]$, $T_j[5]$ being flipped in T_{j+2} at different positions, and therefore HDT_{j+1} and HDT_{j+2} are different in three places. We have

$$\begin{aligned}\Delta HD^{01}_{j+2} &= HDT^{01}_{j+2} - HDT^{01}_{j+1} \\ &= \overline{T_j[127]} \oplus T_j[0] + \overline{T_j[0]} \oplus T_j[1] + \overline{T_j[5]} \oplus T_j[6] \\ &- T_j[127] \oplus T_j[0] - T_j[0] \oplus T_j[1] - T_j[5] \oplus T_j[6]\end{aligned} \quad (8)$$

We know $\overline{a} \oplus b - a \oplus b = \overline{a \oplus b} - a \oplus b$, with value of 1 or -1. Therefore we have $\Delta HD^{01}_{j+2} \in \{-3, -1, 1, 3\}$ by enumerating all the possibilities.

The case of (1, 0), is similar to (0, 1), with only the positions of the difference varied, we also have $\Delta HD^{10}_{j+2} \in \{-3, -1, 1, 3\}$

For (1, 1), with the same analysis we have $\Delta HD^{11}_{j+2} = 0$. In conclusion,

$$\Delta HD_{j+2} = \begin{cases} 0 & \text{for } T_j[127] = T_{j+1}[127] \\ \Delta & \text{otherwise}, \Delta \in \{\pm 1, \pm 3\} \end{cases} \quad (9)$$

Different from the Hamming weight model, we can only recover the relationship between any two consecutive $T_j[127]$ and $T_{j+1}[127]$ bits by SPA of the power trace. To reconstruct T_0, we start with a guess of $T_0[127]$, and $T_j[127]$ for $j \geq 1$ can be determined according to (9). Then T_0 can be reconstructed in the same fashion as in Hamming weight leakage model by using (4), but with two candidates instead of one (one for $T_0[127] = 0$, one for $T_0[127] = 1$). The erroneous one can be eliminated through the verification method in next section.

A simulation of HW(T_j) and HDT_{j+1} with a random T_0 is shown in Fig. 3. If $T_j[127] = 1$, HW(T_j) and HDT_{j+1} are represented as $\circ$, otherwise $*$. For the Hamming weight model case, we see if $T_j[127] = 1$, the next Hamming weight changes. For the Hamming distance model case, if $T_j[127]$ is not equal to the previous one, the next Hamming distance changes.

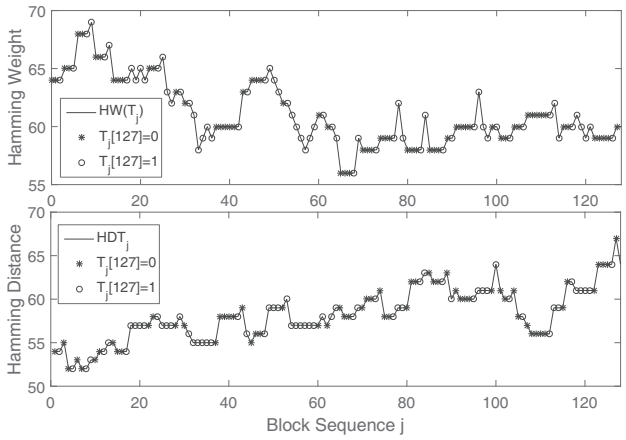

Fig. 3: Hamming weight and distance of block tweak

B. Block Tweak Leakage Analysis with Noise

Because the following analysis is similar for the Hamming weight and Hamming distance model, we target the Hamming distance model here only. The physical side-channel measurements (*e.g.*, power consumption) P_js are noisy observations of the targeted Hamming distances $P_j = \epsilon HDT_j + N_j$, $j \geq 1$, where N_j is the Gaussian noise with a constant mean and σ standard deviation, whose distribution is denoted by $\mathcal{N}(0, \sigma^2)$. The side-channel signal-to-noise ratio (SNR) is defined as the ratio between the variance of the deterministic component of the power consumption, ϵHDT_j, and the variance of random noise, N_j. For the 128-bit T_j, $SNR = 32\epsilon^2/\sigma^2$. The attacker can find the noise variance σ^2 by repeating the encryption of a data unit at certain logical address (the same i, and therefore the same T_0 and the following modular multiplication) and

finding the variance of the power measurement at a fixed j position, which is just due to the noise. He can find the signal variance by collecting the power consumptions across different j within one data unit. The variance of such power consumptions is therefore $Var(\epsilon HDT_j) + \sigma^2$. For simplicity, we consider the normalized (by ϵ) side-channel information so that

$$P_j = HDT_j + N_j \text{ for } j \geq 1. \tag{10}$$

With the noisy observations, the maximum likelihood principle [14] leads us to search for the T_0 value that will maximize the likelihood. Under Gaussian noises for an observed trace of $(P_1, ..., P_n)$ within one data unit, this becomes the least square estimator that minimizes the mean squared error (MSE):

$$MSE = \frac{1}{n}\sum_{j=1}^{n}(P_j - HDT_j^*)^2, \tag{11}$$

where HDT_j^* ($j \geq 1$) is the Hamming distance value corresponding to the T_j^* produced by the modular multiplication of (1) with a guessed value T_0^*. Since $(P_j - HDT_j^*)^2 = (HDT_j - HDT_j^* + N_j)^2$, for large n, $MSE(T_0^*) \approx \sigma^2$ for $T_0^* = T_0$, and $MSE(T_0^*) \approx 64 + \sigma^2$ for $T_0^* \neq T_0$. Therefore, when σ^2 is small (i.e., the SNR is big), the least square estimator can distinguish the correct tweak value T_0. The attacker can always increase the SNR by attacking the averages of r traces $(\bar{P}_1, ..., \bar{P}_n)$, whose noise $\bar{N}_j$ variance is σ^2/r (i.e., SNR is increased r times to $32r/\sigma^2$) according to the Central Limit Theorem.

Finding the minimum of MSE in (11), however, is prevented by the data complexity of enumerating all the 2^{128} possible values for T_0. Therefore, we first recover the relationship $\delta_j = T_j[127] \oplus T_{j+1}[127]$ from the noisy power observations, and then reconstruct T_0 as shown in Section III-A.

Relationship δ_j Recovery and Its Bit Error Rate: By (9), $\delta_j = 0$ when $\Delta HD_{j+2} = 0$ and $\delta_j = 1$ when $\Delta HD_{j+2} \neq 0$. Hence we judge whether ΔHD_j is zero from the differential of the noisy power consumption:

$$\Delta P_j = P_j - P_{j-1} = \Delta HD_j + \Delta N_j \tag{12}$$

where $\Delta N_j = N_j - N_{j-1}$ follows the distribution $\mathcal{N}(0, \tilde{\sigma}^2 = 2\sigma^2)$, and $\Delta HD_j \in \{-3, -1, 0, 1, 3\}$ with the corresponding probability {1/16, 3/16, 1/2, 3/16, 1/16}. Hence the absolute value of ΔP_j tends to be smaller when $\Delta HD_j = 0$ than when $\Delta HD_j \neq 0$, and the maximum likelihood test will judge the bit value $\delta_j = 1$ ($\Delta HD_{j+2} = 0$) if the observed $|\Delta P_{j+2}|$ is smaller than a threshold TH. The bitwise error rate (BER) of such recovered δ_j is therefore:

$$\begin{aligned} BER =& \frac{3}{8}[\Phi_{\tilde{\sigma}}(TH-1) - \Phi_{\tilde{\sigma}}(-TH-1)] \\ &+ \frac{1}{8}[\Phi_{\tilde{\sigma}}(TH-3) - \Phi_{\tilde{\sigma}}(-TH-3)] + \Phi_{\tilde{\sigma}}(-TH) \end{aligned} \tag{13}$$

where $\Phi_\sigma(\cdot)$ denote the cumulative distribution function of $\mathcal{N}(0, \sigma^2)$. We can find the best TH and the minimal error rate for a given SNR, by numerically minimization (13). The result is shown in Table I. When the SNR is big ($SNR > 100$), the optimal threshold can be solved analytically as $TH \approx 1/2 + \log_e(8/3)\tilde{\sigma}^2$.

TABLE I: Threshold and BER for Different SNR

SNR	8	16	32	64	128	256	512
TH	3.03	2.26	1.74	1.33	0.98	0.74	0.62
BER	0.460	0.430	0.385	0.330	0.266	0.182	0.093

With probability $(1 - BER)^{127}$, the noisy attack can recover all 127 δ_j bits correctly as the noiseless attack of Section III-A. Thus, for the noisy attack to achieve the same result as the noiseless attack with 99% probability, we need $BER = 0.00008$ which corresponds to SNR= 3750. As mentioned above, by attacking the averages of r traces, SNR increases r times. With big enough r, the attacker can ensure the attack is the same as the noiseless attack. The two initial guess of $T_0[127]$, together with the $(\delta_1, ..., \delta_{127})$ results in two guessed T_0 values, with the correct one minimizing the MSE in (11).

Trade-off between Search Complexity and Traces Used: We can lower the number of traces r needed by not insisting all δ_js are recovered correctly as in the noiseless attack. Assuming there are up to m flipped $(T_0[127], \delta_1, ..., \delta_{127})$ bits, we search the minimum MSE over the recovered T_0^* values. Searching among m bit flips ($T_0[127]$ and (m-1) δ_j bits) requires a data complexity of $2\binom{127}{m-1} = 2 \times 127!/((m-1)!(128-m)!)$ as listed in Table II. If the highest allowable complexity is 2^{64}, then we can search among at most $m = 16$ bit flips. The correct tweak value is included in this search space if there are at most 15 incorrect δ_j bits. To ensure at most 15 errors with 99% probability, we need $BER = 0.066$ which corresponds to SNR= 650. Compared to the SNR= 3750 needed for correctly recovering all δ_j bits, this expanded search trades the complexity for a reduction of SNR of about six times (corresponding to six times less traces).

TABLE II: Complexity of Search Among Erroneous Bits

Bits	1	2	4	8	16
Complexity($\log_2$)	1.0	8.0	19.4	37.5	64.5

C. Experimental Results

We implemented an XTS-AES algorithm on an SASEBO-GII (FPGA) board [15], dedicating one cycle to each modular multiplication. We first send the fixed sector address to the FPGA, which will fix the unit tweak i. Then a sector of data (256 blocks) is sent to the FPGA for encryption. The first-phase encryption will encrypt the unit tweak i with Key_T to generate the first block tweak T_0. For the following second-phase encryption, we use 128 consecutive modular multiplications to compute the corresponding block tweaks. There is a trigger signal associated with every block tweak generation and block encryption, allowing us to record one power trace for each data block in a unit.

The first cycle of the power trace is doing modular multiplication. We pick the point of interest in the power trace where

the variance is the largest in that cycle as P_j, and normalize it to fit (10). To measure the variance of noise, we keep both the sector address and data constant, repeat the encryption of first two blocks multiple times, and calculate the variance at the point of interest as the estimation of noise variance. On the FPGA, we have the noise variance of 3 (SNR=10.7), and the distribution of ΔP_j under the condition of ΔHD_j is shown in Fig. 4. The distributions are distinct for different ΔHD, but the overlap is significant, especially for $\Delta HD \in \{-1, 0, 1\}$. With such a small SNR (< 650), the attack cannot succeed even with the aforementioned expanded search. We then attack the average of 500 traces (which increases the SNR to be above 5000). The averaged ΔP_j and ΔHD_j for $2 \leq j \leq 128$ are shown in Fig. 5. For each j, they are very close to each other. With a threshold of 0.5 (two symmetrical lines for 0.5 and -0.5), the zero-valued and non-zero-valued ΔHD_j can be determined correctly by comparing the observed $|\Delta P_j|$ and the threshold. We can see that this essentially becomes a noiseless attack as predicted in previous analysis, and the true value of T_0 is recovered.

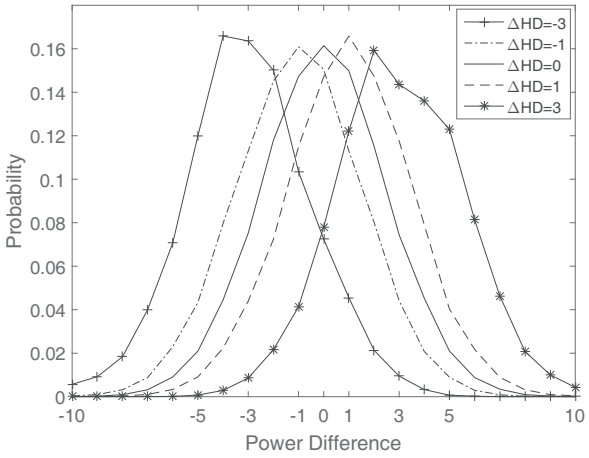

Fig. 4: Distribution of power difference ΔP_j

IV. Vertical Attack: CPA on XTS-AES

Without obtaining T_0 value from the first attack, alternatively, we propose a variant CPA on the second-phase AES encryption with Key_E. This attack requires a number of encryption runs, each with a data unit at the same sector address and block address (therefore the same block tweak) but with different plaintext, and we call it *vertical* attack as it involves multiple plaintexts at the same block. In this vertical attack, we first recover the tweaked last round key (*i.e.*, the XOR result of a tweak and the key), and propose a new method to recover both the encryption key and the block tweak without attacking the second-last round.

For simplicity, we fix the block tweak to T_0 in the analysis. While the objective is to recover the last round key $Rkey$ and block tweak T_0, the CPA first recovers the tweaked key $Rkey \oplus T_0$, denoted as RT_0.

We analyze the Hamming distance leakage model. Depending on implementations, there are two kinds of Hamming

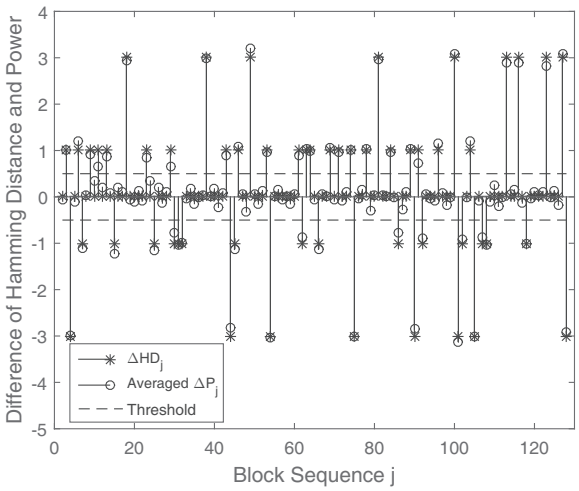

Fig. 5: Comparison of ΔHD_j and ΔP_j

distance leakage. If the implementation does not store the encryption output, CC_j, in register, but only stores the tweaked ciphertext, C_j, the Hamming distance is between the last round input state and the ciphertext. C_j. A straightforward CPA will recover the tweaked last round key RT_0 directly. If the implementation stores every AES round state, i.e., including CC_j, then the last-round Hamming distance is between the input state and CC_j, which has to be obtained from the known ciphertext C_j and T_0. The CPA attack has to guess both RT_0 and T_0. We analyze the second case in detail.

Denote HD_i as the attacked Hamming distance intermediate value related to $Rkey[i]$, the i^{th} byte of the last round key. Then HD_i can be expressed as follows.

$$S_i^{in} = \text{InvSubByte}(RT_0[j] \oplus C[j]) \quad (14)$$

$$S_i^{out} = C[i] \oplus T_0[i] \quad (15)$$

$$HD_i = \text{HW}(Sin_i \oplus Sout_i), \quad (16)$$

where $j = ShiftRow(i))$ is the byte position corresponding to i^{th} byte after the ShiftRow operation. In addition to two known ciphertext bytes, $C[i]$ and $C[j]$, HD_i involves two unknown bytes $RT_0[j]$ and $T_0[i]$.

CPA finds the maximum correlation between a power measurement value and HD_i over the 2^{16} possible values of the combination of $RT_0[j]$ and $T_0[i]$, which will reveal both $RT_0[j]$ and $T_0[i]$. However, due to the linear relationship of HD_i and $T_0[i]$, usually only the $RT_0[j]$ value is recovered reliably and we always have multiple candidates for $T_0[i]$ by the CPA.

We launch this CPA on our SASEBO-GII implementation with 20K traces. Each power trace is measured with different plaintext at the same sector address. The SNR for the second-phase data encryption is 0.128 with 16-byte Hamming distance considered as the signal. Correlation coefficients are calculated with guesses of RT_0 and T_0. Fig. 6 shows the results of CPA on the first byte of the tweaked last round key. When $T_0[0]$ is fixed to the correct value, the correlation coefficient of the correct $RT_0[0]$ is distinguished clearly from other guesses.

However when we fix the $RT_0[0]$ to the correct value and examine the correlation coefficients of different $T_0[0]$ guesses, they change in a zigzag way, and the correct value even does not yield the highest correlation. Based on this observation, the CPA attack can only identify the correct $RT_0[0]$ value with the maximum correlation.

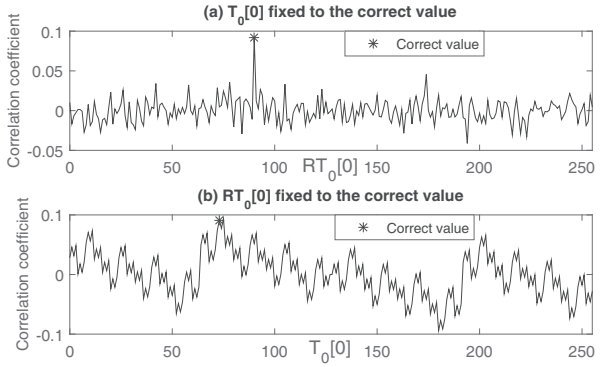

Fig. 6: Correlation coefficient with T[0] and RT[0]

To further recover the encryption key and block tweak, we propose a solution by repeating the attack on the next data block, with the block tweak fixed at T_1. Then the two CPAs reveal $RKey \oplus T_0$ and $RKey \oplus T_1$. By XORing these two results, we recover $\delta = T_0 \oplus T_1$, respectively. To further retrieve the target tweak T_0, we consider two methods. The first method is to treat this as a boolean satisfiability problem [16], with the relationship $T_1 = T_0 \otimes 2$ existing. Considering every bit of T_0 as a boolean variable, we have 128 equations for 128 variables, and thus T_0 can be solved by a SAT solver with a unique solution.

Alternatively, we propose an efficient algorithm to find T_0 by utilizing the properties of modular multiplication. We start with two guesses, $T_0[127]$ equals to 1 or 0. T_0^0 and T_0^1 denote the two possible values for T_0. As shown in Section III-A, if $T_0[127] = 0$, $\delta = T_0 \oplus (T_0 << 1)$, every bit of δ is equal to XORing two consecutive bits of T_0, so that T_0 can be recovered bit by bit from $T_0[126]$ to $T_0[0]$. If $T_0[127] = 1$, $\delta = T_0 \oplus (T_0 << 1) \oplus 0X86$, we can obtain $T_0 \oplus (T_0 << 1)$ from $\delta \oplus 0X86$, and can recover all the bits of T_0 similarly. We then use the last bit of δ to select the correct T_0 value out of the two options, because $\delta[0] = T_0^0[0] \oplus T_0^0[127]$ when $T_0[127] = 0$. Overall the first 127 bits of δ are used to derive the rest 127 bits of T_0, and the last one bit is used as redundancy check to filter out the incorrect guess.

Both the two proposed attacks on XTS-AES recover the block tweak T_0 at a fixed sector address. The *horizontal attack* targets the modular multiplication, collecting power traces within a data unit for consecutive block tweak generations. Because the information is retrieved in a simple power analysis way, the requirement of SNR is relatively high, but requiring much less power traces. The *vertical attack* is launched on encryption of multiple plaintexts (even though only the power consumption of the first two data blocks in each data unit is measured). It has higher tolerance of noise, but more traces are needed to succeed.

V. Conclusion

In this paper, we explore vulnerabilities of the XTS-AES algorithm to side-channel power analysis. We designed two different attacks to retrieve the block tweaks and therefore two encryption keys. Through power analysis of modular multiplication, we can obtain the block tweak value T_0 by SPA with error detection and correction. In our CPA, we attack on the first two data blocks of each data unit, and extract the tweaked key values, $T_0 \oplus Rkey$ and $T_1 \oplus Rkey$. We then utilize the relationship between T_0 and T_1 to recover T_0 and the round key $Rkey$ with very low complexity. Despite two keys and block tweaks, XTS-AES is still vulnerable to side-channel power analysis.

Acknowledgment: This work was supported in part by National Science Foundation under grants SaTC-1314655, MRI-1337854, and STARSS-1618379. Simulation code used in this paper is available at http://tescase.coe.neu.edu.

References

[1] "IEEE standard for cryptographic protection of data on block-oriented storage devices," *IEEE Std 1619-2007*, pp. c1–32, Apr. 2008.
[2] M. Dworkin, "Recommendation for block cipher modes of operation: The XTS-AES mode for confidentiality on storage devices," in *NIST Special Publication, SP 800-38E*, 2010.
[3] J. Jaffe, "A first-order DPA attack against AES in counter mode with unknown initial counter," in *Cryptographic Hardware & Embedded Systems*, 2007.
[4] D. Jayasinghe, R. Ragel, J. A. Ambrose, A. Ignjatovic, and S. Parameswaran, "Advanced modes in AES: Are they safe from power analysis based side channel attacks?" in *IEEE Int. Conf. on Computer Design*, 2014, pp. 173–180.
[5] T. Unterluggauer and S. Mangard, "Exploiting the physical disparity: Side-channel attacks on memory encryption," in *Constructive Side-Channel Analysis & Secure Design*, 2016.
[6] J. Daemen and V. Rijmen, "AES proposal: Rijndael," 1999.
[7] S. Belaïd, P.-A. Fouque, and B. Gérard, "Side-channel analysis of multiplications in GF (2^{128})," in *Int. Conf. on the Theory & Application of Cryptology & Information Security*, 2014, pp. 306–325.
[8] S. Belaïd, J.-S. Coron, P.-A. Fouque, B. Gérard, J.-G. Kammerer, and E. Prouff, "Improved side-channel analysis of finite-field multiplication," in *Cryptographic Hardware & Embedded Systems*, 2015, pp. 395–415.
[9] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *Cryptographic Hardware and Embedded Systems*. Springer, 2004, pp. 16–29.
[10] M. V. Ball, C. Guyot, J. P. Hughes, L. Martin, and L. C. Noll, "The XTS-AES disk encryption algorithm and the security of ciphertext stealing," *Cryptologia*, vol. 36, no. 1, pp. 70–79, 2012.
[11] L. Martin, "XTS: A mode of AES for encrypting hard disks," *IEEE Security & Privacy*, no. 3, pp. 68–69, 2010.
[12] P. Rogaway, "Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC," in *Advances in Cryptology-ASIACRYPT*, 2004, pp. 16–31.
[13] Seagate, "Transition to Advanced Format 4K Sector Hard Drives," http://www.seagate.com/tech-insights/advanced-format-4k-sector-hard-drives-master-ti/, [Online; accessed 2-Sept-2016].
[14] J. Neyman and E. S. Pearson, "On the Problem of the Most Efficient Tests of Statistical Hypotheses," *Royal Society of London Philosophical Transactions Series A*, vol. 231, pp. 289–337, 1933.
[15] T. Katashita, A. Satoh, T. Sugawara, N. Homma, and T. Aoki, "Development of side-channel attack standard evaluation environment," in *European Conf. on Circuit Theory & Design*, Aug 2009, pp. 403–408.
[16] M. Renauld and F.-X. Standaert, "Algebraic side-channel attacks," in *Int. Conf. on Information Security & Cryptology*, 2009, pp. 393–410.