RFTC: Runtime Frequency Tuning Countermeasure Using FPGA Dynamic Reconfiguration to Mitigate Power Analysis Attacks

Darshana Jayasinghe, Aleksandar Ignjatovic, Sri Parameswaran
School of Computer Science and Engineering, University of New South Wales, Sydney, Australia
{darshanaj, ignjat, sridevan}@cse.unsw.edu.au

ABSTRACT

Random execution time-based countermeasures against power analysis attacks have reduced resource overheads when compared to balancing power dissipation and masking countermeasures. The previous countermeasures on randomization use either a small number of clock frequencies or delays to randomize the execution. This paper presents a novel random frequency countermeasure (referred to as *RFTC*) using the dynamic reconfiguration ability of clock managers of Field-Programmable Gate Arrays – FPGAs (such as Xilinx Mixed-Mode Clock Manager – MMCM) which can change the frequency of operation at runtime. We show for the first time how Advanced Encryption Standard (AES) block cipher algorithm can be executed using randomly selected clock frequencies (amongst thousands of frequencies carefully chosen) generated within the FPGA to mitigate power analysis attack vulnerabilities. To test the effectiveness of the proposed clock randomization, Correlation Power analysis (CPA) attacks are performed on the collected power traces. Preprocessing methods, such as Dynamic Time Warping (DTW), Principal Component Analysis (PCA) and Fast Fourier Transform (FFT), based power analysis attacks are performed on the collected traces to test the effective removal of random execution. Compared to the state of the art, where there were 83 distinct finishing times for each encryption, the method described in this paper can have more than 60,000 distinct finishing times for each encryption, making it resistant against power analysis attacks when preprocessed and demonstrated to be secure up to four million traces.

1. INTRODUCTION

In this paper, we propose a novel random execution time countermeasure based on frequency randomization, *R*untime *F*requency *T*uning *C*ountermeasure –*RFTC*. In *RFTC*, we randomly choose from 3,072 distinct clock frequencies (arranged in 1,024×3 groups) which are chosen carefully and fixed during design time; these selections are done at runtime to mitigate power analysis attack vulnerabilities of FPGAs. *RFTC* uses dynamic reconfiguration of clock managers of FPGAs (such as Xilinx Mixed-Mode Clock Manager - MMCM) to generate the desired clock frequencies within FPGA to run cryptographic circuits. Our methodology can be extended to a large number of clock frequencies (limitation of equipment prevents us from collecting measurements at high frequencies). To test the effectiveness of the proposed frequency scrambling countermeasure, we perform Correlation Power Analysis (CPA) attacks and three of the most powerful attacks against random delay countermeasures: one, Dynamic Time Warping (DTW) [22], which is also referred to as dynamic/elastic alignment (DTW-CPA attacks); two, Principal Component Analysis (PCA) [12] based CPA attacks (PCA-CPA attacks); and, three, Fast Fourier Transform (FFT) based CPA attacks [16] (FFT-CPA attacks). Theoretical information leakage is evaluated using the Test Vector Leakage Assessment (TVLA) methodology described in [6].

Randomization-based countermeasures on FPGAs work by adding random delays [14], phase shifted clock-based clock delays [10, 19], random execution of dummy operations [3] or random frequency change [9] to mitigate the power analysis attack vulnerabilities. The random frequency countermeasures, so far proposed in the literature, have used relatively fewer number of clocks to randomize the power dissipation (four random clocks in [9] and eight phase shifted clocks in [10, 19]). Therefore, the randomness achieved by previously proposed randomization-based countermeasures are not significantly large enough to mitigate powerful power analysis attacks.

The rest of the paper is organized as follows. Section 2 compares the related work and describes the contributions of this paper. The background is briefly explained in Section 3. Section 4 and Section 5 present the proposed *RFTC* countermeasure. The experimental setup is explained in Section 6, and the results are presented and discussed in Section 7. The discussion and the future work of this paper are presented in Section 8. The paper is concluded in Section 9.

2. RELATED WORK

Random execution time countermeasures on FPGAs are enabled by the addition of random delays (such as Random Delay Insertion (RDI) using buffers [14] and phase shifted clocks [10]), random execution of dummy operations (such as Random Clock Dummy Data (RCDD) [3]) or random frequency change [9] to misalign power traces. The level of security provided by randomization-based countermeasures depends on the number of distinct delays or cumulative delays [15].

In the RDI countermeasure proposed in [14], a delay chain was formed using buffers, each adding delays to the outputs of the registers. Due to the propagation delays of buffers, the output signal will be delayed based on the number of preceding buffers in the buffer chain. Therefore, by choosing a buffer output randomly, a random (but from a selection of fixed values) delay can be introduced in the register outputs. The RDI countermeasure proposed in [14] was implemented on an FPGA (authors in [14] did not state the details) and was shown to be secure against CPA attacks using 256,000 encryptions (approximately 5,000 encryptions required to attack the unprotected circuit).

In [10], a single clock frequency was used to generate eight phase shifted clocks using two Phase Locked Loops - PLLs

DAC '19 The 56th Annual Design Automation Conference 2019 June 2–6, 2019 Las Vegas, NV, USA
 ISBN 978-1-4503-6725-7/19/06...$15.00
https://doi.org/10.1145/3316781.3317899

(which are integrated into MMCMs) on a Xilinx Virtex-II Pro FPGA (XC3VP7). One clock signal was chosen using a three-stage clock randomizer; the three-stage clock randomizer used in [10] required seven clock multiplexers - BUFGs [23]. The clock randomization countermeasure proposed in [10] was improved in [19] (referred to as iPPAP), so that it has a floating mean random number generator [7] to generate a uniform random number and was implemented on a Xilinx Virtex 5 FPGA. Three million traces were required to reveal the secret key using CPA attacks according to [10] (unprotected circuit required 3,000 encryptions to reveal the key), and CPA attacks were not reported against the implementation proposed in [19].

RCDD [3] processes random dummy data using a dummy data scheduler which adds random delays to cryptographic operations using a random number to misalign cryptographic operations. Authors in [3] collected 70,000 power traces and showed RCDD to be secure against CPA attacks (around 400 encryptions were needed to reveal the key for the unprotected circuit). In [9], the author used four random clocks (3×, 4×, 5× and 6× the frequency of the input clock) generated via an MMCM to drive an AES circuit randomly using a 16-bit random number generator. CPA attacks against three million power traces were carried out and could not reveal the secret key (the unprotected implementation of [9] required 600,000 encryptions to reveal the secret key).

The number of distinct delays in [14] were not reported; however, 2^n number of buffers are required to generate 2^n distinct delays [5]. The number of distinct delays in RCDD [3] was also not reported. According to [19], the method proposed in [10] has ≈15 distinct cumulative delays while iPPAP [19] could achieve ≈39 distinct cumulative delays (based on Figure [4] in [19]). Based on information provided in [9], we calculated the number of distinct cumulative delays in [9] to be ≈83. Compared to [3, 5, 9, 10, 14, 19], *RFTC* countermeasure proposed in this paper is shown to have 67,584 distinct cumulative delays/completion times.

RFTC uses the ability to reconfigure MMCMs dynamically in the modern Xilinx FPGAs and uses existing block RAMs to store reconfiguration details compared to [14] where buffers are added to generate delays. Therefore, the proposed *RFTC* uses less power (1.48× power overhead) compared to RCDD countermeasure (processes dummy data) proposed in [3] (4.4 × power overhead). Compared to seven BUFGs and two PLLs used in [10,19], *RFTC* requires just two MMCMs and up to three clock multiplexers to achieve high randomness of block cipher execution.

Last but not least, none of [3, 9, 10, 14, 19] were tested and proven to be secure against PCA-CPA, DTW-CPA and FFT-CPA attacks. *RFTC* is tested against all three aforementioned attacks and shown to be secure for up to four million encryptions.

Contribution

The main contribution of this paper is summarized as follows.

- We propose for the first time a highly random execution time countermeasure using dynamic reconfiguration of MMCMs on FPGAs to mitigate power analysis attacks.

To the best of our knowledge, this paper is the first paper where a block cipher circuit running on an FPGA has been executed with a large number (up to 3,072 different clock frequencies arranged in 1,024×3 sets demonstrated) of clock frequencies which are carefully chosen and is the first paper to propose and demonstrate the use of dynamic reconfiguration of MMCMs to achieve high clock randomization on block ciphers to mitigate power analysis attacks.

Threat Model

The threat model of this paper can be identified as that the adversary has control over the cryptographic device and can record the power dissipation of the FPGA for known plaintexts and observe the ciphertexts.

Assumptions

The generated random number to choose a clock frequency set and to choose a clock output of MMCM to perform a block cipher round are sufficiently random. However, if the generated random numbers are not sufficiently random, methods explained in [7] can be used to generate a uniform random number.

3. BACKGROUND

In this section, Xilinx MMCM, AES, CPA attacks, DTW algorithm, PCA algorithm, FFT algorithm and TVLA method are explained briefly.

MMCMs– Clock Managers on Xilinx FPGAs

Clock manager is a hardware component fabricated into Xilinx FPGAs, which can change output clock frequencies (multiply or divide), change phase and/or duty cycle of output clock for different peripherals using a Voltage Controlled Oscillator (VCO) and a set of counters [23]. Readers are advised to refer [23] for more information about Xilinx clock managers.

The dynamic reconfiguration of MMCM involves writing precalculated configuration settings (clock counter values) using a state machine (MMCM_DRP in [21]) or Advanced eXtensible Interface (AXI) bus into MMCM. In order to calculate configuration settings via a state machine, Xilinx has provided table lookups to convert desired frequency configurations into clock counter values.

Advanced Encryption Standard (AES)

Advanced Encryption Standard is the Federal Encryption Standard and was promulgated under FIPS 197 [1]. In this paper, we have used the AES circuit presented in [11] to test *RFTC* against power analysis attacks.

Correlation Power Analysis (CPA) attacks

CPA attacks are one of the most powerful side channel distinguishers proposed in the literature and work well even in the presence of noise [4]. Due to space limitation, the readers are advised to refer [4,13,15] regarding CPA attacks on AES.

Dynamic Time Warping (DTW), Principal Component Analysis (PCA) and Fast Fourier Transform (FFT)

DTW algorithm finds the shortest path between a reference trace and a misaligned trace [22] to align the misaligned trace; thus, DTW-CPA attacks deploy CPA attacks to reveal secret keys from power traces preprocessed by DTW algorithm. DTW algorithm has an $O(n^2)$ time and memory complexity (n is the number of sampling points in the power trace).

PCA-CPA power analysis attacks were proposed against randomization-based countermeasures and misaligned power traces by [20]. In PCA-CPA power analysis attacks, the principal components of the power traces are assumed to carry information regarding the secret operations which can reveal the secret key, and higher principal components are treated as noise. Thus, first principal components were used to perform PCA-CPA attacks to filter out the effects of randomization.

FFT preprocessing is capable of revealing the information of the secret key in the frequency domain when the power traces are misaligned in the time domain [17]. Once the power traces are processed using FFT algorithm, CPA attacks are carried out as explained in the Section 3 to reveal secret keys.

Test Vector Leakage Assessment (TVLA)

TVLA is the most popular methodology to evaluate information leakage which uses Welch's T-test. In TVLA, two

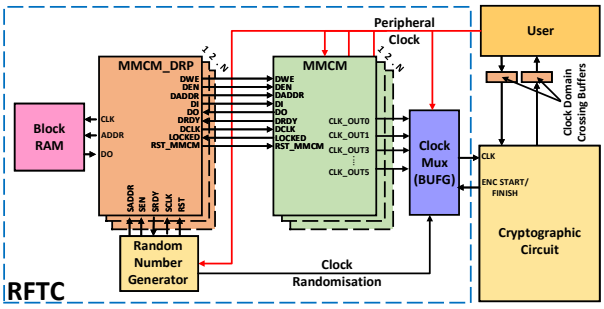


Figure 1: Block Diagram of RFTC Countermeasure using MMCM Dynamic Reconfiguration

sets of power traces are collected: for fixed inputs and for random inputs under the same secret key. Authors in [6] showed that if the separation between two power trace sets is less than ±4.5, then with a 99.99% confidence level, the secret key cannot be deduced. The readers are advised to refer to [6] for more information on the TVLA test.

4. RANDOM RUNTIME FREQUENCY TUNING AND SWITCHING

This section presents how *RFTC* uses dynamic reconfiguration of Xilinx MMCMs to tune clock frequencies to randomize the execution time of cipher circuits.

The Figure 1 depicts the architecture of *RFTC*. As shown in Figure 1, *RFTC* uses N number of MMCMs to generate clocks (N depends on the FPGA, there is at least one in small Xilinx FPGAs [19] and up to 10 in larger FPGAs), and each MMCM has M clock outputs (typically M is six in Xilinx FPGAs [21]). A clock multiplexer using BUFGs [23] is used to select a single clock output from the M outputs of each MMCM (BUFGs can switch the clocks without glitches [23]). N number of MMCM_DRP modules [21] are used to reconfigure the N MMCMs. The reconfiguration information is precalculated and stored in Block RAM (let us assume that Block RAMs hold reconfiguration information for P number of random frequencies for each M) and a random number generator using either a Linear-Feedback Shift Register (LFSR) or the floating mean method [7] is used to select the stored precalculated data in the Block RAMs (described later) to reconfigure MMCMs. A total of $M \times P$ number of configurations (or frequencies) are stored in the Block RAMs. MMCM_DRP module has to have all M clock outputs dynamically reconfigured and does not allow just one output out of M to be reconfigured.

When there are M number of outputs in each MMCM, one out of M clock outputs is selected to execute the first round. In a similar manner, the next round can also be executed using one out of M clock outputs of the MMCM. This process can be repeated R times where R is the number of clock cycles to complete the block cipher execution. Therefore, given that all M clock outputs have unique frequencies, there are $\binom{R+M-1}{R}$ number of ways execute a block cipher circuit which takes R clock cycles (Please note that since MMCM_DRP module does not allow each clock output to be programmed individually; therefore, the number of ways are not M^R). When there are P number of configurations with distinct clock frequencies, there are $P \times \binom{R+M-1}{R}$ completion times in the block cipher execution. If only one MMCM undergoes reconfiguration at a given time, then there are $P \times (N-1) \times \binom{R+M-1}{R}$ of random frequencies to execute one encryption.

MMCM reconfiguration time is much higher than the time taken to complete one encryption (AES circuits, such as [11], take 10 clock cycles). Since *RFTC* uses N MMCMs, when

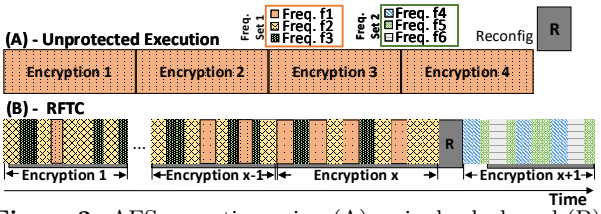


Figure 2: AES execution using (A) a single clock and (B) *RFTC(3, 1024)*

one MMCM undergoes dynamic reconfiguration, the other $N-1$ MMCMs can drive the AES circuit. The number of distinct clock frequencies *RFTC* can generate depends on the number of different configurations stored in the memory and does not depend on N, the number of MMCMs. The N number of MMCMs allow *RFTC* to run block cipher circuits using a set of different clock frequencies after each encryption without waiting for the reconfiguration to complete.

Based on the generated random number (in between 0 to $P-1$), a new MMCM configuration with a set of unique frequencies is chosen and corresponding MMCM_DRP module writes the configuration values to the MMCM and resets the MMCM. This allows *RFTC* to send ciphertext, receive plaintext and perform block cipher execution in parallel while MMCMs are being reconfigured.

Let us denote an *RFTC* implementation which uses M number of clock outputs of each MMCM with P number of clock frequencies as *RFTC(M, P)*. Let us take an example of *RFTC(3, 1024)* implementation which clocks a block cipher circuit with R=10 which results in 66 completion times per 10 rounds (calculated based on $\binom{10+3-1}{10}$). The random number generator generates a random value in between 0 and 1023 (ten random bits) to reconfigure one MMCM and generates another random number between 0 and 2 (two random bits to choose among three random clock outputs) to select a random clock output to execute one block cipher round using the second MMCM. This allows up to 1024 $\times$ 3 = 3,072 differing clock frequencies to run the block cipher circuit which results in 1,024 $\times$ 66 = 67,584 distinct completion times for 10 rounds in the AES circuit presented in [11].

5. RFTC

This section describes *RFTC* and its resistance against power analysis attacks. As explained in 4, *RFTC* deploys N MMCMs which have M clock outputs each, and each clock output can be programmed with P unique clock frequencies (the total number of frequency sets are P). Figure 2-A shows unprotected AES (which takes 10 clock cycles, R=10) being operated using a single clock frequency (frequency $f1$). When AES is being operated using a single constant clock frequency, the completion time of AES is constant and power traces of all the rounds align.

Let us consider *RFTC(3, 1024)* and $N = 2$. Figure 2-B shows the operation of *RFTC(3, 1024)*. As shown in Figure 2-B, from **Encryption 1** to **Encryption x**, the AES circuit is driven by one MMCM while the other MMCMs undergo reconfiguration; **x** depends on MMCM reconfiguration time. Xilinx Kintex 7 325T FPGA (based on which the experimental setup of this paper is built) running at 24MHz takes 34μs for reconfiguration. Therefore, *RFTC* was able to perform ≈82 encryptions (**x**=82) while the reconfiguration takes place. Once the reconfiguration is completed, the AES circuit is driven by the newly generated clock frequencies by the second MMCM while the first MMCM undergoes reconfiguration (as shown in **Encryption x+1**).

As shown in Figure 2-B, the completion time of AES driven by *RFTC(3, 1024)* implementation is not constant. Each AES encryption, as shown in the Figure 2-B, is executed

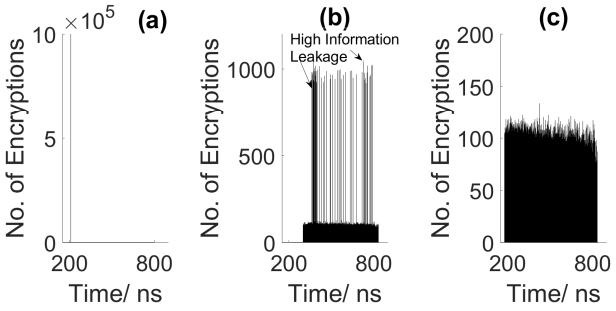


Figure 3: Completion Time Graph Generated from MATLAB for (a)-unprotected AES; (b)- *RFTC(3, 1024)* without carefully choosing random frequencies; (c)- *RFTC(3, 1024)* with carefully choosing random frequencies

using three clock frequencies which are chosen randomly to execute each round. **Encryption 1** to **Encryption x** are executed by frequency $f1$, $f2$ and $f3$ (clock frequency set 1). From **Encryption x+1** to **Encryption 2x** (not shown in the Figure 2-B), frequency $f4$, $f5$ and $f6$ (clock frequency set 2) are randomly chosen to execute each round. During each encryption, the completion time of AES is one of 67,584 possible completion times as derived in Section 4.

Each frequency set (combination) must be chosen carefully to avoid overlappings of the completion times. Overlapping of completion time happens when two different frequency sets produce identical completion times. As an example, let us assume two frequency sets for an *RFTC(3, 1024)* implementation as {12.012MHz, 40.240MHz and 30.744MHz} and {24.024MHz, 20.120MHz and 30.744MHz}. When an AES implementation is driven by the first set of clock frequencies, if two AES rounds are executed at 12.012MHz, four AES rounds are executed at 40.240MHz and four AES rounds are executed at 30.744MHz, the completion time of AES execution is 396.1ns ($\frac{2}{12.012}+\frac{4}{40.240}+\frac{4}{30.744}ns\times$ 10^3). When the identical AES implementation is driven by the second set of clock frequencies, if four rounds are executed at 24.012MHz, two rounds are executed at 20.120MHz and four rounds are executed at 30.744MHz, the completion time of AES execution is 396.1ns ($\frac{4}{24.024}+\frac{2}{20.120}+\frac{4}{30.744}ns\times$ 10^3). Such overlappings align power dissipation of the secret operation; thus, power analysis attacks can be carried out by exploiting such leakages [15].

If the random number used to choose MMCM reconfiguration is sufficiently random and uniform, the time histogram of the completion graph of an *RFTC* implementation without overlapping of completion times would be fairly uniform. Choosing frequencies for each MMCM configuration to prevent overlappings of completion times can be easily calculated by exhaustively searching for duplicated completion times. We omit the detail explanation of choosing frequencies due to space restrictions.

RFTC countermeasure belongs to the class of randomization-based countermeasures which misalign the power dissipation of cryptographic algorithm executions in the time dimension. In power analysis attacks, such as CPA, attacks based on the power dissipation of a particular round or an operation is considered [13, 15]. The time completion graph is plotted by simulation using MATLAB for one million encryptions for an AES circuit similar to [11] driven a constant stable 48MHz clock is shown in Figure 3-a. As shown in Figure 3-a, all one million encryptions take equal amount of time (208.33ns). Figure 3-b shows the completion time of *RFTC* countermeasure running the same circuit for one million encryptions similar to Figure 3-a using *RFTC(3, 1024)* implementation in-between 12MHz and 48MHz (0.5x of 24MHz and 2x of 24MHz, respectively) divided into 3,072 clock frequencies (3×1,024 sets) with 0.012MHz increments without considering the overlaps of completion times. As annotated in the Figure 3-b, the concentrated completion times (the peaks) can leak the secret key[15]. Figure 3-c shows the completion time of the *RFTC(3, 1024)* countermeasure running the same circuit for one million encryptions similar to Figure 3-b using *RFTC(3, 1024)* by carefully choosing clock frequencies to prevent overlapping. The completion time of *RFTC(3, 1024)* implementation running between 12MHz and 48MHz varies between 833.32ns and 208.33ns. As shown in Figure 3-c, there are less than 130 encryptions with identical completion times among one million encryptions (which results in low Signal to Noise Ratio - SNR). The completion time graph of the *RFTC(3, 1024)* when each frequency configuration is chosen carefully (Figure 3-c) follows a uniform distribution due to uniform random number generation of MATLAB. Therefore, when the adversary records the power dissipation, without any preprocessing, the power dissipation of the different encryptions of AES are significantly misaligned.

6. EXPERIMENTAL SETUP

RFTC was implemented on a SASEBO GIII board, which has a Xilinx Kintex 325T FPGA. In order to test the proposed *RFTC* countermeasure against power analysis attacks, we used a Verilog implementation of AES circuit presented in [11] which takes 10 clock cycles ($R = 10$) to produce ciphertext using a 128-bit key. Power traces were collected using an Agilent DSO-X 2012A Oscilloscope (has a bandwidth of 100MHz). SASEBO GIII board operates at 24MHz; the random frequency range of *RFTC* was set between 12MHz (0.5× of the clock frequency of the SASEBO GIII frequency) and 48MHz (2× of the clock frequency of the SASEBO GIII frequency). We used a 128-bit LSFR to choose a random frequency configuration from Block RAMs. CPA attacks, PCA-CPA attacks, DTW-CPA attacks and FFT-CPA attacks were performed on the power traces obtained from the last round of AES [13] as explained in Section 3. Information leakage of *RFTC* is measured using TVLA as explained in Section 3.

7. RESULTS

This section presents the results obtained by performing CPA attacks on the *RFTC* implementations and information leakage as explained the Section 3.

We collected 30,000 power traces for the AES circuit presented in [11] without any countermeasures. CPA attacks, PCA-CPA attacks, DTW-CPA attacks and FFT-CPA attacks were carried out to build a baseline AES implementation to measure the efficiency of the *RFTC*. Success Rates (SR) [18] are calculated by repeating each CPA attack 100 times with a random number of power traces. The unprotected AES circuit requires around 2,000 encryptions to reveal the secret key with CPA attacks, PCA-CPA attacks and DTW-CPA attacks. FFT-CPA attacks required around 8,000 encryptions to reveal the secret key (the figure is omitted due to space restrictions).

In order to test the effects of increasing the number of clock outputs of each MMCM and the number of distinct clock frequencies in *RFTC* implementations, we varied M (the number of clock outputs of each MMCM) such that, $M = 1$ (one clock output of each MMCM is used), $M = 2$ (two clock outputs of each MMCM is used), $M = 3$ (three clock outputs of each MMCM is used),with varying number of random frequencies ($P = \{4, 16, 64, 256, 1024\}$) for each MMCM output on *RFTC* implementations. When $M > 3$, Xilinx ISE 14.7 was unable to route the design (place and route error occurred), we believe that this is due the additional BUFGs for switching between clock outputs of MMCM.

Success rates for CPA attacks, PCA-CPA attacks, DTW-CPA attacks and FFT-CPA on *RFTC(1, 4)*, *RFTC(1, 16)*, *RFTC(1, 64)*, *RFTC(1, 256)* and *RFTC(1, 1024)* implementations are presented in Figure 4-(a), Figure 4-(b), Figure 4-

(c) and Figure 4-(d), respectively.

According to Figure 4, both CPA attacks and PCA-CPA attacks reveal the secret key from *RFTC(1, 4)* implementation in around 700,000 encryptions. When a large number of clock frequencies are used (*RFTC(1, 16)* to *RFTC(1, 1024)*), both CPA attacks and PCA-CPA attacks fail to reveal the secret key. DTW-CPA attacks can reveal the secret keys from *RFTC(1, 4)*, *RFTC(1, 16)* and *RFTC(1, 64)* in less than 200,000 encryptions. *RFTC(1, 256)* reveals the secret key for DTW-CPA attacks in around 800,000 encryption while *RFTC(1, 1024)* implementation did not reveal the secret key for a million encryptions. FFT-CPA attacks were able to reveal the secret keys from both *RFTC(1, 4)* and *RFTC(1, 16)* implementations in about 800,000 encryptions. Since DTW-CPA was able to reveal the secret keys of *RFTC(1, 64)* which has 64 distinct completion times in around 200,000 encryptions and *RFTC(1, 256)* which has 256 distinct completion times in around 800,000 encryptions, we believe countermeasure proposed in [19], which has 39 distinct completion times can be broken in less than 200,000 encryptions and countermeasure proposed in [9], which has 83 completion times can be broken using DTW-CPA attacks in less than 800,000 encryptions under similar SNR.

Success rates for CPA attacks, PCA-CPA attacks, DTW-CPA attacks and FFT-CPA attacks are carried out for *RFTC(2, 4)*, *RFTC(2, 16)*, *RFTC(2, 64)*, *RFTC(2, 256)* and *RFTC(2, 1024)* implementations, and the results are presented in Figure 5-(a), Figure 5-(b), Figure 5-(c) and Figure 5-(d), respectively.

According to Figure 5, CPA attacks, PCA-CPA attacks and FFT-CPA attacks do not reveal the secret keys of any of the *RFTC* implementations when $M = 2$. DTW-CPA attacks are successful when a small number of clocks such as *RFTC(2, 4)* and *RFTC(2, 16)* are used.

We collected four million power traces using oscilloscope for *RFTC(3, 4)*, *RFTC(3, 16)*, *RFTC(3, 64)*, *RFTC(3, 256)* and *RFTC(3, 1024)* implementations. CPA attacks, PCA-CPA attacks, DTW-CPA attacks and FFT-CPA attacks could not reveal the secret key for any of the above implementations (the graphical results are omitted due to space limitations).

TVLA test results for *RFTC(1, 4)* and *RFTC(1, 1024)* are plotted in Figure 6-a. *RFTC(1, 4)* clearly has information leakages as the TVLA values are above ±4.5. TVLA leakages of *RFTC(1,1024)* is lower than TVLA values of *RFTC(1, 4)*. As shown in Figure 6-b, the TVLA information leakage goes little over ±4.5 limits for *RFTC(2, 4)* and the TVLA value is nearly within ±4.5 for *RFTC(2, 1024)* implementation. As shown in Figure 6-c, in *RFTC(3, 4)* and *RFTC(3, 1024)* implementations, TVLA values are less than the ±4.5 limits (only plaintext load stage is vulnerable and cannot be attacked using Differential Power Analysis attacks [15]); therefore, power analysis attack methods cannot deduce the secret keys [19] of *RFTC(3, 4)* and *RFTC(3, 1024)* implementations.

RFTC countermeasure is compared with the state of the art and tabulated in Table 1. The first column of Table 1 depicts the evaluation criteria (*Eva.*), such as maximum number of cumulative delays achievable, the security parameter, resistance against power analysis attacks, time overhead, power overhead and area overhead, respectively. Second to fifth columns show the state of the art related work (*Cou.*) against the evaluation criteria. Sixth column shows results of the evaluation criteria of *RFTC(3, 1024)* implementation. The security parameter is the ratio between the maximum number of encryptions for which the particular countermeasure is shown to be effective ($T_{Count.}$) and the average number of encryption to reveal the secret key from the unprotected implementation ($T_{Unprot.}$). Therefore, the security parameter can be represented as shown in Equation 1. According to the Table 1, *RFTC(3, 1024)* implementation is the only random execution time countermeasure which can achieve over 60,000 completion times in AES with a security parameter (calculated based on Equation 1) over 2,000 while shown to be secure against CPA, PCA-CPA, DTW-CPA attacks and FFT-CPA attacks.

$$\textbf{Security Parameter} = \frac{T_{Count.}}{T_{Unprot.}} \quad (1)$$

Table 1: *RFTC* compared to the related work

Cou. / *Eva.*	RDI [14]	RCDD [3]	Phase Shifted Clocks [10]	iPPAP [19]	Clock Rand. [9]	**RFTC (3, 1024)**
# Delays	NA	NA	15	39	83	**67,584**
Sec. Para.*	≥500	≥226	100	NA	≥6	**≥ 2000**
CPA	✓	✓	✓	NA	✓	✓
PCA-CPA	NA	NA	NA	NA	NA	✓
DTW-CPA	NA	NA	NA	NA	NA	✓
FFT-CPA	NA	NA	NA	NA	NA	✓
Time ×	1.64	1.94	3.77	NA	3	**1.72**
Power ×	4.11	NA	NA	NA	1.00	**1.48**
Area ×	1.81	1.70	NA	1.05†	1.02†	**1.3**†

* is calculated based on Equation 1; NA - Not Available.
† without area of RAMB36E1, MMCM/PLL

RFTC(3, 1024) takes 20 Block RAMs (RAMB36E1 components) to store reconfiguration details of 3,072 clock frequencies (3×1,024 sets). According to Xilinx XPower tool, *RFTC(3, 1024)* has power overhead of 1.48× compared to the unprotected AES implementation. The time overhead of *RFTC* depends on the frequency range. On experimental setup, the average time overhead was 1.72× for one million encryptions. Due to space restrictions, we omitted resource consumption of other *RFTC* implementations.

8. DISCUSSION AND FUTURE WORK

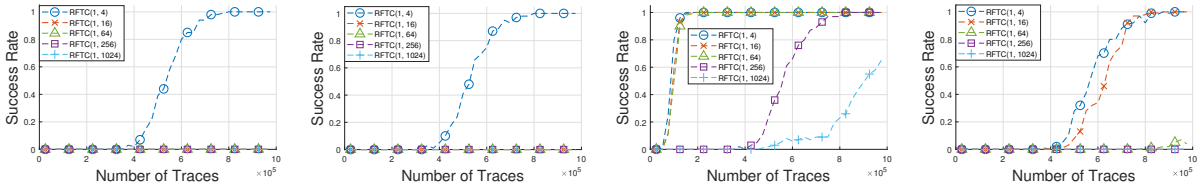
PCA-CPA attacks perform similarly to CPA attacks without any preprocessing against *RFTC*. We believe that this is due to the fact that when higher randomization is achieved, orthogonal transformation calculated in PCA is less effective. FFT-CPA attacks perform better than CPA attacks and PCA-CPA attacks. DTW-CPA attacks can align the traces when the number of frequency sets are small ($P = 4$ or $P = 16$). When a large number of clock frequencies are used, DTW fails to align the traces. When the frequency is changed, the shape of the power trace changes, thus failing to align traces for high frequency variations.

When the number of frequency sets increase, the information leakage is reduced. As an example, the TVLA values of *RFTC(1, 1024)* is much smaller than the TVLA values of *RFTC(1, 4)* implementation. But the TVLA values of both *RFTC(1, 4)* and *RFTC(1, 1024)* are higher than the ±4.5 limit [6]. When multiple clock outputs of MMCMs are used to execute within encryptions (when $M = 2$ and $M = 3$), the TVLA leakages are reduced and bounded within ±4.5 limit. Therefore, we believe, *RFTC(3, 1024)* implementation is an effective countermeasure against power analysis attacks even when combined with the preprocessing of power traces.

Dynamic reconfiguration of FPGA clock managers is not limited to Xilinx FPGAs. Altera (Intel) FPGAs clock manager (IOPLL) can also be dynamically reconfigured [2]. Therefore, *RFTC* is not limited to Xilinx FPGAs and can be implemented on Altera FPGAs as well. We propose to test Rapid Alignment Method (RAM) [16] and Sliding-Window CPA attacks [8] against *RFTC* as a future work of this paper.

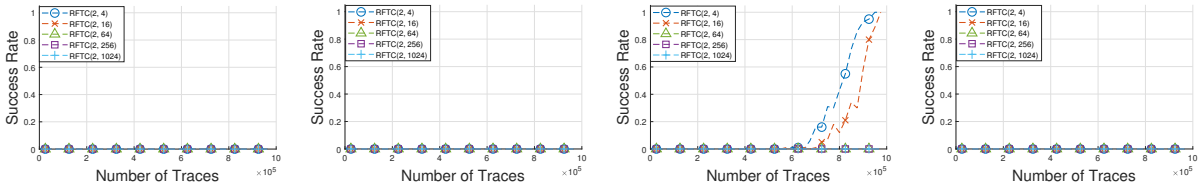
9. CONCLUSION

Power analysis attacks are an immense threat to embedded devices running cryptographic algorithms. The previous



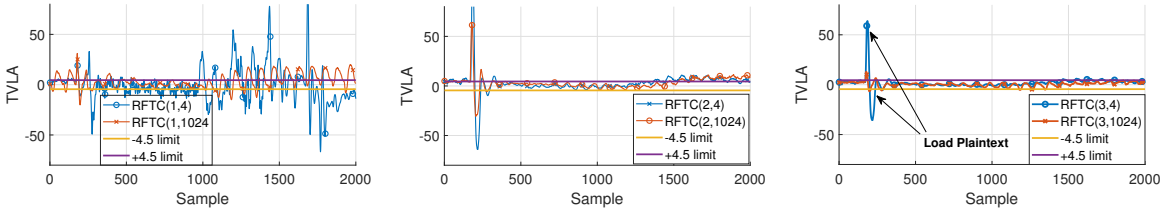
(a) CPA attacks **(b)** PCA-CPA attacks **(c)** DTW-CPA attacks **(d)** FFT-CPA attacks

Figure 4: CPA attacks, PCA-CPA attacks, DTW-CPA attacks and FFT-CPA attacks on *RFTC* $M = 1$



(a) CPA attacks **(b)** PCA-CPA attacks **(c)** DTW-CPA attacks **(d)** FFT-CPA attacks

Figure 5: CPA attacks, PCA-CPA attacks, DTW-CPA attacks and FFT-CPA attacks on *RFTC* $M = 2$



(a) *RFTC(1, 4)* and *RFTC(1, 1024)* **(b)** *RFTC(2, 4)* and *RFTC(2, 1024)* **(c)** *RFTC(3, 4)* and *RFTC(3, 1024)*

Figure 6: TVLA of *RFTC* countermeasure when $M = 1$, $M = 2$ and $M = 3$, and $P = 4$ and $P = 1,024$ for one million traces

countermeasures on randomization have limited number of random completion times, which can be vulnerable to attacks. The Random execution time countermeasure for FPGAs proposed in this paper exploits the reconfigurable MMCMs and can achieve completion times which are close to three orders magnitude higher (approximately 814× —67,584 vs. 83 completion times) when compared to the state-of-the-art random delay and random frequency countermeasures proposed in the literature. Furthermore, this paper shows the effects of increasing the number of clock frequencies with respect to the performance of power analysis attacks and information leakage.

10. REFERENCES

[1] Federal Information Processing Standards Publication 197 announcing the ADVANCED ENCRYPTION STANDARD (AES). 2001.

[2] Altera. Altera I/O Phase-Locked Loop (Altera IOPLL), 2015. https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ug/archives/ug-altera-iopll-15.0.pdf.

[3] K. H. Boey, Y. Lu, M. O'Neill, and R. Woods. Random clock against differential power analysis. In *APCCS 2010*, pages 756–759, Dec 2010.

[4] E. Brier, C. Clavier, and F. Olivier. Correlation power analysis with a leakage model. In M. Joye and J.-J. Quisquater, editors, *CHES 2004*, volume 3156, pages 16–29. 2004.

[5] M. Bucci, R. Luzzi, M. Guglielmo, and A. Trifiletti. A countermeasure against differential power analysis based on random delay insertion. In *ISCS 2005*, pages 3547–3550, 2005.

[6] J. Cooper, E. D. G. Goodwill, J. Jaffe, G. Kenworthy, and P. Rohatgi. Test Vector Leakage Assessment (TVLA) methodology in practice. ICMC 2013, 2013.

[7] J.-S. Coron and I. Kizhvatov. Analysis and improvement of the random delay countermeasure of CHES 2009. pages 95–109. Springer Berlin Heidelberg, CHES 2010.

[8] D. Fledel and A. Wool. Sliding-window correlation attacks against encryption devices with an unstable clock. *IACR*, 2018:317.

[9] A. W. Fritzke. *Obfuscating Against Side-Channel Power Analysis Using Hiding Techniques for AES*. PhD thesis, Department Of The Air Force Air University, 2012.

[10] T. Güneysu and A. Moradi. Generic side-channel countermeasures for reconfigurable devices. CHES'11, pages 33–48, Berlin, Heidelberg, 2011. Springer-Verlag.

[11] A. Hodjat, D. D. Hwang, B. Lai, K. Tiri, and I. Verbauwhede. A 3.84 gbits/s AES crypto coprocessor with modes of operation in a 0.18-um CMOS technology. GLSVLSI '05, 2005.

[12] L. B. J. Hogenboom. Principal component analysis and side-channel attacks-master thesis. In *Principal component analysis and side-channel attacks-master thesis*, pages 536–539, 2010.

[13] D. Jayasinghe, R. Ragel, J. Ambrose, A. Ignjatovic, and S. Parameswaran. Advanced modes in AES: Are they safe from power analysis based side channel attacks? In *ICCD 2014*, pages 173–180, Oct 2014.

[14] Y. Lu, M. P. O'Neill, and J. V. McCanny. FPGA implementation and analysis of random delay insertion countermeasure against dpa. In *FTP 2008*, pages 201–208, Dec.

[15] S. Mangard, E. Oswald, and T. Popp. *Power Analysis Attacks: Revealing the Secrets of Smart Cards (Advances in Information Security)*. Springer-Verlag, 2007.

[16] R. A. Muijrers, J. G. J. van Woudenberg, and L. Batina. Ram: Rapid alignment method. In *Smart Card Research and Advanced Applications*, pages 266–282. Springer, 2011.

[17] D. Oswald and C. Paar. Improving side-channel analysis with optimal linear transforms. In S. Mangard, editor, *Smart Card Research and Advanced Applications*, pages 219–233, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[18] A. A. Pammu, K. Chong, N. K. Z. Lwin, W. Ho, N. Liu, and B. Gwee. Success rate model for fully AES-128 in correlation power analysis. In *APCCAS 2016*, 2016.

[19] P. Ravi, S. Bhasin, J. Breier, and A. Chattopadhyay. PPAP and iPPAP: PLL -based protection against physical attacks. In *ISVLSI 2018, July 8-11, 2018*, pages 620–625, 2018.

[20] Y. Souissi, M. Nassar, S. Guilley, J.-L. Danger, and F. Flament. *First Principal Components Analysis: A New Side Channel Distinguisher*, pages 407–419. Springer, 2011.

[21] J. Tatsukawa. MMCM and PLL dynamic reconfiguration. https://www.xilinx.com/support/documentation/application_notes/xapp888_7Series_DynamicRecon.pdf, 2017.

[22] J. G. J. van Woudenberg, M. F. Witteman, and B. Bakker. Improving differential power analysis by elastic alignment. CT-RSA'11, pages 104–119, 2011.

[23] Xilinx. 7 series FPGAs clocking resources, 2011. https://www.xilinx.com/support/documentation/user_guides/ug472_7Series_Clocking.pdf.