Robustness Analysis of Real-Time Scheduling Against Differential Power Analysis Attacks

Ke Jiang*, Lejla Batina†, Petru Eles*, Zebo Peng*
ke.jiang@liu.se, lejla@cs.ru.nl, petru.eles@liu.se, zebo.peng@liu.se
*Department of Computer and Information Science, Linköping University, Sweden
†Institute for Computing and Information Sciences, Radboud University Nijmegen, The Netherlands

***Abstract*—Embedded systems (ESs) have been a prominent solution for enhancing system performance and reliability in recent years. ESs that are required to ensure functional correctness under timing constraints are referred to as real-time embedded systems (RTESs). With the emerging trend of utilizing RTESs in safety and reliability critical areas, security of RTESs, especially confidentiality of the communication, becomes of great importance. More recently, side-channel attacks (SCAs) posed serious threats to confidentiality protection mechanisms, namely, cryptographic algorithms. In this work, we present the first analytical framework for quantifying the influence of real-time scheduling policies on the robustness of secret keys against differential power analysis (DPA) attacks, one of the most popular type of SCAs. We validated the proposed concept on two representative scheduling algorithms, earliest deadline first scheduling (EDF) and rate-monotonic scheduling (RMS), via extensive experiments.**

***Index Terms*—Embedded systems; Real-time scheduling; Differential Power Analysis Attacks; AES; Robustness analysis**

I. Introduction

Modern embedded systems can be found in all aspects of our daily lives controlling various applications. Very often, the correctness of a job execution in such systems relies not only on the correctness of the delivered result, but also on the timeliness of the response. One important aspect to achieve real-time requirements is to deploy efficient scheduling policies, e.g., [1], [2]. In this work, we focus on two illustrative algorithms, namely earliest deadline first scheduling (EDF) and rate-monotonic scheduling (RMS) [1], [3], to demonstrate our analytical framework. Nevertheless, the conclusions drawn are general enough to be applied on any scheduler.

Current real-time embedded systems (RTESs), e.g., cyber-physical systems, require intensive communication with other peers or service centers. However, security aspects of the communication were seriously overlooked in the past, although, for those RTESs used in critical applications, the communication messages may contain sensitive information that must not be obtained by unauthorized parties. Therefore, security mechanisms should be applied to all messages exchanged. Among the large set of available cryptographic algorithms, the Advanced Encryption Standard (AES) [4] stands out because of its high applicability on embedded platforms, e.g., its sound protection strength and high throughput rate. Hence, we concentrate on the use of AES in this work. However, AES is known to be vulnerable to many threats [5], [6], aiming to retrieve secret information (the secret key or the message content). Consequently, how to design secure systems with AES and also to thwart different attacks has been extensively studied. For example, the authors of [7] presented efficient concurrent error detection architectures that can be used by AES to resist under fault-based side-channel cryptanalysis. In [8], the authors proposed hardware solutions against differential power analysis (DPA) attacks, and verified them on AES.

In addition, designing RTESs with high performance and strong security protection under cost and timing constraints is an emerging research area in recent years. The authors of [9] presented a co-design approach for finding the minimal hardware overhead under timing and security constraints. The authors of [10] proposed a scheduling policy that distributes slack times to security services based on calculated security levels. The problem of designing secure multi-mode and mixed-criticality RTESs have also been addressed in [11] and [12], respectively. However, none of these works considered the impact of side-channel attacks (SCAs) to the systems. An automated protection technique for embedded cryptographic hardware was proposed in [13] to withstand SCAs. But the potential real-time requirements to the device was not mentioned. To the best of our knowledge, there is no work describing how to analyze and, consequently, design RTESs to withstand potential SCAs.

In this work, we propose an analytical framework for measuring the robustness of AES secret keys in RTESs against DPA attacks which are a specific SCA introduced by Paul Kocher et al. [14]. DPA exploits the data dependency of the power consumption on a target device, and applies statistical analysis on a large amount of obtained power traces to retrieve secret information. Since its invention, DPA has become the most severe threat to AES implementations on embedded platform [15]. In order to counteract DPA attacks, various countermeasures are required. *Hiding in time or amplitude dimension in software or hardware* [16] are common approaches for protecting AES against DPA attacks. However, previous works require either additional hardware or modifications to off-the-shelf platforms, and may incur significant timing or energy overhead. Moreover, none of existing solutions considered aspects specific to real-time systems. To the best of our knowledge, this is the first work focused on quantifying the impact of real-time schedulers on the robustness of a given RTES design against DPA attacks on AES.

Many RTES designs utilize dynamic scheduling policies, such as EDF and RMS. A dynamic scheduler, by its nature, can be considered as one special kind of countermeasure against DPA attacks that implements the *hiding in time dimension* concept. DPA works better on aligned power traces, which means that the operations at each relative time point in the obtained samples are the same. More specifically, it is much more efficient for DPA attacks to retrieve the secret key if the useful leakage fingerprints, i.e., processing the intermediate value, in all obtained power traces (the samples) occurred at the same relative time point. Fig. 1 illustrates the idea with power traces of two AES encryptions on two messages

978-1-4799-3765-3/14 $31.00 © 2014 IEEE
DOI 10.1109/ISVLSI.2014.11

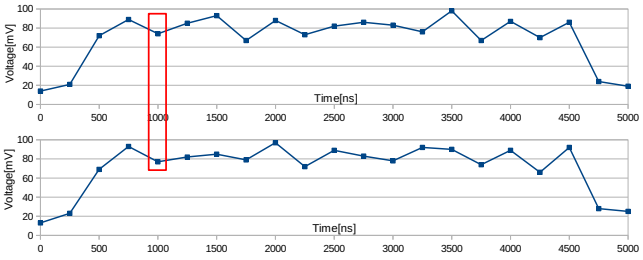

Fig. 1. Power Traces of AES encryption on Two Messages

with the same key. The power consumption, i.e., indicated by supply voltage, is uniformly sampled every $250ns$ over each AES process of $5\mu s$. Let us assume that two leakage fingerprints are required to occur at the same relative time to reveal the secret key using DPA attack. Then if the attacker obtained two samples as in Fig. 1, in which both the leakage points occurred at time $1\mu s$, she can reveal the key. If the two fingerprints happened at different times, she has to measure more samples in order to see two leakages happened at the same relative time. More details will be presented in the following sections.

II. System and Application Model

In this paper we consider mono-processor embedded systems. The system interacts with the environment by various peripherals, e.g., sensors and actuators, and communicates with other peers or service centers via a communication module (by wire or wirelessly). The computations in the system are modeled as a set of preemptive and periodic tasks $\mathcal{T}$ to be executed on the microprocessor μP. A task $\tau_i \in \mathcal{T}$ is associated with a set of attributes (e_i, T_i, $\mathcal{M}_i$). The attribute e_i is the execution time of τ_i on μP. Task τ_i is released every T_i time units, which is its period and also deadline. $\mathcal{M}_i$ represents the set of messages, via which τ_i interacts with the outside world. They are either generated from, or received and processed by τ_i. If there is no message associated with τ_i, then $\mathcal{M}_i = \emptyset$. The length (in number of AES blocks) of message $m_{ij} \in \mathcal{M}_i$ is represented by a value l_{ij}.

In order to secure the communication from/to the system, we must carry out AES encryption/decryption on critical incoming and outgoing messages. To simplify the forthcoming discussions, we assume that the encryption/decryption τ_{ij}^{AES} implied by message m_{ij} is merged into the run time of the corresponding task τ_i. That is, all corresponding AES operations on the incoming and/or outgoing messages $m_{ij} \in \mathcal{M}_i$ are considered to be part of the execution of τ_i. By this, the execution time overhead of τ_{ij}^{AES} is also included in the execution time e_i of τ_i. A simple example with four tasks is given in Table I. Tasks τ_1 and τ_2 do not have message communication, while, tasks τ_3 and τ_4 are both associated with a 128-bits long message, i.e., m_{31} and m_{41}, respectively.

III. Attacker Model

In order to make a trustworthy analysis, we make a pessimistic assumption of a strong attacker who aims to find the secret AES key (keys) used in the system. The attacker has physical access to the system, and can accurately measure the power consumption of the microprocessor. She feeds the tasks

TABLE I
An illustrative application

Task	e	T	$\mathcal{M}$
τ_1	1	5	$\emptyset$
τ_2	2	8	$\emptyset$
τ_3	4	10	$\{m_{31}\}$
τ_4	3	20	$\{m_{41}\}$

running on the microprocessor with arbitrary data, e.g., by replacing the messages from sensors. She knows the periods of all tasks[1], but does not know their actual execution times.

The attacker tries to find the secret key(s) of AES using DPA attack. The power consumption of a device depends on the operation it performs and the data it processes. DPA exploits the fact that there exists an intermediate result (leakage fingerprint) in the AES process which is a function of a given text and a few key bits, e.g., 8-bits, referred to as a subkey. Therefore, the inputs are processed by AES with the same subkey and same operation at certain fixed time. Although the input processed are different, the power consumptions at these time points have certain relations with each other. Based on this understanding, the subkeys of a secret key are attacked one by one until the whole key is obtained, or until it is trivial for mounting a brute-force attack on the rest key bits.

Thereby, the attacker first tries to feed one AES task (having secret key Key) with a set of different plaintexts $\mathcal{PT}$, and records the power consumption of μP. After that, she chooses a sample window $\mathcal{W}$, and divides the whole measured power sequence of D time units into $S = D/\mathcal{W}$ samples captured by a two dimensional matrix $P = [i-j](i = 1, ..., S; j = 1, ..., I)$ with size $S * I$. Dimension I is the number of recorded power consumption points within the corresponding sample, and is determined by the measurement granularity. Each element $P_{i,j}$ of P is a measured power result.

After that, the attacker chooses an intermediate result from the AES operation, and then calculates the hypothetical intermediate results from all the input plaintexts $\mathcal{PT}$ with all K possible subkey values, e.g., $K = 256$, if the attack is based on 8-bits of an AES secret key. She organizes all hypothetical intermediate results into a matrix $V = [i-j](i = 1, ..., S; j = 1, ..., K)$, which is later mapped to a hypothetical power consumption matrix H based on simulations or accurate power models. Until now, the attacker has gathered all information for performing a DPA attack. Then she compares (looks for correlations) between each column of the real power consumption matrix P and her hypothetical power matrix H. One common metric of correlation for this purpose is to calculate the Pearson correlation coefficient (PCC) between two columns from P and H. For example, we denote the t-th column of P as $P_{S,t}$ and the k-th column of H as $H_{S,k}$, where S includes all rows. Columns $P_{S,t}$ and $H_{S,k}$ represent the measured power consumptions at time t and hypothetical power consumptions on subkey k in all samples, respectively. Then the PCC ρ can be calculated as follows,

$$\rho(P_{S,t}, H_{S,k}) = \frac{\sum_{s=1}^{S}(H_{s,k} - \overline{H_{S,k}})(P_{s,t} - \overline{P_{S,t}})}{\sqrt{\sum_{s=1}^{S}(H_{s,k} - \overline{H_{S,k}})^2 \sum_{s=1}^{S}(P_{s,t} - \overline{P_{S,t}})^2}}, \quad (1)$$

[1] which, e.g., can be deduced from arrival rates of signals

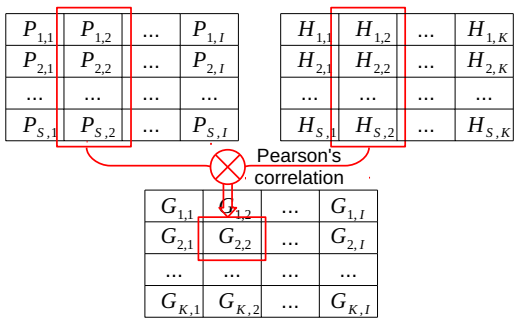

Fig. 2. The Correlation Calculation Procedure

where, $\overline{H_{S,k}}$ and $\overline{P_{S,t}}$ are the means of $H_{S,k}$ and $P_{S,t}$, respectively. The result ρ shows the linear relationship between $P_{S,t}$ and $H_{S,k}$. The higher ρ is, the stronger correlation the two columns have. After analyzed all pairs of columns, the attacker obtains a new matrix $G = [i-j](i = 1, ..., K; j = 1, ..., I)$ that collects all the correlation coefficients between all pairs of columns from P and H (as shown in Fig. 2). The highest values in G reveals the most likely used subkey by the device, since most columns are highly uncorrelated. Assuming that at time t_c in all samples, the system processes the intermediate values using subkey k_c, then ρ_{k_c,t_c} should have the dominating value in G. We refer to ρ_{k_c,t_c} as ρ_{max} which mainly determines the difficulty of attacking the device. We will elaborate more on the difficulty issues in the next section.

IV. Time Randomization Based Countermeasure

In this section, we discuss how randomization in the time dimension influences the difficulty of mounting DPA attacks on AES. As can be noticed, DPA attacks have high requirements on the samples. That is, it is important that the recorded power consumptions of the leakage fingerprints are not affected by noise and purely rely on the AES operation and the used subkey, and the samples are correctly aligned. In other words, the power consumption of leakage points at the considered time t in all samples should be caused by processing the intermediate values and not by other operations. Therefore, two ways of realizing countermeasures against DPA attacks on AES are (1) to reduce the signal-noise-ratio of executed operations, and (2) to randomize the leakage point occurrences (shuffling the occurrence time of attackable intermediate results) along the time dimension, namely, to make the the leakage points occur at different times to reduce the potential correlations [17].

The first aspect has already been studied earlier, e.g., [18], [19]. In this paper, we are the first to study the second approach under the context of real-time systems. In fact, dynamic scheduling algorithms serve this purpose by nature. Because of dynamic preemptions, the leakage fingerprints may occur at different times in different samples, which reduces the correlations between columns of matrices P and H. The impact of real-time scheduling on DPA attacks has not been considered in literature. In this section, we propose an analytical framework to quantitatively capture the impact of task scheduling on the robustness against DPA attacks.

All countermeasure techniques on DPA attacks try to obfuscate the attacker from obtaining a straight-forward correlation between her hypothetical power consumptions and the recorded samples. Now let us discuss how the time dimensional shuffling method increases the difficulty of mounting DPA attacks. In order to reduce ρ_{max}, the intermediate values should be processed at different time points in different samples. By this, the time t_c when the same subkey k_c is used is different in different samples. In the optimal case, there exists no two samples that process the intermediate values with k_c at the same relative time, i.e., $\rho_{max} \approx 0$. In this case, the attacker needs infinite amount of samples to correctly reveal the subkey k_c.

Let us denote the moment of time when the leakage fingerprints occur with the highest probability in the samples as $\hat{t}$. It is clear that, the power consumptions $P_{S,\hat{t}}$ have the highest correlation with power hypotheses H_{S,k_c} from the real subkey k_c among all $t \in \{0, I\}$ (we remind that I is the measurement granularity, see section III). We denote as $\hat{p}$ the probability of the leakage point occurring at time $\hat{t}$. The power consumption of the device at $\hat{t}$ is denoted as $\hat{P}$. So, the probability that $\hat{P}$ is caused by processing the intermediate results (that gives leakage fingerprints) is $\hat{p}$. Similarly, the probability that $\hat{P}$ is caused by other tasks is $(1-\hat{p})$. Therefore, in the presence of time shuffling, $\hat{\rho}_{max}$ can be calculated from the definition of PCC, and reduced to:

$$\hat{\rho}_{max} = \rho(H_{S,k_c}, \check{P}_{S,\hat{t}}) * \hat{p} * \sqrt{\frac{Var(\check{P}_{S,\hat{t}})}{Var(P_{S,\hat{t}})}}, \tag{2}$$

where, $P_{S,\hat{t}}$ and $\check{P}_{S,\hat{t}}$ are all power consumption points and the points related to leakage fingerprints at time $\hat{t}$ in the samples, respectively. Coefficient $\rho(H_{S,k_c}, \check{P}_{S,\hat{t}})$ solely depends on the accuracy of the attacker's simulation or power model about the targeted device, and is set to the most conservative value 1. The variances $Var(P_{S,\hat{t}})$ and $Var(\check{P}_{S,\hat{t}})$ are determined by the device characteristics. We assume $Var(P_{S,\hat{t}}) = Var(\check{P}_{S,\hat{t}})$ to make our analysis independent of devices. Thus, we have

$$\rho_{max} = \hat{p}. \tag{3}$$

We define the system robustness against DPA attacks as the difficulty in terms of time overhead for the attacker to gather sufficient amount of information to observe a high correlation between the hypothetical power consumptions and the measured samples. We first calculate the lower bound of number of samples $\mathcal{N}$ for noticing a significant peak of the correlation coefficient ρ_{max} in G. In fact, the sampling distribution of the PCC can be transformed to a normal distribution using Fisher transformation. Then we can calculate the attacker's confidence interval of her hypothetical attacks on the obtained power traces. As shown in [8], this calculation can be transformed into the following equation, allowing the designer to calculate the lower bound $\mathcal{N}$ of the required number of samples, i.e.,

$$\mathcal{N} = 3 + 8\left(\frac{z_\alpha}{ln(\frac{1+\rho_{max}}{1-\rho_{max}})}\right)^2, \tag{4}$$

where z_α is the quantile of standard normal distribution that determines the distance between the distribution of $\rho = 0$ and $\rho = \rho_{max}$. The value α in z_α is often called the error probability, and reflects how likely the attack can observe a significant peak in G, i.e., the higher α, the more likely. In this

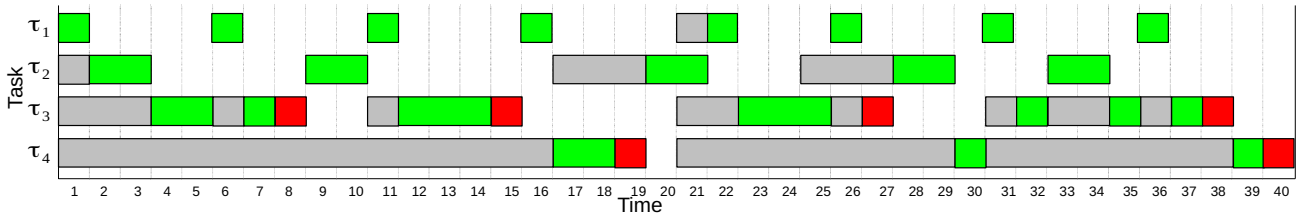

Fig. 3. System Schedule under Earliest Deadline First Scheduling (EDF)

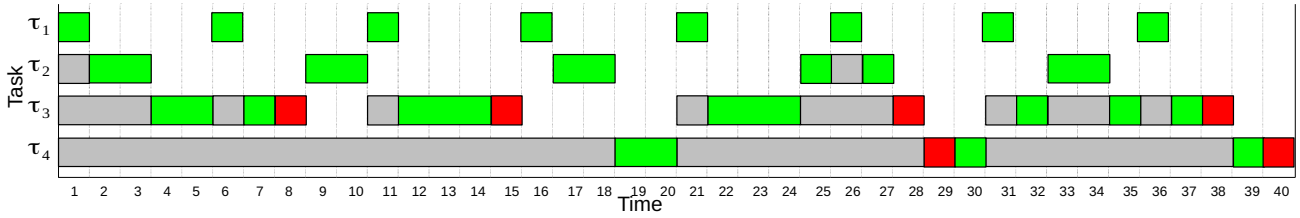

Fig. 4. System Schedule under Rate-Monotonic Scheduling (RMS)

paper, we set $\alpha = 0.9$ (giving $z_\alpha = 1.282$) in our analysis.

The result of $\mathcal{N}$ is the determinant value for measuring how good a time shuffling countermeasure is. Combining Eq. 3 and 4, we can rewrite the calculation of $\mathcal{N}$ as follows,

$$\mathcal{N} = 3 + \frac{13.148}{ln^2(\frac{1+\hat{p}}{1-\hat{p}})}. \quad (5)$$

As can be noticed, $\mathcal{N}$ is strictly decreasing with $\hat{p}$. In addition to $\mathcal{N}$, the time that the system takes to generate one sample also influences the difficulty of mounting DPA attacks. As the measured power are grouped into samples based on $\mathcal{W}$, it therefore takes $\mathcal{W}$ time units to generate a sample. Now, we can define our quantification of the system's robustness $\mathcal{R}$ against DPA attacks:

$$\mathcal{R} = \mathcal{N} * \mathcal{W}. \quad (6)$$

The result $\mathcal{R}$ is the value that captures the robustness of a particular real-time scheduling solution against DPA attacks.

V. An Illustrative Example

In this section, we evaluate two representative scheduling policies, i.e., earliest deadline first scheduling (EDF) and rate-monotonic scheduling (RMS). As discussed in the previous section, the time point t_c at which AES leakage fingerprints (processing the intermediate value) happen with the highest probability $\hat{p}$ is the decisive parameters for the robustness of the system. One way to obtain the approximation of $\hat{p}$ is to simulate the actual system execution and then analyze the simulated schedule using statistical methods. After that, the robustness of the system can be quantified using Eq. 6.

Let us consider the application with four tasks from Table I. Tasks τ_1 and τ_2 do not have communication demands. Tasks τ_3 and τ_4 are application tasks that interact with the service center wirelessly via 128-bits long messages m_{31} and m_{41}, respectively. The two messages are encrypted by AES operations τ_{31}^{AES} and τ_{41}^{AES} before being sent over the communication interface. The AES operations are included in the execution of tasks τ_3 and τ_4. For illustration purposes, we assume that the attacker measures the power consumption of the processor at each time unit. The leakage fingerprints of τ_{31}^{AES} and τ_{41}^{AES} are assumed to be at the last time units of τ_3 and τ_4, respectively.

The two system schedules obtained from simulations of EDF and RMS over one hyperperiod $\mathcal{HP}$ of $\mathcal{T}$ are depicted in Fig. 3 and 4, respectively. The hyperperiod $\mathcal{HP}$ of a task set is defined as the least common multiplier of all task periods (in this case, 40). The green rectangles indicate the executions of the normal tasks and non-leakage part of AES. The red rectangles represent the leakage points of τ_{31}^{AES} and τ_{41}^{AES}, and generate attackable power consumptions (refer to the highlighted points in Fig. 1). The gray rectangles indicate that the task executions are blocked by higher prioritized tasks. Depending on whether the designer decides to use the same secret key or not in τ_{31}^{AES} and τ_{41}^{AES}, two different analyses need to be conducted.

Case of $Key(\tau_{31}^{AES}) \neq Key(\tau_{41}^{AES})$: In this case, the attacker may target one or both of the keys. As shown in Eq. 6, the robustness of a key depends on two aspects, the lower bound $\mathcal{N}_i$ of the number of required samples to observe high correlations and the time for obtaining a sample, i.e., the sample window $\mathcal{W}$. Now, let us first study the robustness of the two keys $\mathcal{R}(Key(\tau_{31}^{AES}))$ and $\mathcal{R}(Key(\tau_{41}^{AES}))$ separately. We assume that the attacker groups her obtained power traces into samples based on the period of message m_{ij}, i.e., $\mathcal{W} = T_i$.

Let us first analyze the simulated EDF schedule as shown in Fig. 3. As we assumed that the attacker measures power consumption of the device at each time unit, she gets 40 discrete power values over one hyperperiod $\mathcal{HP}$. Each power value corresponds to a task operation on μP that can be retrieved from Fig. 3. Assume that $Key(\tau_{31}^{AES})$ for message m_{31} is the current target of the attacker. If the attacker defines $\mathcal{W} = T_3 = 10$, then she can get 4 samples within $\mathcal{HP}$, and further align the samples, as shown in Fig. 5. Since all operations from other tasks including the leakage points of τ_{41}^{AES} (because $Key(\tau_{31}^{AES}) \neq Key(\tau_{41}^{AES})$) are independent with the leakage fingerprints of $Key(\tau_{31}^{AES})$, we depict these operations in gray. We can observe from Fig. 5 that, in the first and forth sample, leakages happened both at the 8-th time unit, while in the second and third, leakages happened at the 5-th and 7-th time unit, respectively. Therefore, the highest probability $\hat{p}$ that the leakage fingerprints occur at the same relative

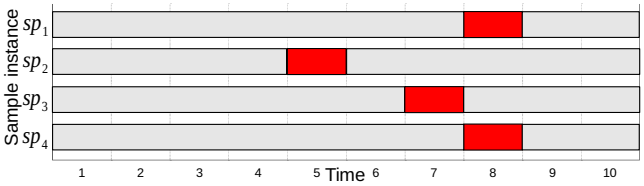

Fig. 5. Aligned samples from EDF for $Key(\tau_{31}^{AES})$

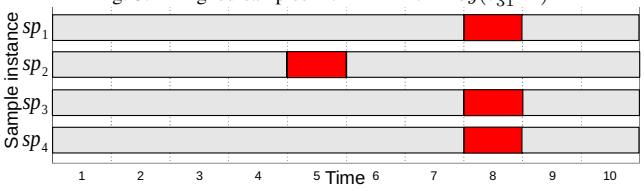

Fig. 6. Aligned samples from RMS for $Key(\tau_{31}^{AES})$

time point is $\hat{p} = \frac{2}{4} = 0.5$ at $\hat{t} = 8$. Now we can quantify the robustness of the $Key(\tau_{31}^{AES})$ against DPA attacks using Eq. 6, and get $\mathcal{R}^{EDF}(Key(\tau_{31}^{AES})) = 140$. Similarly, we get the robustness of $Key(\tau_{41}^{AES})$ as $\mathcal{R}^{EDF}(Key(\tau_{41}^{AES})) = 280$ when she chooses $\mathcal{W} = T_4 = 20$.

Now let us study the influence of RMS on key robustness. As depicted in Fig. 4, the attacker also gets four samples for τ_{31}^{AES} and two samples for τ_{41}^{AES} when she organizes the samples based on T_3 and T_4, respectively. For $Key(\tau_{31}^{AES})$, we can notice from the aligned samples shown in Fig. 6 that three leakage fingerprints happened at the 8-th time units, thereby, probability $\hat{p} = 0.75$ at $\hat{t} = 8$. Consequently, we can calculate $\mathcal{R}^{RMS}(Key(\tau_{31}^{AES})) = 70$. If we compare the robustnesses of the secret key $Key(\tau_{31}^{AES})$ under EDF and RMS, we can find that $\mathcal{R}^{EDF}(Key(\tau_{31}^{AES})) > \mathcal{R}^{RMS}(Key(\tau_{31}^{AES}))$, which means that, for this example, it is more difficult for the attacker to mount a successful DPA attack, if the system is scheduled by EDF policy. In other words, the key $Key(\tau_{31}^{AES})$ is better protected when EDF is applied for scheduling the application. In the same way, we can calculate $\mathcal{R}^{RMS}(Key(\tau_{41}^{AES})) = 280$. So the key $Key(\tau_{41}^{AES})$ is equally protected by EDF and RMS. However, the first instance of τ_4 missed its deadline under RMS, which is not the case under EDF.

The case of $Key(\tau_{31}^{AES}) = Key(\tau_{41}^{AES})$: If the secret keys used by τ_{31}^{AES} and τ_{41}^{AES} are the same, the leakages fingerprints from the two AES encryptions are identical. Therefore, the weakest key $Key(\tau_{31}^{AES})$ from the previous analysis determines the current system-wise robustness. In addition, the system may be even more vulnerable under DPA attacks since probability $\hat{p}$ can become higher under the current situation because some of the leakage points due to τ_{41}^{AES} might be aligned with those from τ_{31}^{AES}. Thus, we need to do a new analysis on the simulated schedule using $\mathcal{W} = 10$. The system robustness are then calculated, i.e., $\mathcal{R}^{EDF} = 140$ and $\mathcal{R}^{RMS} = 70$.

VI. Experimental Evaluation

We have conducted experimental analyses on a Linux machine with a quad-core Intel Xeon 2.66GHz CPU. The test applications were generated with random task parameters under the requirements of different evaluation scenarios. Each task is associated with at most one message having length (in AES blocks) randomly selected from the set $\{0, 1, 2, 3, 4\}$ with

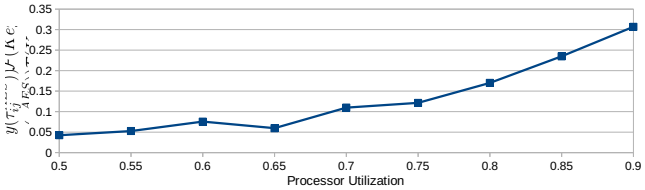

Fig. 7. Results of 5 Tasks under Different Processor Utilizations

probability $\{50\%, 12.5\%, 12.5\%, 12.5\%, 12.5\%\}$. If $l_{ij} = 0$, then τ_i does not have any communication demand, i.e., $\mathcal{M}_i = \emptyset$. The number of AES leakage fingerprints depends on the number of blocks of the message. We carried out experiments on EDF and RMS under three different scenarios to study the influence of the scheduler against DPA attacks from different perspectives. Since the case of all AES operations having the same secret key is a special case of different keys, we only present the results of the second case due to space limit.

a) Evaluation under different processor utilizations: Now let us first have a look at how processor utilization influences the robustness of the system under EDF and RMS. In order to judge which scheduler outperformed the other in an experiment, we have the following definition that focuses on which scheduler gives higher robustness:

$$f(\mathcal{R}^{EDF}(k), \mathcal{R}^{RMS}(k)) = \begin{cases} 1 & , \text{if } \mathcal{R}^{EDF}(k) > \mathcal{R}^{RMS}(k) \\ 0 & , \text{if } \mathcal{R}^{EDF}(k) = \mathcal{R}^{RMS}(k) \\ -1 & , \text{if } \mathcal{R}^{EDF}(k) < \mathcal{R}^{RMS}(k). \end{cases} \quad (7)$$

Then, the overall performance indicator $\mathcal{F}$ for a set of experiments, i.e., the robustness of the secret key $Key(\tau_{ij}^{AES})$ from all experiments under the same evaluation criteria, is defined as follows,

$$\mathcal{F}(Key(\tau_{ij}^{AES})) = \frac{1}{n}\sum_{e=1}^{n} f(\mathcal{R}_e^{EDF}(Key(\tau_{ij}^{AES})), \mathcal{R}_e^{RMS}(Key(\tau_{ij}^{AES}))), \quad (8)$$

where, n is the total number of valid experiments where $\mathcal{M}_i \neq \emptyset$. The result $\mathcal{F}(Key(\tau_{ij}^{AES}))$ captures the average overall superiority of one scheduler over the other. EDF delivers higher robustness for $Key(\tau_{ij}^{AES})$ in more cases, if $\mathcal{F} > 0$. Similarly, RMS outperforms EDF on average, if $\mathcal{F} < 0$. EDF delivers better results in all the experiments if $\mathcal{F} = 1$. While, $\mathcal{F} = -1$ would indicate that RMS gives higher robustnesses in all cases.

The results shown in Fig. 7 were obtained under 9 different utilization levels U from 0.5 to 0.9. On each utilization level, 200 test applications with 5 tasks were randomly generated. The results demonstrate that, when the processor utilization was low, both policies provided similar protections on average, e.g., $\mathcal{F} = 0.043$ at utilization $U = 0.5$. This is because when the utilization is low, task executions are very sparsely spread over time. Therefore, the executions have little preemptions and, thus, occur more regularly at fixed time points. As the utilization became higher, EDF delivered better and better robustness on average than RMS, e.g., $\mathcal{F} = 0.307$ at utilization $U = 0.9$. This can be explained by the fact that task priorities are dynamically changing based on their timelinesses in EDF, that introduces more randomness into the occurrences of tasks

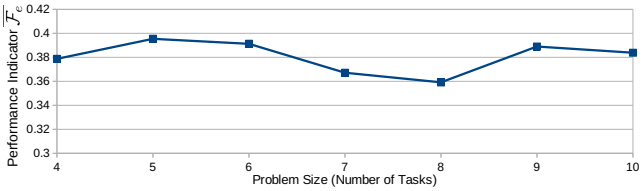

Fig. 8. Results of Different Problem Sizes

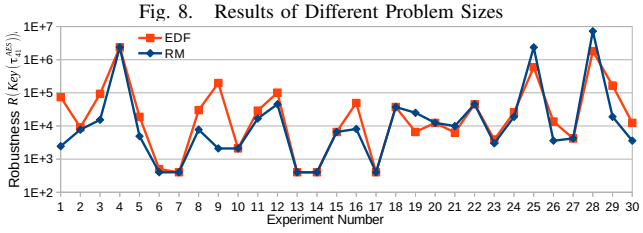

Fig. 9. Robustness $R(Key(\tau_{41}^{AES}))$ of 30 Exp. under EDF and RMS

executions.

b) Evaluation on different problem sizes: We also conducted experiments on seven different problem sizes having $|\mathcal{T}| = 4, 5, ..., 10$ tasks. On each problem size, we randomly generated 200 test applications with utilization $0.7 \leq U \leq 1$, and used the average performance indicator $\overline{\mathcal{F}_e} = \frac{1}{|\mathcal{T}|}\sum_{i=1}^{|\mathcal{T}|}\mathcal{F}(Key(\tau_{ij}^{AES}))$ of all messages as the final indication for each problem size. The results are presented in Fig. 8. From the results, we can notice that on average EDF outperformed RMS on all problem sizes under the utilization bound $0.7 \leq U \leq 1$.

c) Evaluation on the same problem size: Fig. 9 presents the actual robustness values of secret key $Key(\tau_{41}^{AES})$ from 30 randomly selected valid experiments from the previous experiments, with number of tasks $|\mathcal{T}| = 5$. As can be noticed, EDF provided better protection for $Key(\tau_{41}^{AES})$ in 15 experiments, while RMS outperformed EDF in 4 tests (Exp. 19, 21, 25, and 28). In the rest of the cases, EDF and RMS gave the same robustness values. The experiments showed that on average, i.e., in the majority of cases, EDF produced superior robustness. While, there exists individual cases in which RMS was better. Also worth-mentioning, EDF delivered 4.83 times higher robustness on average in the experiments, and satisfied all deadline requirements. While, task τ_4 suffered from deadline misses in 2 cases when the system was scheduled with RMS. However, we cannot conclude that EDF was always preferable, since, in several cases, RMS delivered excellent results while also satisfying deadline constraints for all tasks in several experiments, e.g., Exp. 19 and 28.

VII. Conclusion

Security has become an emerging topic for RTES designs in which confidentiality of sensitive communication is often of central importance. More recently, cryptographic algorithms, e.g., AES, have been deployed in RTESs for protecting sensitive information. However, AES implementations in embedded platforms are known to be vulnerable towards side-channel attacks, e.g., DPA attacks. Therefore, the problem of how to design secure RTESs with robust AES must be carefully studied. In this paper, we make the very first attempt to proposing an analytical framework for quantifying the impact of the scheduling policy on the robustness of secret keys against DPA attacks. Extensive experiments were conducted, and demonstrated that different scheduling policies do have different impacts on robustness. Therefore, the designer must carefully choose the most suitable scheduler for the given application considering both deadlines and robustnesses. This paper opens up a new research direction, and urges the needs for proposing new scheduling policies that not only consider timing aspects, but also reinforce the AES resistance against SCAs. Finally, it is worth mentioning that, while in this paper we considered DPA attacks, similar analyses can be applied for other side-channel attacks based on the observations of parameters as temperature and sound.

References

[1] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *Journal of the ACM (JACM)*, vol. 20, no. 1, pp. 46–61, 1973.

[2] J. Lehoczky, L. Sha, and Y. Ding, "The rate monotonic scheduling algorithm: exact characterization and average case behavior," in *Real-Time Systems Symposium (RTSS)*, Dec 1989, pp. 166–171.

[3] G. C. Buttazzo, *Hard real-time computing systems: predictable scheduling algorithms and applications*. Springer, 2011, vol. 24.

[4] J. Daemen and V. Rijmen, *The design of Rijndael: AES–the advanced encryption standard*. Springer, 2002.

[5] E. Biham and N. Keller, "Cryptanalysis of reduced variants of Rijndael," in *3rd AES Conference, New York, USA*, 2000.

[6] Y. Li, K. Sakiyama, L. Batina, D. Nakatsu, and K. Ohta, "Power Variance Analysis breaks a masked ASIC implementation of AES," in *Design, Automation and Test in Europe Conference (DATE)*, 2010, pp. 1059–1064.

[7] R. Karri, K. Wu, P. Mishra, and Y. Kim, "Concurrent error detection schemes for fault-based side-channel cryptanalysis of symmetric block ciphers," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 21, no. 12, pp. 1509–1517, 2002.

[8] S. Mangard, "Hardware countermeasures against DPA–a statistical analysis of their effectiveness," in *Topics in Cryptology–CT-RSA*, 2004, pp. 222–235.

[9] K. Jiang, P. Eles, and Z. Peng, "Co-design techniques for distributed real-time embedded systems with communication security constraints," in *Design, Automation and Test in Europe Conference (DATE)*, 2012, pp. 947–952.

[10] T. Xie and X. Qin, "Improving security for periodic tasks in embedded systems through scheduling," *ACM Transactions on Embedded Computing*, vol. 6, no. 3, p. 20, 2007.

[11] K. Jiang, P. Eles, and Z. Peng, "Optimization of secure embedded systems with dynamic task sets," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2013*. IEEE, 2013, pp. 1765–1770.

[12] X. Zhang, J. Zhan, W. Jiang, Y. Ma, and K. Jiang, "Design optimization of security-sensitive mixed-criticality real-time embedded systems," in *1st workshop on Real-Time Mixed Criticality Systems (ReTiMiCS)*, 2013.

[13] A. G. Bayrak, N. Velickovic, F. Regazzoni, D. Novo, P. Brisk, and P. Ienne, "An eda-friendly protection scheme against side-channel attacks," in *Design, Automation Test in Europe Conference Exhibition (DATE)*, 2013, pp. 410–415.

[14] P. C. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis," in *CRYPTO*, 1999, pp. 388–397.

[15] S. B. Ors, F. Gurkaynak, E. Oswald, and B. Preneel, "Power-Analysis Attack on an ASIC AES implementation," in *International Conference on Information Technology: Coding and Computing (ITCC)*, vol. 2, 2004, pp. 546–552.

[16] S. Mangard, E. Oswald, and T. Popp, *Power analysis attacks: Revealing the secrets of smart cards*. Springer, 2007.

[17] J.-S. Coron and I. Kizhvatov, "An efficient method for random delay generation in embedded software," in *Cryptographic Hardware and Embedded Systems (CHES)*. Springer, 2009, pp. 156–170.

[18] K. Tiri, M. Akmal, and I. Verbauwhede, "A dynamic and differential CMOS logic with signal independent power consumption to withstand differential power analysis on smart cards," in *the 28th European Solid-State Circuits Conference (ESSCIRC)*, 2002, pp. 403–406.

[19] D. Mesquita, J.-D. Techer, L. Torres, G. Sassatelli, G. Cambon, M. Robert, and F. Moraes, "Current mask generation: a transistor level security against DPA attacks," in *18th Symposium on Integrated Circuits and Systems Design*, 2005, pp. 115–120.