A New Power Analysis Attack and a Countermeasure in Embedded Systems

Fangming Chai and Kyoung-Don Kang
Department of Computer Science
State University of New York at Binghamton
{fchai1, kang}@binghamton.edu

***Abstract*—Recent works on embedded system security, which is becoming increasingly important, claim that dynamic voltage and frequency scaling (DVFS) supports a natural defense against power analysis attacks. In this paper, however, we design a new DVFS-aware attack that 1) identifies the voltage and frequency values used for DVFS and 2) performs power analysis to extract cryptographic keys. Further, we propose a simple yet effective defense against DVFS-aware power analysis attacks: we generate noise against power analysis attacks by running random cryptographic instructions in slack time (if any) generated when a real-time task (e.g., an engine control task) finishes earlier than its worst-case execution time. To analyze the effectiveness of the new proposed attack and defense technique, we undertake a simulation study using a cycle-accurate micro-architectural simulator and an advanced power model. In the simulation study, our DVFS-aware power analysis attack increases the accuracy of secret key extraction by 1-22% compared to most existing power analysis attacks unaware of DVFS. Moreover, our defense policy decreases the success rate of the DVFS-aware power analysis attack by 2-22% compared to state-of-the-art approaches that use DVFS as a countermeasure against power analysis attacks.**

***Index Terms*—Embedded Systems; Security; DVFS; Power Analysis Attack; Countermeasure;**

I. Introduction

The importance of security in embedded systems is increasing fast due to the dramatic growth of the Internet of Things (IoT) and cyber-physical systems (CPS) [1], [2], [3], [4]. Traditionally isolated embedded systems, e.g., heating, ventilation, and air conditioning (HVAC) systems in buildings, home automation systems, and electronic control units (ECUs) in automobiles, are increasingly connected to the Internet. In this paper, we consider power analysis attack—one of the most powerful side channel attacks—against embedded systems. It analyzes power consumption patterns to infer cryptographic keys that lay a foundation for cyber security, such as encryption, authentication, and digital signature [5], [6], [7], [8].

Earlier work on differential power analysis [5], [6], [7], [8] applied statistical techniques to steal keys from relatively simple embedded devices, e.g., smart cards [9], [10], [11], [12], based on the observation that executing a cryptographic algorithm, e.g., an encryption algorithm, using different keys consumes different amounts of power. Recently, advanced machine learning techniques are applied for more effective power analysis attacks [13], [14], [15]. Dynamic voltage and frequency scaling (DVFS) [16] is provided in most modern processors to reduce the power/energy consumption by dynamically adjusting the voltage and frequency at runtime. Recent works [17], [18] claim that DVFS is an effective defense against power-analysis attacks, since the frequency and voltage may change significantly due to DVFS while the processor executes thousands of instructions to encrypt a single byte using a standard encryption algorithm, such as Advanced Encryption Standard (AES). In this paper, we show that the DVFS-based defense mechanisms [17], [18] provide a false sense of security by devising a new power-analysis attack, called **C-DVFS** (Circumventing-DVFS), which generally works as follows:

1) C-DVFS first identifies the voltage and frequency values used for DVFS by applying fundamental digital signal processing and machine learning techniques;
2) it normalizes the power consumption data of the identified voltage and frequency values with respect to the common voltage and frequency (e.g., the maximum voltage and frequency provided by a specific processor); and
3) it applies power analysis attacks based on machine learning [13], [14], [15] to the power consumption data denoised and normalized in the previous steps.

In general, devising a new attack can advance security research by shedding light on overlooked vulnerabilities and potential threats that could be devastating if exploited by attackers. Despite the importance, relatively little work has been done to analyze whether DVFS widely used in embedded systems for power/energy efficiency can serve as a power-analysis countermeasure except for [17], [18] that claim DVFS provides a strong defense. In this paper, however, we show that C-DVFS can considerably decrease the effectiveness of DVFS as a defense against power analysis attacks.

In addition, we propose a new approach to mitigate the potential impact of C-DVFS attacks. To meet stringent timing constraints, real-time tasks are usually scheduled based on their worst-case execution times (WCETs) known *a priori*; however, they often complete before the WCETs, producing slack time [19]. The key idea of our defense against C-DVFS is to extend the earliest deadline first (EDF) scheduling algorithm [19] by executing random instructions similar to the instructions frequently used in cryptographic algorithms during the slack time (if any) to insert noise to power con-

sumption data, while missing no deadlines of the real-time tasks, e.g., control tasks in ECUs, implanted medical devices, or factory automation. Thus, our approach, called security-enhanced EDF (**S-EDF**), can mitigate the threat of C-DVFS without adversely affecting the timeliness important in real-time embedded systems. Also, it is relatively easy to deploy, since it requires no extra circuitry to hide power consumption patterns. Although we do not claim our approach is fully secure or reliable, it is a stepping stone to more advanced research to address C-DVFS that is an important yet under-explored problem.

To evaluate the effectiveness of the C-DVFS attack and the mitigation technique, we conduct a simulation study using a cycle-accurate micro-architectural simulator, gem5 [20], and an advanced power model, called McPAT [21], to realistically model the power consumption and power analysis attack. In the simulation study, C-DVFS increases the accuracy of distilling secret keys for AES by 1-22% compared to well-known power analysis attacks, e.g., [13], [14], [15], which are based on machine learning but unaware of DVFS. Thus, we observe that the proposed C-DVFS attack is a serious threat. Moreover, our defense policy, S-EDF, decreases the success rate of C-DVFS by 2-22% compared to the state-of-the-art approaches [17], [18] that use DVFS as a countermeasure.

The rest of the paper is organized as follows. Related work is discussed in Section II. Section III describes the real-time task model and threat model considered in this paper. In Section IV, the C-DVFS attack is described. Our approach to mitigate the attack (S-EDF) is discussed in Section V. In Section VI, the proposed attack and mitigation schemes are evaluated in comparison to the approaches representing the state of the art. Finally, Section VII concludes the paper and discusses future work.

II. Related Work

Simple power analysis (SPA) and differential power analysis (DPA) were first introduced in [5], [8]. SPA analyzes power signals directly, while DPA considers statistical correlations between the key and power signals. Template-based DPA [7], [12] takes a more advanced step by storing the probabilistic distribution as templates and matches power signals to the stored templates. These attack methods are applied to distill secrete key used by cryptographic algorithms such as DES [22], AES [23], and RSA [24]. For example, AES and DES implemented in smart cards or ASIC are attacked [9], [10], [11], [12], [25].

More recently, machine learning techniques have been applied to power analysis attacks [13], [14], [26]. In these approaches, feature selection is performed to select relevant features out of massive power consumption data to reduce the computational complexity and the impact of irrelevant features on the model. The power consumption data is divided into the training set and test set to train and evaluate different models to extract secrete keys via power analysis attacks. Popular feature selection algorithms in machine learning, e.g., the Pearson correlation coefficient and principal component analysis (PCA) [15], [27], were found to be effective for feature selection in power analysis attacks. In [14], the effectiveness of well-established supervised learning models, e.g. decision tree and random forest [28], [29], self-organizing map [30], and support vector machine [31], [32], for power analysis attacks is evaluated. Besides, [26] proves that power analysis via machine learning can successfully attack masked AES. However, most existing approaches to power analysis do not consider potential noise produced by DVFS.

The principle of countermeasures against power analysis attacks is to break the correlation between the processed data (keys) and power consumption [33]. Masking and hiding are two main approaches. Masking dynamically changes the power consumption characteristics of devices, while hiding randomizes the intermediate processing. Countermeasures could be implemented in different layers in software [17], [18], [33] or hardware [11], [25]. In [17], [18], DVFS is shown to be effective against most existing power analysis attacks that are unaware of the potential noise injected into power consumption data by DVFS. In this paper, however, we show that DVFS may give a false sense of security by devising a new DVFS-aware power analysis attack, C-DVFS, and propose S-EDF—a countermeasure in embedded systems.

III. Real-Time Task Model and Threat Model

In this paper, we consider that there is a task set Γ, which consists of N sporadic tasks, in the real-time embedded system. A task $\tau_i \in \Gamma$ is associated with the worst-case execution time (WCET), C_i, and period, T_i, that defines the minimum inter-arrival time between two consecutive instances (jobs) of a task. In real-time embedded systems, it is assumed that C_i and T_i are known *a priori* to meet all deadlines [19]. The utilization of τ_i is $U_i = C_i/T_i$. For the sake of simplicity, we assume that τ_i's deadline, D_i, is equal to T_i (implicit deadline). We assume the tasks are scheduled via the earliest deadline first (EDF) algorithm, which is an optimal real-time scheduling algorithm. (Our approach is not limited to a specific real-time scheduling algorithm. For example, the rate monotonic algorithm [19] can be used instead.) In EDF, all deadlines will be met if the total utilization $U = \sum_{i=1}^{N} U_i \leq 1$ [19]. In addition, we assume that DVFS is supported to save power/energy, while meeting all deadlines. The real-time task model above and DVFS in real-time embedded systems are well studied and widely accepted [19], [34].

We assume that a subset of the tasks $\Gamma_s \subseteq \Gamma$ are required to execute a cryptographic algorithm, for example, to encrypt or digitally sign certain data. In this paper, we consider AES, which is one of the most widely used cryptographic algorithms. In our threat model, we assume the following:

- When a sporadic task $\tau_i \in \Gamma_s$ is released, it performs a specified task and executes AES. We assume that the time for τ_i to process the AES algorithm is included in its WCET, C_i.
- A correct implementation of a strong encryption algorithm is not necessarily secure due to potential information leakage through side channels, e.g., power consump-

tion in this paper, not considered in the cryptographic algorithm design.

- Attackers can apply powerful machine learning techniques publicly available as libraries/tools, e.g., R [35] or Tensorflow [36], to process power traces and extract secret keys.
- An attacker can acquire (e.g., purchase) target embedded devices and create a set of random keys and plaintext data to design and train power analysis models offline. (Although the knowledge of plaintext is not required, knowing both plaintext and ciphertext usually expedites the cryptanalysis [37].)
- Alternatively, side-channel power analysis attacks can be performed without physical access to target embedded systems. For example, embedded operating systems based on Linux provide `procfs` (the `/proc` file system) that stores a broad range of system information, e.g., the CPU utilization, memory usage, cache latency, and power consumption [2]. Most processes in the system can freely access them to enhance the performance and energy efficiency. Thus, an attacker can penetrate IoT networks, which often lack strong security [1], to download power traces stored in the background without being detected.
- An attacker's computational system, a desktop machine or server, is more powerful than embedded devices are.

Considering the current practice in IoT and the general availability of computational resources and machine learning tools, our threat model is very realistic.

IV. C-DVFS: DVFS-Aware Power Analysis Attack

In [17], [18], DVFS was shown to mitigate power analysis attacks, since frequency or voltage variations directly change the power consumption and, therefore, power analysis attacks based on pre-built models or statistical templates may fail to extract keys. In practice, many processors have many DVFS profiles enabled, further increasing the challenge. In this section, we describe the C-DVFS attack that can filter out noise in power traces produced by DVFS for power analysis attacks.

In general, the power signal of a DVFS-enabled system in the i^{th} measurement period, $P(i)$, is a tuple that consists of the time, frequency, voltage, and power consumption when the i^{th} measurement is taken:

$$P(i) \triangleq (t, f_i, v_i, p_i) \tag{1}$$

Conceptually, building a power trace database that consists of the collected power signal tuples is the first step in any power analysis attacks. Using the power trace database, the C-DVFS attack proceeds in three stages illustrated in Figure 1 and summarized below.

- **Stage 1:** Divide the power data trace into a training set and test set. Train the frequency model, M_f, and voltage model, M_v, that will be used to identify the frequency and voltage values used by DVFS (using the training set). In addition, normalize the power consumption data to the power consumption of a single common frequency and voltage setting, e.g., f_{max} and v_{max} (the maximum frequency and voltage of the target processor), used for differential power analysis in C-DVFS.
- **Stage 2:** Train the key prediction model, M_p, that distills keys based on the transformed power trace acquired in Stage 1. Further, test the trained models, M_f, M_v, and M_p using the test set.
- **Stage 3:** Perform a power analysis attack to identify the voltage and frequency used by DVFS, normalize power consumption data, and extract keys in the victim system using M_f, M_v, and M_p trained and tested in the previous stages.

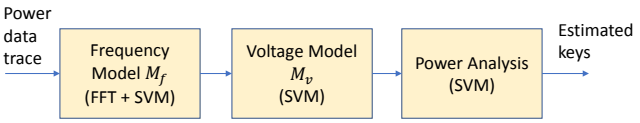

Fig. 1. C-DVFS Flow

When DVFS is used for power/energy efficiency, power data cannot be used to directly train the machine learning models and estimate keys, since there are many combinations of voltage and frequency that consume different amounts of power. Also, the voltage and frequency may change anytime, posing more challenges for power analysis attacks. For example, if a processor has 25 possible frequency and voltage profiles (5x5 frequency and voltage settings) and runs 1,000 instructions to encrypt data, a brute-force approach should consider 1000^{25} combinations. Thus, finding a secrete key in the huge search space is infeasible.

To address the challenge, in Stage 1, C-DVFS trains the frequency model, M_f, to identify the frequencies, f_i at time t in (1), used by DVFS. Next, C-DVFS trains the voltage model M_v to identify the voltage, v_i at time t in (1), used by DVFS. In Stage 2, the key prediction model, M_p, is trained using the power consumption data, p_i in (1), normalized to f_{max} and v_{max}. Finally, in Stage 3, C-DVFS distills keys using the trained models, M_f, M_v, and M_p. In the following subsections, our approaches to training M_f, M_v, and M_p as well as C-DVFS attack using the trained models are discussed.

A. Training the Frequency Model

The dynamic power consumption of a processor at time t is proportional to the frequency and the square of the voltage [16]:

$$P(t) = CV^2 f \tag{2}$$

where C is the dynamic power dissipation capacitance. The voltage and frequency can be configured independently as specified by the manufacturer.

To extract frequency features, we convert the power consumption signals to the frequency domain using the fast Fourier transform (FFT) as depicted in Figure 1. Thus, we derive the frequency f used at time t from the power consumption data at time t, $P(t)$:

$$f = FFT(P(t)) \tag{3}$$

By performing FFT, we divide power consumption data into the corresponding frequency components independent of the voltage to greatly simplify the frequency modeling.

Also, we apply the support vector machine (SVM) algorithm, which is a powerful machine learning algorithm for classification, to classify f into the closest discrete frequency provided by the target processor. In this way, we train the frequency model M_f:

$$M_f \triangleq SVM(f)\ \forall f \in \text{power data trace} \tag{4}$$

where f is detected via FFT in (3).

B. Training the Voltage Model

After training M_f, we use M_f to process power data to group the power signals with the same frequency f but with potentially different voltages together and use them to train the voltage model M_v in C-DVFS. In particular, we apply SVM as illustrated in Figure 1 to classify the power signals with the same frequency f into the signals with *a fixed number of discrete voltage levels* in the target processor:

$$M_v \triangleq SVM(f, v_i)\ \forall v_i \text{ provided by the processor.} \tag{5}$$

C. Training the Key Prediction Model

After training the frequency and voltage models, we train the key estimation model based on the identified frequency and voltage as shown in Figure 1. To train the key estimation models, we normalize the power consumption data to the power consumption by f_{max} and v_{max} as discussed before. (A detailed description is given in Section IV-D.) Further, to efficiently train the key prediction model M_p, we select most relevant features only, since a large number of CPU cycles are used to encrypt data using a sophisticated cryptographic algorithm such as AES. Also, many instructions performing non-cryptographic algorithms are relatively irrelevant. Thus, it is required to select relevant features to improve the training performance by reducing the impact of irrelevant features on the model.

There are many techniques to select relevant features in machine learning [38]. Among them, the principal component analysis (PCA) technique [39], [40] is a well established technique. PCA utilizes orthogonal transformation to convert a set of features to linearly uncorrelated variables, called the principal components (PCs) [40]. Each PC is a linear combination of the original features. Thus, the least square method can be applied to project data on the PCs. As a result, the PCs become the lower-dimensional representations of the original features that can be used to select features for training models. (A detailed discussion of PCA for dimensionality reduction in power analysis attacks is omitted due to space limitations.) After dimensionality reduction via PCA, the remaining training process of M_p is similar to leading-edge power analysis techniques, e.g., [13], [14], [15], which are DVFS-agnostic.

D. Performing a C-DVFS Attack

When all the models, M_f, M_v, and M_p, are trained, an attacker verifies them using the test set as discussed before. (If the result of the test is not satisfactory, s/he may perform more training.) If the models provide sufficient accuracy, an attacker mounts an attack on the target system. Our proposed attack, C-DVFS, is summarized in Algorithm 1 and discussed in the subsection.

Algorithm 1: C-DVFS Attack

```
1 while attack in progress do
2     while data collection do
3         slide the window to identify the next time
            interval, [t_start, t_end], in which the same
            frequency f and voltage v are employed by
            DVFS based on M_f and M_v;
4         for t ∈ [t_start, t_end] do
5             derive power data P̂(t) with respect to f_max
                and v_max using (6);
6     Apply PCA;
7     Apply M_p to extract secret keys;
```

During the data collection phase, C-DVFS identifies all the frequencies and voltages employed by DVFS using a sliding window based on our frequency and voltage models, i.e., M_f and M_v, to identify every time interval where DVFS uses a pair of a specific frequency f and voltage v, e.g., 1GHz and 1.1V, in Lines 2–3 of Algorithm 1.

Since f and v are constant in $[t_{start}, t_{end}]$, we can transform the power consumption by f and v with respect to f_{max} and v_{max} as follows:

$$\hat{P}(t) = P(t_{start} + (t - t_{start}) \times \frac{f_{max}}{f}) \times \frac{v_{max}^2}{v^2} \tag{6}$$

because the instructions executed between $[t_{start}, t]$ would have been completed earlier at $t_{start} + (t - t_{start}) \times \frac{f}{f_{max}}$ if f_{max} instead of f had been used. Also, the impact of voltage scaling is handled in (6) by transforming the power value with respect to v_{max}. Thus, the processed signal $\hat{P}(t)$ is characterized with the unified f_{max} and v_{max}.

After computing $\hat{P}(t)$ for every discrete time in $[t_{start}, t_{end}]$ in Lines 4–5 in Algorithm 1, C-DVFS repeats Lines 3–5 in Algorithm 1 for the next time interval that uses f' and v' where $f' \neq f$ and/or $v' \neq v$ due to DVFS until enough data for a power analysis attack are collected.

Once enough data are collected, the C-DVFS attack applies PCA in Line 6 of Algorithm 1 to reduce the dimensionality and applies M_p to extract secret keys in Line 7 where M_p can be any advanced model for DVFS-unaware power analysis attacks, e.g., [13], [14], [15], as discussed before.

In summary, C-DVFS is new in that it: 1) identifies the frequencies and voltages used by DVFS and 2) transforms the power data trace to characterize them via the unified f_{max}

and v_{max}, as if no DVFS has been undertaken. Therefore, C-DVFS significantly diminishes the effectiveness of DVFS as a defense against power analysis attacks aware of DVFS.

V. S-EDF: Mitigating the Impact of C-DVFS Attacks

TABLE I
PERCENTAGE OF MATHEMATICAL INSTRUCTIONS IN AES

Instruction	Number	Percentage
add	3581	16.5%
xor	860	3.9 %
sub	288	1.3 %

Algorithm 2: S-EDF: Secure Real-Time Scheduling for Mitigating C-DVFS Vulnerabilities

```
1 while EDF-Q is not empty do
2     τ_ij = head(EDF-Q);
3     execute τ_ij for one time unit;
4     if τ_ij completes after running total c_i < C_i then
5         slack = C_i − c_i;
6         while slack > 0 and EDF-Q is empty do
7             randomly execute add, sub, or xor;
8             slack = slack - elapsed time;
```

Cryptographic algorithms usually runs several common mathematical instructions. For example, Table I lists frequently used mathematical instructions in AES. As shown in the table, add, sub, and xor together sum to 22% of the instructions executed in AES. Based on this observation, our approach, S-EDF, extends the EDF scheduling algorithm to mitigate C-DVFS vulnerabilities in real-time embedded systems as summarized in Algorithm 2. In EDF, jobs (periodic instances of the real-time tasks) are sorted in non-descending order of deadlines such that the shortest deadline job at the head of the EDF queue is executed first. Also, the currently running job is preempted when a new job with a shorter deadline arrives. EDF is optimal and it can meet all deadlines if the total utilization does not exceed 1 [19] (discussed in Section III). Notably, S-EDF meets all deadlines, while alleviating the potential impact of DVFS-aware attacks, e.g., C-DVFS.

Real-time tasks often finish before their WCETs since worst-case rarely happens in practice. If a real-time job τ_{ij}—the j^{th} periodic instance of the real-time task τ_i—finishes after executing for total c_i time units smaller than its WCET C_i, the slack time $C_i - c_i$ is generated. During the slack time, we randomly execute add, sub, and xor instructions using any data in the registers (that are not keys) rather than doing DVFS as described in Algorithm 2. (If there is a secret key in any register, we replace it with a random data.) In this way, we insert noise in power traces to reduce the vulnerability of embedded systems to power analysis attacks, causing no deadline miss. Our approach can better protect secrete keys in real-time embedded systems even in the presence of C-DVFS attacks or other possible power analysis attacks in the future designed to minimize the effectiveness of DVFS in protecting secrets from power analysis.

The protection of secretes in non-real-time systems against power analysis attacks can also be enhanced by running an adapted version of Algorithm 2 when the system becomes idle or when certain performance reduction due to random cryptography-like instruction executions is acceptable. A thorough investigation is reserved for future work.

VI. Performance Evaluation

TABLE II
MCPAT PARAMETERS

Parameter	Value
number of cores	1
virtual address width	32 bits
physical address width	32 bits
virtual memory page size	4096 bytes
instruction length	32 bits
ALU per core	3
FPU per core	1

We evaluate the effectiveness of C-DVFS and Algorithm 2 using gem5 [20] micro-architecture simulator to simulate the ARM A9 [41] processor. We also use the McPAT power model [21] to simulate the processor power consumption. Although our basic power consumption model is similar to [21] as summarized in Table II, we need to extend McPAT to model per-instruction power consumption variations based on data values (e.g., key values).

In McPAT, overall power consumption is calculated based on the sum of the power consumption of individual instructions. In McPAT, the power consumption of main process components and the dynamic, short-circuit, and leakage power consumption [21] are modeled. McPAT simulates power consumption per instruction that goes through logical units in a processor, since different logical units have different circuits and power consumption. It is insufficient, however, for our simulation, since the power consumption of the same instructions may vary based on the data values (e.g., key values) processed by the instructions. Further, power analysis attacks builds correlations between keys and power traces based on this observation [13], [14], [15].

To address this issue, we extend McPAT power model by modeling the power consumption of instruction i, p_i, as the weighted sum of p_t—the power consumption determined by McPAT based on the type of the instruction–and p_o we newly introduced to consider the power consumption due to operands as follows:

$$p_i = p_{i,t} + p_{i,o} = p_{i,t} + \alpha(\text{sum of operands} + \gamma_i) \quad (7)$$

where $\alpha < 1$ to give a higher weight to $p_{i,t}$ and γ_i is a random number generated using a Gaussian distribution.[1]

[1]In our simulation, $\alpha = 10^{-7}$. Also, the Gaussian distribution with mean $\mu = 0$ and standard deviation $\sigma = 10$ is used to generate γ_i.

Using the extended McPAT power model and the AES algorithm data encryption, we analyze the *accuracy of the proposed approaches and the baseline that performs DVFS-unaware differential power analysis attacks to represent the state-of-the-art machine learning techniques*, such as [13], [14], [15], in the following subsections.

A. Accuracy of Classifying DVFS Profiles

In this experiment, we simulate the power consumption of AES with different DVFS settings and measure the accuracy of the classification of the voltage and frequency, which is the foundation for DVFS-aware power analysis attacks, such as C-DVFS. For this experiment, we increase the frequency from 500MHz to 1600MHz by 50MHz at a time and change the voltage from 1.05 to 1.35V by the step size of 0.05V. We change the frequency and voltage incrementally by a relatively small step size to make frequency and voltage detection more challenging. We sample the power data traces at 3.2GHz (=2 $\times$ 1.6GHz) according to the Nyquist–Shannon sampling theorem [42]. We have observed that the frequency and voltage detection accuracy is 100% when the sample size used for training the frequency and voltage models, M_f and M_v, is 2000 or more. (Approximately 24% of the samples are used to train M_f and M_v, while the rest of the samples are used for testing.) We have achieved the complete accuracy, since FFT readily transforms power trace tuples into the frequency domain to detect the frequency for every tuple (Section IV-A) and our voltage classification via SVM is effective (Section IV-B). These results indicate that DVFS cannot generate enough noise to considerably decrease the success ratio of DVFS-aware power analysis attacks, such as C-DVFS attacks, which is further verified as follows.

B. DVFS-Unaware vs. DVFS-Aware Power Analysis Attacks

TABLE III
AES KEY EXTRACTION ACCURACY OF THE DVFS-UNAWARE BASELINE

Experiment	$f_1 \leftrightarrow f_2$ (MHz)	Accuracy (%)
1	600 $\leftrightarrow$ 750	24.5
2	750 $\leftrightarrow$ 900	25.9
3	900 $\leftrightarrow$ 1000	21.9
4	1000 $\leftrightarrow$ 1100	23.9
5	1100 $\leftrightarrow$ 1250	25.0
6	1250 $\leftrightarrow$ 1400	22.7

TABLE IV
AES KEY EXTRACTION ACCURACY OF C-DVFS

Experiment	$f_1 \leftrightarrow f_2$ (MHz)	Accuracy (%)
1	600 $\leftrightarrow$ 750	32.9
2	750 $\leftrightarrow$ 900	26.7
3	900 $\leftrightarrow$ 1000	27.2
4	1000 $\leftrightarrow$ 1100	46.3
5	1100 $\leftrightarrow$ 1250	44.0
6	1250 $\leftrightarrow$ 1400	38.6

In this subsection, we compare the performance of the DVFS-unaware baseline based on machine learning (similar to [13], [14], [15]) and C-DVFS in terms of key distillation accuracy they achieve via power analysis. The frequency is randomly alternated between f_1 and f_2 as shown in Tables III and IV, while the voltage is randomly varied between 1.05 and 1.25V with the step of 0.05V to make detecting frequency or voltage changes relatively hard. In each experiment in Tables III and IV, we have used 4 different keys to encrypt 128 different data using each key. Among the 128 data entries, we use 40 entries for training and the rest for testing.

As shown in Table III, the accuracy of the baseline ranges between 21.9–25.9%. On the other hand, the accuracy C-DVFS ranges between 26.7–46.3% as summarized in Table IV. C-DVFS increases the accuracy by approximately 1% only when the frequency alternates between 750 and 900MHz in Experiment 2; however, it achieves the biggest accuracy increase of about 22% compared to the DVFS-unaware baseline as the frequency alternates between 1 and 1.1GHz in Experiment 4. Therefore, our DVFS-aware attack, viz. C-DVFS, poses considerable security threats. Also, it demonstrates that DVFS is not as reliable against power analysis attacks as claimed in recent works [17], [18] that have not considered DVFS-aware attacks, such as C-DVFS.

C. Effectiveness of S-EDF

TABLE V
AES KEY DISTILLATION ACCURACY OF C-DVFS UNDER S-EDF

Exp.	f (MHz)	Slack (%)	Accuracy (%)	Energy $\uparrow$ (%)
1	750	19.7	24.5	13.7
2	900	16.4	27.8	12.9
3	1000	14.8	25.0	11.3
4	1100	13.5	24.5	10.3
5	1250	11.8	23.0	8.3
6	1400	10.6	27.3	7.3

In this subsection, we evaluate the performance of S-EDF against C-DVFS attacks. In our experiments summarized in Table V, we consider several different frequencies and (use the fixed voltage 1.25V). As the frequency is increased from 750MHz to 1.4GHz, the slack time is reduced since the processor executes more instructions within a unit time. In our experiments, we have intentionally considered a relatively small slack times that range between 10-20% approximately. In this way, we make it more challenging for S-EDF to insert random cryptography-like instructions to alleviate the potential impact of C-DVFS attacks. As shown in Table V, the accuracy of C-DVFS's key extraction under S-EDF ranges only between 23–27.8%. Recall that, in Table IV, the accuracy of C-DVFS under EDF with no random insertions of cryptography-like instructions during the slack time ranged between 26.7–46.3%. Although the smallest accuracy decrease of C-DVFS between EDF and S-EDF is only 2% for Experiment 3 as summarized Tables IV and V, the biggest accuracy decrease due to S-EDF is roughly 22% regarding the Experiment 4

results (in Tables IV and V). Therefore, we observe that S-EDF considerably reduces the potential threat imposed by C-DVFS. A downside of S-EDF is it may consume more energy as it inserts random encryption-like instructions to mitigate advanced power analysis attacks instead of doing DVFS. The additional energy consumption of S-EDF, however, is acceptable: it ranges between 7.3–13.7%, while the slack time ranges between 10.6–19.7%. Essentially, there is a trade-off between energy efficiency and power analysis attacks, which is another interesting observation of this paper.

D. C-DVFS Parameters

Notably, our attack, C-DVFS, has only two parameters as specified in Algorithm 1: 1) the sliding window size w_s and 2) the step size $step_s$ for moving the sliding window. Thus, it is easy to mount C-DVFS attacks. In our simulation, the voltage and frequency detection accuracy reach 100% when w_s is at least 1280 sampled data which is equivalent to 200 CPU cycles for 500MHz CPU frequency and 560 CPU cycles for 1400MHz CPU frequency from original power consumption data. In addition, when we set $step_s = 300$ sampled data, our design can accurately detect the frequency or voltage change.

VII. Conclusions and Future Work

In this paper, a new power analysis attack, C-DVFS, that takes DVFS into account to improve the accuracy of key distillation is devised. Further, we design a new real-time scheduling policy, S-EDF, that extends the EDF scheduling algorithm to improve the resilience of real-time embedded systems against DVFS-aware C-DVFS. In our simulation study, C-DVFS attacks considerably improve the accuracy of key distillation under EDF, while S-EDF effectively mitigates C-DVFS attacks. Overall, the problem of DVFS-aware power analysis attacks is under-explored with many open research issues for future work. For instance, S-EDF reduces the accuracy of key extraction by inserting random instructions; however, it consumes more energy proportional to the slack time (if any). Although a novel approach that can minimize both the attack success ratio and the required energy consumption is desirable, it is a very challenging problem beyond the scope of this paper. Other machine learning techniques, e.g., deep learning, could be applied to further enhance the accuracy of C-DVFS attacks. Also, a thorough investigation of the applicability of C-DVFS attacks against other systems, such as could computing frameworks, is warranted. For example, attackers might be able to adapt/extend C-DVFS attacks by analyzing power trace data collected by operating systems or power management schemes in the cloud without requiring physical access to the cloud. New countermeasures will be needed, if such attacks turn out to be successful. To summarize, in this paper, we show that DVFS-aware power analysis attacks pose a real threat. We also take a first step to mitigating the threat in the context of real-time embedded systems.

VIII. Acknowledgement

This work was supported, in part, by NSF Project CNS-1526932. We appreciate anonymous reviewers for their help to enhance the paper.

References

[1] T. Yu, V. Sekar, S. Seshan, Y. Agarwal, C. Xu, *Handling a trillion (unfixable) flaws on a billion devices*. Proceedings of the 14th ACM Workshop on Hot Topics in Networks, 2015.

[2] P. Kocher, R. Lee, G. McGraw, A. Raghunathan, S. Ravi, *Security as A New Dimension in Embedded System Design*. Proceedings of the 41st annual Design Automation Conference, 2004, Pages 753-760.

[3] R. Spreitzer, V. Moonsamy, T. Korak, S. Mangard, *Systematic Classification of Side-Channel Attacks: A Case Study for Mobile Devices* . IEEE Communications Surveys and Tutorials, 2018, Volume 20, Issue 1.

[4] Y. Michalevsky, A. Schulman, G. A. Veerapandian, D. Boneh, G. Nakibly, *PowerSpy: Location Tracking Using Mobile Device Power Analysis*. USENIX Security Symposium 2015, Pages 785-800.

[5] P. Kocher, J. Jaffe, B. Jun, *Introduction to Differential Power Analysis and Related Attacks*. http://www.cryptography.com/dpa/technical, 1998.

[6] P. C. Kocher, J. Jaffe, B. Jun, *Differential Power Analysis: Leaking Secrets*. Crypto Springer-Verlag, LNCS 1666, Pages 388-397, 1999.

[7] S. Chari, J. R. Rao, P. Rohatgi, *Template Attacks*. CHES, Volume 2523 of LNCS, Pages 13-28, 2002.

[8] P.C. Kocher, P.C., Jaffe, J., Jun, *Differential power analysis*. Advances in Cryptology, Volume LCNS 1666, Pages 388397, Springer-Verlag, USA (1999).

[9] M. Petrvalsky, M. Drutarovsky, M. Varchola, *Differential power analysis attack on ARM based AES implementation without explicit synchronization*. 24th International Conference Radioelektronika, 2014.

[10] S.B. Ors, F. Gurkaynak, E. Oswald, B. Preneel, *Power-analysis attack on an ASIC AES implementation*. International Conference on Information Technology: Coding and Computing, 2004.

[11] C. Herbst, E. Oswald, S. Mangard, *An AES Smart Card Implementation Resistant to Power Analysis Attacks*. Applied Cryptography and Network Security, 2006.

[12] O. Lo, W. Buchanan, D. Carson, *Power analysis attacks on the AES-128 S-box using differential power analysis (DPA) and correlation power analysis (CPA)*. Journal of Cyber Security Technology, 2017, Pages 88-107.

[13] G. Hospodar, B. Gierlichs, E. De Mulder, I. Verbauwhede, J. Vandewalle, *Machine learning in side-channel analysis: A first study*. Journal of Cryptographic Engineering, 2011.

[14] L.Lerman, G. Bontempi, O. Markowitch, *Side channel attack: an approach based on machine learning*. Center for Advanced Security Research, 2014.

[15] Y. Souissi, M. Nassar, S. Guilley, J. Danger, *First Principal Components Analysis: A New Side Channel Distinguisher*. International Conference on Information Security and Cryptology, 2010.

[16] F. Dehmelt, *Adaptive (Dynamic) Voltage (Frequency) Scaling - Motivation and Implementation*. Texas Instruments, 2014.

[17] K. Baddam, M. Zwolinski, *Evaluation of Dynamic Voltage and Frequency Scaling as a Differential Power Analysis Countermeasure* . VLSID, 2007.

[18] S. Yang, P. Gupta, M. nWolf, D. Serpanos, V.Narayanan, Y. Xie, *Power Analysis Attack Resistance Engineering by Dynamic Voltage and Frequency Scaling*. ACM Transactions on Embedded Computing Systems, 2012.

[19] Jane Liu, *Real-Time Systems*. Prentice Hall, 1st Edition, 2000.

[20] *The gem5 Simulator: A modular platform for computer-system architecture research*. http://www.gem5.org/.

[21] S. Li, J. Ahn, R. Strong, J. Brockman, D. Tullsen, N. Jouppi, *McPAT: An Integrated Power, Area, and Timing Modeling Framework for Multicore and Manycore Architectures*. 42nd Annual IEEE/ACM International Symposium on Microarchitecture, 2009.

[22] *Data Encryption Standard (DES)*. Federal Information Processing Standards Publication (FIPS PUB 46-3), Volume 25, Pages 1-22, 1999.

[23] *ADVANCED ENCRYPTION STANDARD (AES)*. Federal Information Processing Standards Publication 197, 2001.

[24] J. Jonsson, B. Kaliski, *RSA Cryptography Specifications*. Internet Engineering Task Force(IETF), Version 2.1, 2003.

[25] M. Rahaman, M. Hossain, *Side channel attack prevention for AES smart card*. Proceedings of 11th International Conference on Computer and Information Technology, 2008, Pages 376-380.
[26] L. Lerman, S. Medeiros, G. Bontempi, O. Markowitch, *A machine learning approach against a masked AES*. Lecture Notes in Computer Science, 8419 LNCS, Pages 6-75, 2014.
[27] L. Molina, L. Belanche, A. Nebot, J. Girona, C, Campus Nord, *Feature Selection Algorithms: A Survey and Experimental Evaluation*. Data Mining, 2002.
[28] L. Breiman, *Random Forests* . Machine Learning, Volume 45, Issue 1, Pages 5-32, 2001.
[29] S.B. Kotsiantis, *Supervised Machine Learning: A Review of Classification Techniques*. Informatica, Volume 21, Pages 249-268, 2007.
[30] T. Kohonen, *The self-organizing map* . Proceedings of the IEEE, Volume: 78, Issue 9, 1990.
[31] C. Cortes, V. Vladimir *Support-vector networks*. Machine learning 20.3 (1995): 273-297.
[32] M. Hearst, S. Dumais, E. Osman, J. Platt, *Support vector machines*. IEEE Intelligent Systems archive, Volume 13, Issue 4, Pages 18-28, 1998.
[33] T. Popp, S. Mangard, E. Oswald, *Power Analysis Attacks and Countermeasures*. IEEE Design and Test of Computers, 2007, Volume 24, Issue 6, Pages 535-543.
[34] M. Bambagini, M. Marinoni, H. Aydin, G. Buttazzo, *Energy-Aware Scheduling for Real-Time Systems: A Survey*. ACM Transactions on Embedded Computing Systems, 2016.
[35] *The R Project for Statistical Computing*. https://www.r-project.org/.
[36] *TensorFlow: An open source machine learning framework for everyone* https://www.tensorflow.org/.
[37] Bruce Schneier, *Applied Cryptography*, John Wiley & Sons, 1996.
[38] G. Chandrashekar, F. Sahin, *A survey on feature selection methods*. Computers and Electrical Engineering, Volume 40, Pages 16-28, 2014.
[39] R documentation https://www.rdocumentation.org/.
[40] I.T. Jolliffe, *Principal Component Analysis*. Springer, Second Edition, 2002.
[41] *Cortex-A9 Technical Reference Manual*. Revision r4p1, www.arm.com, 2012.
[42] A.I. Zayed, *Advances in Shannon's Sampling Theory*. CRC Press, 1st Edition, 1993.