

Power Attack Resistant Cryptosystem Design: A Dynamic Voltage and Frequency Switching Approach

Shengqi Yang[†], Wayne Wolf[†], N.Vijaykrishnan[‡], D.N.Serpanos[¶] and Yuan Xie[‡]

[†]Department of Electrical Engineering, Princeton University, Princeton, NJ, 08544

[‡]Microsystems Design Lab, The Penn State University, University Park, PA, 16802

[¶]Electrical and Computer Engineering Department, Patras University, Greece

{shengqi|wolf}@princeton.edu {vijay|yuanxie}@cse.psu.edu {serpanos}@ee.upatras.gr

Abstract—A novel power attack resistant cryptosystem is presented in this paper. Security in digital computing and communication is becoming increasingly important. Design techniques that can protect cryptosystems from leaking information have been studied by several groups. Power attacks, which infer program behavior from observing power supply current into a processor core, are important forms of attacks. Various methods have been proposed to countermeasure the popular and efficient power attacks. However, these methods do not adequately protect against power attacks and may introduce new vulnerabilities. In this work, we addressed a novel approach against the power attacks, i.e., Dynamic Voltage and Frequency Switching (DVFS). Three designs, naive, improved and advanced implementations, have been studied to test the efficiency of DVFS against power attacks. A final advanced realization of our novel cryptosystem was given out, which achieved enough high power trace entropy and time trace entropy to block all kinds of power attacks, with 27% energy reduction and 16% time overhead for DES encryption and decryption algorithms.

I. INTRODUCTION

This paper describes a novel DVFS approach to improve the security of cryptographic algorithms. Our approach uses hardware techniques to obscure the cryptoprocessor's activities, making it more difficult to attack. Differential power analysis (DPA), which identifies cryptographic keys by monitoring processor power supply current, is a very real threat that has been used to break the security of smart cards [1]. Our new method makes it substantially harder for an attacker to use DPA against a secure system.

Security is emerging as a discipline of utmost importance. This is especially true for embedded systems due to several unique security challenges [2]. Cryptography is the best known answer to these challenges, as it provides a way to securely encode digital information [3]. A cryptographic scheme is an encryption function $C = f(K, P)$ that maps an unprotected block of bytes P (plaintext) into a secure block of bytes C (cyphertext), and gives an easy way (a decryption function $P = f^{-1}(K, C)$) to recover the plaintext from the cyphertext if some additional information K (key) is available. Cryptographic algorithms can be implemented as hardware ASIC (Application Specific Integrated Circuits), software routines running on embedded cryptography processor (cryptoprocessor), or hybrid hardware-software modules. In this paper, we use a cryptoprocessor as a part of our novel cryptosystem, and try to realize a cost effective, flexible and power attack resistant platform for cryptographic algorithms software implementation.

Numerous attacks have been devised to break cryptoprocessor and can be classified into three categories, i.e., software attacks [4], microprobing/physical attacks [5], and side-channel attacks [6]. Software attack uses the normal communication interface of the processor and exploits security vulnerability found in the protocols, cryptographic algorithms, or their implementation. Microprobing is an invasive technique that involves physical manipulation of the cryptoprocessor by depackaging, layout reconstruction, manual microprobing and memory/bus readout. Side-channel attacks include timing analysis [7], power analysis [1], electromagnetic analysis [8] and fault induction [9]. Here, we are most interested in protection of cryptoprocessor against power attacks, which are considered as a fundamental class of attacks that are practically quite difficult to defend against.

Various countermeasures against power attacks have been proposed to remove the symptoms that make a cryptoprocessor vulnerable to monitoring and analysis of side-channel information. The proposed defenses, for example, reduction of signal sizes [1], introduction of noise to signals [10], self-timed circuit [11], [12], masking techniques [13], [14], and duplication techniques [15], mainly try to make it more difficult to measure. These techniques either make the sample size for the analysis significantly large, or try to break the ability of the attacker to correlate information which is used for statistical analysis to work.

In this work, we propose a novel cryptosystem design using Dynamic Voltage and Frequency Switching (DVFS) against all kinds of power attacks. We show that naive random changes to the operating voltage and frequency are not sufficient to obscure the operation of the encryption system. We develop advanced circuit methods that substantially obscure encryption operations at minimum hardware cost in addition to much power savings without missing the timing deadline. The remainder of this paper is organized as follows. Section II presents some background material and motivates this work. Section III describes the novel cryptosystem design and experimental results. Section IV concludes this paper.

II. BACKGROUND AND MOTIVATION

This section describes previous work that motivates our study of DVFS. Specifically, we briefly introduce a typical cryptographic algorithm, DES algorithm, which will be used as a benchmark to test the efficiency of our novel cryptosystem. Then we discuss power attack techniques, which include Simple Power Attack (SPA) and Differential Power Attack (DPA). We also review the traditional idea of DVFS and show the potential of DVFS in the design of tamper resistant cryptosystem. Finally we point out the issues our novel approach needs to address and motivation of this work.

A. Data Encryption Standard

The DES [16] algorithm is the most widely used symmetric cryptographic algorithm. It uses 64 bits for the key, of which 56 bits are used for encryption and decryption while the other 8 bits are used for error detection in key generation, distribution and storage. The plaintext is permuted first before it goes to the encryption process. The core of the DES encryption process consists of 16 identical rounds, each of which has its own sub secret key, called K_n ($n = 1, 2, \dots, 16$). Each K_n is 48 bits produced from the original secret key using key permutation and shift operations. Each round consists of 8 S-box selection functions. Each of the unique selection functions, S_1, S_2, \dots, S_8 , takes a 6-bit block as input and yields a 4-bit block as output. For DES decryption algorithm, it is the reverse of encryption algorithm.

B. Power Attacks

B.1 Simple Power Attack

SPA is based on measuring and analyzing power traces during cryptographic computations. Basically by using inspection, the attacker analyzes the trace and identifies executed instructions (such

as whether a branch at time t is taken or not taken or whether a multiplication or squaring operation is performed) and derives data bits of the involved operands (including secret keys). Due to its straightforward methodology, protecting against this type of simple attack can be achieved fairly easily by restructuring the code [17], introducing noise into power measurement by adding dummy modules which are activating at random intervals [10], etc. The key point of countermeasures is tuning the power traces to prevent information leakage.

B.2 Differential Power Attack

DPA is currently the most popular high order power analysis and is a very real threat which has been used to break the security of smart cards. This scheme utilizes power traces gathered from several runs and relies on the power consumption variation due to data dependency to break the key. It proceeds as follows. A DPA attack begins by running the encryption algorithm for N random values of plaintext inputs. For each of the N plaintext inputs, P_i , a discrete power signal, $S_i[j]$, is collected using high speed analog to digital converters and the corresponding ciphertext, C_i , may also be collected. The power signal $S_i[j]$ is a sampled version of the power consumed during the execution of the algorithm that is being attacked. The index i corresponds to the P_i that produces the signal and the index j corresponds to the time of the sample. After the data collection, the attacker makes a hypothesis about the key. For example, if the target algorithm is DES algorithm, a typical prediction might be that the 6 key bits entering S-box4 are equal to '011010'. If correct, an assertion of this form allows the attacker to compute four bits entering the second round of the DES computation. If the assertion is incorrect, however, an effort to predict any of these bits will be wrong roughly half the time. For any of the four predicted bits, the $S_i[j]$ signals are split into two sets using a partitioning function, $D(\cdot)$:

$$\begin{aligned} S_0 &= \{S_i[j] | D(\cdot) = 0\} \\ S_1 &= \{S_i[j] | D(\cdot) = 1\} \end{aligned} \quad (1)$$

The next step is to compute the average power signal for each set:

$$\begin{aligned} A_0[j] &= \frac{1}{|S_0|} \sum_{S_i[j] \in S_0} S_i[j] \\ A_1[j] &= \frac{1}{|S_1|} \sum_{S_i[j] \in S_1} S_i[j] \end{aligned} \quad (2)$$

where $S_0 + S_1 = N$. By subtracting the two averages, a discrete time DPA bias signal, $T[j]$, is obtained:

$$T[j] = A_0[j] - A_1[j] \quad (3)$$

If the original hypothesis is incorrect, the criteria, $D(\cdot)$ function, used to create the subsets will be approximately random. Any randomly chosen subset of a sufficiently large data set will have the same average as the main set. As a result, the difference, $T[j]$, will be effectively zero at all points, and the attacker repeats the process with a new guess. If the hypothesis is correct, however, choice of the subsets will be correlated to the actual computation. In particular, the second round bit will be '0' in all traces in one subset and '1' in the other. When this bit is actually being manipulated, its value will have a small effect on the power consumption, which will appear as a statistically significant deviation from zero in the $T[j]$. This method allows the attacker to learn all 48 bits of the subkey for the 16th round. The remaining 8 bits can be found by brute force or by successively applying this approach backward to previous round.

From the above description, we can see that there is a very important assumption about DPA, i.e., the output of the S-Box must occur at a fixed time or the operational frequency of the cryptoprocessor must be constant. Thus, traces collected by an attacker can be correlated in time, so that the statistical analysis is

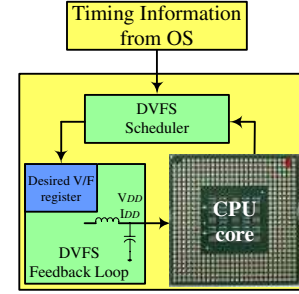


Fig. 1. DVFS cryptosystem.

feasible. If the power signals are collected at time J , while the output of the S-Box occurs at time J' when the attacker runs the algorithm with the guessed key, there will be no spike signals in $T[j]$. Dynamic frequency switching will give a fatal attack to the DPA.

C. Dynamic Voltage and Frequency Switching

DVFS was proposed as an effective approach to reduce energy and realized in two levels, low-level hardware support [18], [19] and high-level software support (operating system, compiler, etc) [20], [21]. In most current low-power DVFS processor designs [22]–[24], the voltage range is limited from full supply voltage V_{dd} to approximately half V_{dd} at most, for example, the voltage ranges and frequency ranges are 1.0V-1.8V and 153M-333MHz for IBM PowerPC 405LP processor, 0.8v-1.3v and 300M-1GHz for Transmeta Crusoe TM5800 processor, and 0.95V-1.55V and 333M-733MHz for Intel Xscale 80200 processor. Theoretical study [19] showed that extending the voltage range below $1/2V_{dd}$ will improve the energy efficiency for most processor designs, while extending this range to subthreshold operation is beneficial only for very specific applications.

D. Motivation

Based on the above analysis, we propose a new defense against the power attacks (SPA and DPA), which is effective, simple to implement and, unlike previous defenses, does not require algorithmic alterations. The defense is based on the implicit assumption of power attacks that the operational frequency of the cryptoprocessor is constant and thus, traces collected by an attacker can be correlated in time, so that the statistical analysis is feasible. The way to realize it is dynamic voltage and frequency switching during the execution of a cryptographic algorithm. Its methodology and implementation will be explained in the next section.

III. CRYPTOSYSTEM DESIGN

Our novel power attack resistant cryptosystem is composed of processor core, DVFS Feedback Loop (DVFSFL), and DVFS Scheduler (DVFS), as shown in Figure 1. The DVFSFL [18], [19] takes the value of desired Voltage/Frequency (V/F) as an object, which is stored in an internal register by the DVFS. DVFSFL physically implements the desired supply voltage and operating frequency by using ring oscillator, phase locked loop, etc, and finally outputs the supply voltage to the cryptoprocessor core. The DVFS unit randomly generates a voltage or frequency (V/F) value under certain limits and stores it in the register. This unit requires some timing information about the application program from the Operating System (OS), for example, the desired running time. In this work, we focus our attention on the design of DVFS unit which effectively makes our cryptosystem unbreakable by power attacks.

A. DVFS Efficiency Metrics

In order to quantitatively measure the efficiency of our DVFS technique against power attacks, we define three metrics: Signal Trace Entropy (STE), Energy Overhead (EO) and Time Overhead (TO). STE measures the uncertainty of the signal trace monitored by the attacker over time. To calculate STE, we bin the signal values in a signal trace into N equally spaced containers, and count the number of signal values in each container, n_i . Then STE can be expressed as Equation (4). Specifically in this work, we use Power Trace Entropy (PTE) and Time Trace Entropy (TTE) to represent the uncertainty of the power traces and the clock period traces, respectively. The power trace measures the power consumption for each clock cycle during the algorithm execution; while time trace gives the clock period for each clock cycle. For EO and TO, they measure the energy overhead and time overhead caused by using DVFS. We define $V_{dd,normal}$ and F_{normal} as the normal supply voltage and the normal operating frequency for the cryptoprocessor without DVFS. In general, they are the highest supply voltage and frequency. When the supply voltage is switched to a lower value $v_{dd,scale}$, the operating frequency f_{scale} is scaled according to Equation (5), where V_{th} represents threshold voltage. Finally, EO and TO can be expressed as Equation (6) and Equation (7), respectively, where $TNCC$ is the total number of clock cycles, and C_i is the capacitive load for i th clock cycle.

$$STE = - \sum_{i=1}^N \left(\frac{n_i}{\sum_{j=1}^N n_j} \log_2 \frac{n_i}{\sum_{j=1}^N n_j} \right) \quad (4)$$

$$f_{scale} = \frac{(v_{dd,scale} - V_{th})^{1.3}}{v_{dd,scale}} \times F_{normal} \quad (5)$$

$$EO = \sum_{i=1}^{TNCC} \frac{1}{2} C_i (v_{dd,scale}^2 - V_{dd,normal}^2) \quad (6)$$

$$TO = \sum_{i=1}^{TNCC} \left(\frac{1}{f_{scale}} - \frac{1}{F_{normal}} \right) \quad (7)$$

B. Experimental Setup

For the following designs and experiments, we make some reasonable assumptions. (1) In order to test the efficiency of the DVFS unit, the underlying hardware of the DVFSFL unit is assumed to be realized already and implements DVFS from full range 1.8V/450MHz to 0.9V/250M. Furthermore, we abstract its function as a software program to communicate with the cryptoprocessor and the DVFS unit. (2) The power overhead for the cryptoprocessor switching from high voltage/high frequency mode to low voltage/low frequency mode and vice versa is negligible. This assumption is tested to be reasonable in [18], [19]. For the power consumption by DVFS unit and DVFSFL unit, it is not counted when we measure the power traces and it is much less than that consumed by the cryptoprocessor. (3) For the cryptoprocessor, the supply voltage and operating frequency can be changed instantly during any clock cycle. It is an ideal condition. When an application runs for thousands of clock cycles, however, this ideal assumption tends to be true from a statistical viewpoint as voltage/frequency switching may just take a few clock cycles.

For this work, we implement the DES encryption and decryption algorithms in software as benchmarks, and capture their energy consumption at each clock cycle when running on the processor core by using a customized version of the publicly available SimplePower [25]. The cryptoprocessor is based on the architecture of a five-stage pipelined datapath which consists of the instruction fetch stage IF, the instruction decode stage ID, the execution stage EXE, the memory access stage MEM, and the write back stage WB. The simulator uses validated transition-sensitive and cycle-accurate energy models for both the datapath and memory systems. The flexibility of working with the simulator provides us the ability to

monitor the energy consumed in every cycle along with details of actual instructions executed.

C. Cryptosystem Design

For power analysis, the attackers uncover the secret key bits by observing some significant characteristics in the original power traces and the differential power traces after running the algorithm many times. If the measured power signal is random and cannot be repeated by running the algorithm any times using same plaintext and same key, the underlying secret key cannot be broken. Based on this analysis, an efficient power attack resistant method is to realize a random number generating algorithm in the pivotal design of the cryptosystem, i.e. DVFS unit, which can randomly generate a voltage/frequency value and store it in the designated register. After the random operating voltages are applied to the cryptoprocessor, the output signal will achieve the desired property. In the following, we will implement this DVFS unit step by step.

C.1 First DVFS: Naive Design

For each clock cycle of the cryptoprocessor, a naive and extreme design of the DVFS unit generates a random voltage value (frequency is determined by Equation (5)) and informs the DVFSFL unit to physically implement this voltage and feed it to the processor core. Here, we neglect the time consumed to generate the random number in the DVFS and the time consumed to implement it in the DVFSFL. This means the DVFS unit can generate as many random numbers as we want during a clock cycle and the DVFSFL unit can provide the cryptoprocessor with as many new voltage values as it needs in every clock cycle. With this assumption, we can analyze the effectiveness of DVFS against power attacks under the extreme condition. The algorithm of this naive implementation is shown in Figure 2. The procedure takes $V_{dd,normal}$, F_{normal} , $Signal_Done$ and RW_Enable as inputs; the first two signals come from the OS and give initial values for the DVFS; the third signal is from the processor core which indicates whether the encryption/decryption finishes; and the fourth signal is sent out from DVFSFL to reveal the status of the internal register which stores the desired voltage/frequency value. By using the initial voltage/frequency value, the DVFS generates random voltage/frequency number every clock cycle and outputs it to the DVFSFL register. DES encryption algorithm is tested in the cryptosystem and the power signal is measured by using customized SimplePower tool. In order to make the statistical results confident, the DES encryption algorithm is run for 1000 times on the naive implementation. The power traces without DVFS and with DVFS each clock cycle are shown in Figure 3. We list the EO, TO, PTE and TTE in Table I. For Figure 3(b), it is a typical case. However, its profile is similar as other power trace profiles with DVFS per clock cycle.

Figure 3(a) shows the power behavior of the DES encryption algorithm when running on the cryptoprocessor without DVFS technique. It clearly reveals the 16 DES rounds of operation. Figure 3(b) illustrates the power profile with DVFS per clock cycle, for DES algorithm running on the naive implementation. Although there is a little improvement on the information leakage, Figure 3(b) still makes the 16 DES rounds obvious. This phenomenon can be explained by the following fact. The total 16 DES rounds need about 211000 clock cycles when completely running on the processor core, and each round consumes about 10^5 clock cycles. For each round, a high power peak value lasts for about 2.5×10^4 clock cycles, while a low power peak value lasts for another 2.5×10^4 clock cycles. Although the voltage/frequency is randomly changed every clock cycle, which makes the power value for each clock cycle to be a random number, the power peak values still will appear when the power trace is plotted over such a big sample size as 2.5×10^4 clock cycles. This fact makes the naive implementation unable to mask the power consumption and prevent the information leakage. From Table I we can see, with an increase in the entropies of power trace and time trace, there is a

Input : $V_{dd_normal}, F_{normal}, Signal_Done, RW_Enable$
Output : V_{dd_scale/f_scale}

Algorithm :
1: **While** $Signal_Done \neq 1$ **Do**
2: generate random V_{dd_scale/f_scale} by using $V_{dd_normal}, F_{normal}$
3: **if** $RW_Enable == 1$ **then**
4: assign V_{dd_scale/f_scale} to register in DVFSFL
5: **End if**
6: **End While**

Fig. 2. Naive implementation of DVFS unit.

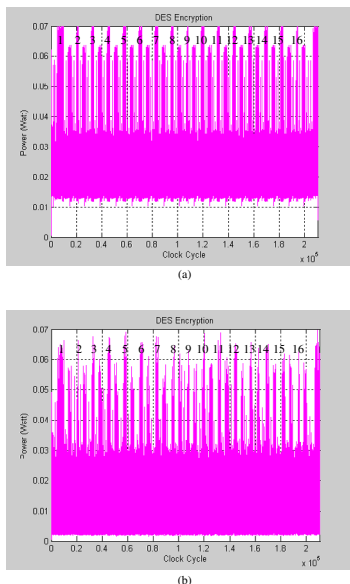


Fig. 3. Power trace for DES encryption algorithm without DVFS (a) and with DVFS per clock cycle (b).

big reduction on the total energy consumption and a big increase in the total time consumed by the DES algorithm, compared with that without DVFS.

TABLE I
PERFORMANCE COMPARISON FOR DES ENCRYPTION ALGORITHM
RUNNING ON THE CYRPTOSYSTEM.

DES Encryption	EO	TO	PTE	TTE
Without DVFS	0	0	4.96	0
With DVFS/clock cycle	-35.17%	26.55%	5.05	6.65

C.2 Second DVFS: Improved Design

We now propose an improved design. The algorithm is shown in Figure 4. The input and output signals of this implementation are same as that of the naive one. For the naive DVFS, the voltage/frequency is changed every clock cycle and the time consumption for the random number generating algorithm is neglected. In this improved implementation, once the voltage/frequency is changed, the DVFS unit will wait for a random time period before generating another random voltage/frequency value. As a result, there are two time periods introduced by this implementation, one is the time consumed by the random number generating algorithm, the other is the time consumed for waiting (Here, we still neglect the time consumed for voltage/frequency physical implementation in DVFSFL unit, and we treat it as a fixed time). The first time period is a constant (first order approximation), while the second time period is a random time period. As a final effect, the voltage/frequency is changed randomly in terms of value and generating time. In order

Input : $V_{dd_normal}, F_{normal}, Signal_Done, RW_Enable$
Output : V_{dd_scale/f_scale}

Algorithm :
1: generate random V_{dd_scale/f_scale} by using $V_{dd_normal}, F_{normal}$
2: assign V_{dd_scale/f_scale} to register in DVFSFL
3: **While** $Signal_Done \neq 1$ **Do**
4: generate a random number NCC
5: **for** $i=1:1:NCC$
6: NOP
7: **end for**
8: generate random V_{dd_scale/f_scale}
9: **if** $RW_Enable==1$
10: assign V_{dd_scale/f_scale} to register in DVFSFL
11: **end if**
12: **End While**

Fig. 4. Improved implementation of DVFS unit.

to test the efficiency of this implementation, power and clock period are measured and calculated per clock cycle.

Figure 5 illustrates the power traces and clock period traces for this improved implementation under two typical cases. The first case, as shown in Figures 5 (a1) and (a2), has long random waiting time between two random voltage/frequency generations. From the beginning to the end of the program running on the processor, The DVFS only generates 10 random voltage/frequency numbers. The second case, as shown in Figures 5 (b1) and (b2), however, has a very short waiting time and generates 211 random voltage/frequency values. It is obvious that the first case cannot mask the information leakage and clearly reveals the 16 DES rounds. While the second case shows a very random power trace, and the DES rounds are efficiently masked.

To quantify the statistical properties of these two cases, the algorithm implementation is modified. The total number of random voltage/frequency values is fixed to 10 (for 1st case) or 211 (for 2nd case), while the voltage/frequency values and the time when the voltage/frequency values are generated are randomly determined. The experiments are repeated to 1000 times to make the statistics confident. Finally, the EO, TO, PTE, and TTE are listed in Table II. It shows that the second case with short waiting time improves the PTE, TTE and EO compared with the first case with 1.06% more time consumption. Although the first case will appear with small probability when the cryptography algorithm runs on our cryptosystem, this case should be avoided.

TABLE II
PERFORMANCE COMPARISON FOR IMPROVED DVFS IMPLEMENTATIONS.

DES Encryption	EO	TO	PTE	TTE
long waiting time	-34.18%	25.15%	5.20	3.17
short waiting time	-35.23%	26.21%	5.33	5.97

To implement the advanced DVFS unit, there are two problems we need to solve. The first is long random waiting time which should be avoided. The second is tradeoff between energy reduction and timing overhead which is a big problem for real time cryptographic applications with restricted deadline time.

C.3 Third DVFS: Advanced Design

The algorithm of the advanced design of DVFS unit is shown in Figure 6. Two new signals, TB and $ETCC$, are input to the DVFS unit from the OS. Signals TB and $ETCC$ are estimations of the timing budget and the total clock cycles for the application program when running on the processor core. Both of them can be generated by the OS after the application program is compiled.

For the advanced DVFS unit, it first generates a random v_{dd_scale}/f_{scale} and stores it in the register for the processor core using it to run the program. After that, it initializes and determines the $HighLimit_WT$, which represents the high limit of the waiting

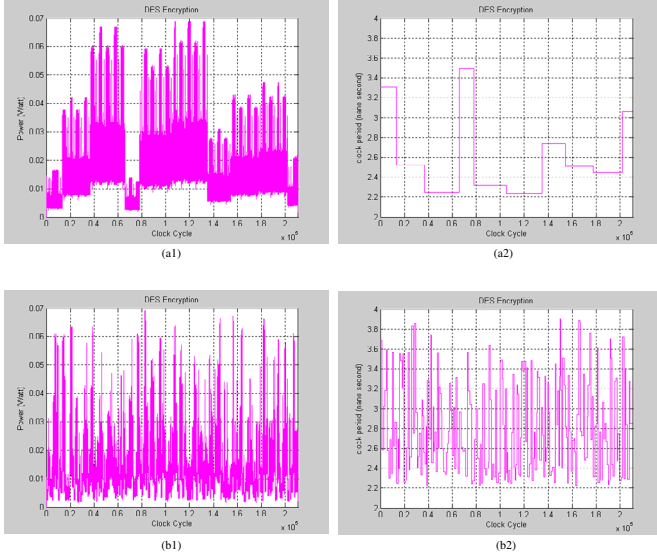


Fig. 5. Comparison between long random waiting time (for (a1) and (a2)) and short random waiting time (for (b1) and (b2)) in terms of power traces and clock period traces. For all the four figures, the X axis represents clock cycle number.

time. This limit prevents the DVFS unit from a long random waiting time. For convenience, this waiting time limit can be calculated by the OS and input to the DVFS unit. Here we assign it a constant number. During the program running in the processor core, the DVFS unit iteratively generates random voltage/frequency values. In each iteration, it firstly waits a random time limited by $High_limit_WT$, then calculates some timing information, determines the lowest value (LV) for the next voltage/frequency generation, and produces a random voltage/frequency which is not lower than the LV . As a final step, it outputs the voltage/frequency value to the register. Variables $timing$ and $timing_space$ count the total elapsed time and the time elapsed for one iteration, respectively. They are used to calculate the LV .

For our cryptosystem, the first goal is preventing the information leakage instead of saving power which is a by-product of DVFS. However, DVFS does bring much timing overhead that is not desired. In order to reduce it, the DVFS unit first calculate the lower bound, i.e. LV , and then generates the suitable voltage/frequency. Equation 8 shows how to calculate the lower bound for the scaled frequency. And the corresponding scaled supply voltage can be calculated by Equation (5).

$$LV = \frac{(ETCC - \sum_i timing_space / f_{scale}(i-1))}{(TB - (timing + \Delta))} \quad (8)$$

where $f_{scale}(i-1)$ represents the scaled frequency randomly generated in the $(i-1)th$ iteration. The summation is done over the number of iterations and calculates the total time elapsed. Δ is an estimation of the time spent for steps from 16 to 21. The numerator gives the total clock cycles needed to finish the remainder workload, while the denominator gives the total time left for the application program to meet its deadline according to the time budget. Finally, this equation gives the lowest frequency for the processor to finish the application. The DVFS unit uses this number as a lower bound and generates a random frequency number which is higher than the bound.

Figure 7 compares two differential power traces generated by using two different keys and the same plaintext before using DVFS (Figure 7 (a)) and after using advanced DVFS (Figure 7 (b)). And it illustrates the difference between two differential power traces

Input : V_{dd_normal} , F_{normal} , TB, ETCC,
Signal_Done, RW_Enable

Output : $V_{dd_scale/f_{scale}}$

Algorithm :

- 1: $time_1 = clock()$
- 2: generate random $V_{dd_scale/f_{scale}}$ by using V_{dd_normal} , F_{normal}
- 3: assign $V_{dd_scale/f_{scale}}$ to register in DVFSFL
- 4: initialize $timing$, $timing_space$ to zero
- 5: determine $High_limit_WT$
- 6: **While** Signal_Done != 1 **Do**
- 7: $time_3 = clock()$
- 8: generate a random number $NCC < High_limit_WT$
- 9: **for** $i=1:NCC$
- 10: NOP
- 11: **end for**
- 12: $time_2 = clock()$;
- 13: $timing = (time_2 - time_1) / clocks_per_second_DVFS$
- 14: $timing_space += timing_space$
- 15: use $timing$, $timing_space$, TB, ETCC to determine the lowest value of $V_{dd_scale/f_{scale}}$.
- 16: generate random $V_{dd_scale/f_{scale}} >$ the lowest value
- 17: **if** RW_Enable==1
- 18: assign $V_{dd_scale/f_{scale}}$ to register in DVFSFL
- 19: **end if**
- 20: $time_4 = clock()$
- 21: $timing_space = (time_4 - time_3)$
- 22: **End While**

Fig. 6. advanced implementation of DVFS unit.

generated by using two different plaintexts and the same key before using DVFS (Figure 7 (c)) and after using our advanced DVFS (Figure 7 (d)). For Figures 7 (a) and (b), the secret keys vary in the first bit; while for Figures 7 (c) and (d) the Plaintexts vary in the tenth bit. These bit variations are randomly selected, and same profiles can be observed for other choices. Figures 7 (a) and (c) demonstrate that without DVFS significant information is leaked by using different key bits or different plaintext bits to run the algorithms and doing time correlation. When the advanced DVFS is used, this information leakage is prevented as shown in Figures 7 (b) and (d).

Two factors contribute to this good property of our cryptosystem. (1) When the encryption/decryption program runs in our cryptosystem for the first time, operation A is executed at time t ; while it is executed at time $(t + \Delta t)$ when the program runs again. Δt represents a time shift with positive or negative value. With high probability, it is not equal to zero. As a result, at a fixed time point for many power traces which are achieved by using different plaintexts or keys, different operations are carried out in the processor core. This makes time correlation, which is a very important step in power attacks, impossible. (2) Even if the same operation is executed at time t , the supply voltages for different runs may be different with much probability. This makes the power values at time t for different traces unequal and results in a random number for DPA bias signals. All the two aspects efficiently prevent the advanced cryptosystem from the power attacks, such as SPA and DPA.

To test and quantify the efficiency of our advanced cryptosystem against power attacks, DES encryption and decryption algorithms were run in this cryptosystem for one thousand times. Table III shows the energy overhead, time overhead, power trace entropy, time trace entropy and Long Time Waiting Effect (LTWE) for DES encryption and decryption algorithms. From the table, we can see the PTE and the TTE are greatly increased and much higher than that of the improved implementation. As a result, the LTWE does not exist for DES encryption and decryption algorithms. Furthermore, the TO is much less than that of the previous two implementations with a sacrifice of energy reduction. As an advanced implementation, it overcomes all the shortcomings of the naive and the improved implementations. By adjusting the lowest value for v_{dd_scale}/f_{scale} in the algorithm to be a higher value, we can further reduce the time overhead. Experiments show that the TO range for encryption and decryption is about 17% to 5%.

Finally, we compare the performance of our DVFS cryptosystem and the cryptosystem without DVFS by using DES algorithm as a

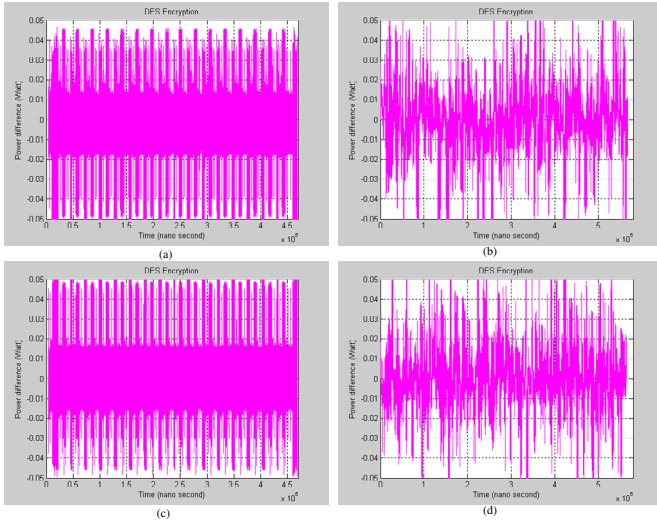


Fig. 7. Difference between power traces generated with two different keys (vary in the first bit of the key) before using DVFS (a) and after using advanced DVFSS (b); Difference between power traces generated using two different plaintext (vary in the 10th bit of the plaintext) before using DVFS (c) and after using advanced DVFSS (d).

TABLE III
ACHIEVED PERFORMANCE FOR DES ENCRYPTION AND DECRYPTION BY USING ADVANCED DVFSS.

DES	EO	TO	PTE	TTE	LTWE
Encryption	-27.32%	16.15%	5.42	6.02	No
Decryption	-26.89%	16.01%	5.44	6.05	No

benchmark and summarize it in Table IV. As we can see, our novel cryptosystem obtains 27% energy reduction, 7.5% increase in power trace entropy, and $\infty\%$ increase in time trace entropy, with a timing overhead of 16%, compared with the cryptosystem without DVFS.

TABLE IV
PERFORMANCE GAIN OF DVFSS CRYPTOSYSTEM OVER NORMAL CRYPTOSYSTEM WITHOUT DVFSS.

	Energy	Time	PTE	TTE
DVFS Cryptosystem	-27%	+16%	+7.5%	$+\infty\%$

IV. CONCLUSIONS

In this paper, we presented a design of novel power attack resistant cryptosystem, which uses dynamic voltage and frequency switching to make the power traces show random property and prevent the power attackers from doing time correlation between different power traces. Experiments were carried out through three levels, naive, improve, and advanced implementations, to test the efficiency and performance of this dynamic voltage and frequency switching approach. For the naive and the improved implementations, they did enhance the resistance of the cryptosystem to the power attacks, but with inborn shortcomings which make this cryptosystem still attackable. Advanced cryptosystem realization overcame all the shortcomings and exhibited excellent efficiency to block the SPA and DPA attacks. DES encryption and decryption algorithms were tested on this advanced cryptosystem. Results show that both the power trace entropy and time trace entropy are high enough to prevent information leakage. While the energy consumption is averagely reduced by 27%, the execution times for both algorithms are delayed

for 16%. This advanced DVFSS implementation provides an efficient method to countermeasure the popular power analysis attacks, and is difficult to be broken because of its random property.

REFERENCES

- [1] P. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis," in *Proc. 19th Intl. Advances in Cryptology Conference-CRYPTO'99*, Aug. 1999, pp. 388–397.
- [2] P. Kocher, R. Lee, G. McGraw, A. Raghunathan, and S. Ravi, "Security as A New Dimension in Embedded System Design," in *Proc. Design Automation Conf.*, June 2004, pp. 753–760.
- [3] B. Schneier, *Applied Cryptography, Protocols, Algorithms, and Source Code in C*, 2nd ed. New York, NY: John Wiley, 1996.
- [4] S. Ravi, A. Raghunathan, and S. Chakradhar, "Embedding Security in Wireless Embedded Systems," in *Proc. Int. Conf. VLSI Design*, Jan. 2003, pp. 269–270.
- [5] O. Kömmerling and M. G. Kuhn, "Design Principles for Tamper-resistant Smartcard Processors," in *Proc. USENIX Workshop on Smart-card Technology*, May 1999, pp. 1–12.
- [6] J. J. Q. adn D. Samyde, "Side-channel Cryptanalysis," in *Proc. SECI*, Sept. 2002, pp. 179–184.
- [7] P. Kocher, "Timing Attacks on Implementations of Diffe-Hellman, RSA, DSS and Other Systems," in *Proc. 16th Intl. Advances in Cryptology Conference-CRYPTO'96*, Aug. 1996, pp. 104–113.
- [8] J. J. Quisquater and D. Samyde, "ElectroMagnetic Analysis (EMA): Measures and Counter-measures for Smart Cards," in *Proc. 19th Intl. Advances in Cryptology Conference-CRYPTO'99*.
- [9] D. Boneh, R. DeMillo, and R. Lipton, "On the Importance of Eliminating Errors in Cryptographic Computations," *J. Cryptology*, vol. 14, no. 2, pp. 101–119, May 2001.
- [10] L. Benini, *et al.*, "Energy-Aware Design Techniques for Differential Power Analysis Protection," in *Proc. Design Automation Conf.*, June 2003, pp. 36–41.
- [11] S. Moore, R. Anderson, P. Cunningham, R. Mullins, and G. Taylor, "Improving Smart Card Security Using Self-timed Circuits," in *Proc. Int. Symp. Asynchronous Circuits and Systems*, Apr. 2002, pp. 211–218.
- [12] K. Tiri and I. Verbauwhede, "Securing encryption algorithms against DPA at the logic level: next generation smart card technology," in *Cryptographic Hardware and Embedded Systems, International Workshop, CHES'03*, ser. Lecture Notes in Computer Science, vol. 2779. Springer-Verlag, Aug. 2003, pp. 125–136.
- [13] H. Saputra, *et al.*, "Maksing the Energy Behavior of DES Encryption," in *Proc. Design & Test Europe Conf.*, Mar. 2003, pp. 84–89.
- [14] L. Benini, A. Galati, A. Macii, E. Macii, and M. Poncino, "Energy-Efficient Data Scrambling on Memory-Processor Interface," in *Proc. Int. Symp. Low Power Electronics and Design*, Aug. 2003, pp. 26–29.
- [15] L. Goubin and J. Patarin, "DES and Differential Power Analysis - the Duplication Method," in *Cryptographic Hardware and Embedded Systems, First International Workshop, CHES'99*, ser. Lecture Notes in Computer Science, vol. 1717. Springer-Verlag, Aug. 1999, pp. 158–172.
- [16] U. D. of Commerce/National Institute of Standards and Technology, *Data Encryption Standard (DES)*, fips pub 46-3 ed. Federal Information Processing Standards Publication, Oct. 1999.
- [17] J. Coron, "Resistance Against Differential Power Analysis for Elliptic Curve Cryptosystems," in *Cryptographic Hardware and Embedded Systems, 3rd International Workshop, CHES'01*, ser. Lecture Notes in Computer Science, vol. 1717. Springer-Verlag, May 1999, pp. 292–302.
- [18] K. J. Nowka, *et al.*, "A 32-bit PowerPC System-on-a-Chip with Support for Dynamic Voltage Scaling and Dynaic Frequency Scaling," *IEEE Jour. Solid-State Circuit*, vol. 37, no. 11, pp. 1441–1447, 2002.
- [19] B. Zhai, D. Blaauw, D. Sylvester, and K. Flautner, "Theoretical and Practical Limits of Dynamic Voltage Scaling," in *Proc. Design Automation Conf.*, June 2004, pp. 868–873.
- [20] K. Choi, R. Soma, and M. Pedram, "Off-Chip Latency-Driven Dynamic Voltage and Frequency Scaling for an MPEG Decoding," in *Proc. Design Automation Conf.*, June 2004, pp. 544–549.
- [21] S. Ghiasi, J. Casmira, and D. Grunwald, "Using IPC Variation in Workload with Externally Specified Rates to Reduce Power Consumption," in *Workshop on Complexity Effective Design (Held in conjunction with ISCA)*, June 2000.
- [22] "Transmeta crusoe." [Online]. Available: <http://www.transmeta.com>
- [23] "Intel xscale." [Online]. Available: <http://www.intel.com/design/intelxscale>
- [24] "Ibm powerpc." [Online]. Available: <http://www.chips.ibm.com/products/powerpc>
- [25] N. Vijaykrishnan, M. Kandemir, M. J. Irwin, H. S. Kim, and W. Ye, "Energy-Driven Integrated Hardware-Software Optimizations Using SimplePower," in *Computer Architecture, 2000. Proceedings of the 27th international Symposium on*, June 2000, pp. 95–106.