Energy-Aware Design Techniques for Differential Power Analysis Protection

Luca Benini*
lbenini@deis.unibo.it

Alberto Macii‡
alberto.macii@polito.it

Enrico Macii‡
enrico.macii@polito.it

Elvira Omerbegovic◇
elvira.omerbegovic@bulldast.com

Massimo Poncino#
poncino@sci.univr.it

Fabrizio Pro◇
fabrizio.pro@bulldast.com

◇BullDAST s.r.l.
Torino, ITALY

‡Politecnico di Torino
Torino, ITALY

*Univ. di Bologna
Bologna, ITALY

#Univ. di Verona
Verona, ITALY

ABSTRACT

Differential power analysis is a very effective cryptanalysis technique that extracts information on secret keys by monitoring instantaneous power consumption of cryptoprocessors. To protect against differential power analysis, power supply noise is added in cryptographic computations, at the price of an increase in power consumption. We present a novel technique, based on well-known power-reducing transformations coupled with randomized clock gating, that introduces a significant amount of scrambling in the power profile without increasing (and, in some cases, by even reducing) circuit power consumption.

Categories and Subject Descriptors

B.5 [**Hardware**]: Register-Transfer-Level Implementation; B.6 [**Hardware**]: Logic Design; B.7 [**Hardware**]: Integrated Circuits

General Terms

Design

Keywords

Low-power design, differential power analysis

1. INTRODUCTION

As electronics becomes more pervasive in every-day's life, an increasing amount of confidential information is electronically processed, stored and communicated.

DAC 2003, June 2-6, 2003, Anaheim, California, USA.

Protection of confidential data is thus becoming a serious concern for many electronic systems. Cryptography is the best known answer to this challenge, as it provides a way to securely encode digital information [1, 2]. In synthesis, a cryptographic scheme is an *encryption function* $C = f(P, K)$ that maps an unprotected block of bytes P (the *plain-text*) into a secure block of bytes C (the *cyphertext*), and gives an easy way (a *description function* $P = f^{-1}(C, K)$) to recover the plain-text from the cyphertext if some additional information K is available (the *key*).

Security protocols, which enable secure exchange of information, have cryptographic algorithms as basic building blocks [3]. Given their strategic importance in security engineering, many cryptographic schemes have been developed over the years. A complete taxonomy of the field is outside the scope of this work (the interested reader is referred to [1, 2]); however, a simple distinction can be made between *symmetric (or secret) key* and *asymmetric or (public) key algorithms*. Symmetric-key schemes rely on a secret key being in exclusive possession of securely communicating peers, while asymmetric schemes relax this assumption by requiring a secret key only for message decryption (encryption is performed using a public key). It is important to notice that both symmetric and asymmetric schemes require, at some point, the manipulation of a secret key. If an adversary comes into possession of the secret key, security is violated. Cryptographic algorithms can be implemented as hardware blocks or as software routines. The advantages of software solutions are in terms of flexibility and adaptability. Yet, dedicated hardware cryptoprocessors are still very common in practice, because many good algorithms have been standardized and they do not require in-field modifications. In addition to that, dedicated cryptoprocessors are significantly faster, more compact and much more energy-efficient than their SW counterparts [2, 4, 5, 6].

The work we present in this paper focuses on protection of cryptoprocessors against *power analysis*, a particularly insidious class of security attacks aiming at recovering the values of secret keys that are stored inside the cryptoprocessor.

Numerous attacks have been devised to break cryptoprocessors (good surveys are presented in [7, 3]). A very successful class of attacks is based on *side channel information*. The basic strategy in side-channel attacks is to monitor and measure various phenomena (side channels), such as electromagnetic emissions and power supply currents during encryption and decryption, and try to extract from them information on the secret key value. These attacks have reported a worrysome rate of success especially in cracking portable secure devices (e.g., smart cards) which cannot be kept under continuous surveillance and are frequently in contact with non-trusted devices (e.g., public smart card readers). In particular, power analysis (PA) techniques, which monitor the current absorbed over time by the cryptoprocessor, have been remarkably successful in breaking industry-standard cryptographic algorithms (e.g., DES, RSA) in smart cards [8].

To counter power analysis attack, various techniques have been proposed. Physical protection of the power supply side channel [9, 10] can be very effective but it is not always viable because of system constraints (e.g., cost limitations, power supply accessibility specifications). Software implementations have been protected by modifying the algorithms in an effort to de-correlate execution and secret key value [11]. Finally, hardware-supported techniques have been proposed that modify the cryptoprocessor architecture trying to enhance its robustness against PA [12, 13], at the price of significantly increased power consumption. Our approach belongs to the latter class. Similarly to previously published work, we increase PA resilience by adding random noise to deterministic encryption-decryption computation, but as a key element of novelty we do so without increasing power consumption. On the contrary, our approach *reduces power consumption* if compared with a baseline implementation.

Furthermore, we leverage EDA technology to enhance the applicability of our technique to many crypto-algorithms in an automated fashion. The basic idea is to provide a set of PA-resilient hardware primitives (soft macros) that can be instantiated by a cryptoprocessor designer. These primitives are commonly used in many algorithms. Additionally, the designer is allowed to trade off power savings for PA-resilience in a controlled fashion at design time. This is a very desirable feature, because encryption and decryption are often required in environments with tight power constraints. Preliminary results show that the application of our approach to the design of a HW unit for RSA encryption preserves the original average power consumption in spite of a non-deterministically scrambled pattern-by-pattern power profile.

The paper is organized as follows: Section 2 reviews the previous work on attacks based on side-channel information. Section 3 describes the design of the randomized, power-managed units. Finally, Section 4 gives some results on the power masking potential of these units.

2. POWER ATTACKS

Power analysis attacks come in many variants. The basic PA technique, known as *simple power analysis* (SPA) [14], monitors the power supply of a cryptoprocessor while executing a known encryption algorithm, and correlates the time-domain current waveform with various phases of the algorithm. If the algorithm has a key dependent execution flow, for instance if it has branches dependent on the value of some bits of the secret key, the current waveform presents easily recognizable features that reveal key information. SPA has limited effectiveness if the algorithm flow is data-independent, but it can be effective to crack naive implementations of the encryption algorithm.

Differential power analysis (DPA) [8] and its variants (such as higher-order power analysis [11]) is significantly more dangerous, in fact it is effective even when execution flow is not data-dependent, and for some encryption algorithms it does not even require knowledge of the plain-text. To perform DPA, an attacker needs a collection of m power traces $T_i[j]$, $i = 1, \ldots, m$ (j is the discrete time index of the values in the trace), and their corresponding cyphertext values C_i. The critical step in applying DPA is the definition of a *selection function* $D(K_b, C_i) \rightarrow \{0, 1\}$ that, given sub-key K_b consisting of a (small) subset of b key bits, can split the set of m traces and cyphertext values in 2 disjoint subsets. The definition of D depends on the encryption algorithm, and it is the critical step in a successful DPA attack. It can be viewed as a clustering algorithm, which given the cyphertext and assuming the knowledge of the key bits K_b, divides the cyphertext into two clusters.

DPA proceeds as follows. It assumes a value of the sub-key K_b, it applies the selection function to partition the $T_i[j]$ in two disjoint subsets D_0 and D_1, such that $T_i[j] \in D_1$ if $D(K_b, C_i) = 1$ and $T_i[j] \in D_0$ if $D(K_b, C_i) = 0$. Then, it computes the average trace for the two subsets, and their difference. In symbols:

$$\Delta[j] = \frac{\sum_{T_i \in D_1} T_i[j]}{|D_1|} - \frac{\sum_{T_i \in D_0} T_i[j]}{|D_0|} \qquad (1)$$

The crypto-analysts then analyzes $\Delta[j]$. If it appears like random noise, then the assumed value of the key bits is incorrect. If it shows visible peaks, then, with very high probability, the sub-key K_b has been found. Clearly, in the worst case all 2^b combinations of b bits must be tried, thus a viable selection function should operate on a small sub-key. Once the sub-key has been discovered, the analysis can then proceed to other bits. Viable selection functions have been found for industry-standard encryption algorithms like DES and RSA [14, 8], hence DPA is a serious treat to security, and practical demonstration of successful attacks on smart cards have been published [11].

The fundamental premise for the applicability of DPA is that the power profile for an encryption algorithm depends in some parts on the value of the secret key. All known DPA countermeasures attempt to falsify this premise. An intuitive approach is to enforce independence of $T_i[j]$ on the value of the secret key, or in other words, modify the algorithm or the hardware to reduce the sensitivity of the power profile to the secret key value. This approach is advocated in [13, 12]. Unfortunately, it is extremely difficult to guarantee perfect equalization, and high-resolution, high-accuracy DPA defeats equalization attempts. For this reason, most recently proposed DPA countermeasures focus on an alternative approach, namely *randomization* or, equivalently, *random noise insertion* [15].

Randomization techniques have many embodiments, ranging from randomized masking of the secret key, to clock noise insertion, to repetition of the algorithm on randomly generated secret keys, to embedding the secret key in a much larger random key [15, 16, 17].

Even though DPA can reduce the effect of randomization by

averaging over a larger number of traces, it has been shown that the number of traces to be averaged increases rapidly with the noise level [15]. Hence, DPA becomes ineffective because too many data need to be collected. For this reason, noise injection via randomization appears to be more generally accepted as DPA countermeasure than equalization.
Unfortunately, noise injection approaches proposed in the past always imply redundant computation (often, a significant amount of it), and they have a sizable power overhead. Our approach addresses this limitation. It is based on a simple intuition: Executing elementary computations in encryption algorithms in a randomized fashion, using functional blocks that non-deterministically select among different implementations of a functional unit. Power is saved because the alternative implementations are not structurally nor functionally equivalent, but they are low-power, limited functionality versions of the original unit. Randomly selecting them has the double beneficial effect of increasing algorithmic noise (thereby foiling DPA) and saving power (to amortize the cost of redundant hardware instantiation).

3. POWER MASKABLE UNITS

This section describes the general principles that enable the design of *power-managed* modules, and how the latter can be used as a base architectural template to realize power-maskable, low-power data-path units.

3.1 Power-Managed Units

The design of a power-managed unit is based on the principle of the "common case" computation. In its most general form, shown in Figure 1, a unit A, whose inputs are latched by a register $R1$, is put in parallel with a smaller block B, that implements the most typical behavior of A, with a smaller power cost. Block B may in general depend on fewer inputs (I_B) than the number of inputs of block A (I_A), as well as it might produce a result on fewer outputs (O_B) than those of A (O_A).

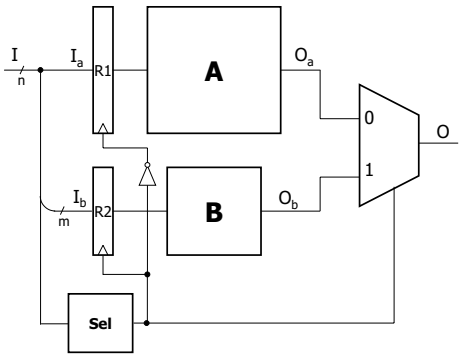

Figure 1: Architecture of a Power-Managed Unit.

Power savings are achieved by activating block B instead of the complete functionality A for most of the cycles. Unnecessary switching in either block is prevented by selectively "freezing" the inputs (thus reusing the values of the previous cycle) through the deactivation of the corresponding input registers. The choice of which block to activate in a given cycle is done by a selector (*Sel*) which, based on the observation of the inputs, enables the corresponding input registers. The selection logic also drives the output multiplexer that steers the appropriate output (A or B).
The energy savings come at the cost of extra area, that must be taken into account when evaluating the overall power of the unit. In general, if the probability of activating block B is significantly greater than 0, *average* power will benefit of this scheme. Such probability is equivalent to the fraction of cycles p in which the output of the *Sel* block evaluates to 1. This allows to derive a rough energetic balance of the power-managed scheme. Power decreases if:

$$(1-p)\cdot P_A + p\cdot P_B + P_{overhead} < P_A \qquad (2)$$

where, P_A (P_B) represents the average power consumption of block A (B), and $P_{overhead}$ accounts for the extra power consumption due to the selection function, the output MUX, and the corresponding wiring.
The scheme of Figure 1 is mostly conceptual, and its actual implementation must carefully consider the detailed timing of the operations. In the literature, this architectural template has found many implementations [18, 19, 20, 21, 22], that differ in the type of target designs (combinational vs. sequential circuits) and the abstraction level at which they are applied (gate-level vs. RTL). The reader is referred to the book of Benini and De Micheli [23] for a comprehensive survey of the various implementations.

3.2 Masking Power Consumption

Using power management to reduce the average power consumption of data-path units in cryptographic hardware is not beneficial from the point of view of their resistance to power attacks. As a matter of fact, the principle of power management is precisely that of explicitly exposing noticeable variations in the power consumption over time (i.e., separating high power-consuming "states" from low-power consuming ones). Therefore, power management schemes apparently seem to be incompatible with the objective of masking power consumption.
At a more careful analysis, however, we notice that the principle of instantaneous power randomization can also be applied to power-managed units. Conventional randomization exploits redundant hardware that adds noise to a given functionality (the entire algorithm, or part of it) so that the signal-to-noise ratio is reduced and a power attack is less likely to work. This addition is done at the cost of extra power consumption.
In contrast, power-managed units can create randomization in a quite natural way. Consider the variant of the basic power-managed architecture depicted in Figure 2, where the output of the *Sel* block is now ANDed with a hardware randomizer. This implies that probability p will be decreased of a random amount, depending on the output of the randomizer. On a cycle-by-cycle basis, this will randomly alter the power consumption of the power-managed module, thus achieving the objective of masking the power consumption over time, yet still with lower average power than the original unit.
The randomizer can be implemented in hardware as a linear feedback shift register (LFSR); the randomizing input that "modulates" the output of the *Sel* block can be any of the outputs of the LFSR. To further increase the degree of randomness, the LFSR may include a parallel input that can be used to load a new seed once every user-specified number of cycles.

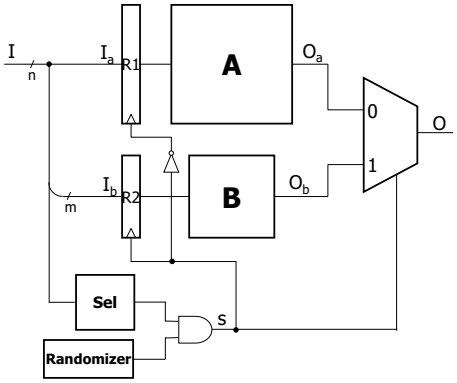

Figure 2: Power Masking of a Power-Managed Unit.

It should be observed that, given the typical duration of a cryptographic computation, a relatively small LFSR (i.e., less than 16-bit) already provides an acceptable degree of randomness. In fact, using appropriate characteristic polynomials, a r-bit LFSR can guarantee a repetition period of $2^r - 1$, i.e., any generated r-bit value will not be generated again within the successive $2^r - 1$ attempts. For a 16-bit LFSR, this guarantees about 16K repetition-free cycles, which is definitely a long enough period for any *single* cryptographic computation. In most cases, much smaller values of r may suffice.

The possibility of using a small-width LFSR reduces its hardware overhead. In fact, if we compare the typical size of the data inputs used in cryptographic applications (e.g., 1024-bit values), the addition of a 16-bit LFSR becomes barely noticeable.

3.2.1 Power Savings/Masking Tradeoffs

If the random number generator implemented by the LFSR provides a sufficiently uniform distribution, we can assume that the probability of any single output bit of the LFSR is about 0.5. This implies that the probability p used in Equation 2 must be replaced by $p' = \frac{p}{2}$. In many cases, where the value p is not very large, this reduction of the probability is too drastic.

We have thus devised a simple scheme that allows to trade off power reduction (i.e., the reduction of p) with the efficiency of the masking operated by the randomizer. The actual probability of activating block B (i.e., the probability output s of the AND gate in Figure 2) is given by $p' = p \cdot q$, where p is the probability of the *Sel* being 1, and q is the probability of the randomizing signal, under the assumption of their mutual statistical independence.

The most efficient randomization is achieved when $q = 0.5$. If we skew this probability towards either 0 or 1, we will increase the predictability of the masking process. However, lower values of q have different impact on the power consumption of the power-managed module than higher ones. In particular, values of $q > 0.5$ will be preferable, because they will tend to keep p' closer to p.

One way of increasing q simply consists of taking k LFSR outputs, and ORing them to generate the actual randomizing input that is to be ANDed with the output of the *Sel* block. This arrangement is shown in Figure 3.

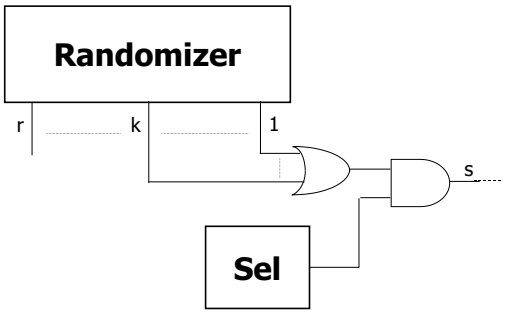

Figure 3: Trading Off Power Savings for Masking.

Assuming that all LFSR outputs are statistically independent of each other, the resulting probability q at the output of the k-input OR gate is given by:

$$q = 1 - \Pi_{i=1}^{k}(1 - Prob(o_i)) \quad (3)$$

where o_i is the i-th LFSR output, and *Prob()* its probability. Since $Prob(o_i) \equiv 0.5, \forall i$, Equation 3 is equivalent to $q = 1 - 2^{-k}$. As an example, by OR-ing $k = 3$ LFSR outputs, we get $q = 1 - 2^{-3} = \frac{7}{8} = 0.875$.

4. EXPERIMENTAL RESULTS

We have implemented a number of data-path components using the scheme proposed in Figure 2, and we have used them as basic building blocks for a power-masked cryptoprocessor implementing the RSA algorithm. Details on the achieved results are summarized in the sequel.

4.1 Analysis of Power-Maskable Units

In order to illustrate the benefits in terms of both power masking and power consumption control that the architectural template of Figure 2 may provide, let us consider a 32-bit add-comparator unit. Its traditional implementation (synthesized using Synopsys DesignCompiler onto the $0.18\mu m$ HCMOS8 technology library by STMicroelectronics), when exercised with a given input stream, produces the power consumption profile (cycle-by-cycle power, estimated using Synopsys PowerCompiler) shown in Figure 4, whose average value is 6.05 mW. By applying the automatic precomputation generation technique of [18], we obtain a low-power version of the macro whose average consumption is 4.54 mW, with the cycle-by-cycle power profile of Figure 5. Clearly, power consumption has decreased significantly, but no power masking occurs, as the power profile is fully deterministic and dependent on the input data. By further modifying the architecture of the macro in accordance to the template of Figure 2, for the same input stream, we obtain the power profile depicted in Figure 6, whose average value is 5.42 mW. It is evident that the power profile of the latter implementation is scrambled with respect to that of the macro with deterministic precomputation logic.

The degree of power masking is quantified by the amount of randomness introduced in the profile is related to probability q of the randomizer to mask precomputation (25% in our case). Yet, power consumption s reduced by approximately 10.4% w.r.t. the traditional macro.

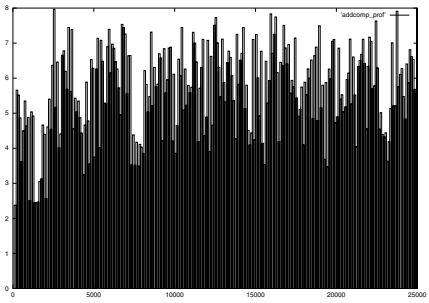

Figure 4: Power Profile of Original Unit.

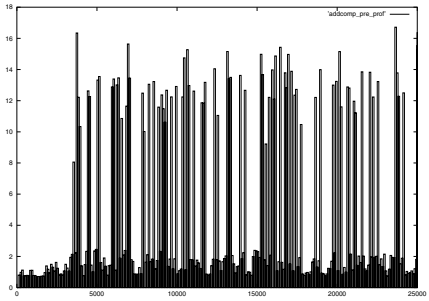

Figure 5: Power Profile of Precomputed Unit.

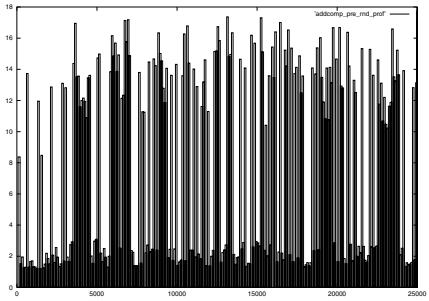

Figure 6: Power Profile of Masked Unit.

4.2 Library of Power-Maskable Units

We built a library of power-maskable data-path units, whose usage is thought for the implementation of secure cryptoprocessors. The library includes a comparator, an add-comparator, an adder, a multiplier and a residue-to-weighted number converter. All units are written in synthesizable VHDL, and have parametric bit width of the I/O operands. Although all implementations fall into the general template of Figure 2, for some units (i.e., comparator, add-comparator and RTW number converter), we resorted to the precomputation schemes of [18], while for the others (i.e., adder and multiplier) we devised specific architectures based on data-range analysis and null-data detection.

The results of the power simulations, performed on different input streams and assuming 32-bit operands, are collected in Table 1. The *Worst* stream is such that precomputation never occurs; thus, power consumption is always increased, but this is obviously not what happens in real usage of the units. Similarly, the *Best* stream is such that precomputation is always active. Thus, power savings are significant. Finally, streams *Real1* and *Real2* represent typical examples of usage of the units. In both cases (and for all units) average power does not increase after the randomizer is inserted.

Unit	*Str*	Orig	preC	ΔP	Rnd	ΔP
AddC	Best	6.09	4.12	-32.35	5.15	-15.44
	Real1	7.19	5.13	-28.65	7.02	-2.36
	Real2	6.05	4.54	-24.96	5.42	-10.41
	Worst	5.51	11.61	110.71	12.74	131.22
R2W	Best	5.76	3.97	-31.08	4.21	-26.91
	Real1	5.31	4.22	-20.53	4.29	-19.21
	Real2	6.72	6.41	-4.61	6.57	-2.23
	Worst	4.35	4.49	3.22	4.61	5.98
Comp	Best	2.34	1.45	-38.03	2.03	-13.25
	Real1	2.33	1.56	-33.05	2.26	-3.00
	Real2	3.21	2.98	-7.17	3.04	-5.30
	Worst	1.82	3.36	84.62	4.28	135.16
Add	Best	3.24	1.98	-38.89	2.70	-16.67
	Real1	16.19	11.25	-30.51	12.47	-22.98
	Real2	5.88	4.28	-27.21	4.93	-16.16
	Worst	4.19	5.29	26.25	5.92	41.28
Mult	Best	67.95	42.45	-37.53	50.19	-26.14
	Real1	134.29	121.29	-9.68	126.60	-5.73
	Real2	68.20	48.68	-28.62	57.76	-15.31
	Worst	143.34	157.49	9.87	157.89	10.15

Table 1: Power Results for Library of Units.

4.3 Case Study

We applied the proposed power masking approach to the implementation of a typical cryptographic operation that is used in many public key algorithms, namely, *modular exponentiation* (ME). In the RSA algorithm, for example, ME coincides with the actual encryption/decryption step.

ME is the operation that is usually subject to DPA attacks, because during decryption, it directly involves the private key [11, 24].

ME is defined as $M^e \bmod N$; in RSA decryption, M represents the cyphertext, e is the private key, and N is part of the public key generated in an initial phase of the algorithm. Conversely, in the encryption step, M represents the plaintext, while e and N are the public key of the destination entity. In this case, M is represented as an integer in the interval $[0 \dots N-1]$. Notice that the private key is used by ME only in the decryption phase.

A direct hardware implementation of exponentiation is usually deemed as impractical. For this reason, several efficient implementations of ME have been proposed by researchers. Since efficiency is not the main objective of our analysis, we have chosen a relatively simple implementation of ME, namely the *square-and-multiply* algorithm, that is widely used in many cryptosystems [24]. The algorithm explicitly uses multiplication (2), division (2), addition (1) and comparison (1). Therefore, many of these data-path units can be implemented using the components taken from our library of power-maskable units.

The power profiles of the three implementations of the ME unit (traditional, precomputed and randomized) are shown in Figures 7, 8 and 9. The increased degree of randomness derived by the instantiation of the power maskable units is evident in Figure 9 (it corresponds to a probability $q = 0.25$ used for all randomized units). The average power consumption of the traditional implementation is of 351.1 mW, that of the precomputed implementation is 339.7 mW and the power of the randomized implementation is 350.9 mW, thus slightly below the original. We can thus conclude that, although the presented results are preliminary, the approach introduced in this paper is very promising.

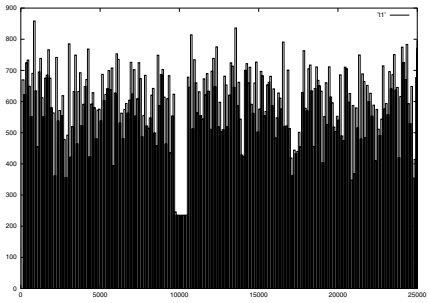

Figure 7: Power Profile of ME Unit.

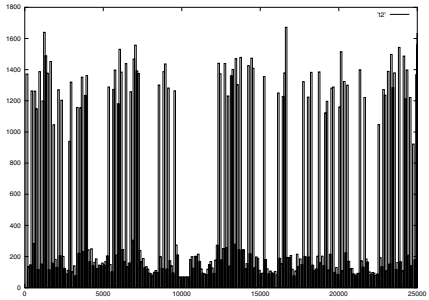

Figure 8: Power Profile of Precomputed ME Unit.

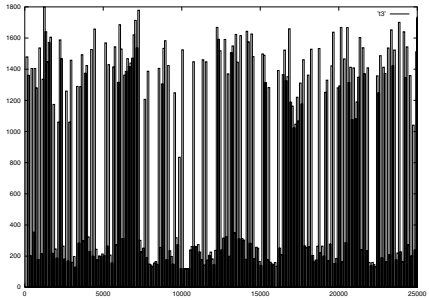

Figure 9: Power Profile of Masked ME Unit.

5. CONCLUSIONS

We have presented a novel design technique aimed at protecting cryptoprocessors against attacks based on power profile analysis. In particular, we have proposed the concept of randomized power masking, which is based on a combination of power-management techniques and randomized clock gating. End result of the application of this new design paradigm is the introduction into the cryptoprocessor of a significant amount of non-deterministic scrambling in the power profile, which prevents external attacks at no cost in the circuit average power consumption.

Acknowledgments

This work was supported, in part, by the European Commission, under grants IST-2001-30125 "POET" and IST-2001-30093 "EASY".

6. REFERENCES

[1] A. Menezes, P. van Oorschot, S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1998.

[2] B. Schneier, *Applied Cryptography*, II Edition, Wiley, 1996.

[3] R. Anderson, *Security Engineering: A Guide for Building Dependable Distributed Systems*, Wiley, 2001.

[4] D. Yuliang, M. Zhigang, W. Tao, "Implementation of RSA Cryptoprocessor based on Montgomery Algorithm," *IEEE International Solid-State Circuits Conference*, pp. 254-256, Jan. 1998.

[5] J. Goodman, A. Chandrakasan, "An Energy-Efficient Reconfigurable Public-Key Cryptography Processor," *IEEE Journal of Solid-State Circuits*, vol. 36, no. 11, pp. 1808-1820, Nov. 2001.

[6] L. Shuguo, Z. Runde, G. Yuanging, "A 1024-bit RSA Crypto-Coprocessor for Smart Cards," *IEEE International Conference on ASICs*, pp. 352-355, Sept. 2001.

[7] O. Koemmerling, M. Kuhn, "Design Principles for Tamper-Resistant Smart Card Processors," *USENIX Workshop on Smart Card Technology*, pp. 9-20, May 1999.

[8] P. Kocher,J. Ja, B. Jun, "Differential Power Analysis," *CRYPTO 99: Advances in Cryptology,,* Springer-Verlag, pp. 388-397, 1999.

[9] A. Shamir, "Protecting Smart Cards from Passive Power Analysis with Detached Power Supplies," *CHES-00: International Workshop on Cryptographic Hardware and Embedded Systems* Springer-Verlag, pp. 71-77, 2000.

[10] P. Rakers, L. Connell, T. Collins, D. Russel, "Secure Contactless Smart Card ASIC with DPA Protection," *IEEE Journal of Solid-State Circuits*, vol. 36, no.3, pp. 559-565, March 2001.

[11] T. Messerges, E. Dabbish, R. Sloan, "Examining Smart-Card security under the thread of power analysis attacks," *IEEE Transactions on Computers*, Vol. 51, no. 5, pp. 541-552, 2002.

[12] S. Moore, R. Anderson, M. Kuhn, "Improving Smart Card Security using Self-Timed Circuit Technology," *IEEE International Symposium on Ansychnronous Circuits and Systems*, pp. 120-126, 2002.

[13] H. Saputra, N. Vijaykrishnan, M. Kandemir, M. J. Irwin, R. Brooks, S. Kim, W. Zhang, "Masking the Energy Behavior of DES Encryption," *DATE-03: IEEE Design Automation and Test in Europe*, pp. 84-89, 2003.

[14] P. Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS and Other Systems," *CRYPTO-96: Advances in Cryptology*, Springer-Verlag, pp. 104-113, 1996.

[15] S. Chari, C. Jutla, J. Rao, P. Rohatgi, "Towards Sound Approaches to Counteract Power-Analysis Attacks," *CRYPTO-99: Advances in Cryptology*, Springer-Verlag, pp. 398-412, 1999.

[16] M. Hasan, "Power Analysis Attacks and Algorithmic Approaches to their Countermeasures for Koblitz Curve Cryptosystems," *IEEE Transactions on Computers*, Vol. 50, no. 10, pp. 1071-1083, Oct. 2001.

[17] J. Golic, C. Tymen, "Multiplicative Masking and Power Analysis of AES," *CHES-02: International Workshop on Cryptographic Hardware and Embedded Systems*, Springer-Verlag, pp. 198-212, 2002.

[18] M. Alidina, J. Monteiro, S. Devadas, A. Ghosh, M. Papaefthymiou, "Precomputation-based Sequential Logic Optimization for Low Power," *IEEE Transactions on VLSI Systems*, Vol. 2, no. 4, pp. 426-436, Dec. 1994.

[19] V. Tiwari, S. Malik, P. Ashar, "Guarded Evaluation: Pushing Power Management to Logic Synthesis/Design," *IEEE Transactions on CAD*, Vol. 17, no. 10, pp. 1051-1060, Oct. 1998.

[20] L. Benini, G. De Micheli, A. Lioy, E. Macii, G. Odasso, M. Poncino, "Synthesis of Power-Managed Sequential Components Based on Computational Kernel Extraction,", *IEEE Transactions on CAD*, Vol. 20, no. 9, pp. 1118-113, Sep. 2001.

[21] L. Benini, G. De Micheli, E. Macii, *M. Poncino*, R. Scarsi, "Symbolic Synthesis of Clock-Gating Logic for Power Optimization of Synchronous Controllers," *ACM Transactions on Design Automation of Electronic Systems*, Vol. 4, No. 4, pp. 351-375, Oct. 1999.

[22] G. Lakshminarayana, A. Raghunathan, K. S. Khouri, N. K. Jha, S. Dey, "Common-Case Computation: A High-Level Technique for Power and Performance Optimization," *DAC-36: ACM/IEEE Design Automation Conference*, pp. 56-61, June 1999.

[23] L. Benini, G. De Micheli, *Dynamic Power Management of Electronic Systems*, Kluwer Academic Publishers, 1998.

[24] T. S. Messerges, E. A. Dabbish, R. H. Sloan, "Power Analysis Attacks of Modular Exponentiation in Smartcards," *CHES-99: International Workshop on Cryptographic Hardware and Embedded Systems* Springer-Verlag, pp. 144-157, 1999.