Energy-Efficient Data Scrambling on Memory-Processor Interfaces

Luca Benini† Angelo Galati * Alberto Macii *

Enrico Macii * Massimo Poncino ‡

† Università di Bologna
Bologna, ITALY 40136

*Politecnico di Torino
Torino, ITALY 10129

‡ Università di Verona
Verona, ITALY 37134

ABSTRACT

Crypto-processors are prone to security attacks based on the observation of their power consumption profile. We propose new techniques for increasing the non-determinism of such profile, which rely on the idea of introducing randomness in the bus data transfers. This is achieved by combining data scrambling with energy-efficient bus encoding, thus providing high information protection at no energy cost. Results on a set of bus traces originated by real-life applications demonstrate the applicability of the proposed solution.

Categories and Subject Descriptors

B.4 [**Hardware**]: Input/Output and Data Communication; E.3 [**Data**]: Data Encryption

Keywords

Bus Encoding, Data Scrambling, Power Attacks

General Terms

Algorithms, Design, Security

1. INTRODUCTION

One of the consequences of the pervasiveness of electronic devices in everyday's life is that an increasing amount of confidential information is electronically processed, stored and communicated. Protection of confidential data is thus becoming a serious concern for many electronic systems. Smart cards, in particular, are critical devices from the security viewpoint, because they often contain "critical" data (e.g., credit card PINs) and because their physical size and weight prevent them from being under continuous surveillance.

Modern smart cards offer many dedicated features for implementing native security measures. For example the P9CC160 smartcard by Philips [1] contains a 32-bit RISC CPU, several storage resources (ROM, FLASH, SRAM, caches), a MMU, a crypto-engine, a separate DES/AES unit, a random number generator, and Java support. All these hardware and software resources can be used to implement cryptographic protocols. Unfortunately, the strength of these protection capabilities seems to be often overestimated: A large number of smart card processors have been successfully reverse-engineered.

A very successful class of attacks is based on the observation of *side channel information*. The basic strategy is to unintrusively monitor various physical quantities (side channels) during encryption and decryption, and try to extract from them information such as secret keys. In particular, power analysis (PA) techniques, which monitor the absorbed current over time, have been quite successful in breaking industry-standard cryptographic algorithms (e.g., DES, RSA, AES) in smart cards [2]. Among the various countermeasures to PA attacks, hardware-supported techniques based on the addition of random noise to deterministic encryption-decryption computation have proven quite effective [4, 3].

Decreasing the determinism of a computation through noise insertion, however, has the undesirable drawback of increasing the energy consumption required by that computation. This issue can be particularly critical for devices like smart cards, where power is supplied only through external devices (e.g., card readers).

Recently, there has been some research effort in trying to combine noise insertion with energy-efficient design solutions. The work by Benini et al. [5] focused on the design of arithmetic units typically employed by cryptographic algorithms (e.g., multipliers, modular operations, etc.).

Protecting computation against power analysis addresses only one facet of a complex problem. On-chip buses are one of the most critical sources of side channel information: Switching heavily loaded bus lines causes significant peaks in the current profile, which leak information on the transmitted data.

In this work, we focus on protecting information exchange on the data bus interface. Our technique combines a protection mechanism based on randomization with energy-efficient bus encoding techniques, to achieve a reversible scrambling of the information transmitted. We propose several high-security, low-power encoding and decoding circuits, tuned for various power constraints. Preliminary results show that a high degree of protection of the bus information can be achieved while keeping power consumption below a given budget.

ISLPED'03, August 25–27, 2003, Seoul, Korea.

2. DPA AND COUNTERMEASURES

Power analysis cryptographic attacks are based on the idea that the power consumption over time of a module can be used to recover secret information (e.g., keys) used during the computation. Power analysis attacks have been first introduced by Kocher et al. [6], and can be categorized into two classes. The first, Simple Power Analysis (SPA), consists of the direct analysis of the monitored power data to determine actions and data, by correlating the time-domain supply current profile with the steps of the algorithm. If the algorithm has a key dependent execution flow (e.g., if its branching behavior depends on the key value), the current waveform can reveal key information.

Differential Power Analysis (DPA) [2], is a much more dangerous attack that combines power measurements with statistical and signal processing techniques on a large number of power consumption traces, to reduce noise and strengthen the differential signal so that it is easier to distinguish between logical values (0/1). DPA can be successful even when the execution flow is not data-dependent. For a detailed discussion of the technical details of DPA the reader is referred to [2].

The key assumption in DPA is that the supply current 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 [8]. This approach, also known as *whitening*, is pursued in [3, 4]. Unfortunately, it is extremely difficult to guarantee perfect whitening, and high-resolution, high-accuracy DPA defeats equalization attempts.

Most recently proposed DPA countermeasures take an alternative approach, namely *random noise insertion* [9]. Randomization techniques range from randomized masking of the secret key, to clock noise insertion, to embedding the secret key into a much larger random key [9]. DPA can in principle cancel randomization noise by averaging over a larger number of traces. However, it has been shown that the number of traces to be averaged increases rapidly with the noise level [9]. For this reason, randomization appears to be more generally accepted as DPA countermeasure than equalization.

System buses are one of the most severe sources of DPA vulnerability in cryptography hardware, because switching of their large capacitive loads at a high rate creates significant current spikes on the power supply. Obviously, these peaks are data-dependent. Hence, an attacker can gather a large amount of information on the data transmitted on a bus by just sampling the power supply. Randomization on bus communication is therefore highly desirable as a power analysis countermeasure. Unfortunately, noise injection and whitening approaches proposed in the past to protect buses have always implied a hardware redundancy and power overhead. For instance, several authors [3, 4] have proposed dual-rail bus encoding, characterized by an equal number of raising and falling transitions for any bus value, to whiten the bus power profile.

Our approach aims at reducing the power overhead in data protection. It is based on the simple idea of transmitting scrambled data on the bus in a reversible way. The power cost of scrambling and de-scrambling is reduced by low-power encoding and conditional de-activation of the scrambling hardware (if data protection is not needed for 100% of the bus operation time).

3. BUS SCRAMBLING

3.1 Pure Scrambling

In the following, D will denote the *plain data*, as sent by the processor on the bus. The scrambling process transforms D into a new value D_S by applying an injective (i.e., one-to-one) function f to D (i.e., $D_S = f(D)$). The inverse process is done at the receiver's end.

The first scrambling scheme we consider (Figure 1-(a)) is similar to those used for data whitening in wireless standards, such as IEEE 802.15 or Bluetooth. The scrambler consists of two main blocks: A random pattern generator (block *Randomizer*, e.g., a LFSR), that produces a new pattern r at each new bus cycle, and the block "$\oplus$", which performs modulo-2 addition (i.e., bitwise XOR) between the D and r. Let us denote the scrambled word by $D_r = D \oplus r$. In this scheme, called *pure scrambling* (PS), $D_S \equiv D_r$. PS will be used as a reference for comparing other schemes. Correct decoding is guaranteed by keeping sender's and receiver's clocks synchronized, and by using the same initial seed for the two LFSRs.

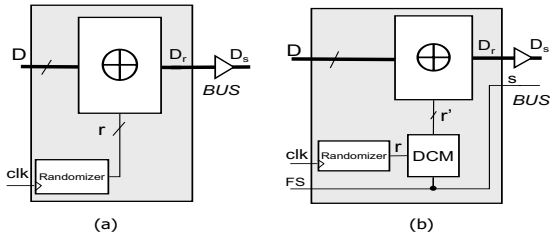

Figure 1: Pure (a) and Conditional (b) Scrambling.

Pure scrambling has the usual drawback of all randomization techniques: The noise added to scramble the sequence will increase energy consumption. This increase in switching depends on the statistical property of the stream D. For close-to-random distributions, the modulo-2 addition with a random word will not generally change the amount of switching. However, as soon as the source exhibits some correlation between symbols, scrambling will increase the switching. As an example, we have analyzed a sequence of integer samples used for a FFT computation over a 32-bit bus. The original, unscrambled sequence yields 18087 transitions; after scrambling, the number of transitions grows to 38375 (a 114% increase).

3.2 Conditional Scrambling

Pure scrambling tends to yield random sequences, and assumes that scrambling is applied to any transmitted pattern. This may be unavoidable when the bus is dedicated to the transmission of critical data. However, in most cases, buses will transmit a mix of critical and non-critical information. For example, a cryptographic computation may consist of an initial phase in which instructions and plain data are fetched from memory, followed by a second phase in which the encryption keys are fetched from another memory. In this case, scrambling could be applied only to the second phase. To allow this, we introduce a new scheme, in which the scrambling can be conditionally activated (Figure 1-(b)). We call this scheme *conditional scrambling* (CS).

In CS, the randomization is controlled by an external signal `FS` (Forced Scrambling). The block labeled *DCM* (Duty Cycle Modulator) works as a switch driven by `FS` that, at

each cycle, determines what operand r' must be supplied to the modulo-2 adder. The DCM implements the following function:

if (FS==1) r' = r **else** r'=0;

Given the non-periodic nature of FS, this scheme requires an additional bus line s that signals to the decoder what word (D or $D_r = D \oplus r'$) has been transmitted.

The FS signal can also be used for trading off energy consumption for the degree of data protection. For instance, even when transmitting critical data, we can decide not to apply scrambling, in order to keep energy consumption within some assigned budget.

The energy efficiency of CS is related again to the degree of randomness of the transmitted data. Consider, for example, the sequence of FFT samples described in the previous section. By forcing FS to be activated at a fixed rate, the results show that even a moderate amount of scrambling (e.g., 10%) already causes a significant increase in the number of transitions (21383).

4. ENERGY-EFFICIENT SCRAMBLING

To solve the energy increase problem, we resort to a more complex scrambling scheme, which exploits the existing correlation to realize some kind of low-switching encoding of the data; for instance, by replacing the modulo-2 addition with a suitable encoder.

When the protection of the transmitted data is the primary target, however, this is not the best choice. Such scheme, in fact, would result in the transmission of either the plain data, or, at random times, the encoded ones. Under the assumption that the attacker has full knowledge of the underlying hardware (and thus, also of the used encoding scheme), it is clear that extracting the plain data becomes a relatively simple task: By observing the sequence D_S, and by knowing the encoding function used, re-constructing the sequence D simply implies to determine whether the encoding has been applied or not.

We can conclude that *randomizing the application* of an encoding function is not as effective as *randomizing the encoding* itself. In Figure 1-(b), the randomness is intrinsic to the encoding function, and even the knowledge of the hardware implementation (the modulo-2 sum) does not provide enough information. The solutions proposed in this section leverage the basic scheme of Figure 1-(b) and *combine* it with low-power bus encoding techniques.

4.1 Scrambling and Bus Encoding

The choice of what encoding scheme to use is driven by (i) its complexity (performance and energy of the encoding circuitry), and (ii) its applicability to generic bus streams (in particular, its effectiveness on streams with close-to-random distributions).

In fact, regardless of the existing correlation, the candidate encoding scheme should be effective on non-correlated streams, since the transmission of a random value at random times will destroy any existing correlation in the original stream. Therefore, we regard the actual data sent on the bus (after scrambling) as random patterns. Among the schemes that satisfy the two above requirements, we have chosen the Bus-Invert encoding [11]. In Bus-Invert, the current word is compared with the previously transmitted one. If the Hamming distance between the two exceeds $N/2$ (for N bus lines) the complement of the word is sent on the bus. This choice is communicated to the receiver through an extra bus line.

4.2 Bus-Inverted Pure Scrambling

The most straightforward combination of scrambling and Bus-Invert (BI) consists of cascading the scrambler block of Figure 1-(a) and a BI encoder, as shown in Figure 2-(a). We call this scheme *PS+BI*. At every cycle, the transmitted word D_S is either D_r or $\overline{D_r}$ (i.e., its complement), depending on what word will minimize bus switching.

Based on the same arguments of Section 3, we expect the PS+BI scheme to perform well in the case of highly uncorrelated streams, and, in particular, better than PS. In fact, the same efficiency of Bus-Invert is guaranteed: The resulting encoded stream will have no more than $N/2$ (for N bus lines) transitions for each pair of consecutive patterns.

In the case of streams with some degree of correlation, however, the whitening effect of scrambling may possibly cancel the benefits obtained by applying BI. Finally, with respect to the non-redundant scheme of Figure 1-(a), the bus now includes an extra line INV that signals whether D_r or $\overline{D_r}$ is transmitted.

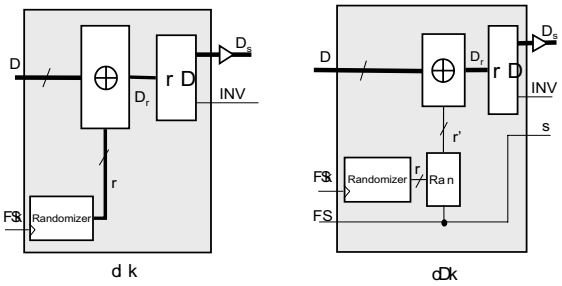

Figure 2: Pure Scrambling (a) and Conditional Scrambling (b) with Bus-Invert.

4.3 Bus-Inverted Conditional Scrambling

A more flexible scheme consists of the combination of CS and Bus-Invert (CS+BI). As for CS, signal FS determines whether to apply scrambling or not. Whatever the decision, BI is always applied to the pattern (Figure 2-(b)).

We expect CS+BI to be more effective than PS+BI, when scrambling is applied selectively. In such cases, in the cycles in which FS is kept low, the encoding becomes a plain BI. This is not necessarily more energy-efficient than plain CS, because the effectiveness of BI depends on the correlation of the target stream D. It has been shown [11] that BI tends to become virtually useless when applied to streams with significant correlation. In the remaining cycles, FS is kept high and scrambling is activated, and CS+BI is equivalent to PS+BI.

Notice that now two extra lines are required: One to signal the occurrence of scrambling s, and another one to signal the transmission of an inverted pattern INV.

5. EXPERIMENTAL RESULTS

5.1 Random Streams

By random streams we mean sequences of values, such as cryptographic keys, that require some form of protection at all times. In this case, since scrambling must always be active, conditional schemes are not of interest.

We have generated two streams (*RAND1* and *RAND2*), of equal length (10^5 patterns), using the API described in [12]. Table 1 shows the results for the two streams when applying PS and PS+BI. Values are relative to a 32-bit, 1-mm long bus, in a 0.18μm, 1.8V technology by STM.

The values in the table only consider bus energy, and do not include the cost of the encoder. Column E reports the total energy of the original stream, in nJ, while columns PS and PS+BI report energy and percentage savings with respect to the unscrambled stream.

Stream	E [nJ]	PS		PS+BI	
		E [nJ]	Δ [%]	E [nJ]	Δ [%]
RAND1	7317	7315	0.1	6756	7.7
RAND2	7329	7315	0.2	6759	7.8

Table 1: Results for Random Streams.

We notice that even in the case of PS, scrambling does not increase energy. Although the hardware overhead is limited for PS (a battery of 32 XOR gates), the PS encoder in the same technology consumes 195nJ, about 3% of the overall bus energy. Therefore, PS does not allow to do scrambling without, although marginally, increasing the total power. With PS+BI, conversely, although the encoder consumes about twice the PS encoder, the power savings allow to implement a continuous scrambling without any power increase.

5.2 MultiMedia Streams

The second class of streams we consider represent typical data processed by a smartcard such as multimedia files (images, sound files, or movies). Since all standard formats (e.g., MPEG/JPEG) are compressed, the streams exhibit very little correlation, and we expect results similar to the case of random streams.

Results (not reported for space reasons) show that energy savings are only slightly higher than those for random streams: The average savings are 0.8% for PS, and 10.5% for PS+BI. For this type of streams, however, conditional scrambling schemes can be used to trade the degree of scrambling for the energy consumption. Figure 3 plots the energy savings vs. the percentage of *forced scrambling* (i.e., the fraction of cycles in which signal FS in Figure 1-(a) is high). The case of *FS=100%* is equivalent to PS.

The plot shows some interesting facts. First, due to the intrinsic randomness, the energy savings are always below 2%. Second, for benchmarks with very weak, but non-null correlation (e.g., MPEG and AVI), increasing *FS* decreases energy savings. Conversely, WAV is insensitive to *FS* (denoting an almost null correlation). Therefore, CS by itself does not provide any significant benefit over PS, for streams with little or no correlation.

Cascading BI to CS, however, changes the situation, as shown by Figure 4. For this scheme, the energy savings are always greater than 10%, allowing enough slack to accommodate the energy consumption of the encoder. Furthermore, all streams but WAV are now insensitive to the value of *FS*. This behavior can be explained by the fact that Bus-Invert cancels any weak correlation possibly existing in the streams.

In summary, the effectiveness of conditional scrambling depends on the stream, even when randomness is high. From the results, in most cases, PS+BI is as effective as CS+BI.

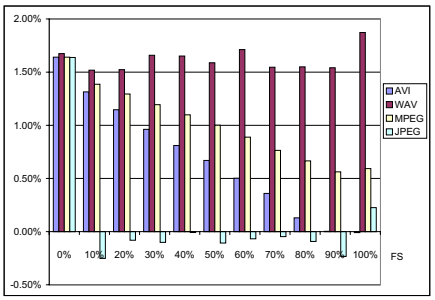

Figure 3: CS Results: MultiMedia Benchmarks.

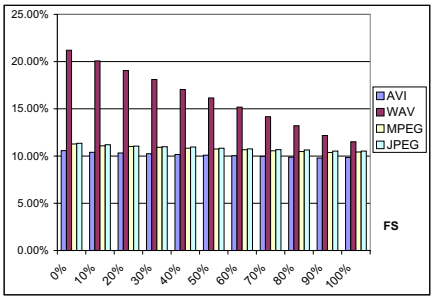

Figure 4: CS+BI Results: MultiMedia Benchmarks.

6. CONCLUSIONS

We have proposed a set of techniques to enhance the security of information transmitted on a communication bus, by combining data scrambling with energy-efficient bus encoding, with different trade-offs between the achieved degree of security and energy consumption. Preliminary results show that a high degree of protection of the data transferred on the bus can be achieved at no power cost.

7. REFERENCES

[1] Philips Semiconductors, HiPerSmart Computing Platform, *http://www.semiconductors.philips.com/markets/identification/products/hipersmart/*

[2] P. Kocher, J. Ja, B. Jun, "Differential Power Analysis," *CRYPTO'99*, Springer-Verlag, pp. 388-397, 1999.

[3] N. Vijaykrishnan, M. Kandemir, M. Irwin, "Masking the Energy Behavior of DES Encryption," *DATE'03*, Munich, Germany, 2003, pp. 84–89.

[4] S. Moore, R. Anderson, M. Kuhn, "Improving Smart Card Security using Self-Timed Circuit Technology," *IEEE Int. Symposium on Asynchnronous Circuits*, Manchester , UK, pp. 120-126, 2002.

[5] L. Benini, A. Macii, E. Macii, E. Omerbegovic, M. Poncino, F. Pro, "Design of Power Maskable Units for Cryptographic Applications," *DAC-40*, Anaheim (CA) 2003.

[6] P. Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS and Other Systems," *CRYPTO96*, Springer-Verlag, pp. 104-113, 1996.

[7] 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.

[8] T. Lash, *A Study of Power Analysis and the Advanced Encryption Standard*, George Mason University, MS Scolarly Paper, 2002.

[9] S. Chari, C. Jutla, J. Rao, P. Rohatgi, "Towards Sound Approaches to Counteract Power-Analysis Attacks," *CRYPTO99,* Springer-Verlag, pp. 398-412, 1999.

[10] S. Rankl and W. Effing. *Smart Card Handbook*, John Wiley & Sons, 1999.

[11] M. Stan, W. Burleson, "Bus-Invert Coding for Low-Power I/O," *IEEE Transactions on VLSI Systems*, Vol. 3, No. 1, pp. 49-58, 1995.

[12] S. Park, K. Miller, "Random Number Generators: Good Ones Are Hard To Find" *Communications of the ACM*, Vol. 31, No. 10, pp. 1192-1201, 1988.