Signal Processing for Cryptography and Security Applications

Miroslav Knežević, Lejla Batina, Elke De Mulder, Junfeng Fan,
Benedikt Gierlichs, Yong Ki Lee, Roel Maes and Ingrid Verbauwhede

Abstract Embedded devices need both an efficient and a secure implementation of cryptographic primitives. In this chapter we show how common signal processing techniques are used in order to achieve both objectives. Regarding efficiency, we first give an example of accelerating hash function primitives using the retiming transformation, a well known technique to improve signal processing applications. Second, we outline the use of some special features of DSP processors and techniques earlier developed for efficient implementations of public-key algorithms. Regarding the secure implementations we outline the concept of side channel attacks and show how a multitude of techniques for preprocessing the data are used in such scenarios. Finally, we talk about fuzzy secrets and point out the use of DSP techniques for an important role in cryptography — a key derivation.

1 Introduction

The implementation of cryptographic and security applications has adapted techniques more than expected from the architectures, design methods and tools of signal processing systems. When implementing cryptographic applications on embedded devices, in hardware, software or a combination of both, there are two opti-

Miroslav Knežević
NXP Semiconductors, Leuven, Belgium, e-mail: miroslav.knezevic@nxp.com

Lejla Batina, Elke De Mulder, Junfeng Fan, Benedikt Gierlichs, Roel Maes, Ingrid Verbauwhede
Katholieke Universiteit Leuven, ESAT/COSIC and IBBT, Belgium, e-mail: {lejla.batina, elke.demulder, junfeng.fan, benedikt.gierlichs, roel.maes, ingrid.verbauwhede}@esat.kuleuven.be

Lejla Batina
Radboud University Nijmegen, ICIS/Digital Security Group, e-mail: lejla@cs.ru.nl

Yong Ki Lee and Ingrid Verbauwhede
University of California, Los Angeles, Electrical Engineering, e-mail: jfirst@ee.ucla.edu

mization goals. The first request is one of efficiency: the implementations should be efficient in area, time, power consumption etc. This is very similar to the implementation of signal processing applications. The other requirement is one of a secure implementation. Implementations should not leak information. Signal processing techniques are used for both. As research progresses, signal processing and cryptography find more overlaps.

In this chapter, an overview is given of the use of signal processing techniques as they are used for efficient implementations as well as used for the analysis of secure implementations. The techniques that are used for efficient implementations in particular, are discussed in more detail in [57] and [32].

Section 2 gives an overview of efficient implementations for both secret-key and public-key. It describes techniques borrowed from fast filter implementations to optimize the implementation of hash functions. It also describes implementations of public-key on DSP processors. Section 3 focuses on secure implementations and the signal processing techniques used in analysis. Section 4 discusses the usage of signal processing techniques for fuzzy secrets. Section 5 gives the conclusion and describes some new trends.

2 Efficient Implementation

2.1 Secret-Key Algorithms and Implementations

2.1.1 Cryptographic Hash Functions and Implementations

A cryptographic hash function is a deterministic algorithm that takes input strings — M of arbitrary length and returns short fixed-length strings, so called message digests — Hash(M), and it is required to satisfy the following cryptographic properties:

1. *Preimage resistance*: It must be infeasible to find any preimage, M, for a given hash output, H, such that H=$Hash$(M).
2. *Second Preimage resistance*: It must be infeasible to find another preimage, M_1, for a given input, M_0, such that $Hash(M_0)$=$Hash(M_1)$.
3. *Collision resistance*: It must be infeasible to find two different inputs of the same hash output, M_0 and M_1, such that $Hash(M_0)$=$Hash(M_1)$.

To be useful in applications such as Digital Signature Algorithm and message authentications, the performance becomes an important factor since inputs are easily large. To obtain architectures with a better performance, a design methodology is developed in [43] based on theoretically proven signal processing techniques [58]. Here we illustrate this methodology with the example of MD6, which was one of

the SHA-3 candidates[1] [13]. Parhi and Chen [57] provide more detailed analysis of the common DSP techniques used in this example.

First, we derive a DFG (Data Flow Graph) of MD6 as shown in Fig. 1. This DFG represents one iteration of the algorithm, where the square nodes are registers and each register is paired with an *algorithmic delay* (D) so that the register values can be updated on each cycle. The circle nodes are functional nodes where $\wedge$ denotes the bitwise "AND" operator, $\oplus$: the bitwise "XOR" operator, $\ll b$: left-shifted by b bits (zero shifting in), and $\gg b$: right-shifted by b bits (zero shifting in). S_i is constant, but changes from round to round. Each round is composed of 13 iterations and the number of rounds depends on the algorithm parameters.

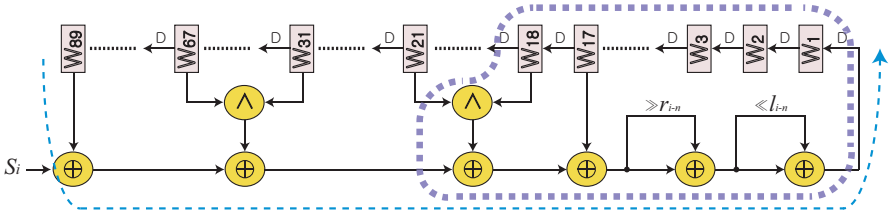

Fig. 1 MD6 DFG — Critical Path

The critical path delay can be measured as the maximum delay among any two consecutive algorithmic delays (D's), which for MD6 consists of 6 XOR delays as shown in Fig. 1 (indicated by an arrow with a dotted line). Before applying some techniques to minimize the critical path delay, we first need to know how much we can achieve. This can be done by analyzing the iteration bound of a DFG, which defines a theoretical lower bound of the achievable critical path delay as defined by the following equation:

$$T_\infty = \max_{l \in L} \left\{ \frac{t_l}{w_l} \right\} = \frac{4 \times Delay(\oplus) + Delay(\wedge)}{18} \tag{1}$$

where t_l is the loop calculation time, w_l is the number of algorithmic delays in the l-th loop, and L is the set of all possible loops. The iteration bound of MD6 is defined with the loop indicated by a thick dotted line in Fig. 1.

In order to achieve a critical path delay as the iteration bound, it requires the unfolding transformation with the factor of the un-canceled denominator, which is 18 in this case. This introduces an overhead by duplicating the functional nodes (18 times duplication). Therefore, we choose not to perform the unfolding transformation but perform retiming transformation only. The achievable critical path delay only with the retiming transformation can be defined as follows:

[1] SHA-3 competition, organized by the National Institute of Standards and Technology (NIST), is a worldwide competition for the development of a new hash standard.

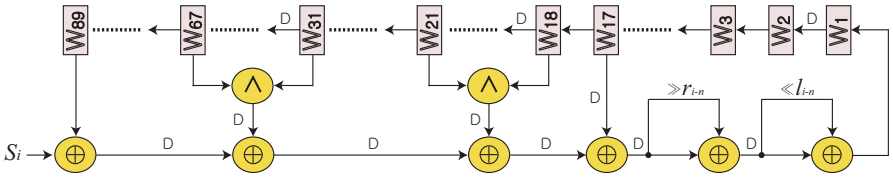

Fig. 2 MD6 DFG — Retiming Transformation

$$\lceil T_\infty \rceil = \left\lceil \frac{4 \times Delay(\oplus) + Delay(\wedge)}{18} \right\rceil = Prop(\oplus) \tag{2}$$

$\lceil \cdot \rceil$ is the maximum part when $4 \times Delay(\oplus) + Delay(\wedge)$ is evenly distributed into 18 parts. Since it is assumed that a functional node can not be split into multiple parts in the iteration bound analysis, $\oplus$ cannot be further split. By retiming transformations, we replace the algorithmic delays (D's) in order to reduce the critical path delay. Some of the algorithmic delays between registers (including those in the abbreviated area) can be moved to proper positions as shown in Fig. 2 where the new critical path delay is $Prop(\oplus)$. The details about retiming transformations and implementation issues can be found in [43] [58]. It shows that techniques which were originally developed for improving the throughput of digital filters with feedback can be also applied to hash function implementations. For more information about hardware design for cryptographic hash function we refer the reader to [44].

In the light of the running SHA-3 competition, an extensive hardware evaluation of the competing algorithms has been performed by the research community. One of the examples is the work of Sharif et al. [64], which specifically explores the influence of the embedded FPGA resources (such as DSP units) on hardware performance of the five SHA-3 finalists. Due to the presence of addition operation in their compression functions, two out of five candidates as well as the current SHA-2 standard benefit from using the DSP units. Since addition is a relatively simple operation, the benefit is more reflected in reducing the usage of other resources rather than increasing the overall throughput.

2.2 Public-Key Algorithms and Implementations

2.2.1 Efficient Modular Multiplication

Public-key cryptography (PKC), a concept introduced by Diffie and Hellman [15] in the mid 70's, has gained its popularity together with the rapid evolution of today's digital communication systems. The best-known public-key cryptosystems are based on factoring *i.e.* RSA [62] and on the discrete logarithm problem in a large prime field (Diffie-Hellman, ElGamal, Schnorr, DSA) [49] or on an elliptic/hyperelliptic curve (ECC/HECC) [39] [52] [40]. Based on the hardness of the underlying mathematical problem, PKC usually deals with large numbers ranging

from a few hundreds to a few thousands of bits in size. Consequently, the efficient implementations of the PKC primitives has always been a challenge.

An efficient implementation of the mentioned cryptosystems highly depends on the efficient implementation of modular arithmetic. Namely, modular multiplication forms the basis of modular exponentiation which is the core operation of the RSA cryptosystem. It is also present in many other cryptographic algorithms including those based on ECC and HECC. In particular, if one uses projective coordinates for ECC/HECC, modular multiplication remains the most time consuming operation for ECC. Hence, an efficient implementation of PKC relies on efficient modular multiplication.

Two algorithms for modular multiplication, namely Barrett [4] and Montgomery [53] algorithms are widely used today. Both algorithms avoid multiple-precision divisions, the operation that is considered to be expensive, especially in hardware. Implemented on a first generation of DSP produced by Texas Instruments (TMS 32010), the Barrett's algorithm is one of the famous examples in the area of signal processing for cryptography. The properties of a DSP such as fast multiply and accumulate (MAC) operation and a fast microprocessor on a single chip seemed to be an ideal combination for the proposed algorithm. Furthermore, additional speed-ups were obtained by taking advantage of a feature of the TMS320 DSP which allowed auto increment and decrement of data pointers during MAC operations. This ensured the data fetching for free. The original Barrett reduction algorithms is given in Alg. 1.

Algorithm 1 Barrett modular reduction.

Input: $A = (A_{2n-1} \dots A_0)_2$, $M = (M_{n-1} \dots M_0)_2$ where $M_{n-1} \neq 0$, $\mu = \lfloor 2^{2n}/M \rfloor$.
Output: $R = A \bmod M$.

$\hat{Q} \Leftarrow \left\lfloor \frac{\lfloor \frac{A}{2^{n-1}} \rfloor \mu}{2^{n+1}} \right\rfloor$
$R \Leftarrow A \bmod 2^{n+1} - \hat{Q}M \bmod 2^{n+1}$
while $R \geq M$ **do**
  $R \Leftarrow R - M$
end while
return R.

Montgomery's algorithm is the most commonly used reduction algorithm, nowadays. It is widely used to speed up the modular multiplications and squarings required during the exponentiation process. The Montgomery reduction algorithm computes $T = A \cdot r^{-1} \bmod M$, given $A < M^2$ and r such that $\gcd(r, M) = 1$. Even though the algorithm works with any r, relatively prime to M, r is very often chosen to be a power of 2. In that case, the Montgomery algorithm performs integer divisions by a power of 2, which is an intrinsically fast operation on both general-purpose and specialized processors. The algorithm is given in Alg. 2.

There is a number of publications reporting the implementations of Montgomery's algorithm on a DSP. One of the first reported results was a work of Michael Wiener who had developed a general software implementation of RSA on the Mo-

Algorithm 2 Montgomery modular reduction.

Input: $A = (A_{2n-1} \dots A_0)_2, M = (M_{n-1} \dots M_0)_2, r = 2^n$ where $\gcd(M,2) = 1, M' = -M^{-1} \bmod r$.
Output: $T = Ar^{-1} \bmod M$.
$S \Leftarrow (A \bmod r)M' \bmod r$
$T \Leftarrow (A + SM)/r$
if $T \geq M$ **then**
$T \Leftarrow T - M$
end if
return T.

torola DSP56000 that achieved 10.2 Kbps for 512-bit modular exponentiation with the Chinese remainder theorem (CRT) [7]. Dusse and Kaliski have published an RSA implementation on the same chip achieving 11.6 Kbps for 512-bit modular exponentiation with the CRT [17]. They adopted the Montgomery's algorithm by reducing the number of shift operations. Since on many processors the "high part" and the "low part" of accumulators are separately addressable, the shift operations have to be performed with move instructions. This was also true for the DSP56000 and the idea of reducing the number of shifts turned out to be a very good approach. Itoh et al. proposed a fast implementation method of Montgomery multiplication on a DSP TMS320C6201 [35]. This DSP operates eight function units in parallel achieving a performance of 11.7 ms for 1024-bit RSA signing (87.5 Kbps). In 2004, Großschädl et al. [27] analyzed the performance of Montgomery multiplication on the MIPS32 4Km and the ARM 946E-S processors. In particular, they explored the suitability of the DSP architectural enhancements to speed up multiple-precision modular multiplication. Recently, Gastaldo et al. [22] have published an enhanced Montgomery multiplication algorithm, a variant of the finely integrated product scanning (FIPS) algorithm, that is targeted to digital signal processors. More about various approaches for implementing Montgomery's algorithm on a DSP can be found in [10].

2.2.2 Implementations of Public-Key algorithms

There are several implementations reported in the literature describing efficient PK cryptosystems on DSP architectures, mainly relying on the algorithm of Montgomery. For RSA cryptosystems, modular multiplication is practically the only required operation. As it is commonly implemented as a sequence of repeated squarings and multiplications, some authors explored possible speed-ups that can be achieved for a dedicated squaring. Krishnamurthy et al. proposed a new method for the Montgomery squaring reduction especially suited for DSP processors. The new method restructures the loop in the squaring reduction and when implemented on the TI TMS320C6201 DSP platform results in speed-ups of 10-15 % compared to the previous work in [35]. The results are obtained by removing the redundancy of previous proposals and by maximizing the feature of pipelining as much as the platform permits.

Guajardo et al. [28] described a strategy for an efficient implementation of ECC over $GF(p)$ on the 16-bit TI MSP430x33x family. This family of devices is commonly used for ultra-low power applications such as electronic meters for gas, water and electricity and also for sensor systems that collect analogue signals. The signal are converted to digital ones before being transmitted to a host. The properties make the platform very suitable for embedded security applications.

The authors use a Generalized-Merssenne prime to facilitate the underlying field arithmetic. The operations i.e. multiplication, squaring and modular reduction are adapted to explore the specifics of the architecture. For example, multiplications with small constants were traded for additions as the ratio of multiplication and addition is around 10 on this processor. Also, squaring was found to be much faster when implemented as a special routine because the communication with data memory was reduced (requiring only one input in this case). The use of a special prime, as suggested by standards, was additionally tailored to the 16-bit architecture. However, all the tricks used were not enough to meet the requirements for the 160-bit security level (as required minimum for ECC key lengths). Therefore, they propose to use 128-bit EC implementations as sufficient for the targeted applications.

We note here that implementing squaring in a different manner from general multiplication is found to be sensitive to side-channel attacks (see Sect. 3). Therefore, the ideas of dedicated squaring are abandoned for embedded applications.

By extensive use of the DSP blocks on Xilinx's Virtex-4 SX55 FPGA board, Güneysu and Paar [31] report the fastest point multiplication on commercially available FPGA platforms. Their implementation performs more than 24,000 and 37,000 point multiplications per second for ECC over the NIST P-256 and P-224 fields, respectively.

Morozov et al. [54] investigate the design space of ECC implementation on Texas Instruments' OMAP 3530 platform (accommodating the ARM Cortex A8 core), specifically considering its DSP core for accelerating the underlying modular arithmetic. A speedup of more than 9 times is achieved, compared to the implementation executing on the ARM Cortex processor only.

2.2.3 Other Ideas for Arithmetic Borrowed from Signal Processing

Cryptographic application as well as signal processing require arithmetic-intensive operations, hence there are several ideas that were applied in both. Examples include Residue Number System (RNS) arithmetic [65], signed-digit arithmetic, computation in frequency domain etc.

Nowadays ever faster arithmetic is demanded e.g. for RSA cryptosystems due to the constant improvements in factoring and the resulting requirements for even longer key sizes. In order to achieve that, the Residue Number System is an alternative to the common radix representation. RNS arithmetic is a very old idea which relies on the Chinese Remainder Theorem (CRT) and it provides a good means for very long integer arithmetic.

Let $\langle x \rangle_a$ denote an RNS representation of x, then:

$$\langle x \rangle_a = (x[a_1], x[a_2], \ldots, x[a_n]) \tag{3}$$

where, $x[a_i] = x \bmod a_i$. The set $a = \{a_1, a_2, \ldots, a_n\}$ is called a base (of size n). It is required that $gcd(a_i, a_j) = 1$ for $i \neq j$. CRT implies that the integer x which satisfies $0 \leq x < \prod_{i=1}^{n} a_i$ is uniquely represented by $\langle x \rangle_a$.

A well known advantage of RNS is that to add, subtract and multiply such numbers we only need to compute the addition, subtraction and multiplication of their components, of size much smaller than the original modulus. Also carry-free arithmetic makes parallelization possible. The final result is obtained by means of the CRT. The disadvantage of an RNS representation is the overhead introduced by the input and output conversions from binary to RNS and vice versa. It is also difficult to perform division. To overcome this disadvantage, a combination with Montgomery multiplication was proposed [60]. Recent results show that RNS Montgomery brings a higher speed for both ECC and Pairing implementations on FPGAs [30] [12].

Exponent recoding techniques for modular exponentiation (as used for RSA) replace the binary representation of an exponent with a representation which has fewer non-zero terms (see Gollmann *et al.* [26]). Many techniques for exponent recoding have been proposed in the literature. Here we mention the signed-digit representation. Consider an integer representation of the form $k = \sum_{i=0}^{l} s_i 2^i$, where $s_i \in \{-1, 0, 1\}$. This is called the (binary) signed digit (SD) representation (see Menezes *et al.* [49]). The representation is redundant. For example, the integer 3 can be represented as $(011)_2$ or $(10\bar{1})_2$, where $\bar{1} = -1$. It is said that an SD representation is *sparse* if it has no adjacent non-zero digits. A sparse SD representation is also called a *non-adjacent form* (NAF). Every integer k has a unique NAF which has the minimum weight of any signed digit representation of k.

For RSA, the NAF techniques are not beneficial because they assume performing the division operation, which should be avoided due to the complexity of the operation. However, for ECC it is considered advantageous because the Hamming weight of the scalar is minimal. Also in this case "-1" means point subtraction instead of addition, which is an addition of the inverse point. More precisely, the scalar is decomposed as a NAF and the scalar multiplication is done with a series of addition/subtractions of elliptic curve points.

Other redundancy-based techniques include on-line arithmetic as proposed by Ercegovac [18]. In particular on-line arithmetic for DSP applications can be found in [19].

ECC implementation in the frequency domain is given by Baktir et al. [2]. The work is based on Discrete Fourier transform (DFT) modular multiplication and ECC processor architecture using the multiplier is presented. In this way multiplication in $GF(q^m)$ is achieved by only a linear number of base field $GF(q)$ multiplications in addition to a quadratic number of simpler base field operations such as additions/subtractions and bitwise rotations.

An idea for implementation of large integer multiplication in the discrete Fourier domain originates from Kalach and David [37].

2.3 Architecture

The architecture design of cryptographic systems requires special considerations on performance, cost and security. Many cryptographic systems, especially public-key cryptosystems, involve complex operations with large integers or long polynomials. Thus, cryptographic co-processors, just like accelerators for multimedia, communication or image processing, are introduced to offload the heavy processes from CPU. As a result, a Hardware/Software (HW/SW) co-design approach is usually used. Both the co-processor and the interface need to be carefully designed such that the co-processor can efficiently offload the work from CPU while security and flexibility are maintained. We illustrate the idea with an Elliptic Curve Processor.

2.3.1 Datapath

Datapath normally accounts for a large percentage of the total area and is a determinative factor of the performance. Taking ECC over GF(p) as an example, each scalar point multiplication involves hundreds of modular multiplications. As a result, speeding up the modular multiplication is the key to improve the overall performance. The multiplier can be implemented in a bit-parallel, digit-parallel or digit-serial way. When using digit-serial multiplier, the area of the datapath can be by adjusted by changing the digit-size. On some FPGA platforms, we can also use the dedicated multipliers or MACs (e.g. 25-bit MAC on Virtex-5) as building blocks of a modular multiplier [50] [31].

When targeting high speed applications, an implementation can use multiple datapaths to run several modular multiplications in parallel [63] [20]. The basic idea is similar to a classical superscalar processor that explores the instruction level parallelism. However, such kind of architecture normally requires aggressive memory accesses, which requires a high throughput memory.

2.3.2 Interface

The interface between the co-processor and CPU is of vital importance, as the HW/SW partitioning has a large impact on the performance and flexibility. Take ECC as an example, one can immediately see the following three choices, as shown in Fig. 3.

When choosing P_L as the partitioning point, the co-processor performs the modular arithmetic while the CPU generates instruction sequences for point addition/doubling and all functions above. This choice is offers a very good flexibility. On the other side, the CPU has to frequently transfer data and instructions to the co-processor, which may become a bottleneck. If we choose P_H as the partitioning point, then the CPU only needs to send the data at the very beginning of a scalar multiplication, and substantially reduces the communication overhead. However, it

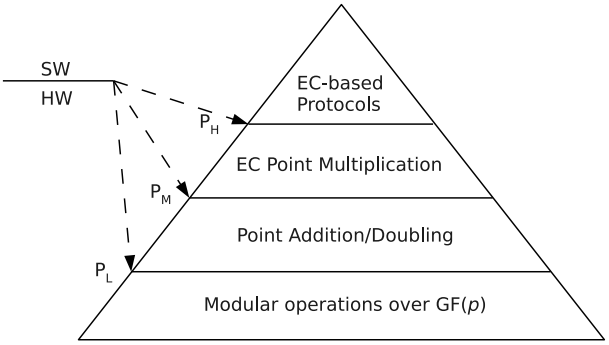

Fig. 3 Hardware/Software co-design for ECC.

is difficult to update the scalar multiplication algorithm since it is now implemented in hardware.

Being able to update the functionality of an cryptographic implementation is very important as standards may change or the old algorithm is no longer safe due to the discovery of new attacks. For example, when new side-channel attacks are founded, corresponding countermeasures need to be added. Thus, the co-processor should have a certain degree of programmability. One method to achieve low communication overhead and high flexibility is to introduce control hierarchy. Instead of fixed state machine (FSM), a programmable micro-controller can be used in the co-processor.

2.3.3 Low Power Architecture

When using cryptography in constrained devices, i.e. passive RFID tags and wireless sensor nodes, power consumption becomes a big challenge. Several low power architectures [45] [33] for ECC have been proposed. Traditional techniques such as reducing the hardware size, lowering the clock frequency and using clock-gating are helpful. For ECC implementations, registers account for more than 60% of the area and most of the dynamic power. Thus, reducing the size of register file and register flipping can significantly reduce the power consumption.

3 Secure Implementations: Side Channel Attacks

Cryptographic algorithms, i.e. mathematical objects describing the input-output behavior of a black box, have to go through a thorough process of cryptanalysis before they are widely accepted as secure and possibly standardized. Implementations of cryptographic algorithms essentially instantiate these black boxes.

An implementation of an algorithm does however not automatically inherit the algorithm's security. The fact that the abstract black box is implemented in software, hardware, or a combination of both gives rise to new security risks. Indeed, measurable physical properties of an implementation become an additional, unintended source of information, giving away knowledge about the secrets involved in cryptographic implementations. Those sources are called side channels. While the execution time of an implementation can be measured even from a distance, e.g. over a network, the physical accessibility of embedded devices in particular gives rise to even more side channels such as the power consumption, electromagnetic radiation, acoustics, heat dissipation, light emission, etc.

Cryptanalytical methods exploiting such information leakage to extract secret data are called side channel attacks. Side channel attacks are a serious concern as they allow to extract secret information from unprotected implementations of black box secure algorithms with moderate effort. With the ever increasing availability and ubiquity of embedded devices, side channel attacks are a realistic threat nowadays.

One type of side channel attacks, a differential side channel attack, sequences several steps to extract the secret information. In a first stage the attacker collects measurements of the side channel, e.g. power consumption, as a function of the time. Each measurement is the physical representation of the execution of the cryptographic algorithm with a different message but a fixed key. In a second stage, the attacker preprocesses the data, chooses a hypothetical power leakage model and calculates the hypothetical leakage using this model for each message and for every possible key guess. An example of one such model is the hamming weight of the processed data at a certain time instant. In this case an adversary assumes that the amplitude of the side channel measurement is related to the amount of binary ones in the data. The validity of this model in certain cases is demonstrated in Fig. 4. In

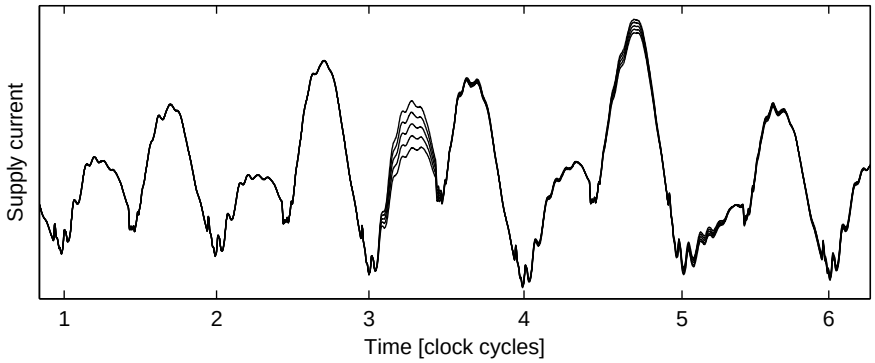

Fig. 4 Measurements of supply current over time. Data dependent operations in clock cycles three and four leak information about the operand(s).

the last phase of the attack, the attacker uses a statistical test to quantify the similar-

ity between the hypothetical leakage and the real side channel leakage for every key guess. The key that reveals the strongest relation is the adversary's best guess.

3.1 DSP Techniques Used in Side Channel Analysis

The side channel analysis research domain has become a mature research area over the last decade, hence a multitude of techniques for preprocessing the data and quantifying the relation between the real and hypothetical leakage have been proposed.

The measurements of the side channel are susceptible to external and internal noise contamination. Therefore, a first group of preprocessing techniques aims at reducing the amplitude noise. The easiest preprocessing technique is averaging the measurements. This was first pointed out by Kocher et al. in [41]. More noise reduction can be achieved through filtering the data [3]. Results have been published for the power side channel [51] as well as for the electromagnetic side channel [1]. Another, more recently proposed technique is calculating the fourth-order cumulant [42]. In some cases, the useful information is modulated onto a carrier. For this type of data extraction, demodulation techniques are appropriate [1] [38] [56].

Due to clock jitter, absence of a suitable trigger or simply due to countermeasures, measurements can suffer from temporal misalignment. Some publications deal with this problem and try to remove the temporal misalignment. Homma et al. explain the phase-based waveform matching procedure [34] and Gebotys and White introduce the phase replacement technique [24]. Pelletier and Charvet use wavelets to get rid of misalignment [59]. The rapid alignment method [55] of Muijrers et al. also builds on wavelets. Woudenberg et al. propose to use dynamic time warping [66].

In the final phase of a side channel attack, the attacker uses a statistical tool to find the correct key by analyzing the relation between the hypothetical leakage and the measured leakage. The difference of means test was proposed by Kocher et al. in their seminal paper [41]. Later, more advanced techniques were put forward: the T-test [14], the Pearson correlation coefficient [8], Spearman rank correlation [5], mutual information [25], and maximum likelihood [11]. While all the previous techniques are applied in the time domain, the differential frequency analysis exploits the frequency information [23].

Advanced signal processing techniques strengthen the adversary's capabilities and raise the need for proper and better countermeasures against side channel attacks.

4 Working with *Fuzzy Secrets*

4.1 Fuzzy Secrets*: Properties and Applications in Cryptography*

Secrets play an important role in cryptography, since the alleged security of many cryptographic schemes is solely based on the assumed confidentiality of certain data, mostly called a *key*. In order for a secret key to be secure, it needs to be hard for an adversary to guess its value. Keys are mostly represented as digital n bit strings and when they are sampled uniformly at random from $\{0,1\}^n$, an adversary can only guess their value with probability 2^{-n}, which becomes negligibly small when n is chosen large enough. Also, a cryptographic algorithm should only succeed when the correct key is applied, and fail when even the slightest bit is changed, in order to prevent an adversary from guessing *close* keys, which is easier than 2^{-n}. This means that secret keys cannot be subject to any kind of noise or distortion.

In a number of security applications, secret data is present, but not in the form of a uniformly distributed and noise-free binary key string. This is often the case when the secret stems from a physical phenomenon which is mostly analogue in nature, not uniformly distributed and subject to random noise. Such data is called a *fuzzy secret*. Some examples of cryptographic settings involving fuzzy secrets are listed below:

- Biometric readings such as fingerprints, iris scans, face structure and voice patterns are believed to be unique to each human being and can hence serve as a unique, and possible secret, identifier. To obtain digital data from these readings, a feature extraction process is necessary. However, the extracted features are most likely not uniformly distributed and some features are measured differently at consecutive readings, *i.e.* they are subject to measurement noise. Extracted biometrical features can hence not be directly used as a cryptographic key.
- Physically unclonable functions [48] or PUFs are cryptographic hardware primitives that fulfill a function similar to biometrics, *i.e.* they measure unique physical properties of their embedding device. Interesting constructions of PUFs are those that are embedded into integrated circuits, since they allow to generate device-unique and highly protected chip identifiers [29] [21]. The uniqueness of every chip stems from uncontrollable nano-scale manufacturing variability in the production process of the circuits. Like biometrics, PUF responses suffer from a non-uniform distribution and unreliability due to measurement inaccuracy and random thermal noise in the electrical measurement circuits.

More settings involving fuzzy secrets exist, *e.g.* in authentication with long passphrases and in quantum key agreement.

The non-uniformity and the noise of a fuzzy secret can be quantified. Information entropy is commonly used to describe the uncertainty about a random variable. A special case of entropy called *min-entropy* describes the worst case uncertainty, *i.e.* it evaluates the best chances an adversary has in guessing a secret value chosen from a known distribution. The min-entropy of a random variable X is denoted as

$H_\infty(X)$ and defined as $H_\infty(X) \stackrel{def}{=} -\log_2 \max_x \Pr[X = x]$. A uniformly distributed n-bit variable has maximal min-entropy equal to n, while a deterministic value has min-entropy zero. To describe the noise, it is assumed that the measured fuzzy secret X is a bit vector of length n. Two distinct observations of the fuzzy secret possibly differ in a number of bit locations due to noise. The *amplitude* of the noise affecting X can be quantified by the maximum number $\delta(X)$ of differing bits between two measurements.

4.2 *Generating Cryptographic Keys from* Fuzzy Secrets *by means of Error Correction*

Mostly, a physical variable cannot be directly used. It should first be measured as accurately as possible and afterwards quantized into a discrete value which can be used by a digital algorithm. The quantization process translates physical noise into bit errors of a bit vector. Also, due to rounding, part of the information in the measured value is already lost. A careful choice for quantization should be made to optimize these parameters given the implementation constraints.

After measurement and quantization, one is left with a possibly non-uniform and unreliable bit string $X \in \{0,1\}^n$. By performing multiple measurements, the aforementioned parameters $H_\infty(X)$ and $\delta(X)$ can be estimated. The next step in the transformation is dealing with the noise which can introduce up to $\delta(X)$ bit errors. Error correction techniques are frequently used in channel coding theory. A process known as the *code-offset construction* [36] [16] [46] is an elegant way of using error correcting block codes in order to get rid of bit errors in a measurement of X. The technique works in two phases:

1. In the generation phase, X is measured and a codeword C is randomly selected from an $[n,k,t]$-block code, with n the code length, k the code dimension and t the error correcting capability. The binary offset between X and C ia calculated as $W = X \oplus C$ and W is made publicly available, *e.g.* it is published in an online database.
2. In the reproduction phase, X' is measured which can differ from X in up to $\delta(X)$ bit positions. Using the publicly available W, one can calculate $C' = X' \oplus W$, which differs from C in at most $\delta(X)$ bits. If the used code is chosen appropriately such that $t \geq \delta(X)$, the decoding algorithm will succeed to compute $C = \mathsf{Decode}(C')$. In that case the same X as in the generation phase can be reconstructed as $X = C \oplus W$.

More elaborate error-correction techniques can be applied to transform a noisy measurement value into a reliable bit string. In the *syndrome construction* [16] the syndrome, instead of the offset, of a linear block code is used, which reduces the communication overhead. In [47], it is shown that certain fuzzy secrets produce *soft-decision information*, which allows the use of soft-decision error-correcting tech-

niques. This allows to extract a longer and hence more secure key from the same fuzzy secret.

All these techniques uses a public communication to establish a noise-free secret from a noisy secret. The data that is passed through this channel, W is called *helper data*. It is clear that the publishing of W decreases the uncertainty an adversary may have about the value of X. In fact, for the code-offset construction it can be shown that the min-entropy of X is reduced from $H_\infty(X)$ to $H_\infty(X) - n + k$, *i.e.* that the helper data induces a *min-entropy loss*.

To conclude, the now noise-free secret with limited min-entropy needs to be transformed into a uniformly distributed key K. In order to do this, *randomness extractors* are used. Randomness extractors succeed in extracting uniform randomness from non-uniformly distributed variables [9] [6]. It is obvious that the output domain of a randomness extractor is smaller than the input, hence they are compression functions. In fact, the min-entropy of a random variable X is a measure for the maximum number of uniform bits an ideal randomness extractor can extract from X. After error correction with the code-offset technique, the fuzzy secret X can hence contribute at most $H_\infty(X) - n + k$ uniformly random bits. Generic randomness extractors can be constructed relatively easy by using so-called *universal hash functions*: if $\mathscr{H}$ is a universal hash family, the process that selects a random function $h_\sigma \leftarrow \mathscr{H}$ and calculates $K = h_\sigma(X)$ is a randomness extractor. The *seed* σ can again be published as helper data to allow reconstruction of K at later times, this time however without additional loss in min-entropy.

5 Conclusion and Future Work

The purpose of this chapter is to illustrate the usage of signal processing techniques into the design and implementation of cryptographic and security applications. This is only the beginning and by no means complete. New directions are being explored. One example is the exploration of signal processing in the encrypted domain, which is the topic of the SPEED project [61].

Acknowledgements This work is supported in part by the IAP Programme P6/26 BCRYPT of the Belgian State, by the European Commission under contract numbers ICT-2007-216676 ECRYPT NoE phase II and ICT-2007-238811 UNIQUE, and by the Research Council K.U.Leuven: GOA 11/007 TENSE. Benedikt Gierlichs is a Postdoctoral Fellow of the Fund for Scientific Research - Flanders (FWO).

References

1. D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi. The EM Side-Channel(s). In *Cryptographic Hardware and Embedded Systems — CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 29–45. Springer, 2002.

2. S. Baktir, S. Kumar, C. Paar, and B. Sunar. A State-of-the-art Elliptic Curve Cryptographic Processor Operating in the Frequency Domain. In *Mobile Networks and Applications (MONET) Journal, Special Issue on Next Generation Hardware Architectures for Secure Mobile Computing*, pages 259–270, 2007.
3. A. Barenghi, G. Pelosi, and Y. Teglia. Improving First Order Differential Power Attacks through Digital Signal Processing. In *Proceedings of the 3rd international conference on Security of information and networks*, SIN 2010, pages 124–133. ACM, 2010.
4. P. Barrett. Implementing the Rivest Shamir and Adleman Public Key Encryption Algorithm on a Standard Digital Signal Processor. In *Advances in Cryptology — CRYPTO '86*, volume 263 of *Lecture Notes in Computer Science*, pages 311–323. Springer, 1987.
5. L. Batina, B. Gierlichs, and K. Lemke-Rust. Comparative Evaluation of Rank Correlation Based DPA on an AES Prototype Chip. In *Proceedings of the 11th international conference on Information Security – ISC 2008*, volume 5222 of *Lecture Notes in Computer Science*, pages 341–354. Springer, 2008.
6. C. H. Bennett, G. Brassard, and J.-M. Robert. Privacy Amplification by Public Discussion. *SIAM Journal on Computing*, 17(2):210–229, 1988.
7. E. F. Brickell. A Survey of Hardware Implementations of RSA. In *Advances in Cryptology — CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 368–370. Springer, 1990.
8. E. Brier, C. Clavier, and F. Olivier. Correlation Power Analysis with a Leakage Model. In *Cryptographic Hardware and Embedded Systems — CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2004.
9. J. L. Carter and M. N. Wegman. Universal Classes of Hash Functions. In *STOC '77: Proceedings of the 9th ACM symposium on Theory of computing*, pages 106–112. ACM, 1977.
10. Ç.K. Koç, T. Acar, and B.S. Kaliski Jr. Analyzing and comparing Montgomery multiplication algorithms. *IEEE Micro*, pages 26–33, June 1996.
11. S. Chari, C.S. Jutla, J.R. Rao, and P. Rohatgi. Towards Sound Approaches to Counteract Power-Analysis Attacks. In *Advances in Cryptology — CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412. Springer, 1999.
12. R. Cheung, S. Duquesne, J. Fan, N. Guillermin, I. Verbauwhede, and G. Yao. FPGA Implementation of Pairing using Residue Number System and Lazy reduction. In *Cryptographic Hardware and Embedded Systems — CHES 2011*, volume 6917 of *Lecture Notes in Computer Science*, pages 421–441. Springer, 2011.
13. SHA-3 Competition. `http://csrc.nist.gov/groups/ST/hash/sha-3/index.html`.
14. J.-S. Coron, P. C. Kocher, and D. Naccache. Statistics and Secret Leakage. In *Financial Cryptography, 4th International Conference*, volume 1962 of *Lecture Notes in Computer Science*, pages 157–173. Springer, 2000.
15. W. Diffie and M. E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22:644–654, 1976.
16. Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith. Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data. *SIAM Journal on Computing*, 38(1):97–139, 2008.
17. S. R. Dusse and B. S. Kaliski Jr. A Cryptographic Library for the Motorola DSP56000. In *Advances in Cryptology — CRYPTO '90*, volume 537 of *Lecture Notes in Computer Science*, pages 230–244. Springer, 1991.
18. M. D. Ercegovac. On-line Arithmetic: An Overview. In *SPIE Real-Time Signal Processing VII*, pages 86–93, 1984.
19. M. D. Ercegovac and T. Lang. On-line Arithmetic for DSP Applications. In *32nd IEEE Midwest Symposium on Circuits and Systems*, 1989.
20. J. Fan, K. Sakiyama, and I. Verbauwhede. Elliptic Curve Cryptography on Embedded Multicore Systems. *Design Automation for Embedded Systems*, 12(3):231–242, 2008.
21. B. Gassend, D. Clarke, M. van Dijk, and S. Devadas. Silicon Physical Random Functions. In *CCS 2002: Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 148–160. ACM, 2002.

22. P. Gastaldo, G. Parodi, and R. Zunino. Enhanced Montgomery Multiplication on DSP Architectures for Embedded Public-Key Cryptosystems. *EURASIP J. Embedded Syst.*, pages 1–9, 2008.
23. C. H. Gebotys, S. Ho, and C. C. Tiu. EM Analysis of Rijndael and ECC on a Wireless Java-Based PDA. In *Cryptographic Hardware and Embedded Systems — CHES 2005*, volume 3659 of *Lecture Notes in Computer Science*, pages 250–264. Springer, 2005.
24. C. H. Gebotys and B. A. White. EM Analysis of a Wireless Java-Based PDA. *ACM Transactions on Embedded Computing Systems*, 7(4):1–28, 2008.
25. B. Gierlichs, L. Batina, P. Tuyls, and B. Preneel. Mutual Information Analysis - A Generic Side-Channel Distinguisher. In *Cryptographic Hardware and Embedded Systems — CHES 2008*, volume 5154 of *Lecture Notes in Computer Science*, pages 426–442. Springer, 2008.
26. D. Gollmann, Y. Han, and C. Mitchell. Redundant Integer Representation and Fast Exponentiation. *Designs, Codes and Cryptography*, 7:135–151, 1998.
27. J. Großschädl, K. C. Posch, and S. Tillich. Architectural Enhancements to Support Digital Signal Processing and Public-Key Cryptography. In *Proceedings of the 2nd Workshop on Intelligent Solutions in Embedded Systems – WISES 2004*, pages 129–143, 2004.
28. J. Guajardo, R. Blumel, U. Krieger, and C. Paar. Effcient Implementation of Elliptic Curve Cryptosystems on the TI MSP 430x33x Family of Microcontrollers. In *Public Key Cryptography*, volume 1992 of *Lecture Notes in Computer Science*, pages 365–382. Springer, 2001.
29. J. Guajardo, S. Kumar, G.-J. Schrijen, and P. Tuyls. FPGA Intrinsic PUFs and Their Use for IP Protection. In *Cryptographic Hardware and Embedded Systems — CHES 2007*, volume 4727 of *Lecture Notes in Computer Science*, pages 63–80. Springer, 2007.
30. N. Guillermin. A High Speed Coprocessor for elliptic curve scalar multiplication over Fp. In *Cryptographic Hardware and Embedded Systems — CHES 2010*, volume 6225 of *Lecture Notes in Computer Science*, pages 48–64. Springer, 2010.
31. T. Güneysu and C. Paar. Ultra High Performance ECC over NIST Primes on Commercial FPGAs. In *Cryptographic Hardware and Embedded Systems — CHES 2008*, volume 5154 of *Lecture Notes in Computer Science*, pages 62–78. Springer, 2008.
32. O. Gustafsson and L. Wanhammar. Arithmetic. In S. S. Bhattacharyya, E. F. Deprettere, R. Leupers, and J. Takala, editors, *Handbook of Signal Processing Systems*. Springer, second edition, 2012.
33. D. Hein, J. Wolkerstorfer, and N. Felber. ECC Is Ready for RFID — A Proof in Silicon. In *Selected Areas in Cryptography — SAC 2008*, volume 5381 of *Lecture Notes in Computer Science*, pages 401–413. Springer, 2009.
34. N. Homma, S. Nagashima, Y. Imai, T. Aoki, and A. Satoh. High-Resolution Side-Channel Attack Using Phase-Based Waveform Matching. In *Cryptographic Hardware and Embedded Systems — CHES 2006*, volume 4249 of *Lecture Notes in Computer Science*, pages 187–200. Springer, 2006.
35. K. Itoh, M. Takenaka, N. Torii, S. Temma, and Y. Kurihara. Fast Implementation of Public-Key Cryptography on a DSP TMS320C6201. In *Cryptographic Hardware and Embedded Systems — CHES 1999*, volume 1717 of *Lecture Notes in Computer Science*, pages 61–72. Springer, 1999.
36. A. Juels and M. Wattenberg. A Fuzzy Commitment Scheme. In *CCS '99: Proceedings of the 6th ACM Conference on Computer and Communications Security*, pages 28–36. ACM Press, 1999.
37. K. Kalach and J. P. David. Hardware Implementation of Large Number Multiplication by FFT with Modular Arithmetic. In *3rd International IEEE-NEWCAS Conference*, pages 267–270. IEEE, 2005.
38. T. Kasper, D. Oswald, and C. Paar. Side-Channel Analysis of Cryptographic RFIDs with Analog Demodulation. In *Workshop on RFID Security and Privacy*, Lecture Notes in Computer Science. Springer, 2011.
39. N. Koblitz. Elliptic Curve Cryptosystem. *Math. Comp.*, 48:203–209, 1987.

40. N. Koblitz. A Family of Jacobians Suitable for Discrete Log Cryptosystems. In *Advances in Cryptology — CRYPTO '88*, volume 403 of *Lecture Notes in Computer Science*, pages 94–99. Springer, 1988.
41. P. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In *Advances in Cryptology — CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.
42. T.-H. Le, J. Clédière, C. Servière, and J.-L. Lacoume. Noise Reduction in Side Channel Attack Using Fourth-Order Cumulant. *IEEE Transactions on Information Forensics and Security*, 2(4):710–720, 2007.
43. Y. K. Lee, H. Chan, and I. Verbauwhede. Design Methodology for Throughput Optimum Architectures of Hash Algorithms of the MD4-class. *Journal of Signal Processing Systems*, 53:89–102, November 2008.
44. Y. K. Lee, M. Knežević, and I. Verbauwhede. Hardware Design for Hash functions. In *Secure Integrated Circuits and Systems*, Integrated Circuits and Systems, pages 79–104. Springer, 2010.
45. Y. K. Lee, K. Sakiyama, L. Batina, and I. Verbauwhede. Elliptic-Curve-Based Security Processor for RFID. *IEEE Transaction on Computers*, 57(11):1514–1527, 2008.
46. J.-P. Linnartz and P. Tuyls. New Shielding Functions to Enhance Privacy and Prevent Misuse of Biometric Templates. In *4th international conference on Audio- and Video-based Biometric Person Authentication – AVBPA'03*, pages 393–402, 2003.
47. R. Maes, P. Tuyls, and I. Verbauwhede. Soft decision helper data algorithm for SRAM PUFs. In *Proceedings of the 2009 IEEE international conference on Symposium on Information Theory - Volume 3*, ISIT 2009, pages 2101–2105. IEEE, 2009.
48. R. Maes and I. Verbauwhede. Physically Unclonable Functions: A Study on the State of the Art and Future Research Directions. In *Towards Hardware-Intrinsic Security*, Information Security and Cryptography, pages 3–37. Springer, 2010.
49. A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.
50. N. Mentens, K. Sakiyama, B. Preneel, and I. Verbauwhede. Efficient Pipelining for Modular Multiplication Architectures in Prime Fields. In *GLSVLSI 2007: Proceedings of the 17th ACM Great Lakes symposium on VLSI*, pages 534–539. ACM, 2007.
51. T.S. Messerges, E. A. Dabbish, and R. H. Sloan. Examining Smart-Card Security under the Threat of Power Analysis Attacks. *IEEE Transactions on Computers*, 51(5):541–552, May 2002.
52. V. Miller. Uses of Elliptic Curves in Cryptography. In *Advances in Cryptology — CRYPTO '85*, volume 218 of *Lecture Notes in Computer Science*, pages 417–426. Springer, 1986.
53. P. Montgomery. Modular Multiplication without Trial Division. *Mathematics of Computation*, 44(170):519–521, 1985.
54. S. Morozov, C. Tergino, and P. Schaumont. System integration of Elliptic Curve Cryptography on an OMAP platform. *Symposium on Application Specific Processors*, pages 52–57, 2011.
55. R. Muijrers, J. Van Woudenberg, and L. Batina. RAM: Rapid Alignment Method. In *Smart Card Research and Advanced Applications — CARDIS 2011*, Lecture Notes in Computer Science. Springer, 2011. To appear.
56. D. Oswald and C. Paar. Breaking Mifare DESFire MF3ICD40: Power Analysis and Templates in the Real World. In *Cryptographic Hardware and Embedded Systems — CHES 2011*, volume 6917 of *Lecture Notes in Computer Science*, pages 207–222. Springer, 2011.
57. K. K. Parhi and Y. Chen. Signal flow graphs and data flow graphs. In S. S. Bhattacharyya, E. F. Deprettere, R. Leupers, and J. Takala, editors, *Handbook of Signal Processing Systems*. Springer, second edition, 2012.
58. K.K. Parhi. *VLSI Digital Signal Processing Systems: Design and Implementation*. Weley, 1999.
59. H. Pelletier and X. Charvet. Improving the DPA Attack using Wavelet Transform. NIST Physical Security Testing Workshop, 2005.

60. K.C. Posch and R. Posch. Modulo reduction in residue number systems. *IEEE Transactions on Parallel and Distributed Systems*, 6(5):449–454, 1998.
61. SPEED Project. `http://www.speedproject.eu/`.
62. R. L. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.
63. K. Sakiyama, L. Batina, B. Preneel, and I. Verbauwhede. Superscalar Coprocessor for High-Speed Curve-Based Cryptography. In *Cryptographic Hardware and Embedded Systems — CHES 2006*, volume 4249 of *Lecture Notes in Computer Science*, pages 415–429. Springer, 2006.
64. M. U. Sharif, R. Shahid, M. Rogawski, and K. Gaj. Use of Embedded FPGA Resources in Implementations of Five Round Three SHA-3 Candidates. In *Ecrypt II Hash Workshop*, 2011.
65. M. A. Soderstrand, W. K. Jenkins, G. A. Jullien, and F. J. Taylor. Residue Number System: Modern Applications in Digital Signal Processing. IEEE Press, 1986.
66. J.G.J. van Woudenberg, M.F. Witteman, and B. Bakker. Improving Differential Power Analysis by Elastic Alignment. In *Topics in Cryptology — CT-RSA 2011*, volume 6558 of *Lecture Notes in Computer Science*, pages 104–119. Springer, 2011.