Design Methods for Embedded Security

Miroslav Knežević, Vladimir Rožić, and Ingrid Verbauwhede

***Abstract*—Embedded devices need both an efficient and a secure implementation of cryptographic algorithms. In this overview paper we show a typical top-down approach for secure and efficient implementation of embedded systems. We outline the security pyramid by illustrating the five primary abstraction levels in an embedded system. Focusing only on two levels - architecture and circuit level - we show how the design can be implemented to be both efficient and secure.**

***Index Terms*—Security, embedded systems, design methods, design pyramid, public key cryptosystems (PKC).**

I. Introduction

A highly developed information society has changed our lives drastically. Many types of data are converted into the streams of bits and we often communicate private information via an open network such as the Internet. Therefore, it becomes very important to protect the digital information appropriately in order to prevent information leakage and to detect impersonation and data substitution.

In recent years, we have observed an increasing number of security breaches on telecommunication networks. One notable example is the eavesdropping on the GSM communications of members of the Greek government [1]. Traffic analysis and privacy become an increasing concern; in this context we note the publication of the European Data Retention Directive that obliges telecommunications system operators to collect and store traffic data [2]. The mass media repeatedly cover frightful stories that describe abuse of electronic wireless tags (RFID). Breaking the tags has become possible only by observing an RF signal [3], by clever cryptanalysis [4] or by software hacks [5].

The embedded system field, with devices such as cellular phones, PDAs, RFIDs and smart cards grows rapidly and we need these electronics to become more trustworthy. On the horizon are futuristic technologies such as embedded network sensors and wearable computers, which point towards an even greater interaction between humans and machines. Embedded biometric authentication devices are already part of our lives [6]. Designers are already well acquainted with design for low-power, small-footprint, high-throughput and so forth, but secure electronic design is still an ad-hoc process.

The goal of this survey is to outline a design methodology for secure and efficient implementation of cryptographic algorithms on embedded devices. By optimizing different levels of a typical top-down approach we try to integrate a secure electronic design in the whole design pyramid. The design target is to achieve both efficient and trustworthy implementation. Why this is a difficult task we can illustrate with a typical approach where engineers are trained to make something happen. In contrast to this approach, secure electronic design requires making something that will not happen (attacks). It involves more effort and makes the whole design process more difficult.

The authors are with Katholieke Universiteit Leuven, ESAT/SCD-COSIC, Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium (e-mails: {Miroslav.Knezevic, Vladimir.Rozic, Ingrid.Verbauwhede}@esat.kuleuven.be).

We use an example of implementing Public Key cryptosystems (PKC) on a constrained device to demonstrate the necessity of addressing all levels of the security pyramid to ensure a fully robust and secure embedded system. However, in this work we focus on building and optimizing only two levels of abstraction, namely the algorithm level and the circuit level.

The paper is structured as follows. Section II discusses the model evolution of embedded security and outlines the security pyramid by illustrating the five primary abstraction levels in an embedded system. In Section III the protocol level is mentioned. Algorithm level is outlined in Section IV. Section V discusses architecture and microarchitecture level and describes the way to achieve efficient modular multiplication as a key operation in Public Key cryptosystems. In Section VI we show how the side channel attacks can be countermeasured on a circuit level. Section VII concludes the paper.

II. Embedded Security - Model Evolution

A simplified representation of the old model of embedded security [7] is given in Fig. 1a. It assumes attacks on a *channel* between communicating parties. Encryption and other cryptographic operations are in *black* boxes. Protection is guaranteed by strong mathematical algorithms and protocols which are computationally secure.

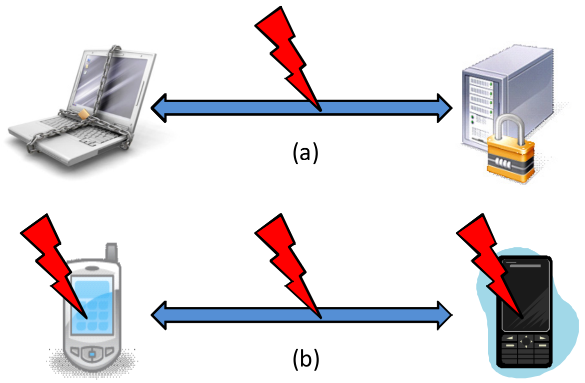

Fig. 1. Embedded security – Model evolution.

The model evolution brings us to the new model of embedded security [7] that is depicted in Fig. 1b. The obvious difference is that an adversary attacks both *channel* and *endpoints*. Hence, the encryption and other cryptographic operations are executing within *gray* boxes. Protection is still guaranteed by strong mathematical algorithms and protocols which are computationally secure. The attacks on the endpoints are of a newer date and are classified as the *side channel attacks*. We discuss the side channel attacks in more detail in Sect. VI.

This model evolution points out that we need not only secure and strong crypto-algorithms, but secure implementations as well. The security pyramid [6] in Fig. 2 illustrates the five primary abstraction levels in an embedded system:

- *Protocol* level, which includes the design of protocols to be performed on embedded devices to achieve such security goals as confidentiality, identification, data integrity, data origin authentication and non repudiation.
- *Algorithm* level, consisting of the design of cryptographic primitives (such as Public Key, Symmetric Key crypto algorithms and hash functions) and application-specific algorithms used at the protocol level.
- *Architecture* level, consisting of secure hardware/software partitioning and embedded software techniques to prevent software hacks.
- *Microarchitecture* level, which deals with the hardware design of modules (the processors and coprocessors) required and specified at the architecture level.
- *Circuit* level, which requires implementing transistor-level and package-level techniques to thwart various physical-layer attacks.

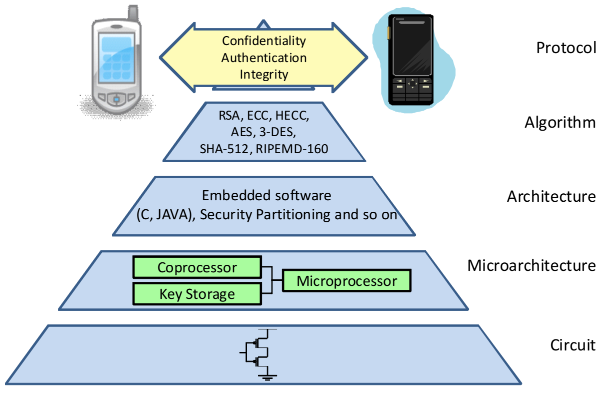

Fig. 2. Embedded security pyramid.

III. Protocol Level: Application Specific

A *protocol* is a multi-party algorithm, defined by a sequence of steps precisely specifying the actions required of two or more parties in order to achieve a specified objective [8]. This general definition of a protocol considers achieving security goals such as confidentiality, identification, data integrity, data origin authentication and nonrepudiation. The protocol level is application specific and includes the design of protocols to be performed on embedded devices.

IV. Algorithm Level: Public Key Cryptosystems

Public Key cryptosystems (PKC) are unavoidable today in almost all spheres of digital communication *e.g.* for financial, governmental or medical applications. Even for pervasive security i.e. extremely low-cost applications running on RFIDs and sensor nodes PKCs are sometimes necessary [9].

The best-known and most commonly used public-key cryptosystems are based on factoring (RSA) [10] and on the discrete logarithm problem in a large prime field (Diffie-Hellman, ElGamal, Schnorr, DSA) [8]. Elliptic Curve Cryptography (ECC) [11], [12], and Hyper-elliptic Curve Cryptography (HECC) [13] are based on a different algebraic structure. ECC and HECC rely on a group structure induced on an elliptic curve and on the group of Jacobian on a hyperelliptic curve respectively. For example, a set of points on an elliptic curve together with the point at infinity, and with point addition as a binary operation has the structure of an Abelian group. After more than two decades of extensive research on both theoretical and practical aspects it is evident that ECC and HECC (so-called curve-based cryptography) offer equivalent security as RSA for much smaller key sizes. This results in hardware of smaller footprint and lower power consumption. These features are very important for applications with very strict constraints on area, power, energy etc. i.e. for embedded security.

V. Architecture and Microarchitecture Level: Efficient Modular Multiplication

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.

The most popular algorithm for modular multiplication is the Montgomery's method [14]. The approach of Montgomery avoids the time consuming trial division that is the common bottleneck of other algorithms. The Montgomery's algorithm has one property, namely a precomputational step, where the inverse of the modulus is calculated and stored together with the value of the modulus. As the modular inversion operation is computationally very expensive (especially in hardware), one usually fixes the value of modulus and uses the precomputed value of the inverse. This reduces flexibility as well as the performance of the implementation increasing the time and memory needed for precomputation of the modulus inverse.

The interleaved Montgomery multiplication algorithm for a w-bit architecture in the finite field of characteristic 2 is outlined in Alg. 1. To make the algorithms more clear, we use the following notations. Each element of the field GF(2^n) we represent as a polynomial of degree less than or equal to $n-1$, written as $A(x) = \sum_{i=0}^{n-1} a_i x^i$, where $a_i \in GF(2)$. Similarly, the same element $A(x)$ can be written in word-representation as $A(x) = \sum_{i=0}^{n_w-1} A_i(x) x^{iw}$, where $n_w = \lceil n/w \rceil$ is the number of words and $A_i(x)$ is a word, represented as a polynomial of degree $w-1$ such that $A_i(x) = \sum_{j=0}^{w-1} a_{iw+j} x^j$.

Algorithm 1 Interleaved word-serial modular multiplication with Montgomery modular reduction in GF(2^n).

```
Input: elements of GF($2^n$) $A(x) = \sum_{i=0}^{n_w-1} A_i(x)x^{iw}$, $B(x) = \sum_{i=0}^{n_w-1} B_i(x)x^{iw}$, $R(x) = x^w$ and modulus $M(x) = \sum_{i=0}^{n_w-1} M_i(x)x^{iw}$, where $n_w = \lceil n/w \rceil$, $M_0(x) \neq 0$ and
  PRE-COMPUTED $\lambda(x) = -M_0(x)^{-1} \bmod R(x)$.
Output: $T(x) = A(x)B(x)R(x)^{-n_w} \bmod M(x)$.
  $T_{-1}(x) \Leftarrow 0$
  for $i = 0$ to $n_w - 1$ do
    $U_i(x) \Leftarrow T_{i-1}(x) + A_i(x)B(x)$
    $s(x) \Leftarrow (U_i(x) \bmod R(x))\lambda(x) \bmod R(x)$
    $T_i(x) \Leftarrow (U_i(x) + M(x)s(x))/R(x)$
  end for
  $T(x) \Leftarrow T_{n_w-1}(x)$
  return $T(x)$.
```

Algorithm 2 Interleaved word-serial modular multiplication with modified Montgomery modular reduction in GF(2^n) without pre-computation.

```
Input: elements of GF($2^n$) $A(x) = \sum_{i=0}^{n_w-1} A_i(x)x^{iw}$, $B(x) = \sum_{i=0}^{n_w-1} B_i(x)x^{iw}$, $R(x) = x^w$ and modulus $M(x) = x^{n-1} + \Delta(x) + 1$, where $\Delta(x) = \sum_{i=w}^{n-2} m_i x^i$ and $n_w = \lceil n/w \rceil$.
Output: $T(x) = A(x)B(x)R(x)^{-n_w} \bmod M(x)$.
  $T_{-1}(x) \Leftarrow 0$
  for $i = 0$ to $n_w - 1$ do
    $U_i(x) \Leftarrow T_{i-1}(x) + A_i(x)B(x)$
    $s(x) \Leftarrow U_i(x) \bmod R(x)$
    $T_i(x) \Leftarrow (U_i(x) + M(x)s(x))/R(x)$
  end for
  $T(x) \Leftarrow T_{n_w-1}(x)$
  return $T(x)$.
```

To skip the precomputational step and to further speed up the algorithm, we have proposed a modified version of the Montgomery's algorithm. The algorithm is given in Alg. 2 and it uses a special type of moduli, where w-bits of the modulus need to be fixed. Besides skipping a precomputational step, this algorithm simplifies the quotient evaluation, $s(x)$, which makes the implementations both faster and smaller. Two 32-bit architectures for 192×192-bit multipliers were synthesized based on the Algs. 1 and 2, respectively. The improvement resulted in approximately 10 % faster and 10 % smaller design. For more details about the implementation and the proof of the Alg. 2, please refer to [15].

VI. Circuit Level: Power-Constant Logic

It is not enough to have mathematically strong cryptographic algorithms, it is also important that these algorithms are implemented in a secure way. There are attack techniques that target the physical implementation rather than the algorithm itself. These attacks can be classified into two categories – *active attacks* where the malicious party alters the hardware or software in some way or changes the operating conditions (introduces glitches in the power supply, changes operating temperature) and *passive attacks* where the attack is based on monitoring side-channel information (power supply, electromagnetic radiation). The latter are also called *side-channel attacks* (SCA). These attacks are noninvasive since the attacker does not tamper with the device, but only observes the information that leaks out of the system. The most obvious way to gather side-channel information of the device is to monitor its power consumption.

The attack called *Simple Power Analysis* (SPA) is based on directly interpreting power consumption measurements. When sampled at high rates and examined in more detail, power waveforms can reveal the key bits. If there is a conditional branching depending on the secret information (key or subkey bits) it will be shown in the powertrace. One way to protect the device from the SPA attack is to avoid conditional branching. This is an algorithm level solution and it can have a performance penalty. This way some of the SPA characteristics are eliminated and variations in power consumption are reduced to a reasonable measure. However, this does not protect the device from more sophisticated attacks such as *Differential Power Analysis* (DPA) [16]. DPA exploits power variations correlated to data values which can be very small and masked with noise and measurement errors. However, statistical operations performed on a large set of measurements can detect very small correlations between the power traces.

The reason why SCAs are possible is data-dependant power consumption which is a phenomenon present in most standard logic cells. We explain this on the example of a static CMOS cell. As depicted in Fig. 3, a CMOS inverter consumes energy only during the 0–1 transition of the output. During the 1–0 transition the energy is dissipated. If there is no transition, no power is consumed. This asymmetric power consumption provides the information used in DPA to find the secret key.

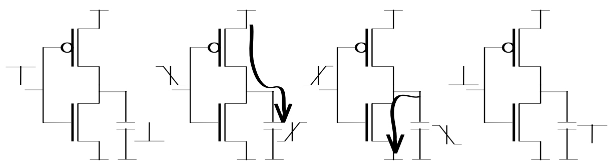
Fig. 3. CMOS output transitions and their power consumptions.

A solution to the problem regarding the side-channel attacks is to use a logic style with data independent power consumption. This allows a designer to use a secure library without worrying about SCA. Dynamic differential cascode voltage switch logic (DDCVSL) [17] in Fig. 4a seems like a good solution. By making logic style differential, input values are masked – there is no difference between 1–0 and 0–1 events since in both cases exactly one node is discharged. By making it dynamic it is achieved that there is no difference between 0–0 and 1–0 event or between 0–1 and 1–1 event. Therefore there is no difference in power consumption between any two transitions. However, the effects of parasitic capacitances are not taken into account so for different combination of inputs, different combination of internal node capacitances will discharge which will cause data-dependant power variations. Sense Amplifier Based Logic (SABL) [18] in Fig. 4b solves this problem since in every clock cycle all internal nodes discharge.

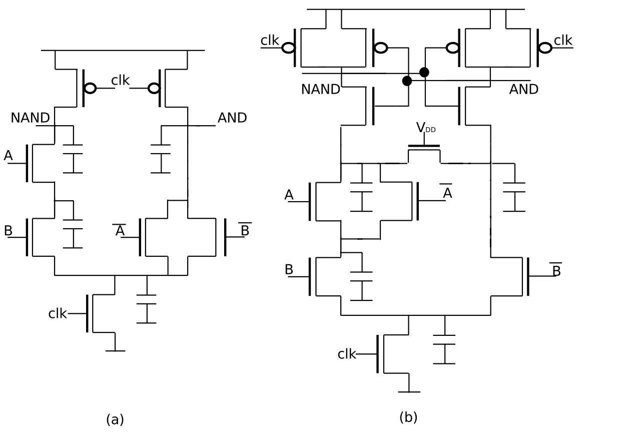

Fig. 4. (a) Dynamic DCVSL AND-NAND gate and (b) SABL AND-NAND gate.

The quantity used as a measure for the resistance against DPA is the number of measurements to disclosure (MDTs) [19]. This is the average number of measurements needed to distinguish a correct secret key from all other key guesses. In secure implementations the number of MTDs is increased for more than two orders of magnitude while area and power consumption are increased less than two times.

VII. Conclusion and Outlook

As embedded systems evolve from isolated devices to always-on networked devices, security becomes a paramount issue. Embedded security cannot be solved at a single security abstraction layer, but rather is a system problem spanning multiple abstraction levels. Designers need to change their strategy and consider the secure electronic design not as an ad-hoc process, but as a systematic top-down approach. In this paper we address some of the common issues related to this topic.

Acknowledgment

This work has been funded partially by IBBT, Katholieke Universiteit Leuven (OT/06/40) and FWO projects (G.0300.07 and G.0450.04). This work has been supported in part by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy), by the EU IST FP6 projects (ECRYPT) and by the IBBT-QoE project.

References

[1] D. Spinellis and V. Prevelakis. The Athens Affair. 2005. http://www.spectrum.ieee.org/print/5280.

[2] EU parliament. Directive 2006/24/EC of the European Parliament. 2006. www.ispai.ie.

[3] R. Merritt. Cellphone could crack RFID tags, says cryptographer. In *Electronic Engineering Times, 14/2/06*, 2006.

[4] J. Schwartz. Graduate Cryptographers Unlock Code of 'Thiefproof' Car Key. In *New York Times, Section A, p. 14, 1/29/05*, 2005.

[5] J. Markoff. Study Says Chips in ID Tags Are Vulnerable to Viruses. In *New York Times, Section C, p. 3, 3/15/06*, 2006.

[6] D. Hwang, P. Schaumont, K. Tiri, and I. Verbauwhede. Securing Embedded Systems. In *IEEE Security and Privacy Magazine 4*, pages 40–49, 2006.

[7] I. Verbauwhede. Tutorial - Part I: Introduction to Side-Channel Attacks. In *INDOCRYPT 2007*, 2007.

[8] A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.

[9] Y. K. Lee, L. Batina, K. Sakiyama, and I. Verbauwhede. Elliptic Curve Based Security Processor for RFID. In *IEEE Transactions on Computers*, 2008.

[10] 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.

[11] V. Miller. Uses of elliptic curves in cryptography. In H. C. Williams, editor, *Advances in Cryptology: Proceedings of CRYPTO'85*, number 218 in Lecture Notes in Computer Science, pages 417–426. Springer-Verlag, 1985.

[12] N. Koblitz. Elliptic curve cryptosystem. *Math. Comp.*, 48:203–209, 1987.

[13] N. Koblitz. A family of Jacobians suitable for Discrete Log Cryptosystems. In S. Goldwasser, editor, *Advances in Cryptology: Proceedings of CRYPTO'88*, number 403 in Lecture Notes in Computer Science, pages 94–99. Springer-Verlag, 1988.

[14] P. Montgomery. Modular multiplication without trial division. *Mathematics of Computation*, 44(170):519–521, 1985.

[15] M. Knežević, F. Vercauteren, and I. Verbauwhede. Improved Modular Multiplication based on Barrett and Montgomery Reduction Algorithms. In *COSIC Internal Report*, 2008.

[16] P. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In M. Wiener, editor, *Advances in Cryptology: Proceedings of CRYPTO'99*, number 1666 in Lecture Notes in Computer Science, pages 388–397. Springer-Verlag, 1999.

[17] J. Rabaey. *Digital Integrated Circuits: A Design Perspective*. Prentice Hall, 1996.

[18] K. Tiri, M. Akmal, and I. Verbauwhede. A dynamic and differential CMOS logic with signal independent power consumption to withstand differential power analysis on smart cards. In *Proceedings of 29th European Solid-State Circuits Conference (ESSCIRC 2002)*, pages 403–406, 2002.

[19] K. Tiri and I. Verbauwhede. A digital design flow for secure integrated circuits. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 25(7):1197–1208, 2006.