Microprocessors and Microsystems 000 (2016) 1–14

Contents lists available at ScienceDirect

Microprocessors and Microsystems

journal homepage: www.elsevier.com/locate/micpro

Dummy operations in scalar multiplication over elliptic curves: A tradeoff between security and performance

Simon Pontie[a,b,*], Paolo Maistri[a,b], Régis Leveugle[a,b]

[a] Univ. Grenoble Alpes, TIMA, F-38031 Grenoble, France
[b] CNRS, TIMA, F-38031 Grenoble, France

ARTICLE INFO

Article history:
Received 19 June 2015
Revised 21 January 2016
Accepted 29 February 2016
Available online xxx

Keywords:
Cryptoprocessor
Elliptic curves
Power analysis attacks
Side channel analysis
Scalar multiplication
Dummy operations
Randomized windows

ABSTRACT

A large number of embedded systems require a high level of security. Elliptic curve cryptography is well suited for these constrained environments, but some countermeasures must be implemented to prevent leakage of critical data through side-channel analyses. This work attempts to propose one such counter-measure, without affecting performance. A windowing approach at the scalar multiplication level saves time, which is then used to perturb the attacker by inserting dummy operations at random instants. To increase our power analysis protection, the length of the windows in the scalar partitioning is chosen randomly. Our countermeasure makes the simple power analysis attack ineffective; robustness against differential power analysis is also increased. In order to meet the target security level, performance, or area constraints, designers only need to choose the suitable parameters of the proposed protected scalar multiplication. A new attack based on pattern identification on several power traces is also explored; this attack may be used against the proposed counter-measure but it is shown that with more dummy doublings the attack becomes ineffective with a small performance penalty.

1. Introduction

Security in embedded systems is a real technical challenge. Asymmetric cryptography with a public key infrastructure can solve authentication and confidentiality problems. Among others, designers may choose elliptic curve cryptography (ECC) [1, 2] or RSA [3] to implement a public key infrastructure. RSA was historically the most common choice, but the emergence of new stricter constraints opened by the recent development of the Internet of Things (IoT) and embedded systems is changing this situation.

1.1. ECC better than RSA

The complexity of the best algorithm solving the integer factorization problem (RSA) is sub-exponential, while solving the elliptic curve discrete logarithm problem is fully exponential [4]. That is why an ECC key is shorter than a RSA key for the same security level [5]. An ECC signature computed with the ECDSA algorithm is more bandwidth- and energy-efficient than RSA [6] and unlike with RSA, the time and the energy required to sign or to verify a signature are comparable with ECC. Another advantage with respect to the RSA cryptosystem is that elliptic curves permit to have forward secrecy in a Diffie–Hellman key exchange (ECDHE). In the end, ECDHE consumes less energy than the classical Diffie–Hellman on *GF(p)* [6].

1.2. ECC hierarchy

All elliptic curve crypto-systems have the same hierarchical structure: at the top level a protocol (ECDSA or ECDHE); then, the scalar multiplication that is a multiplication between a scalar coefficient and a point of an elliptic curve; at the lower level the basic operations on points, which are called addition and doubling (not to be mixed up with usual integer or floating point additions and multiplications); and finally, the underlying field where to express the coordinates of the points, usually a prime field *GF(p)* or a binary extension field $GF(2^d)$, both being finite fields. Our study is focused on the level 3: the scalar multiplication. This is a critical operation because in all the protocols there is always a multiplication between secret data (scalar) and a point.

1.3. Side channel attacks

Like all other cryptographic systems, ECC can be the target of Side-Channel Attacks (SCA) [7]. These attacks exploit the leakages about secret data manipulated during computations in either hardware or software implementations. Side-channels that can be

* Corresponding author: Tel.: +33476574730.
E-mail addresses: simon.pontie@imag.fr (S. Pontie), paolo.maistri@imag.fr (P. Maistri), regis.leveugle@imag.fr (R. Leveugle).

http://dx.doi.org/10.1016/j.micpro.2016.02.016
0141-9331/

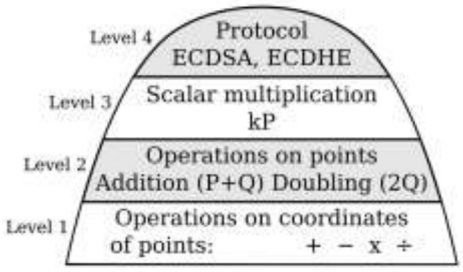

Fig. 1. The hierarchical structure of ECC.

easily monitored are: time of computation; power consumption; electromagnetic emanations. The temporal side channel can be exploited by a distant attacker, for example by measuring the time of response to a challenge. Other side channels require a physical access. A malicious card reader can easily exploit the power consumption channel to clone a smart card, because it is the power supplier. The electromagnetic (EM) channel is more difficult to spy than the power channel but it is more powerful because the attacker can use several EM probes to locally measure this channel and exploit several sources of information [8]. In addition, contactless attacks are possible with the EM channel but not with the power consumption channel.

It is also possible to actively perturb a crypto-processor by injecting faults by various means such as glitches on the power line or on the clock, laser beams or EM coupling. The faulty result may be used to gather information allowing the discovery of secret data [9].

Our work focuses on the two simplest passive attacks, which can be mounted very cheaply and with very little expertise: timing attacks and power consumption analysis. The dissimilarity between addition and doubling on elliptic curves is the vulnerability exploited by the Simple Power Analysis (SPA) attack [7]. In this paper, we propose an SPA counter-measure at the scalar multiplication level. We detail here a hardware implementation but our counter-measure may also be used in software implementations. Software implementations may introduce additional side channel leakage; for example caches were exploited to attack the OpenSSL implementation of ECC [10]. Memory access timings of software implementations executed on processors with caches are not controllable. To keep under control the leakage of information as much as possible, computation levels 1 to 3 in Fig. 1 should be therefore preferably implemented in hardware.

1.4. Our proposal

In this article, we propose a solution to protect a hardware implementation of ECC. We store precomputed points in a table, which allows improving performance by taking advantage of windowed computations at the scalar multiplication level [11]. More details about the implemented architecture have been previously published in [12]. The windows method is further used as a side-channel counter-measure because we randomly choose the length of each window, and we sacrifice part of the performance gain to insert dummy operations at random times. A preliminary version of this countermeasure was proposed in [13], but the implementation suffered from a few weaknesses: the use of dummy additions was not distributed efficiently, and there were security leaks that will be identified in this paper. Here, we propose a new method to efficiently distribute dummy operations. We will show that the SPA attack is no longer possible, but we will then show how pattern identification can be used on multiple traces. In order to prevent this, we will extend the counter-measure to include also dummy point doublings. The DPA attack should also be more difficult due to the desynchronization of traces that use the same secret scalar, but this will not be studied in this paper.

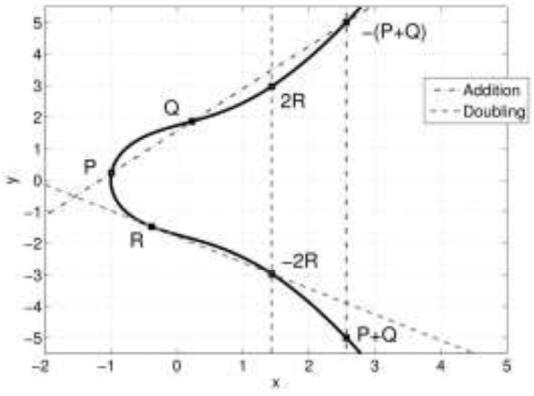

Fig. 2. The two basic operations on points over the curve: Addition and doubling.

The paper is organized as follows: in Section 2, more details are given on ECC computations and side-channel leakage; in Section 3, existing counter-measures are reviewed; Section 4 is focused on randomized windowed computations with dummy additions; Section 5 discusses possible attacks against this scheme and extends it to dummy doublings; Section 6 summarizes the impact on the implementation characteristics.

2. Unprotected design

In this section we will overview bases of an ECC crypto-processor to identify specific constraints and possible sources of leakage in side-channels. Our choices in implementing a hardware crypto-processor are also discussed.

2.1. Elliptic curves

Fig. 2 shows how to compute addition and doubling of points from a geometric point of view. Computation of the addition between a point P and a point Q has two steps. Firstly, we compute the point $-(P+Q)$ that is the intersection between the line (PQ) and the curve. The second step is the identification of $P+Q$ that is the opposite of $-(P+Q)$ (a symmetry by the x-axis if the field is $GF(p)$). The doubling computation is different because when P and Q are the same point, the line (PQ) is not defined. The computation of a doubling, for example $2R$, uses the tangent line passing by R. It must be said that this example does not correspond to a real-world scenario, because in cryptographic applications float values are not used for point coordinates (thus there is no such representation as in Fig. 2), but it allows to understand easily why addition and doubling cannot be unified on classical elliptic curves, such as Weierstrass curves.

The general equation of a Weierstrass elliptic curve is:

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

with some constraints on the values of the coefficients. This equation can be used over the prime field $GF(p)$ or the binary field $GF(2^d)$. For each one of these fields, the general form can be reduced to a short Weierstrass curve form:

- Over $GF(p)$:

  $$y^2 = x^3 + a_4x + a_6$$

- Over $GF(2d)$:

  $$y^2 + xy = x^3 + a_2x^2 + a_6$$

Please cite this article as: S. Pontie et al., Dummy operations in scalar multiplication over elliptic curves: A tradeoff between security and performance, Microprocessors and Microsystems (2016), http://dx.doi.org/10.1016/j.micpro.2016.02.016