# Flexible Hardware Design for RSA and Elliptic Curve Cryptosystems*

Lejla Batina[1], Geeke Bruin-Muurling[2], and Sıddıka Berna Örs[1]

[1] Katholieke Universiteit Leuven, ESAT/COSIC, Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium
[2] Rembrandtlaan 45, 5261 XG, Vught, The Netherlands
{Lejla.Batina, Siddika.BernaOrs}@esat.kuleuven.ac.be
geekemuurling@hotmail.com

**Abstract.** This paper presents a scalable hardware implementation of both commonly used public key cryptosystems, RSA and Elliptic Curve Cryptosystem (ECC) on the same platform. The introduced hardware accelerator features a design which can be varied from very small (less than 20 Kgates) targeting wireless applications, up to a very big design (more than 100 Kgates) used for network security. In latter option it can include a few dedicated large number arithmetic units each of which is a systolic array performing the Montgomery Modular Multiplication (MMM). The bound on the Montgomery parameter has been optimized to facilitate more secure ECC point operations. Furthermore, we present a new possibility for CRT scheme which is less vulnerable to side-channel attacks.

**Keywords:** FPGA design, Systolic array, Hardware implementation, RSA, ECC, Montgomery multiplication, Side-channel attacks

## 1 Introduction

Security of communication or in general of some digital data is founded by various cryptographic algorithms. Especially implementations of Public Key Cryptography (PKC) present a challenge in vast majority of application platforms varying from software to hardware. Software platforms are cheap and a more flexible solution but it appears that only hardware implementations provide a suitable level of security especially related to side-channel attacks. Two best known and most widely used public-key cryptosystems are RSA [26] and ECC [18], [13]. When it comes to RSA, it is believed to be on its "sunset" but still keeping up with requirements. Namely, because of various factors such as well developed speed-ups in the form of Chinese Remainder Theorem (CRT) techniques and

* Lejla Batina and Sıddıka Berna Örs are funded by a research grants of the Katholieke Universiteit Leuven, Belgium. This work was supported by Concerted Research Action GOA-MEFISTO-666 of the Flemish Government and by the FWO "Identification and Cryptography" project (G.0141.03).

T. Okamoto (Ed.): CT-RSA 2004, LNCS 2964, pp. 250–263, 2004.

its suitability for hardware, RSA is the main technology for high-speed applications in network security, financing etc. On the other hand, ECC is expected to take the lead within wireless applications. The reason is that ECC operates with higher speed, lower power consumption and smaller certificates, which are all necessities within these areas including the smartcard industry. In short, it is mostly desired to develop an architecture which can efficiently perform both RSA and ECC, RSA for VPNs, banking etc. and ECC still mostly for wireless applications.

Our contribution deals with an FPGA implementation of RSA and ECC cryptosystems over a field of prime characteristic. The architecture for Montgomery Modular Multiplication (MMM) used in this work is efficient and secure [22]. The systolic array is used for arbitrary precision in bits, hence easily bridging the gap between the bit-lengths for ECC from 160 bits to 2048 (or higher) bit long modulus for RSA. The notion of scalability we discuss includes both, freedom in choice of operand precision as well as adaptability to any desired gate complexity. To the latter is usually referred to as "flexibility". We use modular exponentiation based on Montgomery's method without any modular reduction achieving the best possible bound [29], [3]. We are first to introduce a similar bound for ECC which allows us to perform a very secure and yet efficient point addition and doubling. We show that in the case of two or more arithmetic units a high level of parallelism can be achieved altering ECC operations between those units. The eventual parallelism between more units and also between cells of the systolic array is beneficial for side-channel resistance. Moreover, in this work we introduce a new variation of Garner's scheme for CRT decryption, which has built-in countermeasure against timing and power analysis based attacks.

Since the introduced architecture was dedicated to RSA applications, it was natural to implement elliptic curve arithmetic in $GF(p)$. In this way all required components were already available as ECC in $GF(p)$ is based on ordinary modular arithmetic. Assuming one uses projective coordinates modular multiplication remains as the most time consuming operation for ECC. Hence, efficient implementation relies on efficient modular multiplication, as is the case for RSA. Nevertheless, it is also important to focus on time-constant algorithms which are less likely to leak side-channel information. To conclude, in this work we aimed to introduce a secure combined RSA-ECC implementation which as well meets high demands in speed implied by state of art for RSA hardware implementation. See for example [8].

The remainder of this paper is organized as follows. Section 2 gives a survey of previous implementations of public-key algorithms in hardware relevant for our work. In Section 3, we outline the architecture of the targeted implementation platform. Section 4 describes new options for point operations. In Section 5, the implementation results and timings are given. Section 6 introduces a new variant of Garner's scheme for CRT which is as well efficient but more resistant to side-channel attacks. Implications of the proposed changes on security of both RSA and ECC are considered in Section 7. Sections 8 concludes the paper.

## 2 Related Work

This section reviews some of the most relevant previous work in hardware implementations for PKC. The vast majority of published work that is considering implementations of PKC deals with software platforms. Some of the work is done on FPGAs and only very few implementations are presenting an ASIC implementation of ECC in the field of prime characteristic. Most of the work is done in binary field and some authors have considered dual field implementations i.e. ECC in prime and binary field.

Goodman and Chandrakasan proposed a domain-specific reconfigurable cryptographic processor (DSRCP) in [8]. The DSRCP performs a variety of algorithms ranging from modular integer arithmetic to elliptic curve arithmetic. They mainly discussed the arithmetic in binary field. Most recent published work is the one of Satoh and Takano [27]. They presented the dual field multiplier with the best performance so-far in both type of fields. The throughput of EC scalar multiplication is maximized by use of Montgomery multiplier and on-the-fly redundant binary converter. The great quality of their design is in scalability in operand size and also flexibility between speed and hardware area. Another hardware solution for both types of fields was presented by Wolkerstorfer in [31]. The author introduced low power design which features short critical path to enable high clock frequencies. Most operations are executed within a single clock cycle and the redundant number representation was used. The idea of unified multiplier was first introduced by Savaş et al. in [28]. The authors have discussed a scalable and unified architecture for a Montgomery multiplication module. They deployed an array of word size processing units organized in a pipeline. The same idea is the basis of work in Grosschädl [9]. The bit-serial multiplier which is introduced is performing multiplications in both types of fields. The author also modified the classical MSB-first version for iterative modular multiplication. All concepts are introduced in detail, but the actual VLSI implementation is not given. Some hardware implementations in $GF(p)$ on FPGA are also known. The ECC-only processor over fields GF($p$) was proposed by Orlando and Paar [21]. They proposed so-called Elliptic Curve Processor (ECP) which is scalable in terms of area and speed. The ECP is also best suited for projective coordinates and it is using a new type of high-radix precomputation-based Montgomery multiplier. The scalability of the multiplier to larger fields was also verified in the field whose size is 521 bits. The authors have estimated eventual timing of 3 $ms$ for computing one point multiplication in 192-bit prime field. Örs et al discussed an ECC-processor which is optimized for MMM in [23]. They described an efficient implementation of an elliptic curve processor over $GF(p)$. The processor can be programmed to execute a modular multiplication, addition/subtraction, multiplicative inversion, EC point addition/doubling and multiplication. A detailed overview of hardware implementations for PKC is given in [4] .

Still plenty of the work in ECC over GF($p$) deals with software implementations, where there exist many hardware implementations over binary field. It appears that the arithmetic in characteristic 2 is easier to implement and area

and power consumption are smaller than in the case of $GF(p)$. This is believed to be true, but only for platforms where specialized arithmetic coprocessors for finite field arithmetic are not available. On the other hand, an advantage of prime field is in its suitability for both RSA and ECC with an a resourceful sharing of hardware.

## 3 Previous Work and Background

In this paper we discuss how an FPGA implementation of Montgomery multiplication that was originally designed for RSA can efficiently be used to perform prime field ECC operations. This design consists of a Large Modular Montgomery Multiplier (MMM), designed as a systolic array. This array is one-dimensional and consists of a fixed number of Processing Cells (PCs). The MMM performs Montgomery modular multiplication that consists of the following operation: $Mont(X, Y) = XYR^{-1} \mod N$.

In the remainder we call $AR \bmod N$ the Montgomery representation of $A$. For modular exponentiation with the MMM all intermediate results are in this form. A number can be transformed to its Montgomery representation by performing a Montgomery multiplication of that number with $R^2 \bmod N$. For the transformation from Montgomery representation to the normal form a Montgomery multiplication with 1 will suffice.

### 3.1 Systolic Array

Figure 1 shows a schematic of the systolic array that was implemented in the MMM. A PC contains adders and multipliers that can process $\alpha$ bits of $X$ and $\beta$ bits of $Y$ in one clock cycle. Here $X$ and $Y$ are the multiplicand and multiplier. Each PC calculates $\frac{T_j + x_i y_j + m_i N_j}{2^\alpha}$ in each clock cycle. The detailed description is given in Section 4.4.

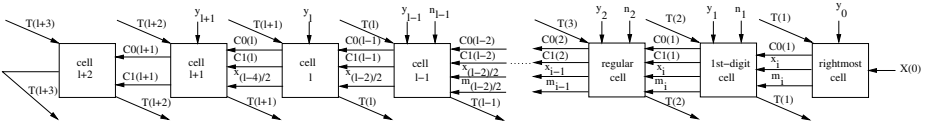



**Fig. 1.** Schematic of the Modular Montgomery Multiplier.

In the original notation of Montgomery after each multiplication in the exponentiation algorithm a reduction was needed [19]. The input had the restriction $X, Y < N$ and the output $T$ was bounded by $T < 2N$. The result of this is that in the case $T > N$, $N$ must be subtracted so that the output can be used as input of the next multiplication. To avoid this subtraction a bound for $R$ can

be calculated such that for inputs $X, Y < 2N$ also the output is bounded by $T < 2N$.

In [3] the need of avoiding this reduction after each multiplication is addressed. In practice this means that the output of the multiplication can be directly used as an input of the next Montgomery multiplication. The following theorem is proven in [3].

**Theorem 1.** *The result of a Montgomery multiplication* $XYR^{-1} \bmod N < 2N$ *when* $X, Y < 2N$ *and* $R > 4N$.

The final round in the modular exponentiation is the conversion to the integer domain, i.e. calculating the Montgomery multiplication of the last result and 1. The same arguments prove that this final step remains within the following bound: $Mont(T, 1) \leq N$. In practice, $A^B \bmod N = N$ will never occur since $A \neq 0$.

### 3.2 ECC Processor

The MMM need not only be used for fast RSA implementation but also for ECC point operations in the prime field. Due to the scalability of the design, the FPGA architecture can perform both, i.e. efficient exponentiations on large operands (for RSA) and modular multiplication on the smaller ECC operands. In Figure 2 a schematic of an FPGA implementation for ECC is given. One or two MMMs are used to perform the modular (Montgomery) multiplications. A Large Number Co-Processor (LNCP) is added to the design to perform the additions and subtractions. These units have their own RAM's and are connected with a data bus.

As already explained, the performance of an elliptic curve cryptosystem is primarily determined by the efficient realization of the arithmetic operations (addition, multiplication and inversion) in the underlying finite field. If projective coordinates are used the inversion operation becomes negligible. Therefore, coprocessors for elliptic curve cryptography are primarily designed to accelerate the field multiplication. Considering multiplication in the prime field i.e., $GF(p)$, the whole work which is done for the RSA implementation is relevant. The only difference is that shorter bit-lengths are used i.e., 160-300 bits. Scalability is again a point of concern and even more inter operability between different implementations.

## 4 New Implementation

In this section we present our FPGA implementation for ECC point operations for prime fields.

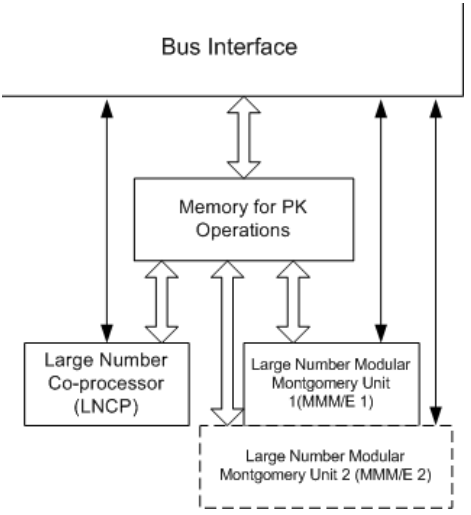



**Fig. 2.** Schematic of the Modular Montgomery Multiplier.

### 4.1 Point Addition

Point addition and doubling can be performed according to the algorithm given in [5].

Here we assume that the two points that will be added i.e., $P = (X_1, Y_1, Z_1)$ and $Q = (X_2, Y_2, Z_2)$ are already transformed to the Projective coordinates and Montgomery representation. The result point $R = P + Q = (X_3, Y_3, Z_3)$

**Scheduling of point addition.** Point addition can be even performed more efficient if two MMM units are used. The operations can be conveniently divided between the two units. (Modular) addition and subtraction will be done on a Large Number Co-processor. Those operations can be performed in the same time as the Montgomery multiplication. The following scheduling as shown in Table 1 can be used. Table 1 shows that the performance can almost be doubled by using two MMM units.

### 4.2 Point Doubling

Here we discuss a special case of point addition i.e. point doubling, where the points $P$ and $Q$ are respectively given as: $P = (X_1, Y_1, Z_1)$ and $R = 2P = (X_3, Y_3, Z_3)$.

**Scheduling of point doubling.** In Table 2 a possible schedule for point doubling over the 2 MMMs and the LNCP is given .

The difficulty in the scheduling of point doubling lies in the operations scheduled in MMM2 and the LNCP, which are all depending on the answer of the previous operation.

**Table 1.** Scheduling of point addition.

| MMM1 | MMM2 | LNCP |
|---|---|---|
| $Z_2{}^2$ | $Z_1{}^2$ | |
| $\lambda_1 = X_1 Z_2{}^2$ | $\lambda_2 = X_2 Z_1{}^2$ | |
| $Z_2{}^3$ | $Z_1{}^3$ | $\lambda_3 = \lambda_1 - \lambda_2$ |
| | | $\lambda_7 = \lambda_1 + \lambda_2$ |
| $\lambda_4 = Y_1 Z_2{}^3$ | $\lambda_5 = Y_2 Z_1{}^3$ | |
| | $\lambda_3{}^2$ | $\lambda_6 = \lambda_4 - \lambda_5$ |
| | | $\lambda_8 = \lambda_4 + \lambda_5$ |
| $\lambda_6{}^2$ | $\lambda_7 \lambda_3{}^2$ | |
| $\lambda_3{}^3$ | $Z_1 Z_2$ | $X_3 = \lambda_6{}^2 - \lambda_7 \lambda_3{}^2$ |
| | | $\lambda_9 = \lambda_7 \lambda_3{}^2 - 2X_3$ |
| $\lambda_8 \lambda_3{}^3$ | $\lambda_9 \lambda_6$ | |
| $Z_3 = Z_1 Z_2 \lambda_3$ | | $Y_3 = \frac{\lambda_9 \lambda_6 - \lambda_8 \lambda_3{}^3}{2}$ |

**Table 2.** Scheduling of point doubling.

| MMM1 | MMM2 | LNCP |
|---|---|---|
| $3X_1{}^2$ | $Z_1{}^2$ | |
| $Y_1{}^2$ | $Z_1{}^4$ | |
| $\lambda_2 = 4X_1 Y_1{}^2$ | $aZ_1{}^4$ | |
| $\lambda_3 = 8Y_1{}^4$ | | $\lambda_1 = 3X_1{}^2 + aZ_1{}^4$ |
| | $\lambda_1{}^2$ | |
| $Z_3 = 2Y_1 Z_1$ | | $X_3 = \lambda_1{}^2 - 2\lambda_2$ |
| | | $\lambda_2 - X_3$ |
| | $\lambda_1(\lambda_2 - X_3)$ | |
| | | $Y_3 = \lambda_1(\lambda_2 - X_3) - \lambda_3$ |

Point multiplication can be implemented as a repeated combination of point addition and point doubling.

### 4.3 Modular Addition and Subtraction

Modular (i.e. Montgomery) multiplication, modular addition and modular subtraction are the basic operations for point addition. MMM is performed on our highly scalable Montgomery based multiplier. Modular addition and modular subtraction can be implemented as a repeated addition. However, the number of additions/subtractions would be data dependent. Let us take a better look at these two operations. As proven in Section 3.1, the result of an operation on our multiplier will always be smaller than twice the modulus ($2N$). All modular additions and subtractions in the point addition scheme are with two outputs of the Montgomery multiplier.

For example:

$$\begin{aligned} \lambda_1 &= X_1 {Z_2}^2 < 2p \quad \text{and} \quad \lambda_2 = X_2 {Z_1}^2 < 2p \\ \lambda_3 &= \lambda_1 - \lambda_2 \bmod p \\ \lambda_7 &= \lambda_1 + \lambda_2 \bmod p \end{aligned} \tag{1}$$

The result of the modular addition and subtraction is again the input of another Montgomery multiplication and can therefore be larger than the modulus but should be positive. If it would be possible to calculate the previous calculations as "normal" i.e. non-modular addition and subtraction, this would make the operations very efficient but more importantly time constant.

Keeping in mind the "$2p$" bound for the operands as a result of the bound for the Montgomery parameter, we get:

$$\begin{aligned} 0 < \lambda_1 + \lambda_2 &< 4p - 1 \\ 0 < \lambda_1 + 2p - \lambda_2 &< 4p \end{aligned} \tag{2}$$

Our target is now to try to fix a bound for the Montgomery parameter such that we can use these non-modular addition and subtraction instead of the

modular forms. To achieve this we must ensure that the inputs $X$ and $Y$ of the Montgomery multiplier that are smaller than $4p$ result in a Montgomery product that is smaller than $2p$.

As already mentioned, in the original implementation of our MMM the inputs of a Montgomery multiplication should be smaller than $2p$. We will use the following lemma.

**Lemma 1.** *If the Montgomery parameter $R$ satisfies the following inequality $R > 16N$ , then for inputs $X, Y < 4N$ the result $T$ of the MMM will satisfy: $T < 2N$ (as required).*

**Proof:** The Montgomery multiplication as implemented in the MMM calculates the following:

$$T = \frac{AB + mN}{R} = \frac{AB}{R} + \frac{m}{R}N \tag{3}$$

here $m$ is calculated modulo $R$. Filling in the bounds for the inputs and $R > 16N$ we get

$$T = \frac{AB}{R} + \frac{m}{R}N < \frac{4N \cdot 4N}{R} + N \leq 2N. \tag{4}$$

If $n$ is the length of modulus $N$ in bits then the following is valid: $N < 2^n$ and $16N < 2^{n+4}$. With $R = 2^r$, we get $r \geq n + 4$. □

We have shown that for all modulus lengths, inputs smaller than $4p$ will result after a Montgomery multiplication on the MMM in a value which is smaller than $2p$. Therefore we can use the more efficient and time constant implementation of modular addition and subtraction. Furthermore, there is no any loss in efficiency caused by this enlarged bound because $R$ is usually already bigger than this bound (especially for $\alpha, \beta > 1$.)

### 4.4 Montgomery Modular Multiplication

The processing cells in the systolic array shown in Figure 1 performs Equation 5. $x_i$ and $m_i$ have $\alpha$ bits, $y_j$ and $n_j$ have $\beta$ bits. $c1_{i,j}$ and $c0_{i,j}$ denote the carry chain on the array. Because the critical path of the systolic array is the same as the critical path of one PC, the clock frequency of the Montgomery multiplier will be the same for all bit-lengths. This property gives the advantage of using the circuit for RSA and ECC.

$$2^2 \times c1_{i,j} + 2 \times c0_{i,j} + t_{i,j} = t_{i-1,j+1} + x_i \times y_j + m_i \times n_j + 2 \times c1_{i,j-1} + c0_{i,j-1} \tag{5}$$

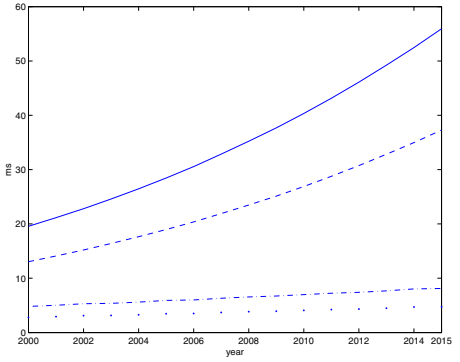
Parameters $\alpha$ and $\beta$ are 4 for this implementation. Table 3 shows the performance of the FPGA implementation of the Montgomery multiplier. Parameter $n$ is the bit-length of $N$, $l$ in Figure 1 is $\frac{n}{4}$.

**Table 3.** The performance of the FPGA implementation of the Montgomery multiplier.

| | |
|---|---|
| Number of clock cycles | $3\frac{n}{4}+7$ |
| Clock period | 19 ns |
| Clock frequency | 53 MHz |
| Total latency | $14.25n + 133$ ns |
| Number of gates | 4547 |

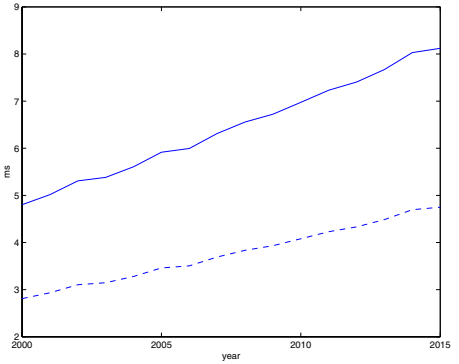
## 5 Results and Timings

In the work of Lenstra and Verheul [16], the authors made a security comparison between RSA and ECC key lengths. They introduced a table that included corresponding key bit-lengths assuring minimal security in the years to come for the two Public Key systems. In Figure 3 the performances for ECC and RSA are given according to the key sizes that were given in their paper. The figures show also that especially for the future applications the performance of ECC is more attractive than the performance of RSA.




— RSA with 1 MMM, – RSA with 2 MMMs, -. ECC with 1 MMM, .. ECC with 2 MMMs

**Fig. 3.** Performance of RSA and ECC.




— ECC with 1 MMM, – ECC with 2 MMMs

**Fig. 4.** Performance ECC with 1 or 2 MMM.

Figure 4 shows the performance for an ECC implementation with one and two MMMs. The implementation with 2 MMMs is scheduled according to the schedule given in Table 1 and Table 2. Figure 4 shows a speed-up of a factor of 2 for the two MMMs variant.

For the sake of preciseness we give detailed performance results in the Table 4.

**Table 4.** RSA and ECC performance in $ms$ at 53 MHz.

| year [16] | RSA | | | ECC | | |
|---|---|---|---|---|---|---|
| | size | 1 MMM | 2 MMM | size | 1 MMM | 2 MMM |
| 2002 | 1024 | 22.8 | 15.2 | 139 | 5.3 | 3.1 |
| 2014 | 1536 | 52.5 | 35 | 172 | 8 | 4.7 |
| 2023 | 2048 | 90.6 | 60.4 | 197 | 10.5 | 6.1 |
| 2051 | 4096 | 350.9 | 234 | 275 | 11.4 | 6.7 |

## 6 Side-Channel Security of CRT

We will now briefly review some benefits of Montgomery's Multiplication Method, which are also evident for CRT implementations. In [11,29], $R > 4N$ is proposed which, with some savings in hardware, omits completely all reduction steps.

Especially implementations of CRT schemes are found to be very sensitive to side-channel attacks. For example, recently a new SPA-based attack was introduced by Novak [20], which is targeting the algorithm of Garner [17]. This scheme is often used in all sorts of applications, including smartcards. It is usually implemented as follows:

**Algorithm 1.** Garner's algorithm for CRT

INPUT: ciphertext c, $N = p \cdot q$, $(p > q)$ and precomputed values $d_1 \equiv d \bmod (p-1), d_2 \equiv d \bmod (q-1)$ and $U = p^{-1} (\bmod q)$ $(C_1 \equiv C \,(\bmod p)$, $C_2 \equiv C \,(\bmod q))$

OUTPUT: $R = M \equiv t + q \cdot (s \cdot (q^{-1} \bmod p)) \bmod p$

1. $s = M_p \equiv {C_1}^{d_1} \bmod p$,
2. $t = M_q \equiv {C_2}^{d_2} \bmod q$
3. $x = (s - t) (\bmod p)$
4. $R = t + q \cdot ((x \cdot U) (\bmod p))$

The third step is the critical one. Novak observed that if the modular subtraction is implemented in the common way it may leak information. More precisely, to perform subtraction (mod $p$) one has to check the sign of $s - t$ and conditionally add $p$ if $s - t < 0$ ($p > q$ is required). Novak managed to build a successful attack based on this observation. An implementation of the above algorithm can produce the optional pattern in a power trace as a result of the conditional addition.

We propose the following solution. Instead of the subtraction $\bmod p$, one can compute the following:

$$x = s + p - t. \tag{6}$$

For $p > q$ the result stays within the following bounds $0 < x < 2p$ which can be handled easily if Step 4 is implemented by use of the algorithm of Montgomery. Namely, the algorithm as proposed in [11,29] for Montgomery modular multiplication takes two inputs $0 < X, Y < 2p$ and the result is also within the same interval, if the proper bound for Montgomery parameter $R$ is chosen. This result is converted from the Montgomery domain to the usual domain by a Montgomery multiplication with 1. Changing Garner's scheme in this way the algorithm is always performing a constant execution path. We prove this in more detail.

*Claim.* The result of $x = s + p - t$ is always smaller than $2p$, for the parameters $s, p, t$ defined as above, i.e. $s = M_p \equiv {C_1}^{d_1} \bmod p$, $t = M_q \equiv {C_2}^{d_2} \bmod q$ and $p > q$.

**Proof:** It is shown that with the use of Montgomery's algorithm and $R > 4N$, the final result of modular exponentiation is bounded by the modulus $N$. (See Theorem 1.)

Now, we can prove the claim. It is obvious that $0 \leq s < p$ and $0 \leq t < q$. We assume $p > q$ with which this proof does not loose its generality, the other case is almost the same. Then we get $x = s + p - t < p + p - 0 = 2p$. Hence, if the multiplication in the Algorithm 1 is implemented as the one of Montgomery, no conditional subtraction is required as in original algorithm. This concludes the proof. □

## 7 Security Remarks

In this section we address side-channel security i.e. resistance to timing [14], [10] and power analysis based attacks [15]. These types of attacks, together with fault-analysis based attacks [6], [12], [2] electromagnetic analysis attacks (EMA) [25], [7] and other physical attacks such as probing attacks [1] are a major concern especially for wireless applications. Mainly because of space limitation we only briefly discuss the first two, which are also believed to be the most practical .

Namely, computations performed in non-constant time i.e. computations which are time-dependent on the values of the operands, may leak secret key information. This observation is the basis for timing attacks. On the other hand, power analysis based attacks use the fact that the power consumed at any particular time during a cryptographic operation is related to the function being performed and data being processed. The attack can be usually performed easily because smartcards, for example receive the power externally and an attacker can easily get to hold on the source of this side-channel information.

In our implementation all modular reductions are excluded. The weaknesses in the conditional statements of the algorithm (used for realization of the reduction step) are time variations and therefore these should be omitted. By use of an optimal upper bound the number of iterations required in the algorithm based on Montgomery's method of multiplication can be reduced [30]. Another timing

information leakage that was observed by Quisquater [10] et al. and Walter [30] was the timing difference between "square" and "multiply". This information can be used to attack RSA, even advanced exponentiation methods were used. In our architecture, this weakness is removed, because the same systolic array is performing squarings and multiplications, which are therefore indistinguishable with respect to timing.

Besides that, when considering power analysis attacks, some other precautions have also been introduced. The fact that all of the PCs operate in parallel makes these types of attacks far less likely to succeed. Both, RSA and ECC can benefit from this fact.

As already mentioned, this architecture can be an option for wireless devices, although we have chosen here to introduce a network security devoted product. Again, because of space limitation we were not able to discuss the smaller, compact implementation but that also features very secure low-power design with attractive performances.

Örs et al characterized the power consumption of a XILINX Virtex 800 FPGA in [24]. They showed that it is possible to draw conclusions about vulnerability of an ordinary ASIC in CMOS technology by performing power-analysis attacks on an FPGA-implementation. With respect to this, an FPGA design can serve as a good model for ASIC platform not just for usual hardware related properties but also for security.

## 8 Conclusions

We have presented the hardware implementation on systolic array architecture that is scalable in all parameters and ideally suitable for RSA and ECC algorithms.

We have also introduced a bound on Montgomery parameter $R$, which allows us to perform the most efficient point addition and doubling for ECC, as well as modular exponentiation. Even in the case of CRT the Montgomery's algorithm is proven to be the best option for side-channel resistance.

## References

1. R. Anderson and M. Kuhn. Low cost attacks on tamper resistant devices. In B. Christianson *et al.*, editor, *Proceedings of 1997 Security Protocols Workshop*, number 1361 in Lecture Notes in Computer Science, pages 125–136. Springer-Verlag, 1997.
2. C. Aumüller, P. Bier, W. Fischer, P. Hofreiter, and J.-P. Seifert. Fault attacks on RSA with CRT: Concrete results and practical countermeasures. In B. S. Kaliski Jr., Ç. K. Koç, and C. Paar, editors, *Proceedings of Cryptographic Hardware and Embedded Systems (CHES 2002)*, Lecture Notes in Computer Science, 2002.
3. L. Batina and G. Muurling. Montgomery in practice: How to do it more efficiently in hardware. In B. Preneel, editor, *Proceedings of RSA 2002 Cryptographers' Track*, number 2271 in Lecture Notes in Computer Science, pages 40–52, San Jose, USA, February 18–22 2002. Springer-Verlag.

4. L. Batina, S. B. Örs, B. Preneel, and J. Vandewalle. Hardware architectures for public key cryptography. *Elsevier Science Integration the VLSI Journal*, 34, 2003.
5. I. Blake, G. Seroussi, and N. P. Smart. *Elliptic Curves in Cryptography.* London Mathematical Society Lecture Note Series. Cambridge University Press, 1999.
6. D. Boneh, R. A. DeMillo, and R. J. Lipton. On the importance of checking cryptographic protocols for faults (extended abstract). In W. Fumy, editor, *Advances in Cryptology: Proceedings of EUROCRYPT'97*, number 1233 in Lecture Notes in Computer Science, pages 37–51. Springer-Verlag, 1997.
7. K. Gandolfi, C. Mourtel, and F. Olivier. Electromagnetic analysis: Concrete results. In Ç. K. Koç, D. Naccache, and C. Paar, editors, *Proceedings of Cryptographic Hardware and Embedded Systems (CHES 2001)*, number 2162 in Lecture Notes in Computer Science, pages 255–265, 2001.
8. J. Goodman and A. P. Chandrakasan. An energy-efficient reconfigurable public-key cryptography processor. *IEEE Journal of Solid-State Circuits*, 36(11):1808–1820, November 2001.
9. J. Grosschädl. A bit-serial unified multiplier architecture for finite fields GF($p$) and GF($2^n$). In Ç. K. Koç, D. Naccache, and C. Paar, editors, *Proceedings of Cryptographic Hardware and Embedded Systems (CHES 2001)*, number 2162 in Lecture Notes in Computer Science, pages 206–223, 2001.
10. G. Hachez, F. Koeune, and J.-J. Quisquater. Timing attack: what can be achieved by a powerful adversary? In A. Barbé, E. C. van der Meulen, and P. Vanroose, editors, *Proceedings of the 20th symposium on Information Theory in the Benelux*, pages 63–70, May 1999.
11. G. Hachez and J.-J. Quisquater. Montgomery exponentiation with no final subtractions: Improved results. In Ç. K. Koç and C. Paar, editors, *Proceedings of Cryptographic Hardware and Embedded Systems (CHES 2000)*, number 1965 in Lecture Notes in Computer Science, pages 293–301, 2000.
12. M. Joye, A. K. Lenstra, and J.-J. Quisquater. Chinese remaindering based cryptosystem in the presence of faults. *Journal of Cryptology*, 4(12):241–245, 1999.
13. N. Koblitz. Elliptic curve cryptosystem. *Math. Comp.*, 48:203–209, 1987.
14. P. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS and other systems. In N. Koblitz, editor, *Advances in Cryptology: Proceedings of CRYPTO'96*, number 1109 in Lecture Notes in Computer Science, pages 104–113. Springer-Verlag, 1996.
15. 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.
16. A. Lenstra and E. Verheul. Selecting cryptographic key sizes. In H. Imai and Y. Zheng, editors, *Proceedings of Third International Workshop on Practice and Theory in Public Key Cryptography (PKC 2000)*, number 1751 in Lecture Notes in Computer Science, pages 446–465. Springer-Verlag, 2000.
17. A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1997.
18. 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.
19. P. Montgomery. Modular multiplication without trial division. *Mathematics of Computation*, Vol. 44:519–521, 1985.

20. R. Novak. SPA-based adaptive chosen-ciphertext attack on RSA implementation. In D. Naccache and P. Pallier, editors, *Proceedings of 5th International Workshop on Practice and Theory in Public Key Cryptosystems (PKC 2002), Paris, France, February 2002*, number 2274 in Lecture Notes in Computer Science. Springer-Verlag, 2002.
21. G. Orlando and C. Paar. A scalable GF($p$) elliptic curve processor architecture for programmable hardware. In Ç. K. Koç, D. Naccache, and C. Paar, editors, *Proceedings of Workshop on Cryptograpic Hardware and Embedded Systems (CHES 2001)*, number 2162 in Lecture Notes in Computer Science, pages 356–371, Paris, France, May 14-16 2001. Springer-Verlag.
22. S. B. Örs, L. Batina, B. Preneel, and J. Vandewalle. Hardware implementation of a Montgomery modular multiplier in a systolic array. In *The The 10th Reconfigurable Architectures Workshop (RAW)*, Nice, France, April 22 2003.
23. S. B. Örs, L. Batina, B. Preneel, and J. Vandewalle. Hardware implementation of an elliptic curve processor over GF($p$). In *IEEE 14th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, The Hague, The Netherlands, June 24–26 2003.
24. S. B. Örs, E. Oswald, and B. Preneel. Power-analysis attacks on an FPGA – first experimental results. In C. Walter, Ç. K. Koç, and C. Paar, editors, *Proceedings of Cryptographic Hardware and Embedded Systems (CHES 2003)*, Lecture Notes in Computer Science, pages 35–50, Cologne, Germany, September 7–10 2003.
25. J. J. Quisquater and D. Samyde. Elecromagnetic analysis EMA: Measures and coutermeasures for smart cards. In *in Smart Card Programming and Security*, number 2140 in Lecture Notes in Computer Science, pages 200–210. Springer-Verlag, 2001.
26. 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.
27. A. Satoh and K. Takano. A scalable dual-field elliptic curve cryptographic processor. *IEEE Transactions on Computers, special issue on cryptographic hardware and embedded systems*, 52(4):449–460, April 2003.
28. E. Savaş, A. F. Tenca, and Ç. K. Koç. A scalable and unified multiplier architecture for finite fields GF($p$) and GF($2^m$). In C. Paar and Ç. K. Koç, editors, *Proceedings of Cryptographic Hardware and Embedded Systems (CHES 2000)*, number 1965 in Lecture Notes in Computer Science, pages 281–296. Springer-Verlag, 2000.
29. C. D. Walter. Precise bounds for Montgomery modular multiplication and some potentially insecure RSA moduli. In B. Preneel, editor, *Proceedings of Topics in Cryptology- CT-RSA 2002*, number 2271 in Lecture Notes in Computer Science, pages 30–39, 2002.
30. C. D. Walter and S. Thompson. Distinguishing exponent digits by observing modular subtractions. In D. Naccache, editor, *Proceedings of Topics in Cryptology - CT-RSA*, number 2020 in Lecture Notes in Computer Science, pages 192–207, San Francisco, 8-12 April 2001. Springer-Verlag.
31. J. Wolkerstorfer. Dual-field arithmetic unit for $GF(p)$ and $GF(2^m)$. In B. S. Kaliski Jr., Ç. Koç, and C. Paar, editors, *Proceedings of Cryptographic Hardware and Embedded Systems (CHES 2002)*, Lecture Notes in Computer Science, Redwood Shores, CA, USA, August 13–15 2002. Springer-Verlag.