MicroACP - A Fast and Secure Reconfigurable Asymmetric Crypto-Processor

–Overhead Evaluation of Side-Channel Countermeasures–

Christopher Pöpper[1], Oliver Mischke[2], and Tim Güneysu[2]

[1] ESCRYPT GmbH - Embedded Security
[2] Horst Görtz Institute for IT Security, Ruhr University Bochum, Germany
christoph.poepper@escrypt.com, {mischke,gueneysu}@crypto.rub.de

Abstract. In this work we present a lightweight co-processor for asymmetric cryptography. While focusing on standardized elliptic curve cryptography over prime fields, the architecture has been chosen generic enough to also allow to perform RSA operations on the same hardware. Compared to previous work our processor distinguishes itself by not only having on par performance with recent work in this field, but also by being able to additionally apply state of the art side-channel analysis countermeasures to protect the implementation against timing and power analysis attacks. Different countermeasures can be dynamically selected at runtime, allowing a flexible trade-off between security and performance. Utilizing a specialized 32-bit ALU and a microcode-based control unit, it is possible to easily reprogram the controller after deployment allowing to make changes to the implemented algorithm or countermeasures by updating the microcode. This allows to keep some of the reconfigurability of FPGA-based designs even when fabricating the proposed core as an ASIC.

1 Introduction

Computing is no longer restricted to powerful mainframes or personal computers. Nowadays almost everyone carries a smartphone which is more powerful then most computers a few years back. But not only users actively interact, the Internet of Things becomes more and more a reality, where different devices autonomously communicate which each other. With the increased communication rises also the need for reliable asymmetric cryptographic primitives to provide the necessary security. This is especially true in the vehicle-2-vehicle communication where based on messages by other cars automatic actions might be performed, for example emergency brakes to prevent collisions.

Rivest-Shamir-Adleman (RSA) [9] was for a long time the algorithm of choice for asymmetric cryptography and still provides some advantages like fast verification times by using small public exponents. In case both the signing and verification of messages is needed, Elliptic Curve Cryptography (ECC) [5,7] is a better solution since it provides equal security using shorter operands which reduces not only computation time but also signature size.

D. Goehringer et al. (Eds.): ARC 2014, LNCS 8405, pp. 240–247, 2014.

While these algorithms remain secure from a mathematical point of view, they can easily be attacked by so called side-channel attacks if no precautions are taken. In the late 90s and early 2000s several ways have been discovered to extract secret information from a circuit by observing e.g., timing behaviour, power consumption, or electromagnetic emanation. Most notably is the work of Kocher *et al.* introducing differential power analysis [6]. Since then there exists an arms race between designers of countermeasures and attackers where the outcome is still open. Fan *et al.* [3] gives a good overview of the current state of the art in ECC countermeasures.

Implementations of pure ECC or combined ECC-RSA processors have been an ongoing research topic for some time. Many implementation techniques have been evaluated and fill different niches. As example, Batina *et al.* [1] proposed the design of an ECC and RSA processor based on systolic arrays. While this method is inherently protected against some attacks because of the static operation flow, it requires a high amount of logic resources and is not protected against e.g., differential power analysis. In [4] the authors focus on achieving a very high throughput by utilizing a large number of FPGA hard macros like DSPs and BRAMs, which makes the design less scalable and leads to high area requirements. [12] and later [11] on the other hand proposed designs based on a so called microcode architecture where a small ALU is controlled by a flexible state machine which can be easily reprogrammed by changing the microcode.

This work also uses the microcode approach aiming for a small implementation footprint and high reconfigurability. In addition our implementation is significantly stronger protected against side-channel attacks. We have implemented several countermeasures and evaluate the performance overhead. Furthermore, the designed ALU is more flexible being able to not only perform ECC but also RSA operations while maintaining a similar performance as in [11].

The remaining article is organized as follows: Section 2 describes our design architecture and the implemented countermeasures. Performance results together with a comparison to recent work is given in Section 3. This section also states the performance overhead of the chosen countermeasures and combinations. Finally, Section 4 concludes our research.

2 Our Design

In this work we are aiming to implement a design which is capable of performing both ECC and RSA operations using the same logic. We have chosen to build our design on the ideas of [12] and [11] utilizing the microcode approach. This means that a small but powerful ALU is controlled by a dedicated tiny processor which controls the program flow and the memory management based on stored opcodes in the program memory. By updating the program memory it is possible to easily update algorithms or implemented countermeasure even after deployment in the field. This gives us the highest flexibility for a very efficient ALU on a small footprint. Beside choosing an efficient architecture to perform both ECC and RSA operations, our focus is mainly on secure implementations of those by implementing several countermeasures against side-channel attacks.

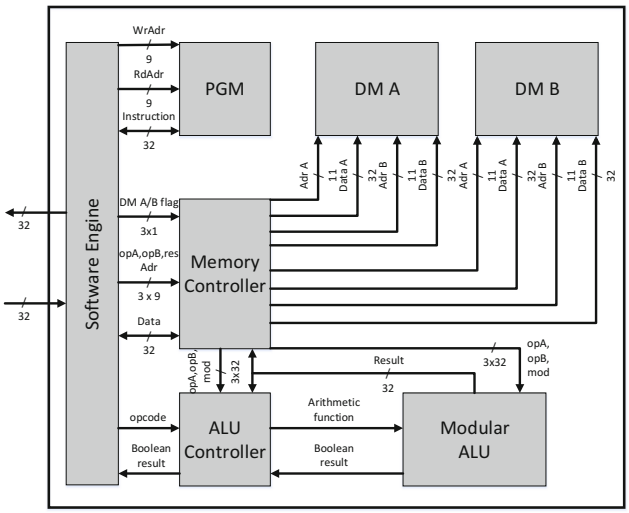

Fig. 1. Architecture overview

Our solution, the **MicroACP** is a cryptographic co-processor which implements ECC over prime fields and generic RSA exponentiations with a maximum operand size of 2048 bits. For the ECC part, the core is primarily designed to work with the NIST prime curves, especially *NIST-P256*. If lower security is sufficient, *NIST-P224*, or even *NIST-P192* can be chosen using exactly the same hardware. It is possible to switch between different curves at runtime by updating the internal program code, but for the remainder of this paper we are focusing on *NIST-P256* for high security applications.

2.1 Architecture

Figure 1 depicts the architecture of the **MicroACP**. The program memory (PGM) stores the necessary algorithms for the execution of the secure ECC and RSA operations. It can either be preloaded by the bitstream or loaded via the software interface during runtime. This also allows a flexible update of the code to either support different ECC curves or countermeasures. The ALU itself utilizes a 32-bit datapath for all operations. Four hardware multipliers are used and the ALU is also able to perform addition, subtraction, compare and reduction operations. The operands and the results of ALU operations are stored in two true dual port block rams (BRAM). The interaction between the memories and the ALU is handled by the memory and the ALU controller.

2.2 Implemented Algorithms

For the fast and unprotected version of the ECC we implemented the Double-And-Add (DAA) algorithm, which is also called *Left-to-right binary method for*

point multiplication. Furthermore, for an efficient computation we chose the Jacobian projective coordinates for the required point additions and doublings which results in an better overall-performance than standard affine coordinates.

Since all of the computations are made in prime fields, a reduction algorithm is required. Due to our choice of the *NIST-P256* curve, which standardized a so called pseudo-Mersenne prime number as modulus, the required reduction can be efficiently computed with simple additions and subtractions.

Similar to the ECC we choose the Square-and-Multiply (SAM) algorithm for the implementation of the modular exponentiation, used in the RSA crypto system. In contrast to ECC operations, we have to deal with random moduli for the RSA, which requires a generic reduction algorithm. For our design we implemented the Montgomery Reduction [8] which is a well-known and efficient technique.

2.3 Implementation of Countermeasures

To defeat side-channel attacks a set of countermeasures against SPA and DPA attacks on ECC were chosen. According to the work from Fan *et al.* [3] due to the usage of the following presented countermeasures the implementation is secure against all known SPA attacks and against the traditional DPA as well as against the RPA (Refined Power Analysis) and the ZPA (Zero-Value Point Analysis) attack.

All these countermeasures are not exclusive but can be combined in order to fulfill high security requirements or to choose a trade-off between security and performance.

Double-and-Add-Always. To defeat SPA attacks, J. Coron proposed in 1999 an alternative version of the DAA algorithm [2]. The principle is that an addition takes place in every loop iteration, either with the real output registers or with an unrelated dummy register. Consequently, a dependency of the scalar k and the runtime of the algorithm is prevented.

Scalar Randomization. In [2], beside the DAAA, also some DPA countermeasure are presented. The first is the *Scalar Randomization*, or also known as *Coron's first Countermeasure*. The idea is that the secret scalar k is randomized or masked. This is done by adding a random multiple of the order $\#E$ to the scalar: $r \cdot \#E + k \cdot P$. Due to this tampering of k at each ECC execution, the dependency between the secret key and the operations on the chip, and consequently the leakage of information about k is prevented.

Point Blinding. *Coron's second Countermeasure* [2] blinds or masks the input point instead of the scalar. Since in every scalar multiplication the point is blinded by adding a secret random point, the leakage of information about the scalar, and therefore the attack target, is prevented.

Randomized Projective Coordinates. The third countermeasure that was proposed by Coron in [2] is working on the projective, in our implementation on the Jacobian, coordinate representation. The principle is again a randomization

of the point P. For this task the affine point P is converted to Jacobian representation. But instead of setting $Z = 1$, we use $Z = r$, where r is a random number. It is obvious, that we have to compute X and Y according to r. Due to this translation the coordinates do not have a dependency to the hypothetical values which are computed by the attacker. Since this is an appropriate representation of the input point, the inverse conversion to the affine coordinates after the multiplication returns the correct result.

Square-and-Multiply-Always. To defeat SPA attacks on the RSA computation, a dependency between the runtime and the exponent k must be prevented. Similar to the used DAAA for the ECC, we implemented the Square-And-Multiply-Always (SAMA) which was presented 1999 by Kocher *et al.* [6] and uses dummy registers to avoid runtime varieties.

3 Results

For the evaluation of the implementation, we chose the SASEBO GII platform which is equipped with a Xilinx Virtex-5 LX50 FPGA. All results have been obtained post Place&Route using Xilinx ISE 14.3.

For the ECC performance measurements, we used 256 bit random numbers as the scalar and averaged the results of the multiplication with the base point of the

Table 1. Comparison between different approaches

Design	**This work**	**[11]**	**[12]**	**[10]**	
Device	**Xilinx Virtex-5**	**Xilinx Virtex-II Pro**		**Xilinx Spartan-3**	
Curve	**P256**	**P256**	**any**	**P256**	**not supported**
RSA size	**2048 bit**	**not supported**		**not supported**	**2048 bit**
Max. Clk. (MHz)	210	210	68.17	40	95
Logic	1914	1158	2085	27597	
RAM Blocks	6	3	9	0	
HW Mults	4	4	7	0	
ECPM [cycles]	830000	949951	1074625	708000	**not supported**
ECPM [ms]	3.95	4.52	15.75	17.7	**not supported**
MEXP [cycles]	372000	**not supported**		**not supported**	74100
MEXP [ms]	1.77	**not supported**		**not supported**	0.78

NIST-P256 curve. On embedded devices such as Engine Control Units (ECUs) usually only the signature verification is required. Therefore, for efficiency reasons, in nearly all of the currently deployed real-world RSA implementations the public exponent $2^{16} + 1$ is used. As a result, this exponent was also used for the performance measurement. Usually, the signature verification with the RSA crypto system is done with a public key, which means that countermeasures against side-channel attacks are not necessary. Nevertheless, the execution of the SAMA algorithm is tested in order to show the time difference and the feasibility of countermeasure implementations for RSA using our core.

Table 1 shows the efficiency of the core compared to other recent work. The term *ECPM* denotes Elliptic Curve Point Multiplications while MEXP means Modular Exponentiations. While we have chosen a Virtex-5 FPGA for our real performance measurements, numbers obtained for the older Virtex-2 Architecture are only slightly worse then in the Virtex-5 case. Also note that our ALU is slightly more complex then in the given comparisons since we are also able to perform RSA operations.

Table 2 depicts the performance overhead required if different side-channel countermeasures are chosen. Focusing on the ECC case which requires most security for the private signing operation, overhead numbers for each of Coron's countermeasures and their possible combinations are given. The overheads are surprisingly low showing the efficiency of the countermeasures. The minimum protection against timing attacks (as well as simple power analysis attacks) can

Table 2. Comparison of unprotected and protected variants of an *ECPM* and a MEXP in terms of Runtime, Cycles and Operations per second of our approach on a FPGA with 210 MHz

ECC Countermeasures	**Runtime**	**Cycles in 1000**	**Operations per second**	**Runtime in percent**
Without countermeasures	3.95 ms	830	252	100 %
DAAA	5.22 ms	1097	191	132 %
Random projective coordinates	4.54 ms	953	220	115 %
DAAA and random projective coordinates	5.79 ms	1217	172	147 %
Point blinding	4.85 ms	1018	206	123 %
Point blinding and DAAA	6.09 ms	1279	164	154 %
Point blinding and random projective coordinates	5.46 ms	1147	28	138 %
Point blinding, random projective coordinates, DAAA	6.67 ms	1401	149	169 %
RSA Countermeasures	**Runtime**	**Cycles in 1000**	**Operations per second**	**Runtime in percent**
Without countermeasures	1.77 ms	372	564	100 %
Square-And-Multiply-Always	3.23 ms	679	309	182 %

be gained by using the DAAA countermeasure at the cost of 32% more computation time. Unifying the computation time is especially important since timing attacks can even be performed by remote attackers over, e.g., ethernet/WiFi in the vehicle-2-vehicle case. If a local adversary is assumed, additional protection against differential power analysis can be gained by applying point blinding and or randomized projective coordinates. Using all countermeasures at the same time to maximize the desired security only leads to a 69% increase in computation time which is very reasonable considering the risks caused by unprotected implementations.

4 Conclusion

In this work we have presented the design of an low-area highly reconfigurable co-processor for asymmetric cryptography. The design is focused on side-channel resistant executions of elliptic curve operations but can also be used to additionally compute RSA exponentiations. Using less than 2000 slices on a Virtex-5 FPGA and only four hardware multipliers, the proposed core is able to compute approximately 250 scalar point multiplications or double point multiplications. Because of the low area utilization, the core is inherently highly scalable since additional cores can just be instantiated in parallel to achieve the desired throughput.

We have also implemented a set of countermeasures achieving resistance against various side-channel attacks. The constant execution time and fixed program flow of the Double-and-Add-Always countermeasure thwarts not only timing attacks but also SPA or SEMA attacks. Point blinding and the use of randomized projective coordinates are state-of-the-art countermeasures to protect against DPA and DEMA attacks. We also analyzed the performance overhead of these countermeasures and found that even when using all countermeasures the core still delivers a respectable throughput of 150 scalar point or double-point multiplications per second. This overhead is quite low when compared to symmetric cryptography where masking schemes to protect against differential power analysis attacks usually lead to a performance drop of factor 3x-10x.

Using both standardized NIST curves over elliptic curve prime fields and protecting the implementation against side-channel attacks makes this work highly relevant from an industry point of view. The reason behind this is that the upcoming Federal Information Processing Standard (FIPS) 140-3 (to accredited cryptographic modules) will require mandatory side-channel testing for certain security levels. The possibility to update the core by new microcode even when deployed as ASIC allows to react to new attacks by e.g., updating countermeasures as well.

Acknowledgment. This project has been partially funded by the European Union, Investing in your future, European Regional Development Fund.

References

1. Batina, L., Bruin-Muurling, G., Örs, S.B.: Flexible Hardware Design for RSA and Elliptic Curve Cryptosystems. In: Okamoto, T. (ed.) CT-RSA 2004. LNCS, vol. 2964, pp. 250–263. Springer, Heidelberg (2004)
2. Coron, J.-S.: Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems. In: Koç, Ç.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 292–302. Springer, Heidelberg (1999)
3. Fan, J., Verbauwhede, I.: An Updated Survey on Secure ECC Implementations: Attacks, Countermeasures and Cost. In: Naccache, D. (ed.) Cryphtography and Security: From Theory to Applications. LNCS, vol. 6805, pp. 265–282. Springer, Heidelberg (2012)
4. Güneysu, T., Paar, C.: Ultra High Performance ECC over NIST Primes on Commercial FPGAs. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 62–78. Springer, Heidelberg (2008)
5. Koblitz, N.: Elliptic curve cryptosystems. Mathematics of Computation 48, 203–209 (1987)
6. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
7. Miller, V.S.: Use of Elliptic Curves in Cryptography. In: Williams, H.C. (ed.) CRYPTO 1985. LNCS, vol. 218, pp. 417–426. Springer, Heidelberg (1986)
8. Montgomery, P.L.: Modular Multiplication without Trial Division. Mathematics of Computation 44(170), 519–521 (1985)
9. Rivest, R.L., Shamir, A., Adleman, L.M.: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Commun. ACM 21(2), 120–126 (1978)
10. Sakiyama, K., Mentens, N., Batina, L., Preneel, B., Verbauwhede, I.: Reconfigurable modular arithmetic logic unit supporting high-performance RSA and ECC over GF(p). International Journal of Electronics, 501–514 (2007)
11. Varchola, M., Güneysu, T., Mischke, O.: MicroECC: A Lightweight Reconfigurable Elliptic Curve Crypto-processor. In: Athanas, P.M., Becker, J., Cumplido, R. (eds.) ReConFig, pp. 204–210. IEEE Computer Society (2011)
12. Vliegen, J., Mentens, N., Genoe, J., Braeken, A., Kubera, S., Touhafi, A., Verbauwhede, I.: A compact fpga-based architecture for elliptic curve cryptography over prime fields. In: 2010 21st IEEE International Conference on Application-specific Systems Architectures and Processors (ASAP), pp. 313–316 (2010)