

Hardware Architectures for HECC

Gabriel Gallin, Arnaud Tisserand

▶ **To cite this version:**

Gabriel Gallin, Arnaud Tisserand. Hardware Architectures for HECC. CryptArchi 2017: 15th International Workshops on Cryptographic architectures embedded in logic devices , Jun 2017, Smolenice, Slovakia. hal-01545625

HAL Id: hal-01545625
https://hal.science/hal-01545625

Submitted on 22 Jun 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Hardware Architectures for HECC

Gabriel GALLIN
CNRS – IRISA UMR 6074
gabriel.gallin@irisa.fr

Arnaud TISSERAND
CNRS – Lab-STICC UMR 6285
arnaud.tisserand@univ-ubs.fr

1. Context

Nowadays, there is an increasing number of applications and systems requiring strong security on small hardware devices. Public-key cryptography (PKC) is mandatory for providing key exchange and digital signature. The first standard for public-key cryptosystems was RSA. However, to be compliant with the current recommended theoretical security levels, RSA based cryptosystems must use large keys – at least two thousand bits – which make them too costly for embedded applications.
Curves based cryptography such as *Elliptic Curve Cryptography* (ECC) or *Hyper-Elliptic Curve Cryptography* (HECC) is known to provide a given security level at a lower cost than RSA. For instance, 226-bit ECC keys offer the same security level as 2048-bit RSA. Due to its reduced cost and better performance, ECC is now recommended as the PKC standard.

2. Hyper Elliptic Curve Cryptography

Recent research has pointed out HECC as an attractive alternative to ECC. HECC is based on a different kind of curves, which allows the size of the field elements to be halved, but at the expense of an increased number of finite field operations. In [9], Renes *et al.* very recently presented software implementations of key exchange and signature schemes based on HECC and Kummer surfaces, targeting embedded processors (*ARM Cortex M0* and *AVR ATmega*). The provided results show very interesting speedups compared to state-of-the-art ECC: 30% speedup for Diffie-Hellman key exchange and up to 70% for signature.
Operations on HECC involve more operations on the underlying finite field than ECC. However, one can observe that in ECC, most of the computations are dependent and must be mostly done in a sequential way. For this reason, the internal parallelism in ECC is quite limited compared to HECC. For instance, in the formulas presented in [9], one can find regular patterns of four to eight independent modular multiplications – the most costly and common finite field operation – feasible in parallel during the whole scalar multiplication. HECC internal parallelism brings forward numerous questions for hardware implementation. Those questions can be summarized as follows: *how can one take advantage of the parallelism in HECC to design efficient hardware cryptosystems?*

3. Arithmetic Units

In order to build an efficient accelerator, the first step is to build efficient arithmetic units. These units are dedicated to the computations over finite field elements. The most common and costly finite field operation in (H)ECC is the *modular multiplication*. For instance, depending on the multiplier area, one multiplication requires from 30 to 100 clock cycles depending on the field elements width. In [7], Peter L. Montgomery presented an algorithm for modular multiplication, which is still now the base of state-of-the-art multiplication in prime finite fields. It is known as *Montgomery modular multiplication* (MMM). Many algorithms have been derived from this paper. They mainly aim at improving efficiency by interleaving the multiplication and modular reduction steps in order to reduce the size of the intermediate data and to gain some speedup. One of the most famous variant is the *Coarsely Integrated Operand Scanning* (CIOS) method presented by Koç *et al.* in [5]. However, besides these improvements, Montgomery multiplication still suffers from strong dependencies inside the main loop of partial products accumulation and reduction. This makes hardware implementations difficult to optimize in the case of FPGAs using DSP blocks. In order to reach high frequencies, DSP blocks must indeed use three to four internal pipeline stages. Due to data dependencies, one cannot feed efficiently this pipeline, resulting in a loss of efficiency in the circuit utilization.
In [6], Ma *et al.* proposed a FPGA implementation of MMM based on an improvement of the algorithm presented by H. Orup in [8]. This implementation is known to be one of the fastest implementations on FPGA in the literature. However,

to get rid of some of the internal dependencies, the method implies huge overheads in terms of the size of the computed data, increasing the circuit area of the design.

4. Proposed HECC Architectures

Our research group has been studying arithmetic operators and implementations of hardware accelerators for ECC, with robustness against physical attacks such as Side Channel Analysis (SCA) and faults injections. We are now designing hardware accelerators for HECC scalar multiplication by exploring different types of architectures.
For this, we first improved the hardware utilization of the multiplier unit. We decided to use the classical CIOS method as a basis to design an *hyper-threaded* modular multiplier. The idea behind this hyper-threaded multiplier is to fill the unused stages of the DSP blocks with other independent modular multiplications. In our multiplier, we enter 3 independent sets of operands $\{(A_1, B_1), (A_2, B_2), (A_3, B_3)\}$ before the first product $P_1 = A_1 \times B_1$ is computed. This way, all the stages in the DSP blocks are full after the very first latency. This is more efficient for HECC since the internal parallelism allows more than 3 independent multiplications at each step during the scalar multiplication.
We then developed a specific CABA (Cycle Accurate, Bit Accurate) simulator for our architectures. With this simulator, we can study the impact of the type, number and size of the arithmetic units and of the choice between different types of parallel architectures on the performances, circuit area and resistance against physical attacks.
We will present implementation results on different FPGAs for various configurations of our modular multiplier, both in terms of computation time and circuit area. As an example, for 128-bit field elements, we reach the same computation time with half the number of DSP blocks compared to the best state-of-the-art [6]. We will show that it provides a better computation time / circuit area cost trade-off when several modular multiplications can be computed in parallel, which is always the case in HECC.
In a second time, and using our simulator, we will explore various architectures for our accelerator, starting from a classical Harvard architecture and changing architectural parameters, such as the numbers and types of arithmetic units. We will also compare different ways to manage internal data transfers and different control flow implementations. The most interesting configurations will be implemented on FPGA and evaluated on our attack setup.

Acknowledgments

This work is partly funded by the HAH project (Labex CominLab and Lebesgue, Brittany Region, `http://h-a-h.inria.fr/`).

References

[1] J. W. Bos, C. Costello, H. Hisil, and K. Lauter. Fast cryptography in genus 2. *Journal of Cryptology*, 29(1):28–60, Jan. 2016.
[2] H. Cohen, G. Frey, R. Avanzi, C. Doche, T. Lange, K. Nguyen, and F. Vercauteren. *Handbook of Elliptic and Hyperelliptic Curve Cryptography*. Discrete Mathematics and Its Applications. Chapman & Hall/CRC, July 2005.
[3] P. Gaudry. Fast genus 2 arithmetic based on theta functions. *Journal of Mathematical Cryptology*, 1(3):243–265, 2007.
[4] D. Hankerson, A. Menezes, and S. Vanstone. *Guide to Elliptic Curve Cryptography*. Springer, 2004.
[5] Ç. K. Koç, T. Acar, and B. S. Kaliski, Jr. Analyzing and comparing Montgomery multiplication algorithms. *Micro, IEEE*, 16(3):26–33, June 1996.
[6] Y. Ma, Z. Liu, W. Pan, and J. Jing. A high-speed elliptic curve cryptographic processor for generic curves over GF(p). In *Proc. 20th International Workshop on Selected Areas in Cryptography (SAC)*, volume 8282 of *LNCS*, pages 421–437. Springer, Aug. 2013.
[7] P. L. Montgomery. Modular multiplication without trial division. *Mathematics of Computation*, 44(170):519–521, Apr. 1985.
[8] H. Orup. Simplifying quotient determination in high-radix modular multiplication. In *Proc. 12th Symposium on Computer Arithmetic (ARITH)*, pages 193–199. IEEE Computer Society, July 1995.
[9] J. Renes, P. Schwabe, B. Smith, and L. Batina. μKummer: Efficient hyperelliptic signatures and key exchange on microcontrollers. In *Proc. Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, volume 9813 of *LNCS*, pages 301–320. Springer, Aug. 2016.