A Survey of Hardware Implementations of Elliptic Curve Cryptographic Systems

Basel Halak
School of Electronics and Computer Science
University of Southampton
Email: bh9@ecs.soton.ac.uk

Said Subhan Waizi
School of Electronics and Computer Science
University of Southampton
Email: ssw1e15@soton.ac.uk

Asad Islam
School of Electronics and Computer Science
University of Southampton
Email: ai1u14@soton.ac.uk

***Abstract*—Elliptic Curve Cryptography (ECC) has gained much recognition over the last decades and has established itself among the well known public-key cryptography schemes, not least due its smaller key size and relatively lower computational effort compared to RSA. The wide employment of Elliptic Curve Cryptography in many different application areas has been leading to a variety of implementation types and domains ranging from pure software approaches over hardware implemenations to hardware/software co-designs. The following review provides an overview of state of the art hardware implemenations of ECC, specifically in regard to their targeted design goals. In this context the suitability of the hardware/software approach in regard to the security challenges opposed by the low-end embedded devices of the Internet of Things is briefly examined. The paper also outlines ECC's vulnerability against quantum attacks and references one possible solution to that problem.**

***Index Terms*—Elliptic Curve Cryptography, public key, cryptoprocessor, cryptographic coprocessor, instruction set extension, hardware/software codesign**

I. Introduction

Elliptic curve cryptography (ECC) has gained much recognition over the last decades and has established itself among the well known public-key cryptography schemes. The fact that ECC can have smaller key sizes than RSA, while maintaining comparable levels of security, makes it well suited for use in a variety of application areas [1]. Being deployed to effectively provide a means of public-key encryption, key establishment or signature schemes it can drastically enhance the overall security level of an application. The benefits of ECC gave rise to a variety of applications areas ranging from securing internet protocols such as HTTPS or SSH over internet currencies such as *Bitcoin*[1] to embedded systems in form of smart cards, RFID devices and wireless sensor networks [2]. The numerous application areas and environments lead to different forms of implementation, each of them tailored towards the specific use-case and optimized in some manner, be it cost, area, performance or energy efficiency.

The performance of Elliptic Curve Cryptography systems is determined by the efficient implementation of the arithmetic in the underlying finite field [3]. ECC systems that are completely realized in software offer lowest cost and a high degree of flexibility [4]. They can effectively adapt to changes in current standards, making them well suited for fast prototyping. Detailed information about efficient software implementations of ECC can be found in [5]. However these are usually of several orders slower than hardware implementations which makes their use impractical for applications in time-constrained computing environments that require fast responsiveness and processing. Under these restrictions hardware implementations turn out to be the more suited alternative. There are mainly two approaches to adapt ECC into hardware. The first approach uses dedicated hardware in form of a coprocessor or hardware acceleration block. This application specific and modular hardware design can achieve extraordinary performance and offers high reliability but comes with higher cost due to the need of additional hardware. The second approach enhances an existing processor architecture by extending its instruction set to speed up certain operations of finite field arithmetic. This is often referred to as a hardware/software co-design as the ECC scheme is implemented by combining both domains [6].

Existing designs mostly vary in the underlying finite field ($GF(2^m)$ and $GF(p)$), implementation of the arithmetic algorithms in these fields and key sizes. The following review provides an overview of current hardware implementations of ECC. Both dedicated hardware in form of a coprocessor and the hardware/software co-approach realized through an enhanced architecture is examined. Section II presents a brief introduction to the theoretical background and motivation for Elliptic Curve Cryptography. Section III gives an overview of state of the art implementations of cryptographic coprocessors followed by reviewing current architectures for hardware/software co-design in section IV. Having talked about the vulnerability of ECC systems against quantum attacks in section V the conclusion in section VI provides a summary of ECC and gives glimpse of future application areas.

II. Theoretical Background and Motivation

The Elliptic Curve Cryptography scheme is built on the mathematical properties of elliptic curves and was proposed by Victor Miller [7] and Neal Koblitz [8] in 1985. An elliptic curve over any field K can be defined as the set of all solutions $(x, y) \in K$ x K that fullfill the following general Weierstraß

[1]digital currency used for direct peer-to-peer tranactions

equation, where a_i lie in K.

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \quad (1)$$

Note that in cryptographic applications mostly non-supersingular elliptic curves are considered ($a_1 = 0, a_4 = 0$) as they provide the highest security. When K is a finite field, often denoted as a Galois field $GF(q)$, the order q is equal to the number of elements in the finite field. A Galois field of order q only exits if q is a prime power ($q = p^m$), where p and m denote a prime number and a positive integer respectively. $\mathcal{O}$ denotes the point at infinity and is also considered as a point on the curve. Finite fields used for cryptographic applications mainly consist of prime fields $GF(p)$, binary fields $GF(2^m)$ and extension fields. However the use of extension fields is not as common as the first two. Each field is equipped with a set of arithmetic operations, mainly defined by multiplications and additions. Elliptic curve additions and multiplications make use of the underlying finite field arithmetic. As elliptic curve point multiplication (ECPM) is the computationally most demanding operation of ECC it is often used as a means of comparing designs regarding performance.

The mathematical problem that equips ECC with its high security level is the elliptic curve discrete logarithm problem (ECDLP). Consider an elliptic curve E and a point $P \in E$ of order m (point is of order m if $mP = \mathcal{O}$ and m is the smallest integer statisfying this equation). Let $Q \in \langle P \rangle$, so that $Q = aP$ for any integer a, $0 \leq a < m$. The ECDLP is defined as the problem to find a given the points Q and P. Currently the two fastest algorithms that can solve the ECDLP are the *Number Field Sieve* and *Pollard's rho algorithm*. Especially for longer key sizes however the problem of solving the ECDLP remains intractable [3].

III. Review of ECC Coprocessors

Elliptic Curve Cryptography processors (ECP), often referred to as cryptoprocessors can provide the immense performance that is often required in server-based domains such as e-commerce or online banking systems. They can maintain the high throughput while offering the robustness and reliability needed in this area but usually involve the highest cost. High-speed ECP implementations generally operate on elliptic curves over binary fields $GF(2^m)$ as the carry-free arithmetic often leads to a more efficient hardware architecture [9], however elliptic curves over prime fields $GF(p)$ are often preferred due to standards in Europe and the US [10].

A. Implementation over binary fields

Although the paper of Orland and Paar [9] was introduced sixteen years ago, it is still regarded as relevant and is thus extensively referenced. Orlando and Paar designed a reconfigurable elliptic curve architecture and verified it over $GF(2^{167})$, but pointed out that their design can be effectively reconfigured to any arbitrary binary field $GF(2^m)$. The proposed design, containing two loosely coupled processors, consists of three main components and can perform ECPM in 210 μs when implemented on a Xilinx FPGA XCV400E. The main controller (MC) initiates the elliptic curve point multiplication aP and acts as an interface to the surrounding circuitry. While the arithmetic unit (AU) takes care of all finite field operations, an arithmetic unit controller (AUC) performs the remaining subroutines. Their design is highly reconfigurable and can according to them be easily adapted for more efficient elliptic curve algorithms, however the use of two processors leads to high area overhead and ultimately to higher costs and only one prime field is supported at a time.

Khan and Benaissa [11] presented an ECC processor making use of a novel algorithm for point additions and doublings. Their work focused on maintaining high throughput at low area and outperformed the previously introduced implementations both in area and speed. This was mainly achieved by combining an optimized and efficient memory unit, a pipelined digit-serial multiplier and careful scheduling of the arithmetic logic blocks in order to avoid idle clock cyles. They adapted the LD modified Montgomery point multiplication algorithm as it offers adequate speed and allows parallelization at a low area overhead. They claim that their design has led to the best throughput / area efficiency figure reported to date. A point multiplication aP over $GF(2^{163})$ takes 10.51 μs while only using 1476 slices on a Virtex4 LX2512 FPGA.

Lijuan and Shuguo [12] recently proposed a pipelined architecture for ECPM over $GF(2^m)$ making use of bit-parallel finite field multipliers based on the Karatsuba-Ofman algorithm. They investigated in both finding the most efficient number of pipeline stages and the best placement of the pipeline registers by evaluating different scheduling algorithms. Their architecture supports all five binary fields recommended by NIST[2] and was implemented on the Virtex4 XC4VLX200 FPGA, using a three-staged pipeline which according to them gave the most efficienct performance. An ECPM over $GF(2^{163})$ could be performed in 6.1 μs on the Virtex4. Compared to the ECC processor presented by [11] this architecture lead to a speed increase of a factor close to two, however by using 7365 slices they also increased resource utilisation by a factor of 5.

Table I provides a summary of state of the art binary field coprocessor implementations of Elliptic Curve Cryptography. It highlights the different key lengths, the platform it was implemented on including the maximum frequency it can run on and furthermore the time required for an elliptic curve point multiplication.

B. Implementation over prime fields

Chi et al. [13] proposed a scalable cryptographic processor that supports all five prime fields that are recommended by NIST. The ECP can perform at high speed by massive paralellizing of prime field operations. Furthermore area is saved by using the same arithmetic buildings blocks for modular addition/subtraction and inversion. Point multiplications were performed using the double-and-add algorithm while using

[2]National Institute of Standards and Technology

TABLE I
COMPARISON OF BINARY FIELD HARDWARE IMPLEMENTATIONS

Ref.	HW/SW	Field	Platform	Frequency	ECPM
5	HW	$GF(2^{167})$	Virtex4 XCV400E	76.7 MHz	$210\mu s$
6	HW	$GF(2^{163})$	Virtex4 LX25	290 MHz	$14.39\mu s$
7	HW	$GF(2^{163})$	Virtex4 LX200	222 MHz	$6.1\mu s$
13	HW	$GF(2^{160})$	$0.13\mu m$ CMOS ASIC	510.2 MHz	$190\mu s$
14	HW	$GF(2^{233})$	Virtex4 XC4VLX80	185 MHz	n.a.

mixed Jacobian and affine coordinates for point additions and Jacobian coordiantes for point doublings. They claim that their processor is the fastest and smallest ECP that supports all prime fields, being able to compute ECPM between 1.709 and 28.04 ms while only 8 DSP slices on a Virtex4 XC4VFX100 FPGA. The implementation however is not generic but specific to only Xilinx FPGAs as the required DSP48 slices are only available on these chips. As all key sizes are supported this scalable implementation offers the option of small key sizes that are currently used and will still be able to adapt to satisfy the demand for higher security levels in regard to future applications.

The ECP proposed by Güneysu and Paar [10] significantly exceeds the speed of the previously mentioned design but uses more than three times the amount of DSP slices on a Virtex4 XC4VFX12 FPGA. It operates over $GF(224)$ and $GF(256)$ and can finish high-performant ECPM in 0.452 and 0.620 ms, depending on the chosen prime field. This is mainly achieved by making extensive use of parallely working DSP slices for the modular multiplication implementing the school-book multiplication as the underlying algorithm. Most modern FPGAs are equipped with hardware specific DSP blocks so their design might be adaptable to a wide range of available FPGAs. The implementation furthermore uses the concept of asynchronous clocks which drives the DSP blocks at higher frequencies than the remaining modules and uses multicore processing to achieve exceptional performance. However the ECP needs to be reconfigured when changing prime fields contrary to the design of Chi et al. [13] that supports all prime fields without the need of hardware reconfiguration.

Martin and Andreas [14] implemented the elliptic curve cryptosystem on an FPGA for prime fields. The main goal of their implementation was to provide a basic understanding for people in the academic sector so that if someone wants to start research in this field, this can be a good starting point. Due to this design goal, preference was given to simplicity rather than performance or low cost. This implementation supports all prime fields recommended by NIST, i.e. 192, 224, 256, 384 and 521 bits. Fairly simple algorithms were used for the modular arithmetic. Modular multiplication was done using the shift-and-add method. For point multiplication, double and add algorithm was used. The implementation made sure that the hardware was as lightweight as possible and no importance was given to the speed of the algorithms. The paper said that the security of the implementation could be improved by adding dummy operations protecting against side channel attacks. Due to this it would be really difficult for the attacker to evaluate timing and power consumption information which could help him in decrypting or getting the secret key.

Table II gives an overview of state of the art prime field co-processor implementations of ECC. It highlights the different key lengths, the platform it was implemented on including the maximum frequency it can run on and furthermore the time required for an elliptic curve point multiplication. Comparing table II to table I we can observe that prime field implementations of ECC are generally slower than binary field systems.

TABLE II
COMPARISON OF PRIME FIELD HARDWARE IMPLEMENTATIONS

Ref.	HW/SW	Field	Platform	Frequency	ECPM
2	HW	$GF(192)$	Virtex4 FX100	182 MHz	$2.36ms$
2	HW	$GF(256)$	Virtex4 FX100	182 MHz	$5.457ms$
3	HW	$GF(256)$	Virtex4 XC4VSX55	375 MHz	$0.041ms$
13	HW	$GF(192)$	$0.13\mu m$ CMOS ASIC	138 MHz	$1.44ms$
14	HW	$GF(256)$	Virtex5	45 MHz	n.a.

C. Implementation over both prime and binary fields

There are some designs that aim to support a wide range of curves both over prime fields and over binary fields. Satoh and Takano [15] for example designed such a versatile coprocessor that supports Galois fields $GF(p)$ and $GF(2^n)$. They used the Montgomery multiplier and a redundant binary converter in their design which enables their building blocks to support arithmetic over both fields. By this setup they reach operation times of 1.21ms and 0.19ms for 160-bit elliptic curve point multiplications over prime fields and binary fields respectively. They boast that all of the popular cryptography functions like DSA, RSA, CRT are supported by this setup. There is a lot of flexibility for speed and area which allows the user to reach the best suited trade-off in their own design. The main benefit of this implementation is that it can support different fields, curves and sizes without changing the hardware, provided that the memory capacity of the target platform is sufficient. This can be used as a starting point, if you want to use a general purpose ECC coprocessor and adjust it to your requirements.

Nagaraja and Sridhar [16] also designed an unified architecture supporting both fields. The proposed design is supposed to be highly scalable and flexible. This implementation also used Montgomery multiplier and redundant binary converter. In order to be less susceptible to side channel attacks randomized multiplication operations are inserted to cover up both timing and power consumption which could help an attacker deduce the secret key. The dual field ECC processor design can reach operation frequencies up to 124.347 MHz while consuming 1.091W and occupying 3066 slices on a Xilinx 13.4 Virtex 5 FPGA.

IV. HARDWARE/SOFTARE CO-APPRAOCHES OF ECC

The hardware/software co-approach (HSC) has proven to be an adequate alternative when a balance between cost,

performance and flexibillity is aimed for [6]. Enhancing the existing instruction set of an architecture in order to accelerate application-specific operations can lead to a significant gain in performance compared to a software only implementation.

Großschädl and Kamendje [17] implemented a functional model of a 16-bit RISC-like processor equipped with a single-stage pipleine. It is able to perform usual arithmetic/logical functions, however the use of a unified multiplier and a dual field adder additionally enhances dual field multiplications and additions using the same datapath with minor modifications leading to minimal area overhead. The instruction set is extended by a special $MULGF2$ instruction that initiates and accelerates polynomial multiplication over binary fields using the pencil-and-paper method. The oustanding feature of their design that plays a major role regarding performance is the employment of word-level operations instead of bit-level operations which are slow on microprocessors. The setup was evaluated by performing the elliptic curve digital signature algorithm (ECDSA) over $GF(2^{191})$ that includes one ECPM in approximately 650 ms. They pointed out that the performance of their design is comparable to the Infineon SLE 66CL160S, a 16-bit smart card processor with an ECC coprocessor which requires 436 ms for the same operation.

Großschädl and Savaş [18] evaluated and identified several algorithms in respect to performance and suitability for instruction set extension. In this manner they proposed five custom instructions that can be easily implemented on a RISC processor and help accelerating ECC operations both over prime fields $GF(p)$ and binary fields $GF(2^m)$. Comba's method was used for prime field multiplication and squaring and the pencil-and-paper method for multiplication over binary fields. They developped a functional SystemC model of a single-stage pipeline MIPS32 core enhanced with their suggested instructions which can perform ECPM over $GF(2^{191})$ and $GF(192)$ in 21 and 36 msec (@33MHz) respectively. Compared to the previous work more versatility and a higher performance is achieved at the cost of higher area-overhead. Both [17] and [18] verified their design in terms of a functional model but did not provide a full implementation on a commercial FPGA or ASIC.

The findings of [18] highly motivated the SmartMIPS™, an architecture that was originally developed for security enhancements in smartcards. Its extended instruction set is built on the RISC MIPS32 architecture and facilitates arithmetic of both private and public key cryptography schemes and in regard to ECC allows acceleration in binary and prime fields.

Bartolini et al. [19] analyzed the employability of ECC on another embedded processor architecture, namely the ARM. They investigated the relationship between performance of different finite field and ECP multiplication algorithms and the micro-/architectural features of the ARM-based Intel XScale processor. In this context they proposed an enhanced architecture that contains a word-level finite field polynomial multiplier in its datapath accessed by a new $MULGF$ instruction. It was shown that the new design achieved a performance gain of around 41% compared to a software-only implementation on the same platform.

Instead of designing a coprocessor Hani et al. [20] introduced custom instructions to the instruction set architecture of the 32-bit Altera Nios II architecture. According to the paper co-processors are slower than this approach because they require separate registers and data paths which can ultimately slow down the performance. Interprocessor communication between processor and coprocessor might also be another factor that can have impact on the performance. The elliptic curve arithmetic buildings blocks are tightly coupled to the main Nios II processor in this setup. Custom logic was created for field multiplication and field squaring. Least significant digit first multiplication algorithm was used for field multiplication while a simple custom made algorithm was used for field squaring was used for field inversions. The implementation supports ECC acceleration over the binary fields $GF(2^{163})$ and $GF(2^{193})$. This implementation provides a good example of how the HSC can be combined with an cryptographic coprocessor design.

Table III gives an overview of hardware/software co-designs of ECC systems. It highlights the different finite fields, key lengths, the reference platform including the maximum frequency it can run on and furthermore the time required for an elliptic curve point multiplication. It shows that the hardware/software co-design is slower than both most prime field and binary field coprocessors.

TABLE III
COMPARISON OF HSC IMPLEMENTATIONS

Ref.	HW/SW	Field	Platform	Frequency	ECPM
8	HSC	$GF(2^{191})$	16-bit RISC	5 MHz	$650ms$
9	HSC	$GF(2^{191})$	MISP32	33 MHz	$21ms$
9	HSC	$GF(192)$	MIPS32	33 MHz	$36ms$

V. SHOR'S ALGORITHM

In 2008 Proos and Zalka [21] published a paper which stated that ECC was more vulnerable to quantum computing attacks than a comparable RSA implementation by using the Shor's algorithm. According to Proos and Zalka it would take 4096 qubits (quantum bits) to break a 2048-bit RSA system while only 1300 qubits would be needed to break a 224-bit ECC system. Their findings could determine if ECC systems will be still regarded as secure in future implementations.

In NSA's B Suite [22] it was stated that they wanted to move from ECC to encryption methods that would be invulnerable or at least less susceptible to quantum attacks. According to them the time of ECC has ended and it would be best to develop a new public-key encryption algorithm for this purpose.

In 2011 however, four years before the announcement of the NSA, Jao and Feo [23] published a paper proposing a different approach for Elliptic Curve Cryptography systems which would be resistant quantum attacks based on Shor's algorithm.

The sudden decision of the NSA to leave behind the well established Elliptic Curve Cryptography scheme has been a cause for a lot of criticism. People speculate that this sudden decision may have been a result of the Edward Snowden leaks [24]. According to the leaked information, one of the algorithms created by the NSA and adapted as a NIST standard had been equipped with a backdoor function known to the NSA, enabling them to access secret information. Based on this observation, NSA's sudden move away from ECC and towards a new public-key encryption scheme appears to not be solely based on its vulnerability to quantum attacks.

VI. Conclusion

Pure software implementations of ECC, despite offering best flexibility at lowest cost, cannot cope with the speed demands of many application areas as general purpose processors are not designed for efficient handling of ECC's underlying finite field arithmetic. When performance and robustness are of highest importance dedicated hardware in form of cryptographic coprocessors are used and can effectively take the workload off the main processing unit. Server systems, like they are used in e-commerce and online banking, where inefficiencies in thorughput and reliability can have negative financial effects, drastically benefit from this setup. The additional cost involved for a coprocessor stands in no relation to the usefullnes of employment in these areas. This however is not true for cost-sensitive embedded systems in which an additional coprocessor often cannot be justified in terms of cost-performance ratio. In this case the employment of a hardware/software co-design could be a suitable alternative as a god balance of performance, flexibility, area and cost is achieved. Enhanced architectures employing instructions specifically designed to accelerate computational expensive operations of ECC have established themselves. The trend has shown that in some embedded applications, such as smart cards for example, where cost-performance ratio plays a significant role due to high manufacturing volumes, the HSC has been effectively replacing the standard approach consisting of processor and supportive coprocessor.

The Internet of Thigns (IoT) expresses the idea of connectivity that includes billions of embedded devices, sensors and computers exchanging and collecting data. Experts claim that there will be over 50 billion connected devices by the year 2020. Lack of appropriate security mechanisms could lead to privacy breaches or in the worst case harm life and is one of the major concerns of IoT but at the same time employing those mechanisms in devices that are utterly constrained in resources becomes a considerable challenge. The hardware/software co-design for ECC schemes presents an attractive alternative, especially in consideration of ECC's lower key sizes and relatively lower computational effort compared to RSA. In combination with a lightweight RISC architecture, such as MIPS or picoMIPS this approach can provide a means of security with aedequate performance while overcoming the challenges of low cost, low area and low power imposed by the constraints of low-end IoT devices. The latter mentioned architecture has not been used for instruction set extension in regard to ECC yet and could be an interesting topic for future research work.

References

References

[1] N. Koblitz, A. Menezes and S.Vanstone, *The State of Elliptic Curve Cryptography*, in *Des. Codes Cryptography*, vol. 19, pp. 173-193, 2000.

[2] C. Lee and H. Chien, *An Elliptic Curve Cryptography-Based RFID Authentication Securing E-Helath System*, in *International Journal of Distributed Sensor Networks*, Vol. 2015, 2015.

[3] D.R. , A.J. Menezes and S. Vanstone, *Guide to Elliptic Curve Cryptography*, New York: Springer-Verlag, 2004.

[4] M. Bluhm and S. Gueron, *Fast software implementation of binary elliptic curve cryptography*, in *Journal of Cryptographic Engineering*, 2015.

[5] D. Hankerson, J.C. Lopez Hernandez and A.J. Menezes, *Software implementation of Elliptic Curve Cryptography over binary fields*. In *Cryptographic Hardware and Embedded Systems - CHES 2000*, 2000.

[6] M. Koschush, *Hardware/Software Co-design of Elliptic Curve Cryptography on an 8051 Microcontroller*, in *Cryptographic Hardware and Embedded Systems - CHES 2006*, 2006.

[7] V. Miller, *Use of elliptic curves in cryptography*, in *Proc. Adv. Crypto. (CRYPTO)*, pp. 417-426, 1986.

[8] N. Koblitz, *Elliptic curve cryptosystems - Mathematics of Computation*, in *Math. Comp. Vol 48*, pp. 203-209, 1987.

[9] G. Orlando and C. Paar, *A High-Performance Reconfigurable Elliptic Curve Processor for GF(*2^m*)* in *Cryptographic Hardware and Embedded Systems*, CHES 2000, pp. 41-56, 2000.

[10] T. Güneysu and C. Paar, *Ultra high performance ECC over NIST primes on commercial FPGAs*, in *Cryptographic Hardware and Embedded Systems*, CHES 2008, pp. 62-78, Germany: Springer-Verlag, 2008.

[11] Z. Khan, M. Benaissa, *Throughput/Area-efficient ECC Processor Using Montgomery Point Multiplication on FPGA*, IEEE, 2015.

[12] L. Lijuan and L. Shuguo, *High-Performance Pipelined Architecture of Elliptic Curve Scalar Multiplication over* $GF(2^m)$ in *IEE Transactions on VLSI Systems*, pp. 1223-1232, 2016.

[13] K.C.C. Loi and S. Ko, *Scalable Elliptic Curve Cryptosystem FPGA Processor for NIST Prime Curves* in *IEE Transactions on VLSI Systems, Vol 23, No. 11*, 2015.

[14] A. Schramm, A.Grzemba, *On the Implementation of a Lightweight Generic FPGA ECC Crypto-Core over GF(p)"*, IEEE, Pilsen, 2013.

[15] A. Satoh, and K. Takano, *A scalable dual-field elliptic curve cryptographic processor*, in *IEEE Transactions on Computers*, vol 48, no. 3. pp.171-177, 2003.

[16] S. Nagaraja, and V. Sridhar, *A Unified Architecture for a Dual Field ECC Processor Applicable to AES*, in *Fifth International Conference on Computational Intelligence MOdelling and Simulation*, 2013.

[17] J. Großschaedl and G. Kamendje, *Instruction Set Extension for Fast Elliptic Curve Cryptography over Binary Finite Fields GF(*2^m*)*, IEEE, 2003.

[18] J. Großschaedl and E. Savas, *Instruction Set Extension for Fast Arithmetic in Finite Fields* $GF(p)$ *and GF(*2^m*)*, CHES, pp. 133-147, 2004.

[19] S. Bartolini, I. Branovic, R. Giorgi and E. Martinelli, *Effects of Instruction-Set Extensions on an Embedded Processor: A Case Study on Elliptic-Curve Cryptography over* $GF(2^m)$, IEEE, 2008.

[20] M. Khalil-Hani, A. Irwansyah and Y.W.Hau, *A tightly coupled finite field arithmetic hardware in an FPGA-based embedded processor core for Elliptic Curve Cryptography*, in *International Conference on Electronic Design*, 2008.

[21] J. Proos, and C. Zalka, *Shor's discrete logarithm quantum algorithm for elliptic curves*, in *Quantum information & Computation*, vol.3, no.4, pp.317-344, 2003.

[22] NSA, *National Security Agency*, 19 August 2015, [*Online*], Available: https://www.nsa.gov/ia/programs/suiteb_cryptography/, [Accessed 15 April 2016].

[23] D. Jao, and L. Feo, *Towards Quantum-Resistant Cryptosystems from Supersingular Elliptic Curve Isogenies*, in *Post-Quantum Cryptography Lecture Notes in Computer Sciences*, pp. 19-34, 2011.

[24] N. Koblitz, Alfred and J. Menezes, *A riddle wrapped in an enigma*, Available: https://eprint.iacr.org/2015/1018.pdf, 2015.