Verification of Elliptic Curve Digital Signature

Dr. I Manju jacklin[1] , Dr. R Dhanagopal[2], R Pandimeena[3], G Archana[4], S sharmila[5]

Professor[1, 2], Asst. Professor[3], UG Scholar[4, 5]

Electronics and Communication Engineering[1, 2, 3, 4, 5]

Chennai Institute of Technology[1, 2, 3, 4, 5], Chennai, India.

Abstract – **To Verify an ECDSA signature for a double scalar multiplication on an elliptic curve. In this the computation to perform on a twisted Edwards curve with an endomorphism, which reduces the number of point doublings by 50% compared to a conventional implementation. We focus on a curve over the 207-bit prime field F*p* . To develop an optimizations to the operation by describe two hardware architectures for performing the operation. The first architecture is a small processor implemented in 0.13 *μ*m CMOS ASIC to resource the Internet of Things. The second architecture is designed for fast signature verifications by using FPGA acceleration by using the server-side. It offer various trade-offs between performance and resource requirements and they are valuable for IOT applications.**

1 Introduction - The Internet of Things generally refers to scenarios where network connectivity and computing capability extends to objects, sensors and everyday items not normally considered computers, allowing these devices to generate, exchange data with human intervention. There is, however, no single, universal definition. It is a paradigm in which objects, such as Radio Frequency Identification (RFID) tags, sensors, mobile phones etc. A digital signature is a scheme to verify the digital messages or documents. A digital signature gives a recipient to believe that the message was created by a sender, so that the sender can reject the sent message, and that message was not altered. Digital signatures are an indispensable component of modern security protocols such as Transport Layer Security point arithmetic than a basic Weierstrass curve.

In this we introduce, the **twisted Edwards curves** are plane models of elliptic curves, a generalisation of Edwards curves an efficiently computable endomorphism φ and demonstrate how such endomorphism can be used to speed up the ECDSA verification process, and the properties is defined as,

$$ET / Fp : -x^2 + y^2 = 1 + x^2y^2$$

The Elliptic Curve Digital Signature Algorithm (ECDSA) is the elliptic curve analogue of the Digital Signature Algorithm

2 Twisted Edward Curve

Twisted Edwards curves is a generalization of the recently introduced Edwards curves. It shows that the curves include more curves over finite fields, in all elliptic curve to form Montgomery .It shows how to cover more curves to explicit fast formulas for twisted Edwards curves in projective and inverted coordinates and shows that twisted Edwards curves save time for many curves that were already defined by Edwards curves.

$$Ea,d : ax2 + y2 = 1 + dx2y2$$

There is a remarkable addition law on Twisted Edwards curves which can be complete when is a square and a non-square in Fp . Here completeness means the addition produces a correct result for any two points on Ea,d without exception.

In remaining, we adopt the twisted Edwards model for our desired curve, which provides very efficient elliptic curve group arithmetic and high performance . On the one side, the group formulae of twisted Edwards curve are usually more efficient compared with other curve models, i.e., requiring less finite field arithmetic operations [13], [14]. On the other side, the complete group law of twisted Edwards curves admits a more secure execution pattern and thus the implementation of scalar multiplication on such curve would resist against certain side-channel attack

2.1 Group Law

For all elliptic curves and also for the twisted Edwards curve, it is possible to perform some operations between in this points, such as adding two of them or doubling one. The results of these operations to points the curve itself. In that some formulas are given to obtain the coordinates of a point resulted from an addition between two other points (addition), or the coordinates of point resulted from a doubling of a single point on a curve

2.2 GLV Method

The Gallant-Lambert-Vanstone (**GLV**) **method** is a very efficient **technique** for accelerating point multiplication on elliptic curves with effciently computable endomorphisms. The GLV method replaces the computation by a multiscalar multiplication , where the sub-scalars k_1 and k_2 have lengths of approximately half of the original scalar k. These two scalar multiplications can be computed simultaneously by using the so-called Shamir's trick

We find that the new method runs in between 0.70 and 0.84 the time of the previous best methods. The exact performance on that platform is used for our best result in 8-bit processors. Our methods (unlike methods using Montgomery ladders, such as can also be used for signature verification.

The Gallant-Lambert-Vanstone's computation method [9] will be briefly summarized in this part. Assume that Fq is a finite field. The point P = (x, y) is a point on an elliptic curve E defined over a field Fq, with order n such that the cofactor h = #E(Fq)/n is small, say h ≤ 4. The characteristic polynomial of a nontrivial endomorphism ψ defined over Fq takes the form X2 + rX + s, where r and s are actually small fixed integers. By the Hasse bound, since n is large, then ψ(P) = λP for some λ ∈ [1, n − 1]. As a matter of fact, there is only one copy of Z/n inside E(Fp) and ψ(P) has also an order dividing n. Moreover, the parameter λ is a root of X2 + rX + s modulo n, where the case λ = 0 is excluded from all cases. The definition of the group homomorphism T as follows,

$$T = Z \times Z \rightarrow Z/n$$
$$(i, j) \rightarrow i + \lambda j \pmod{n}$$

3 Curve Generation

3.1 CM Method

Let E/Fp be our desired elliptic curve with CM discriminant D. The group order of E/Fp is #E(Fp) = p + 1 t, where t is the Frobenius trace. It is well known that p and t also satisfy the CM equation as 4p = t2 Ds2, where s Z. Note that the j-invariant of such a curve is also determined, and there are only 2, 4, or 6 possible group orders for a desired curve

4 Double Scalar Multiplication

Scalar multiplication is one of the basic operations defining vector space in linear algebra . In common geometrical contexts, scalar multiplication of a real euclidian vector by a positive real number multiplies the magnitude of the vector without changing its direction. The term scalar derives from this usage of a scalar is that which scales vectors. Scalar multiplication is multiply of scalar vector

Double scalar multiplication is the most time-consuming operation of ECDSA signature verification and, therefore, deserves efficient implementation and optimization. Formally, double scalar multiplication. It form k t + l Q and computes the sum of two scalar products, where t is fixed and Q is an arbitrary point. In the following, we review several approaches for performing the double scalar multiplication and describe how to speed up this operation by exploiting an endomorphism. For convenience, we assume that both scalars k and l are exactly m bits long.

4.1 Two Single Scalar Double Multiplication

The most straightforward method to perform the double scalar multiplication is to compute the two single scalar multiplications separately and then add up the results. The first scalar multiplication k takes a fixed and a-priori- known point as an input, which can be efficiently performed through the fixed-base comb method. The second scalar multiplication, l Q, is performed with an arbitrary base point Q not known in

advance. The simplest option for its computation is the binary method. In that case, the arbitrary-base scalar multiplication requires m point doublings and m/2 point additions in average. In total, the double scalar multiplication requires m + m/w point doublings and m/2 + m (2w −1) point additions on average. Windows methods (e.g., width-w non-adjacent form allow reducing the number of point additions in arbitrary-base scalar multiplications by using pre computations, but also they require m point doublings

4.2 Interleaving Method

A method to speed up the computation of k t + l Q is to perform them in a simultaneous (or interleaved) fashion by using Shamir's trick . This method first computes the sum of t and Q. Then, the scalars k and l are scanned simultaneously starting from the most significant bit. One adds t if k = 1 and li = 0, Q if k = 0 and li = 1, and S if k = li = 1. This method reduces the number of point doublings so that a double scalar multiplication requires m point doublings and 3m/4 point additions on average.

4.3 Joint Sparse Form

The method works analogously to the above interleaving method but uses also point subtractions for negative digits. A double scalar multiplication performed in an interleaved fashion with JSF requires m point doublings and only m/2 point additions on average

4.4 Window Method

Another approach to reduce the number of point additions in a double scalar multiplication is to use a window method .After that, we generate the look-up table with 15 points (line 4). Finally, the four scalar multiplications needed for computing k1 t + k2 φ(t) + l1 Q + l2 φ(Q) are performed simultaneously, i.e. in an interleaved fashion. A double scalar multiplication using Algorithm 1 requires approximately m/2 point doublings

5 Implementation Result

We implemented the arithmetic processor in Verilog and synthesized it with Design Compiler 2013.12 using the UMC 130 nm 1P8M Low Leakage Standard Cell Library with typical values (i.e. voltage of 1.2V and temperature of 25◦C). The area (in gate equivalents, GE) after placement and routing is calculated by dividing the overall area by the area of a single two-input NAND gate. The design has been synthesized for a clock frequency of 50 MHz, which is more than sufficient for common IOT devices such as RFID tags or sensor nodes.

5.1 Execution Time

As mentioned in the previous section, we implemented the multiplication, squaring, addition and subtraction to have constant execution time. Constant execution time (and, thus, constant pattern of operations) gives protection against simple side-channel attacks that target a single side-channel trace. Protection against more elaborated attacks relying on statistical analysis of one or more traces is left for future work.

5.2 Memory Optimised

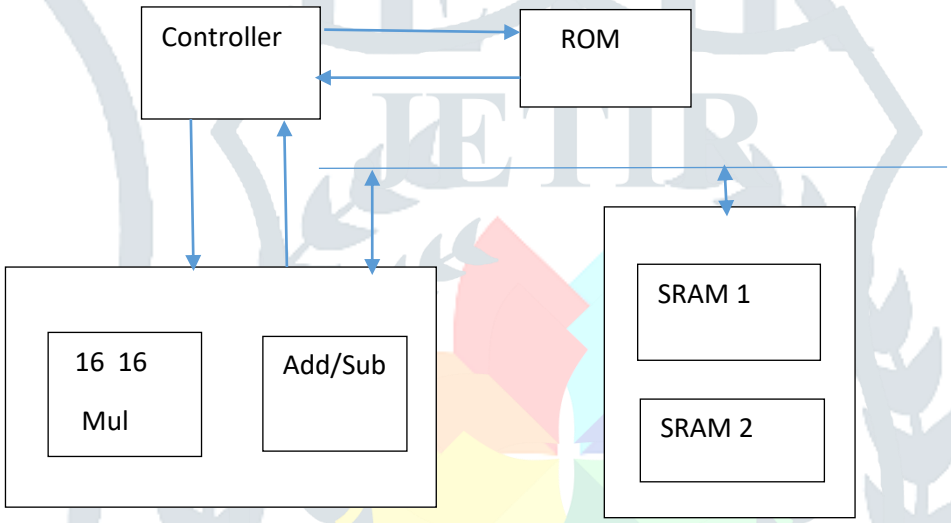
A look-up table with 15 ex- tended affine points requires 45 field elements to be stored in RAM. Instead of generating a look-up table with extended affine points, the memory-optimized implementation generates a look-up table with standard affine coordinates (x, y) and reduces the RAM requirements to only 30 field elements. In the process of look-up table generation, we adopt the point addition formula with Z1 = 1 and Z2 = 1 [14, Sect. 3.1] and directly convert the projective representation into standard affine representation for each point.

5.3 Latencies

The ALU computes field operations with the following latencies: multiplication 61 or 63 clock cycles, addition/subtraction 7–18 clock cycles depending on the required reductions (average 11.5), and addition/addition 7– 11 clock cycles (average 9). Division by two requires 12 or 17 clock cycles if the lsb is zero or one, respectively. Hence, when two divisions are performed in parallel, the average latency is 15.75 clock cycles. A Fermat-based inversion in Fp takes on average (206 + 14) 62 = 13, 640 clock cycles. One iteration of Alg. 4 requires 7 62 + 5 11.5 + 9 = 500.5 clock cycles on average. Computing only the point addition or point doubling parts of Algebra

5.4 Hardware Architecture

It consist of a micro controller, ROM, Prime field arithmetic unit and two dual ports SRAM. ROM is used to execute high level function such as point addition, point doubling The architecture of the ALU are one (16 16)-bit multiplier, one 3-input adder, a trailing-zero detection module (tlz), a left shifting module (l shifter), and a right shifting mod- are depicted. 16 bit datapath to achieve trade off in performance. ALU supports modular multiplication, modular squaring, modular addition, modular subtraction, modular inversion.



6 Conclusion

In this we introduced Twisted Edward Curve, to speed up double scalar multiplication. We used two hardware implementations utilizing the endomorphism and to resource constrained IOT devices and FPGAs for the server- side.

The processor is built in a 16-bit datapath with CMOS cell library and support various trade off between execution time and memory requirements. The verification core was designed to use parallel processing with two ALUs and RAM memories. This results both fast and compact FPGA implementation of double scalar multiplication.

It is used for achieving high throughputs for signature verifications in server- side operations related to the IOT by using parallel instances of the core inside one FPGA device