ABSTRACT

The data security, authentication and integrity have become an important and urgent need for health care information, confidential communication, storage and financial services etc. The public key cryptosystem is the most efficient way to secure data transaction and messaging. The challenge to implement the most popular public key cryptosystem, RSA (Rivest, Shamir and Adleman) is the rapidly growing key size. Elliptic Curve cryptography (ECC) has been considered an alternative to RSA. The effectiveness of using elliptic curve cryptography is that it provides same security level with shorter keys than in RSA. Therefore ECC can be used in smart cards, credit cards and mobile phones where area is a constraint. It is estimated that security level of 160 and 210 bits ECC cryptosystem is equivalent to the 1024 and 2048 bits RSA respectively.

The research on different algorithms and hardware accelerators have focussed on efficient implementation of elliptic curve scalar point multiplication Q=k.P. This is the fundamental operation of all elliptic curve cryptosystems. Software based implementations of ECC are flexible but inefficient because general purpose instruction set architecture of the underlying hardware is not optimized for cryptographic computations. An instruction set architecture can be extended to provide partial support for ECC related arithmetic operations. Hence a better approach would be to introduce a special arithmetic unit for accelerating modular operations or even complete scalar multiplications.

Two types of Finite Fields are generally used in ECC. Those are Finite Field over a large prime called as Galois Field GF(p) and Binary Field that is known as Galois Field GF(2^k). The hardware complexity to implement ECC in GF(p) is little bit higher than that of in GF(2^k) but the advantage is that the k-bit arithmetic unit is capable to process any i-bit data where 1≤i≤k. Confining designs to binary fields limit the flexibility. Since the objective of the research concentrates on Galois Field GF(p) much focus has been given to arithmetic over Galois Field GF(p).

The National Institute of Standards and Technology (NIST) has recommended elliptic curves over the five prime fields with moduli:

\[
\begin{align*}
p_{192} & = 2^{192} - 2^{64} - 1 \\
p_{224} & = 2^{224} - 2^{96} + 1 \\
p_{256} & = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1 \\
p_{384} & = 2^{384} - 2^{128} - 2^{96} + 2^{32} - 1 \\
p_{521} & = 2^{521} - 1
\end{align*}
\]

Except for p521, the powers appearing in these expressions are all multiples of 32. This property yields arithmetic operations especially fast on machines with word size 32. These arithmetic operations include addition, subtraction, multiplication, reduction and inversion. If all arithmetic operations are to be used and to represent a full range of integers in ‘n’ bits, then other NIST
recommended primes will not work except NIST recommended prime with modulus $2^{521} - 1$. This NIST recommended prime field is a Mersenne prime number which belongs to the group of numbers of the form $2^k - 1$.

In some of the earlier hardware accelerators reported, the focus is given to architecture flexibility which reduces the efficiency in terms of area and speed. A cryptography algorithm is secured as long as no effective attack is found. If there is a threat, the algorithm has to be replaced, in particular of switching to a higher key length which restricts operations with lower key length. In this situation, when working with higher key length the architecture can be more specific. This increases its efficiency in terms of area and speed with a compromise on flexibility. So the proposed method has focused on the concept of switching to higher key length and has been applied to NIST recommended prime field $p_{521}$ which has the highest key length among other NIST recommended prime fields.

Inversion is the costliest operation among other modular operations addition, subtraction and multiplication in terms of area and speed. Inversion operation is almost eliminated with projective coordinate systems with the cost of using parallel multipliers. But in small devices like smart cards where area is a constraint, adding more multiplier units need more memory and this method increases the cost. Speeding up inversion operation has been gaining attention because inversion is the most time consuming operation when affine coordinates are selected. Hence the proposed method has aimed to obtain an area efficient architecture of inversion unit with an improvement in throughput of point multiplication specifically for NIST recommended prime field $p_{521}$.

The two commonly used approaches for the computation of modular inverse over GF($p$) are Binary Inversion Algorithm (BIA) and Montgomery algorithm. Both of these algorithms are suitable for implementation in hardware. Montgomery Algorithm is attractive in the situations where fast reduction algorithms are not available. Fast reduction algorithm is available for the NIST recommended prime field $2^{521} - 1$. The Extended Euclidian Algorithm for computation of inversion requires expensive division operation and requires complex hardware. Another algorithm for the computation of inversion is by using Fermat’s theorem. But the number of multiplications required for the computation of inversion is proportional to the number of 1s of the exponent. The number of 1s in the NIST recommended prime field $2^{521} - 1$ is 521. Hence the most suitable algorithm to compute inversion over GF($p$) is Binary Inversion Algorithm. Hence the research has been focussed on reducing the architectural area of Binary Inversion Algorithm and increasing the throughput of point multiplication using the proposed architecture. This thesis has highlighted some properties of numbers of the form $2^k - 1$, arithmetic operations with these numbers with less hardware resources and methodology to achieve the objective.

The objective of this research is to achieve an area efficient architecture of Binary Inversion Algorithm and to increase the throughput of point multiplication specifically to prime field $p_{521}$ with a compromise on flexibility. This has been achieved
by employing the properties of numbers of the form $2^n - 1$. A new property of numbers of the form $2^n - 1$ has been found which has been used to reduce the area. Another property of Mersenne prime numbers has been used to increase the throughput of point multiplication. These properties have been applied to BIA separately and then combined together to achieve the above objective.

A new property of numbers of the form $2^n - 1$ has been found and this new property has been applied to Binary Inversion Algorithm and the algorithm has been modified. At the algorithmic level ‘division by 2 followed by addition’ operation has been replaced by ‘rotate right’ operation. Then the modified algorithm has been mapped to hardware which offered high throughput. The ‘division by 2 followed by addition’ operation at algorithmic level is performed in the hardware by ‘hard rotate right followed by addition’ and ‘rotate right’ operation at algorithmic level is performed by ‘hard rotate right’ in the hardware. The architectures of Binary Inversion algorithm those have been reported have two blocks of ‘division by 2 followed by addition’ which are computed in parallel for keeping throughput high. Since the area occupied by hard shift right and area occupied by hard rotate right are same, this technique has reduced the area occupied by two adder blocks. This proposed technique has reduced the area by removing two 521 bit adder blocks from the earlier reported architectures of BIA.

Another technique has been proposed to increase the throughput of point multiplication. It has been observed that most of the numbers in the upper half of number space of a finite field having the most significant bit ‘1’ need more number of iterations in BIA compared to the number of iterations require when the same number is complemented. This has been verified by the statistical analysis. Another property is that the inverse of a number and inverse of the number complemented over Mersenne prime number are also complementary. A point multiplication involves several computations of inversion. The above mentioned property and the observation made have been applied in the architecture of BIA. This technique has reduced the number of iterations (clock cycles) of most of the numbers in the upper half of number space and has speed up the computation of point multiplication. Reducing the number of clock cycles required for the computation of inversion of the numbers belonging to the upper half of number space has been obtained by complementing these numbers and by processing these numbers in BIA. The complementing has been performed with 521 EXOR gates with the MSB of that number as a common input to all the EXOR gates. The other input of EXOR gates are the respective bits of the same number. This has complemented only the numbers of above specified range and has reduced the number of clock cycles of most of the numbers of the above specified range needed for the computation of inversion. The required output is obtained by complementing the intermediate output obtained with BIA. This complement operation has been performed with 521 EXOR gates with MSB of the input common to all the EXOR gates with respective bits of intermediate output as another input. These two stages of complementing, one at the input and another at output need 1042 EXOR gates. This technique has increased the throughput of point multiplication but with 1042 more number of EXOR gates compared to the earlier architectures of BIA.
The proposed area efficient architecture to increase the throughput of point multiplication has been obtained by integrating the above described two techniques in the architecture of BIA. The new architecture proposed has achieved reduction in area and offered improvement in throughput of point multiplication. The area and the throughput improvement of the proposed architecture have been compared with earlier reported architectures of BIA. The proposed architecture has lesser number of logic gates and has computed the point multiplication with lesser number of clock cycles. Most of the earlier reported architectures on GF(p) have used simple carry propagation binary adders. The proposed method also uses similar kind of adders and the results have been tabulated based on this. The proposed arithmetic unit of inversion block has 1042 number of EXOR, 1042 number of two input OR and 2084 number of two input AND gates lesser than the earlier architectures.

The improvement in throughput obtained has been tabulated based on the statistical analysis. The statistical analysis on NIST recommended prime $2^{521}-1$ and other Mersenne primes have shown that proposed architecture has reduced on average 452.69 number of clock cycles during computation of point multiplication by binary point multiplication algorithm when affine coordinate is used. The throughput improvement obtained is approximately 0.1%. The functionality of the proposed architecture has been verified.

The proposed method of replacing the addition operations in the computation of inversion has been also extended to software environment where point multiplication is computed by instruction set of general purpose microprocessors. The statistical analysis has proved that the proposed method approximately has reduced 2,03,156.345 addition operations (instructions) during point multiplication by using binary point multiplication algorithm, a high significant improvement in the speed compared to the existing techniques.