The use of FFT's and other algorithms for fast Elliptic Curve operations

C. Dreyer

Abstract— **Over the past few years there has been a lot of research into public key cryptography. In recent years Elliptic Curve Cryptography (ECC) has become increasingly attractive due to the relatively small key sizes (as opposed to RSA for example). This paper gives an overview of some of the basic principles of ECC and then presents some well known, as well as some not-so-well known algorithms for computations that can speed up Elliptic Curve operations. The paper ends with a short overview of attacks on cryptosystems and attempts to give pointers on how to avoid Simple Power Attacks (SPA's) and Differential Power Attacks (DPA's).**

I. Introduction

Since their introduction around 1985 by Koblitz and Miller, there has been a wealth of new research conducted into the feasibility of using elliptic curves as a tool for cryptography. Many papers have been written on this subject, and many advances have been made. This paper attempts to give an overview of some of these advances and also to bring to light methods that have not received much attention. Section 2 of this paper gives some mathematical background upon which elliptic curve cryptography is based. From Section 3 onwards we discuss methods of speeding up computations of Elliptic curves. Section 3 discusses some multiplication methods that help speed up computations on elliptic curves (scalar computation is done a lot in ECC, and so it makes sense to try to speed up this portion of the process). Section 4 discusses how choosing a good elliptic curve helps speed up computations. Section 5 deals with the use of the Fast Fourier Transform (FFT) in computations, and section 6 ends with a survey of attacks that an ECC implementation should expect, and how to thwart these attacks.

II. Definitions for Finite Fields, Elliptic curves, and Fourier Transforms

In this section we introduce the unfamiliar reader to some of the terminology, definitions and theorems associated with finite fields and elliptic curves. Those readers who already have a basic knowledge may wish to skip this section, but may at the same time benefit from notational definitions. For a more thorough development of Finite Field theory see [1] and also [2].

A. Finite fields

Definition (Group): A group, G, is a non-empty set with the following properties:

Author is Undergraduate in the Department of Electrical & Computer Engineering, Oregon State University, Corvallis, Oregon 97331. E-mail:{dreyerca}@onid.orst.edu

(i) (ab)c = a(bc) for all a,b,c in G.

(ii) There exists an element e in G, with ea = ae = a for all a in G.

(iii) For each a in G, there exists b in G such that ab = ba = e.

Definition (Abelian group): A group, G, is Abelian if ab = ba for a,b in G.

Definition (Ring): A ring is a non-empty set, R, with two binary compositions (for example addition and multiplication) defined on it. The ring must satisfy

(i) R is Abelian wrt the additive composition.

(ii) Multiplication in R is associative.

(iii) Both the distributive laws : a(b+c) = ab + ac and (a+b)c = ac + bc hold.

Definition (Field): A Field, is a set with at least two elements with two compositions defined on it.

Definition (Galois field): A Galois field and a finite field are synonymous. A finite field is simply a field with a finite number of elements.

Definition (F*): F* is the set of all non-zero elements of F. ie F* = F - 0.

Definition (Characteristic of F): The characteristic of a field F is the number of times we can add 1 onto the current number until the modulus reduces this number to zero. The characteristic of a non-finite field (F $\supseteq$ rational numbers) is defined to be zero.

B. Elliptic Curves

Definition (Elliptic Curve): An elliptic curve is defined as a set of 2-dimensional coordinates (x,y) which form a solution set to the bivariate cubic equation:

$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$

over a field F. If the characteristic of F is not 2 or 3, then one can transform the above equation into:

$y^2 = x^3 + ax + b$

a,b element of F.

If F is a Galois field, GF(2^n), with characteristic 2, then we can reduce the bivariate cubic equation to:

$y^2 + xy = x^3 + ax^2 + b$

The set of points (x,y) on the elliptic curve, as well as the point at infinity (by definition), denoted infinity, form an Abelian group. To be an Abelian group, the set must satisfy the conditions of the definition. We define addition over F as follows:

For char(F) not equal to 2 or 3:

Let P = (x_1, y_1) not equal to infinity and Q = (x_2, y_2) be points on the elliptic curve. Denote inverse of P by -P = (x_1, -y_1) not equal to Q. Then the sum of P + Q = (x_3, y_3) can be calculated from the following formulas:

$x_3 = \lambda^2 - x_1 - x_2$
$y_3 = \lambda(x_1 - x_3) - y_1)$
$\lambda = \frac{y_2 - y_1}{x_2 - x_1}$ if $P \neq Q$
$= \frac{3x_1^2 + a}{2y_1}$ if $P = Q$

To subtract Q - P, simply add the inverse of P to Q.

For char(F) = 2:

Let P = (x_1, y_1) not equal to infinity be a point on the elliptic curve, and define it's inverse by -P=(x_1, $x_1 + y_1$). Let Q = (x_2, y_2) be a second point with Q not equal to -P. The sum of P and Q, P+Q = (x_3, y_3), can then be found from:

for $P \neq Q$:
$x_3 = \lambda^2 + \lambda + x_1 + x_2 + a$
$y_3 = \lambda(x_1 x_3) + x_3 + y_1$
$\lambda = \frac{y_1 + y_2}{x_1 + x_2}$

for P = Q:
$x_3 = \lambda^2 + \lambda + a$
$y_3 = x_1^2 + (\lambda + 1)x_3$
$\lambda = x_1 + \frac{y_1}{x_1}$

For information on how elliptic curve cryptography (ECC) works, security of ECC, and some information on the basics of hardware and software implementations the reader is encouraged to read [3].

C. Fourier Transforms

Definition (Fourier Transform): The Fourier Transform of a function, denoted *F(X)*, is defined as:

$F(X) = X(\omega) = \int_{\infty}^{-\infty} f(x)e^{-j\omega t}\,dt$.

Definition (DFT): The Discrete Fourier Transform of a set of numbers is defined by: $X_k = \sum_{n=0}^{N-1} x[n]e^{-j2\pi kn/N}$. The computation of the DFT of a set of N numbers, requires N^2 multiplications.

Definition (Nth root of unity): The Nth root of unity, denoted ω_n, is defined by: $\omega_n = e^{j2\pi k/2n} = \omega^n$.

Definition (Fourier Transform (again)): We can write the Fourier Transform in the following form:

$F(X) = (X_0, X_1, ..., X_{2N-1})$ and can compute the $X_k's$ by : $X_k = \sum_{n=0}^{2N-1} x_n \omega^{nk}$

Definition (FFT): The Fast Fourier Transform is an algorithm which can compute the DFT of a set in $Nlog_2N$ multiplications. The mechanism by which this is done is based on the fact that two numbers being multiplied (or in Signal Processing lingo "convolved"), can each be split in half recursively and then multiplied. Basically, we can rewrite the Fourier coefficients by:

$X_k = \sum_{n=0}^{N-1} x_{2n}(\omega^2)^{kn} + \omega^k \sum_{n=0}^{2N-1} x_{2n+1}(\omega^2)^{kn}$

In other words, find the X_k coefficients by splitting the set x into two parts:

$a = (x_0, x_2, x_4, ..., x_{2n-2})$ and $b = (x_1, x_3, x_5, ..., x_{2n-1})$

Next, compute the Fourier Transform of a (A_k) and b (B_k). Finally deduce the Fourier Transform, X_k, using the formulas:

$X_k = A_k + \omega^k B_k$
$X_{(N/2)+k} = A_k - \omega^k B_k$
(For a derivation, see [4]).

III. Multiplication on elliptic curves

The speed of scalar multiplication of a point P on an elliptic curve is of paramount importance since this is a fundamental operation on elliptic curves which is executed relatively often compared to other operations. Scalar multiplication on points in an elliptic curve group is analogous to exponentiation in a multiplicative group. There are many ways of computing the scalar product kP, the easiest of which is the double-and-add algorithm. There are a number of other algorithms which are generally either more efficient or faster. Often one must choose one of the elements in this tradeoff - a faster algorithm, or an algorithm that takes up more memory. The algorithm choice is thus implementation dependent.

[5] gives an implementation which allows fast multiplication without doing any precomputation. This is a desirable feature in implementations such as cellular phones, MP3 players, or smart cards where memory and speed are both at a premium. Their algorithm looks like this:

Algorithm
Input: Integer k ≥ 0 and point $P = (x, y)\epsilon F$
Output: $Q = kP$

1: if $k = 0$ or $x = 0$ then output (0,0) and stop
2: Set $k \leftarrow (k_{l-1}...k_1k_0)_2$
3: Set $X_1 \leftarrow x, Z_1 \leftarrow 1, X_2 \leftarrow x^4 + b, Z_2 \leftarrow x^2$
4: for i from l-2 downto 0 do
  if $k_i = 1$ then
    Madd(X_1, Z_2, X_2, Z_1), Mdouble(X_2, Z_2)
  else
    Madd(X_2, Z_2, X_1, Z_1), Mdouble(X_1, Z_1)
5: return Q = Mxy(X_1, Z_1, X_2, Z_2)

For multiplication algorithms related to RSA, the reader is referred to [6], and for multiplication algorithms for ECC, see [7].

IV. Elliptic Curve selection

Since so many of the computations for ECC require scalar multiplication, this area has received a lot of attention. Not so much attention, however, has been given to the choice of elliptic curve. Choosing a suitable elliptic curve can be very beneficial, because certain elliptic curves have properties that make scalar multiplication easier and faster.

In [8] this issue of Elliptic Curve choice (particularly curves of prime order) is examined. The authors precalculate class polynomials as a separate off-line process, and then choose a discriminant, after which they begin to search for primes (traditionally primes are chosen first, and discriminants found later).

The authors use a variant of the complex curve generation algorithm, which allows them to construct and store the class polynomial, and then to search for primes whose Complex Multiplication (CM) discriminants are in the set. Their algorithm is as follows:

1. Off-line: Determine a set δ of CM discriminants with correspondingly small class numbers.

2. Off-line: Calculate and store the class polynomials of CM discriminants in δ.

3. Randomly select an element D of δ and obtain the corresponding class polynomial $H_D(x)$.

4. Search for a prime number p satisfying the equation $4p = t^2 + Ds^2$ (First select random t and s, and then determine if p prime).

5. Compute $u_1 = p+1-t$ and $u_2 = p+1+t$, the orders of the elliptic curve candidates, and determine if either has an admissible factorization. If not, go back to 4, else continue.

6. If u_1 has proper factorization set $u = q_1$, otherwise $u = q_2$.

7. Find a root j_0 of $H_D(x)$ mod p (where j is the j-invariant of the curve).

8. Set $k = j_0/(1728 - j_0) mod p$ and the curve of order u_1 or u_2 will be:

$\xi_c : y^2 = x^3 + ax + b$

where $a = 3kc^2$, $b = kc^3$ and $c \epsilon F_p$ is chosen randomly.

9. Check the resultant curve's order. If it is equal to u, then stop. Otherwise, select a non-square number $e \epsilon F_p$, and calculate the twist by e, $\xi_e(F_p) = x^3 + ae^2 + be^3$.

V. Multiplication using the Fast Fourier Transform

The idea of all mathematical transforms in Electrical Engineering, is that one can transform a rather difficult, or time consuming process into a simpler, faster process. In the case of the Fourier and Laplace transforms this simplification comes from the fact that one can transform differential equations into simple algebraic equations, and convolution can be transformed into simple multiplication.

For multiplying two numbers together, we are not really interested in convolution, but essentially this is what is being done. To see this, consider the process by which we multiply two numbers using the FFT:

Suppose we want to multiply together two numbers (x and y) together, then

Step 1.

$x = (x_0, x_1, x_2, ..., x_{2N-1})$

$y = (y_0, y_1, y_2, ..., y_{2N-1})$

Note that if either x or y has more digits than the other we "pad" the smaller number with zeros. (i.e if x = 101 and y = 1101, then pad x with a single zero to get x = 0101)

Step 2.

Compute the FFT of x and y:

$FFT(x) = X_k = (X_0, X_1, X_2, ..., X_{N-1}, 0, ..., 0)$

$FFT(y) = Y_k = (Y_0, Y_1, Y_2, ..., Y_{N-1}, 0, ..., 0)$

Notice that the length of X and Y are double the length of x and y respectively.

Step 3.

Compute the product Z_k using your favorite method:

$Z_k = X_k Y_k$

where $Z_k = (Z_0, Z_1, Z_2, ..., Z_{2n-1})$

and $Z_i = X_i Y_i$ (i.e. compute product point-wise)

Step 4.

Compute the inverse Fourier transform z, of Z.

$IFFT(Z) = z = (z_0, z_1, z_2, ..., z_{2N-1})$

Notice that the length of z is the same as the length of x and y. Step 5.

The product of x and y is then

$z = \sum_{n=0}^{2N-1} z_n B^n$

Essentially what the above algorithm does is to take x and y, transform them into the Fourier domain, then slide them past one another multiplying at each point, and then take the inverse Fourier transform of the result. This is exactly the same concept of convolution being applied to two signals.

VI. Errors incurred by using the FFT multiplication algorithm

When the FFT multiplication algorithm is applied to two numbers in a software environment, it is possible for some error due to truncation to occur. In the application of cryptography this sounds disastrous, but it turns out that it's not so bad. If we take the inverse transform of the slightly erroneous result and then round each element to the closest value, then this error is eliminated.

In a hardware environment, this will not occur provided that there are enough registers to store the Fourier transforms of x and y. Recall from the above algorithm that if the larger of x and y has a length N, then the result of the Fourier transform will require 2N registers.

For more information about FFT multiplication and errors, see [9].

VII. Architectures for Elliptic Curve Cryptosystems

There are many emerging architectures for systems dedicated to elliptic curve cryptography. An example of such an architecture is [10]. This article introduces an architecture with an optimized bit-parallel squarer, digit-serial multiplier and two programmable processors. The system consists of program memory, a main controller(MC), an arithmetic control unit (ACU) and an arithmetic unit (AU).

The AU consists of a Least Significant Digit first (LSD) multiplier, a bit-parallel squarer (that can square in one clock cycle), a register file, an accumulator, and a zero tester. The multiplier is responsible for field additions, squares, multiplications and inversions.

Using this simple architecture, the authors reported speed results of at least 19 times faster than any documented hardware implementation, and at least 37 times faster than any documented software implementation.

Also of interest is [11] which gives an overview of an implementation utilizing a Samsung CalmRISC microcontroller with a MAC2424 coprocessor. The beauty of this implementation is that it can process most instructions in only one clock cycle.

The authors tested three different implementations for field multiplication and squaring: Karatsuba-Offman, (see [2]), column major method, and row major method. They used a slightly improved version of the Bailey-Paar inversion algorithm for field inversion. Using these methods,

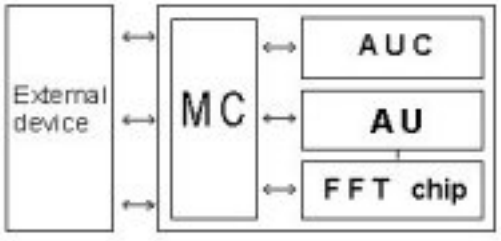

Fig. 1. Scalable EC architecture with FFT chip

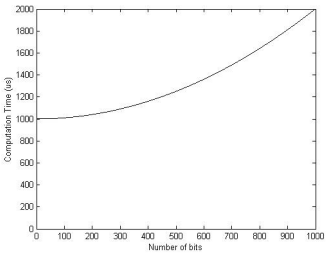

Fig. 2. Computation time versus number of bits

they were able to implement elliptic curve exponentiation in 122ms for a 50ns clock cycle.

Recently Motorolla released their own cryptochip. The Motorolla MPC185 is a very versatile chip which can run many different types of algorithms. For more information, see Motorolla's website at [12].

The reader may now wonder, how can one implement an Elliptic Curve Arithmetic Unit (AU) using the Fourier Transform algorithm for fast multiplication. There are of course many ways to accomplish this task. A possible implementation is to use a Spiffee ULP FFT chip. This chip, designed by Stanford University, is ideal for implementations like cell phones and smart cards because it runs on a 0.4 V power supply, with a mere 8mW of power consumption, and runs at 85 MHz. See Table 1 for a comparison of some competing FFT chips. For more comparisons, see [13].

Table 1: Comparison of FFT chips.

Chip	Voltage	Power	Clock
Spiffee ULP	0.4 V	8 mW	85 MHz
DSP-24	3.3 V	3500 mW	100 MHz
DoubleBW	3.3 V	8000 mW	128 MHz
C40 (TI)	5 V	4500 mW	60 MHz

It would also be desirable to follow the example of [10], because their layout is "scalable". The architecture of our device would look somewhat like Fig 1.

The only difference between Fig 1 and [10]'s implementation is the added FFT chip which computes the FFT of x and y which the AUC (Arithmetic Unit Controller) tells the AU (Arithmetic Unit) to send to it. The FFT chip then sends the FFT of the number back to the AU, which does the multiplication. The AUC then tells the AU to send the result back to the FFT chip which computes the inverse FFT. The FFT chip then sends the result back to the AU and the controlling program then receives this value.

Using this implementation, we can project fairly good speeds of computation. For a comparison of computation time (in microseconds) to the number of bits of numbers being multiplied, see Graph1.

VIII. The Hartley Transform

The Hartley Transform is another transform, much like the Fourier Transform. It differs from the Fourier Transform in that the Hartley Transform produces a real output given a real input, and is its own inverse. This means that multiplications with the Hartley Transform require no inverse transform of the result. This is a very useful property. The discrete Hartley transform is defined by:

$H[n] = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x_n[cos(\frac{2\pi kn}{N}) - sin(\frac{2\pi kn}{N})]$

as presented in [14].

IX. Power attacks on Elliptic Curve Cryptosystems

Like any system that involves money, there will always be people who wish to defeat the security of the system in order to gain access to it. It is thus extremely important to keep in mind that when you build a cryptosystem, you do not only need to protect the data, but also the cryptosystem itself. Over the years designers have come to realize this very real danger, and have begun to design their systems to protect both data and the system itself. In order to test the reliability of the system a number of attacks have been designed to attempt to defeat the system's security.

(i) Simple Power Attack (SPA): This attack is based on the fact that by looking at the power consumption of a cryptosystem one can guess what operation the system is performing, and thus find ways to compromise the system's internal security features.

(ii) Differential Power Analysis (DPA): This is an extremely powerful attack that consists of statistical analysis of the power consumption of a cryptosystem. It is possible to find the secret key in a cryptosystem by DPA in as few as 1000 encryptions.

In order to make a cryptosystem resistant to SPA, one should ensure that the algorithms do not depend on the data being processed (for example, branches in programs should not be initiated by particular data).

To immunize a cryptosystem against DPA is a more difficult task. One can do various things to increase the number of encryptions before the key is extracted, but with enough time, the key probably can be extracted. The trick is to make it infeasible to do this (i.e. make it take 1000 years or so to extract). [15] gives a few ways that one can do this:

(i) Let $\sharp E$ be the number of points on the curve. Select a random d of n bits (d=20 is sufficient). Compute $k' = k + d * \sharp E$. Compute k'P = kP since $\sharp EP = \infty$.

(ii) "Blind" out P by adding a secret, random point, R, for which we know S = kR. Do scalar multiplication by k(R+P) and then subtract S = kR from this result to get Q = kP.

(iii) Randomize the projective coordinate.

See [15] for a more in-depth discussion of both the problems and solutions to cryptosystem security.

X. Conclusion

As [16] says, few algorithms are as elegant, as technologically advanced, or as dramatically different from the norm, as the use of the Fast Fourier Transform for multiplication.

We have shown that the FFT is a fast algorithm for implementing multiplication on elliptic curves, and is relatively easy to integrate into an existing architecture.

We have also tried to show that this process should be scrutinized during design because the cryptosystem may be attacked, and should not fail during an attack.

Elliptic Curve cryptography may very well be the cryptographic algorithm of choice for the 21st century, and the FFT may be a significant aid in this process.

References

[1] A.J. Menezes ed., "Applications of finite fields," *Unpublished*, 1993.

[2] F. Rodriguez-Henriquez, "New algorithms and architectures for arithmetic in GF(2^m) suitable for elliptic curve cryptography," *Oregon State University PhD thesis*, 2000.

[3] S. Vanstone N. Koblitz, A. Menezes, "The state of elliptic curve cryptography," *Kluwer Academic Publishers, Boston*, vol. 19, pp. 173–193, 2000.

[4] B.S. Heck E.W. Kamen, "Fundamentals of Signals and Systems using the Web and Matlab," *Prentice Hall*, 2000.

[5] R Dahab J Lopez, "Fast multiplication on elliptic curves over GF(2^m) without precomputation," *CHES 1999, Springer-Verlag*, vol. 1, pp. 316–327, 1999.

[6] C.K. Koc, "High-speed RSA implementation," *Unpublished*, 1994.

[7] J.E. Cremona, "Algorithms for modular elliptic curves," *Cambridge University Press*, 1992.

[8] C.K. Koc E. Savas, T.A. Schmidt, "Generating elliptic curves of prime order," *CHES 2001, Springer-Verlag*, pp. 145–161, 2001.

[9] P. Sebah X. Gourdon, "http://numbers.computation.free.fr," *Internet*, 2001.

[10] C. Paar G. Orlando, "A high-performance reconfigurable elliptic curve processor for GF(2^m)," *CHES 2000, Springer-Verlag*, pp. 41–56, 2000.

[11] P.J. Lee J.W. Chung, S.G. Sim, "Fast implementation of elliptic curve defined over GF(p^m) on CALMRisc with MAC2424 coprocessor," *CHES 2000, Springer-Verlag*, pp. 57–70, 2000.

[12] Motorola, "MPC185 fact sheet," *http://e-www.motorola.com*, 2002.

[13] Stanford University, "http://nova.stanford.edu/ bbaas/spiffee.html," *Internet*, 1997.

[14] Wolfram, "http://mathworld.wolfram.com/hartleytransform.html," *Internet*, 1999.

[15] J. Coron, "Resistance against differential power analysis for elliptic curve cryptosystems," *CHES 1999, Springer-Verlag*, vol. 1717, pp. 292–302, 1999.

[16] J.D. Lipson, "Elements of algebra and alebraic computing," *Addison-Wesley*, 1981.