METHOD AND APPARATUS FOR CONSTRUCTING EFFICIENT ELLIPTIC CURVE CRYPTOSYSTEMS

Field of the Invention

The present invention relates to the field of implementing elliptic curve cryptosystems, and particularly to methods and apparatus for efficient implementation thereof. In this regard the present invention may be applied to information and document security systems using public key encryption technology, including systems where such operations are performed by low cost low power computing devices.

Background

With the increasing implementation of electronic communication more and more information is stored in electronic form. This form of storage is more efficient and space-saving as compared with paper documents, but electronic information is also subject to different, and potentially damaging, security issues. That is, electronic information is more prone to unauthorised disclosure, alteration, substitution and destruction.

A number of approaches have been developed to address these problems, one being cryptography. Cryptography transforms electronic data to a modified form and the transformation is controlled by the use of a key or keys, which takes the form of an electronic string.

One type of encryption is public-key encryption, where both the originator of the information and the recipient have different keys, being private and public keys respectively. Various types of public key cryptographic systems have been developed, including elliptic curve cryptography.

The security of an elliptic curve cryptosystem (ECC) is measured by the largest prime factor of the curve order, which is in practice approximate to the field order. The finite field order is the number of elements it contains. Therefore the field size in bits is usually taken as the security parameter of an ECC. Currently, 160 bit is regarded as the lower bound for the field size used in ECCs.

An ECC typically uses an elliptic curve as the group acting the role of *GF(p)* as in traditional Deffe-Hellman and ElGamal schemes. An ECC over a finite field requires arithmetic operations of addition, multiplication, squaring and inversion. Additionally, subtraction and modular arithmetic operations may also be required.

An elliptic curve is defined over a finite field *K*, and can have either affine or projective representation. The group operation on an elliptic curve is formulated in operations in the underlying finite field. In affine representation, one curve operation (point addition or doubling) needs a few field multiplications and one inversion, while in projective representation, one curve operation needs many more multiplications but no inversion. The cost ratio of multiplication/inversion is the main concern on choice between affine or projective representation, and the cross-point is around 7.

While various ECC methods have been developed, in general the technology is either not sufficient in performance, or the hardware required for implementation is too expensive.

There is therefore the need for a more efficient ECC method, particularly a method that does not require costly hardware for implementation.

The main task for building an efficient ECC is to construct a finite field of size exceeding the security parameter and with efficient field operations.

In this regard, the two main types of field constructions for ECC are *GF(p)* and $GF(2^n)$ in polynomial basis. These constructions have reasonable performance for desktop applications. For *GF(p)*, inversion is very slow, and projective representation must be used. For $GF(2^n)$, multiplication is slower than that for *GF(p)*. This is due to the fact that multiplication of binary polynomials has to be implemented completely in software while integer multiplication can utilize the built in instruction for multiplication of two word-size integers. Inversion in $GF(2^n)$ is implemented using extended Euclidean division. Although $GF(2^n)$ with polynomial basis has reasonable performance on desktop computers, both multiplication and inversion have complexity $O(n^2)$.

One method for implementing ECCs for desktop computers uses Optimal Extension Fields (OEF) [D.V.Bailey and C.Paar, "Optimal Extension Fields for Fast Arithmetic in Public-Key Algorithms", Proceedings of Advances in Cryptology - Crypto'98, pp. 472-485, Springer Verlag, 1998].

There are two types of OEFs. Type I OEF is defined as $GF(p^m)$ with irreducible polynomial $X^m - w$ for some small integer w where $p = 2^n \pm 1$ is a Fermat or Mersene prime. Type II OEF is defined as $GF(p^m)$ with irreducible polynomial $X^m - 2$ where $p = 2^n - c$; $[c] < n/2$ is a pseudo-Mersene prime. The multiplication in an OEF can make use of Karatsuba-Ofman technique to improve efficiency. There are 3 approaches to implement the inversion in an OEF. The first one is to compute the inverse of an element as raising it to a power of $q-1$, however it needs a lot of field multiplications. The second one uses a modified almost inverse algorithm [E. J. Lee, D. S. Kim, and P. J. Lee, Speed up of GF(p^m) Arithmetic For Elliptic Curve Cryptosystems. Proceedings of ICICS'98, Berlin, 1998. Springer Lecture Notes in Computer Science], however it needs about $3n^2$ multiplications in $GF(p)$. A third method [T. Kobayashi, H. Morita, K.Kobayashi, and F. Hoshino. Fast Elliptic Curve Algorithm Combining Frobenius Map and Table Reference to Adapt to Higher Characteristic. Advances in Cryptography-EUROCRYPT'99. Springer-Verlag, 1999] uses linear transformations which is only efficient for m <4. With these methods, the inversion in OEF is still relatively slow compared with multiplication.

Therefore it is apparent that in many ECC methods, the inversion operation is a bottleneck of ECC performance.

There is therefore a need for a more efficient mechanism for effecting inversion operations as well as optimizing other basic operations.

There are various hardware implementations of finite field operations such as described in United States Patent Numbers 5,612,910, 5,768,168 and 6,003,057. The drawback of these implementations, however, is that such circuits are too large and hence too expensive for a typical ECC application.

There is therefore a need for an improved apparatus and/or method for improving the efficiency of field operations in ECCs.

The present invention seeks to overcome or at least ameliorate at least one of the problems of the prior art.

Summary of the Invention.

In a first aspect the present invention provides a method of implementing elliptic curve cryptography including performing arithmetic operations over a field K_o ; and using a result of the arithmetic operations over the field to perform arithmetic operations in one or more extension fields.

According to another aspect, the present invention provides a method of electronically converting an electronic message to an encrypted message for transmission over a transmission medium, said method comprising the steps of: using an ECC to perform arithmetic operations on a private key and a point, wherein said point is a point on an elliptic curve over a finite field K_o; using a result of the arithmetic operations over the field to perform arithmetic operations in one or more extension fields K_j, based upon the operations in the previous field K_{j-1}, in order to determine an enciphering key; using an encryption/decryption means to convert said electronic message to said encrypted message using said enciphering key; and using a transmitting means to transmit said encrypted message over said transmission medium.

According to a further aspect, the present invention provides a computer program product including a computer usable medium having computer readable program code and computer readable system code embodied on said medium for implementing elliptic curve cryptography within a data processing system, said computer program product further including computer readable code within said computer usable medium for constructing a finite field K_o, such that the size of the field exceeds a security parameter k; and performing arithmetic operations in K_o and in at least one subsequent extension field K_j, based upon the operations in the previous field K_{j-1}.

According to a still further aspect, the present invention provides a function module for performing large finite field operations comprising: (a) a plurality of devices for carrying out arithmetic operations in a field K_0, being from the following group:

i) One or more K_0-adders for performing additions and/or subtractions in K_0.

ii) One or more K_0-multipliers for performing multiplications in K_0.

iii) One or more K_0-inverters for performing inversions in K_0.

(b) Logic means for utilizing the devices in (a) to iteratively form one or more multipliers and/or inverters in one or more extension fields K_l in order to carry out arithmetic operations in the one or more extension fields.

The essence of the present invention lies in the features of utilizing the operators in the underlying finite field for an ECC that is built up recursively by a series of smaller and smaller sub-sub-field operations. The present invention is based upon the realisation that an operation in K_n can be factorised into a plurality of operations in K_0 which are more efficient.

In this way, the arithmetic operations are simplified and hence the efficiency improved. Also, for hardware implementations, only operations in the base field need be circuit integrated, and subsequent field iterations can all be implemented using this hardware in combination with additional programming logic. This therefore greatly reduces the size and cost of the hardware.

Brief Description of the Drawings

A preferred embodiment of the present invention will now be described, by way of example only, with reference to the accompanying drawings in which:

Figure 1 which illustrated a flow chart of iterative arithmetic operations in a plurality of expansion fields K_j according to an embodiment of the invention.

Figure 2 illustrates a flow chart of a method of encrypting a message for transmission according to an embodiment of the invention.

Detailed Description

The efficiency of field operation implementation generally depends on the hardware. In ECC applications, there are three standard types of hardware: powerful general-purpose processors for desktop computers, microprocessor for digital devices such as smart cards and hand-phones, and specialized circuits. For these different types of hardware, the most efficient choice of field construction will differ.

In this regard, a first embodiment of the present invention will not be described with reference to Figure 1:

Let K be any finite field. An extension $K^{(n)}$ of K is defined by an irreducible polynomial $P(X)$ of degree n over K. Elements of $K^{(n)}$ are polynomials of degree $<n$. Addition in $K^{(n)}$ is just addition of polynomials. Multiplication in $K^{(n)}$ is defined to be multiplication of polynomials mod $P(X)$. Inversion of $A(X)$ is define to be the polynomial $B(X)$ such that $A(x)B(X) \equiv$ mod $P(X)$.

The multiplication in $K^{(n)}$ is carried out in two steps. The first step is multiplication of polynomials. In this regard, the following algorithms may be used for this step:

Multiplication of Polynomials of Degree 1

Input: $A(X) = a_0 + a_1X$; $B(X) = b_0 + b_1X$.

Output: $C(X) = A(X)B(X) = c_0 + c_1X + c_2X^2$.

Begin

$$c_0 = a_0b_0;\ c_2 = a_1b_1;\ c_1 = (a_0 + a_1)(b_0 + b_1) - c_0 - c_2;$$

End

Multiplication of Polynomials of Degree 2

Input: $A(X) = a_0 + a_1X + a_2X^2$; $B(X) = b_0 + b_1X + b_2X^2$.

Output: $C(X) = A(X)B(X) = c_0 + c_1X + c_2X^2 + c_3X^3 + C_4X^4$.

Begin

$m_0 = (a_0 + a_1)(b_0 + b_1)$; $m_1 = (a_1 + a_2)(b_1 + b_2)$; $m_2 = (a_0 + a_2)(b_0 + b_2)$; $m_3 = a_1b_1$;

$$c_0 = a_0b_0;\ c_1 = m_0 - c_0 - m_3;\ c_4 = a_2b_2;$$

$$c_3 = m_1 - m_3 - c_4;\ c_2 = m_2 + m_3 - c_0 - c_4;$$

End

Note: In above formulae, addition and subtraction is the same as X or when characteristic is 2.

The second step multiplication in $K^{(n)}$ is reduction mod $P(X)$. The complexity of this step depends on the choice of $P(X)$. The choices of $P(X)$ and corresponding implementation of this step is illustrated in the following subsections.

The inversion in $K^{(n)}$ can in general be implemented by the modified extended Euclid algorithm which needs an inversion in K and about $3n^2$ multiplications in K. Another method to invert $A(X)$ is solving the linear equation $A(X)B(X) = 1 \bmod P(X)$ where $B(X)$ is regarded as the unknown and multiplication by $A(X)$ is regarded as a linear transformation on $K^{(n)}$. When $n = 2$, both the two methods result in the same algorithm as follows:

Inversion Algorithm in Extension Field of Degree 2

Assume $P(X) = X^2 + bX + a$.

Input: $A(X) = a_0 + a_1X \in K^{(2)}$.

Output: $B(X) = b_0 + b_1X = A(X)^{-1} \in K^{(2)}$.

Begin

$$r = ba_1 - a_0;\ s = ra_0 + aa_1^2;\ t = s^{-1};$$

$$b_0 = tr;\ b_1 = ta_1;$$

End

When $P(X)$ has simple coefficients a, b, this algorithm requires three multiplications and one squaring and one inversion in K. For odd characteristic, this is roughly 4 multiplications and 1 inversion; and for even fields it is little more than three multiplications and one inversion, since squaring is much cheaper in this case.

When $n = 3$, solving a linear equation is a preferred approach, which results in the following algorithm:

Inversion Algorithm in Extension Field of Degree 3

Assume $P(X) = X^3 + cX^2 + bX + a$.

Input: $A(X) = a_0 + a_1X + a_2X^2 \in K^{(3)}$.

Output: $B(X) = b_0 + b_1X + b_2X^2 = A(X)^{-1} \in K^{(3)}$

Begin

$r_1 = a_0 - ba_2$; $r_2 = a_1 - ca_2$; $s_1 = -(aa_2 + br_2)$; $s_2 = r1 - cr_2$; $r = r_1s_2 - r_2s_1$;

if $r = 0$ {

$s = (aa_2s_1 - ar_2r_1)^{-1}$; $b_0 = 0$; $b_1 = -ss_1$; $b_2 = sr_1$;

}

else {

$s = a_1s_2 - a_2r_2$; $t = a_2r_1 - a_1s_1$; $u = -(ra_0 + asa_2 + atr_2)^{-1}$;

$b_0 = -ur$; $b_1 = us$; $b_2 = ut$;

}

End

When $P(X)$ has simple coefficients a, b, c, this algorithm requires no more than twelve multiplications and one inversion in K.

In the next subsections, we will illustrate how to select the irreducible polynomial for each extension step.

Selecting irreducible polynomials: Case of characteristic 2

Assume $K_0 = GF(2^n)$. If in the first extension step K_1/K_0, the extension degree is 3 and n is prime to 3, then let the irreducible polynomial be $P(X) = X^3 + X + 1$; if $3 | n$ the simplest $P(X)$ depends on the details of the said K_0-multiplier and can be determined by computer searching. Now we can let K_1 play the role of K_0 in the subsequent extension steps. So we may assume all extensions starting from $K_0 = GF(2^n)$ are of degree 2.

If n is odd, we can let $P_0(X) = X^2 + X + 1$ in the first extension step K_1/K_0 and let x_1 be a root of $P_0(X)$ in K_1. Then $P_1(X) = X^2 + x_1X + 1$ is irreducible over K_1 and we can let it define the extension K_2/K_1. In general, let x_j be a root of $P_{j-1}(X)$ in K_j, then $P_j(X) = X^2 + x_jX + 1$ is irreducible over K_j and we can let it define the extension K_{j+1}/K_j.

If $n = 2^k n'$ *with* n' odd, then $GF(2^n)$ contains an element y_0 which is algebraically equivalent to x_k defined above. Now let the above $P_0(X)$ be replaced by $X^2 + y_0X + 1$, then the statements run the same as above.

When the irreducible polynomials are chosen as above, the operations in K_j can be formulated based on those in K_j-1 as follows. Denote an element $a + bx_j \in K_j$ *as* (a, b), and consider 4 kinds of operations in K_j:

1. Multiplication-by-x_j:

$$(a; b)x_j = (b, a + bx_{j-1})$$

It needs one addition (XOR) plus one multiplication-by-x_{j-1} *in* K_{j-1}. By recursive induction, this finally reduces to $2^j - 1$ additions and one multiplication-by-x_0 in K_0.

2. Squaring:

$(a, b)^2 = ((a + b)^2; b^2x_{j-1})$

It needs one addition (XOR) plus one multiplication-by-x_{j-1} and 2 squaring in K_{j-1}. By recursive induction, this finally reduces to $< j2^j$ additions, $(j+1)j/2$ multiplication-

by-x_0 and 2^j squarings in K_0.

3. Multiplication:

$$(a, b)(c, d) = (ac + bd, ad + bc + bdx_{j-1})$$

It can be done by 3 multiplications $(ac, bd, (a + b)(c + d))$, 5 additions and one multiplication-by-x_{j-1} in K_{j-1}, and finally reduces to 3^j multiplications, $< \sum_{i<j} 6 * 2^{j-i-1} * 3^i = 6(3^j-2^j)$ additions and $< 0{:}5 \times 3^j$ multiplications-by-x_0.

4. Inversion:

$$(a, b)^{-1} = (a^2 + b^2 + abx_{j-1})^{-1}(a + bx_{j-1}, b)$$

It can be done by 3 multiplications, one inversion and one squaring, 2 additions $(a^2 + b^2 + x_{j-1}ab = b^2 + a(a + bx_{j-1}))$, and one multiplication-by-x_{j-1} in K_j.

$_1$; and finally reduces to 1:5 x 3^j multiplications, $\sum_{i<j} 2^j < 2^j$ squarings, $\sum_{0<i<j}(i2^j + 18(3^j - 2^j) + 3 * 2^j) < 9 \times 3^j - (15 - 2j)2^j + 15$ additions, $< j+2^j+3^j$ multiplications-by-x_0, and one inversion in K_0.

Note that if $K_0 = GF(2^n)$ with n odd, then $x_0 = 1$ and all multiplications-by-x_0 above are not needed. It can be seen that an inversion costs only about 1.5 multiplications.

Selecting irreducible polynomials: Case of odd characteristic

Suppose $K_0 = GF(p)$ is a k_0 bit field and k is the security parameter. Let m be the smallest positive integer of the form 3 x 2^{j-1} or 2^j such that $m \times k_0 > k$. If there exists a binomial irreducible polynomial $X^m - w$ over K_0, then the irreducible polynomial in each extension step can be chosen as follows:

For the first step K_1/K_0, let $P_0(X) = X^3 - w$ or $P_0(X) = X^2 - w$; for subsequent steps let $P_i(X) = X^2 - x_i$, where x_i is a solution of the previous P_{i-1} in K_i. The multiplication- by-x_i can be formulated as $(a,b)x_i = (bx_{i-1}, a)$, where $x_0 = w$ and if K_1/K_0 is of degree 3, then $(a,b,c)x_1 = (cw, a, b)$. So it can be finally reduced to a single multiplication-by-w in $GF(p)$. The condition for the existence of such irreducible $X^m - w$ in $GF(p)$ is as follows:

1. If $3 \mid m$ and $j = 2$, then $3 \mid p - 1$.
2. If $3 \mid m$ and $j > 2$, then $12 \mid p - 1$.
3. If $3 \mid m$ and $j >= 2$, then $4 \mid p - 1$.

When the condition is satisfied, w can be chosen as a primitive root of p.

When irreducible X^m_w as above does not exist, the irreducible polynomials can be chosen as follows. If $3 \mid m$, we can let $P_0(X)$ be any irreducible polynomial of degree 3 with simple coefficients. For example, if $3 \mid p - 1$, we can search a $w \sum GF(p)$ with lowest hamming weight such that $P_0(X) = X^3 - w$ is irreducible; otherwise, we can search irreducible polynomials of the form $X^3 - X - w$ where w has lowest hamming weight. Since the subsequent irreducible polynomials are irrelevant to the choice of the first degree 3 extension, we can assume $m = 2^j$ in the following when considering successive quadratic extensions.

If $p \equiv 1 \bmod 4$, we can choose a quadratic non-residue w with lowest hamming weight, and let $P_0(X) = X^2 - w$, and let $P_i(X) = X^2 - x_i$ where x_i is a solution of P_{i-1} similar as above.

If $p \equiv 3 \bmod 4$, we can let $P_0(X) = X^2 + 1$choose an element of the form $x_1 = x_0 + w \in K_1$ such that $P_1(X) = X^2 - x_1$ is irreducible, where x_0 is a root of P_0 and $w \in GF(p)$ has lowest hamming weight. The subsequent P_i can be defined in the same way as above. In this case, a multiplication-by-x_j can be reduced to a multiplication-by-x_1 which is two additions and two multiplications-by-w in $GF(p)$. **Performance**

The performance of an ECC system depends both on the field construction and on the hardware. In a typical application context, a suitable choice of sub-field K_0 followed by a single step field extension, which is known as OEF or "sub-field method", generally offers better performance than the traditional $GF(p)$ and $GF(2^n)$ method. Compared to the "sub-field method", the current invention gives the same efficient multiplication but faster inversion and hence gives additional performance improvement. This improvement is illustrated by the examples in the following section.

Examples

In the following examples, we assume the security parameter is 160 bits.

1. $K_0 = GF(p)$ where $p = 2^{31} - 1$:

The K_0-adder, multiplier can be implemented on 32-bit CPUs using the instructions for integer arithmetic. The K_0-inverter can be implemented using binary extended Euclid division as follows.

Inversion in $GF(2^{31})$

Input: integer $0 < a < p = 2^{31} - 1$.

Output: integer $b = a^{-1} \bmod p$.

Begin: integer $a_0 = p$; $a_1 = a$; $u = 0$; $v = 1$; $k = 0$;

if a_1 is even, do

{$a_1 = p - a_1$; $v = -1$;}

while $a_1 > 1$, do {

$a_0 = a_0 - a_1$; $u = u - v$; $k = k + 1$;

while a_0 is even, do

{$a_0 = a_0/2$; $v = 2v$; $k = k + 1$;}

if $a_0 < a_1$, swap (a_0, a_1), swap (u, v);

}

$b = v \times 2^{31-k} \bmod p$.

End

Define K_1 to be the extension of K_0 with irreducible polynomial $X^3 - 7$, and K_2 over K_1 is defined by $X^2 + 1$. Elements of K_1 are represented by 3-tuples (a_0, a_1, a_2), and Elements of K_2 are represented by 6-tuples (α_0, α_1) = (a_0, a_1, a_2, a_3, a_4, a_5) where the first half and last half can be regarded as elements in K_1. The multiplier and inverter of K_1 are described as follows, where all +; x are in $GF(p)$.

Multiplication in $K_1 = GF(2^{31} - 1)^3$

Input: $\underline{a} = (a_0, a_1, a_2)$; $b = (b_0, b_1, b_2) \in K_1$.

Output: $\underline{c} = (c_0, c_1, c_2) = \underline{ab}$.

Begin

$m_0 = (a_0 + a_1)(b_0 + b_1)$; $m_1 = (a_1 + a_2)(b_1 + b_2)$; $m_2 = (a_0 + a_2)(b_0 + b_2)$;

$m_3 = a_0b_0$; $m_4 = a_1b_1$; $m_5 = a_2b_2$;

$c_0 = m_3 + 7(m_1 - m_4 - m_5)$; $c_1 = m_0 - m_3 - m_4 + 7m_5$; $c_2 = m_2 + m_4 - m_3 - m_5$;

End

Inversion in $K_1 = GF(2^{31} - 1)^3$

Input: $\underline{a} = (a_0, a_1, a_2) \in K_1$.

Output: $\underline{b} = (b_0, b_1, b_2) = \underline{a}^{-1}$.

Begin

$r = a_0^2 - 7a_1a_2$;

if $r = 0${

$s = 7(a_0a_2 - a_1^2)^{-1}$; $b_0 = 0$; $b_1 = sa_0$; $b_2 = -sa_1$;

}

else {

$s = a_1(a_0 - a_2)$; $t = a_2(a_0 - 7a_1)$; $u = (-ra_0 + 7sa_2 + 7ta_1)^{-1}$;

$b_0 = -ur;\ b_1 = us;\ b_2 = ut;$

}

End

The multiplier and inverter of K_2 are formulated in the following.

Multiplication in $K_2 = (GF(2^{31} - 1)^3)^2$

Input: $(\alpha_0, \alpha_1), (\beta_0, \beta_1) \in K_2$.

Output: $(\alpha, \beta) = (\alpha_0, \alpha_1)(\beta_0, \beta_1)$.

Begin

$$\alpha = \alpha_0\beta_0 - \alpha_1\beta_1;$$

$$\beta = (\alpha_0 + \alpha_1)(\beta_0 + \beta_1) - \alpha_0\beta_0 - \alpha_1\beta_1;$$

End

Inversion in $K_2 = (GF(2^{31} - 1)^3)^2$

Input: $(\alpha_0, \alpha_1) \in K_2$.

Output: $(\beta_0, \beta_1) = (\alpha_0, \alpha_1)^{-1}$.

Begin

$$\alpha = (\alpha_0^2 + \alpha_1^2)^{-1};$$

$$\beta_0 = \alpha\alpha_0;\ \beta_1 = \alpha\alpha_1;$$

End

One ECC reported in D. V. Bailey and C.Parr's paper referred to above uses OEF $K_0^{(6)}$ with $K_0 = GF(2^{31}-1)$. The cost ratio of field multiplication/inversion with this method is about 1/5. Compared to this, the above construction gives a ratio about 1/2:5, and hence improves the ECC performance by at least 25%.

2. $K_0 = GF(2^7)$:

The operations in K_0 can be implemented on 8-bit processors as follows. The elements of K_0 are represented by integers in the range [0, 127]. Choose a primitive element g of K_0. Make a powers-table $\exp[i] = g^i\ 0 \le i \le 126$ and make a logarithms-table $\log[a] = \log_g^a\ 1 \le a \le 127$. The multiplication in K_0 can be implemented as

$$ab = \exp[\log[a] + \log[b] \bmod 127]\ a \neq 0;\ b \neq 0:$$

The inversion can be implemented as

$$a^{-1} = exp[127 - log[a]] \; a \geq 2:$$

There are 4 extension steps to get the final 168-bit field K_4. K_1/K_0 has degree 3 and K_i/K_{i-1}; $1 < I \leq 4$ are quadratic. The irreducible polynomials and implementation of the operations in K_i can follow the process described in the previous section.

Compared with the "sub-field method" with the same K_0, this construction improves the multiplication/inversion cost ratio from about 1/8 to 1/1:5, and thus improves the ECC speed by about 2:5 times.

3. $K_0 = GF(2^{31})$:

In this case, the K_0-multiplier and inverter are best suited for hardware implementation. The irreducible polynomials for K_1; K_2 are $X^3 + X + 1$ and $X^2 + X + 1$ respectively. The implementation of operations in K_2 are described in the previous section. Compared with the "sub-field method", this improves the multiplication/inversion cost ratio from about 1/5 to 1/1:5, and thus improves the ECC speed by about 1:8 times.

Therefore, in summary, in a preferred embodiment of the present invention, the construction of the finite field consists of devices to perform operations in a small base field K_0 and methods for successive field extensions. The first extension step K_1/K_0 may have degree 2 or 3 according to size k_0 of K_0 and the security parameter or key k. Subsequent extensions should all be quadratic. For a degree 3 extension K_1/K_0, one multiplication in K_1 needs 6 multiplications in K_0, while one inversion in K_1 needs no more than 12 multiplications and one inversion in K_0. For a quadratic extension K_i/K_{i-1}, one multiplication in K_i needs 3 multiplications in K_{i-1}, and one inversion in K_i needs 3 or 4 (according to the characteristic being even or odd) multiplications in K_{i-1} and one inversion in K_{i-1}. Thus both multiplication and inversion in the final field can be implemented very efficiently via the devices to perform operations in the base field K_0.

On desktop computers, the best choice for K_0 is $GF(p)$ as in OEFs. In this case, the present invention maintains all advantages of OEFs and improves the inversion operation efficiency significantly.

On 8-bit general purpose microprocessors, K_0 may be chosen as $GF(2^7)$, and the multiplication and inversion in this base field can be implemented via table look-up.

For hardware implementation, only operations in K_0 need be circuit integrated, the rest can be implemented via simple programming logic and thus greatly reduce the size and cost of the hardware. In this case, K_0 can be chosen as $GF(2^n)$ where n is selected according to cost-effectiveness requirement of the application context.

The invention may be used in a method for encrypting/decrypting a message for transmission, as indicated in Figure 2.

Variations and additions are possible within the general inventive concept as will be apparent to those skilled in the art.

29 SEP 2005

THE CLAIMS DEFINING THE INVENTION ARE AS FOLLOWS:

1. In an electronic information encryption/decryption system, a method of implementing elliptic curve cryptography including:

performing arithmetic operations over a base field K_o; and

undertaking arithmetic operations in one or more extension fields K_j, based upon the operations in the previous field K_{j-1}.

2. Method of claim 1 wherein K_o is $GF(p)$ where p is a prime number of the form $p = 2^n \pm c$ and where $c < 2^{n/2}$ is a small integer.

3. Method of Claim 1 where K_0 is $GF(2^n)$, the characteristic is 2, the extension degree is 2 and the one or more subsequent extensions and further including the steps of:

selecting irreducible polynomials for each extension step, such that:

if n is odd $P_o(X) = X^2 + X + 1$ is an irreducible polynomial in the first extension step K_1 / K_o ;or

if $n = 2^k n'$ with n' odd $P_o(X) = X^2 + y_o X + 1$ is an irreducible polynomial in the first extension step K_1 / K_o; and

for all subsequent extension steps x_j is a root of $P_{j-1}(X)$ in K_j, so that $P_j(X) = X^2 + x_j X + 1$ is irreducible over K_j and defines the extension K_{j-1} / K_j

4. Method of Claim 3 further including the step of performing a plurality of operations in K_j, on an element $a + bx_j \in K_j$ denoted (a,b), wherein the operations may be from the group comprising:

Multiplication by x_j : $(a, b)x_j = (b, a + bx_{j-1})$;

Squaring: $(a, b)^2 = ((a + b)^2, b^2 x_{j-1})$;

Multiplication: $(a, b)(c, d) = (ac + bd, ad + bc + bdx_{j-1})$; and

Inversion: $(a, b)^{-1} = (a^2 + b^2 + a\, bx_{j-1})^{-1}(a + bx_{j-1}, b)$.

5. Method of Claim 1 or 2 where K_0 is *GF(p)*, the characteristic is odd, the security parameter is k, m is the smallest positive integer of the form $3 \times 2^{j-1}$ or 2^j such that $m \times k_0 > k$ and further including the steps of:

ascertaining whether a binomial irreducible polynomial of the form $X^m - w$ exists, such that $P_0(X) = X^2 - w$ or $P_0(X) = X^3 - w$ and $P_i(X) = X^2 - x_i$ for all subsequent steps, where x_i is a solution of the previous P_{i-1} in K_i and wherein such an irreducible polynomial will exist if one of the following conditions is met:

(a) $3|m$ and $j = 2$, then $3|p - 1$;

(b) $3|m$ and $j > 2$, then $12|p - 1$;

(c) $3|m$ and $j < 2$, then $4|p - 1$.

If a condition is satisfied, and such an irreducible polynomial exists, w is the primitive root of p;

If such an irreducible polynomial does not exists, choosing an irreducible polynomial according to the following criteria:

(d) if $3|m$, then $P_0(X)$ may be any irreducible polynomial of degree 3 with simple coefficients;

(e) if $3|p - 1$, then $P_0(X) = X^3 - w$ or $P_0(X) = X^3 - X - w$ such that $w \in GF(p)$ with lowest hamming weight required for $P_0(X)$ to be irreducible;

(f) if $p = 3 \bmod 4$ and $m = 2^j$, $P_0(X) = X^2 + 1$ and $x_i = x_0 + w \in K$ such that $P_1(X) = X^2 - x_i$ is irreducible, where x_0 is a quadratic non-residue with lowest hamming weight;

(g) if $p = 1 \bmod 4$ and $m = 2^j$, $P_0(X) = X^2 - w$ and $P_1(X) = X^2 - x_i$, where x_i is a solution of P_{i-1} and $w \in GF(p)$ and has lowest hamming weight.

6. Method of claim 8 wherein $n = 7$ and the arithmetic operations are performed via table lookup.

7. Method of claim 8 wherein arithmetic operations in K_o are circuit integrated and all sub-field operations are implemented via programming logic.

8. Method of claim 1 performed on an 8 bit microprocessor.

9. Method of electronically converting an electronic message to an encrypted message for transmission over a transmission medium, said method comprising the steps of:

using an ECC to perform arithmetic operations on a private key and a point, wherein said point is a point on an elliptic curve over a finite field K_o; and

undertaking arithmetic operations in one or more extension fields K_j, based upon the operations in the previous field K_{j-1}, in order to determine an enciphering key;

using an encryption/decryption means to convert said electronic message to said encrypted message using said enciphering key; and using a transmitting means to transmit said encrypted message over said transmission medium.

10. Computer program product including a computer usable medium having computer readable program code and computer readable system code embodied on said medium for implementing elliptic curve cryptography within a data processing system, said computer program product further including computer readable code within said computer usable medium for:

constructing a finite field K_o, such that the size of the field exceeds a security parameter k; and

performing arithmetic operations in K_o and in at least one subsequent extension field K_j, based upon the operations in the previous field K_{j-1}.