An Improved Implementation of Elliptic Curve Digital Signature by Using Sparse Elements

Essam Al-Daoud
Computer Science Department, Zarka Private University, Jordan

Abstract: *This paper introduces several new techniques and algorithms to speed up the elliptic curve digital signature and reduce the size of the transited parameters. The basic idea is to use sparse elements for the curve coefficients and the first base point coordinate. The implementation analysis shows that the addition formula calculations are improved about 40 percent. The sparse elements are introduced with a compact representation, thus the digital signature calculations are speeded up about 40-60 percent, and the public key parameters are reduced about 37-48 percent.*

Keywords: *Elliptic curve cryptography, projective coordinate, sparse elements, elliptic curve digital signature.*

Received July 14, 2003; accepted September 4, 2003

1. Introduction

The main advantage of using the finite group of Elliptic Curve (EC) is that its discrete logarithm problem is believed to be harder than the discrete logarithm problem for the multiplication group of a finite field. There is no known sub-exponential algorithm that can be applied to the elliptic curve discrete logarithm problem. Another advantage that makes elliptic curves more attractive is the possibility of optimizing the arithmetic operations in the underlying field [6]. This has led to appearance of several elliptic curve cryptography products such as security builder, SSL plus, WTLS pus, TrustPoint etc. In addition many companies have purchased licenses to use EC codes and embed them in their products [4, 5, 7].

By using Elliptic Curve Cryptosystem ECC we can use smaller key size with the same level of cryptographic security for DSA or RSA, whereby we will get smaller public key certificates, faster implementation, lower power requirements and smaller hardware processors [15, 8]. Subsequently ECC can be applied to many systems and applications [17, 18].

Elliptic curve cryptography applications and protocols rely on the elliptic curve group operations such as adding, doubling and scalar multiplication, which will not be feasible unless a suitable elliptic curve finite group and efficient underlying finite field operations are used. Thus any enhancement in the underlying finite field operations will speed up all the EC applications [2, 16]. Our approach to enhance the operations is the high utilization of the sparse elements in $GF(2^n)$. Several new algorithms are introduced such as selecting random sparse elements algorithm, finding sparse base points, compressing and decompressing the sparse elements and sparse multiplication over a polynomial basis. The new algorithms improve the EC in three aspects namely: Speed up the curve coefficients multiplications, speed up the multiplication of the first base point coordinate and reduce the size of the transited parameters. This new approach does not reduce the security to the fact that the elliptic curves over $GF(2^n)$ with sparse coefficients are isomorphic to the curves which have coefficients selected randomly. Furthermore, although the base points are restricted to be sparse; the number of sparse base points is still very huge and provides the users with rich choices. The experiment shows that the result of this improvement varies from one protocol to another based on the rate of using the base point, the number of transited bits, the key size and the ratio of doubling to adding.

The remainder of this paper is organized as follows. Section 2 presents the most efficient elliptic curves projective coordinate formulas. In section 3, we introduce the algorithms to select and to compress the sparse elements. Moreover, we discuss the abundance of the sparse points. Section 4 introduces the sparse multiplication over polynomial basis. Finally, in section 5, we show the improvement in elliptic curve digital signature by using the suggested approaches.

2. EC Projective Coordinate Operations

In order to find the sum of two distinct points on the elliptic curve *E over* $(GF(2^n))$ by using affine coordinates, one inverse and one multiplication are needed, but to double a point one inverse and two multiplications are required. Since the implementation of elliptic curve operation indicates that the inverse operation is still more expensive than a field

multiplication, where Hankerson and others show that the cost – ratio of the inversion to the multiplication over polynomial basis is 1-10 [3, 9, 10]. Thus, the projective coordinates X, Y and Z on the curve $y^2 + xy = x^3 + a_2 x^2 + a_6$ over $GF(2^n)$ are used to replace the inverse operation by multiplications such that [1]:

Formula 1: $(X_1, Y_1, 1) + (X_2, Y_2, Z_2) = (X_3, Y_3, Z_3)$, where

$$U = Z_2^2 Y_1 + Y_2$$
$$S = Z_2 X_1 + X_2,\ T = Z_2 S,\ Z_3 = T^2$$
$$V = Z_3 X_1,\ X_3 = U^2 + T(U + S^2 + T a_2)$$
$$Y_3 = (V + X_3)(TU + Z_3) + Z_3^2 C$$

Formula 1 needs 9 field multiplications and 8 temporary variables are required. López and Dahab introduce a new doubling formula which requires 5 field multiplications as follows [11-13]:

Formula 2: $2(X_1, Y_1, Z_1) = (X_2, Y_2, Z_2)$, where

$$Z_3 = Z_1^2 X_1^2$$
$$X_3 = X_1^4 + a_6 Z_1^4$$
$$Y_3 = a_6 Z_1^4 Z_3 + X_3(a_2 Z_3 + Y_1^2 + a_6 Z_1^4)$$

3. Sparse Elements

This section introduces algorithms to select random curves have sparse coefficients and sparse base points. The complexity analysis for the suggested algorithms indicates that the time to generate sparse elements and base points is relatively ignored. Moreover, the reduction of the sparse element length is clarified.

3.1. Select Random Sparse Coefficients

The first step to find a suitable elliptic curve is to select random coefficients (a_2 and a_6) and the selection is repeated until a prospective curve is found. However, there is no security threat if the coefficients are restricted to be sparse in $GF(q)$. Moreover the number of generated curves is still very huge. Algorithm 1 is suggested to generate random curves with sparse coefficients.

Algorithm 1: Generate random sparse coefficients in $GF(q)$
Input: The finite field $GF(2^n)$, s (the maximum number of ones)
Output: The sparse elements a_2 and a_6 in $GF(q)$
1. $j \neg 0$
2. For $i = 1$ to rand (s)
   $v \neg$ rand (n)
   If $x_v = 1$ Then
   $i \neg i - 1$ (x_v is the v^{th} bit in the element x)
   Else $x_v = 1$
3. If $j = 0$ then
   $a_2 \neg x$,
   $j \neg 1$, $x \neg 0$
   and goto step 2
   Else $a_6 \neg x$
4. Return a_2 and a_6.

3.2. The Upper Bound of Sparse Base Points

This subsection shows that even if the base point is restricted to be sparse, the number of generated points is still very huge.

Definition: Let G be a point on $E(GF(2^n))$ represented in the binary expand. Then the point G is sparse if and only if the first coordinate has a few ones so that the number of ones is less than 5 percent from the field size. Moreover the point G is called sparse with s ones if the maximum number of the ones in the first coordinate is s.

Theorem: Let E be any elliptic curve over $GF(2^n)$; then the upper bound of the sparse points with s ones on E is

$$2\sum_{t=1}^{s}\left(\prod_{i=0}^{t-1}(n-i)\right)/t!$$

Proof: Let x be any element in $GF(q)$ and has t ones, then x can be represented in

$$\binom{n}{t} = \frac{n!}{t!(n-t)!} = \left(\prod_{i=0}^{t-1}(n-i)\right)/t!$$

different ways. If the maximum number of the ones in the x coordinate is s, then x can be represented in

$$\sum_{t=1}^{s}\left(\prod_{i=0}^{t-1}(n-i)\right)/t!$$

different ways. Since the quadratic equation has two solutions when x is in $GF(q)$, then the upper bound for the number of sparse points with s ones on E is

$$2\sum_{t=1}^{s}\left(\prod_{i=0}^{t-1}(n-i)\right)/t!$$

□

The order of the selected elliptic curve must be prime or nearly prime (the curve E_j has a nearly prime order if $\#E_j = r_j p_j$ for small integer r_j and large prime number p_j, where j is an integer), then the approximately average number (if the test is run over many curves E_j) of sparse base points with s ones is:

$$\sum_{t=1}^{s}\left(\prod_{i=0}^{t-1}(n-i)\right)/d\,(t!)$$

where d is the average of r_j. Thus, this number is large enough to give the users rich choices, for example if $n = 160$ and $s = 7$ then the number of sparse base points are

nearly 10^{17}. However there is no security threat known in case if many users choose the same base point.

3.3. Selecting a Sparse Base Point

Algorithm 2 is suggested to find a random sparse base point *P*, with s ones for any nearly prime elliptic curve over $GF(2^n)$, where P has a large prime order.

Algorithm 2: Choosing Random sparse base point with s ones.
Input: An elliptic curve E over GF(q), the curve order rk, and the maximum number of ones s.
Output: A sparse base point with s ones.
1. *For i=1 to rand (s)*
2. *v ¬ rand (n)*
3. *If xv= 1 Then*
   i ¬ i -1 (xv is the vth bit in the element x)
   Else xv= 1
4. *End For*
5. *Find the coordinate y by solving the quadratic equation, if y does not exist go to step 1 else set y to one solution of the quadratic equation (randomly).*
6. *G ¬ (x ,y)*
7. *P ¬ k G*
8. *If P= O then go to step 1*
9. *Q ¬ r G*
10. *If Q ¹ O then*
    output "wrong order" and stop
11. *Output G.*

The complexity of the previous algorithm is equal to the complexity of standard algorithm to generate random base points.

3.4. Compact Sparse Elements Representation

To utilize the sparse field elements in the real communication and implementation; Algorithms 3 and 4 are introduced to compress and decompress any sparse element with *s* ones in relatively ignored time.

Algorithm 3: Compression of any sparse element in $GF(2^n)$, where n £ 256
Input: Sparse element x.
Output: Compressed representation array C.
1. *m ¬ 1*
2. *For i=1 to n*
   If x_i ¹ 1 then continue
   C_m= i
   m ¬ m + 1
3. *Return C*

Algorithm 4: Decompression of compact representation
Input: Compressed representation array C with length s.
Output: Sparse element x
1. *For i=1 to s*
   t ¬ C_m
   x_t ¬ 1
2. *Return x*

Since the discrete problem for elliptic curves with field size less than 256 is sufficient for the current applications and at least for next few years; discussion will be restricted to this field size, but it can be extended easily to any other field size. Thus each element in the array C (which forms the compact representation for a sparse element) can be represented in 8 bits, so the size of the array C is *(8s)*. Table 1 shows the reduction rate of the sparse elements.

Table 1. The reduction rate of the sparse elements.

Field Size (n)	s	Conventional	Compact	Reduction Rate
131	6	131	48	63.4
163	7	163	56	65.7
191	5	191	40	79.1
239	5	239	40	83.3

4. Sparse Multiplication Over Polynomial Basis

The field multiplication is the most time consuming operation for adding two elliptic curves points particularly in the projective coordinate. Therefore this section shows the great improvement whenever sparse field elements are used over polynomial basis.

Algorithm 5 [12] is used to find first the polynomial multiplication and then the result is reduced by the irreducible polynomial *f(x)*. The following notations are used; if *a(x)* is element in $GF(2^n)$ then *a(x)* is associated with the binary vector $a= (a_{n-1}, \ldots, a_2, a_1, a_0)$, *w* is the word size , *t* is the number of words that are necessary to represent the vector a, thus the array

$$A= (A[t-1], \ldots, A[2], A[1], A[0])$$

where *A[h]* is the h^{th} word in the vector *a*, if

$$C= (C[m], \ldots, C[1], C[0])$$

then the truncated vector is

$$C\{j\}= (C[m], \ldots, C[j+1], C[j])$$

Algorithm 5: Polynomial multiplications (Right to left).
Input: The polynomials a(x) and b(x) in $GF(2^n)$.
Output: c(x)= a(x) . b(x).
1. *C ¬ 0*
2. *For k= 0 to w-1 do*
   For j= 0 to t –1 do
   If the k^{th} bit of A[j] is 1 then
   add B to C{j}
   If k ¹ w-1 then
   C ¬ C x
3. *Return c*

The previous algorithm needs *w-1* shift operations and nearly n/2 addition operations. Algorithm 6 is suggested to compute the sparse elements multiplication in the polynomial basis representation.

Algorithm 6: Sparse polynomial multiplications.
Input: The array A with length s (the compressed representation of a(x)) and the polynomial b(x) in GF(2^m).
Output: c(x)= a(x) . b(x).
1. *For i=1 to s*
   T= b<< (A_i –1)
   c= c Å T
2. *Return c*

The number of left shift and bit wise additions in the previous algorithm is equal to the number of ones in the sparse elements. To simplify the calculations, assume that the left shift cost is equal to the addition operations. Then Table 2 summarize the improvement by using sparse multiplication algorithm over the general field multiplication in algorithm 5 for word size *w*= 32.

Table 2. Comparison between the multiplications of sparse and random field elements in polynomial basis.

Field Size (n)	Algorithm (5) #Operations	Algorithm (6) s = 6 #Operations	The Cost Ratio
131	97	12	8.03
163	113	12	9.41
191	127	12	10.58
211	137	12	11.41
239	151	12	12.58

if a_2, a_6 and the first coordinate of the base point are sparse, then formula 1 can be implemented with only 5 general field multiplications and 4 sparse elements multiplication. While formula 2 can be implemented with only 3 general field multiplications and 2 sparse elements multiplication.

5. The Improvement in EC Digital Signature

The digital signature is an important algorithm that relies on public key technology, therefore any enhancement in the signature generation or signature verification will have impact on the e-commerce, e-payments and other applications. The curve and base point generation is the first step to implement a digital signature based on elliptic curve cryptography; the second is generating the keys which are used to sign a message and verify the signature. These steps can be described as follows [3, 14]:

EC and Base Point Generation: The first party *A* does the following:

1. Select a suitable elliptic curve *E* defined over *GF(q)*,
2. Select a base point *P*∈ *E(GF(q))* of order *l* , where *l* is the elliptic curve *order*.

EC Key Generation:

1. Select an integer *t* in the interval [1, *l* - 1].
2. Compute the point *Q= t P*.

A's public key is (a_2, a_6, *P, l, Q*) and *A's* private key is *t*.

EC Signature Generation: To sign a message *m, A* does the following:

1. Select an integer *k* in the interval [1, *l* - 1].
2. Compute G= *kP*= (x_1, y_1).
3. Compute the first signature component $r = x_1 \bmod l$. (Here x_1 is the first coordinate in the point G.) If $r = 0$, go to step 1.
4. Compute $k^{-1} \bmod l$ *and e= h*(*m*).
5. Compute $s = k^{-1}\{e + t\, r\} \bmod l$, where *h* is any Secure Hash Algorithm.
6. If *s*= 0, then go to step 1.

The signature for the message *m is* the pair of integers (*r, s*).

EC Signature Verification: To verify *A* signature (*r, s*):

1. Obtain a copy of *A*'s public key (a_2, a_6, *P, l, Q*).
2. Verify that *r* and *s* are integers in the interval [1, *l* - 1].
3. Compute $w = s^{-1} \bmod l$ and *f = h(m)*.
4. Compute $u_1 = f\, w \bmod l$.
5. Compute $u_2 = r\, w \bmod l$.
6. Compute $u_1P + u_2Q = (x_0, y_0)$ and $v = x_0 \bmod l$.
7. Accept the signature if and only if *v= r*.

If a_2, a_6 and the first coordinate of the base point P are sparse, then by using the suggested approach and algorithms; the number of public key bits will be reduced to *(24s + 2n + 2)* from *(5n +2)* for the standard setting. Subsequently reducing the necessary time for sending and receiving the EC digital signatures. Table 3 shows the percentage of the bits reduction in the public digital signature parameters.

Table 3. The reduction rate of the bits by using the new approach.

Field Size (n)	s	PK Standard	PK Compact	Reduction Rate
131	6	657	408	37.8
163	7	817	496	39.2
191	5	957	504	47.3
211	5	1057	544	48.5
239	6	1197	624	47.8

Assume that the curve order *l, t, k,* u_1 and u_2 are equal to the field size *n, d* is the ratio of doubling to adding. If the Formula 1 is used for adding and Formula 2 is used for doubling, then number of multiplications over

$GF(2^n)$ for the signature generation and key generation are given by the equation

Number of Multiplications= 5(The number of doubling)+9*(The number of adding)*

So

$$\#mult=5n+9(n/d)= n\,(5+9/d) \quad (1)$$

and the verification:

$$\#mult= n(10 + 18/d) \quad (2)$$

while the number of multiplications for the signature generation and key generation by using the new approach are reduced to

$$\#mult= n(3\ \text{general field} + 2\ \text{sparse} + 5/d\ \text{general field} + 4/d\ \text{sparse} \quad (3)$$

and the verification process are reduced to

$$\#mult= n(6\ \text{general field} + 4\ \text{sparse} + 13/d\ \text{general field} + 5/d\ \text{sparse}) \quad (4)$$

Thus the improvement in digital signature is
(1-multiplication cost of (1)/ cost of (3)) * 100

and the improvement in the verification is
(1-multiplication cost of (2)/ cost of (4)) * 100

By using the data in Table 2 we can consider that the sparse multiplication is approximately 10 times faster than the general field multiplication, therefore the improvement in the key generation, the signature generation, and the verification process are shown in Table 4.

Table 4. The improvement in the digital signature by using the new approach.

Field Size (n)	Doubling/ Adding *(d)*	Signature Generation Improvement %	Verification Process Improvement %
163	2/1	55.73	40.74
163	3/1	60.00	46.78
191	2/1	55.73	40.74
191	3/1	60.00	46.78
211	2/1	55.73	40.74

6. Conclusion

To satisfy efficient computations and communications, several algorithms are introduced such as selecting random sparse elements algorithm, finding sparse base points, compressing and decompressing the sparse elements, sparse multiplication over polynomial basis. The new approaches and algorithms lead to enhance the digital signature implementation about 40-60 percent, high reduction in the public key parameters by 37-48 percent and did not sacrifice in the elliptic curve cryptography security. Therefore, the elliptic curve application as e-cash and e-commerce can be implemented with better performance using the suggested approach.

References

[1] Al-Daoud E. and Ramlan M., "A New Addition Formula for Elliptic Curves Over $GF(2^n)$," *IEEE Transactions on Computers*, vol. 51, no. 8, pp. 972-975, August 2002.

[2] Al-Daoud E. and Ramlan M., "Elliptic Curve Arithmetic Operations Over $GF(2^n)$ and GF(P) for Cryptosystems Purposes," *in Proceedings of the International Conference on Mathematics and its Applications in the New Millennium*, pp. 381-388, 2000.

[3] Blake I. F., Seroussi G., and Smart N. P., *Elliptic Curve in Cryptography*, University Press, London, Cambridge, 1999.

[4] Certicom, http://www.certicom.com, accessed on March 5, 2003.

[5] Gupta V., Gupta S., and Sheueling C., "Performance Analysis of Elliptic Curve Cryptography for SSL," *in Proceedings of ACM Workshop on Wireless Security*, Atlanta, Georgia, 2002.

[6] Gura N., Eberle H., and Shantz S. H., "Generic Implementations of Elliptic Curve Cryptography Using Partial Reduction," *in Proceedings of 9th ACM Conference on Computers and Communications Security*, Washington DC, 2002.

[7] Gura N., Shantz S. C., Eberle H., Finchelstein D., Gupta S., Gupta V., and Stebila D., "An End-to-End Systems Approach to Elliptic Curve Cryptography," *in Proceedings of CHES'2002 Workshop on Cryptographic Hardware and Embedded Systems*, Lecture Notes in Computer Science, Springer-Verlag, Redwood City, California, 2002.

[8] Ha J. S., Kim Y. H., and Lee K. Y., "Compact Implementation of Elliptic Curve Cryptography System Using a FPGA," *in Proceedings of the 9th Korean Conference on Semiconductors*, pp. 813-814, 2002.

[9] Hankerson D., López J. and Menezes A. J., "Software Implementation of Elliptic Curve Cryptography over Binary Fields," *in Proceedings of CHES'2000*, LNCS 1965, pp. 1-24, 2000.

[10] IEEE P1363 Draft, "Standard Specifications for Public Key Cryptography," http://grouper.ieee.org/groups/1363/, version D13, 1999.

[11] King B., "An Improved Implementation of Elliptic Curves Over GF(2) When Using Projective Point Arithmetic," *in Proceedings of 8th Annual International Workshop on Selected Areas in Cryptography (SAC'2001)*, pp. 134-150, 2001.

[12] López J. and Dahab R., "High-Speed Software Multiplication in $GF(2^m)$," *IC Technical Reports*, IC-00-09, Institute of Computing, University of Campinas, 2000.

[13] López J. and Dahab R., "Improved Algorithms for Elliptic Curve Arithmetic in $GF(2^n)$," *in*

Proceedings of 5th Annual International Workshop on Selected Areas in Cryptography (SAC'98), pp. 201-212, 1998.

[14] Menezes A. J., Oorschot P. C., and Vanstone S. A., *Handbook of Applied Cryptography*, CRC Press, Boca Raton, 1996.

[15] Paar C., "Implementation Options for Finite Field Arithmetic for Elliptic Curve Cryptosystems," *in Proceedings of Invited Presentation at the 3rd Workshop on Elliptic Curve Cryptography (ECC'99)*, University of Waterloo, Waterloo, Canada, pp. 1-3, 1999.

[16] Rosing M., *Implementing Elliptic Curve Cryptography*, Manning, Greenwich, 1999.

[17] Sklavos N. and Koufopavlou O., "Mobile Communications World: Security Implementations Aspects- A State of the Art," *Computer Science Journal of Moldova*, Institute of Mathematics and Computer Science, vol. 31, no. 2, 2003.

[18] Weimerskirch A., Paar C., and Shantz C. S., "Elliptic Curve Cryptography on a Palm OS Device," *in Proceedings of the 6th Australasian Conference on Information Security and Privacy (ACISP'2001)*, LNCS 2119, Macquarie University, Sydney, Australia, pp. 502-514, 2001.

Essam Al-Daoud graduated from Mu'tah University in 1991. He continued his master study at Al Al-Bayt University majoring in numerical analysis. He recieved his PhD from University Putra Malaysia. His research interests are namely sparse matrices and cryptosystems.