An Efficient Algorithm for Exponentiation in DH Key Exchange and DSA in Cubic Extension Fields

Guang Gong, Anwar Hasan, Huapeng Wu and Amr Youssef

August 7, 2002

Abstract

In this report, an efficient algorithm for exponentiation in cubic extensioa fields GF(q^3) for Diffie-Hellman (DH) key exchange and the digital signature algorithm (DSA) is proposed. The complexities of multiplication and squaring in GF(q^3) are also optimized where q is a prime or a power of prime p. Complexity comparison is made between the proposed exponentiation algorithm and the optimized "square-and-multiply" type methods. It is shown that DH key exchange and DSA utilizing the new algorithm is much more efficient than using the "square-and-multiply" type methods. For example, when $q = p^2$, in the calculation of DSA with the new method takes only 57% of the number of ground field operations required by the standard "square-and-multiply" method.

Keywords:

Exponentiation, extension field, LFSR sequence, DH key exchange, DSA.

1 Introduction

Many cryptosystems require extensive computation of exponentiation over finite fields, for instance, Diffie-Hellman (DH) key exchange [2] and the digital signature algorithm (DSA) [8]. Algorithms for exponentiation have been proposed which are most efficient in the fields with a small characteristic, *i.e.,* two [7]. However, little work has been seen in the literature on exponentiation in the fields with large characteristic and low order. On the other hand, Gong and Harn [3] proposed an algorithm to compute the $\pm k$-th terms of a sequence generated with an irreducible cubic polynomial over GF(q), where q is a prime or a power of a prime and GF(q) has a large characteristic. A modified version of their algorithm was later proposed in [4].

In this report, we consider exponentiation in cubic extension fields GF(q^3) where q is a prime or a power of prime p. Such cubic extension fields can be used in public key cryptography (see for example [3]). We propose an algorithm for exponentiation using the dual of a polynomial basis of GF(q^3). The dual basis representation of the elements of GF(q^3) can be obtained from a cubic Linear Feedback Shift Register (LFSR) sequence, for which we use the results of [4]. A cubic LFSR sequence is defined as the output sequence of an LFSR whose characteristic polynomial is an irreducible polynomial of order 3 whose coefficients belong to GF(q) [3].

Since DH key exchange and the DSA can be realized in such cubic field GF(q^3) using the cryptographic system proposed in [3], we show that by using the new exponeniation algorithm the complexities of DH key exchange and the DSA can be significantly reduced compared to these using the "square-and-multiply" methods.

We organize the report as follows. In Section 2, we give the background for efficient arithmetic in GF(p^r), $r \geq 2$, and computation of the $\pm k$-th terms of a cubic LFSR sequence. In Section 3, the multiplication and squaring operations in the cubic field are optimized and the complexity of exponentiation is obtained with the "square-and-multiply" method. A new algorithm for exponentiation in the cubic field is proposed in Section 4. In this section, complexity comparison is also made between the proposed algorithm and the "square-and-multiply" method. The application of the new exponentiation method to DSA is described in Section 5. A few concluding remarks are given in Section 6.

2 Computation of LFSR sequence terms over cubic extension fields

When q is a power of a prime number p, we measure the complexity of an exponentiation in GF(q^3) with the number of arithmetic operations in GF(p). So, in this section, we will first optimize finite field arithmetic in GF(q), where q is a prime or a power of a prime. The second part of this section will provide us with results on the complexity of computing the terms $s_{\pm k}$ of the LFSR sequence with a cubic characteristic polynomial [4]. These results will be used in evaluating the complexity of the new exponentiation algorithm.

2.1 Efficient arithmetic in GF(q), $q = p^r$

First we state a lemma on the finite field arithmetic in quadratic extension of $GF(p)$.

Lemma 1 *[4]*

1. *A multiplication in GF(p^2) can be realized by performing three multiplications in GF(p).*

2. *A squaring in GF(p^2) needs two multiplications in GF(p).*

Clearly, the cost of a squaring in GF(p^2) is about two thirds of that of a multiplication in GF(p^2).

When $q = p^r, r > 2$, a multiplication operation in GF(p^r) can be realized in two steps: Polynomial multiplication and reduction modulo the irreducible polynomial. Polynomial multiplication can be efficiently computed with the Karatsuba-Ofman algorithm (KOA), where $r^{\log_2 3}$ ground field multiplications are needed [5, 10]. The second step of polynomial modular reduction can be performed using an algorithm proposed in [12], where the complexity depends on the choice of the irreducible polynomial in GF(p^r). If there exists an irreducible binomial $x^r + c, c \in$ GF(p), the number of multiplications in GF(p) required for polynomial modular reduction is $r - 1$. The condition for the existence of an irreducible binomial is given in [6]. Noting that for the general case that an irreducible polynomial of degree r with Hamming weight w can be found over GF(p), the modular reduction step would need $(w-1)(r-1)$ multiplications in GF(p) [12]. Also note if r is a composite integer (it has multiple prime factors) and if there does not exist an irreducible binomial of degree r over GF(p), then an *optimal extension field* may be found to reduce the complexity of multiplication in GF(p^r) [1]. In the rest of the report and when complexity comparisons are made, for the purpose of simplification, we will assume that we can choose the field GF(p^r) such that an irreducible binomial exists.

Now we consider a squaring operation A^2 in GF(p^r), where $A = \sum_{i=0}^{r-1} a_i x^i$, $a_i \in$ GF(p). By applying KOA to A^2, we have $A^2 = A_l^2 + x^{\frac{r}{2}}[(A_l + A_h)^2 - A_l^2 - A_h^2] + x^r \cdot A_h^2$, where $A_l = \sum_{i=0}^{\frac{r}{2}-1} a_i x^i$ and $A_h = \sum_{i=0}^{\frac{r}{2}-1} a_{\frac{r}{2}+i} x^i$. Recursively repeating the above process, it follows that $r^{\log_2 3}$ squaring operations in GF(p) are required for polynomial squaring A^2. Then a step of polynomial modular reduction is needed whose complexity has been discussed above.

We summarize the above discussion in the following lemma.

Lemma 2

1. *A multiplication in GF(p^r) can be realized by performing $r^{\log_2 3} + r - 1$ multiplications in GF(p).*

2. *A squaring in GF(p^r) needs $r^{\log_2 3}$ squaring operations in GF(p) and $r - 1$ multiplications in GF(p).*

2.2 Computing LFSR sequence terms over cubic extension fields

As in [3], let $f(x) = x^3 - ax^2 + bx - 1, a, b \in GF(q)$ be an irreducible polynomial over $GF(q)$ and α be a root of $f(x)$ in the extension field $GF(q^3)$. Let $\underline{s} = \{s_i\}$ be a characteristic sequence generated by $f(x)$ [6]. Then the initial state of the sequence $\{s_i\}$ can be given by $s_0 = 3, s_1 = a,$ and $s_2 = a^2 - 2b$ [3].

If k is given in a binary signed-digit (SD) representation, then the k-th term of sequence $\underline{s}$ can be computed with Lemma 3. Further computation efficiency can be achieved by representing k in a so-called *maximal-weight* SD form [4]. Algorithm 1 shows how to compute s_k when k is in this *maximal-weight* signed-digit representation.

Lemma 3 *[4] Let k be given in its SD representation as $k = k_0k_1 \cdots k_r = \sum_{i=0}^{r} k_i 2^{r-i}$, $k_i \in \{-1, 0, 1\}$. Let $T_0 = k_0 = 1$, and $T_j = k_j + 2T_{j-1}, 1 \leq j \leq r$. So, $T_r = k$. If $(s_{T_{j-1}-1}, s_{T_{j-1}}, s_{T_{j-1}+1})$ is given, then $(s_{T_j-1}, s_{T_j}, s_{T_j+1})$ can be computed according to the following formulas.*

For $k_j = 0$

$$s_{T_j-1} = s_{T_{j-1}} s_{T_{j-1}-1} - b s_{-T_{j-1}} + s_{-(T_{j-1}+1)} \quad (1)$$

$$s_{T_j} = s_{T_{j-1}}^2 - 2 s_{-T_{j-1}} \quad (2)$$

$$s_{T_j+1} = s_{T_{j-1}} s_{T_{j-1}+1} - a s_{-T_{j-1}} + s_{-(T_{j-1}-1)} \quad (3)$$

For $k_j = 1$

$$s_{T_j-1} = s^2_{T_{j-1}} - 2s_{-T_{j-1}} \tag{4}$$

$$s_{T_j} = s_{T_{j-1}} s_{T_{j-1}+1} - a s_{-T_{j-1}} + s_{-(T_{j-1}-1)} \tag{5}$$

$$s_{T_j+1} = s^2_{T_{j-1}+1} - 2s_{-(T_{j-1}+1)} \tag{6}$$

For $k_j = -1$

$$s_{T_j-1} = s^2_{T_{j-1}-1} - 2s_{-(T_{j-1}-1)} \tag{7}$$

$$s_{T_j} = s_{T_{j-1}} s_{T_{j-1}-1} - b s_{-T_{j-1}} + s_{-(T_{j-1}+1)} \tag{8}$$

$$s_{T_j+1} = s^2_{T_{j-1}} - 2s_{-T_{j-1}} \tag{9}$$

Algorithm 1 *[4]* Computing a pair of dual terms s_k and s_{-k}

Input: The characteristic polynomial $f(x) = x^3 - ax^2 + bx - 1$ and the initial LFSR sequence terms $s_0 = 3, s_1 = a, s_2 = a^2 - b$ and $s_{-1} = b, a_{-2} = b^2 - a$.

Output: The k-th and $(-k)$-th terms of the LFSR sequence $\{s_i\}$.

1. Set up initial values: $s_{T_0-1} = s_{-T_0+1} = 3$; $s_{T_0} = a$; $s_{T_0+1} = a^2 - 2b$; $s_{-T_0} = b$; $s_{-T_0-1} = b^2 - 2a$;

2. IF $k_r = 0$ THEN find $h < r$, such that $k_h \neq 0$ and $k_{h+1} = k_{h+2} = \cdots = k_r = 0$, ELSE $h = r$;

3. IF $h > 1$ THEN FOR $i = 1$ TO $h - 1$

   (a) IF $k_i = 1$ THEN

      i. compute s_{T_i} and $s_{T_i \pm 1}$ using (4)-(6);

      ii. compute s_{-T_i} and $s_{-T_i \pm 1}$ using (7)-(9);

   (b) ELSE

      i. compute s_{T_i} and $s_{T_i \pm 1}$ using (7)-(9);

      ii. compute s_{-T_i} and $s_{-T_i \pm 1}$ using (4)-(6);

4. FOR $i = \text{Max}\{1, h\}$ TO r

   (a) compute $s_{\pm T_i}$ using (2);

5. The final values of $s_{\pm T_i}$ are $s_{\pm k}$.

Note that T_j and T_{j-1} should be respectively replaced by $-T_j$ and $-T_{j-1}$ in (1)-(9), when these formulas are used in computing s_{-T_i} and $s_{-T_i \pm 1}$ as shown in Steps 3(a)(ii), 3(b)(ii) and 4(a) in the above algorithm.

It can be seen from Algorithm 1 that both Steps 3(a) and 3(b) require two multiplications and two squarings in GF(q). As indicated in [3], during the encryption phase and the signing phase of signature using the GH cryptosystem, either of a and b can be chosen as a constant, such as $0, 1$ and 2, then Step 3(a) or Step 3(b) will require only one multiplication and two squarings in GF(q). Note that the average value of h is $\ll \log_2 k$ and the average numbers of 1's and -1's are the same, which is about $\frac{1}{2}\log_2 k$, we have a slightly extended version of Lemma 9 presented in [4].

Lemma 4 *Let k be given in its maximal-weight SD representation, with $\log_2 k \geq 10$, then, on the average, a dual pair $\{s_{-k}, s_k\}$ can be computed with $4\log_2 k$ multiplications and $4\log_2 k$ squaring operations in GF(q). In particular, computation of $\{s_{-k}, s_k\}$ requires*

(1) $4\log_2 k$ multiplications and $4\log_2 k$ squaring operations in GF(p), if $q = p$ is a prime.

(2) $20\log_2 k$ multiplications in GF(p), if $q = p^2$, where p is a prime.

(3) $4(r^{\log_2 3} + 2r - 2)\log_2 k$ multiplications and $4r^{\log_2 3}\log_2 k$ squaring operations in GF(p), if $q = p^r$, where p is a prime.

Remark: Note that when either a or b can be chosen as certain constant element, *i.e.*, $0, 1$, and 2, the complexities of computing a dual pair $\{s_{-k}, s_k\}$ can be significantly reduced. In this case, a dual pair $\{s_{-k}, s_k\}$ can be computed with $3\log_2 k$ multiplications and $4\log_2 k$ squaring operations in GF(q), on the average. And in particular, computing $\{s_{-k}, s_k\}$ requires

(1) $3\log_2 k$ multiplications and $4\log_2 k$ squaring operations in GF(p), if $q = p$ is a prime.

(2) $17\log_2 k$ multiplications in GF(p), if $q = p^2$, where p is a prime.

(3) at least $(3r^{\log_2 3}+7r-7)\log_2 k$ multiplications and $4r^{\log_2 3}\log_2 k$ squaring operations in GF(p), if $q = p^r$, where p is a prime.

3 Exponentiation in the cubic extension fields

In this section, first we give an optimized evaluation of the complexity of the multiplication and squaring operation in the cubic extension field GF(q^3), where the irreducible polynomial is given by $f(x) = x^3 - ax^2 + bx - 1$. Then we estimate the complexity involved in exponentiation in the cubic fields GF(q^3) using the "square-and-multiply" type methods.

3.1 Multiplication and squaring in the cubic extension fields

Since α be a root of $f(x)$, $\{1, \alpha, \alpha^2\}$ is a *basis* in GF(q^3). Let A and B be two elements in GF(q^3). We can write $A = a_0+a_1\alpha+a_2\alpha^2$ and $B = b_0+b_1\alpha+b_2\alpha^2$, where $a_i, b_i \in$ GF(q). Then following the KOA [5]

$$\begin{aligned} AB &= [(a_0 + a_1\alpha) + a_2\alpha^2][(b_0 + b_1\alpha) + b_2\alpha^2] \\ &= (a_0 + a_1\alpha)(b_0 + b_1\alpha) + \alpha^2[b_2(a_0 + a_1\alpha) + a_2(b_0 + b_1\alpha)] + \alpha^4 a_2b_2 \\ &= (a_0 + a_1\alpha)(b_0 + b_1\alpha) + \alpha^2[(a_0 + a_2 + a_1\alpha)(b_0 + b_2 + b_1\alpha) \\ &\quad -(a_0 + a_1\alpha)(b_0 + b_1\alpha) - a_2b_2] + \alpha^4 a_2b_2. \end{aligned} \tag{10}$$

It can be seen that

$$(a_0 + a_1\alpha)(b_0 + b_1\alpha) = a_0b_0 + \alpha[(a_0 + a_1)(b_0 + b_1) - a_0b_0 - a_1b_1] + \alpha^2 a_1b_1 \tag{11}$$

requires three multiplications in $GF(q)$ and so does

$$\begin{aligned} (a_0 + a_2 + a_1\alpha)(b_0 + b_2 + b_1\alpha) &= (a_0 + a_2)(b_0 + b_2) + \alpha[(a_0 + a_2 + a_1)(b_0 + b_2 + b_1) \\ &\quad -(a_0 + a_2)(b_0 + b_2) - a_1b_1] + \alpha^2 a_1b_1. \end{aligned} \tag{12}$$

Since the term a_1b_1 is common in (11) and (12), the cost of equation (10) is six multiplications in GF(q). Since the product of A and B will be of degree at most 2, the higher degree terms can be removed as follows:

$$\alpha^3 = a\alpha^2 - b\alpha + 1, \;\; \alpha^4 = (a^2 - b)\alpha^2 - (ab - 1)\alpha + a, \tag{13}$$

where $a, b \in$GF(q). This would allow one to verify that the multiplication of any two arbitrary elements A and B of the cubic extension field GF(q^3) requires 12 multiplications and one squaring in GF(q).

Consider the cases that one of a, b can be chosen as zero or a special integer such as 1 and 2. When $a = 0$, it follows from (13)

$$\alpha^3 = -b\alpha + 1, \;\; \alpha^4 = -b\alpha^2 + \alpha.$$

Then one multiplication in GF(q^3) needs only 8 multiplications in GF(q). When $a = 1$ or 2, (13) can be written as

$$\begin{array}{c} \alpha^3 = \alpha^2 - b\alpha + 1 \quad \text{and} \quad \alpha^4 = (1 - b)\alpha^2 - (b - 1)\alpha + 1, \text{ or,} \\ \alpha^3 = 2\alpha^2 - b\alpha + 1 \quad \text{and} \quad \alpha^4 = (4 - b)\alpha^2 - (2b - 1)\alpha + 1. \end{array}$$

Then we can see that one multiplication in GF(q^3) needs 9 multiplications in GF(q). For the cases that $b = 0$ and $b = 1, 2$, the complexity of multiplication in GF(q^3) can also be obtained in a similar manner.

For squaring operation, we have

$$\begin{aligned} A^2 &= [a_0 + a_1\alpha + a_2\alpha^2]^2 \\ &= (a_0 + a_1\alpha)^2 + \alpha^2[(a_0 + a_2 + a_1\alpha)^2 - (a_0 + a_1\alpha)^2 - a_2^2] + \alpha^4 a_2^2. \end{aligned}$$

And

$$(a_0 + a_1\alpha)^2 = a_0^2 + \alpha[(a_0 + a_1)^2 - a_0^2 - a_1^2] + \alpha^2 a_1^2$$

requires three squarings in $GF(q)$ and

$$(a_0 + a_2 + a_1\alpha)^2 = (a_0 + a_2)^2 + \alpha[(a_0 + a_2 + a_1)^2 - (a_0 + a_2)^2 - a_1^2] + \alpha^2 a_1^2$$

also needs three squarings in $GF(q)$, but one squaring (a_1^2) can be reused. Then it follows from (13) that A^2 needs six multiplications and seven squarings in GF(q). Fewer multiplications and squarings in GF(q) are required when either of a and b can be chosen among $0, 1$ and 2.

We summarize the above results in the following lemma.

Lemma 5 *Let $\alpha \in GF(q^3)$ and $f(x) = x^3 - ax^2 + bx - 1$, $a, b \in GF(q)$, be the minimal polynomial of α. Then a multiplication operation in GF(q^3) can be realized with 12 multiplications and one squaring in GF(q), and a squaring operation in GF(q^3) can be performed with six multiplications and seven squarings in GF(q).*

Remark: When either of a and b can be chosen to be certain constant element such as $0, 1$ and 2, then a multiplication in GF(q^3) can be performed with

1. 8 multiplications in GF(q), if $a = 0$;
2. 9 multiplications in GF(q), if $a = 1$ or 2;
3. 9 multiplications and one squaring in GF(q), if $b = 0$;
4. 10 multiplications and one squaring in GF(q), if $b = 1$ or 2.

While a squaring operation in GF(q^3) can be computed with

1. 2 multiplications and 6 squarings in GF(q), if $a = 0$;
2. 3 multiplications in and 6 squarings in GF(q), if $a = 1$ or 2;
3. 3 multiplications and 7 squarings in GF(q), if $b = 0$;
4. 4 multiplications and 7 squarings in GF(q), if $b = 1$ or 2.

3.2 Exponentiation in GF(q^3) using "square-and-multiply" type methods

With the "squaring-and-multiply" algorithm, an exponentiation $y = \alpha^k$ in GF(q^3) can be computed with about $\frac{\log_2 k}{2}$ multiplications and $\log_2 k$ squaring operations in GF(q^3), on average. If k is represented in the NAF, the exponentiation requires about $\frac{\log_2 k}{3}$ multiplications and $\log_2 k$ squaring operations in GF(q^3), along with one inverse operation in GF(q^3) to obtain α^{-1}.

Choice of a, b	Number of modular operations in GF(p), where p is a prime.				
	$q = p$		$q = p^2$	$q = p^r$	
	mul.	squ.	mul.	mul.	squ.
any a, b	$9 \log_2 k$	$4 \log_2 k$	$35 \log_2 k$	$(9r^{\log_2 3} + 13r - 13) \log_2 k$	$4r^{\log_2 3} \log_2 k$
$a = 0$	$5 \log_2 k$	$3 \log_2 k$	$21 \log_2 k$	$(5r^{\log_2 3} + 8r - 8) \log_2 k$	$3r^{\log_2 3} \log_2 k$
$a = 1$ or 2	$6 \log_2 k$	$3 \log_2 k$	$24 \log_2 k$	$(6r^{\log_2 3} + 9r - 9) \log_2 k$	$3r^{\log_2 3} \log_2 k$
$b = 0$	$6 \log_2 k$	$4 \log_2 k$	$26 \log_2 k$	$(6r^{\log_2 3} + 10r - 10) \log_2 k$	$4r^{\log_2 3} \log_2 k$
$b = 1$ or 2	$7 \log_2 k$	$4 \log_2 k$	$29 \log_2 k$	$(7r^{\log_2 3} + 11r - 11) \log_2 k$	$4r^{\log_2 3} \log_2 k$

Table 1: Complexity of α^k in GF(q^3) using the standard "square-and-multiply" method.

Therefore, we can summarize the above results in the following lemma:

Lemma 6 *When the standard "square-and-multiply" method, where the exponent is represented in the binary form, is used, the exponentiation $y = \alpha^k$ in the cubic extension field GF(q^3), where the irreducible polynomial is given by $f(x) = x^3 - ax^2 + bx - 1$, can be computed with $9 \log_2 k$ multiplications and $4 \log_2 k$ squarings in GF(q).*

Remark: If either of a and b can be chosen from the set $\{0, 1, 2\}$, then $y = \alpha^k$ in GF(q^3) requires

(1) $5 \log k$ multiplications and $3 \log k$ squarings in GF(q) if $a = 0$,

(2) $6 \log k$ multiplications and $3 \log k$ squarings in GF(q) if $a = 1$ or 2,

(3) $6 \log k$ multiplications and $4 \log k$ squarings in GF(q) if $b = 0$,

(4) $7 \log k$ multiplications and $4 \log k$ squarings in GF(q) if $b = 1$ or 2.

If we consider that q is a power of prime p, then the complexity can be given in terms of the numbers of operations in the ground field GF(p). The details are given in Table 1.

When the "square-and-multiply" method with the exponent being represented in the NAF is used, the complexity of exponentiation $y = \alpha^e$ in the GF(q^3) is also given in Table 2.

Choice of a, b	Number of modular operations in GF(p), where p is a prime.				
	$q = p$		$q = p^2$	$q = p^r$	
	mul.	squ.	mul.	mul.	squ.
any a, b	$7\log_2 k$	$3.8\log_2 k$	$28.7\log_2 k$	$(7r^{\log_2 3} + 10.8r - 10.8)\log_2 k$	$3.8r^{\log_2 3}\log_2 k$
$a = 0$	$3.7\log_2 k$	$3\log_2 k$	$17\log_2 k$	$(3.7r^{\log_2 3} + 6.7r - 6.7)\log_2 k$	$3r^{\log_2 3}\log_2 k$
$a = 1, 2$	$4.5\log_2 k$	$3\log_2 k$	$19.5\log_2 k$	$(4.5r^{\log_2 3} + 7.5r - 7.5)\log_2 k$	$3r^{\log_2 3}\log_2 k$
$b = 0$	$4.5\log_2 k$	$3.8\log_2 k$	$21.2\log_2 k$	$(4.5r^{\log_2 3} + 8.3r - 8.3)\log_2 k$	$3.8r^{\log_2 3}\log_2 k$
$b = 1, 2$	$5.3\log_2 k$	$3.8\log_2 k$	$23.7\log_2 k$	$(5.3r^{\log_2 3} + 9.2r - 9.2)\log_2 k$	$3.8r^{\log_2 3}\log_2 k$

Table 2: Complexity of α^k using the "square-and-multiply" method with the exponent in the NAF.

4 A New Efficient Exponentiation Algorithm

4.1 Algorithm

Let the cubic extension field GF(q^3) and $f(x)$ be given as in the previous sections. Let k be an integer, $0 < k <$ the order of α. We will propose an efficient algorithm for computing $y = \alpha^k$ in GF(q^3). Assume that $\{\beta_0, \beta_1, \beta_2\}$ is a basis in GF(q^3) and it is dual to the polynomial basis $\{1, \alpha, \alpha^2\}$. Then y can be represented with respect to this dual basis as

$$y = y_0\beta_0 + y_1\beta_1 + y_2\beta_2,$$

where $y_i \in$ GF(q) and $i = 0, 1, 2$. Then it follows

$$y_i = Tr(\alpha^i y) = Tr(\alpha^{i+k}),\ i = 0, 1, 2,$$

where the trace function in the cubic field is given by Tr(γ) $= \gamma + \gamma^q + \gamma^{q^2}$, $\gamma \in$ GF(q^3).

It is obvious that $s_i =$Tr(α^i) is the i-th term of the LFSR sequence with characteristic polynomial $f(x)$. A new algorithm for exponentiation in subgroup $G = \langle\alpha\rangle \subset$GF($q^3$) can be given as follows:

Algorithm 2

Input: *Integer k, $\alpha \in GF(q^3)$, irreducible polynomial $f(x) = x^3 - at^2 + bx - 1$.*
Output: *$y = \alpha^k$ with respect to the dual basis $\{\beta_0, \beta_1, \beta_2\}$.*

1. *Set up initial values $s_{-2} = b^2 - 2a, s_{-1} = b, s_0 = 3, s_1 = a, s_2 = a^2 - 2b$;*

2. *Compute s_k, s_{k+1}, s_{k+2} using Algorithm 1.*

Algorithm	Number of operations in GF(q)	
	multiplication	squaring
S&M(binary)	$9\log_2 k$	$4\log_2 k$
S&M(NAF)	$7\log_2 k$	$3.8\log_2 k$
Algorithm 2	$4\log_2 k$	$4\log_2 k$

Table 3: Complexities of exponentiation α^k in GF(q^3).

3. $y = \alpha^k = (s_k, s_{k+1}, s_{k+2})$, *with respect to the basis* $\{\beta_0, \beta_1, \beta_2\}$.

The complexity of the above algorithm is determined by Step 2, which evaluates the k-th term of the cubic LFSR sequence. From Lemma 4, we have the following results.

Lemma 7 *The exponentiation* $y = \alpha^x$ *in the cubic extension field GF*(q^3) *can be computed with about* $4\log_2 x$ *multiplications and* $4\log_2 x$ *squarings in GF*(q)*. This is equivalent to*

(1) $4\log_2 x$ *modular multiplications and* $4\log_2 x$ *squarings in GF*(p)*, if* $q = p$ *is a prime;*

(2) $20\log_2 x$ *modular multiplications in GF*(p) *if* $q = p^2$ *where* p *is a prime.*

4.2 Complexity analysis for new and square-and-multiply type methods

A comparison of the exponentiation complexity using Algorithm 2 and the "square-and-multiply" algorithm is now given in two tables (namely Tables 3 and 4). Table 3 presents the comparison in terms of the number of operations in GF(q), while Table 4 shows the exponentiation complexity in terms of the number of operations in GF(p), where cases $q = p$, $q = p^2$ and $q = p^r, r > 2$, are considered. Note that the "square-and-multiply" algorithm is denoted as S&M in the tables.

When $q = p$ is a prime, it can be seen from Table 4 that about a half of the number of multiplication and squaring operations in GF(p) is required in the proposed algorithm, compared to the "square and multiply" type methods. In the case that $q = p^2$ and p is a prime, the table also shows that the number of multiplications has been significantly reduced by using the new algorithm and it improves the efficiency by about 43%.

Algorithm	Number of operations in GF(p), where p is a prime.	
	multiplication	squaring
	$q = p$	
S&M(binary)	$9 \log_2 k$	$4 \log_2 k$
S&M(NAF)	$7 \log_2 k$	$3.8 \log_2 k$
Algorithm 2	$4 \log_2 k$	$4 \log_2 k$
	$q = p^2$	
S&M(binary)	$35 \log_2 k$	0
S&M(NAF)	$28.6 \log_2 k$	0
Algorithm 2	$20 \log_2 k$	0
	$q = p^r, r > 2$	
S&M(binary)	$(9r^{\log_2 3} + 13r - 13) \log_2 k$	$4r^{\log_2 3} \log_2 k$
S&M(NAF)	$(7r^{\log_2 3} + 10.8r - 10.8) \log_2 k$	$3.8r^{\log_2 3} \log_2 k$
Algorithm 2	$4(r^{\log_2 3} + 2r - 2) \log_2 k$	$4r^{\log_2 3} \log_2 k$

Table 4: Exponentiation complexity in terms of ground field operations.

4.3 Software implementation of Algorithm 2

We choose q to be a 170 bit prime,

$$q = 1176254344504458005461445829343386029310716668857l9$$

and $f(t) = x^3 - ax^2 + bx - 1$, where

$$\begin{aligned} a &= -3302230324135273625652197182276660750650850048530 39 \\ \text{and } b &= 3943823602955887083176622180100325857547604595 17975. \end{aligned}$$

Using Victor Shoup's Number Theory Library (NTL) version 5.2 [11], the timing results for solving α^k using Algorithm 2 with different values of k are given in Table 5. The machine configuration used for this implementation is IBM 300PL with PII 450 MHZ CPU. It has a RAM size of 256 MB and Win2K OS.

$\lvert k \rvert$ in bits	170	340	510	680	850
Time	0.3324s	0.6729s	0.9934s	1.3379s	1.6754s

Table 5: The average run time for software implementation of Algorithm 2.

5 Application to Digital Signature Algorithm in GF(q^3)

We extend the DSA of [9] for GF(q) into GF(q^3) which is shown below.

Let $q = p^r$, $r \geq 1$. Let x and $y = \alpha^x$ be the private and the public key of the user, respectively. Then the signing and verification processes can be described as follows.

Algorithm 3 *DSA in GF(q^3)*

To sign a message m:

1. Randomly choose $k : \ 0 < k < P$, where P is the order of α and is a prime factor of $q^2 + q + 1$.
2. Compute $r = \alpha^k = (r_0, r_1, r_2)$ using Algorithm 2.
3. Form $R = r_0 + r_1 q + r_2 q^2$.
4. Compute s from $m = xR + ks \bmod P$.

Then the signature of m is (r, s). Note that m is in fact a hashed value.

Verifying the signature:

$$\alpha^m = y^R r^s. \tag{14}$$

where $R = \sum_{i=0}^{2} r_i q^i$ and $r = \sum_{i=0}^{2} r_i \alpha^i$.

It can be seen that to verify the above equation is equivalent to checking if or not the following equation holds [9]:

$$\alpha^{ms^{-1}} = y^{Rs^{-1}} r. \tag{15}$$

Compared with (14), it can be seen that one exponentiation can be saved by using (15).

A comparison of the costs for verifying a signature using the "square-and-multiply" method and the proposed algorithm is shown in Table 6. Note that the minimal polynomial of $y = \alpha^e$ has the form $f(x) = x^3 - a'^2 + b'x - 1$ [3]. Note that in this case a' and b' can not be chosen as a constant, Table 6 shows only the case for arbitrary a and b. For the "square-and-multiply" method

Verifying (15) with	No of modular operations in GF(p)	
	multiplication	squaring
$q = p$ is a prime.		
S&M (binary)	$18 \log_2 P$	$8 \log_2 P$
S&M (NAF)	$14 \log_2 P$	$7.6 \log_2 P$
Algorithm 2	$8 \log_2 P$	$8 \log_2 P$
$q = p^2$, where p is a prime.		
S&M (binary)	$70 \log_2 P$	0
S&M (NAF)	$57.2 \log_2 P$	0
Algorithm 2	$40 \log_2 P$	0
$q = p^r$, where $r > 2$ and p is a prime.		
S&M (binary)	$2(9r^{\log_2 3} + 13r - 13) \log_2 P$	$8r^{\log_2 3} \log_2 P$
S&M (NAF)	$(14r^{\log_2 3} + 21.6r - 21.6) \log_2 P$	$7.6r^{\log_2 3} \log_2 P$
Algorithm 2	$8(r^{\log_2 3} + 2r - 2) \log_2 P$	$8r^{\log_2 3} \log_2 P$

Table 6: Complexity comparison of DSA verifying phase.

with the NAF representation, an inverse operation in GF(q^3) is required for each exponentiation, which is not shown in Table 6. Also note that the order of the cyclic group $\langle \alpha \rangle$ is P.

It can be seen from the table that the cost of verifying $\alpha^{ms^{-1}} = y^{Rs^{-1}} r$ using Algorithm 2 is less than one third of that of the previous methods.

Algorithm 1 can not have further saving for computing $y^{Rs^{-1}}$ if we choose b or a to be zero in $f(x)$. So, direct computation of y^t using the standard "square-and-multiply" algorithm will have the same cost as incurred in Algorithm 1. The reason is that we compute y^t in terms of the LFSR with the minimal polynomial $f(x) = x^3 - ax^2 + bx - 1$. Note a_{-x} is automatically obtained by Algorithm 1 when it computes (a_x, a_{x+1}, a_{x+2}), because Algorithm 1 also outputs $(a_{-x}, a_{-(x+1)}, a_{-(x+2)})$. We may not have one of a_x and a_{-x} being zero.

6 Conclusions

In this report, we have proposed an efficient exponentiation algorithm over cubic extension field which can be used in important cryptosystems, such as, DH exchange system and DSA. It has been shown that the proposed method has a much lower complexity, compared to the "square-and-multiply" type methods. When DSA is defined over the cubic field extensions, the use of the proposed algorithm leads to significant reduction in the computational complexity for signature generation and verification.

References

[1] D. V. Bailey and C. Paar, “Efficient arithmetic in finite field extensions with application in elliptic curve cryptography”, *Journal of Cryptology*, 14(3), pp 153-176, 2001.

[2] W. Diffie and M. E. Hellman, ”New directions in cryptography”, *IEEE Trans. on Infor. Theory* vol. IT-22, no. 6, pp 644-654, Nov.1976.

[3] G. Gong and L. Harn, “Public-key cryptosystems based on cubic finite field extensions”, *IEEE IT* vol 45, no 7, pp 2601-2605, Nov. 1999.

[4] G. Gong, L. Harn and H. Wu, “The GH public-key cryptosystem”, in *the Proceedings of Selected Areas in Cryptography-SAC 2001*, Toronto, ON Canada, August 2001, pp. 301-316. Also to appear in Springer *LNCS* series.

[5] A. Karatsuba and Y. Ofman, “Multiplication of multidigit numbers on automata”, *Sov. Phys.-Dokl. (English translation)*, Vol 7, no. 7, pp 595-596. 1963.

[6] R. Lidl and H. Niederreiter, *Finite Fields*, 2nd edition, Cambridge University Press, 1997.

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

[8] NIST, FIPS PUB 186-2, Digital Signature Standard (DSS), Jan. 2000, http://csrc.nist.gov/publications/fips/fips186-2/fips186-2.pdf

[9] “Standard Specifications For Public Key Cryptography”, *IEEE P1363 Draft*, available at http://grouper.ieee.org/groups/1363/P1363/.

[10] C. Paar, *Efficient VLSI Architectures for Bit-Parallel Computation in Galois Fields*, VDI-Verlag, Düsseldorf, Ph.D Thesis, 1994.

[11] V. Shoup, *Number Theory Library (NTL) version 5.2*, available at http://www.shoup.net/ntl.

[12] H. Wu, "On computation of polynomial modular reduction", *Technical Report CORR 2000-31*, The centre for applied cryptographic research, University of Waterloo, 2000, available at http://www.cacr.uwaterloo.ca/.