The GH Public-key Cryptosystem

Guang Gong, Lein Harn*, and Huapeng Wu**

Department of Electrical and Computer Engineering
University of Waterloo
Waterloo, Ontario N2L 3G1, CANADA

*Department of Computer Networking
University of Missouri-Kansas City
Kansas City, Missouri 64110-2499, USA

**Department of Combinatorics & Optimization
University of Waterloo
Waterloo, Ontario N2L 3G1, CANADA

Abstract. This paper will propose an efficient algorithm that utilizes the signed-digit representation to compute the kth term of a characteristic sequence generated by a linear feedback shift register of order 3 over $GF(q)$. We will also propose an efficient algorithm to compute the $(h - dk)$th term of the characteristic sequence based on the knowledge of the kth term where k is unknown. Incorporating these results, we construct the ElGamal-like digital signature algorithm for the public-key cryptography based on the 3rd-order characteristic sequences which was proposed by Gong and Harn in 1999.
Key words. Public-key cryptosystem, digital signature, third-order linear feedback shift register sequences over finite fields.

1 Introduction

Gong and Harn have published papers on applying third-order linear feedback shift register (LFSR) sequences with some initial states to construct public-key cryptosystems (PKC) in the ChinaCrypt'98 [4] and in the IEEE Transactions on Information Theory [5], respectively. This type of LFSR sequences is called the characteristic sequence in the area of sequence study. The security of the PKC is based on the difficulty of solving the discrete logarithm (DL) in $GF(q^3)$; but all computations involved in the system are still performed in $GF(q)$.

In [5], Gong and Harn have proposed the Diffie-Hellman (DH) key agreement protocol [1] and the RSA-like encryption scheme [14] as examples of the applications of the GH public-key cryptosystem. Along this line, Lenstra and Verheul [7] have published their XTR public-key system at the Crypto'2000. In the XTR public-key system, they have also used the 3rd-order characteristic sequence; but with a special polynomial. They have proposed the XTR DH and the XTR Nyberg-Rueppel signature scheme as examples.

In this paper, we will review some fundamental properties of 3rd-order characteristic sequences and the original GH public-key cryptosystem and point it out that the XTR cryptosystem is constructed based on a special type of 3rd-order characteristic sequences as Gong and Harn have analyzed in [5]. Then, we will

[1] To be presendted at *the Eighth Annual Workshop on Selected Areas in Cryptography*, August 2001, Toronto, Canada.

explore some useful properties of 3rd-order characteristic-sequences over $GF(q)$ and utilize these results in the construction of the GH ElGamal-like digital signature algorithm [2] [13].

The paper is organized as follows. In section 2, we introduce LFSR sequences and 3rd-order characteristic sequences over $GF(q)$. Then, we will review the original GH public-key cryptosystem and explain the relations of design approach between GH and XTR public-key cryptosystems. In Section 3, using the maximal-weight signed-digit representation, we propose a fast algorithm for evaluating the kth term of a pair of reciprocal characteristic sequences over $GF(q)$ and discuss the computational complexity for a special case when $q = p^2$. This algorithm is more efficient than the previously proposed one [5]. In Section 4, we will introduce the Duality Law of a pair of reciprocal characteristic sequences. Using this law, we show the property of redundancy in states of characteristic sequences over $GF(q)$. Lenstra and Verheul in [8] have also found this type of redundancy for a special case of characteristic sequences over $GF(p^2)$. However the technique used in [8] can not be extended to the general case of the characteristic sequences over either $GF(p^2)$ or $GF(q)$ for any arbitrary q. In Section 5, using the linear feedback shift register concept, we will propose an efficient algorithm to compute the $(h - dk)$th term of a characteristic sequence based on the knowledge of the kth term where k is unknown to the user. This mentioned property is required for digital signature verification. This algorithm can save the matrix computation needed in Algorithm 2.4.8 proposed by Lenstra and Verheul in [7]. Then we will apply these results to the design of the GH ElGamal-like digital signature algorithm as an example of digital signature schemes for the GH public-key cryptosystem.

2 Characteristic Sequences and the GH Public-key Cryptosystem

In this section, we will briefly introduce LFSR sequences, characteristic sequences, and the GH Public-key Cryptosystem and explain that the sequences used to construct the XTR cryptosystem is just a special case used in the design of the GH cryptosystem. We will use the notation $K = GF(q)$ where $q = p^r$, p is a prime and r is a positive integer throughout this paper.

2.1 LFSR Sequences

Let

$$f(x) = x^n - c_{n-1}x^{n-1} - \cdots - c_1 x - c_0, c_i \in K$$

be a polynomial and

$$\underline{\mathbf{s}} = \{s_i\} = s_0, s_1, s_2, \cdots, s_i \in K$$

be a sequence over K. If $\underline{\mathbf{s}}$ satisfies the following linear recursive relation

$$s_{k+n} = \sum_{i=0}^{n-1} c_i s_{k+i}, k = 0, 1, \cdots.$$

then we say that $\underline{\mathbf{s}}$ is an LFSR sequence of order n (generated by $f(x)$). $(s_0, s_1, \cdots, s_{n-1})$ is called an *initial state* of the sequence $\underline{\mathbf{s}}$ or $f(x)$. A vector $(s_k, s_{k+1}, \cdots, s_{k+n-1})$ containing consecutive n terms of $\underline{\mathbf{s}}$ is called a state of $\underline{\mathbf{s}}$, or the kth state of $\underline{\mathbf{s}}$, which is denoted by $\underline{\mathbf{s}}_k$.

Example 1. Let $K = GF(5), n = 3$ and $f(x) = x^3 + x - 1$ which is an irreducible polynomial over K. An LFSR sequence generated by $f(x)$ is given below:

```
3  0   3   3 2   0   1   2   4  4
3  0   1   3 4   3   4   1   4  3
2  1   1   1 0   0   1   0   4  1
1  ...
```

which has period $31 = 5^2 + 5 + 1$ and the initial state is $\underline{\mathbf{s}}_0 = (3, 0, 3)$.

2.2 Irreducible Case and Trace Representation

If $f(x)$ is an irreducible polynomial over K, let α be a root of $f(x)$ in the extension $E = GF(q^3)$, then there exists some $\beta \in K$ such that

$$s_i = Tr(\beta\alpha^i), i = 0, 1, 2, \cdots,$$

where $Tr(x) = x + x^q + \cdots + x^{q^{n-1}}$ is the trace function from E to K. If $\beta = 1$, then $\underline{\mathbf{s}}$ is called a *characteristic sequence* of $f(x)$, or a *char-sequence* for short.

The sequence given in Example 1 is the characteristic sequence of $f(x)$.

2.3 Period and Order

Let $f(x) \in K[x]$, we say that $f(x)$ has period t if t is the smallest integer such that $f(x)|x^t - 1$. We denote it as $per(f) = t$.

For $\beta \in E$, the order of β is the smallest integer t such that

$$\beta^t = 1.$$

We denote it as $ord(\beta) = r$. A proof of the following result can be found in several references on sequences, for example, in [9, 10].

Lemma 1. *If $f(x) \in K[x]$ is irreducible over K and $\underline{\mathbf{s}}$ is generated by $f(x)$, then*

$$per(\underline{\mathbf{s}}) = per(f) = ord(\alpha)$$

where α is a root of $f(x)$ in the extension $GF(q^n)$.

2.4 Third-order Characteristic Sequences

Let

$$f(x) = x^3 - ax^2 + bx - 1, a, b \in K \tag{1}$$

be an irreducible polynomial over K and α be a root of $f(x)$ in the extension field $GF(q^3)$. Let $\underline{\mathbf{s}} = \{s_i\}$ be a characteristic sequence generated by f(x). Then the initial state of $\{s_i\}$ can be given by

$$s_0 = 3, s_1 = a, \text{ and } s_2 = a^2 - 2b.$$

Example 1 is a characteristic sequence of order 3.

We list the following lemmas which appeared in [5].

Lemma 2. *With the same notation, we have*

- $per(\underline{\mathbf{s}})|q^2 + q + 1$, *i.e., period of* $\underline{\mathbf{s}}$ *is a factor of* $q^2 + q + 1$.
- $\underline{\mathbf{s}}$ *has the following trace representation:*

$$s_k = Tr(\alpha^k) = \alpha^k + \alpha^{kq} + \alpha^{kq^2}, k = 0, 1, 2, \cdots.$$

If the order of α satisfies an additional condition, then we have the following result whose proof can be found in [7].

Lemma 3. *With the same notation, let* $K = GF(p^2)$, *if* $ord(\alpha)|p^2 - p + 1$, *then*

$$f(x) = x^3 - ax^2 + a^p x - 1, a \in K.$$

2.5 Fundamental Results on 3rd-Order Char-sequences

Here we summarize some results obtained previously. In [5], all results on 3rd-order char-sequences related to the GH Diffie-Hellman key agreement protocol are presented in the finite field $GF(p)$. However, all these results are also true in $K = GF(q)$. So we just list the following two lemmas and the proofs of these lemmas are exactly the same as that described in [5]. Similar results can also be found in corollary 2.3.5 in [7]. For the sake of simplicity, we write $s_k = s_k(a, b)$ or $s_k(f)$ to indicate the generating polynomial. Let $f^{-1}(x) = x^3 - bx^2 + ax - 1$, which is the reciprocal polynomial of $f(x)$. Let $\{s_k(b, a)\}$ be the char-sequence of $f^{-1}(x)$, called *the reciprocal sequence of* $\{s_k(a, b)\}_{k \geq 0}$. Then we have $s_{-k}(a, b) = s_k(b, a), k = 1, 2, \cdots$ (see [5]).

Lemma 4. *Let* $f(x) = x^3 - ax^2 + bx - 1$ *be an irreducible polynomial over* K *and* $\underline{\mathbf{s}}$ *be the char-sequence of* $f(x)$ *and* α *be a root of* $f(x)$ *in* $GF(q^3)$. *Then*

1. *For all integers* r *and* e,

$$s_r(s_e(a, b), s_{-e}(a, b)) = s_{re}(a, b).$$

2. *For all integers* n *and* m,

(a) $s_{2n} = s_n^2 - 2s_{-n}$, *and*

(b) $s_n s_m - s_{n-m}s_{-m} = s_{n+m} - s_{n-2m}$.

3. If $gcd(k, per(\underline{\mathbf{s}})) = 1$, *then* $\alpha^{kq^i}, i = 0, 1, 2$ *are three roots of* $g(x) = x^3 - s_k x^2 + s_{-k}x - 1$ *in* $GF(q^3)$.

Lemma 5. *Let* $k = k_0 k_1 \cdots k_r = \sum_{i=0}^{r} k_i 2^{r-i}$ *be the binary representation of* k. *Let* $T_0 = k_0 = 1$, *and* $T_j = k_j + 2T_{j-1}, 1 \le j \le r$. *So,* $T_r = k$. *If* $(s_{T_{j-1}-1}, s_{T_{j-1}}, s_{T_{j-1}+1})$ *is computed, then* $(s_{T_j-1}, s_{T_j}, s_{T_j+1})$ *can be computed according to the following formulas.*

For $k_j = 0$

$$s_{T_j-1} = s_{T_{j-1}} s_{T_{j-1}-1} - b s_{-T_{j-1}} + s_{-(T_{j-1}+1)} \tag{2}$$

$$s_{T_j} = s_{T_{j-1}}^2 - 2s_{-T_{j-1}} \tag{3}$$

$$s_{T_j+1} = s_{T_{j-1}} s_{T_{j-1}+1} - a s_{-T_{j-1}} + s_{-(T_{j-1}-1)} \tag{4}$$

For $k_j = 1$

$$s_{T_j-1} = s_{T_{j-1}}^2 - 2s_{-T_{j-1}} \tag{5}$$

$$s_{T_j} = s_{T_{j-1}} s_{T_{j-1}+1} - a s_{-T_{j-1}} + s_{-(T_{j-1}-1)} \tag{6}$$

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

Thus, to calculate a pair of k*th terms* s_k *and* s_{-k} *of the sequence* $\underline{\mathbf{s}}$ *needs* $9 log k$ *multiplications in* $GF(q)$ *in average.*

2.6 The GH Diffie-Hellman Key Agreement Protocol

In this subsection, we will review the GH Diffie-Hellman (DH) key agreement protocol. (Note. In [5], the GH-DH was presented in $GF(p)$. As we have mentioned in the beginning of Section 2.5, all results are also true in $GF(q)$, where q is a power of a prime.) In the following discussion, we will present the GH-DH in $GF(q)$ (Note. $q = p^2$ is the same setting as in the XTR cryptosystem.)

GH-DH Key Agreement Protocol (Gong and Harn, 1999) [5] :

System parameters: p is a prime number, $q = p^2$ and $f(x) = x^3 - ax^2 + bx - 1$ which is an irreducible polynomial over $GF(q)$ with period $Q = q^2 + q + 1$.

User Alice chooses $e, 0 < e < Q$, with $gcd(e, Q) = 1$ as her private key and computes (s_e, s_{-e}) as her public key. Similarly, user Bob has $r, 0 < r < Q$, with $gcd(r, Q) = 1$ as his private key and (s_r, s_{-r}) as his public key. In the key distribution phase, Alice uses Bob's public key to form a polynomial:

$$g(x) = x^3 - s_r x^2 + s_{-r}x - 1$$

and then computes the eth terms of a pair of reciprocal char-sequences generated by $g(x)$. I.e., Alice computes

$$s_e(s_r, s_{-r}) \text{ and } s_{-e}(s_r, s_{-r}).$$

Similarly, Bob computes

$$s_r(s_e, s_{-e}) \text{ and } s_{-r}(s_e, s_{-e}).$$

They share the common secret key as (s_{er}, s_{-er}).

Let $\mathbb{Z}_Q = \{0, 1, 2, \cdots, Q-1\}$, $\mathbb{Z}_Q^*$ contain all numbers in $\mathbb{Z}_Q$ and these numbers are coprime with Q, R_Q contains all numbers in $\mathbb{Z}_Q^*$ and these numbers are not conjugate modulo Q respect to q (i.e., any two numbers t and r are conjugate modulo Q if there exists some integer j such that $r \equiv tq^j \mod Q$).

The mathematical function used in the GH public-key cryptosystem is:

$$\begin{array}{rcl} \mu : R_Q & \to & K \times K \\ i & \mapsto & (s_i, s_{-i}) \end{array} \tag{8}$$

where $\underline{\mathbf{s}}$ is the 3rd-order char-sequence over $GF(q)$ generated by $f(x)$ which is an irreducible polynomial with a period of $Q = q^2 + q + 1$. In [5], it is shown that this is an injective map from R_Q to $K \times K$.

Remark 1. The XTR [7] is designed based on the char-sequences generated by the 3rd-order polynomial of $f(x) = x^3 - ax^2 + a^p x - 1$ which is irreducible over $GF(q)$ where $q = p^2$ with period $Q | p^2 - p + 1$. The XTR only uses one char-sequence instead of a pair of reciprocal char-sequences. The mathematical function used in the XTR public-key system is:

$$\begin{array}{rcl} \nu : R_Q & \to & K \\ i & \mapsto & s_i \end{array} \tag{9}$$

However, the GH system is based on the char-sequences generated by the 3rd-order polynomial of $x^3 - ax^2 + bx - 1$, where a and b are from $GF(q)$ ($q = p^r$). Thus, the 3rd-order char-sequence used to construct the XTR cryptosystem is just a special case used in the design of the GH cryptosystem for $q = p^2$. For $q = p^2$, two schemes have the same ratio of the computational cost to the number of bits of the key shared by Alice and Bob when these two schemes are applied to the DH key agreement protocol, because the GH-DH computes a pair of elements over $GF(p^2)$ and shares a pair of elements over $GF(p^2)$, and the XTR-DH computes one element over $GF(p^2)$ and shares one element over $GF(p^2)$.

In the following sections, we will explore some useful properties of 3rd-order char-sequences over K and use these results to the design of a new GH digital signature algorithm. For additional results on LFSR sequences and finite fields, the reader can refer to [3, 9, 10].

3 Fast Computational Algorithm Based on the Signed-Digit Representation

3.1 A New Signed-Digit Number Representation

Definition 1. Let $A = a_{n-1}a_{n-2}\cdots a_0$, $a_i \in \{0, 1\}$ be a binary representation. Then $A = b_{n-1}b_{n-2}\cdots b_0$, $b_i \in \{-1, 0, 1\}$ is called the binary *maximal-weight* signed-digit (SD) representation of A, if there does not exist another binary SD representation of length n for A whose Hamming weight is higher than that of the maximal-weight SD representation.

An algorithm to obtain such an SD representation is given in Appendix A. The following lemma is obvious from Algorithm 4 in Appendix A.

Lemma 6. *Let $a_{n-1}a_{n-2}\cdots a_0$, $a_i \in \{0,1\}$ and $a_{n-1} = 1$ be the binary representation of integer A. Let $d, 0 \le d \le n-2$, be the smallest integer such that $a_d \neq 0$. Then the Hamming weight of the binary maximal-weight SD representation of A is $n-d$. Moreover, all the zeroes are associated with least significant bit positions.*

Some examples are given for the maximal-weight SD representations: $101100111 = 1\ 1\ \overline{1}\ 1\ 1\ \overline{1}\ \overline{1}\ 1\ 1$ and $11001011000 = 1\ 1\ 1\ \overline{1}\ \overline{1}\ 1\ \overline{1}\ 1\ 0\ 0\ 0$. When this SD representation is involved in computing exponentiation-like functions using square-and-multiply method, it is obvious that efficiency can only be achieved when the "squaring and multiplication" is less expensive than the "squaring". It appears to be the case when computing terms in a third-order recurrence sequence as it will be discussed in detail in the next subsections.

3.2 Fast Computational Algorithm of Recurrence Terms

Let k be given in its maximal-weight SD representation as $k = k_0k_1\cdots k_r = \sum_{i=0}^{r} k_i 2^{r-i}$, $k \in \{-1, 0, 1\}$. It can be proven that Lemma 5 still holds true if we add the following formulas.

For $k_j = -1$

$$s_{T_j-1} = s^2_{T_{j-1}-1} - 2s_{-(T_{j-1}-1)} \tag{10}$$

$$s_{T_j} = s_{T_{j-1}}s_{T_{j-1}-1} - bs_{-T_{j-1}} + s_{-(T_{j-1}+1)} \tag{11}$$

$$s_{T_j+1} = s^2_{T_{j-1}} - 2s_{-T_{j-1}} \tag{12}$$

With values of initial terms as $s_0 = 3$, $s_1 = a$, $s_2 = a^2 - 2b$, $s_{-1} = b$, and $s_{-2} = b^2 - 2a$, $T_0 = k_0 = 1$ and $T_j = k_j + 2T_{j-1}, 1 \le j \le r, T_r = k$, a pair of dual terms, s_k and s_{-k} for $k \ge 0$, can be computed based on the following algorithm.

Algorithm 1 Computing a pair of dual terms s_k and s_{-k}

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 (5)-(7);
      ii. compute s_{-T_i} and $s_{-T_i \pm 1}$ using (10)-(12);
   (b) ELSE
      i. compute s_{T_i} and $s_{T_i \pm 1}$ using (10)-(12);
      ii. compute s_{-T_i} and $s_{-T_i \pm 1}$ using (5)-(7);
4. FOR $i = \text{Max}\{1, h\}$ TO r
   (a) compute $s_{\pm T_i}$ using (3);

The final value is $s_{T_r} = s_k$. Note that T_j and T_{j-1} should be respectively replaced by $-T_j$ and $-T_{j-1}$ in (2)-(12), 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.

Since the implementation of (5)-(7) or (10)-(12) is less costly than that of (2)-(4), certain efficiency can be achieved by using the maximal-weight SD representations. It can be shown that an evaluation of (5)-(7) or (10)-(12) needs one multiplication, two squarings, and one constant multiplication in GF(q); while an evaluation of (3) requires one squaring in GF(q). Thus, Step 3 needs two multiplications, four squarings, and two constant multiplications in GF(q); while Step 4 requires two squarings in GF(q). With the assumptions that Step 3(a) or Step 3(b) has to be performed for $h-1$ times and Step 4(a) has to be executed for $r-h+1$ times, and also with the estimation of the average value of h as included in Appendix B, we have the following lemma:

Lemma 7. *Let k be given in its maximal-weight SD representation, with $\log_2 k \geq 10$, then, on the average case (which is also statistically equal to the worst case), to compute a dual pair $\{s_{-k}, s_k\}$ using Algorithm 1 needs $4\log_2 k$ multiplications and $4\log_2 k$ squarings in GF(q). On the best case to compute both s_k and s_{-k} needs $2\log_2 k$ multiplications in GF(q).*

Note that half number of $4\log_2 k$ multiplications are in fact constant multiplications if both a and b are constant.

3.3 Complexity of Computing Recurrence Terms When $q = p^2$

When -1 is a quadratic non-residue in GF(p), the binomial $f(x) = x^2 + 1$ is irreducible over GF(p). Let α be a root of $f(x)$. Under previous assumptions, $\{1, \alpha\}$ forms a polynomial basis in GF(p^2) over GF(p). Any two elements, *xandy* $\in$GF(p^2), can be represented in the polynomial basis as $x = x_0 + x_1\alpha$ and $y = y_0 + y_1\alpha$, $x_0, x_1, y_0, y_1 \in$GF(p). (It is worth to mention that, in XTR system [7], the irreducible trinomial $f(x) = x^2 + x + 1$ and the normal basis have been chosen.) A multiplication in GF(p^2) can be represented by $xy = (x_0 + x_1\alpha)(y_0 + y_1\alpha) = (x_0y_0 - x_1y_1) + (x_0y_1 + x_1y_0)\alpha = [x_0(y_0 + y_1) - y_1(x_0 + x_1)] + [x_0(y_0 + y_1) + y_0(x_1 - x_0)]\alpha$. Thus, three multiplications in GF(p) are needed. Since the squaring can be represented by $x^2 = (x_0 + x_1\alpha)^2 = (x_0^2 - x_1^2) + 2x_0x_1\alpha = (x_0 - x_1)(x_0 + x_1) + 2x_0x_1\alpha$, two multiplications in GF(p) are required for a squaring in GF(p^2). The constant multiplication is the case where the multiplicand is a fixed element. If the constant element can be chosen to be a number with a specific form, then the constant multiplication can be extremely efficient. For example, if both x_0 and x_1 can be chosen to be a small power of two, then it can be seen from $xy = (x_0 + x_1\alpha)(y_0 + y_1\alpha) = (x_0y_0 - x_1y_1) + (x_0y_1 + x_1y_0)\alpha$ that the multiplication of xy can be obtained for free. If we choose the constant element $x = x_0 + x_1\alpha$ such that one of the two coefficients x_0 and x_1 is a small power of 2, then only two multiplications in GF(p) are needed to perform multiplication of xy in GF(p^2). We summarize the above results in the following lemma:

Lemma 8.

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).*

3. *A constant multiplication in GF(p^2)*
   (a) *can be realized for free if both the coefficients of the constant element can be chosen to be a small power of* 2.
   (b) *can be realized by performing two multiplications in GF(p), if one of the two coefficients of the constant element can be chosen as a small power of* 2.

¿From Lemmas 7 and 8, we can find the complexity of computing $s_{\pm k}$ using Algorithm 1 and we summarized this result in the following lemma:

Lemma 9. *Let $q = p^2$ and k be given in its maximal-weight SD representation, with $\log_2 k \geq 10$, then on the average case (which can also be the worst case) to compute a dual pair $\{s_{-k}, s_k\}$ using Algorithm 1 needs:*

1. *at most $20 \log_2 k$ multiplications in GF(p);*
2. *at most $18 \log_2 k$ multiplications in GF(p), if one of the two coefficients of the constant elements $a, b \in GF(p^2)$, can be chosen to be a small power of* 2*;*
3. *at most $16 \log_2 k$ multiplications in GF(p), if one constant element is chosen in such a way that one of the coefficients is a small power of* 2*, and the other constant element is chosen such that both coefficients are small powers of* 2*;*
4. *at most $14 \log_2 k$ multiplications in GF(p), if the both constant elements can be chosen such that all the coefficients are small powers of* 2.

On the best case to compute both s_k and s_{-k} needs $4 \log_2 k$ multiplications in GF(p).

4 Redundancy in States of the 3rd-Order Char-Sequences

In this section, we will introduce the duality law of a pair of reciprocal char-sequences. Under this duality law, we can address some redundancy in states of char-sequence over K.

4.1 Duality Law

Let $f(x) = x^3 - ax^2 + bx - 1$ be an irreducible polynomial over K and $\underline{\mathbf{s}}$ be its char-sequence. We define a dual operator as given below:

$$D(s_k) = s_{-k}$$

$$D(s_k, s_{k+1}, \cdots, s_{k+t}) = (s_{-k}, s_{-(k+1)}, \cdots, s_{-(k+t)}), k \in \mathbb{Z}, t \geq 0,$$

where $T = (s_k, s_{k+1}, \cdots, s_{k+t})$ is a segment of $\underline{\mathbf{s}}$.

We call $(s_k, s_{k+1}, \cdots, s_{k+t})$ and $D(s_k, s_{k+1}, \cdots, s_{k+t})$ a *dual segment* of $\underline{\mathbf{s}}$ or $f(x)$. If $t = 0$, we call s_k and s_{-k} a *dual pair* of $\underline{\mathbf{s}}$ or $f(x)$.

Let $h(x_1, x_2, \cdots, x_t) \in \mathbb{K}[x_1, x_2, \cdots, x_t]$, i.e., h is a multivariables polynomial over $\mathbb{K}$. We define

$$D(h(s_{i_1}, s_{i_2}, \cdots, s_{i_t})) = h(s_{-i_1}, s_{-i_2}, \cdots, s_{-i_t}), i_j \in Z.$$

Duality Law. Let $f(x) = x^3 - ax^2 + bx - 1$ be an irreducible polynomial over K, $\underline{\mathbf{s}}$ be its char-sequence and D be the dual operator. Then $D(D(T)) = T$, $D(D(h)) = h$ and

$$h(s_{i_1}, s_{i_2}, \cdots, s_{i_t}) = 0 \leftrightarrow D(h(s_{i_1}, s_{i_2}, \cdots, s_{i_t})) = 0.$$

4.2 Property of Redundancy

In the following theorem, we will show that three elements in any state of the 3rd-order char-sequence are not independent. If we know any two consecutive elements, the third remaining one can be uniquely determined according to a formula.

Theorem 1. *Let $f(x) = x^3 - ax^2 + bx - 1$ be an irreducible polynomial over K and $\underline{\mathbf{s}}$ be its char-sequence. For given the dual segment (s_k, s_{k+1}) and $(s_{-k}, s_{-(k+1)})$, we assume that $\Delta = s_{k+1}s_{-(k+1)} - s_1 s_{-1} \neq 0$. Then s_{k-1} and its dual $s_{-(k-1)}$ can be computed by the following formulas:*

$$s_{k-1} = \frac{e s_{-(k+1)} - s_{-1} D(e)}{\Delta} \tag{13}$$

$$s_{-(k-1)} = \frac{D(e) s_{(k+1)} - s_1 e}{\Delta} \tag{14}$$

where

$$e = -s_{-1} D(c_1) + c_2, \text{ where } c_1 = s_1 s_{k+1} - s_{-1} s_k \text{ and } c_2 = s_k^2 - 3s_{-k} + (b^2 - a) s_{-(k+1)}.$$

(Note. Here $s_1 = a$ and $s_{-1} = b$. In order to keep symmetric forms in the formulas, we keep on using s_1 and s_{-1}.)

Proof. A sketch to prove this theorem is given below. From $U = (s_{k-1}, s_k, s_{k+1}, s_{k+2})$ and its dual, we will form four linear equations in terms of four variables $s_{k+2}, s_{-(k+2)}, s_{k-1}, s_{-(k-1)}$. Then based on linear algebra, we can solve these equations and obtain (13) and (14).

We now start to construct the linear equations. Since U is a segment of $\underline{\mathbf{s}}$ generated by $f(x)$, it satisfies the linear recurrent relation,

$$s_{k+2} = s_1 s_{k+1} - s_{-1} s_k + s_{k-1}.$$

Thus, we have

$$s_{k+2} - s_{k-1} = c_1 \text{ where } c_1 = s_1 s_{k+1} - s_{-1} s_k. \tag{15}$$

Applying Duality Law to the above expression, we have

$$s_{-(k+2)} - s_{-(k-1)} = D(c_1). \tag{16}$$

Let $n = k - 1$, $m = k + 1$ in formula $2(b)$ in Lemma 4, we get

$$s_{k-1}s_{k+1} - s_{-2}s_{-(k+1)} = s_{2k} - s_{-(k+3)}. \tag{17}$$

Since $(s_{-}k, s_{-(k+1)}, s_{-(k+2)}, s_{-(k+3)})$ is a dual of U, it satisfies the following linear recursive relation

$$s_{-(k+3)} = s_{-1} s_{-(k+2)} - s_1 s_{-(k+1)} + s_{-k}.$$

Note that $s_{2k} = s_k^2 - 2s_{-k}$ from $2(a)$ in Lemma 4. Substituting $s_{-(k+3)}$ and s_{2k} in (17) respectively with the above two identities, it follows that

$$s_{k-1}s_{k+1} - s_{-2}s_{-(k+1)} = s_k^2 - 2s_{-k} - s_{-1}s_{-(k+2)} + s_1 s_{-(k+1)} - s_{-k}.$$

Since $s_{-2} + s_1 = b^2 - 2a + a = b^2 - a$, we have

$$s_{-1}s_{-(k+2)} + s_{k+1}s_{k-1} = c_2 \text{ where } c_2 = s_k^2 - 3s_{-k} + (b^2 - a)s_{-(k+1)}. \tag{18}$$

By Duality Law, we have

$$s_1 s_{(k+2)} + s_{-(k+1)}s_{-(k-1)} = D(c_2). \tag{19}$$

The equations (15), (16), (18), and (19) form four linear equations in terms of variables $s_{k+2}, s_{-(k+2)}, s_{k-1}, s_{-(k-1)}$. Let A be the matrix of the coefficients of this linear system, i.e.,

$$A = \begin{pmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & s_{-1} & s_{k+1} & 0 \\ s_1 & 0 & 0 & s_{-(k+1)} \end{pmatrix}$$

Therefore, this linear system can be written as

$$AS^T = C^T, \tag{20}$$

where $S = (s_{k+2}, s_{-(k+2)}, s_{k-1}, s_{-(k-1)})$, $C = (c_1, D(c_1), c_2, D(c_2))$ and X^T is the transpose of the vector X. Let $\tilde{A} = (A, C^T)$. Then the reduced row-echelon form of $\tilde{A}$ is given below:

$$\tilde{A} \sim \begin{pmatrix} 1 & 0 & -1 & 0 & c_1 \\ 0 & 1 & 0 & -1 & D(c_1) \\ 0 & 0 & s_{k+1} & s_{-1} & e \\ 0 & 0 & s_1 & s_{-(k+1)} & D(e) \end{pmatrix}$$

where $e = -s_{-1}D(c_1) + c_2$. Thus (20) has a unique solution if and only if $det(B) \neq 0$, where

$$B = \begin{pmatrix} s_{k+1} & s_{-1} \\ s_1 & s_{-(k+1)} \end{pmatrix}.$$

Since $det(B) = \Delta \neq 0$, then s_{k-1} is given by

$$s_{k-1} = \frac{det \begin{pmatrix} e & s_{-1} \\ D(e) & s_{-(k+1)} \end{pmatrix}}{\Delta},$$

which yields (13). The validity of (14) follows from the Duality Law.

Corollary 1. *With the same notation as used in Theorem 1, the dual pair s_{k+2} and $s_{-(k+2)}$ are given by*

$$s_{k+2} = s_{k-1} + c_1 \text{ and } s_{-(k+2)} = s_{-(k-1)} + D(c_1).$$

Remark 2. If $\Delta \neq 0$, then three elements in a state (s_{k-1}, s_k, s_{k+1}) and their duals are dependent. With the knowledge of any two consecutive elements and their duals, the third one and its dual can be uniquely determined by Theorem 1. If $\Delta = 0$, then the third element in a state of $\underline{\mathbf{s}}$ may have more than one solution. For the case of knowing (s_{k-1}, s_k) and its dual to compute s_{k+1} and its dual, it is similar to the previous case that we have discussed. We will not include the discussion here.

Remark 3. In [8], Lenstra et. al. have also given a formula to compute s_{k-1} (or s_{k+1}) with the knowledge of (s_k, s_{k+1}) (or (s_{k-1}, s_k)) for a special case of $K = GF(p^2)$ and $f(x) = x^3 - ax^2 + a^p x - 1$. Here, we have discussed more general cases and proposed a simpler proof. The formulas between these two approaches are different. The technique used in [8] can not be extended to the general case of the char-sequences.

5 The GH Digital Signature Algorithm

In this section, we explain the method to evaluate $s_{c(h-dk)}$ and its dual with the knowledge of s_k and its dual; but without knowing k. Then, we apply this result together with Theorem 1 in Section 4 and Algorithm 1 in Section 3 to the design of GH ElGamal-like digital signature algorithm (GH-DSA).

5.1 Computation of A Mixed Term $S_{c(h-dk)}$

The following lemma is a direct result from the definition of LFSR sequences.

Lemma 10. *With the same notation of $f(x)$, $\underline{\mathbf{s}}$, let (s_{k-1}, s_k, s_{k+1}) be a state of $\underline{\mathbf{s}}$ and $\underline{\mathbf{u}}$ be a sequence generated by $f(x)$ with (s_{k-1}, s_k, s_{k+1}) as an initial state. I.e.,*

$$u_0 = s_{k-1}, u_1 = s_k, \text{ and } u_2 = s_{k+1}.$$

Then, $\underline{\mathbf{u}}_{v-1} = (u_{v-1}, u_v, u_{v+1})$, the $(v-1)$th state of $\underline{\mathbf{u}}$, is equal to the $(v-1+k)$th state of $\underline{\mathbf{s}}$. In other words, we have

$$(u_{v-1}, u_v, u_{v+1}) = (s_{v-1+k}, s_{v+k}, s_{v+1+k}).$$

For simplicity, we denote $((s_{k-1}, s_k, s_{k+1}), f(x))$ as a sequence generated by $f(x)$ with an initial state (s_{k-1}, s_k, s_{k+1}).

Algorithm 2 *Assume that $f(x)$,(s_k, s_{k+1}) and its dual are given. Let $Q = per(\underline{\mathbf{s}})$. Assume that c, h and d are given integers with $gcd(d, Q) = 1$. Then $s_{c(h-dk)}$ and its dual can be computed according to the following procedures:*

1. *Compute $v = -hd^{-1} \bmod Q$ and $u = -cd \bmod Q$.*
2. *Compute the s_{k-1} and its dual according to Theorem 1.*
3. *Compute $(v-1)$th state of a sequence generated by $((s_{k-1}, s_k, s_{k+1}), f(x))$ according to Algorithm 1. This step gives s_{v+k} and its dual.*
4. *Construct $g(x) = x^3 - s_{v+k}x^2 + s_{-(v+k)}x - 1$ and compute $s_u(g), s_{-u}(g)$ according to Algorithm 1.*

Here, we have $s_u(g) = s_{c(h-dk)}$ and $s_{-u}(g) = s_{-(c(h-dk))}$.

Note. All results that we have discussed so far are true for general q and Q.

Lemma 11. *With the same notation as used in Algorithm 2, to compute $s_{c(h-dk)}$ and its dual needs $2 \cdot 4(logv + logu)$ multiplications and $2 \cdot 4(logv + logu)$ squarings in $GF(q)$ in average. In particular, if $q = p^2$, to compute $s_{c(h-dk)}$ and its dual needs $2 \cdot 20(logv + logu)$ multiplications in $GF(p)$ in average.*

Proof. In Algorithm 2, the computational cost depends only on how many times Algorithm 1 is invoked. Since Algorithm 2 invoked Algorithm 1 twice, applying Lemma 7, it needs 16 multiplications in $GF(q)$. According to Lemma 9, for invoking Algorithm 1 each time, it needs $20(logv + logu)$ multiplications in $GF(p)$. In total, it needs $2 \cdot 20(logv + logu)$ multiplications in $GF(p)$.

Remark 4. When we apply Algorithm 2 to the char-sequences used in the XTR, it can save the matrix computation as given in Algorithm 2.4.8 [7]. So, this algorithm is more efficient than the algorithm given in [7].

5.2 The GH Digital Signature Algorithm

We are now ready to present the GH ElGamal-like digital signature algorithm. Note that the GH signature scheme can also be modified into variants of generalized ElGamal-like signature schemes as listed in [6].

Algorithm 3 *(**GH-DSA**)*

System public parameters: *p is a prime, $q = p^2$, and $f(x) = x^3 - ax^2 + bx - 1$ which is an irreducible polynomial over $GF(q)$ with period Q, where Q satisfies the condition that $Q = P_1P_2$, P_1 is a prime divisor of $p^2 + p + 1$ and P_2 is a prime divisor of $p^2 - p + 1$. Let $p \equiv 3 \pmod 4$ and $GF(p^2)$ be defined by the irreducible polynomial of $x^2 + 1$ (see Subsection 3.3) and γ be its root in $GF(p^2)$.*

Alice: *Choose x, with $0 < x < Q$ and $gcd(x, Q) = 1$ as her private key and compute (s_x, s_{-x}) as her public key. For a message m that Alice needs to sign, she follows the procedures:*

1. *Randomly choose k, with $0 < k < Q$, and $gcd(k, Q) = 1$, and use Algorithm 1 to compute (s_{k-1}, s_k, s_{k+1}) and its dual such that $r = s_{k,0} + s_{k,1}p$ is coprime with Q, where $s_k = s_{k,0} + s_{k,1}\gamma$. (Here, we adopt the similar approach as used in the Elliptic Curve digital signature algorithm [12] to form an integer r in digital signing process.)*
2. *Compute $h = h(m)$, where h is a hash function.*
3. *Compute $t = k^{-1}(h - xr) \bmod Q$ (i.e., the signing equation is: $h \equiv xr + kt \bmod Q$.)*

Then (r, t) is a digital signature of the message m. Alice sends Bob (m, r, t) together with (s_k, s_{k+1}) and its dual.

Bob: *Performing the following verifying process*

Check if $gcd(t, Q) = 1$.

Case 1. *$gcd(t, Q) = 1$.*

1. *Compute $v = tr^{(-1)} \bmod Q$ and $u = hr^{(-1)} \bmod Q$.*
2. *Compute s_{u-vk} and its dual according to Algorithm 2.*
3. *Check if both $s_{u-vk} = s_x$ and $s_{-(u-vk)} = s_{-x}$. If so, Bob accepts it as a valid signature. Otherwise, Bob rejects it.*

Case 2. *$gcd(t, Q) > 1$.*

1. *Compute s_{h-rx} and its dual according to Algorithm 2.*
2. *Form $g(x) = x^3 - s_k x^2 + s_{-k} x - 1$ and compute $s_t(g)$ and its dual according to Algorithm 1.*
3. *Check if both $s_{h-rx} = s_t(g)$ and $s_{-(h-rx)} = s_{-t}(g)$. If so, Bob accepts it as a valid signature. Otherwise, Bob rejects it.*

Lemma 12. *The security of the GH-DSA is based on the difficulty of solving the discrete logarithm in $GF(q^3) = GF(p^6)$. The signing and verifying processes need respectively $20logQ$ multiplications and $2 \cdot 20logQ$ multiplications in $GF(p)$ in average.*

Proof. Since $f(x)$ is an irreducible polynomial over $GF(q^3)$ and the period of $f(x)$ is $Q = P_1 P_2$, where $P_1 | p^2 + p + 1$ and $P_2 | p^2 - p + 1$, a root of $f(x)$ is in $GF(p^6) - (GF(p^3) \cup GF(p^2))$. Similarly, as we have proved in [5], the problem of solving for x from (s_x, s_{-x}) or solving for k from (s_k, s_{-k}) is equivalent to compute DL in $GF(p^6)$. Thus, the first assertion is established.

Note that the probability of any number less than Q which is not coprime with Q is given by

$$Prob\{gcd(z, Q) > 1 : 0 < z < Q\} = \frac{P_1 + P_2 - 1}{P_1 P_2}. \tag{21}$$

Thus, in the signing process, we only need to estimate the computational cost for invoking Algorithm 1 at Step 1, which is $20logQ$ multiplications in $GF(p)$ in average. For case 1 in the verifying process, it can be seen that from Lemma 9 invoking Algorithm 2 at Step 2 needs $2 \cdot 20logQ$ multiplications in $GF(p)$ in average. In Case 3, it invokes Algorithm 2 at Step 2 and Algorithm 1 at Step 3. Thus, it needs $3 \cdot 20logQ$ multiplications in $GF(p)$ in average. Combined with (21), the verifying process needs $2 \cdot 20logQ$ multiplications in $GF(p)$ in average.

6 Conclusion

In this paper, we discuss an efficient algorithm that utilizes the signed-digit representation to compute the k term of a characteristic sequence generated by a linear feedback shift register of order 3 over $GF(q)$. Then we propose an efficient algorithm to compute the $(h - dk)$th term of the characteristic sequence based on the knowledge of the kth term where k is unknown. By using these new results on the characteristic sequences, the GH-DSA (Digital Signature Algorithm) is developed.

Remark 5. The GH cryptosystem, just like the elliptic curve public-key cryptosystem, enjoys the benefit of using a shorter key to achieve high security. Also, the GH cryptosystem can be resistant to power analysis attack and timer analysis attack without increasing cost of computation. This is due to their evaluation formulas as given in Lemma 5 of Section 2.

References

1. 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.

2. T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Trans. on Inform. Theory*, vol. IT- 31, no. 4, July 1985, pp.469-472.
3. S. Golomb, *Shift Register Sequences,* Holden-Day, Inc., 1967. Revised edition, Aegean Park Press, 1982.
4. G. Gong and L. Harn, "A new approach on public-key distribution", *ChinaCRYPT '98*, pp 50-55, May, 1998, China.
5. G. Gong and L. Harn, "Public-key cryptosystems based on cubic finitie field extensions", *IEEE IT* vol 45, no 7, pp 2601-2605, Nov. 1999.
6. L. Harn and Y. Xu, "On the design of generalized ElGamal type digital signature schemes based on the discrete logarithm", *Electronics Letters* vol. 30, no. 24, pp 2025-2026, Nov. 1994.
7. A. K. Lenstra and E. R. Verheul, "The XTR public key system", *Advances in Cryptology, Proceedings of Crypto'2000*, pp. 1-19, Lecture Notes in Computer Science, Vol. 1880, Springer-Verlag, 2000.
8. A. K. Lenstra and E. R. Verheul, "Key improvements to XTR", *the Proceedings of Asiacrypt'2000*, Lecture Notes in Computer Science, vol. 1976, 2000.
9. R. Lidl and H. Niederreiter, *Finite Fields*, Addison-Wesley Publishing Company, Reading, MA, 1983.
10. R.J. McEliece, *Finite Fields for Computer Scientists and Engineers*, Kluwer Academic Publishers, Boston, 1987.
11. A. J. Menezes, *Elliptic Curve Public Key Cryptosystems*, Kluwer Academic Publishers, Boston, 1993.
12. A. J. Menezes and P. C. van Oorschot and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1996.
13. NIST, FIPS PUB 186-2, Digital Signature Standard (DSS), Jan. 2000, http://csrc.nist.gov/publications/fips/fips186-2/fips186-2.pdf
14. R. Rivest, A. Shamir and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems", in*Comm. Of the ACM*, vol. 21, no. 2, pp 120-126, 1978.

APPENDIXES

A An algorithm to obtain maximal-weight SD representation

When the binary representation of an integer is given, its binary maximal-weight SD representation can be generated with the following algorithm.

Algorithm 4 Maximal-weight signed-digit recoding
Input: the binary representation of A: $a_{n-1}a_{n-2}\cdots a_0$, $a_i \in \{0,1\}$ and $a_{n-1}=1$;
Output: the binary maximal-weight representation of A: $b_{n-1}b_{n-2}\cdots b_0$, $b_i \in \{-1,0,1\}$;

1. initialize the flag: $t=0$;
2. FOR $i=0$ TO $n-2$
   (a) IF $t=0$ THEN
      i. IF $a_i=0$ THEN $b_i=0$;
      ii. ELSE $\{t=1$; IF $a_{i+1}=0$ THEN $b_i=-1$; ELSE $b_i=1;\}$
   (b) ELSE
      i. IF $(a_i=1$ AND $a_{i+1}=0)$ THEN $b_i=-1$;
      ii. IF $(a_i=1$ AND $a_{i+1}=1)$ THEN $b_i=1$;
      iii. IF $(a_i=0$ AND $a_{i+1}=0)$ THEN $b_i=-1$;
      iv. IF $(a_i=0$ AND $a_{i+1}=1)$ THEN $b_i=1$;
3. $b_{n-1}=a_{n-1}$;

The correctness of this algorithm can be proved. It is worth to point out that the maximal-weight SD representation always has the same length as the binary form. If the maximal-weight SD representation of a negative integer $(-A)$ is required, it can be obtained by negating each bit in the maximal-weight SD representation of A. It can be seen that the Hamming weight of the maximal-weight SD representation of $-A$ is the same as that of A.

B An estimation of the parameter h

In Algorithm 1, Let $k = k_0 k_1 \cdots k_r$, $2^r \leq k \leq 2^{r+1} - 1$, be given in its maximal-weight SD representation, then from Lemma 6, we have $\Pr\{k_i = 0, 0 < i < r\} = \frac{\frac{r(r-1)}{2}}{(r-1) \times 2^r} = \frac{r}{2^{r+1}}$. Thus, the average value of h can be given by $\overline{h} = r - (r-1)\frac{r}{2^{r+1}}$. The following table shows some values of $\overline{h}(r)$ as a function of r.

r	2	3	4	5	6	7	8	9	10	12	$r > 15$
$\overline{h}(r)$	1.75	2.62	3.62	4.69	5.77	6.84	7.89	8.93	9.96	11.98	r
$\text{Max}\{h(r)\}$	2	3	4	5	6	7	8	9	10	12	r
$\text{Min}\{h(r)\}$	0	0	0	0	0	0	0	0	0	0	0

Table 1. Some values of $\overline{h}(r)$, $\text{Max}\{h(r)\}$, $\text{Min}\{h(r)\}$ and r.

Although the value of h can be as small as 0, it can be seen that the average value $\overline{h}(r)$ is approximately equal to its maximal value r when $r \geq 10$.