New LFSR-Based Cryptosystems and the Trace Discrete Log Problem (Trace-DLP)

Kenneth J. Giuliani[1] and Guang Gong[2]

[1] Dept. of Combinatorics and Optimization
University of Waterloo
Waterloo, ON, Canada, N2L 3G1
kjgiulia@cacr.math.uwaterloo.ca

[2] Dept. of Electrical and Computer Engineering
University of Waterloo
Waterloo, ON, Canada, N2L 3G1
ggong@calliope.uwaterloo.ca

Abstract. In order to reduce key sizes and bandwidth, cryptographic systems have been proposed using minimal polynomials to represent finite field elements. These systems are essentially equivalent to systems based on characteristic sequences generated by a linear feedback shift register (LFSR). We propose a general class of LFSR-based key agreement and signature schemes based on n-th order characteristic sequences. These schemes have the advantage that they do not require as much bandwidth as their counterparts based on finite fields. In particular, we present a signature scheme based on a new computational problem, the *Trace Discrete Logarithm Problem (Trace-DLP)*. The Trace-DLP and its variants are discussed and their relationship with well-known finite field-based computational problems is examined. In addition, we prove the equivalence between several LFSR-based computational problems and their finite field-based counterparts.

1 Introduction

A good portion of public-key cryptography is based upon finite fields. Some of the most notable examples are Diffie-Hellman key agreement [1] and the Digital Signature Standard [13]. However, since field sizes must be chosen large enough to avoid the so-called "index-calculus" attacks, finite field elements normally require a large amount of bits in order to represent them. For applications where bandwidth is limited, this is undesirable.

As a result, several cryptosystems have been proposed which reduce the representation of finite field elements. Examples of such systems are LUC [9, 16], GH [6], and XTR [7]. These systems reduce representations of finite field elements by representing them with the coefficients of their minimal polynomials. Due to the Newton Identity, these methods are essentially the same as systems based on n-th order characteristic sequences generated by linear feedback shift

registers (LFSR's). In particular, LUC can be considered as a second-order sequence, while GH and XTR can be considered as third-order sequences. We also note that fifth-order sequences have also been proposed [14, 3, 4] and that Niederreiter [10–12] has proposed encryption and key agreement schemes based on general n-th order LFSR sequences.

This paper proposes schemes based on n-th order *characteristic* sequences generated by an LFSR. In particular, we propose general key agreement and signature schemes where the sizes of signatures and keys are directly related to the sizes of the representations of elements. In addition, we present new computational problems, namely the *Trace-Discrete Logarithm Problem (Trace-DLP)* and its variants, on which the security of our signature schemes is based. We present a thorough discussion of these computational problems and tie them to more well-known problems. In particular, we prove the equivalence of several LFSR-based computational problems to their counterparts based on finite fields.

This paper is presented as follows. Section 2 describes the connection between finite fields and characteristic sequences which lead to the reduced representations produced in Section 3. Section 4 lists the operations required to be able to perform the necessary computation for our cryptographic schemes, which are given in Section 5. Section 6 gives a thorough discussion of the computational problems based on finite fields and sequences. Section 7 summarizes and suggests some areas for future study.

2 LFSR's, Characteristic Sequences, and Minimal Polynomials

We first draw the connection between LFSR sequences and finite fields through the use of characteristic sequences and minimal polynomials.

Consider the sequence generated by a linear feedback shift register (LFSR) of order n over $GF(q)$ where q is a prime power. This sequence is given by the recurrence

$$s_{k+n} = a_1 s_{k+n-1} - a_2 s_{k+n-2} + \cdots + (-1)^{n+1} a_n s_k$$

for all $k \geq 0$ where $a_1, \ldots, a_n$ are elements of $GF(q)$.

Let

$$f(x) = x^n - a_1 x^{n-1} + a_2 x^{n-2} - \cdots + (-1)^n a_n \tag{1}$$

and suppose that this polynomial is irreducible over $GF(q)$ with γ a root of f in $GF(q^n)^*$. Also, let $\overline{s}_i = (s_i, s_{i+1}, \ldots, s_{i+n-1})$ be the i-th state of the LFSR sequence. By choosing our initial state $\overline{s}_0$ in a special way, namely $s_i = Tr(\gamma^i)$ for $i = 0, \ldots, n-1$ where Tr is the trace map from $GF(q^n)$ to $GF(q)$, we ensure that $s_k = Tr(\gamma^k)$ for all $k \geq 0$.

The sequence is eventually periodic with period, say, Q. We may then define $s_k = s_{Q+k}$ for all $k \leq 0$. Hence, we may consider the sequence $\{s_k\}$ with indices running over all integers.

A sequence defined in this fashion will be called the *n-th order characteristic sequence* over $GF(q)$ generated by γ or $f(x)$. It is well-known that the following is true.

Proposition 1. *Let $\{s_i\}$ be an n-th order characteristic sequence over $GF(q)$ generated by $\gamma \in GF(q^n)^*$. Then the period of $\{s_i\}$ is equal to the order of γ.*

Characteristic sequences are closely related to minimal polynomials of finite field elements by the Newton Identity.

Lemma 1 (Newton Identity). *Let $\gamma \in GF(q^n)^*$ and $t_i = Tr(\gamma^i)$, $0 \leq i < n$. Then*

$$\prod_{j=0}^{n-1}(x - \gamma^{q^j}) = x^n + \sum_{j=1}^{n}(-1)^j b_j x^{n-j}$$

where for $0 < j \leq n$, the b_j's is defined recursively by the relation

$$t_j - t_{j-1}b_1 + \cdots + (-1)^j j b_j = 0 \tag{2}$$

For all integers k, define the minimal polynomial of γ^k over $GF(q)$ to be

$$f_{\gamma^k}(x) = x^n - a_{1,k}x^{n-1} + a_{2,k}x^{n-2} - \cdots + (-1)^{n-1}a_{n-1,k}x + (-1)^n a_{n,k}$$

The Newton Identity tells us that for any $m \in \{1, \ldots, n\}$, we can efficiently determine the set $\{a_{1,k}, a_{2,k}, \ldots, a_{m,k}\}$ from $\{s_k, s_{2k}, \ldots, s_{mk}\}$ and vice-versa.

Let us now specialize to the case where $n \geq 2$ and γ, and hence γ^k for all k, has order dividing $q^{n-1} + q^{n-1} + \cdots + q + 1$. Now

$$a_{i,k} = \sum_{0 \leq j_1 < j_2 < \cdots < j_i \leq n-1} \gamma^{k(q^{j_1} + q^{j_2} + \cdots + q^{j_i})}$$

When $i = n$, this becomes $a_{n,k} = \gamma^{k(1+q+q^2+\cdots+q^{n-1})} = 1$. Also, for any $1 \leq i \leq n-1$, we have that

$$\begin{aligned} a_{i,k} &= \sum_{0 \leq j_1 < j_2 < \cdots < j_i \leq n-1} \gamma^{k(q^{j_1} + q^{j_2} + \cdots + q^{j_i})} \\ &= \sum_{0 \leq j_1 < j_2 < \cdots < j_{n-i} \leq n-1} \gamma^{-k(q^{j_1} + q^{j_2} + \cdots + q^{j_{n-i}})} \\ &= a_{n-i,-k} \end{aligned} \tag{3}$$

Remark 1. In the previous discussion, it is not necessary that $f(x)$ be irreducible. We may loosen this restriction by saying that $f(x)$ is a polynomial whose roots are conjugate over $GF(q)$. The previous and following analysis will still hold.

3 Reducing Representations of Finite Field Elements

Let $\gamma \in GF(q^n)^*$. Ordinarily, for any integer k, γ^k would require $n \log q$ bits for its representation, normally in a polynomial basis over $GF(q)$. Our goal in this

section is to obtain a smaller representation of γ^k. Again, suppose that $n \geq 2$ and that γ has order dividing $q^{n-1} + q^{n-2} + \cdots q + 1$. Then the minimal polynomial of γ^k would be

$$f_{\gamma^k}(x) = x^n - a_{1,k}x^{n-1} + a_{2,k}x^{n-2} - \cdots + (-1)^{n-1}a_{n-1,k}x + (-1)^n$$

since the constant term is $a_{n,k} = 1$ as shown in the previous section. Thus, at the tradeoff of representing of giving γ^k the same representation as its conjugates $\{\gamma^k, \gamma^{kq}, \ldots, \gamma^{kq^{n-1}}\}$, we may represent γ^k by the set $(a_{1,k}, a_{2,k}, \ldots, a_{n-1,k})$, a total of $(n-1)\log q$ bits. For our purposes, we will accept this tradeoff. Observe that we our now using $\frac{n-1}{n}$ as many bits as in the ordinary case.

We now describe two special cases where we obtain an even shorter representation. Suppose $q = p^2$ and n is even. Let $\gamma \in GF(p^{2n})^*$ have order dividing $p^n + 1$. Then $\gamma^{kp^n} = \gamma^{-k}$ from which we see that for each $1 \leq i \leq n-1$,

$$\begin{aligned} a_{n-i,k} = a_{i,-k} &= \sum_{0 \leq j_1 < j_2 < \cdots < j_i \leq n-1} \gamma^{-k(q^{j_1} + q^{j_2} + \cdots + q^{j_i})} \\ &= \sum_{0 \leq j_1 < j_2 < \cdots < j_i \leq n-1} \gamma^{kp^n(q^{j_1} + q^{j_2} + \cdots + q^{j_i})} \\ &= a_{i,k}^{p^n} = a_{i,k} \end{aligned}$$

where the first equality was established in (3) and the last equality follows from the fact that n is even and $a_{i,k}^{p^2} = a_{i,k}$ since $a_{i,k} \in GF(p^2)$. Hence, we may represent γ^k (and its conjugates) by the set $(a_{1,k}, \ldots, a_{n/2,k})$ which requires $\frac{n}{2}\log q$ bits to represent. This now requires $\frac{1}{2}$ as many bits as in the ordinary case.

Finally, suppose $q = p^2$ and n is odd. Let $\gamma \in GF(p^{2n})^*$ have order dividing $p^{n-1} - p^{n-2} + \cdots - p + 1$. We again have that $\gamma^{kp^n} = \gamma^{-k}$ and for each $1 \leq i \leq n-1$, we get the similar result

$$a_{n-i,k} = a_{i,-k} = a_{i,k}^{p^n} = a_{i,k}^{p}$$

where the last equality is established from the fact that n is odd and $a_{i,k}^{p^2} = a_{i,k}$. Hence, we may represent γ^k (and its conjugates) by the set $(a_{1,k}, \ldots, a_{(n-1)/2,k})$ which requires $\frac{n-1}{2}\log q$ bits to represent. This now requires $\frac{n-1}{2n}$ as many bits as in the ordinary case.

To consider all three cases concurrently, we shall define r by

$$r = \begin{cases} n-1 & \text{for general } q \text{ and } n \\ n/2 & \text{if } q = p^2 \text{ and } n \text{ is even} \\ (n-1)/2 & \text{if } q = p^2 \text{ and } n \text{ is odd} \end{cases}$$

and define the set $A_k = (s_k, s_{2k}, \ldots, s_{rk})$. Observe that from the Newton Identity, we can recover the minimal polynomial coefficients $\{a_{1,k}, a_{2,k}, \ldots, a_{r,k}\}$ and hence the entire minimal polynomial f_{γ^k} from A_k.

Example 1. Let $n = 2$, $q = p$ where p is prime, and the order Q of $\gamma \in GF(p^2)^*$ divides $p + 1$. Then $A_k = (s_k)$ which needs only $\log p$ bits for representation, only $\frac{1}{2}$ as many as in the ordinary case. This is the basis of the LUC [9, 16] cryptosystem. Hence, LUC may be viewed as using second-order characteristic sequences over $GF(p)$.

Example 2. Let $n = 3$, $q = p$ where p is prime, and the order Q of $\gamma \in GF(p^2)^*$ divides $p^2 + p + 1$. Then $A_k = (s_k, s_{2k})$ which needs only $2 \log p$ bits for representation, only $\frac{2}{3}$ as many as in the ordinary case. This is the basis of the GH [6] cryptosystem. Hence, GH makes use third-order characteristic sequences over $GF(p)$.

Example 3. Let $n = 3$, $q = p^2$ where p is prime, and the order Q of $\gamma \in GF(p^2)^*$ divides $p^2 - p + 1$. Then $A_k = (s_k)$ which needs only $2 \log p$ bits for representation, only $\frac{1}{3}$ as many as in the ordinary case. This is the basis of the XTR [6] cryptosystem. Hence, XTR may be viewed as using third-order characteristic sequences over $GF(p^2)$.

Example 4. Let $n = 5$, $q = p$ where p is prime, and the order Q of $\gamma \in GF(p^2)^*$ divides $p^4 + p^3 + p^2 + p + 1$. Then $A_k = (s_k, s_{2k}, s_{3k}, s_{4k})$ which needs only $8 \log p$ bits for representation, only $\frac{4}{5}$ as many as in the ordinary case. This is the basis of the cryptosystem proposed by Giuliani and Gong [3]. Hence, this system makes use of fifth-order characteristic sequences over $GF(p)$.

Example 5. Let $n = 5$, $q = p^2$ where p is prime, and the order Q of $\gamma \in GF(p^2)^*$ divides $p^4 - p^3 + p^2 - p + 1$. Then $A_k = (s_k, s_{2k})$ which needs only $4 \log p$ bits for representation, only $\frac{2}{5}$ as many as in the ordinary case. This is the basis of the cryptosystem proposed by Quoos and Mjølsnes [14] and Giuliani and Gong [3, 4]. Hence, may be viewed as using fifth-order characteristic sequences over $GF(p^2)$.

4 Operations to Calculate Sequence Terms

There are two main operations which will be needed for our schemes in the next section. We describe them here.

Sequence Operation I (SO1): Given A_k and an integer l, where $0 \leq k, l < Q$, compute A_{kl}.

Sequence Operation II (SO2): Given states $\overline{s}_k$ and $\overline{s}_l$ for some $0 \leq k, l < Q$, compute $\overline{s}_{k+l}$.

SO1 can be performed efficiently by the algorithm due to Fiduccia [2], while SO2 can be done efficiently from the theory of shift register sequences [5]. We will now work toward detailing these procedures, starting first with some background.

Define the $n \times n$ matrix

$$C = \begin{bmatrix} 0\,0\,0\cdots 0\,0 & (-1)^{n+1} \\ 1\,0\,0\cdots 0\,0 & (-1)^{n}a_{n-1} \\ 0\,1\,0\cdots 0\,0 & (-1)^{n-1}a_{n-2} \\ \vdots\,\vdots\,\vdots\,\ddots\,\vdots\,\vdots & \vdots \\ 0\,0\,0\cdots 0\,0 & a_3 \\ 0\,0\,0\cdots 1\,0 & -a_2 \\ 0\,0\,0\cdots 0\,1 & a_1 \end{bmatrix}$$

The determinant of C is 1, which means that C is nonsingular. If we now consider the i-th state $\overline{s}_i = (s_i, \ldots, s_{i+n-1})$ as a row vector, then we have that

$$\overline{s}_{i+1} = \overline{s}_i C$$

for all $i \in \mathbb{Z}$. We can iteratively apply this relation to get

$$\overline{s}_{i+k} = \overline{s}_i C^k \tag{4}$$

for all $i, k \in \mathbb{Z}$ with the convention that $C^0 = I_n$, the $n \times n$ identity matrix. For each $i \in \mathbb{Z}$, construct the matrix M_i whose rows consist of the states $\overline{s}_i, \ldots, \overline{s}_{i+n-1}$, that is

$$M_i = \begin{bmatrix} \overline{s}_i \\ \hline \overline{s}_{i+1} \\ \hline \cdots \\ \hline \overline{s}_{i+n-1} \end{bmatrix}$$

Then from (4), we get

$$M_{i+k} = M_i C^k \tag{5}$$

for all $i, k \in \mathbb{Z}$. Note that since C is nonsingular, the M_i's all have the same rank. If the M_i's were singular, we would have a relation of the form

$$\overline{s}_{i+l} = d_1\overline{s}_{i+l-1} + \cdots + d_l\overline{s}_i$$

for all $i \in \mathbb{Z}$ and some l with $1 \leq l < n$, and we could consider this as an l-th order sequence over $GF(q)$. For our purposes, we will assume that the M_i's are nonsingular.

To perform SO2, we simply calculate $C^k = M_0^{-1}M_k$ from (5), whence we can calculate $\overline{s}_{k+l}$ from (4).

To perform SO1, we populate the matrix C with the coefficients $a_{1,k}, \ldots, a_{n-1,k}$ in place of $a_1, \ldots, a_{n-1}$ and use the well-known Cayley-Hamilton theorem.

Theorem 1 (Cayley-Hamilton). *A matrix satisfies its characteristic polynomial. That is, if $F(x)$ is the characteristic polynomial of a matrix C, then $F(C)$ is the zero matrix.*

The characteristic polynomial of C is $f_{\gamma^k}(x)$. Define $\overline{s}_{k,l} = (s_{kl}, s_{k(l+1)}, \ldots, s_{k(l+n-1)})$. This is exactly the l-th state of the n-th order characteristic sequence generated by γ^k. We can obtain the states $\overline{s}_{k,l}, \overline{s}_{k,2l}, \ldots, \overline{s}_{k,rl}$ by Fiduccia's algorithm.

Algorithm 1. Fiduccia

INPUT: C, $f_{\gamma^k}(x)$, $\overline{s}_{k,1}$, l
OUTPUT: $\overline{s}_{k,l}$.

1. $w \leftarrow \lfloor \log_2 l \rfloor$.
2. $l_i \in \{0,1\}$ are such that $l = \sum_{i=0}^{w} l_i 2^i$.
3. $R(x) \leftarrow x$.
4. for i from $w-2$ down to 0 do
   4.1 $R(x) \leftarrow R(x) \cdot R(x) \bmod f_{\gamma^k}(x)$.
   4.2 if $l_i = 1$, $R(x) \leftarrow R(x) \cdot x \bmod f_{\gamma^k}(x)$.
5. $D \leftarrow R(C)$.
6. $\overline{s}_{k,l} \leftarrow \overline{s}_{k,0} D$.

5 Cryptographic Schemes

In this section, we list some cryptographic schemes using finite fields and LFSR sequences. The first two examples are finite field-based key agreement and signature schemes which are well-known in cryptography. The three LFSR-based schemes which follow it are analogues of these systems.

5.1 Finite Field Diffie-Hellman Key Agreement

Diffie and Hellman [1] proposed the following key agreement scheme in their seminal paper.

Domain Parameters: q, n, Q, γ.

Alice:

1. Chooses random private key l, $0 \leq l < Q$.
2. Computes public key γ^l and transmits this to Bob.
3. Receives Bob's public key γ^k.
4. Computes the shared secret $(\gamma^k)^l = \gamma^{kl}$.

Bob: Performs the symmetric operation.

Alice and Bob then both have the shared secret γ^{kl}.

5.2 Finite Field ElGamal Signature Scheme

The following signature scheme has served as the basis for the Digital Signature Standard (DSS) [13]. To be consistent with notation from the DSS, we shall write g in place of γ.

Domain parameters: q, n, Q, g, Q prime.
Private Key: w, $0 \leq w < Q$.
Public Key: $h = g^w$.

Signature Generation:

1. Hash the message M to obtain $H(M)$.
2. Choose random k and compute $r = g^k$.
3. Calculate $s \equiv k^{-1}(H(M) + wr) \pmod{Q}$.
4. The signature is (r, s).

Signature Verification:

1. Compute $u = r^s$.
2. Compute $v = g^{H(M)}h^r$.
3. If $u = v$ accept, otherwise reject.

5.3 LFSR-Based Diffie-Hellman Key Agreement

Niederreiter first proposed a Diffie-Hellman key agreement scheme using general LFSR sequences in [11, 12]. We propose a Diffie-Hellman scheme specifically using characteristic sequences with reduced representations. This has the benefit that it requires Alice and Bob to transmit fewer bits to obtain the shared secret.

Domain Parameters: q, n, Q, A_1.

Alice:

1. Chooses random private key l, $0 \leq l < Q$.
2. Computes public key A_l from A_1 and l using SO1 and transmits this to Bob.
3. Receives Bob's public key A_k.
4. Computes the shared secret A_{kl} from A_k and l using SO1.

Bob: Performs the symmetric operation.

Observe that both Alice and Bob transmit sets of the form A_k which are only $r \log q$ bits.

5.4 General LFSR-Based ElGamal Signature Scheme

Domain parameters: q, n, Q, A_1, Q prime.
Private Key: w, $0 \leq w < Q$.
Public Key: $\overline{s}_w, \overline{s}_{2w}, \ldots, \overline{s}_{mw}$ where $1 \leq m \leq r$.

Signature Generation:

1. Hash the message M to obtain $H(M)$.
2. Choose random k and compute A_k using SO1.
3. Let $r = s_k$ obtained from A_k.
4. Calculate $s \equiv k^{-1}(H(M) + wr) \pmod{Q}$.
5. The signature is (A_k, s).

Signature Verification:

1. Compute r and $l = H(M)r^{-1} \pmod{Q}$.
2. Compute first $\overline{s}_l, \overline{s}_{2l}, \ldots, \overline{s}_{ml}$ from A_1 and l using SO1.
3. Compute $u = (s_{w+l}, s_{2(w+l)}, \ldots, s_{m(w+l)})$ from the public key and the previous step using SO2.
4. Compute $A_{ksr^{-1}}, A_{2ksr^{-1}}, \ldots, A_{mksr^{-1}}$ from A_k and sr^{-1} using SO1 and let $v = (s_{ksr^{-1}}, s_{2ksr^{-1}}, \ldots, s_{mksr^{-1}})$.
5. If $u = v$ accept, otherwise reject.

The signature consists of A_k which is $r \log q$ bits in size and s which is $\log Q$ bits. However, the public key is $mn \log q$ bits which is quite large. We can reduce this to $n \log q$ bits by taking $m = 1$. The public key would then be the same size as a finite field element in canonical representation. However, the signature would still use a reduced representation. The specific signature for $m = 1$ is listed as follows.

5.5 Specific LFSR-Based ElGamal Signature Scheme

Domain parameters: q, n, Q, A_1, Q prime.
Private Key: w, $0 \leq w < Q$.
Public Key: $\overline{s}_w$.

Signature Generation:

1. Hash the message M to obtain $H(M)$.
2. Choose random k and compute A_k using SO1.
3. Let $r = s_k$ obtained from A_k.
4. Calculate $s \equiv k^{-1}(H(M) + wr) \pmod{Q}$.
5. The signature is (A_k, s).

Signature Verification:

1. Compute r and $l = H(M)r^{-1} \pmod{Q}$.
2. Compute first $\overline{s}_l$ from A_1 and l using SO1.
3. Compute $u = s_{w+l}$ from the public key and the previous step using SO2.
4. Compute $A_{ksr^{-1}}$ from A_k and sr^{-1} using SO1 and let $v = s_{ksr^{-1}}$.
5. If $u = v$ accept, otherwise reject.

Observe that signature generation is unchanged. Only signature verification has been modified.

6 Computational Complexity Problems

Let us examine the computational complexity problems relevant to the schemes of the previous section and discuss their relations to more well-known problems.

For a problem $\mathcal{A}$, we write $\mathcal{A} \in P$ if it can be solved in probabilistic polynomial time in its inputs. For two problems $\mathcal{A}$ and $\mathcal{B}$, we shall write $\mathcal{A} \leq_P \mathcal{B}$ if $\mathcal{A}$ can be solved in probabilistic polynomial time with polynomially many queries to an oracle solving $\mathcal{B}$. We also write $\mathcal{A} =_P \mathcal{B}$ if both $\mathcal{A} \leq_P \mathcal{B}$ and $\mathcal{B} \leq_P \mathcal{A}$.

6.1 LFSR-Related Problems

Let us first state some traditional finite field problems.

Definition 1. *The **Discrete Logarithm Problem (DLP)** is, given $\beta \in \langle\gamma\rangle$, to find l such that $\beta = \gamma^l$.*

Definition 2. *The **Diffie-Hellman Problem (DHP)** is, given γ along with γ^k and γ^l, to determine γ^{kl}.*

Definition 3. *The **Decisional Diffie-Hellman Problem (DDHP)** is, given γ along with $\gamma^k, \gamma^l, \gamma^{kl}, \gamma^c$ where c is chosen randomly so that $\gamma^c \neq \gamma^{kl}$, to determine which one of γ^{kl} or γ^c is the solution to the DHP with $\gamma, \gamma^k, \gamma^l$.*

We now define the analogous complexity problems involving LFSR's. We shall refer to them as *LFSR-based* problems.

Definition 4. *The **LFSR-Based Discrete Logarithm Problem (LFSR-DLP)** is, given A_1 and A_l, to find l.*

Definition 5. *The **LFSR-Based Diffie-Hellman Problem (LFSR-DHP)** is, given A_1 along with A_k and A_l, to determine A_{kl}.*

The key agreement scheme in Section 5.3 is based upon the LFSR-DHP.

Definition 6. *The **LFSR-Based Decisional Diffie-Hellman Problem (LFSR-DDHP)** is, given A_1 along with A_k, A_l, A_{kl}, A_c where c is chosen randomly so that $A_c \neq A_{kl}$, to determine which one of A_{kl} or A_c is the solution to the LFSR-DHP with input A_1, A_k, A_l.*

In [17], it was essentially proven that the LFSR-DLP is computationally equivalent to the DLP. We show this proof and prove the analogous for the Diffie-Hellman and Decisional Diffie-Hellman problems below.

Theorem 2. *1. DLP $=_P$ LFSR-DLP.*
2. DHP $=_P$ LFSR-DHP.
3. DDHP $=_P$ LFSR-DDHP.

Proof. In the course of this proof, we will repeatedly transfer from sets of the form A_k to the minimal polynomial γ^k. This can be done by using the Newton Identity. We will also need to find a root of f_{γ^k}. This can be done efficiently using a root-finding algorithm such as the ones due to Rabin [15] or van Oorscot and Vanstone [18].

1) Given an instance A_1, A_k of the LFSR-DLP, we find roots γ and β of the respective minimal polynomials. Then the discrete log l where $\beta = \gamma^l$ is a solution to the LFSR-DLP.

Conversely, if $\beta = \gamma^l$, then we can find the respective minimal polynomials of γ and β and obtain the sets A_1 and A_l. Solving the LFSR-DLP would give an integer k such that $l = kq^i$ for some $i = 0, \ldots, n-1$. A quick check will tell us which is the correct l. This proves the first assertion.

2) Given an instance A_1, A_k, A_l of the LFSR-DHP, we can again find the respective polynomial roots $\gamma, \gamma^{kq^i}, \gamma^{lq^j}$ where $0 \leq i, j < n$. Solving the DHP with these three inputs gives $\gamma^{klq^{i+j}}$ whose minimal polynomial yields the set $A_{klq^{i+j}} = A_{kl}$.

Conversely, suppose we have an instance $\gamma, \gamma^k, \gamma^l$ of the DHP. Converting to minimal polynomial representations, we solve the LFSR-DHP with instances A_1, A_k, A_l and A_1, A_{k+1}, A_l to get A_{kl} and A_{kl+l} respectively. Finding roots gives us γ^{klq^i} and $\gamma^{(kl+l)q^j}$ where $0 \leq i, j < n$. We now calculate $\gamma^{(kl+l)q^j - l}$. Finding the value of j such that this is equal to γ^{klq^i} for some i gives the solution to the DHP. This proves the second assertion.

3) To prove the third assertion, we note that a solution to the DHP with input $\gamma, \gamma^k, \gamma^l$ is γ^c if and only if A_c is a solution to the LFSR-DHP with input A_1, A_k, A_l and A_{c+l} is a solution to the LFSR-DHP with input A_1, A_{k+1}, A_l. Hence, these 2 LFSR-DDHP oracle queries would give the correct decision for the DHP.

Conversely, A_c is a solution to the LFSR-DHP with input A_1, A_k, A_l if and only if γ^{cq^i} is a solution to the DHP with input $\gamma, \gamma^k, \gamma^l$ for some $i = 0, \ldots, n-1$. This can be ascertained with at most n queries to the DDHP oracle. □

We now turn our attention to representations using states. We can define the following computational problems.

Definition 7. *The* ***State-Based Discrete Logarithm Problem (S-DLP)*** *is, given* $\overline{s}_1$ *and* $\overline{s}_l$*, to determine* l*.*

Definition 8. *The* ***State-Based Diffie-Hellman Problem (S-DHP)*** *is, given* $\overline{s}_1$ *along with* $\overline{s}_k$ *and* $\overline{s}_l$*, to determine* $\overline{s}_{kl}$*.*

Definition 9. *The* ***State-Based Decisional Diffie-Hellman Problem (S-DDHP)*** *is, given* $\overline{s}_1$ *along with* $\overline{s}_k, \overline{s}_l, \overline{s}_{kl}, \overline{s}_c$ *where* c *is chosen randomly so that* $\overline{s}_c \neq \overline{s}_{kl}$*, to determine which one of* $\overline{s}_{kl}$ *or* $\overline{s}_c$ *is the solution to the S-DHP with input* $\overline{s}_1, \overline{s}_k, \overline{s}_l$*.*

Lemma 2. *1.* $DLP \leq_P$ *S-DLP.*
2. $DHP \leq_P$ *S-DHP.*
3. $DDHP \leq_P$ *S-DDHP.*

Proof. This lemma follows immediately from the fact that given γ and γ^k, we can calculate $\overline{s}_k$ from the relation

$$s_{k+i} = Tr(\gamma^{k+i}) = Tr(\gamma^k(\gamma)^i)$$

and that Tr is an efficiently computable function.

Lemma 3. *1.* $S\text{-}DLP \leq_P$ *LFSR-DLP.*
2. $S\text{-}DHP \leq_P$ *LFSR-DHP.*
3. $S\text{-}DDHP \leq_P$ *LFSR-DDHP.*

Proof. Given a state $\overline{s}_k = (s_k, \ldots, s_{k+n-1})$, we can use SO2 to calculate $s_{2k}, s_{3k}, \ldots, s_{rk}$ and hence get A_k. Following the proof of Theorem 2, the result follows.

Theorem 3. *1. DLP* $=_P$ *S-DLP.*
2. DHP $=_P$ *S-DHP.*
3. DDHP $=_P$ *S-DDHP.*

Proof. This follows immediately from Lemmas 2 and 3 and Theorem 2.

6.2 Trace-Related Complexity Problems

Let us now formally define problems related to the security of our signature scheme. The security of the signature scheme listed in Section 5.5 is based on the following problem.

Definition 10. *The* ***Trace Discrete Log Problem (Trace-DLP)*** *is the problem of finding, given an element* $t \in GF(q)$*, an index* l *such that* $Tr(\gamma^l) = t$*, or determining that there is no such index.*

The security of the more general signature scheme in Section 5.4 is based on this problem.

Definition 11. *The* m***-Trace Discrete Logarithm Problem (***m***-Trace-DLP)*** *is the problem, given elements* $t_1, \ldots, t_m$*, of finding an integer* l *such that* $Tr(\gamma^{li}) = t_i$ *for all* $i = 1, \ldots, m$*, or determining that there is no such index.*

If this problem were tractable, then we could forge signatures by performing the following algorithm.

Algorithm 2.

INPUT: Signature parameters, public key, message M.
OUTPUT: Valid signature.

1. Perform the first three steps of signature generation as indicated.
2. Use SO1 to calculate $(\overline{s}_{h(M)r^{-1}}, \ldots, \overline{s}_{mh(M)r^{-1}})$.
3. Use SO2 to compute $(s_{h(M)r^{-1}+w}, \ldots, s_{m(H(M)r^{-1}+w)})$.
4. Solve the m-Trace-DLP with $t_i = s_{i(H(M)r^{-1}+w)}$ to get l.
5. The forged signature is (A_k, s) with $s = rlk^{-1} \pmod{Q}$.

The following problems are related to the Trace-DLP problems and will aid is in their examination.

Definition 12. *The* ***Trace Inverse Problem (TraceInv)*** *is the problem, given* $t \in GF(q)$*, of finding an element* $\beta \in \langle\gamma\rangle$ *such that* $Tr(\beta) = t$*, or determining that no such element exists.*

Definition 13. *The* m***-Trace Inverse Problem (***m***-TraceInv)*** *is the problem, given* $t_1, \ldots, t_m \in GF(q)$*, of finding an element* $\beta \in \langle\gamma\rangle$ *such that* $Tr(\beta^i) = t_i$ *for* $i = 1, \ldots, m$*, or determining that no such element exists.*

6.3 The Complexity of Trace-Related Problems

Let us now try to examine how feasible the m-Trace-DLP is, how this changes for different values of m, and whether or not we can relate it to the DLP.

Intuitively, it would seem at first glance that the m_2-Trace-DLP should be at least as difficult as the m_1-Trace-DLP if $m_1 < m_2$. For given $t_1, \ldots, t_{m_2}$ and an element $\beta \in \langle\gamma\rangle$ which solves the m_2-Trace-DLP, β would also solve the m_1-Trace-DLP with input $t_1, \ldots, t_{m_1}$. In addition, there may be elements $\beta \in \langle\gamma\rangle$ which solve the m_1-Trace-DLP , but not the m_2-Trace-DLP with these inputs.

However, when trying to make an actual reduction argument, we run into a problem. Suppose we have an oracle to solve the m_2-Trace-DLP and we wish to solve the m_1-Trace-DLP (again with $m_1 < m_2$) with input $t_1, \ldots, t_{m_1}$. In order to use the oracle, we must extend by including elements $t_{m_1+1}, \ldots, t_{m_2}$ in the oracle call. But it is unclear how to choose these elements. If we choose them at random, then there may not be a solution to the m_2-Trace-DLP with this input, even though there is a solution to the m_1-Trace-DLP with the truncated input.

We now discuss the relations amongst the problems presented in the previous two subsections. We begin by first relating the Trace-DLP and the DLP. We can use TraceInv to aid in this connection.

Lemma 4. *m-TraceInv $\leq_P$ m-Trace-DLP.*

Proof. Given γ and $t_1, \ldots, t_m$, use the oracle for the m-Trace-DLP to find l. Then set $\beta = \gamma^l$.

This theorem is useful for making the following association.

Theorem 4. *For $m_1 < m_2$, if m_1-TraceInv $\in P$, then m_1-Trace-DLP $\leq_P$ m_2-Trace-DLP.*

Proof. Let $t_1, \ldots, t_m \in GF(q)$. Since m_1-TraceInv can be solved in polynomial time, we can find $\beta = \gamma^l$ such that $Tr(\beta^i) = t_i$ for $i = 1, \ldots, m_1$. We then set $t_i = Tr(\beta^i)$ for $i = m_1 + 1, \ldots, m_2$. Now present $t_1, \ldots, t_{m_2}$ to the oracle to solve the m_2-Trace-DLP to get l which solves the m_1-Trace-DLP.

For the moment, let us assume that m-TraceInv $\in P$. Then the larger we choose m, the more difficult the m-Trace-DLP becomes. But does it get increasingly more difficult with every incrementation or is there a limit to how hard it can get? This is answered by the following theorem.

Theorem 5. *r-Trace-DLP $=_P$ DLP.*

Proof. The r-Trace-DLP is exactly the LFSR-DLP, which by Theorem 2 is computationally equivalent to the DLP.

Corollary 1. *For all $m \geq r$, m-Trace-DLP $=_P$ r-Trace-DLP.*

Proof. Clearly, r-Trace-DLP $\leq_P$ m-Trace-DLP since given A_k, we can uniquely calculate s_{ik} from A_k for all $i > r$. But for any instance of the m-Trace-DLP, the solution of the truncated instance to the r-Trace-DLP would be a solution to the m-Trace-DLP.

The question now becomes, how difficult is m-TraceInv? Let us examine just the TraceInv. Given an element $t \in GF(q)$, it is actually quite simple to find an element $\beta \in GF(q^n)$ such that $Tr(\beta) = t$. In fact $\beta = \frac{1}{n}t$ is a preimage of t provided that n is coprime to q. However, when we want our preimage to be in a (relatively small) subgroup of $GF(q^n)^*$, it becomes much more difficult to find preimages. In general, this problem is still open to the knowledge of this author, and appears difficult to solve. But in some instances, TraceInv and m-TraceInv can be solved in polynomial time with high probability.

Proposition 2. *Suppose that γ has order approximately q^r. Then m-TraceInv $\in P$.*

Proof. We may assume that $m \leq r$, since if $m > r$, we may truncate any input $t_1, \ldots, t_m$ to $t_1, \ldots, t_r$ and solve r-TraceInv with this input. Given a solution β, we simply check that $t_i = Tr(\beta^i)$ for $i = r+1, \ldots, m$. Note that any other solution to the r-TraceInv with this input would be conjugate to β and thus give the same traces for all of their powers.

Let $t_1, \ldots, t_m \in GF(q)$. We wish to find one of its solutions $\beta \in \langle\gamma\rangle$, if one exists.

Let A be the m-tuple of elements $t_1, \ldots, t_m$ in $GF(q)$ by choosing $t_{m+1}, \ldots, t_r$ at random in $GF(q)$ such that t_i takes the place of s_{ik} in the representation within A. If $A = A_k$ for some k, we would be able to construct its corresponding minimal polynomial f_{γ^k} over $GF(q)$. Finding a root β of f_{γ^k} would give $\beta = \gamma^{kq^j}$, with $Tr(\beta^i) = t_i$ for $i = 1, \ldots, m$, solving the m-TraceInv Problem.

Given a candidate polynomial, we need only find a root β and then check that β has order dividing Q. These are efficient operations.

Thus, we need only determine the probability of success if we choose our set A in this fashion. Since the order of γ is $\sim q^r$ and each minimal polynomial has n roots, there are approximately, q^r/n polynomials which would give success. But there are q^r possible r-tuples of elements in $GF(q)$. Thus, the probability that a randomly chosen tuple represents the coefficents of a minimal polynomial is approximately $q^r/nq^r = 1/n$. Hence, repreating this trial polynomially many times in n, we are likely to succeed with high probability. If no valid β is produced after a small number of trials, then with high probability, there is no such β.

Remark 2. We note that if γ has order instead approximately q^c for some $c < d$. Then the probability for success would be $1/nq^{d-c}$. It would then take exponentially many attempts to achieve a non-negligible probability.

7 Conclusions and Discussion

We have presented a new general class of cryptosystems based on characteristic sequences generated by LFSR's. We have proposed a signature scheme based on the Trace-DLP and its variants which takes advantage of the compact representations finite field elements. The complexity of the Trace-DLP and related

problems was examined. We also have proven the equivalence of the LFSR-based and State-based sequence problems with their counterparts based on finite fields.

To the knowledge of the authors, this is the first time complexity problems involving traces have been proposed. This is an area which deserves more study. It would also be very nice to find an LFSR-based signature scheme in which both the public key and the signature can be represented by reduced representations. Finally, it would be nice to develop other applications dependent upon the Trace-DLP and TraceInv problems.

References

1. Diffie, W., Hellman, M.E.: New Directions in Cryptography. IEEE Trans. IT. **22** (1976) 644–654.
2. Fiduccia, C.M.: An Efficient Formula for Linear Recurrences. SIAM J. Comput. **14** (1985) 106–112.
3. Giuliani, K., Gong, G.: Analogues to the Gong-Harn and XTR Cryptosystems. Combinatorics and Optimization Research Report CORR 2003-34, University of Waterloo (2003).
4. Giuliani, K., Gong, G.: Efficient Key Agreement and Signature Schemes Using Compact Representations in $GF(p^{10})$. In: Proceedings of the 2004 IEEE International Symposium on Information Theory - ISIT 2004. Chicago (2004) 13–13.
5. Golomb, S. W.: Shift Register Sequences. Holden-Day, San Francisco (1967).
6. Gong, G., Harn, L.: Public-Key Cryptosystems Based on Cubic Finite Field Extensions. IEEE Trans. IT. **24** (1999) 2601–2605.
7. Lenstra, A., Verheul, E.: The XTR Public Key System. In: Advances in Cryptology – Crypto 2000. Lecture Notes In Computer Science, Vol. 1880. Springer-Verlag, Berlin Heidelberg New York (2000) 1–19.
8. Lidl, N., Niederreiter, H.: Finite Fields. Addison-Wesley, Reading (1983).
9. Müller, W. B., Nobauer, R.: Cryptanalysis of the Dickson scheme. In: Advances in Cryptology – Eurocrypt 1985. Lecture Notes In Computer Science, Vol. 219. Springer-Verlag, Berlin Heidelberg New York (1986) 50–61.
10. Niederreiter, H.: A Public-Key Cryptosystem Based on Shift-Register Sequences. In: Advances in Cryptology – Eurocrypt 1985. Lecture Notes In Computer Science, Vol. 219. Springer-Verlag, Berlin Heidelberg New York (1986) 35–39.
11. Niederreiter, H.: Some New Cryptosystems Based on Feedback Shift Register Sequences. Math. J. Okayama Univ. **30** (1988) 121-149.
12. Niederreiter, H.: Finite Fields and Cryptology. In: Finite Fields, Coding Theory, and Advances in Communications and Computing. M. Dekker, New York (1993) 359–373.
13. National Institute of Standards (NIST): Digital Signature Standard. U. S. Government Standard. FIPS-186. (1994).
14. Quoos, L., Mjølsnes, S. F.: Public Key Systems Based on Finite Field Extensions of Degree Five. Presented at Fq7 conference (2003).
15. Rabin, M.: Probabilistic Algorithms in Finite Fields. SIAM J. Comput. **9** (1980) 273–280.
16. Smith, P., Skinner, C.: A Public-Key Cryptosystem and a Digital Signature System Based on the Lucas Function Analogue to Discrete Logarithms. In: Advances in Cryptology – Asiacrypt '94. Lecture Notes In Computer Science, Vol. 917. Springer-Verlag, Berlin Heidelberg New York (1994) 357–364.

17. Tan, C.-H., Yi, X., Siew, C.-K.: On the n-th Order Shift Register Based Discrete Logarithm. IEICE Trans. Fundamentals. **E86-A** (2003) 1213–1216.
18. van Oorchot, P. C.,Vanstone, S. A.: A Geometric Approach to Root Finding in $GF(q^m)$. IEEE Trans. IT. **35** (1989) 444–453