Computer Science Journal of Moldova, vol.**11**, no.2(32), 2003

Finite fields and cryptology

Ennio Cortellini

Abstract

The problem of a computationally feasible method of finding the discrete logarithm in a (large) finite field is discussed, presenting the main algorithms in this direction. Some cryptographic schemes based on the discrete logarithm are presented. Finally, the theory of linear recurring sequences is outlined, in relation to its applications in cryptology.

Keywords: finite field, discrete logarithm, encryption, signature.

1 Finite fields and discrete logarithms

We review some well-known facts about finite fields.

- For any finite field F, there exist a prime p and a positive integer n such that F has p^n elements. The prime p is the characteristic of F.

- Conversely, for any prime p and a positive integer n there is a finite field with $q = p^n$ elements and this field is unique up to isomorphism. Such a field is denoted by F_q. For any $a \in F_q^*$, $a^{q-1} = 1$ by the Theorem of Lagrange from group theory.

- Let F_q be a finite field and Ω its algebraic closure. Then, for any positive integer n, the splitting field of the polynomial $X^{q^n} - X$ over F_q is the unique subfield of Ω with q^n elements. Moreover,

$$F_{q^m} \subseteq F_{q^n}$$

if and only if m divides n. Thus, the lattice of the subfields of Ω containing F_q is isomorphic to the lattice of the nonnegative integers ordered by divisibility.

- For any prime p, F_p is simply the field $\mathbb{Z}_p = \{\widehat{0}, \widehat{1}, ..., \widehat{p-1}\}$ of integers modulo p. In order to construct a field F_q, with $q = p^n$, it is enough to find an irreducible polynomial g of degree n in $\mathbb{Z}_p[X]$, the ring of univariate polynomials with coefficients in $\mathbb{Z}_p$. Then F_q can be taken as the factor ring $\mathbb{Z}_p[X]/(g)$ (which is actually a field).

- $F_q^* = F_q \backslash \{0\}$ is a multiplicative group, which is *cyclic*. A generator of F_q^* is called a *primitive element* of F_q. This simple fact plays a major role in what follows. A polynomial g of degree n in $F_q[X]$ is called *primitive* if it is the minimal polynomial of a primitive element of F_{q^n}.

We fix a prime p and $q = p^t$, with t a positive integer.

1.1 Definition. Let b a primitive element of F_q. Then every $a \in F_q^*$ can be written uniquely as

$$a = b^r,$$

where r is an integer with $0 \leq r \leq q-2$, called *the discrete logarithm of a in base b* and denoted by $\log_b(a)$ (sometimes also called *the index of a relative to b* and denoted $\text{ind}_b a$).

1.2 Remark. The discrete logarithm can be defined in any finite cyclic group G (its operation written multiplicatively). Let b be a generator of G and $n = |G|$ the order of b. For any $a \in G$, there exists a unique integer r with $0 \leq r \leq n-1$ such that $a = b^r$ (r is called the *discrete logarithm of a in base b* and is denoted $\log_b a$).

When b and a as above are given, the task of computationally determining $\log_b(a)$ is known as the *discrete logarithm problem* (DLP) and it is presumed to be difficult if n is large. Let $w = [\log_2 n] + 1$ (the number of bits needed to represent n). The classical algorithm of exponentiation using the base 2 expansion of r computes b^r from b and r in at most $2w$ multiplications in G, so taking powers is of $O(w)$ time.

On the other hand, finding $\log_b(a)$ by an exhaustive search (i.e., compute all powers of b until a is found) takes at most $\text{ord}(b)$ multiplications and equality checks. In the direction of finding lower bounds for the running time of the DLP solving algorithms, Shoup [32] has shown that any "generic" probabilistic algorithm for finding with high probability the discrete logarithm in the cyclic group $\mathbb{Z}_N$ (i.e., which does not use any special properties of the group other than the fact that each group element is encoded as a unique binary string) must

perform at least $cp^{1/2}$ group operations, where p is the largest prime dividing N and c is a constant. So any generic DLP algorithm takes at least $O(2^{w/2}) = O(n^{1/2})$ steps, which can be made prohibitively large if w is large. While the hardness of the DLP is not entirely proven, the prevailing opinion is that discrete logarithm based cryptographic systems are secure enough if implemented appropriately.

2 Algorithms for determining the discrete logarithm

We give some facts on the computation of the discrete logarithm in F_q. If m and n are integers, with $n > 0$, we denote by $m \bmod n$ the positive remainder of m when divided by n. So, $m \bmod n \in \{0, 1, ..., n-1\}$.

2.1 Proposition. *Suppose that, given a prime p, $q = p^t$, b a primitive element of F_q^* and $a \in F_q^*$, there exists an algorithm DL_p which outputs $log_b(a) mod\ p$. Then there is an algorithm which finds $log_b(a)$.*

Proof. Let $r = \log_b(a)$. Then r is written uniquely as:

$$r = x_0 + x_1 p + ... + x_{t-1} p^{t-1}, \text{ with } x_i \in \{0, ..., p-1\}.$$

The x_i are the digits of r in its development in base p. Thus, $r \bmod p = x_0$ can be determined by the algorithm DL_p. We can find now x_1 by applying DL_p to $b^{(r-x_0)/p} =: r_1$. But $b^{(r-x_0)/p} = (b^r b^{-x_0})^{q/p} = (ab^{-x_0})^{q/p}$ and one easily sees that all computations in the last expression are feasible. The process is now applied to r_1. Continuing in this way, one obtains succesively $x_1, ..., x_{t-1}$, which yield $\log_b(a)$.

2.2 Remark. In [18], Mullen and White prove the following explicit formula for $\log_b(a) \bmod p$:

$$\log_b(a) \equiv -1 + \sum_{j=1}^{q-2} a^j (b^{-j} - 1)^{-1} (\text{mod } p).$$

The formula is valid for any $q = p^t$ with $q \geq 3$ and any $a \in F_q^*$. But the ways (if any) to apply this formula to an efficient computation of $\log_b(a)$ for large q are unknown.

2.3 There is an efficient algorithm to compute the discrete algorithm in F_q if $q-1$ decomposes in a product of small primes (the *Pohlig Hellman* algorithm). With the notations above, $r = log_b(a)$ is smaller than $q-1$, so $r = r \bmod (q-1)$. Suppose $q-1 = q_1...q_k$, where q_i is a power of a prime p_i and the p_i's are distinct, for $1 \leq i \leq k$. By the Chinese Remainder Theorem, $r \bmod (q-1)$ can be uniquely found if $r \bmod q_i$ are known for $1 \leq i \leq k$. Furthermore, as shown in the preceding proposition, computing $r \bmod q_i$ amounts to computing $r \bmod p_i$. Let $s_i = r \bmod p_i$. Then

$$a^{(q-1)/p_i} = b^{r(q-1)/p_i} = c_i^r,$$

where $c_i = b^{(q-1)/p_i}$ is an element of F_q^* of order p_i. So $c_i^r = c_i^{s_i}$ and thus s_i can be uniquely determined from $c_i^{s_i}$. To make this task easier, a list of the powers of c_i can be precomputed and s_i read from the list. Moreover, the baby step - giant step method (see below) can be used in this stage for recovering the r from the list.

Obviously, this procedure is of practical value only if the p_i's are small. For details, see Pohlig and Hellman [1], McCurley [12].

The Pohlig Hellman method can be easily generalized to an arbitrary cyclic finite group G satisfying some natural computational properties. Thus, the method reduces the DLP problem in a general group to the case of groups of prime power order.

2.4 Shanks' [28] "*baby step - giant step*" method uses a time-memory trade off algorithm (it uses a large amount of memory to cut down the time it would normally take to solve the problem). It is a deterministic algorithm, devised initially to compute ideal class numbers of quadratic number fields. We present a version of the algorithm that works in any finite cyclic group G (more generally, if in a finite group G, the elements g and a are such that a belongs to the subgroup generated by g, the method determines an integer x such that $a = g^x$).

Let $g, a \in G$ with $n = ordg$, such that $a = g^r$ for some $r \in \{0, ..., n-1\}$ that we want to find. Assume that the set of the computer representations of the elements of G is totally ordered and can be ordered efficiently. Let $m = [n^{1/2}]$.

Step 1. Compute g^{mj}, for $j = 0, ..., m-1$. (the "giant steps")

Step 2. Sort the pairs $(g^{mj}, j), j = 0, ..., m-1$ by the first coordinate and store them in a list GS.

Step 3. Compute ag^{-i} for $i = 0, ..., m-1$. When doing this, for every i check if $ag^{-i} = 1$. If this is the case, set $x = i$ and stop. Otherwise, we end up with a table BS (the "baby steps") with entries $(ag^{-i}, i), i = 0, ..., m-1$.

Step 4: Sort the list BS by the first coordinate.

Step 5: Find a pair in each list with the same first coordinate, i.e., (y, j) in BS and (y, i) in GS. So we have $y = g^{mj} = ag^{-i}$, i.e. $a = g^{mj+i}$.

Step 6: Set $r = mj + i$.

There certainly is a match in step 5, since the list BS contains m consecutive powers of g and GS contains all powers of g that have exponent a multiple of m.

Observe that steps 1 and 2 are independent of a so they can be regarded as a precomputation. Alternatively, one can begin with steps 3 (and 4 if necessary) and proceed with step 1, checking for every j if there is some i such that (g^{mj}, i) belongs to the list BS (easily done if BS is ordered). This avoids storing the list GS, but the computation of the g^{mj}'s has to be done again for every input a.

If $r \geq n$, the algorithm uses one inversion and $m+[n/m]+O(log_2 m)$ multiplications. It uses m (or $2m$ if GS is precomputed) locations for storing group elements and $[n/m]$ table look-ups in a table of size m.

Some variants of the baby-step giant-step method have been proposed, based on a dynamic increase of the giant step size (so the giant steps cannot be performed before the baby steps, as they depend dynamically on them). In Adleman [1], the step size is doubled at certain intervals. In Terr [34], the method increases the giant step size after each baby step and is based on a representation of integers using triangle numbers.

Although (in theory) in this method the running time is reduced by a factor of $\sqrt{2}$, in practice it performs equally well as the original method. Also, in a worst case scenario (i.e. r is close to n), the original method requires less group operations than the modified versions. For a discussion of the efficiency of various baby step - giant step methods, depending on the probability distribution for the solution r on the set

of positive integers, see Teske [35].

2.5 Another case in which a computationally feasible method of determining the discrete logarithm is available is when q is either a prime or a power of a small prime. Of course, it is of interest when q is large. The ideas go back to Western and Miller [40] and Miller [19]. The algorithm appears in Adleman [1], Merkle [18], and Pollard [27]. We sketch the algorithm here.

The method (the *index-calculus algorithm*) depends on the initial choice of a set $\{a_1, ..., a_k\} \subseteq F_q^*$. Given $a_1, ..., a_k$, one then seeks for a set of m identities of the following type:

$$\prod_{j=1}^{k} a_j^{e_{ij}} = b^{f_i}, 1 \leq i \leq m. \tag{1}$$

where e_{ij} and f_i are integers. These identities are equivalent to the following system of congruences mod $q-1$:

$$\sum_{j=1}^{k} e_{ij} \log_b (a_j) \equiv f_i \,(\text{mod } q-1)\,, 1 \leq i \leq m.$$

The e_{ij} should be chosen such that the system above (with the unknowns $\log_b(aj)$) has a unique solution mod $q-1$ and thus $\log_b(a_j)$ can be found.

With the $a_1, ..., a_k$ and their discrete logarithms prepared as above (this is done only once for the given field F_q), the method computes the discrete logarithm $\log_b(a)$ by constructing an identity of the form:

$$\prod_{j=1}^{k} a_j^{h_j} = ab^f. \tag{2}$$

Since $a = b^r$, where $r = log_b(a)$, this is equivalent to

$$r \equiv \sum_{j=1}^{k} h_j \, \log_b(a_j) - f(\text{mod } q-1),$$

which gives $r = \log_b(a)$.

There remains the problem of describing a method for the selection of the system $a_1, ..., a_k$ and how to find the identities (1) and (2). If q is prime, one chooses $a_1, ..., a_k$ to be small primes and the identities (1) are taken to be the decomposition of integers in prime factors. In practice, one way to obtain such identities would be to compute some powers of b, select those that are small and take the canonical factorization of these powers into primes (in this way, the f_i's are known in advance and the e_{ij}'s will be given by factorizing into primes). For an efficient computation, it seems that a probabilistic algorithm is the only suitable alternative in determining the identities (1) and (2). Details can be found in McCurley [12], Odlyzko [24], Lidl and Niederreiter [13].

Coppersmith [6], [7] has devised a probabilistic variant of the index-calculus algorithm for computing the discrete logarithm in the case $q = 2^t$. It was shown (under some assumptions) to have a complexity of $O(\exp(ct^{1/3}\ \log(t^{2/3})))$, for some positive constant c, which is significantly better than $O(exp(t^{1/2}))$ for the basic version; but it does not apply to finite fields with characteristic other than two. A detailed analysis of the Coppersmith algorithm is made by Odlyzko [24].

2.6 *Pollard's rho method* [27], generalized by Teske [36], [37], can be applied in determining discrete logarithms in any cyclic group $(G, *)$ with generator g, provided G satisfies the following computational conditions:

- Given any two elements $g, h \in G$, their product $g * h$ can be computed in G.

- Given any two elements $g, h \in G$, one can decide if $g = h$.

- Given a "small" integer r (originally Pollard assumed $r = 3$), one can find r disjoint subsets $T_1, ..., T_r$ whose union is G, of roughly equal size; moreover, given any group element we can check to which of these sets it belongs.

The method uses an initially chosen function (the "iterating function") $\varphi : G \to G$ and a starting value $y = y_0 \in G$. Then it computes the sequence $(y_k)_k \geq 0$, where $y_{k+1} = \varphi(y_k)$, for any $k \geq 0$.

Since G is finite, there exist two uniquely determined smallest integers $\lambda \geq 1$ and $\mu \geq 0$ such that $y_{k+\lambda} = y_k$, for all $k \geq \mu$. Thus, the sequence y_k is periodic for $k \geq \mu$ (λ is called the *period* of y_k and μ is

the *preperiod* of y_k). The name *rho method* comes from the fact that, when drawing the succesive terms of y_k starting from the bottom of the paper, the terms y_k with $k < \mu$ are arranged in a vertical line and the terms y_k with $k \geq \mu$ form a cycle; thus one gets a picture similar to the Greek letter ρ (rho).

Since the Pohlig and Hellman method allows the reduction to the prime power order, we may suppose that $|G| = p$, a prime integer. Here is the rho method of solving the DLP in G, described using the original Pollard iterating function. Let g be a generator of G and $h \in H$; we want to find $x = \log_g h$.

Pollard's original iterating function φ (adapted to this case) is defined as follows: divide G into 3 pairwise disjoint sets T_1, T_2, T_3 of about the same size and define, for any $y \in G$:

$$\varphi(y) = \begin{cases} g * y, & y \in T_1 \\ y^2, & y \in T_2 \\ h * y, & y \in T_3 \end{cases}$$

Choose a random integer $\alpha \in \{0, ..., p-1\}$ and set $y_0 = g^\alpha$, $y_{k+1} = \varphi(y_k)$, for any $k \geq 0$. Thus, there exist two sequences (α_i) and (β_i) in $\mathbb{Z}_p$ such that $y_i = g^{\alpha_i} * h^{\beta_i}$, for $i \geq 0$. These sequences satisfy:

$$\alpha_0 = \alpha; \beta_0 = 0;$$

$$\alpha_{i+1} = \alpha_i + 1 \text{ and } \beta_{i+1} = \beta_i \text{ or}$$

$$\alpha_{i+1} = 2\alpha_i \text{ and } \beta_{i+1} = 2\beta_i \text{ or}$$

$$\alpha_{i+1} = \alpha_i \text{ and } \beta_{i+1} = \beta_i + 1,$$

depending on the three cases in the definition of φ.

One tries to find a *match*, i.e. a pair (i, j) with $y_i = y_j$ and $i < j$. There are efficient methods to do this (e.g. Brent [3], Schnorr and Lenstra [30]). Once we found a match, $g^{\alpha_j - \alpha_i} = h^{\beta_j - \beta_i} = g^{x(\beta_j - \beta_i)}$, so we get the equation in $\mathbb{Z}_p$:

$$\alpha_j - \alpha_i = x(\beta_j - \beta_i).$$

So $x = \log_g h$ is given by solving the equation above, if $\gcd(\beta_j - \beta_i, p) = 1$. If this is not the case, choose another α and start over again, but this is very rare for large p.

If the iterating function φ is chosen randomly among the $|G|^{|G|}$ functions from G to G, then the expected value for $\lambda + \mu$ is approximately $\sqrt{\pi |G|/2} \approx 1.253\sqrt{|G|}$.

The space requirements of algorithms using the rho method are negligible, so in this respect this method is superior to Shanks' baby step - giant step method that has roughly the same run time.

3 Cryptographic systems based on the discrete logarithm problem

On the assumption of the difficulty of the DLP some cryptographic systems have been proposed.

3.1 The *Diffie-Hellman key agreement protocol* (also called *exponential key agreement*) was developed by Diffie and Hellman [8] in 1976. The protocol allows two users to exchange a secret key over an insecure medium without any prior secrets.

The protocol has two parameters: q (a power of a prime number) and g (a primitive element of F_q^*, also called a *generator*). Parameters g and q are public.

If the users A and B want to agree on a shared secret key, they proceed as follows. First, A generates a random private value a and B generates a random private value b, with a and $b \in \{2, ..., q-2\}$. Then they derive their public keys (which are elements of F_q^*): A's public key is g^a and B's public key is g^b. The public keys are exchanged between A and B. The *common secret key* is then $k = g^{ab}$, which A computes as $(g^b)^a$, and B computes as $(g^a)^b$.

The protocol assumes that it is computationally infeasible to calculate the shared secret key $k = g^{ab}$, given the two public values g^a and g^b, when q is sufficiently large. Maurer [15] has shown that breaking the Diffie-Hellman protocol is equivalent to computing discrete logarithms under certain assumptions, i.e. the only way for an attacker to

compute the secret key g^{ab} would be to compute a from g^a or b from g^b.

3.2 The *ElGamal system* is a public-key cryptosystem based on the discrete logarithm problem. It consists of encryption and signature algorithms. The *encryption algorithm* is based on the same ideas found in the Diffie-Hellman key agreement protocol.

As above, q (a power of a prime number) and a primitive element g of F_q^* are public. A has a private key a and a public key y ($2 \leq a \leq q-2$, $y \in F_q^*$), where $y = g^a$. Suppose B wishes to send a message m ($m \in F_q^*$) to A. B chooses randomly k with $2 \leq k \leq q-2$ and computes in F_q^*

$$y_1 = g^k \text{ and } y_2 = my^k.$$

B then sends (y_1, y_2) to A. We have

$$m = (mg^{ak})g^{-ak} = y_2(g^{-k})^a = y_2(y_1^{-1})^a.$$

So, after receiving the ciphertext (y_1, y_2), A can recover the message m as:

$$m = (y_1^{-1})^a y_2.$$

A variant of this protocol is as follows: if the elements in F_q^* are represented as words of w bits, y_2 above can be replaced by $y_2 = m$ XOR y^k, where XOR denotes the bitwise "exclusive or" between the words of w bits. In this case, m can be computed by A as $m = y_2$ XOR y_1^a.

We describe now the *ElGamal signature algorithm.*

User A, who wishes to sign messages electronically, publishes a prime power q, a primitive root $g \in F_q^*$, and an integer $y, 1 \leq y \leq q-2$, which is generated by choosing a random integer a, which is kept secret, and setting $y = g^a$. The prime power q and the primitive root g can be the same for all the users of the system, in which case only y is special to user A. To sign a message $m \in F_q^*$, user A provides a pair of integers $(r, s), 1 \leq r, s \leq q-2$, such that

$$g^m = y^r r^s.$$

To generate r and s, user A chooses a random integer k with $\gcd(k, q-1) = 1$ and computes $r = g^k$. Since $y = g^a$, this means that s has to satisfy

$$g^m = g^{ar+ks},$$

which is equivalent to $m \equiv ar + ks \pmod{q-1}$. Since $\gcd(k, q-1) = 1$, this equation has a unique solution modulo $q-1$, and this solution is easy to find for A, who knows a, r, and k. An efficient discrete logarithm algorithm would make this scheme insecure, since it would enable the cryptanalyst to compute a from y. No way has been found for breaking this scheme without the ability to compute discrete logarithms.

The ElGamal signature algorithm is similar to the encryption algorithm in that the public key and private key have the same form; however, encryption is not the same as signature verification, nor is decryption the same as signature creation (as in the Rivest-Shamir-Adleman (RSA) system, for example).

The U.S. National Institute of Standards and Technology (NIST) has established the Digital Signature Standard (DSS), which is based on a variant of the ElGamal signature algorithm above, to be the digital authentication standard of the U.S. government. Here is a brief description of the principles of the Digital Signature Algorithm (DSA), according to [33]. The algorithm has the following parameters:

- a 512-bit prime p such that the DLP in F_p^* is intractable and $p-1$ has a prime divisor q which has 160 bits;

- an element g of order q in F_p^*.

Assume user A has a secret key a F_p^*. The user A publishes $g^a = b \in F_p^*$. When A wishes to sign a message $m \in F_p^*$, A randomly chooses $k \in \mathbb{Z}_q$ and computes the pair $(\gamma, \delta) \in \mathbb{Z}_q \times \mathbb{Z}_q$, where:

$$\gamma = g^k (\text{mod } q), \delta = (m + a\gamma)k^{-1} (\text{mod } q)$$

Since g has order q in F_p^*, g^k is well defined in F_p^*. The pair (γ, δ) is then appended to the message m. The receiver of the message can now verify the authenticity of the message m by checking that

$$\gamma = (g^m \cdot b^\gamma)^{\delta^{-1}} (\text{mod } q)$$

For a detailed description of DSA, see [21].

3.3 Another system that uses exponentiation in finite fields to transmit information is based on an idea due to Shamir ([11] pp. 345-346) and Massey and Omura [39]. For example, suppose user A wishes to send a message m (where m is a nonzero element of the publicly known field F_q) to user B. Then A chooses a random integer $c, 1 \leq c \leq q-1, (c, q-1) = 1$, and transmits $x = m^c$ to B. User B then chooses a random integer $d, 1 \leq d \leq q-1, (d, q-1) = 1$, and transmits $y = x^d = m^{cd}$ to A. User A now forms $z = y^{c'}$, where $cc' \equiv 1 \pmod{q-1}$, and transmits z to B. Since

$$z = y^{c'} = m^{cdc'} = m^d,$$

B only has to compute $z^{d'}$ to recover m, where $dd' \equiv 1 \pmod{q-1}$, since $z^{d'} = m^{dd'} = m$.

In this scheme it is again clear that an efficient method for computing discrete logarithms over F_q would enable a cryptanalyst to recover the plaintext message m from the transmitted ciphertext messages m^c, m^{cd}, and m^d.

Finally, we mention that the ability to compute quantities generalizing discrete logarithms in rings of integers modulo composite integers would lead to efficient integer factorization algorithms ([2], [14]). This shows that, in a way, the RSA cryptosystem (based on the difficulty of factoring large integers) and the discrete logarithm based cryptosystems are roughly equally hard to break.

4 Alternatives to the finite field DLP

In view of the progress of the algorithms dedicated to the DLP in groups of type F_q^*, other groups have been proposed for use in cryptographic applications: *ideal class groups of algebraic number fields* (see e.g. [5]) or the group of rational points on an *elliptic curve over a finite field* (for an introduction, see [10]).

The known methods for computing general elliptic curve discrete logarithms are less efficient than those for computing discrete logarithms in finite fields. As a result, shorter key sizes can be used to

achieve the same security of public-key cryptosystems, which might lead to better memory requirements and improved performance. The ElGamal, DSA, and Diffie-Hellman schemes can easily be adapted to construct elliptic curve encryption, signature, and key agreement schemes. These variants seem to offer certain implementation advantages over the original schemes, and consequently they draw increasing attention from the academic community and the industry. For more information on elliptic curve cryptosystems, see the survey articles [17] or [29].

Another idea that generalizes the DLP is based on the theory of the linear *recurring sequences* in a finite field F_q. We present the basic ideas of the theory, which also applies to the generation of *pseudorandom sequences*.

A sequence $(s_n)_{n\geq 0}$ of elements of F_q is called a *linear recurring sequence (LRS) of order* k if there exist $a_0, ..., a_{k-1} \in F_q$ such that

$$(*) \qquad s_{n+k} = a_0 s_n + ... + a_{k-1} s_{n+k-1}, \forall n \geq 0.$$

Such a sequence (abbreviated (s_n)) is also called a *linear feedback shift register sequence*, since these sequences are generated by hardware devices known as linear feedback shift registers.

For instance, the sequence $(g^r)_{r\geq 0}$ that appears in the DLP (where g is a primitive element of F_q) is a linear recurring sequence in F_q of order 1.

For a given LRS (s_n) of order k, define the *state vectors*

$$s_n = (s_n, ..., s_{n+k-1}) \in F_q^k, \forall n \geq 0.$$

To the sequence (s_n) is associated the following $k \times k$ *characteristic matrix* (which is also the *companion matrix* of f):

$$A = \begin{bmatrix} 0 & 0 & \dots & 0 & a_0 \\ 1 & 0 & \dots & 0 & a_1 \\ 0 & 1 & \dots & 0 & a_2 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \dots & 1 & a_{k-1} \end{bmatrix} \in M_k(F_q).$$

The recurrence relation (*) is easily seen to be equivalent with:

$$s_n = s_0 A^n, \forall n \geq 0.$$

This relation enables the efficient computation of arbitrary terms of (s_n) by computing A^n; the standard exponentiation algorithm achieves this in $O(\log_2 n)$ steps.

Since there are only q^k vectors in F_q^k, a LRS of order k is *periodic*: there exist integers n_0 and r such that $s_{n+r} = s_n, \forall n \geq n_0$. The smallest such n_0 is called the *least preperiod* and the smallest such r is the *least period* of (s_n). The least period of a LRS of order k is at most $q^k - 1$. For cryptographic applications, one is interested in LRS with preperiod 0 and least period as large as possible, i.e. $q^k - 1$. In order to determine the least period and the least preperiod of a LRS (s_n) satisfying (*), define a *characteristic polynomial* f of (s_n) as the characteristic polynomial of A:

$$f = X^k - a_0 - a_1 X - ... - a_{k-1} X^{k-1} \in F_q[X].$$

Let F_q^N be the F_q-vector space of all sequences of elements of F_q and let $T : F_q^N \to F_q^N$ be the left shift operator, $T((x_0, ..., x_n, ...)) = (x_1, ..., x_{n+1}, ...)$. It follows that $f(T)((s_n)) = (0)$, the zero sequence. The monic polynomial $m \in F_q[X]$ which is the generator of the ideal

$$\{h \in F_q[X] | h(T)((s_n)) = (0)\}$$

is called the *minimal polynomial of* (s_n). Thus, m divides any characteristic polynomial f. The minimal polynomial carries the relevant information on the periodicity of (s_n):

4.1 Theorem. [13] *Let (s_n) be a LRS and let $m \in F_q[X]$ be its minimal polynomial. Then the least period of (s_n) is the order of m and the least preperiod of (s_n) is the multiplicity of 0 as a root of m.*

The *order* $\mathrm{ord}(g)$ of a nonzero polynomial $g \in F_q[X]$ is defined as the least positive integer e for which g divides $(X^e - 1)X^k$ for some $k \geq 0$. This notion is connected to the *primitive polynomials*: The polynomial $g \in F_q[X]$ of degree n is primitive if and only if $g(0) \neq 0$ and the order of g is $q^n - 1$ ([12], Theorem 3.16).

A LRS (s_n) of order k whose minimal polynomial m is primitive is called a *maximal period sequence*. If $\deg(m) = k$, then the least period of such a sequence is $q^k - 1$ by the theorem above. Thus the collection of the state vectors $\{s_i | i = 0, ..., q^{k-2}\}$ equals all nonzero vectors in F_q^k. This property of maximal period sequences makes them attractive for many applications.

In order to give the analogue of the DLP in the more general setting of linear recurring sequences, the following notion is needed: for a LRS $\sigma = (s_n)$ and a positive integer d, the *decimation of index d* of σ is $\sigma^{(d)} = (s_{nd})$ (i.e, the sequence obtained by taking every d-th term of σ). The decimated sequence $\sigma^{(d)}$ is again a LRS and in fact a characteristic polynomial f_d is obtained from a characteristic polynomial f of σ as follows: if $\alpha_1, ..., \alpha_k$ are the roots of f in its splitting field over F_q, then f_d is the polynomial in $F_q[X]$ with roots $\alpha_1^d, ..., \alpha_k^d$. This follows from the fact that a chracteristic matrix for $\sigma^{(d)}$ is A^d (see [22]).

The analogue to the DLP becomes the following: given the characteristic polynomial f of a LRS σ and the characteristic polynomial f_d of the decimated sequence σ^d, find the decimation index d. Niederreiter [23] has introduced a family of cryptosystems based on these principles.

References

[1] L. M. Adleman, *A subexponential algorithm for the discrete logarithm problem with applications to cryptography*, Proc. 20th IEEE Found. Comp. Sci. Symp., pp. 55–60, 1979.

[2] E. Bach, *Discrete logarithms and factoring*, Computer Science Division (EECS), University of California, UCB/CSD 84/186, 1984.

[3] R.P. Brent, *An improved Monte Carlo factorization algorithm*, BIT 20, pp. 176–184, 1980.

[4] J. Buchmann, M.J Jacobson, E. Teske, *On some computational pro-blems in finite abelian groups*, Math. of Computation, 66: pp. 1663–1687, 1997.

[5] J. Buchmann and H.C. Williams, *A key exchange system based on imaginary quadratic fields*, J. Cryptology 1, pp. 107–118 (1988).

[6] D. Coppersmith, *Evaluating logarithms in GF*(2^n), in Proc. 16th ACM Symp. Theory of Computing, pp. 201–207, 1984.

[7] D. Coppersmith, *Fast evaluation of logarithms in fields of characteristic two*, IEEE Trans. Inform. Theory IT-30, pp. 587–594, 1984.

[8] W. Diffie and M.E. Hellman, *New directions in cryptography*, IEEE Transactions on Information Theory, IT-22: pp. 644–654, 1976.

[9] T. ElGamal, *A public key cryptosystem and a signature scheme based on discrete logarithms*, IEEE Trans. Inform. Theory 31, pp. 469–472 (1985).

[10] N. Koblitz, *A Course in Number Theory and Cryptography*, Springer 1987.

[11] A. G. Konheim, *Cryptography: A Primer*, Wiley, 1981.

[12] R. Lidl and H. Niederreiter, *Finite Fields*, Addison-Wesley, 1983.

[13] R. Lidl and H. Niederreiter, *Introduction to Finite Fields and their Applications*, Cambridge University Press, 1994.

[14] D. L. Long, *Random equivalence of factorization and computation of orders*, Princeton University, DEECS, no. 284, 1981.

[15] U. Maurer, *Towards the equivalence of breaking the Diffie-Hellman protocol and computing discrete logarithms*, in Advances in Cryptology - Crypto '94, pp. 271–281, Springer-Verlag, 1994.

[16] K.S. McCurley, *The discrete logarithm problem, Cryptology and Computational Number Theory*, C. Pomerance, ed., Proc. Symp. Applied Math. 42, pp. 49–74, AMS, Providence RI, 1990.

[17] A. Menezes, *Elliptic Curve Cryptosystems*, CryptoBytes (2) 1 (Summer 1995).

[18] R. Merkle, *Secrecy, authentication, and public key systems*, Ph.D. dissertation, Dept. of Electrical Engineering, Stanford Univ., 1979.

[19] J. C. P. Miller, *On factorization, with a suggested new approach*, Math. Comp. 29, pp. 155–172, 1975.

[20] G.L. Mullen and D. White, *A polynomial representation for logarithms in GF(q)*, Acta. Arith. 47, pp. 255–271, 1986.

[21] National Institute of Standards and Technology (NIST), FIPS Publication 186: *Digital Signature Standard (DSS)*, 1994.

[22] H. Niederreiter, *A simple and general approach to the decimation of feedback shift register sequences*, Problems of Control and Information theory 17, pp. 327–331, 1988.

[23] H. Niederreiter, *Some new cryptosystems based on feedback shift register sequences*, Math. J. Okayama Univ. 30, pp. 121–149, 1988.

[24] A.M. Odlyzko, *Discrete logarithms in finite fields and their cryptographic significance*, Advances in Cryptology, LN in Computer Science vol. 209, Springer Berlin, pp. 224–314, 1985.

[25] A.M. Odlyzko, *Discrete logarithms: The past and the future*, Designs, Codes, and Cryptography 19, pp. 129–145, 2000.

[26] S.C. Pohlig and M.E Hellman, *An improved algorithm for computing logarithms over GF(p) and its cryptographic significance*, IEEE Transactions on Information Theory, 24, pp. 106–110, 1978.

[27] J. M. Pollard, *Monte Carlo methods for index computation (mod p)*, Mathematics of Computation, 32(143): pp. 918–924, 1978.

[28] J. M. Pollard, *Kangaroos, Monopoly and discrete logarithms*, Journal of Cryptology 13, pp. 437–447, 2000.

[29] M.J.B. Robshaw and Y.L. Yin, *Elliptic Curve Cryptosystems*, Technical Note, RSA Laboratories, 1997.

[30] C.P. Schnorr and H.W. Lenstra, jr., *A Monte Carlo factoring algorithm with linear storage*, Math. of Computation, 43 (167), pp. 289–311, 1984.

[31] D. Shanks, *Class number, a theory of factorization and genera*, In Proc. Symp. Pure Math. 20, pp. 415–440. AMS, Providence, R.I., 1971.

[32] V. Shoup, *Lower bounds for discrete logarithms and related problems*, in Proc. Eurocrypt '97, pp. 256–266, 1997.

[33] D. R. Stinson, *Cryptography in theory and practice*, CRC Press, 1995.

[34] D.C. Terr, *A modification of Shanks' baby-step giant-step algorithm*, Math. of Computation, 69: pp. 767–773, 2000.

[35] E. Teske, *Square-root algorithms for the discrete logarithm problem (a survey)*, Technical Report, University of Waterloo, 2001.

[36] E. Teske, *Speeding up Pollard's rho method for computing discrete logarithms*, in Algorithmic Number Theory Seminar ANTS-III, volume 1423 of Lecture Notes in Computer Science, pp. 541–554, Springer-Verlag, 1998.

[37] E. Teske, *On random walks for Pollard's rho method*, Mathematics of Computation, (to appear in print), 2001.

[38] E. Teske, *Computing discrete logarithms with the parallelized kangaroo method*, Research Report CORR 01-01, Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Ontario, Canada, 2001. 18 pages.

[39] P. K. S. Wah and M. Z. Wang, *Realization and application of the Massey-Omura lock*, pp. 175–182, in Proc. Intern. Zurich Seminar, March 6–8, 1984.

[40] A. E. Western and J. C. P. Miller, *Tables of Indices and Primitive Roots*, Royal Society Mathematical Tables, vol. 9, Cambridge Univ. Press, 1968.

Received May 6, 2003

Ennio Cortellini,

MET Department,
University of Teramo, Italy
E–mail: *ennio.cortellini@inwind.it*