The Discrete Logarithm Problem in Matrix Groups

David Freeman
Computer Science 276
dfreeman@math.berkeley.edu

May 19, 2004

Abstract

We consider the discrete logarithm problem in the group of $n \times n$ invertible matrices over a finite field. We show that the problem can be reduced to finding at most n discrete logarithms in extension fields of degree at most n. We propose a variant of the Diffie-Hellman key exchange protocol that cannot be broken using our reduction algorithms.

1 Introduction

The discrete logarithm problem has been intertwined with cryptography ever since the invention of public-key cryptography by Diffie and Hellman in 1976 [5]. Under the Diffie-Hellman key-exchange protocol, if Alice and Bob want to arrive at a shared secret, they first (publicly) agree on a finite abelian group G. Alice chooses $a \in G$ and $0 \leq x < |G|$, and sends the pair (a, a^x) to Bob. Bob chooses $0 \leq y < |G|$ and sends (a^y) to Alice. The secret key is a^{xy}, and if the group G is well-chosen then the system is secure: it is hard to compute a^{xy} from the triple (a, a^x, a^y) that an eavesdropper could obtain by listening in on the channel.

One necessary condition for G to be "well-chosen" is that the discrete logarithm in G is hard. Namely, an eavesdropper cannot easily compute x from the pair (a, a^x). This condition is sufficient in practice, but it is not known in general whether the discrete logarithm being hard is equivalent to the system being secure.

The original Diffie-Hellman protocol specifies that G be the multiplicative group of a finite field of prime order. In the years since its invention, the protocol has been extended to other abelian groups, including the multiplicative group of any finite field, the group of units of $\mathbb{Z}/n\mathbb{Z}$ [14], class groups of real and imaginary quadratic number fields [22, 2], the group of points on an elliptic curve over a finite field [11, 16], and the jacobian of a hyperelliptic curve over a finite field [12]. The security of these systems has been studied extensively. The most general method of attack is the "index calculus" algorithm [7], which solves the discrete logarithm problem in subexponential time in many groups. In the case of elliptic curves, however, there is no known index calculus attack (indeed, Silverman and Suzuki [24] provide theoretical and experimental evidence that one does not exist), and the best known algorithms run in exponential time.[1]

Whereas many different abelian groups have been proposed for public-key cryptosystems, a survey of the literature reveals very little discussion of non-abelian groups. In 2000 Ko et al. [10] introduced a key-agreement protocol based on the difficulty of solving the conjugacy problem in braid groups, and there has been a good deal of recent research in this subject. In 2001 Paeng et al. [20] introduced a Diffie-Hellman type protocol using inner automorphism groups; Paeng [19] subsequently showed that breaking these systems reduced to the discrete

[1] Running times are given in terms of the logarithm of the order of the group, which is approximately the number of bits needed to represent an element.

logarithm problem in certain matrix groups. More recently, Grigoriev and Ponomarenko [9] have constructed cryptosystems using solvable groups, a large class of non-abelian groups.[2]

In this paper, we consider the discrete logarithm problem in matrix groups over a finite field. In Section 2 we show that the discrete logarithm problem in these groups is harder than previously supposed, and in Section 3 we demonstrate an algorithm for reducing the discrete logarithm problem in $n \times n$ matrix groups to at most n discrete logarithms in finite fields. We analyze this algorithm in Section 4, showing that if we use the best known algorithms for computing discrete logarithms in finite fields, our algorithm has running time no worse than a constant times the running time for the finite field algorithm. We conclude that there is no cryptographic advantage in implementing the basic Diffie-Hellman protocol with matrix groups.

In the final part of the paper, we propose a modified key-exchange protocol that makes use of the additional structure of non-abelian groups, and we analyze its implementation in matrix groups. We find that the algorithms of Section 3 cannot be easily adapted to attack this protocol, and we conjecture that this protocol is more secure than the basic Diffie-Hellman protocol.

[Update (July. 2005): Most of the results in this paper also appear in [15]. When I was doing my research I was not aware this paper existed, so while I discovered all of the results on my own, I wish to give credit to Menezes and Wu for finding them first.]

2 A First Look at Matrix Groups

The linear groups, or matrix groups, are one of the most widely studied classes of non-abelian groups. A matrix group may be defined over any ring R; we will consider only matrix groups defined over a finite field $\mathbb{F}_q$ of q elements, where $q = p^d$ is a prime power. (See [25] for a cryptosystem that uses matrices defined over the integers.)

The matrix group we consider foremost is $\mathrm{GL}_n(\mathbb{F}_q)$, the group of $n \times n$ matrices with entries in $\mathbb{F}_q$ and nonzero determinant. This group contains all invertible $n \times n$ matrices, so any $n \times n$ matrix group must be a subgroup of $\mathrm{GL}_n(\mathbb{F}_q)$. Thus to solve the discrete logarithm problem in any matrix group, it suffices to solve the problem in $\mathrm{GL}_n(\mathbb{F}_q)$.

In his analysis [19] of the security of the Paeng et al. cryptosystem [20], Paeng states:

> Most familiar non-abelian groups are linear groups. The DLP [Discrete Logarithm Problem] on these groups can be reduced to the DLP in an extension field if we apply the Jordan decomposition theorem. So there exist subexponential time algorithms to solve the DLP in linear groups.

In the remainder of the paper, Paeng shows how an attack on the cryptosystems in [20] and [21] can be reduced to solving the discrete logarithm problem in certain matrix groups, and from the above statement he concludes that there exist subexponential time algorithms to break these systems.

However, Paeng's statement above is quite vague. First of all, Paeng does not state what parameters the algorithms he proposes are subexponential with respect to. We will assume that for a matrix subgroup of $\mathrm{GL}_n(\mathbb{F}_q)$, the parameters he has in mind are $\log q$ and n. More problematic, however, is the fact that Paeng does not state what extension field he has in mind. Since the Jordan normal form of a matrix A is usually defined only over a field containing all of the eigenvalues of A, we conjecture that Paeng's reduction involves computing discrete logarithms in this field. We then infer that the algorithms Paeng mentions are roughly of the following form.

Algorithm 2.1. *Let $\mathbb{F}_q$ be the finite field of q elements. Let $A \in \mathrm{GL}_n(\mathbb{F}_q)$, let $0 \leq x < |A|$, and let $B = A^x$. The following algorithm computes x given B and A.*

1. *Determine a field extension K of $\mathbb{F}_q$ in which the characteristic polynomial of A splits completely.*

[2] Unfortunately, the paper [9] is in Russian, and I have not been able to translate it.

2. *Find a matrix* $P \in \mathrm{GL}_n(K)$ *such that* $P^{-1}AP$ *is diagonal, with diagonal entries* $\lambda_1, \ldots, \lambda_n$.
3. *Let* $\mu_1, \ldots, \mu_n$ *be the diagonal entries of* $P^{-1}BP$ *(a diagonal matrix). Compute* $x_i \leftarrow \log_{\lambda_i} \mu_i$ *for each* i.
4. *Use the Chinese Remainder Theorem (as in Lemma 3.1 below) to compute* x *such that* $x \equiv x_i \pmod{|\lambda_i|}$ *for each* i.

Note that step (4) is necessary because computing the discrete logarithm $\log_a b$ only gives a result modulo the order of a, and in general the eigenvalues of A may have different orders.

The best known algorithms for finding discrete logarithms in a finite field have running times of the form

$$L(x, c, \alpha) = e^{c(\log x)^{\alpha}(\log\log x)^{1-\alpha}},$$

where x is the size of the field K and c and α are positive constants with $0 < \alpha < 1$. If $x = q^d$, then these algorithms are subexponential in d and $\log q$. Thus for Algorithm 2.1 to be subexponential in n and $\log q$, the degree $[K : \mathbb{F}_q]$ must be a function of n that grows more slowly than $n^{1/\alpha}$. For the field K in Algorithm 2.1, this is not true. We show this by relating the degree of the extension $K/\mathbb{F}_q$ to the order of an element of the permutation group S_n.

Proposition 2.2. *Let* $G(n)$ *denote the maximum order of a permutation of* n *elements. Then there exists an element* $A \in \mathrm{GL}_n(\mathbb{F}_q)$ *such that the splitting field of the characteristic polynomial* $f_A(x)$ *has degree* $G(n)$ *over* $\mathbb{F}_q$.

Proof. Suppose $f_A(x)$ has irreducible factors $g_1, \ldots, g_r$ of degrees $d_1, \ldots, d_r$. The splitting field of each g_i over $\mathbb{F}_q$ is $\mathbb{F}_{q^{d_i}}$, and thus the splitting field of f_A is the compositum of the splitting fields, $\mathbb{F}_{q^{d_1}} \cdots \mathbb{F}_{q^{d_r}}$. This field is isomorphic to $\mathbb{F}_{q^d}$, where $d = \mathrm{lcm}(d_1, \ldots, d_r)$. Thus the maximum value of d is

$$\max\{\mathrm{lcm}(d_1, \ldots, d_r) : d_i \in \mathbb{Z}_{>0}, d_1 + \cdots + d_r = n\}. \tag{1}$$

Since the order of a permutation $\tau \in S_n$ whose cycle decomposition has lengths $\tau_1, \ldots, \tau_r$ is $\mathrm{lcm}(\tau_1, \ldots, \tau_r)$, the quantity (1) is $G(n)$.

To construct an A that achieves this maximum, first find $d_1, \ldots, d_r$ summing to n such that $\mathrm{lcm}(d_1, \ldots, d_r)$ is maximized. Then choose monic irreducible polynomials $g_i \in \mathbb{F}_q[x]$ of degree d_i. (These always exist; see e.g [6, §14.3].) For each i, write $g_i(x) = a_{i0} + a_{i1}x + \cdots + a_{i(d_i-1)}x^{d_i-1} + x^{d_i}$, and let

$$A_i = \begin{pmatrix} 0 & 0 & \cdots & 0 & -a_{i0} \\ 1 & 0 & \cdots & 0 & -a_{i1} \\ 0 & 1 & \cdots & 0 & -a_{i2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & -a_{i(d-1)} \end{pmatrix}.$$

Then A_i has characteristic polynomial $g_i(x)$. If we let A be the block diagonal matrix with the A_i in the blocks, then A has characteristic polynomial $f_A(x) = g_1(x) \cdots g_r(x)$, and the splitting field of f_A is $\mathbb{F}_{q^{G(n)}}$. □

Note that the element A constructed above does not, in general, have maximal *order* in $\mathrm{GL}_n(\mathbb{F}_q)$, but only a maximal *splitting field*. This A has order dividing $(q^{d_1} - 1) \cdots (q^{d_r} - 1)$, while a matrix with an irreducible characteristic polynomial whose eigenvalues are generators of $\mathbb{F}_{q^n}^{\times}$ has order $q^n - 1$, which is larger.

The number $G(n)$ is obtained by choosing powers of prime numbers whose sum is less than n and whose product is maximal. W. Miller [17] shows that as n becomes large, this quantity can be approximated by the product of the first k prime numbers, where k is chosen so that the sum of the first k primes is at most n, and adding the $(k+1)$st prime puts the sum over n. He then uses the Prime Number Theorem to obtain an asymptotic estimate for this quantity. This estimate was first proved by Landau in 1903.

Theorem 2.3 (Landau; Cf. [17]). *Let $G(n)$ be as above. Then*

$$\lim_{n\to\infty} \frac{G(n)}{e^{\sqrt{n \log n}}} = 1.$$

With Proposition 2.2 and this estimate, we see that the running time of Algorithm 2.1 is in fact *super*exponential in n.

Corollary 2.4. *Suppose computing one discrete logarithm in a finite field of size x takes time $L(x, c, \alpha)$ for some $c \geq 1$ and $0 < \alpha < 1$. Then there exists an $A \in \mathrm{GL}_n(\mathbb{F}_q)$ for which Algorithm 2.1 runs in time*

$$e^{c(1+o(1))e^{\alpha\sqrt{n \log n}}(n \log n)^{(1-\alpha)/2}(\log q)^{\alpha}(\log\log q)^{1-\alpha}}.$$

3 A Piecewise Approach

The downfall of Algorithm 2.1 is that it tries to work with all of the eigenvalues of A at once, which requires an enormous extension field of $\mathbb{F}_q$. However, since each eigenvalue of A lives in an extension field of degree at most n, we may expect better success if we work with the irreducible factors g_i one at a time. The polynomial g_i splits completely in the field $K = \mathbb{F}_q[x]/(g_i(x)) \cong \mathbb{F}_{q^{d_i}}$. Let α be a root of g_i in K; we may then compute an eigenvector $w \in K^n$ with eigenvalue λ (at least one exists, though not necessarily more). We then choose $v_1, \ldots, v_{n-1}$ such that $\{w, v_1, \ldots, v_{n-1}\}$ is a basis for K^n, and let

$$P = \begin{pmatrix} | & | & & | \\ w & v_1 & \cdots & v_{n-1} \\ | & | & & | \end{pmatrix},$$

then $P^{-1}AP \in \mathrm{GL}_n(K)$ is a matrix with λ in the upper left corner and all zeroes in the remainder of the left column, and therefore $P^{-1}A^xP = (P^{-1}AP)^x$ has λ^x in the upper left corner. Taking a discrete logarithm in K gives $x \pmod{|\lambda|}$, where $|\lambda|$ is the order of λ in $K^\times$. Repeating this process for each factor g_i is enough to determine x completely in the case that A is diagonalizable.

Lemma 3.1. *Let $A \in \mathrm{GL}_n(\mathbb{F}_q)$. Suppose A is diagonalizable over some extension field of $\mathbb{F}_q$, and suppose the characteristic polynomial of A factors into irreducibles as $g_1(t) \cdots g_r(t)$ in $\mathbb{F}_q[t]$, where the irreducible factor g_i has degree d_i. Choose a root $\alpha_i \in \mathbb{F}_q^{d_i}$ of each g_i. Let x be any integer, and let x_i be the residue of x modulo $|\alpha_i|$. Then there is a polynomial-time algorithm that takes inputs $\{x_i\}$ and $\{\alpha_i\}$, and outputs the unique integer y with $0 \leq y \leq |A|$ such that $y \equiv x \pmod{|A|}$.*

Proof. Let $M = \mathrm{lcm}(|\alpha_1|, \ldots, |\alpha_r|)$. We claim that the x_i determine a unique y congruent to x modulo M, and that $|A| = M$. To see the former assertion, factor M as a product of primes

$$M = p_1^{e_1} \cdots p_l^{e_l}.$$

Since M is the least common multiple of the $|\alpha_i|$, for each j there is some i such that $p_j^{e_j}$ divides $|\alpha_i|$. Let y_j be the residue of x_i modulo $p_j^{e_j}$; since the x_i are all residues of the same integer x, we get the same y_j regardless of which x_i we choose. The y_j are thus the residues of x modulo a set of relatively prime numbers, so we may apply the Chinese Remainder Theorem to the y_j to compute (in polynomial time) a unique y that is congruent to x modulo M.

To see that $|A| = M$, note that since A is diagonalizable $|A|$ is the least common multiple of the orders of its eigenvalues. The eigenvalues are all the roots of the g_i; however, g_i splits completely in $\mathbb{F}_q(\alpha_i)$ and the Galois group $\mathrm{Gal}(\mathbb{F}_q(\alpha_i)/\mathbb{F}_q)$ acts transitively on the roots of g_i, so every root of g_i has the same order as α_i. Thus $|A| = \mathrm{lcm}(|\alpha_1|, \ldots, |\alpha_r|) = M$. □

What if A is not diagonalizable? In this case, the Jordan normal form of A in some extension field has a block D of the form

$$\begin{pmatrix} \lambda & 1 & & \\ & \lambda & \ddots & \\ & & \ddots & 1 \\ & & & \lambda \end{pmatrix},$$

and the corresponding block of D^x is

$$\begin{pmatrix} \lambda^x & x\lambda^{x-1} & & \\ & \lambda^x & \ddots & \\ & & \ddots & x\lambda^{x-1} \\ & & & \lambda^x \end{pmatrix}.$$

Since $\mathrm{char}(\mathbb{F}_q) = p$, the off-diagonal elements of D^x are zero if and only if $x \equiv 0 \pmod{p}$. Since λ is in a field extension of $\mathbb{F}_p$, its order divides $p^d - 1$ for some d and is thus relatively prime to p, so the order of D is $p\,|\lambda|$. The order of a non-diagonalizable matrix A is thus p times the order of the diagonal matrix with the same eigenvalues. To determine $x \pmod{|A|}$, by this reasoning and Lemma 3.1 it suffices to determine $x \pmod{\mathrm{lcm}(|\alpha_1|, \ldots, |\alpha_r|)}$ and $x \pmod{p}$.

If we can find P such that the upper left corner of $P^{-1}AP$ looks like

$$\begin{pmatrix} \lambda & 1 \\ 0 & \lambda \end{pmatrix} \tag{2}$$

and the first two columns are all zero below the diagonal, then we can easily compute $x \pmod{p}$ from A and A^x. To find such a P, we make use of the two different canonical forms of a matrix, the rational canonical form and the Jordan normal form. Recall that rational canonical form splits A into blocks corresponding to the "invariant factor" polynomials $a_1(x)|a_2(x)|\cdots|a_m(x)$, with a basis for each block given by $\{v, Av, A^2v, \ldots, A^{d_i-1}v\}$ for some v (where $d_i = \deg(a_i)$). The Jordan normal form, on the other hand, splits A into blocks corresponding to the relatively prime divisors of each $a_i(x)$, with a basis for each block given by $\{w, (A-\lambda_j I)w, (A-\lambda_j I)^2 w, \ldots, (A-\lambda_j I)^{e_{ij}-1}w\}$ for some w (where e_{ij} is the multiplicity of λ_j in a_i). (For more information, see [6, Ch. 12].)

Algorithm 3.2. *Let $A \in \mathrm{GL}_n(K)$. Suppose that A is not diagonalizable (over $\bar{K}$) and that K contains an eigenvalue λ for which the dimension of the eigenspace is less than the multiplicity of λ. The following algorithm takes input A and outputs P such that the upper left corner of $P^{-1}AP$ has the form (2) and the two leftmost columns are zero below the diagonal.*

1. *Convert A to rational canonical form over K, keeping track of the invariant factor decomposition $a_1(x), \ldots, a_m(x)$. Let d be the degree of $a_m(x)$.*
2. *Find $v \in K^n$ such that $\{v, Av, A^2v, \ldots A^{d-1}v\}$ is a basis for the subspace of K^n corresponding to the invariant factor $a_m(x)$.*
3. *Let $p(x) = a_m(x)/(x-\lambda)^2$. Let $v_1 = (A-\lambda I)p(A)v$ and $v_2 = p(A)v$.*
4. *Complete $\{v_1, v_2\}$ to a basis $\{v_1, v_2, \ldots, v_n\}$ of K^n. Let*

$$P = \begin{pmatrix} | & | & & | \\ v_1 & v_2 & \cdots & v_n \\ | & | & & | \end{pmatrix}.$$

Proof. First note that our hypotheses imply that the minimal polynomial $a_m(x)$ has the root λ with multiplicity at least 2, so the $p(x)$ in step (3) is well-defined as a polynomial in $K[x]$. Since $p(x)$ has degree $d-2$, by the hypothesis on v in step (2) v_1 and v_2 are nonzero. It now suffices to show that $Av_1 = \lambda v_1$ and $Av_2 = v_1 + \lambda v_2$. The second statement is clear from the definition $v_1 = (A-\lambda I)v_2$. The first statement is true since $(A-\lambda I)v_1 = a_m(A)v$, and $a_m(A) = 0$ since $a_m(x)$ is the minimal polynomial of A. □

We now have an algorithm that reduces the discrete logarithm in $\mathrm{GL}_n(\mathbb{F}_q)$ to at most n discrete logarithms in fields of size at most q^n.

Algorithm 3.3. *Let $A \in \mathrm{GL}_n(\mathbb{F}_q)$, let $0 \leq x < |A|$, and let $B = A^x$. The following algorithm takes inputs (A, B) and computes x. The algorithm runs in time*

$$p(\log q, n) + nD(n), \tag{3}$$

where p is a polynomial in two variables and $D(n)$ is the maximum amount of time needed to compute one discrete logarithm in $\mathbb{F}_{q^d}$ for $d \leq n$.

1. *Compute the characteristic polynomial $f_A(x)$ and factor it into irreducibles $g_1, \ldots, g_r \in \mathbb{F}_q[x]$ of degrees $d_1, \ldots, d_r$. Set $i \leftarrow 1$.*
2. *Let α_i be a root of g_i in $K = F_{q^{d_i}}$. Find $P \in \mathrm{GL}_n(K)$ such that $P^{-1}AP$ has α_i in its upper left corner and zeroes in the rest of the left column.*
3. *Let β be the upper-left entry of $P^{-1}BP$. Compute $x_i \leftarrow \log_{\alpha_i} \beta$ in K.*
4. *Factor $f_A(x)$ in $K[x]$, and compute bases for the eigenspaces of A corresponding to all eigenvalues in K (i.e. all linear factors of f_A). If the dimensions of the eigenspaces are equal to the multiplicities of the eigenvalues, go to step (6). Otherwise:*
5. *Let λ be an eigenvalue in K whose multiplicity is greater than the dimension of its eigenspace. Use Algorithm 3.2 to compute Q such that $Q^{-1}AQ$ has upper-left corner in the form (2), and zeroes below the diagonal in the two left columns. Let the upper-left corner of $Q^{-1}BQ$ be*

$$\begin{pmatrix} \mu & \nu \\ 0 & \mu \end{pmatrix}.$$

   Let $x_0 \leftarrow \lambda\nu\mu^{-1}$. Note that we may view x_0 as an element of $\mathbb{F}_p \subset K$.
6. *If $i = d$, go to step (7). Otherwise, let $i \leftarrow i + 1$ and go to step (2).*
7. *Use the Chinese Remainder Theorem as in Lemma 3.1 to compute y such that $y \equiv x_i \pmod{|\alpha_i|}$ for each i.*
8. *If x_0 is uninitialized, let $x \leftarrow y$. Otherwise, use the Chinese Remainder Theorem to compute x such that $x \equiv x_0 \pmod{p}$ and $x \equiv y \pmod{\mathrm{lcm}(|\alpha_1|, \ldots, |\alpha_i|)}$.*

Proof. If A is diagonalizable, by Lemma 3.1 step (5) never executes, and the algorithm computes x modulo $\mathrm{lcm}(|\alpha_1|, \ldots, |\alpha_r|) = |A|$. If A is not diagonalizable, the order of A increases by a factor of p. Since there is some eigenspace with no basis of eigenvectors, step (5) must execute at some point. It computes $x \pmod{p}$, and thus step (8) computes x modulo $|A|$.

There exist polynomial-time algorithms (in n and $\log |K|$) for multiplying or inverting elements of $\mathrm{GL}_n(K)$, computing eigenvectors, and computing characteristic polynomials (see [3, §2.2]). Furthermore, there exists a probabilistic polynomial-time algorithm to factor polynomials in $K[x]$ (see [3, §3.4]). Solving a system of congruences via the Chinese Remainder Theorem also takes polynomial time. Thus all steps except (3) can be executed in polynomial time. The algorithm executes step (3) at most n times, and all of the discrete logarithms are taken in fields of degree at most n. Thus the discrete logarithm steps take at most n times the maximum time for discrete logs in $\mathbb{F}_{q^d}$ as d ranges from 1 to n. □

In practice, the slowest discrete logarithm will almost certainly be in $\mathbb{F}_{q^n}$. If we have an estimate for the speed of the discrete logarithm in $\mathbb{F}_{q^d}$ as a function of d, then the term $nD(n)$ in (3) can certainly be improved. We address this issue in greater depth in Section 4 below.

Note that Algorithm 3.3 does not allow us to choose an $\mathbb{F}_q$-basis for the field $\mathbb{F}_{q^d}$ in which a given eigenvalue λ lies. Instead, the basis is determined by the polynomial $g(x)$ of which λ is a root. One may worry then that a discrete logarithm algorithm whose speed depends on the properties of a particular basis will not be applicable in Step (3). (For example, we may wish to use an "optimal normal basis" as in [18].) However, Neal Zierler [26] has demonstrated a

polynomial-time algorithm to express roots of a degree-n irreducible polynomial $g(x)$ over $\mathbb{F}_q$ in terms of a basis associated with another given polynomial $h(x)$. Thus changing basis before computing the discrete logarithm adds at most a polynomial number of steps to the algorithm.

4 Security Analysis

We wish to determine if the Diffie-Hellman key exchange in $\mathrm{GL}_n(\mathbb{F}_q)$ offers any advantage over the same protocol implemented over a finite field that allows approximately the same private key size (i.e. exponent). Let $A \in \mathrm{GL}_n(\mathbb{F}_q)$, and suppose as usual that the characteristic polynomial of A has irreducible factors $g_1, \ldots, g_r$ of degrees $d_1, \ldots, d_r$. Then the order of A divides

$$\left(q^{d_1} - 1\right) \cdots \left(q^{d_r} - 1\right) \leq q^n - 1.$$

The order of A equals $q^n - 1$ if and only if the characteristic polynomial is irreducible and primitive (i.e. its roots are generators of $\mathbb{F}_{q^n}^\times$). Since elements in $\mathbb{F}_{q^n}$ have order dividing $q^n - 1$, $\mathbb{F}_{q^n}$ is an appropriate finite field for comparison.

We express $\mathbb{F}_{q^n}$ as $\mathbb{F}_q[x]/(f(x))$ for some degree-n irreducible polynomial $f(x)$. The simplest algorithm for multiplying two elements of $\mathbb{F}_{q^n}$ is to express both elements as polynomials modulo $f(x)$, multiply them in $\mathbb{F}_q[x]$, and reduce modulo $f(x)$. This approach requires n^2 multiplications and n^2 additions in $\mathbb{F}_q$ for the multiplication step, and an additional n^2 multiplications and n^2 additions to reduce the result modulo $f(x)$ (assuming we have precomputed a table of values of $x^n, x^{n+1}, \ldots, x^{2n-2}$ modulo $f(x)$).

The simplest algorithm for matrix multiplication (from basic linear algebra) requires n field multiplications and n field additions for each element, giving n^3 multiplications and n^3 additions in total. There exist fast multiplication techniques that reduce the exponent 3 to $2+\delta$ for various $\delta > 0$; these techniques can be adapted to reduce the exponent of 2 in the $\mathbb{F}_{q^n}$ multiplication. Thus we can expect exponentiation in $\mathrm{GL}_n(\mathbb{F}_q)$ to take roughly n times as long as exponentiation in $\mathbb{F}_{q^n}$. (We assume throughout that we are using the same exponentiation algorithm in both groups.)

Another factor we must take into account is the size of the public key, i.e. the group element. An element of $\mathbb{F}_{q^n}$ can be represented in $n \log q$ bits, while an element of $\mathrm{GL}_n(\mathbb{F}_q)$ requires $n^2 \log q$ bits, a factor of n larger.

What about computing the discrete logarithm? The best known algorithms for computing discrete logarithms in a finite field K have running times of the form

$$L(x, c, \alpha) = e^{c(\log x)^\alpha (\log\log x)^{1-\alpha}},$$

where x is the size of the field K and c and α are constants. If $0 < \alpha < 1$, this function is said to be *subexponential* in $\log x$, since it grows faster than any polynomial but slower than an exponential function. (If $\alpha = 1$ the function is exponential; if $\alpha = 0$ it is polynomial.) The best general algorithm with rigorously proven running time is the "index calculus" method, which runs in time $L(x, \sqrt{2} + o(1), 1/2)$ [7]. With the use of some special factoring techniques such as the number field sieve [13] and the function field sieve [1], for prime or characteristic 2 fields this running time has been reduced to $L(x, c + o(1), 1/3)$, with $c \approx 2$ [4, 8, 23]. However, these faster running times are calculated heuristically and have not been rigorously proven.

Given these running time estimates, we see that step (3) in Algorithm 3.3 takes time at most $nL(q^n, c, \alpha)$ for some constants $c \geq 1$ and $\alpha \geq 1/3$. The factor of n can in fact be reduced to a constant; to show this we will need the following algebraic lemma.

Lemma 4.1. *Suppose $\alpha, c \in \mathbb{R}$ such that $1/3 \leq \alpha < 1$ and $c \geq 1$. Let $\{x_1, \ldots, x_r, y\}$ be a set of positive real numbers such that $x_i \geq \log 2$ and $\sum x_i = y$. Then*

$$\sum_{i=1}^{r} L\left(e^{x_i}, c, \alpha\right) \leq 12\, L\left(e^y, c, \alpha\right).$$

Proof. If $r = 1$ the statement is trivial, so we assume that $r \geq 2$. Begin by reordering the x_i such that x_1 is the largest. We split the proof into two cases, depending on whether x_1 is larger or smaller than $y/2$. First suppose that $x_1 \geq y/2$; then for $i \geq 2$, $x_i \leq y/2$. Since $L(x, c, \alpha)$ is increasing in x for fixed c and α, we have

$$\sum_{i=1}^{r} L\left(e^{x_i}, c, \alpha\right) \leq L\left(e^{y}, c, \alpha\right) + (r-1)L(e^{y/2}, c, \alpha). \tag{4}$$

To bound this expression in terms of $L(e^y, c, \alpha)$, we wish to find an upper bound for the ratio

$$\frac{(r-1)L\left(e^{y/2}, c, \alpha\right)}{L\left(e^{y}, c, \alpha\right)}. \tag{5}$$

Since each $x_i \geq \log 2$ and $x_1 \geq y/2$, $(r-1) \leq y/2 \log 2$, so it suffices to maximize

$$\left(\frac{y}{2\log 2}\right)\frac{L\left(e^{y/2}, c, \alpha\right)}{L\left(e^{y}, c, \alpha\right)} = e^{c(y/2)^{\alpha}(\log(y/2))^{1-\alpha} - cy^{\alpha}(\log y)^{1-\alpha} + \log y - \log(2\log 2)}.$$

This quantity is maximized when

$$c(y/2)^{\alpha}(\log(y/2))^{1-\alpha} - cy^{\alpha}(\log y)^{1-\alpha} + \log y$$

is maximized, and

$$\begin{aligned} c(y/2)^{\alpha}(\log(y/2))^{1-\alpha} - cy^{\alpha}(\log y)^{1-\alpha} + \log y &\leq c(\log y)^{1-\alpha}\left(\left(\frac{y}{2}\right)^{\alpha} - y^{\alpha}\right) + \log y \\ &= \left(1 + c\left(\frac{1}{2^{\alpha}} - 1\right)\left(\frac{y}{\log y}\right)^{\alpha}\right)\log y \end{aligned} \tag{6}$$

The derivative of the right hand side with respect to α is

$$c\log y\left(\frac{y}{\log y}\right)^{\alpha}\left(\left(\frac{1}{2^{\alpha}} - 1\right)\log\left(\frac{y}{\log y}\right) - \frac{\log 2}{2^{\alpha}}\right),$$

which is clearly negative for $\log y > 1$. Thus for the range of α we are considering, (6) is maximized when $\alpha = 1/3$. In addition, (6) is decreasing in c, so we may assume $c = 1$.

Taking the derivative of (6) with respect to y and setting the result equal to zero gives the equation

$$c\left(1 - \frac{1}{2^{\alpha}}\right)(\alpha\log y + 1 - \alpha) = \left(\frac{\log y}{y}\right)^{\alpha}.$$

When $\log y > 1$, the left hand side is increasing in y and the right hand side is decreasing. Setting $c = 1$, $\alpha = 1/3$ and graphing the two functions with respect to y shows the intersection point is between $y = 56$ and $y = 57$. The expression (6) is thus bounded by

$$\left(1 + \left(\frac{1}{2^{1/3} - 1}\right)\left(\frac{56}{\log 56}\right)^{1/3}\right)\log 57 < 3\log 2.$$

Therefore,

$$\begin{aligned} \left(\frac{y}{2\log 2}\right)\frac{L\left(e^{y/2}, c, \alpha\right)}{L\left(e^{y}, c, \alpha\right)} &< e^{3\log 2 - \log(2\log 2)} \\ &< 6. \end{aligned}$$

We conclude that

$$(r-1)L(e^{y/2}, c, \alpha) \leq 6L(e^{y}, c, \alpha),$$

so by (4), the Lemma holds in the case $x_1 \geq y/2$.

Now suppose $x_1 \leq y/2$. Then $x_i \leq y/2$ for all i, and thus

$$\sum_{i=1}^{r} L(e^{x_i}, c, \alpha) \quad \leq \quad rL\left(e^{y/2}, c, \alpha\right).$$

Since $x_i \geq \log 2$, $r \leq y/\log 2$. Thus the maximum value of the ratio

$$\frac{rL\left(e^{y/2}, c, \alpha\right)}{L\left(e^{y}, x, \alpha\right)}$$

is twice the maximum value of ratio (5), which we calculated above to be less than 6. Thus the Lemma holds in the case $x_1 \leq y/2$. □

Proposition 4.2. *Suppose computing a discrete logarithm in a finite field of size x takes time at most $L(x, c, \alpha)$ for some $c \geq 1$ and $\alpha \geq 1/3$. Then Algorithm 3.3 computes a discrete logarithm in $\mathrm{GL}_n(\mathbb{F}_q)$ in time at most $12L(q^n, c, \alpha)$.*

Proof. Let $f_A(x)$ be the characteristic polynomial of A, and suppose it factors into irreducibles $g_1, \ldots, g_r$ of degrees $d_1, \ldots, d_r$ respectively. Algorithm 3.3 computes one discrete logarithm in $\mathbb{F}_{q^{d_i}}$ for each i. By the time estimate (3), the total time is at most a polynomial in n plus

$$\sum_{i=1}^{r} L\left(q^{d_i}, c, \alpha\right), \tag{7}$$

and since $\alpha > 0$ the polynomial is negligible. Let $x_i = d_i \log q$ and $y = n \log q$. Then by Lemma 4.1, the quantity (7) is less than $12L(q^n, c, \alpha)$. □

We conclude that if we use the best algorithms to compute discrete logs in $\mathbb{F}_{q^d}$, we may replace the term $nD(n)$ in (3) with $12L(q^n, c, \alpha)$. Thus it takes at most twelve times as long to compute discrete logs in $\mathrm{GL}_n(\mathbb{F}_q)$ as it does in $\mathbb{F}_{q^n}$. In particular, this result verifies Paeng's conclusions in [19].

Conclusion

When comparing a Diffie-Hellman key exchange in $\mathrm{GL}_n(\mathbb{F}_q)$ with a key exchange in $\mathbb{F}_{q^n}$, we find that while the private keys are the same size, the public key is a factor of n larger in $\mathrm{GL}_n(\mathbb{F}_q)$, and the public keys take roughly n times as long to compute. These tradeoffs might be acceptable if the $\mathrm{GL}_n(\mathbb{F}_q)$ system were harder to break, but our analysis shows that computing a discrete logarithm in $\mathrm{GL}_n(\mathbb{F}_q)$ takes less than 12 times as long as a discrete logarithm in $\mathbb{F}_{q^n}$. We conclude that there is no advantage to using matrices as public keys in the basic Diffie-Hellman protocol.

5 Modified Diffie-Hellman Protocol

The group originally proposed for the Diffie-Hellman protocol was the multiplicative group of $\mathbb{F}_q$, which is cyclic. Though the matrix groups G we considered above are non-abelian, once a group element A is chosen we are really only working in a cyclic subgroup of G, and we saw that the discrete logarithm problem can be reduced to a number of discrete logarithms in finite fields. We therefore would like to take advantage of the additional structure of a non-abelian group and find a way to make the discrete logarithm "harder" in some sense.

One way in which non-abelian groups have richer structure than cyclic groups is their automorphism group. The automorphism group $\mathrm{Aut}(G)$ is the set of group isomorphisms from G to itself. The set of *inner automorphisms* $\mathrm{Inn}(G)$ is the elements of $\mathrm{Aut}(G)$ given by conjugation:

$\phi_a(g) = a^{-1}ga$. The subgroup $\mathrm{Inn}(G)$ is normal in $\mathrm{Aut}(G)$, and we denote $\mathrm{Aut}(G)/\mathrm{Inn}(G)$ by $\mathrm{Out}(G)$, the group of *outer automorphisms* of G.

An abelian group has trivial inner automorphism group, and any outer automorphism of a cyclic group is given by $(a \mapsto a^d)$ for some d relatively prime to $|G|$. Thus acting an automorphism on a cyclic group used in the Diffie-Hellman protocol is equivalent to multiplying the public and private keys by some factor d, which does nothing to increase security. A general automorphism of a non-abelian group (or even of a non-cyclic abelian group) does not simply exponentiate all group elements, so perhaps we can act use these automorphisms to increase security. Our approach in this section will be to incorporate automorphisms of G into the Diffie-Hellman Protocol to try to disguise the relationship between the element A and its exponentiation A^x.

Protocol 5.1 (Modified Diffie-Hellman Key Exchange).

1. *Alice and Bob publicly agree on a group G and a subset $\Gamma \subset \mathrm{Aut}(G)$ satisfying $\sigma\tau = \tau\sigma$ for all $\sigma, \tau \in \Gamma$.*
2. *Alice chooses $a \in G$, $\sigma \in \Gamma$, and $0 \leq x < |a|$, and sends the pair $(a, \sigma(a^x))$ to Bob.*
3. *Bob chooses $\tau \in \Gamma$ and $0 \leq y < |a|$ and sends $\tau(a^y)$ to Alice.*

Suppose only Alice knows (σ, x) and only Bob knows (τ, y). Then Alice and Bob can share the secret key $\sigma\tau(a^{xy})$.

Proof. Let $b = \sigma(a^x)$ and $c = \tau(a^y)$. Bob computes $\tau(b^y)$, and since τ is an automorphism commuting with σ,

$$\tau(b^y) = \tau((\sigma(a^x))^y) = \tau(\sigma(a^{xy})) = \sigma\tau(a^{xy}).$$

Similarly, Alice computes $\sigma(c^x) = \sigma\tau(a^{xy})$. □

If $\sigma = \tau = 1 \in \mathrm{Aut}(G)$, this protocol reduces to the ordinary Diffie-Hellman scheme. Thus this protocol is at least as secure as the ordinary Diffie-Hellman protocol; i.e. any algorithm that breaks Protocol 5.1 for Γ containing the trivial automorphism can be used as an oracle to break Diffie-Hellman.

Application to Matrix Groups

We consider implementing Protocol 5.1 using inner automorphisms of matrix groups $\mathrm{GL}_n(\mathbb{F}_q)$. We must first choose a commuting subset Γ of $\mathrm{Aut}(\mathrm{GL}_n(\mathbb{F}_q))$; if we are considering only inner automorphisms this means choosing a set of commuting elements of $\mathrm{GL}_n(\mathbb{F}_q)$. One obvious choice is the subgroup Δ of diagonal matrices. Given any matrix $P \in \mathrm{GL}_n(\mathbb{F}_q)$, the set $P^{-1}\Delta P$ is also an abelian subgroup of $\mathrm{GL}_n(\mathbb{F}_q)$, so we may obtain in this manner a large class of possible choices for Γ.

Given two matrices which are known to be conjugate, the problem of finding a matrix that conjugates one to the other is simply a matter of solving a system of linear equations (see [20]), so to attack this implementation of Protocol 5.1, it suffices to recover one of the private keys x or y.

If choose Γ to be the group of diagonal matrices, then Algorithm 3.3 cannot be used to find the secret exponent x. Algorithm 3.3 relies on the fact that A and A^x are simultaneously diagonalizable, so when we (partially) diagonalize A and read the eigenvalues $\{\alpha_1, \alpha_2, \ldots\}$ off the diagonal of $P^{-1}AP$, the diagonal of $P^{-1}A^xP$ contains $\{\alpha_1^x, \alpha_2^x, \ldots\}$, *in the same order*. In the new protocol, if we construct P to diagonalize A and Q to diagonalize $\sigma(A^x)$, then we obtain the sets of eigenvalues $\{\alpha_1, \ldots, \alpha_n\}$ and $\{\beta_1, \ldots, \beta_n\}$ respectively, but there is no obvious way of determining which β_j is equal to α_i^x.

In the case where the characteristic polynomial of A is irreducible, we can use discrete logarithms in $\mathbb{F}_q^n$ to narrow down the possibilities for x to n different values.

Algorithm 5.2. *Suppose $A \in \mathrm{GL}_n(\mathbb{F}_q)$ has an irreducible characteristic polynomial. Let $D \in \mathrm{GL}_n(\mathbb{F}_q)$, let $0 \leq x < |A|$, and let $B = D^{-1}A^xD$. The following algorithm takes inputs (A, B) and outputs a set of n integers, exactly one of which is equal to x.*

1. *Compute an eigenvalue $\alpha \in \mathbb{F}_{q^n}$ of A.*
2. *Compute an eigenvalue $\beta \in \mathbb{F}_{q^n}$ of B.*
3. *Compute $y \leftarrow \log_\alpha \beta$.*
4. *Output $\{y, qy, q^2y, \ldots, q^{n-1}y\}$ as a set of integers modulo $|A|$.*

Proof. The eigenvalues of A are $\{\alpha, \alpha^q, \ldots, \alpha^{q^{n-1}}\}$, while those of B are $\{\alpha^x, \alpha^{qx}, \ldots, \alpha^{q^{n-1}x}\}$. Thus $\beta = \alpha^{q^i}$ for some $0 \leq i < n$, and $y = \log_\alpha \beta = q^i x$. The logarithm y is determined modulo $|\alpha|$, and $|\alpha| = |A|$ must divide $q^n - 1$. Since $q^n \equiv 1$ modulo any divisor of $q^n - 1$, the set $\{y, qy, \ldots, q^{n-1}y\} \pmod{|A|}$ is equal to the set $\{x, qx, \ldots, q^{n-1}x\} \pmod{|A|}$. □

If the characteristic polynomial f_A is reducible, the situation is much more dire, even if we assume that we can determine a correspondence between the irreducible factors of f_A and those of the characteristic polynomial f_B. Suppose $g_1, \ldots, g_r$ are the irreducible factors of f_A, of degrees $d_1, \ldots, d_r$ respectively. For each i we perform Algorithm 5.2, choosing α_i to be a root of g_i and β_i a root of the corresponding factor of f_B. We then obtain a set of d_i values for x modulo $|\alpha_i|$. Choosing one value from each set and applying Lemma 3.1 allows us to compute a guess for x, but only one of the $\prod d_i$ values will be correct. By Proposition 2.2 and Theorem 2.3, for large n we can choose A such that the number of guesses computed in this manner is approximately $e^{\sqrt{n \log n}}$, so even if we have a discrete logarithm oracle the expected running time is still subexponential.

It gets worse from here. If f_A has two factors of the same degree, there is not necessarily any way to tell which factors of f_B these correspond to. And even if we narrow down the possibilities for the exponent x to a set of some reasonable size, there is no easy way to tell which member of the set is x if all the data we have are A and some conjugate of A^x.

This analysis leads us to hope that Protocol 5.1, implemented with $G = \mathrm{GL}_n(\mathbb{F}_q)$ and Γ a commuting set of inner automorphisms of G, is more difficult to break than the ordinary Diffie-Hellman scheme over $\mathrm{GL}_n \mathbb{F}_q$. We formulate this hope as a challenge for the reader:

Challenge 1. *Let A, D, x, and B be as in Algorithm 5.2. Let S be the output of Algorithm 5.2 performed on the pair (A, B). Find an algorithm that determines which element of S is equal to $x \pmod{|A|}$.*

Finally, we note that since any irreducible polynomial in $\mathbb{F}_q[x]$ is separable, the hypotheses on A of Algorithm 5.2 imply that A is diagonalizable. What happens if A is not diagonalizable? Since we can tell which factors of f_A have multiple roots, it will be easier to determine the correspondences between factors of f_A and f_B as well as the correspondences between roots of these factors. However, we saw above that the order of a non-diagonalizable matrix A is a factor of $p = \mathrm{char}(\mathbb{F}_q)$ larger than the order of a diagonalizable matrix with the same eigenvalues. The discrete logarithm algorithms in this case determine the exponent x modulo $|A|/p$, and we must compute x modulo p.

In Algorithm 3.3 we took advantage of the fact that A and A^x are simultaneously diagonalizable to compute x modulo p; this method fails in attacking Protocol 5.1. Even solving the problem for a 2×2 matrix would be useful, but we have not yet found a way to do this. We thus present another challenge:

Challenge 2. *Let $A \in \mathrm{GL}_2(\mathbb{F}_q)$, and suppose A is conjugate to $\left(\begin{smallmatrix} \lambda & 1 \\ 0 & \lambda \end{smallmatrix}\right)$ for some $\lambda \in \mathbb{F}_q$. Let $0 \leq x < |A|$, and suppose B is conjugate to A^x. Find an algorithm that takes inputs (A, B) and computes $x \pmod{p}$.*

Finding algorithms to answer either of these challenges will help reduce the task of breaking the matrix implementation of Protocol 5.1 to that of finding discrete logarithms in extension

fields of $\mathbb{F}_q$. However, there may be other ways to compute $\sigma\tau(A^{xy})$ from $(A, \sigma(A^x), \tau(A^y))$ that take advantage of the structure of the conjugation automorphism, or of the chosen set Γ of commuting automorphisms.

References

[1] Adleman, L., "The function field sieve," in *Algorithmic number theory, ANTS-I,* Ed. L. Adleman and M.-D. Huang, Springer, Berlin 1994, 108-121.

[2] Buchmann, J., and H. Williams, "A key-exchange system based on imaginary quadratic fields," *Journal of Cryptology* **1** (1988) 107-118.

[3] Cohen, H., *A Course in Computational Algebraic Number Theory,* Springer, Berlin 1996.

[4] Coppersmith, D., "Fast evaluation of logarithms in fields of characteristic two," *IEEE Transactions on Information Theory* **30** (1984) 587-594.

[5] Diffie, W., and M. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory* **22** (1976) 644-654.

[6] Dummit, D., and R. Foote, *Abstract Algebra,* 2nd ed., Prentice-Hall, Upper Saddle River, NJ 1999.

[7] Enge, A., and P. Gaudry, "A general framework for subexponential discrete logarithm algorithms." *Acta Arithmetica* **102** (2002) 83-103.

[8] Gordon, D., "Discrete logarithms in $GF(p)$ using the number field sieve," *SIAM Journal of Discrete Mathematics* **6** (1993) 124-138.

[9] Grigorev, D., and I. Ponomarenko, "On nonabelian homomorphic public-key cryptosystems" (Russian), Zap. Nauchn. Sem. S.-Peterburg. Otdel. Mat. Inst. Steklov. **293** (2002), 39-58.

[10] Ko, K., S. Lee, J. Cheon, J. Han, J-S. Kang, C. Park, "New public-key cryptosystem using braid groups," *Advances in Cryptology - CRYPTO 2000,* Ed. M. Bellare, Springer, Berlin 2000, 166-184.

[11] Koblitz, N., "Elliptic curve cryptosystems," *Mathematics of Computation* **48** (1987) 203-209.

[12] Koblitz, N., "Hyperelliptic cryptosystems," *Journal of Cryptology* **1** (1989) 139-150.

[13] Lenstra, A., and H. Lenstra, *The development of the number field sieve,* Springer, Berlin 1993.

[14] McCurley, K., "A key distribution system equivalent to factoring," *Journal of Cryptology* **1** (1988) 95-105.

[15] Menezes, A., Y-H. Wu, "The discrete logarithm problem in GL(n, q)," *Ars Combinatoria* **47** (1998), 23-32.

[16] Miller, V., "Uses of elliptic curves in cryptography," *Advances in Cryptology - CRYPTO '85,* Ed. H. Williams, Springer, Berlin 1986, 417-426.

[17] Miller, W., "The maximum order of an element of a finite symmetric group," *American Mathematical Monthly* **94** (1987), 497-506.

[18] Mullin, R., I. Onyszchuk, S. Vanstone, R. Wilson, "Optimal normal bases in GF(p^n)," *Discrete Applied Mathematics* **22** (1989) 149-161.

[19] Paeng, S-H., "On the security of cryptosystem using automorphism groups," *Information Processing Letters* **88** (2003) 293-298.

[20] Paeng, S-H., K-C. Ha, J. Kim, S. Chee, C. Park, "New public key cryptosystem using finite nonabelian groups." in *Advances in Cryptology - CRYPTO 2001,* Ed. J. Killian, Springer, Berlin 2001, 470-485.

[21] Paeng, S-H., D. Kwon, K-C. Ha, J. Kim, "Improved public-key cryptosystem using finite non abelian groups," available online at `http://eprint.iacr.org/2001/066` (2001).

[22] Scheidler, R., A. Stein, and H. Williams, "A key-exchange protocol using real quadratic fields," *Journal of Cryptology* **7** (1994), 171-199.

[23] Schirokauer, O., D. Weber, and Th. Denny, "Discrete logarithms: The effectiveness of the index calculus method," in *Algorithmic Number Theory, ANTS-II*, Ed. H. Cohen, Springer, Berlin 1996, 337-362.

[24] Silverman, J., and J. Suzuki, "Elliptic curve discrete logarithms and the index calculus," in *Advances in Cryptology - ASIACRYPT '98,* Ed. K. Ohta and D. Pei, Springer, Berlin 1998, 110-125.

[25] Yamamura, A., "Public-key cryptosystems using the modular group," in *Public Key Cryptography, PKC '98,* Ed. H. Imai and Y. Zheng, Springer, Berlin 1998, 203-216.

[26] Zierler, N., "A conversion algorithm for logarithms on $GF(2^n)$," *Journal of Pure and Applied Algebra* **4** (1974) 353-356.