Lower Bounds on Generic Algorithms in Groups

Ueli Maurer Stefan Wolf

Computer Science Department
Swiss Federal Institute of Technology (ETH Zürich)
CH-8092 Zürich, Switzerland
E-mail addresses: {maurer,wolf}@inf.ethz.ch

Abstract. In this paper we consider generic algorithms for computational problems in cyclic groups. The model of a generic algorithm was proposed by Shoup at Eurocrypt '97. A generic algorithm is a general-purpose algorithm that does not make use of any particular property of the representation of the group elements. Shoup proved the hardness of the discrete logarithm problem and the Diffie-Hellman problem with respect to such algorithms for groups whose order contains a large prime factor. By extending Shoup's technique we prove lower bounds on the complexity of generic algorithms solving different problems in cyclic groups, and in particular of a generic reduction of the discrete logarithm problem to the Diffie-Hellman problem. It is shown that the two problems are not computationally equivalent in a generic sense for groups whose orders contain a multiple large prime factor. This complements earlier results which stated this equivalence for all other groups. Furthermore, it is shown that no generic algorithm exists that computes p-th roots efficiently in a group whose order is divisible by p^2 if p is a large prime.

Keywords. Diffie-Hellman protocol, discrete logarithms, generic algorithms, roots in finite groups, complexity, lower bounds.

1 Introduction

1.1 Computational Problems in Cyclic Groups

Let G be a finite cyclic group with generator g. The security of the well-known Diffie-Hellman (DH) protocol [3] relies on the difficulty of the following computational problem, the so-called *Diffie-Hellman (DH) problem*: Given two group elements g^x and g^y, compute g^{xy}. Of course this problem is at most as hard as the *discrete logarithm (DL) problem*: Given g^x, compute x. The relationship between the two problems has been studied intensively with the objective to prove that the DH problem is as hard as the DL problem for any group [2],[5],[6],[1],[7]. For certain classes of groups or, more precisely, for all groups of certain orders, the DH problem and the DL problem were shown to be polynomial-time equivalent. In Section 1.3 we give two examples of such proofs. We will see in Section 2.2 that these reductions are close to optimal. However, it is shown in Section 2.3 that for cyclic groups of certain orders, such an equivalence *cannot* be proved

universally. This is true for groups of which the order contains at least one multiple large prime factor. In Section 3, a complete characterization is given for when p-th roots can be computed efficiently in a group G by a generic algorithm.

1.2 Generic Algorithms

Shoup considered in [12] the difficulty of the DL problem, the DH problem, and related problems with respect to generic algorithms. Intuitively, a generic algorithm is a general-purpose algorithm that does not make use of any property of the representation of the group elements other than the fact that each group element has a *unique* representation, e.g., by some binary string. More precisely, a generic algorithm for the group $\mathbf{Z}_n$ takes as input a list $(\sigma(x_1), \ldots, \sigma(x_l))$, where the x_i are elements of $\mathbf{Z}_n$ and σ is a random encoding of the group elements, i.e., a random mapping from $\mathbf{Z}_n$ to a set S of size n of binary strings. The generic algorithm is allowed to make calls to oracles which compute the functions add, mapping $S \times S$ to S, and inv, mapping S to S, with

$$\mathrm{add}(\sigma(x), \sigma(y)) = \sigma(x + y) \quad \text{and} \quad \mathrm{inv}(\sigma(x)) = \sigma(-x).$$

Shoup proved that no generic algorithm can solve the DH problem in cyclic groups of order n with non-negligible probability substantially faster than in time $\Theta(\sqrt{p})$, where p is the largest prime factor of n. Clearly, this implies that the same lower bound holds for the DL problem. The Pohlig-Hellman algorithm [10], together with the baby-step giant-step time-memory tradeoff, is a generic algorithm that (almost) matches this bound.

The Pohlig-Hellman discrete-logarithm algorithm works as follows. Let g be a generator of the (multiplicatively written) cyclic group G, and let $a = g^x$. For a fixed prime factor q of the order $|G|$ of G, consider the group element $a^{|G|/q} = g^{x \cdot |G|/q}$. The algorithm is based on the following two observations. Because $(a^{|G|/q})^q = a^{|G|} = e$, where e is the neutral element of G, the group element $a^{|G|/q}$ can take q possible values when a varies over G, namely the q different q-th roots

$$g^0 = e\,,\ g^{|G|/q}\,,\ \ldots\,,\ g^{(q-1)|G|/q}$$

of e. These group elements form a subgroup of G which is generated by $g^{|G|/q}$. Second, the modulus of x with respect to q determines *which* of these roots is equal to $a^{|G|/q}$. More precisely,

$$a^{|G|/q} = g^{i \cdot |G|/q} \quad \Longleftrightarrow \quad x \equiv i \pmod{q}\ .$$

Hence x can be determined modulo q by solving the DL problem in the subgroup $\langle g^{|G|/q} \rangle$ of G. If q is a prime factor of $|G|$ with multiplicity $f > 1$, then the coefficients $x_0, x_1, \ldots, x_{f-1}$ of the q-adic representation

$$x \equiv x_0 + x_1 q + \cdots + x_{f-1} q^{f-1} \pmod{q^f}$$

can be computed as follows. Because $x_0 \equiv x \pmod{q}$, the first coefficient x_0 can be obtained as just described. When x_0 is known, the group element

$$(a \cdot g^{-x_0})^{|G|/q^2}$$

is computed. Because $(a \cdot g^{-x_0})^{|G|/q^2} = g^{x_1 \cdot |G|/q}$, this group element is again equal to one of the q-th roots of the neutral element. The coefficient x_1 can also be determined by solving the DL problem in $\langle g^{|G|/q} \rangle$.

With this method, x can be computed modulo q^f for all prime factors q of $|G|$, and Chinese remaindering yields x modulo $|G|$, i.e., the discrete logarithm of a. The complexity of this algorithm for a group G with $|G| = \prod q_i^{f_i}$ is

$$O\left(\sum f_i(\log |G| + q_i)\right) .$$

Let q be the largest prime factor of $|G|$. If memory space for storing $\sqrt{q}$ group elements is available, the running time reduces to $O(\sum f_i(\log |G| + \sqrt{q_i} \log q_i))$ when the baby-step giant-step method is applied for computing discrete logarithms in the subgroups. This running time is of order

$$\sqrt{q} \cdot (\log |G|)^{O(1)} .$$

The (more efficient) index-calculus methods for the computation of discrete logarithms in the multiplicative group of a finite field (see for example [8]) are not generic. They make use of the fact that the representation of a group element is an integer or a polynomial which can efficiently be factored in some cases, and that this decomposition is compatible with the group operation. However, there exist groups for which no faster algorithms are known than the generic methods so far. Examples are non-supersingular elliptic curves over finite fields.

1.3 Examples of Generic Reductions of the DL Problem to the DH Problem

The following two reductions of the DL problem to the DH problem, which work for all groups of certain orders, are special cases of reductions due to den Boer [2] and Maurer [5] (see also [7],[1]). The two lemmas below are corollaries to the results proved there. A reduction of the DL problem to the DH problem can be given by an efficient algorithm for computing discrete logarithms in a group $G = \langle g \rangle$ of order $|G|$ (where no other assumptions on G or, more precisely, on the representation of the elements of G, are made) which is allowed to make calls to a so-called *Diffie-Hellman oracle* for G.

Definition 1. A *Diffie-Hellman (DH) oracle* for a group G with respect to a generator g takes as inputs two elements $a, b \in G$ (where $a = g^u$ and $b = g^v$) and returns the element g^{uv}.

In Lemmas 2 and 3, reductions of the DL problem to the DH problem are described in special situations, where the number of required calls to the DH oracle is minimized. The motivation is that the resulting lower bound on the complexity of breaking the DH protocol is equal to a possible corresponding lower bound for solving the DL problem, divided by the number of calls to the DH oracle performed by the reduction algorithm.

Lemma 2. [2] *Consider a group $G = \langle g \rangle$ whose order $|G| = p$ is a prime with $p - 1 = 2^l$ for some integer l. Then there exists an algorithm that computes discrete logarithms in G in time polynomial in $\log p$ and makes*[1]

$$\log(p-1) - 1$$

calls to a DH oracle for G.

Remark. Note that the result of Shoup mentioned above implies that *without* a DH oracle, discrete logarithms in G cannot be computed substantially faster than in time $\Theta(\sqrt{p})$ by a generic algorithm.

Proof. Let g^x be given. By calling the DH oracle for G with respect to g one can compute the group elements

$$b_0 = g^x \ , \quad b_1 = g^{x^2} = \mathrm{DH}_g(b_0, b_0) \ , \quad b_2 = g^{x^4} \ , \ \ldots \ , \ b_{l-1} = g^{x^{2^{l-1}}} = g^{x^{(p-1)/2}} \ .$$

Let $x = c^w \pmod p$, where c is a fixed generator of $GF(p)^*$, and let further $w \equiv w_0 + 2w_1 + \cdots + 2^{l-1}w_{l-1} \pmod{p-1}$. First, w_0 can be determined because $b_{l-1} = g$ if $w_0 = 0$ and $b_{l-1} = g^{-1}$ if $w_0 = 1$. Furthermore

$$b_{l-2} = g^{x^{(p-1)/4}} = g^{c^{(w_0+2w_1)(p-1)/4}} = \left(g^{c^{w_0(p-1)/4}}\right)^{c^{w_1(p-1)/2}} , \qquad (1)$$

where

$$g' := g^{c^{w_0(p-1)/4}}$$

is also a generator of G. The last expression of (1) is equal to g' if $w_1 = 0$ and to $(g')^{-1}$ if $w_1 = 1$. Proceeding like this, w and $x = c^w$ can be determined in time polynomial in $\log p$, and by $l - 1 = \log(p-1) - 1$ calls to the DH oracle for G. □

Lemma 3. [5] *Consider a group $G = \langle g \rangle$ whose order $|G| = p$ is a prime with $p \equiv 3 \pmod 4$, and assume that a cyclic elliptic curve $E_{a,b}(p)$ (with generator Q) over $GF(p)$ exists such that $|E_{a,b}(p)| = 2^l$ for some l. Then there exists an algorithm that computes discrete logarithms in G with probability at least $1/2 - 1/\sqrt{p}$ in time polynomial in $\log p$ and makes at most $9 \log p$ calls to a DH oracle for G.*

Proof. Note first that $p - 2\sqrt{p} + 1 \le |E_{a,b}(p)| \le p + 2\sqrt{p} + 1$. (For an introduction to elliptic curves, see for example [9].) Let g^x be given. Then, x can be computed as follows. From g^x, compute

$$g^{(x+d)^3 + a(x+d) + b}$$

for some random offset d (this requires two calls to the DH oracle for G), and

$$g^{((x+d)^3 + a(x+d) + b)^{(p+1)/4}} =: g^y$$

[1] All logarithms in this paper are to the base 2.

by at most $2\log((p+1)/4)$ oracle calls. If $g^{y^2} \neq g^{(x+d)^3+a(x+d)+b}$ (this can be tested by one oracle call), then $(x+d)^3+a(x+d)+b$ is not a quadratic residue modulo p and the algorithm fails. With probability at least

$$\frac{p-2\sqrt{p}+1}{2p} > \frac{1}{2} - \frac{1}{\sqrt{p}}$$

however, the algorithm does not fail, and the pair (g^{x+d}, g^y) is such that $P := (x+d, y)$ is a point of the curve $E_{a,b}(p)$. The number of oracle calls for the computation of this point is at most $2\log((p+1)/4)+3$.

We now compute w (modulo 2^l) such that $P = wQ$. In the following, the points of the elliptic curve are represented in projective coordinates (see [9]). In this representation, doubling of a point in the curve requires 7 multiplications modulo p but, unlike in the affine representation, no inversions. From (g^x, g^y, g^1) one can hence compute, in polynomial time, $(g^{x_1}, g^{y_1}, g^{z_1})$ such that $(x_1, y_1, z_1) = P_1 := 2P$, using 7 oracle calls. Analogously, $(g^{x_i}, g^{y_i}, g^{z_i})$ can be computed for $i = 2, \ldots, l-1$ such that $(x_i, y_i, z_i) = P_i := 2^i P$.

Let $w \equiv w_0 + 2w_1 + \cdots + 2^{l-1}w_{l-1} \pmod{2^l}$, where the w_i are binary. Then we have $P_{l-1} = \mathcal{O}$ if $w_0 = 0$ and $P_{l-1} \neq \mathcal{O}$ if $w_0 = 1$. Here $\mathcal{O}$ stands for the neutral element of $E_{a,b}(p)$, the so-called point at infinity. In projective coordinates, (x, y, z) represents $\mathcal{O}$ if and only if $z = 0$. Hence w_0 can be determined by comparing $g^{z_{l-1}}$ with $g^0 = e$. Furthermore,

$$P_{l-2} = 2^{l-2}P = 2^{l-2}(w_0 + 2w_1)Q = 2^{l-2}w_0Q + 2^{l-1}w_1Q \ .$$

Hence P_{l-2} equals $2^{l-2}w_0Q$ if and only if $w_1 = 0$. Given $(g^{x_{l-2}}, g^{y_{l-2}}, g^{z_{l-2}})$ and $(x', y', 1) = 2^{l-2}w_0Q$, one can check equality of the points by comparing $(g^{z_{l-2}})^{x'}$ with $g^{x_{l-2}}$ and $(g^{z_{l-2}})^{y'}$ with $g^{y_{l-2}}$. This requires no calls to the DH oracle.

Proceeding like this, w and $wQ = P = (x+d, y)$, and hence x, can be computed in polynomial time and by $7(l-1) \leq 7(\log(p+2\sqrt{p}+1)-1)$ calls to the DH oracle. Hence the total number of oracle calls for the computation of x from g^x is

$$2\log((p+1)/4) + 3 + 7(\log(p+2\sqrt{p}+1)-1) \leq 9\log p \ ,$$

and this concludes the proof. □

In the next section we will show that the number of oracle calls in the reductions of Lemmas 2 and 3 are close to optimal and a constant multiple of the optimal number, respectively.

A generalization of the method used in the proof of Lemma 3 allows to prove the following theorem, which is an immediate consequence of the results of [5]. It is well-known that for each number $d \in [p-2\sqrt{p}+1\,,\, p+2\sqrt{p}+1]$ there exists a cyclic elliptic curve over $GF(p)$ of order d. For a number n, we define $\nu(n)$ to be the minimum of the set of largest prime factors of the numbers d in the interval $[n-2\sqrt{n}+1\,,\, n+2\sqrt{n}+1]$.

Theorem 4. [5] *Let $n = \prod p_i^{e_i}$ be a positive integer such that all multiple prime factors of n are of order $(\log n)^{O(1)}$. Then there exists a generic algorithm that makes calls to a DH oracle for G and computes discrete logarithms in groups G of order n in time*

$$\sqrt{\max\{\nu(p_i)\}} \cdot (\log n)^{O(1)} .$$

The following conjecture on the existence of smooth numbers in small intervals has been made in [5].

Conjecture 1. $\nu(n) = (\log n)^{O(1)}$.

This assumption appears plausible when considering related results on smooth numbers in larger intervals, but it has not been proved. If this conjecture is true, then there exists a polynomial-time generic reduction algorithm of the DL problem to the DH problem for every group whose order does not contain a multiple large prime factor.

2 Lower Bounds on Generic Reductions of the DL Problem to the DH Problem

2.1 Motivation

In this section we prove lower bounds on the complexity of reductions of the DL problem to the DH problem that work for all cyclic groups of a certain order, i.e., generic algorithms for the computation of discrete logarithms that make calls to a DH oracle, but which do not exploit any particular property of the representation of the group elements.

As mentioned earlier, Shoup proved in [12] that in the generic model, the DH and DL problems are of roughly equal difficulty. However, this has no implications for any particular group G. On the other hand, a generic reduction between the two problems would prove the relative hardness of the DH problem, with respect to the hardness of the DL problem, for *every (particular) group* of a certain order. All the previously described reductions of the DL problem to the DH problem are generic (see [2],[5],[1], and Section 1.3).

The smaller the number of calls to the DH oracle required in a reduction of the DL problem to the DH problem is, the stronger is the equivalence result implied by the reduction. Ideally, one might wish to find a reduction that uses only one or a constant number of oracle calls. However, the results stated below show that such a reduction cannot exist in the generic model.

2.2 A General Lower Bound on the Number of DH-Oracle Calls

Theorem 5. *Let n be a positive integer and let p be a prime factor of n. For any (possibly probabilistic) generic reduction algorithm of the DL problem to the DH problem that works for groups of order n, runs in time at most T (≥ 4), and*

computes the correct discrete logarithm with probability α, the expected number of calls to the DH oracle is at least

$$\alpha \log(p/T^2) \ .$$

We need the following lemma (see [11] or [12]).

Lemma 6. *When given a polynomial $P(X_1, \ldots, X_k)$ over $\mathbf{Z}_{p^t}$ of total degree d, the probability that $P(x_1, \ldots, x_k) = 0$ for independently and randomly chosen elements $x_1, \ldots, x_k$ of $\mathbf{Z}_{p^t}$ is at most d/p.*

Proof of Theorem 5. The proof method is related to the technique used by Shoup in [12]. The underlying group is $\mathbf{Z}_n$ together with a random encoding σ of the group elements, i.e., a randomly chosen map from $\mathbf{Z}_n$ to some set S of n elements. Let $n = p^t s$ with $\gcd(s, p) = 1$. The generic algorithm must compute discrete logarithms, i.e., it takes as input $\sigma(1)$ and $\sigma(x)$ and should output x. Since we are proving a lower bound, and because additional knowledge can only reduce the running time, we can assume that discrete logarithms modulo s are easy to compute. Hence in what follows, let without loss generality $n = p^t$. The DH oracle, which can be used in addition to the oracles for addition and inversion, is an oracle for multiplication, i.e.,

$$\mathrm{DH}(\sigma(y), \sigma(z)) = \sigma(y \cdot z) \ .$$

Let the generic algorithm make A calls to the addition and inversion oracles and M calls to the multiplication oracle during a particular execution. Hence we have $A + M \leq T$ if T is an upper bound on the number of steps the algorithm performs. The algorithm can interact with the oracles in the following way. By calls to the oracles, $\sigma(P_i(x))$ can be computed for (without loss of generality pairwise distinct) polynomial expressions $P_i(X)$, $i = 1, \ldots, A + M + 2$, with

$$P_1(X) = 1 \,, \ P_2(X) = X \,,$$

and where, for all $i \geq 3$, $P_i(X)$ is equal to either

$$P_k(X) + P_l(X) \,, \ -P_k(X) \,, \text{ or } \ P_k(X) \cdot P_l(X)$$

for some $k, l < i$. Of course the algorithm is free to perform additional computations, but this does not influence the following information-theoretic arguments. The crucial observation is that, unless $P_i(x) = P_j(x)$ for some $i \neq j$, the algorithm sees completely random elements of S and cannot learn anything about x other than the fact that $P_i(x) \neq P_j(x)$ for all $i \neq j$. In other words, let $\mathcal{E} \subseteq \mathbf{Z}_n$ be the event that $P_i(x) = P_j(x)$ for some $i \neq j$, and let $\overline{\mathcal{E}}$ be the complementary event. Then, we have

$$I(x \,; \, [\sigma(P_i(x))]_{i=1,\ldots,A+M+2} \,|\, \overline{\mathcal{E}}) = 0 \ , \tag{2}$$

where $I(U; V|\mathcal{A}) = H(U|\mathcal{A}) - H(U|V, \mathcal{A})$ is the mutual (Shannon) information between the random variables U and V, given the event $\mathcal{A}$. In other words,

equation (2) means that the random variable x is statistically independent of $[\sigma(P_1(x)), \ldots, \sigma(P_{A+M+2}(x))]$, given the event $\overline{\mathcal{E}}$. This holds because σ is chosen randomly among all possible encodings.

Lemma 6 implies that

$$\mathrm{P}[\mathcal{E}] \leq \frac{2^M(A+M+2)(A+M+1)}{2p} , \tag{3}$$

i.e., that the probability of the event $\mathcal{E}$ is upper bounded by D/p, where

$$D := 2^M(A+M+2)(A+M+1)/2 .$$

The number D is equal to the maximal possible degree of the polynomials P_i, namely 2^M, multiplied with the number $(A+M+2)(A+M+1)/2$ of two-element sets $\{i,j\} \subseteq \{1,\ldots,A+M+2\}$.

We conclude from (2) and (3) that the probability that the algorithm answers correctly is upper bounded by

$$\mathrm{P}[\mathcal{E}] + \mathrm{P}[\overline{\mathcal{E}}] \cdot \frac{1}{n \cdot \mathrm{P}[\overline{\mathcal{E}}]} \leq \mathrm{P}[\mathcal{E}] + \frac{1}{p^t} \leq \frac{D+1}{p} \tag{4}$$

because x is a random element of $\mathbf{Z}_n$. Note that $n \cdot \mathrm{P}[\overline{\mathcal{E}}]$ is the number of elements x of $\mathbf{Z}_{p^t}$ compatible with the event $\overline{\mathcal{E}}$. The best strategy of the algorithm, given $\overline{\mathcal{E}}$, is to output one of these elements randomly. From the assumption in the theorem, and from (4), we conclude

$$\mathrm{E}[\min\{(D+1)/p, 1\}] \geq \alpha ,$$

where the expectation is taken over the executions of the probabilistic algorithm and over the random choices of σ and x. Hence $\mathrm{E}[M] \geq \alpha \cdot \log(p/T^2)$ holds for $T \geq 4$. □

Theorem 5 implies that the reduction of Lemma 2 is optimal with respect to the number of calls to the DH oracle, and that the number of oracle calls in the reduction of Lemma 3 is only 18 times greater than the theoretical lower bound.

The result of Theorem 5 is pessimistic in the following sense. Assume that one can prove for a certain class of groups that discrete logarithms cannot be computed faster than in time T_{DL}. Then the best result one can hope for by using a general-purpose reduction is that the DH problem cannot be solved faster than in time $T_{DH} = T_{DL}/\log p$.

2.3 Group Orders with Multiple Large Prime Factors

Let us now consider the problem of a generic reduction of the DL to the DH problem for groups G whose order contains a multiple large prime factor p. It was shown in [7] that such a reduction is possible if the DH oracle can be used to either construct a DH oracle for certain subgroups of $G = \langle g \rangle$ such as $\langle g^p \rangle$, or to compute p-th roots in G. Shoup proved in [12] that the first problem cannot

be solved with generic algorithms, and Theorem 10 below states that the same is true for the second problem. However, at first sight it appears plausible that there exist different ways of constructing generic reductions for such groups, for example by using the general auxiliary-group technique of [7] (see also [6]) with auxiliary groups defined directly over the ring $\mathbf{Z}_{p^t}$ and not over the field $GF(p)$. But, perhaps somewhat surprisingly, it can be shown that no efficient generic reduction exists for such groups. In other words, it is impossible to prove the computational equivalence of the DH and DL problems for all groups when no assumption on the representation of the group elements is made.

Theorem 7. *Let n be a positive integer and let p be a multiple prime factor of n. For any (possibly probabilistic) generic reduction algorithm of the DL problem to the DH problem that works for groups of order n, runs in time at most T, and outputs the correct discrete logarithm with probability α, we have*

$$T \geq \sqrt{2\alpha p} - 2 \; .$$

Proof. Let p^t, $t > 1$, be the maximal power of p dividing n. By the same argument as in the proof of Theorem 5, we can assume that $n = p^t$. Let $x \equiv x_0 + x_1 p + \cdots + x_{t-1}p^{t-1} \pmod{p^t}$ be the p-adic expansion of x modulo p^t. We give a lower bound on the complexity of the computation of x_{t-1}. Again, because additional knowledge can only decrease the running time, we can assume that $x_0, \ldots, x_{t-2}$ are given, and hence that $x \equiv x_{t-1}p^{t-1} \pmod{p^t}$. By interacting T times with the given oracles, the algorithm can compute $\sigma(P_i(x_{t-1}))$ for $i = 1, \ldots, T+2$, where the $P_i(X)$ are (pairwise distinct) polynomials with $P_1(X) = 1$, $P_2(X) = p^{t-1}X$ and for $i \geq 3$ either $P_i(X) = P_k(X) + P_l(X)$, $P_i(X) = -P_k(X)$, or $P_i(X) = P_k(X) \cdot P_l(X)$ for some $1 \leq k, l < i$. The crucial, and somewhat surprising, fact now is that, although the polynomials can be multiplied, $P_i(X)$ is a *linear* polynomial in X for all i. More precisely, the polynomials are of the form $P_i(X) = a_i p^{t-1}X + b_i$. This follows by induction and from the fact that $p^t \equiv 0 \pmod{n}$. According to Lemma 6, the probability of the event $\mathcal{E}$ that $P_i(x_{t-1}) = P_j(x_{t-1})$ for some $i \neq j$ is upper bounded by $(T+2)(T+1)/(2p)$. As in the proof of Theorem 5, we conclude

$$\alpha \leq \frac{(T+2)(T+1)+2}{2p} \; ,$$

and this implies the statement of the theorem. □

Note that each of the pessimistic results described at the beginning of this section also implies a statement similar to the one of Theorem 7.

The bound of Theorem 7 shows that a DH oracle for G is virtually of no help for computing discrete logarithms in G modulo p^t when p is a prime factor of $|G|$ with multiplicity $t > 1$. The reason is that discrete logarithms can be obtained equally efficiently *without* a DH oracle by the baby-step giant-step method.

2.4 A Completeness Result

Theorems 5 and 7 state lower bounds on the complexity of probabilistic algorithms with respect to their *worst-case* running time. These results can be used to derive bounds that hold for the *expected* running time, which of course are the results we are finally interested in.

Theorem 8. *Let n be a positive integer, let p be the largest prime factor of n, and let q be the largest multiple prime factor of n. For any generic reduction algorithm of the DL problem to the DH problem for cyclic groups of order n, the expected running time T of the algorithm and the expected number M of DH-oracle calls are lower bounded by*

$$T \geq \frac{\sqrt{q}}{2} - 1$$

and by

$$M \geq \frac{\log(p/T^2)}{2} - 1 \,.$$

Proof. The proof is based on the fact that the actual running time of a probabilistic algorithm exceeds twice its expected running time only with probability at most $1/2$. We construct from the given algorithm a new algorithm by stopping the execution after $2T$ steps. The new algorithm has worst-case running time $2T$ and answers correctly with probability at least $1/2$. Then the two bounds follow by applying Theorems 7 and 5 to the new algorithm. □

Theorems 4 and 8 imply the following characterization of groups for which the DH and DL problems are equivalent in a generic, but non-uniform, sense.

Corollary 9. *If Conjecture 1 is true, then there exists a polynomial-time generic reduction of the DL problem to the DH problem for groups G of order n if and only if all multiple prime factors of n are of size $(\log n)^{O(1)}$.*

3 Computing Roots in Cyclic Groups

The problem of computing roots in cyclic groups, which is of independent interest, arises in the context of reducing the DL problem to the DH problem in groups whose order contains multiple prime factors, as mentioned in Section 7. We give a complete characterization of which roots can be computed efficiently by generic algorithms in groups of a certain order.

Let $n = p^t s$ with $t \geq 2$ and $\gcd(s, p) = 1$. The following theorem states that no efficient (i.e., polynomial-time in $\log p$) general-purpose algorithm exists that computes p-th (or p^r-th for $1 < r < t$) roots in a cyclic group $G = \langle g \rangle$ of order n. Moreover, this holds even when the algorithm is also allowed to make calls to a DH oracle for G.

Theorem 10. *Let a generic algorithm be given that computes p^r-th ($r \geq 1$) roots in a cyclic group $G = \langle g \rangle$ of order $n = p^t s$, where $\gcd(s,p) = 1$ and $t > r$ hold. Assume further that the algorithm is (besides the usual operations) allowed to make calls to a DH oracle for G (with respect to some generator g). Then the probability that the algorithm outputs a correct root is at most*

$$\frac{((T+2)(T+1)+2)(\lceil t/r \rceil - 1)}{2p}$$

if T is an upper bound on the number of steps the algorithm performs.

Remark. It is easy to see that in case $r \geq t$, p^r-th roots can easily be computed in G (without a DH oracle).

Proof. The proof is related to the proof of Theorem 5 in [12]. Assume without loss of generality that $n = p^t$, and that an encoding σ of the group elements is chosen randomly. The input to the algorithm is $\sigma(1)$ and $\sigma(p^r x)$. The algorithm can interact with the oracles for addition, inversion, and multiplication (the DH oracle) by computing a list of expressions $\sigma(P_i(x))$ for $i = 1, \ldots, T+2$ where $P_1(X) = 1$, $P_2(X) = p^r X$, and, for $i \geq 3$, $P_i(X) = P_k(X) + P_l(X)$, $P_i(X) = -P_k(X)$, or $P_i(X) = P_k(X) \cdot P_l(X)$, where $k, l < i$. It is not difficult to see by induction that the polynomials have the form

$$P_i(X) = \sum_{m=0}^{d} a_{i,m} (p^r X)^m ,$$

where $d := \lceil t/r \rceil - 1$. Note that, although the polynomials can also be multiplied, the degrees are bounded by $d \leq t/r$.

Let the complementary events $\mathcal{E}$ and $\overline{\mathcal{E}}$ be defined as in the proof of Theorem 5. From Lemma 6 we conclude

$$\mathrm{P}[\mathcal{E}] \leq \frac{(T+2)(T+1)d}{2p} .$$

If $\overline{\mathcal{E}}$ occurs, the best the algorithm can do at the end is to output $\sigma(P_i(x))$ for some $1 \leq i \leq T+2$. The probability that this answer is correct, i.e., that $p^r P_i(x) - p^r x = 0$, given $\overline{\mathcal{E}}$, is at most $d/(p \cdot \mathrm{P}[\overline{\mathcal{E}}])$, according to Lemma 6. Hence the probability that the algorithm answers correctly is at most

$$\mathrm{P}[\mathcal{E}] + \mathrm{P}[\overline{\mathcal{E}}] \cdot \frac{d}{p \cdot \mathrm{P}[\overline{\mathcal{E}}]} \leq \frac{(T+2)(T+1)d + 2d}{2p} ,$$

and this concludes the proof of the theorem. □

Theorem 10, together with the argument used in the proof of Theorem 8, implies that no generic algorithm can compute p^r-th roots ($r \geq 1$) in a group whose order is divisible by p^t ($t > r$) substantially faster than in expected time $\Theta(\sqrt{p})$, even when the algorithm is allowed to make calls to a DH oracle. It is not surprising that also this bound is asymptotically tight, as Theorem 11 shows.

Theorem 11. *Let $G = \langle g \rangle$ be a cyclic group of order $|G| = p^t s$ with $\gcd(s,p) = 1$. For $1 \leq r < t$, p^r-th roots can be computed in G in time $\sqrt{p} \cdot (\log |G|)^{O(1)}$ by a generic algorithm (that uses group operations in G, but no calls to a DH oracle).*

Proof. The following algorithm is a generalization of a method, due to Massey, for the computation of square roots modulo a prime number [4]. Let h be a p^r-th power in G. With the Pohlig-Hellman method, together with the baby-step giant-step time-memory tradeoff, the discrete logarithm k (modulo p^t) of h can be computed in time $\sqrt{p} \cdot (\log |G|)^{O(1)}$, where memory space for storing $\sqrt{p}$ group elements is needed. Since h is a p^r-th power in G, k is a multiple of p^r. Let $d :\equiv -s^{-1} \pmod{p^r}$ (note that this inverse exists and can be computed efficiently). Then

$$\left(g^{s \cdot (k/p^r) \cdot d}\right)^{-1} h^{(sd+1)/p^r}$$

is a p^r-th root of h. This algorithm is obviously generic and runs in time $\sqrt{p} \cdot (\log |G|)^{O(1)}$. □

4 Concluding Remarks

We have proved lower bounds on the complexity of generic algorithms solving certain problems in cyclic groups. In particular, we have studied generic reductions of the DL problem to the DH problem. These reductions are useful because they imply the equivalence of the DH and DL problems for *every* particular group of a certain order, whereas the results of [12] concerning generic algorithms for solving the DH and DL problems cannot be directly applied when considering a concrete group. We have derived lower bounds on the number of calls to the DH oracle and on the overall complexity of such reductions. The second of these bounds implies that the DH and DL problems are not computationally equivalent in general for all groups of which the order contains a multiple large prime factor. This pessimistic result completes earlier results stating that an efficient generic algorithm which makes calls to a DH oracle and computes discrete logarithms in a group G exists if $|G|$ does *not* contain a multiple large prime factor, and shows how relevant multiplicity of prime factors is in this context.

We state as an open problem to prove the equivalence of the DH and DL problems also for particular classes of groups whose orders contain multiple large prime factors. However, such a proof must exploit certain properties of the representation of the group elements. Another open question is whether there exists, for all remaining groups, an efficient *uniform* reduction of the DL problem to the DH problem, and whether it is possible to find a reduction algorithm for every group that achieves the theoretical lower bound on the number of required calls to the DH oracle.

References

1. D. Boneh and R. J. Lipton, Algorithms for black-box fields and their application to cryptography, *Advances in Cryptology - CRYPTO '96*, Lecture Notes in Computer Science, Vol. 1109, pp. 283–297, Springer-Verlag, 1996.
2. B. den Boer, Diffie-Hellman is as strong as discrete log for certain primes, *Advances in Cryptology - CRYPTO '88*, Lecture Notes in Computer Science, Vol. 403, pp. 530–539, Springer-Verlag, 1989.
3. W. Diffie and M. E. Hellman, New directions in cryptography, *IEEE Transactions on Information Theory*, Vol. 22, No. 6, pp. 644–654, 1976.
4. J. L. Massey, Advanced Technology Seminars Short Course Notes, pp. 6.66–6.68, Zürich, 1993.
5. U. M. Maurer, Towards the equivalence of breaking the Diffie-Hellman protocol and computing discrete logarithms, *Advances in Cryptology - CRYPTO '94*, Lecture Notes in Computer Science, Vol. 839, pp. 271–281, Springer-Verlag, 1994.
6. U. M. Maurer and S. Wolf, The relationship between breaking the Diffie-Hellman protocol and computing discrete logarithms, to appear in *SIAM Journal of Computing*, 1998.
7. U. M. Maurer and S. Wolf, Diffie-Hellman oracles, *Advances in Cryptology - CRYPTO '96*, Lecture Notes in Computer Science, Vol. 1109, pp. 268–282, Springer-Verlag, 1996.
8. K. S. McCurley, The discrete logarithm problem, in *Cryptology and computational number theory*, C. Pomerance (Ed.), Proc. of Symp. in Applied Math., Vol. 42, pp. 49–74, American Mathematical Society, 1990.
9. A. J. Menezes, *Elliptic curve public key cryptosystems*, Kluwer Academic Publishers, 1993.
10. 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*, Vol. 24, No. 1, pp. 106–110, 1978.
11. J. T. Schwartz, Fast probabilistic algorithms for verification of polynomial identities, *Journal of the ACM*, Vol. 27, No. 4, pp. 701–717, 1980.
12. V. Shoup, Lower bounds for discrete logarithms and related problems, *Advances in Cryptology - EUROCRYPT '97*, Lecture Notes in Computer Science, Vol. 1233, pp. 256–266, Springer-Verlag, 1997.