On the Complexity of Breaking the Diffie-Hellman Protocol

Ueli M. Maurer and Stefan Wolf

Institute for Theoretical Computer Science
ETH Zürich
CH-8092 Zürich, Switzerland
E-mail addresses: {maurer,wolf}@inf.ethz.ch

April 18, 1996

Abstract

It is shown that for a class of finite groups, breaking the Diffie-Hellman protocol is polynomial-time equivalent to computing discrete logarithms. Let G be a cyclic group with generator g and order $|G|$ whose prime factorization is known. When for each large prime factor p of $|G|$ an auxiliary group H_p defined over $GF(p)$ with smooth order is given, then breaking the Diffie-Hellman protocol for G and computing discrete logarithms in G are polynomial-time equivalent. Possible auxiliary groups H_p are elliptic curves over $GF(p)$ or over an extension field of $GF(p)$, certain subgroups of the multiplicative group of such an extension field, and the Jacobian of a hyperelliptic curve. For a list of expressions in p, including $p-1$, $p+1$, and the cyclotomic polynomials of low degree in p, it is shown that an appropriate group H_p can efficiently be constructed if one of the expressions in the list is smooth. Furthermore, efficient constructions of Diffie-Hellman groups with provable equivalence are described.

1 Introduction

Two challenging open problems in cryptography are to prove or disprove that breaking the Diffie-Hellman protocol [10] is computationally equivalent to computing discrete logarithms in the underlying group and that breaking the RSA system [32] is computationally equivalent to factoring the modulus. In this paper we take a number of steps towards the solution of the first of these problems.

Let H be a finite group (written multiplicatively), and for $g \in H$, let $G = \langle g \rangle$ be the cyclic subgroup generated by g. The discrete logarithm problem for the group H (or G) can be stated as follows: Given g and $a \in G$, find the unique integer s in the interval $[0, |G|-1]$ such that $g^s = a$, where s is called the discrete logarithm (DL) of a to the base g. The discrete logarithm problem is sometimes also defined as the generally easier problem of finding any s satisfying $g^s = a$, but if $|G|$ is known then the two problems are equivalent.

The Diffie-Hellman (DH) protocol [10] allows two parties Alice and Bob connected by an authenticated but otherwise insecure channel (for instance an insecure telephone line where Alice and Bob authenticate each other by speaker recognition) to generate a mutual secret key which is computationally infeasible to determine for a passive eavesdropper overhearing the entire conversation between Alice and Bob.

The protocol works as follows. Let $G = \langle g \rangle$ be a cyclic group generated by g for which the discrete logarithm problem is computationally infeasible. Note that it is unknown whether such a group exists. Specific groups that have been proposed for application in this protocol are the multiplicative groups of large finite fields (prime fields [10] or extension fields), the multiplicative group of residues modulo a composite number [23, 25], elliptic curves over finite fields, the Jacobian of a hyperelliptic curve over a finite field and the class group of imaginary quadratic fields [6].

In order to generate a mutual secret key, Alice and Bob secretly choose integers s_A and s_B, respectively, at random from the interval $[0, |G|-1]$. Then they compute secretly $a_A = g^{s_A}$ and $a_B = g^{s_B}$, respectively, and exchange these group elements over the insecure public channel. Finally, they compute $a_{AB} = a_B^{s_A} = g^{s_A s_B}$ and $a_{BA} = a_A^{s_B} = g^{s_B s_A}$, respectively. Note that $a_{AB} = a_{BA}$, and hence this quantity can be used as a secret key shared by Alice and Bob. More precisely, they need to apply a function mapping elements of G to the key space of a cryptosystem.

In contrast to digital signature schemes based on the discrete logarithm problem (e.g., [11],[35]), it is not required for the Diffie-Hellman protocol that the order of the group be known. In this case, s_A and s_B are chosen from a sufficiently large interval. In fact, it has been pointed out (e.g., see [25]) that using groups with unknown order may be advantageous, and the non-interactive public-key scheme of [23] relies crucially on the fact that the group order is unknown.

Our paper is organized as follows. In Section 2 the general index search problem is investigated, and a method for computing discrete logarithms is described which is crucial for our technique. Different types of incomplete DH-oracles are defined in Section 3, and it is shown that they are (almost) as strong as a perfect oracle. In Section 4 a general technique for obtaining equivalence results is presented, using general auxiliary groups, and Section 5 contains examples of groups which are suitable auxiliary groups, such as elliptic curves or a subgroup of the multiplicative group of a finite field. In Section 6 we show how to construct DH groups with provable equivalence of the DH problem and the discrete logarithm problem, and in Appendix D we obtain a stronger equivalence result under the assumption of DH oracle *algorithms* for certain groups.

Considerations on related topics can be found in [34] and [4]. In the latter, the notion of a black-box field is introduced which makes more explicit our concept of computation with implicitly represented elements, presented already in [24].

2 The index search problem and methods for computing discrete logarithms

Let $A = \{a_0, \ldots, a_{n-1}\}$ be a set such that for a given i it is easy to compute a_i. The *index search problem* for A, namely the problem of computing for a given $b \in A$ the index i such that $b = a_i$, can trivially be solved by exhaustive search which requires $O(n)$ comparisons. If the set A has the property that the permutation $\sigma: \ a_i \ \mapsto \ a_{i+1}$ (where the index is reduced modulo n) can efficiently be computed, then the search can be sped up by a time-memory tradeoff known as *baby-step-giant-step* algorithm. Using a table of size M to store the sorted list of values $b, \sigma(b), \ldots, \sigma^{M-1}(b)$ one can compute the elements $a_0, a_M, a_{2M}, \ldots$ until one of them is contained in the table.

The discrete logarithm problem in a cyclic group G of order n with generator g is the index search problem for the set $\{1, g, \ldots, g^{|G|-1}\}$. Multiplication with g corresponds to the above mentioned permutation σ. The baby-step-giant-step method and a well-known algorithm due to Pollard [31] (with no rigorously proven running time) are the fastest known discrete logarithm algorithms for general groups. For special groups such as the multiplicative group of a finite field, more efficient algorithms are known. We refer to [26] for a detailed discussion of the discrete logarithm problem and algorithms for solving it.

We now describe a method due to Pohlig and Hellman [30] which reduces the computation of discrete logarithms to the index search problem in certain subsets. It plays a central role in the paper. In particular we will make use of the fact that it requires only group operations and equality testings of group elements.

Let H be a cyclic group with generator h and order $|H| = \prod_{i=1}^{r} q_i^{f_i}$, and let $a \in H$ be given. For $a = h^x$, we first compute x modulo $q_i^{f_i}$ for all i. This is done by determining, modulo q_i, the coefficients x_{ij} of the q_i-adic representation of x modulo $q_i^{f_i}$,

$$x \equiv \sum_{j=0}^{f_i-1} x_{ij} q_i^j \pmod{q_i^{f_i}}.$$

x_{i0} is the index of $a^{\frac{|H|}{q_i}} = h^{x \cdot \frac{|H|}{q_i}} = h^{x_{i0} \cdot \frac{|H|}{q_i}}$ in the set $\{h^{t \cdot \frac{|H|}{q_i}} : t = 0, \ldots, q_i - 1\}$ (indexed by t) which is a cyclic group generated by $h^{\frac{|H|}{q_i}}$.

Assume that $x_{i0}, \ldots, x_{i,k-1}$ are computed already. x_{ik} is the index of

$$a^{\frac{|H|}{q_i^{k+1}}} = \left(h^{x_{i0} + \ldots + x_{i,f_i-1} q_i^{f_i-1}}\right)^{\frac{|H|}{q_i^{k+1}}} = \left(h^{x_{i0} + \ldots + x_{i,k-1} q_i^{k-1}}\right)^{\frac{|H|}{q_i^{k+1}}} \cdot h^{x_{ik} \cdot \frac{|H|}{q_i}}$$

in the set

$$\left\{ \left(h^{x_{i0} + \ldots + x_{i,k-1} q_i^{k-1}}\right)^{\frac{|H|}{q_i^{k+1}}} \cdot h^{t \cdot \frac{|H|}{q_i}} : t = 0, \ldots, q_i - 1 \right\}$$

(indexed by t). Given x modulo $q_i^{f_i}$ for all i, Chinese remaindering yields the discrete logarithm x of a modulo $|H|$.

For the indexed sets defined above, multiplication with $h^{\frac{|H|}{q_i}}$ corresponds to the cyclic permutation σ. Therefore the index search can be sped up by a factor M when a table of size M is used that can be sorted in time $O(M \log M)$. The complexity of the algorithm is

$$O\left(\sum f_i (\log |H| + q_i)\right),$$

or

$$O\left(\sum f_i (\log |H| + \sqrt{q_i} \log q_i)\right)$$

when the baby-step-giant-step method with $M \approx \sqrt{q_i}$ is used.

The described algorithm is efficient only if $|H|$ is smooth, i.e., if $q_i \leq B$ for a small *smoothness bound* B. In the worst case we have $q_i \approx B$ for all i, i.e., the number of factors is $\log |H| / \log B$, and the complexity is

$$O\left((\log |H|)^2 + \frac{B}{\log B} \log |H|\right),$$

or

$$O\left((\log |H|)^2 + \sqrt{B} \log |H|\right)$$

when the baby-step-giant-step method is used.

3 Equivalence between various types of Diffie-Hellman oracles

3.1 Introduction

In order to prove equivalence results between breaking the Diffie-Hellman protocol and computing discrete logarithms we assume the availability of an oracle that solves the Diffie-Hellman problem. However, such an oracle could be defined in several ways, and in this section we compare various types of such oracles and show their equivalence. The natural definition of a Diffie-Hellman oracle is the following.

Definition 1 A *Diffie-Hellman oracle* (DH-oracle for short) for a group G with respect to a given generator g takes as inputs two elements $a, b \in G$ (where $a = g^u$ and $b = g^v$) and returns (without computational cost) the element g^{uv}.

In this section we show that certain apparently weaker oracles are almost as strong as a DH-oracle.

3.2 ε-DH-oracles

Definition 2 For $\varepsilon > 0$, an *ε-DH-oracle* is a probabilistic oracle which returns for an input (g^u, g^v) the correct answer g^{uv} with probability at least ε, provided the input is uniformly distributed over $G \times G$. The *error* of the oracle's answer g^t to the input (g^u, g^v) is defined as $t - uv \pmod{|G|}$. A *translation-invariant* ε-DH-oracle is an ε-DH-oracle whose distribution of the error is the same for every input (g^u, g^v).

A special case of (non-translation-invariant) ε-DH-oracles are *deterministic* oracles answering correctly for a fraction ε of all inputs. We proceed in two steps to prove that an ε-DH-oracle can (normally) be transformed into a perfect DH-oracle. First, the oracle is made translation-invariant by randomization of the input, and then, the correct answer is detected. We assume that the given ε-DH-oracle is time-invariant, i.e., the error distribution can only depend on the input but remains the same when the oracle is called several times.

Lemma 1 *An ε-DH-oracle for a cyclic group G with order $|G|$ can be transformed into a translation-invariant ε-DH-oracle. One call of the latter requires one call to the former and $O(\log |G|)$ group operations.*

Proof: Given the group elements $a = g^u$ and $b = g^v$ we can randomize the input by choosing r and s at random from $[0, |G| - 1]$, providing the oracle with $a' = ag^r$ and $b' = bg^s$ and multiplying the oracle's answer $g^{(u+r)(v+s)+t} = g^{uv+rv+su+rs+t}$ with $(a^{-1})^s \cdot (b^{-1})^r \cdot g^{-rs} = g^{-(rv+su+rs)}$ to obtain g^{uv+t}. Note that a' and b' are random group elements and statistically independent of a and b. The ε-DH-oracle with randomized input is thus a translation-invariant ε-DH-oracle. $\square$

Remark: If $|G|$ is unknown the input can also be randomized, where r and s are chosen at random from a larger interval. The resulting ε-DH-oracle is then "almost translation-invariant" and applicable in the proof of Theorem 1 if the interval is of size at least $2 \cdot |G|/(\varepsilon^2 \cdot \min\{s, 0.1\})$ (this leads to the larger factor for the number of group operations for this case in Theorem 1).

We now show that a translation-invariant ε-DH-oracle can be transformed into an almost perfect DH-oracle. The straight-forward approach to using a translation-invariant ε-DH-oracle may at first sight appear to be to run it $O(1/\varepsilon)$ times until it produces the correct answer.

However, because the Diffie-Hellman decision problem is difficult, a more complicated approach consisting of two phases must be used. The Diffie-Hellman decision problem is, for given g^u, g^v and g^w, to decide whether $g^w = g^{uv}$ and is of course at most as difficult as breaking the DH protocol. In a first phase, which is independent of the actual input, the oracle's error distribution is determined. In the second phase, the oracle is used for a given input to compute the correct solution with overwhelming probability. These two phases are described in Appendix A in the proof of the following theorem.

Theorem 1 *For every cyclic group G with generator g and known order $|G|$ and for every $\beta > 0$ there exists a DH-oracle algorithm which makes calls to an ε-DH-oracle and whose answer is correct with probability at least $1-\beta$. The number of oracle calls is $O(\log(1/\beta\varepsilon)/\varepsilon^4)$. If the order of G is unknown but all the prime factors of $|G|$ are greater than $(1+s)/\varepsilon$ for some $s > 0$, then the number of required calls to the ε-DH-oracle is*

$$O\left(\frac{1}{(\varepsilon^2 \cdot \min\{s, 0.1\})^2} \cdot \log \frac{1}{\beta\varepsilon}\right).$$

The number of required group operations is $\log|G|$ or $\log(|G|/(\varepsilon^2 \cdot \min\{s, 0.1\}))$ times the number of oracle calls, respectively.

Note that such an oracle is virtually equivalent to a perfect DH-oracle for our application because the correctness of the output of a probabilistic discrete logarithm algorithm can be tested, and because only a polynomial number of oracle calls is required for the computation of a discrete logarithm.

Remark: Examples of ε-DH-oracles which can *not* be transformed into perfect oracles with our method when $|G|$ is unknown are those which answer the input (g^u, g^v) by one of the values $g^{uv+i|G|/z}$, where $z \leq 1/\varepsilon$ is a factor of $|G|$ and where all the values of i between 0 and $z-1$ are equally likely. If $|G|$ is known, the correct one of the z candidates can be found by $O((\log|G|)^2/\varepsilon + \log|G|/\varepsilon^2)$ group operations, as described in Appendix A.

3.3 The squaring oracle

We present an example of an oracle that is weaker than an ε-DH-oracle because the fraction of correctly answered inputs is smaller than a constant fraction. Nevertheless, the oracle turns out to be as strong as the perfect oracle. We call an oracle that answers the input g^u by $g^{(u^2)}$ (where u and u^2 are in $\mathbf{Z}_{|G|}$) a *squaring-DH-oracle*. Note that this is not an ε-DH-oracle for any constant $\varepsilon > 0$ because only one out of $|G|$ inputs is answered correctly, and this fraction vanishes with increasing $|G|$.

Let g^u and g^v be given. One can compute $g^{u+v} = g^u \cdot g^v$ and

$$g^{(u+v)^2} \cdot \left(g^{(u^2)}\right)^{-1} \cdot \left(g^{(v^2)}\right)^{-1} = g^{(u+v)^2-u^2-v^2} = g^{2uv} = (g^{uv})^2 . \tag{1}$$

When given $|G|$, square roots in G can efficiently be computed. If $|G|$ is odd, the square root is unique, but if $|G|$ is even, there exist two square roots,

$$g^{uv} \quad \text{and} \quad g^{uv+\frac{|G|}{2}}$$

which can be computed by a method of Massey [22] (see also Lemma 2). Let $|G|$ be even and let 2^e be the maximal power of 2 dividing $|G|$. From g^u and g^v, one can compute u and v modulo

2^e with $O((\log |G|)^2)$ group operations by the Pohlig-Hellman method [30], and hence one can obtain $u \cdot v$ modulo 2^e. Because $|G|/2$ is not a multiple of 2^e, we have

$$uv \not\equiv uv + \frac{|G|}{2} \pmod{2^e},$$

and one can determine the correct square root g^{uv} by computing the discrete logarithms of the two square roots modulo 2^e. Hence a squaring-DH-oracle is equally powerful as a perfect DH-oracle in a group G whose order is known.

A probabilistic squaring-DH-oracle for a group with known order that answers correctly only with probability ε (ε-squaring-DH-oracle) can be transformed into a translation-invariant ε^3-DH-oracle by randomizing the inputs in (1). The required complexity is $O((\log |G|)^2)$ group operations per call. This proves the following theorem.

Theorem 2 *For every cyclic group G with generator g and known order $|G|$ and for every $\beta > 0$ there exists a DH-oracle algorithm which makes calls to an ε-squaring-DH-oracle and whose answer is correct with probability at least $1-\beta$. The number of oracle calls is $O(\log(1/\beta\varepsilon^3)/\varepsilon^{12})$. The number of required group operations is $(\log |G|)^2$ times the number of oracle calls.*

3.4 The security of subgroups

In this section we assume that the order of G is known. We address the question whether a subgroup is more or less secure than the entire group with respect to the DH protocol. Although the statement of Corollary 5 below is very intuitive (and an analogous result holds trivially for the computation of discrete logarithms), the proofs of Theorems 3 and 4 are not trivial. First we state that a subgroup of G with smooth index is at most as secure as G. We need the following lemma on the computation of roots in cyclic groups.

Lemma 2 *Let G be a cyclic group with generator g, and let p be a prime divisor of $|G|$. One of the p-th roots of a p-th power in G can be computed in time $O((\log |G|)^2 + p \log |G|)$.*

Proof: The square root algorithm of Massey [22] can be generalized as follows. Let $|G| = p^j s$ (where $j \geq 1$ and $(p,s) = 1$), and let h be a p-th power in G. By the method of Pohlig and Hellman [30] we can compute the remainder k of the discrete logarithm of h to the base g with respect to p^j. Note that k is a multiple of p because h is a p-th power. Let $d \equiv -s^{-1} \pmod{p}$. The element

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

is a p-th root of h. This algorithm requires $O((\log |G|)^2 + p \log |G|)$ operations in G. □

Theorem 3 *Let G be a cyclic group with generator g, and let B be a smoothness bound, polynomial in $\log |G|$. For every B-smooth divisor r of $|G|$ there exists a DH-oracle algorithm for the group $\langle g^r \rangle$ which makes one call to the DH-oracle for $\langle g \rangle$ and uses a polynomial number of group operations per call.*

Proof: Let $r = \prod_{i=1}^{s} p_i^{f_i}$, and let $p_i^{e_i}$ be the maximal powers dividing $|G|$ for $i = 1, \ldots, s$. The oracle for G answers the input (g^{ra}, g^{rb}) by $g^{r^2 ab} = g^{p_1^{2f_1} \cdots p_s^{2f_s} ab}$. We obtain $g^{rab} = g^{p_1^{f_1} \cdots p_s^{f_s} ab}$ by computing p_i-th roots and deciding immediately which of the p_i different roots is the correct one. For fixed i and for some $k = 2f_i - 1, 2f_i - 2, \ldots, f_i$, assume that we have already computed

$$g^{p_1^{f_1} \cdots p_{i-1}^{f_{i-1}} \cdot p_i^{k+1} \cdot p_{i+1}^{2f_{i+1}} \cdots p_s^{2f_s} ab} = g^{cp_i^{k+1} ab} ,$$

where $c = p_1^{f_1} \cdots p_{i-1}^{f_{i-1}} \cdot p_{i+1}^{2f_{i+1}} \cdots p_s^{2f_s}$ is explicitly known. According to the above lemma we can compute the p_i-th roots

$$g^{cp_i^k ab + j \cdot \frac{|G|}{p_i}} ,\ j = 0, \ldots, p_i - 1 .$$

Because a and b can be obtained modulo $p_i^{e_i - f_i}$ directly from g^{ra} and g^{rb} by the method of Pohlig and Hellman [30] and c is explicitly known, and because $k \geq f_i$, we can compute $cp_i^k ab$ modulo $p_i^{e_i}$. We have $j \cdot |G|/p_i \equiv 0 \pmod{p_i^{e_i}}$ only for $j = 0$, and the correct root can be determined by computing the discrete logarithms of the candidates modulo $p_i^{e_i}$, using the Pohlig-Hellman method. Finally, we obtain g^{rab}. The running time is polynomial in $\log |G|$ if r is B-smooth. □

Conversely, in many cases a DH-oracle for a subgroup of G or a set of such oracles can be transformed into a DH-oracle for the entire group, and the following theorem gives a criterion for when this is the case. The proof is given in the next section as a first application of our concept of computing with implicit representations.

Theorem 4 *Let G be a cyclic group with generator g and order $|G| = \prod_{i=1}^r p_i^{e_i}$, and let B be a smoothness bound which is polynomial in $\log |G|$. If for certain s_j there exist DH-oracles for the subgroups $G_j := \langle g^{s_j} \rangle$ $(j = 1, \ldots, t)$, and if for all $p_i > B$ there exists j such that p_i does not divide s_j, then there exists a polynomial-time DH-oracle for G with respect to g which calls each subgroup oracle at most $\log |G| / \log B$ times.*

The following result is an immediate consequence of the above theorems.

Corollary 5 *Consider a group $G = \langle g \rangle$ and a subgroup $H = \langle g^k \rangle$ of G with smooth index k. The DH problem for H is polynomial-time equivalent to the DH problem for G.*

4 A general technique for obtaining equivalence results

Let G be a cyclic group generated by g for which the prime factorization of the order $|G|$ is known, and let $a = g^s$ be a given group element for which the discrete logarithm s should be computed using a DH-oracle for G. This can be achieved by computing s modulo each of the prime factors of $|G|$ (or modulo the prime powers if $|G|$ contains multiple prime factors) and combining these values by Chinese remaindering. Only large prime factors are relevant because the Pohlig-Hellman algorithm allows to compute s modulo powers of small prime factors of $|G|$.

In the sequel we consider the problem of computing s modulo p for a large prime factor p of $|G|$ (the atypical case of $|G|$ having multiple large prime factors will be considered later). Let x be the element of $GF(p)$ such that $s \equiv x \pmod{p}$. The structure of the finite field $GF(p)$ will be crucial in the arguments below.

4.1 Computations on implicit representations using a DH-oracle

Every element y of the field $GF(p)$ corresponds to an equivalence class of elements of G (consisting of those whose discrete logarithm is congruent to y modulo p). Any member a of the equivalence class is called an *implicit representation* of y and, conversely, y is called implicitly represented by a. We write

$$y \rightsquigarrow a.$$

The following operations on elements of $GF(p)$ can be performed on their implicit representations, where the result is also obtained only in an implicit representation. Let y and z be elements of $GF(p)$, with

$$y \rightsquigarrow a \ , \quad z \rightsquigarrow b \ .$$

Because

$$y = z \ \text{ if and only if } \ a^{|G|/p} = b^{|G|/p} \ ,$$

equality of two implicitly represented elements of $GF(p)$ can be tested by $O(\log |G|)$ group operations. Furthermore we have

$$\begin{aligned} y + z &\rightsquigarrow a \cdot b \\ yz &\rightsquigarrow \mathrm{DH}(a, b) \\ -y &\rightsquigarrow a^{-1} = a^{|G|-1}, \end{aligned}$$

and these implicit operations in $GF(p)$ require a single group operation in G, a call to the DH-oracle and $O(\log |G|)$ group operations, respectively.

In order to simplify the notation, we also introduce the notion of a power-DH-oracle (PDH_e) that computes an implicit representation of the e-th power of an implicitly represented element. A possible implementation of a PDH_e-oracle is to use a (fixed) algorithm for computing powers in a group (e.g., 'square and multiply') for obtaining an implicit representation of y^e, denoted by $\mathrm{PDH}_e(a)$, by $O(\log e)$ calls to a normal DH-oracle (remember that $y \rightsquigarrow a$). In particular we can compute inverses of implicitly represented elements because

$$y^{-1} \rightsquigarrow \mathrm{PDH}_{p-2}(a) \ .$$

We call addition, subtraction, multiplication, division and equality testing in $GF(p)$ *algebraic* operations. Any computation in $GF(p)$ can be performed on implicit representations whenever it makes use only of algebraic operations. Examples are the evaluation of a rational function, testing quadratic residuosity of y by comparing

$$(\mathrm{PDH}_{(p-1)/2}(a))^{|G|/p} \text{ and } \ g^{|G|/p},$$

or the computation of square roots using the algorithm of Peralta [29] or a faster method due to Massey [22]. We will crucially rely on the fact that algorithms based on exhaustive search (for example to solve the index search problem, in particular the discrete logarithm problem) can be executed on implicitly represented arguments.

We now prove Theorem 4 as an application of computations with implicit representations. Let g^u and g^v be given. We compute g^{uv} by using the available oracles for subgroups. Let $m_i := p_i^{e_i}$, $M_i := |G|/m_i$ and $N_i :\equiv M_i^{-1} \pmod{m_i}$. For prime factors $p_i \leq B$, u and v, and hence also uv, can be computed in polynomial time modulo m_i by the Pohlig-Hellman method [30]. For a prime factor $p_i > B$ let j be such that p_i does not divide s_j. We apply the oracle for G_j to $(g^{s_j})^u = (g^u)^{s_j}$ and $(g^{s_j})^v$ to obtain $(g^{s_j})^{u \cdot v}$, where u, v and $u \cdot v$ are modulo $|G|/s_j$. Because s_j divides M_i, we can compute

$$U_i := g^{M_i \cdot (u \cdot v)} = \left(g^{s_j (u \cdot v)}\right)^{\frac{M_i}{s_j}} ,$$

where $u \cdot v$ is modulo m_i. Finally, g^{uv} is computable by Chinese remaindering with implicitly represented arguments by applying only group operations in G:

$$g^{uv} = g^{\sum_i M_i N_i (u \cdot v)} = \prod_i U_i^{N_i}.$$

□

4.2 Auxiliary groups

When given a DH-oracle for G, the computation of x is shown to work efficiently if an auxiliary group H over $GF(p)$ with certain properties is given. (Remember that $s \equiv x \pmod p$, where p is a large prime factor of $|G|$, and that s is the discrete logarithm we want to compute.) The basic idea is to embed the unknown x into an implicitly represented element of H and to compute the discrete logarithm of this element explicitly. We now define some of the required properties of the auxiliary group H.

Definition 3 A finite (additively written) group H is said to be *defined algebraically over* $GF(p)$ if, for some m, the elements of H can be represented as m-tuples of elements of $GF(p)$ and if the group operation in this representation can be carried out by a polynomial number of algebraic operations in $GF(p)$. We say that H has the *algebraic embedding property* if, when given $x \in GF(p)$, an element $c \in H$ can be constructed by a polynomial number of algebraic operations in $GF(p)$ such that x can be computed efficiently when given c.

In what follows, we neglect the complexity of the computation of x from c. In the auxiliary groups used in the sequel, x is embedded simply as one of the coordinates of the m-tuple, possibly shifted by a known offset.

It will be shown that an abelian group H with bounded rank, defined algebraically over $GF(p)$, with the algebraic embedding property and smooth order can be used as auxiliary group to reduce the computation of discrete logarithms modulo p in G to breaking the DH protocol for G. Examples of such auxiliary groups are elliptic curves or subgroups of finite extension fields, and will be discussed in Section 6.

We assume that an auxiliary group H is given with these properties and B-smooth order. We treat the cases of cyclic and general auxiliary groups separately.

4.3 Cyclic auxiliary groups

Let the auxiliary group H be cyclic with generator h. We extend the definition of implicit representations from elements of $GF(p)$ to m-tuples over $GF(p)$ in a natural way. We say that $(a_1, \ldots, a_m)$ is an implicit representation of $(y_1, \ldots, y_m)$ if and only if $y_i \rightsquigarrow a_i$ for $1 \le i \le m$. Using the algebraic embedding property of H, the implicit representation of a group element $c \in H$ can be computed by application of a DH-oracle for G such that x can efficiently be computed from the *explicit* coordinates of c.

We now address the problem of finding c explicitly when given its implicit representation. This is achieved by computing the discrete logarithm k of c in H to the base h, which is possible efficiently because $|H|$ is assumed to be B-smooth and H is defined algebraically over $GF(p)$. The discrete logarithm algorithm of Section 2 uses only group operations and equality testing of group elements. The (explicit) discrete logarithm k of the implicitly represented group element c of h can hence be computed by

$O(m^2(\log p)^2)$ operations in H with implicitly represented elements,

$O\left(m^2 \frac{B}{\log B} \log p \log |G|\right)$ operations in G, and

$O\left(m^2(\log p)^2 + m \frac{B}{\log B} \log p\right)$ explicit operations in H.

Note that p^m is an upper bound for $|H|$ because H is defined algebraically over $GF(p)$. Using the time-memory tradeoff, the complexities are

$O\left(m^2(\log p)^2 + m \frac{\sqrt{B}}{\log B} \log p\right)$ operations in H with implicitly represented elements,

$O\left(m^2 \frac{\sqrt{B}}{\log B} \log p \log |G|\right)$ operations in G,

$O\left(m^2(\log p)^2 + m \frac{\sqrt{B}}{\log B} \log p\right)$ explicit operations in H, and

$O(m\sqrt{B} \log p)$ steps for sorting the memory.

Given k, first compute $c = k \cdot h$, and then x.

4.4 General auxiliary groups

The method also works if H is not cyclic. We assume that H is abelian of rank r, i.e., H is isomorphic to $\mathbf{Z}_{n_1} \times \cdots \times \mathbf{Z}_{n_r}$ for some $n_1, \ldots, n_r$ satisfying $\prod_{j=1}^r n_j = |H|$ and such that n_{j+1} divides n_j for $j = 1, \ldots, r-1$ (this representation is unique, see [18]). Let $h_1, \ldots, h_r$ be a set of generators of H such that H is the internal product of the cyclic subgroups $\langle h_1 \rangle, \ldots, \langle h_r \rangle$:

$$H = \langle h_1 \rangle \times \cdots \times \langle h_r \rangle.$$

If no generator set for H is known it can be computed by a method described in Appendix B.

Let c be the element of H into which x is embedded, and of which an implicit representation is known. The element $c \in H$ has a unique representation

$$c = \sum_{j=1}^{r} k_j h_j, \qquad 0 \leq k_j < n_j \ .$$

We compute the coefficients k_j. The following is repeated for every prime q dividing $|H|$. We describe the first and second iteration step of an algorithm that computes k_j modulo the highest power of q dividing n_j for all $j = 1, \ldots, r$. The algorithm uses v_j ($j = 1, \ldots, r$) as local variables (initialized by $v_j \leftarrow 0$).

For the first step, let α_1 be the number of generators h_j whose order contains the same number of factors q as n_1. In other words, $(n_1/q)h_j$ is different from the unity e of H exactly for $j = 1, \ldots, \alpha_1$. Because H is algebraically defined over $GF(p)$, an implicit representation of

$$\frac{n_1}{q} c$$

can be computed from the implicit representation of c by $O(\log |H|)$ operations in H with implicitly represented elements. For all $(t_1, \ldots, t_{\alpha_1}) \in \{0, \ldots, q-1\}^{\alpha_1}$, we compute (explicitly)

$$\frac{n_1}{q} t_1 h_1 + \cdots + \frac{n_1}{q} t_{\alpha_1} h_{\alpha_1},$$

transform the coordinates into implicit representations and compare the points with $(n_1/q)c$. Equality indicates that the t_j are congruent to the coefficients k_j modulo q. We set $v_j \longleftarrow t_j$ for $1 \leq j \leq \alpha_1$.

For the second step, let α_2 be the number of points h_j whose order contains at most one factor q less than n_1, i.e., $(n_1/q^2)h_j \neq e$ for $j = 1, \ldots, \alpha_2$. Implicit representations of the points

$$\frac{n_1}{q^2}(v_1 q + t_1)h_1 + \cdots + \frac{n_1}{q^2}(v_{\alpha_1} q + t_{\alpha_1})h_{\alpha_1} + \frac{n_1}{q^2} t_{\alpha_1+1} h_{\alpha_1+1} + \cdots + \frac{n_1}{q^2} t_{\alpha_2} h_{\alpha_2}$$

are computed for all $(t_1, \ldots, t_{\alpha_2}) \in \{0, \ldots, q-1\}^{\alpha_2}$ until equality with the implicitly represented point

$$\frac{n_1}{q^2} c$$

holds. Then assign

$$v_j \longleftarrow v_j q + t_j \quad (j = 1, \ldots, \alpha_1),$$
$$v_j \longleftarrow t_j \quad (j = \alpha_1 + 1, \ldots, \alpha_2).$$

After repetition of this process up to the maximal q-power q^g dividing n_1, the resulting v_j satisfy

$$\frac{n_1}{q^g} c = \sum_{j=1}^{r} \frac{n_1}{q^g} v_j h_j \ ,$$

i.e., k_j is congruent to v_j modulo the highest power of q dividing $n_j = \operatorname{ord} h_j$ for $j = 1, \ldots, r$.

The algorithm can be sped up by a time-memory tradeoff if memory space of size $M = S^r$ is available. We describe the tradeoff in detail only for the first iteration step. Here the implicit representations of

$$\frac{n_1}{q} c + \sum_{j=1}^{\alpha_1} \frac{n_1}{q} u_j h_j \quad (0 \leq u_j < S)$$

are computed and sorted ('baby-step'). Then it is checked by exhaustive search which of the elements

$$\sum_{j=1}^{\alpha_1} \frac{n_1}{q} t_j S h_j \quad (0 \leq t_j < \frac{q}{S})$$

is contained in the list in an implicit representation ('giant-step').

After running the algorithm for all primes q dividing $|H|$, one can compute the coefficients k_j modulo $\operatorname{ord} h_j$ by Chinese remaindering, and x can be obtained by computing c explicitly. The complexity of the computation of x is

$O(m^2(\log p)^2)$ operations in H with implicitly represented elements,

$O\left(m^2 \frac{B^r}{r \log B} \log p \log |G|\right)$ operations in G, and

$O\left(m^2 r(\log p)^2 + m \log p \frac{B^r}{\log B}\right)$ explicit operations in H.

The complexities when the time-memory tradeoff with $S \approx \sqrt{q}$ is applied are

$O\left(m^2(\log p)^2 + m \log p \frac{\sqrt{B^r}}{r \log B}\right)$ operations in H with implicitly represented elements,

$O\left(m^2 \frac{\sqrt{B^r}}{r} \log p \log |G|\right)$ operations in G,

$O\left(m^2r(\log p)^2 + m\log p\frac{\sqrt{B^r}}{\log B}\right)$ explicit operations in H, and

$O\left(\frac{m\sqrt{B^r}}{r}\log p\right)$ steps for sorting the memory.

(For $r = 1$ these are the complexities for the case of a *cyclic* auxiliary group.) Because H is defined algebraically over $GF(p)$, the running time is polynomial if B is polynomial in $\log|G|$, and if $r = O(1)$.

4.5 Summary and generalization to multiple prime factors in $|G|$

In the previous section, we have assumed that all the large prime factors of $|G|$ are single. Additional conditions are required in the non-typical case of multiple large prime factors of $|G|$. If p^e divides $|G|$ (with $e > 1$), the discrete logarithm s must be computed explicitly modulo p^e instead of modulo p. This can be done, as described below, if either an additional DH-oracle for a certain subgroup of G is available, or if p-th roots can efficiently be computed in G.

We write $x \equiv \sum_0^{e-1} x_i p^i \pmod{p^e}$ with $x_i \in GF(p)$ for $i = 0, \ldots, e-1$. Let $k \leq e-1$, assume that $x_0, \ldots, x_{k-1}$ are already computed and consider the problem of computing x_k. Let $a' := a \cdot g^{-x_0 - \cdots - x_{k-1}p^{k-1}}$. Then

$$a' = \left(g^{p^k}\right)^{x_k + p\cdot l}$$

for some l. From a', x_k can be obtained in either of two ways: If a DH-oracle for one of the subgroups $\langle g^{d\cdot p^{e-1}}\rangle$, where $d \cdot p^{e-1}$ divides $|G|/p$, is available, then x_k can be computed from

$$(a')^{d\cdot p^{e-1-k}} = \left(g^{d\cdot p^{e-1}}\right)^{x_k + p\cdot l}$$

by use of this oracle as described in the previous section (see also [4]). Alternatively, assume that p-th roots can be computed in G. If $a'' := g^{x_k + p\cdot l'}$ (for some l') is computed first, x_k can be obtained as usual. In order to get a'', it suffices to compute the p^k-th root (k times the p-th root) of a'. Any p^k-th root of a' is of the required form because p divides $|G|/p^k$.

It is unknown whether a DH-oracle can be transformed into a DH-oracle for a general subgroup of G, and whether p-th roots in G can be computed efficiently for a large prime factor p of $|G|$. We summarize our results.

Theorem 6 *Let G be a cyclic group with generator g, and let B be a smoothness bound, polynomial in $\log|G|$. Assume that $|G|$ and its factorization $|G| = \prod_{i=1}^{s} p_i^{e_i}$ are known, that every prime factor p of $|G|$ greater than B is single and that for every such p, a finite abelian group H_p with rank $r = O(1)$, algebraically defined over $GF(p)$ and with the algebraic embedding property, is given whose order $|H_p|$ is B-smooth and known or computable in time polynomial in $\log p$. Then breaking the Diffie-Hellman protocol for G with respect to g is polynomial-time equivalent to computing discrete logarithms in G to the base g.*

The complexity of the computation of a discrete logarithm modulo p in G is $O(m^2(\log p)^2)$ operations in H_p with implicitly represented elements, $O(m^2B^r \log p \log|G|/\log B)$ group operations in G and $O(m^2(\log p)^2 + m\log p \cdot B^r/\log B)$ explicit operations in H_p.

In case of a multiple prime factor p greater than B, that is if p^e divides $|G|$ for some $e > 1$, the desired equivalence holds with respect to a DH-oracle for one of the subgroups $\langle g^{d\cdot p^{e-1}}\rangle$ (instead of the DH-oracle for G), where $d\cdot p^{e-1}$ divides $|G|/p$, or if a polynomial-time algorithm for computing p-th roots in G is available. □

The complexities stated in the theorem can be reduced by a time-memory tradeoff.

5 Applicable auxiliary groups over $GF(p)$

In this section we show that elliptic curves over $GF(p)$ and over an extension field $GF(p^n)$ and certain subgroups of extension fields are suitable auxiliary groups H_p. In [38] the applicability of Jacobians of hyperelliptic curves (see [17] and [8]) is demonstrated (additional problems arise here because the representation of elements of Jacobians, which are equivalence classes, is not unique).

We will give a list of expressions $A(p)$ in p such that an auxiliary group H_p with order $A(p)$ over $\mathbf{F}_p$ can efficiently be constructed. Theorem 6 implies that if for each prime factor p of $|G|$ one of the expressions in this list is smooth, then breaking the DH protocol and computing discrete logarithms are equivalent for G (if $|G|$ has no multiple large prime factors). Den Boer has shown in [9] that the list contains $A(p) = p - 1$ with $H_p = \mathbf{F}_p^*$.

5.1 Elliptic curves

5.1.1 Applicability

Let $\mathbf{F}$ be a field. The elliptic curve $E_{A,B}(\mathbf{F})$ (with parameters A and B in $\mathbf{F}$) is the set

$$\{(x, y) \in \mathbf{F}^2 : y^2 = x^3 + Ax + B\} \cup \{\mathcal{O}\}$$

(the additional point is called 'point at infinity'). The *projective* representation of (x, y) is $(x : y : 1) = (rx : ry : r)$ (for $r \neq 0$) and $\mathcal{O} = (0 : 1 : 0)$. Together with a certain addition of points, $E_{A,B}(\mathbf{F})$ forms an abelian group of rank at most 2. We refer to [28] for an introduction to elliptic curves.

We show that an elliptic curve E over the field $\mathbf{F}_p$ is defined algebraically over $\mathbf{F}_p$ and has the algebraic embedding property. Therefore it can be applied as the group H_p in Theorem 6 if it has smooth order. Note that the order of an elliptic curve can be computed in polynomial time [36], [5].

E is defined over $\mathbf{F}_p$ (with $m = 2$, that is the points of E can be represented as pairs of $\mathbf{F}_p$-elements), and the group operation can be executed by a constant number of additions, multiplications, divisions and equality tests in $GF(p)$. From the implicit representations (a_1, a_2) and (b_1, b_2) of two points P and Q of E, an implicit representation of the sum $P + Q$ is thus computable by $O(\log |G|)$ group operations (for the equality test) and $O(\log p)$ calls to the DH-oracle (for the inversions). The group operation in projective coordinates requires no divisions, but the conversion from projective to affine coordinates of implicitly represented points has a complexity of $O(\log p)$ oracle calls and is required if equality of elements of E has to be tested.

Elliptic curves over $\mathbf{F}_p$ have the algebraic embedding property. We show directly that, given a with $x \rightsquigarrow a$, one can compute an implicit representation b of y such that $(x + d, y) \in E$ for some known d. First compute c with

$$x^3 + Ax + B \rightsquigarrow c$$

by $O(\log p)$ group operations in G and two calls to the oracle. One can test quadratic residuosity of $x^3 + Ax + B \in GF(p)$ by $O(\log |G|)$ operations in G and $O(\log p)$ oracle calls. If it is not a quadratic residue, x can be replaced by $x + d$ ($d = 1, 2, \ldots$).

Using Massey's algorithm [22] on implicit representations, the computation of an implicit representation of a square root of $x^3 + Ax + B$ requires $O((\log p)^2)$ calls to the oracle and $O(\log p \log |G|)$ group operations for the equality tests. (The generation of an *explicit* random point on the elliptic curve takes $O((\log p)^2)$ multiplications in $\mathbf{F}_p$.)

One can show in a completely analogous manner that an elliptic curve over an extension field $\mathbf{F}_{p^n}$ of $\mathbf{F}_p$ can also be used in Theorem 6. It will shown later that computing with implicitly represented elements of $\mathbf{F}_{p^n}$ is possible.

5.1.2 Existence

It is well-known that

$$p - 2\sqrt{p} + 1 \leq |E_{A,B}(\mathbf{F}_p)| \leq p + 2\sqrt{p} + 1,$$

and that for each $d \in [p - 2\sqrt{p} + 1, p + 2\sqrt{p} + 1]$, there exists an elliptic curve over $GF(p)$ with order d. Very little is known about the existence of smooth numbers in the interval of interest for a given prime p. However, it is known [7] that for every fixed u,

$$\psi(n, n^{1/u})/n = u^{-(1+o(u))u} \tag{2}$$

where $\psi(n, y)$ denotes the number of integers $\leq n$ with no prime divisor $\geq y$. This fact suggests that for some $c > 1$ every integer n has the property P_c defined below.

Definition 4 An integer n has *property* P_c if there exists an integer d in the interval

$$[n - 2\sqrt{n} + 1, n + 2\sqrt{n} + 1]$$

with no prime divisor greater than $n^{c/u}$, where u is defined by $u^{2u} = n$.

Conjecture There exists a constant $c > 1$ such that all n have property P_c.

An even stronger conjecture is that the above conjecture holds for any $c > 1$ and sufficiently large n. For example, let n be a 100-digit number and note that $10^{100} \approx 33^{66}$. Hence, one can expect to find an integer in the interval $[n - 10^{50}, n + 10^{50}]$ with no prime divisor greater than $10^{c \cdot 100/33}$ for some c. For $c = 1.1$ and $c = 2$ this bound is approximately 2000 and 10^6, respectively.

The above conjecture implies the existence of an appropriate group H_p for each prime number p. Therefore for every cyclic group G there exists a small piece of information, which depends on the order of G, that makes breaking DH and computing discrete logarithms equivalent in G. This information is a string S consisting of the prime factors p_i of $|G|$ and appropriate elliptic curve parameters for all p_i.

Corollary 7 *If the stated number-theoretic conjecture is true, then for every group $G = \langle g \rangle$, there exists a side information string S of length at most $3 \log |G|$ such that when given S, breaking the Diffie-Hellman protocol for G and base g is polynomial-time equivalent to computing discrete logarithms in G to the base g. (In the case of a multiple large prime factor p of $|G|$, the equivalence holds with respect to breaking the DH protocol in one of a certain subset of subgroups of G, or if an algorithm for computing p-th roots in G is given.)* □

It need not be assumed that $|G|$ and its factorization are known because this information is contained in S.

Remark: The group order of Jacobians of hyperelliptic curves of genus 2 varies in a larger interval of size $[n - \Theta(n^{3/4}), n + \Theta(n^{3/4})]$, but the more detailed results about the distribution of the orders which are proved in [1] are not sufficient to prove the existence of the side information string without conjecture. The reason is that in [1] the existence of Jacobians with *prime* order is proved, whereas in the above corollary Jacobians with *smooth* order are required.

5.1.3 Construction of elliptic curves

For certain expressions $A(p)$, elliptic curves over $\mathbf{F}_p$ with order $A(p)$ can explicitly be constructed. The curve over $\mathbf{F}_p$ defined by the equation

$$y^2 = x^3 - Dx \tag{3}$$

has order $p+1$ if $p \equiv 3 \pmod 4$, and the curve

$$y^2 = x^3 + D \tag{4}$$

has also order $p+1$ if $p \equiv 2 \pmod 3$. Thus if $p \not\equiv 1 \pmod{12}$, elliptic curves of order $p+1$ are explicitly constructable. We will show later that the subgroup of order $p+1$ of $\mathbf{F}^*_{p^2}$ is a useful auxiliary group for all p. The following statements about the orders of curves of the form (3) or (4) in the case they are *not* $p+1$ are proved in [16].

If $p \equiv 1 \pmod 4$, then p can uniquely be represented as a product in the ring $\mathbf{Z}[i]$ of Gaussian integers as follows:

$$p = \pi\overline{\pi} = (a+bi)(a-bi) = a^2 + b^2, \quad \pi \equiv 1 \pmod{2+2i}.$$

The curves $y^2 = x^3 - Dx$ have the orders

$$p+1 \pm 2a, \quad p+1 \pm 2b, \tag{5}$$

and the four orders occur equally often.

Let $\omega := (-1+\sqrt{-3})/2$. If $p \equiv 1 \pmod 3$, then p can uniquely be represented as a product in the ring $\mathbf{Z}[\omega]$ as follows:

$$p = \pi\overline{\pi} = (a+b\omega)(a-b\omega) = a^2 - ab + b^2, \quad \pi \equiv 2 \pmod 3.$$

The curves $y^2 = x^3 + D$ have the orders

$$p+1 \pm 2a, \quad p+1 \pm a \mp 2b, \quad p+1 \pm (a+b), \tag{6}$$

and the six orders occur equally often.

If $p \equiv 1 \pmod 4$ or $p \equiv 1 \pmod 3$, curves with the orders listed in (5) and (6) are explicitly constructable by varying D.

5.2 Subgroups of the multiplicative group of an extension field over $GF(p)$

5.2.1 Representation

We refer to [21] or [27] for an introduction to finite fields. The group $\mathbf{F}^*_{p^n}$ and hence every subgroup is cyclic. The field $\mathbf{F}_{p^n}$ is an n-dimensional vector space over $\mathbf{F}_p$ and its elements can be represented as n-tuples of $\mathbf{F}_p$-elements with respect to some basis. Let α be an element of $\mathbf{F}_{p^n}$. Let $\alpha_i := \alpha^{p^i}$ for $i = 0, \ldots, n-1$. $\{\alpha_0, \ldots, \alpha_{n-1}\}$ is called a *normal basis* if it is linearly independent in which case α is called a *normal element*. Let $\vec{\alpha} := (\alpha_0, \ldots, \alpha_{n-1})$. The matrix T in $(\mathbf{F}_p)^{n\times n}$ satisfying $\alpha_0\vec{\alpha} = T\vec{\alpha}$ is called the *multiplication table* of the basis.

Normal bases can be found efficiently by trial and error because, if $n \le p^4$, the probability that a randomly chosen $\alpha \in \mathbf{F}_{p^n}$ is normal is at least $1/34$ (see [27]), and there is a test for normality: α is normal if and only if

$$\gcd(x^n - 1, \alpha^{p^{n-1}}x^{n-1} + \cdots + \alpha^p x + \alpha) = 1.$$

The multiplication table can be determined by solving a system of linear equations over $\mathbf{F}_p$. Let H be a subgroup of $\mathbf{F}^*_{p^n}$. The group operation in H is a multiplication in $\mathbf{F}^*_{p^n}$ and requires $O(n^3)$ multiplications in $\mathbf{F}_p$.

5.2.2 The algebraic embedding property

Membership in H can be characterized by an equation over $\mathbf{F}_{p^n}$. Let β be an element of $\mathbf{F}_{p^n}$. Because $\mathbf{F}_{p^n}^*$ is cyclic, β belongs to the subgroup H if and only if $\beta^{|H|} = 1$. The element β can be represented by its coordinates $(y_0, y_1, \ldots, y_{n-1})$ (with $y_i \in \mathbf{F}_p$) in the normal basis, i.e.,

$$\beta = \sum_{i=0}^{n-1} y_i \alpha_i. \tag{7}$$

In this representation the characteristic equation of H is equivalent to a system of n polynomial equations in the y_i. The polynomials depend on the multiplication table.

For some orders $|H|$, the polynomials can easily be computed and have small degree, in particular when $|H|$ is a sum of p-powers, multiplied with only small factors. The p^u-th power of the sum in (7) is equal to the sum of the p^u-th powers of the summands because $\mathbf{F}_{p^n}$ has characteristic p. In addition we have $y_i^p = y_i$ and $\alpha_i^p = \alpha_{i+1}$ (where the index is reduced modulo n). Hence β^{p^u} is represented by the coordinates $(y_{n-u}, y_{n-u+1}, \ldots, y_n, y_0, \ldots, y_{n-u-1})$.

We prove the algebraic embedding property by showing directly that, given an implicit representation of x, an implicit representation of a point β of H can be computed such that x (or $x + d$) is one of the coordinates of β. To do this, we fix some of the other coordinates (for example by assigning the value 0) and solve the implicitly represented equations to get implicitly represented values for the remaining coordinates such that β belongs to H. The number of unknowns over $\mathbf{F}_p$ in this system depends on the cardinality of H. If we solve for k different $\mathbf{F}_p$-coordinates simultaneously, then the expected number of trials until an element of H is found is $p^{n-k}/|H|$.

5.2.3 The case of one equation and one unknown

It is much easier to solve a univariate polynomial than to solve a system of multivariate polynomials. We give a condition under which it suffices to solve for only one unknown even when $|H|/p^n \approx 1/p^k$ for $k > 1$. If k divides n, then $\mathbf{F}_{p^n}$ is an extension field of $\mathbf{F}_{p^k}$. Let $l := n/k$, and let $\{\alpha'_0, \ldots, \alpha'_{l-1}\}$ be a normal basis of $\mathbf{F}_{p^n}$ over $\mathbf{F}_{p^k}$. An element β of $\mathbf{F}_{p^n}$, represented by $(\beta'_0, \ldots, \beta'_{l-1})$ with $\beta'_i \in \mathbf{F}_{p^k}$, belongs to H if and only if $(\sum_{i=0}^{l-1} \beta'_i \alpha'_i)^{|H|} = 1$. As mentioned above, the polynomials are easily computable and are of small degree if $|H|$ is a sum of powers of p^k, multiplied with small factors. When computing in $\mathbf{F}_{p^k}$ instead of $\mathbf{F}_p$ one can solve for k unknowns of $\mathbf{F}_{p^k}$ at once, and the cardinality of H must not be much smaller than p^{n-k}.

For certain $|H|$ only one equation over $\mathbf{F}_{p^k}$ arises. This is the case when $\beta^{|H|} \in \mathbf{F}_{p^k} \subset \mathbf{F}_{p^n}$ for all β. Because $\alpha_0 + \alpha_1 + \cdots + \alpha_{l-1}$ (the trace of α_0, denoted by $\mathrm{Tr}(\alpha_0)$) is an element of $\mathbf{F}_{p^k}$, all the coefficients are then automatically equal, and it suffices to solve one instead of l equations. The condition is

$$\left(\beta^{|H|}\right)^{p^k - 1} = 1 \quad \text{for all} \quad \beta \,,$$

or, equivalently, that $p^n - 1$ divides $(p^k - 1) \cdot |H|$, which holds if

$$|H| = p^{n-k} + p^{n-2k} + \cdots + p^k + 1. \tag{8}$$

(The case $|H| = p^n - 1$ is not interesting, because smoothness of $p^n - 1$ implies smoothness of $p - 1$.) The characteristic equation of the subgroup H with order (8) is

$$\left(\sum_{i=0}^{l-1} \beta'_i \alpha'_{i+l-1}\right) \cdot \left(\sum_{i=0}^{l-1} \beta'_i \alpha'_{i+l-2}\right) \cdots \left(\sum_{i=0}^{l-1} \beta'_i \alpha'_i\right) = 1$$

(where the indices are reduced modulo l), and the equation from the first coefficient is an l-degree polynomial equation in $\beta'_0, \ldots, \beta'_{l-1}$ over $\mathbf{F}_{p^k}$. We assign the (implicitly represented) x to one of the k coordinates of β'_0, and 0 to $\beta'_1, \ldots, \beta'_{l-2}$ (for example) to get an l-degree polynomial for β'_{l-1} with implicitly represented coefficients. The order of H is such that this polynomial has one expected solution. (If no solution is found one can vary the coefficients $\beta'_1, \ldots, \beta'_{l-2}$.)

The roots of a polynomial $f(\gamma)$ over a finite field $\mathbf{F}_{p^k}$ can be computed by Berlekamp's algorithm (see [27], [21]). The key idea is to factor the polynomial $f(\gamma)$ into

$$\gcd(f(\gamma), (\gamma+\delta)^{\frac{p^k-1}{2}} - 1) \quad \text{and} \quad \gcd(f(\gamma), (\gamma+\delta)^{\frac{p^k-1}{2}} + 1)$$

for some $\delta \in \mathbf{F}_{p^k}$. This is repeated with different δ and leads to the linear factors of $f(\gamma)$.

The computation of polynomial gcd's, and thus the entire root-finding algorithm, require only algebraic operations in $\mathbf{F}_{p^k}$, and the latter can be reduced to algebraic operations (and equality tests) in $\mathbf{F}_p$ (with respect to a normal basis representation). The implicit representations of the roots of an implicitly represented polynomial are thus efficiently computable. The complexity of computing one root is $O(nl \log l \log p \cdot (k^2 + \log |G|))$ group operations and $O(n^2 k \log l)$ calls to the DH-oracle.

We conclude that any H whose order is given by (8), where n is polynomial in $\log p$, fulfills the requirements of Theorem 6 if its order is smooth.

5.2.4 The case of a system of equations

If $|H|$ is not of the form (8), but a sum of p-powers (with small coefficients), a system of multivariate polynomial equations with several unknowns must be solved. Let $F(x)$ be a polynomial of positive degree which divides $x^n - 1$. Then there exists a uniquely determined subgroup H of $\mathbf{F}^*_{p^n}$ with order $|H| = F(p)$.

We need some definitions from algebra. The splitting field (over any field $\mathbf{F}$) of $x^n - 1$ is called n-th *cyclotomic field* $\mathbf{F}^{(n)}$ (over $\mathbf{F}$). Let $K^{(n)}$ be the set of roots of $x^n - 1$ in $K^{(n)}$; the generators of (the group) $K^{(n)}$ are called *primitive* n-th roots of unity (of which there are exactly $\varphi(n)$). Let ζ be a primitive n-th root of unity. $\Phi_n(x) := \prod_{(s,n)=1} (x - \zeta^s)$ is called the n-th *cyclotomic polynomial* and has degree $\varphi(n)$.

Because cyclotomic polynomials are irreducible and $x^n - 1 = \prod_{d|n} \Phi_d(x)$, at least one cyclotomic polynomial Φ divides F, and smoothness of $F(p)$ implies smoothness of $\Phi(p)$. Therefore, we can assume without loss of generality that F is a cyclotomic polynomial and, again without loss of generality, that

$$F(x) = \Phi_n(x) = \sum_{j=0}^{\varphi(n)} c_j x^j .$$

Let H be the (unique) subgroup of $\mathbf{F}^*_{p^n}$ with order $|H| = \Phi_n(p)$. For $\beta = \sum_{i=0}^{n-1} y_i \alpha_i$ with $y_i \in \mathbf{F}_p$, we have

$$\begin{aligned} \beta \in H \quad &\Longleftrightarrow \quad \left(\sum_{i=0}^{n-1} y_i \alpha_i\right)^{\sum_{j=0}^{\varphi(n)} c_j p^j} = 1 \\ &\Longleftrightarrow \quad \prod_{c_j \geq 0} \left(\sum_{i=0}^{n-1} y_i \alpha_i\right)^{p^j c_j} - \prod_{c_j < 0} \left(\sum_{i=0}^{n-1} y_i \alpha_i\right)^{p^j (-c_j)} = 0 . \end{aligned}$$

This leads to a system of n polynomials in the y_i over $\mathbf{F}_p$ of degree at most

$$\max\left\{\sum_{c_j>0} c_j, \sum_{c_j<0} |c_j|\right\} \leq \varphi(n) \cdot \max\{|c_j| : j = 0, \ldots, \varphi(n)\} \ .$$

Because $|H| \approx p^{\varphi(n)}$ we have to solve the (implicitly represented) polynomial equations for $n - \varphi(n)$ unknowns, and the number of equations can be reduced to $n - \varphi(n)$ as well (the solutions then have to be checked for the other equations).

Gröbner bases are a tool for solving systems of polynomial equations. They lead to equivalent systems of equations which have triangular form, such that a method for solving univariate equations (as Berlekamp's algorithm) suffices to solve the whole system. For an introduction to Gröbner bases, see [13]. A detailed description of the basic facts about Gröbner basis computations and of our application is given in Appendix C. The idea is to compute the polynomials (with implicitly represented coefficients) of a Gröbner basis of the polynomial ideal generated by the polynomials of the equations. The algorithm for the Gröbner basis computation, due to Buchberger, requires only algebraic polynomial arithmetic and can therefore be executed on implicitly represented arguments. The second step is to solve the separated system of implicitly represented polynomials by Berlekamp's method for univariate polynomials. For the complexity of the computations, see Appendix C. Polynomial time complexity is achieved if $n = O(1)$.

We conclude that the subgroup H of $\mathbf{F}_{p^n}^*$ of order $\Phi_n(p)$, $n = O(1)$, is applicable in Theorem 6 if it has smooth order. For example, smoothness of

$$\begin{aligned}
\Phi_6(p) &= p^2 - p + 1 \ , \\
\Phi_8(p) &= p^4 + 1, \ \text{ or} \\
\Phi_9(p) &= p^6 + p^3 + 1
\end{aligned}$$

implies that an appropriate group H_p over $GF(p)$ is constructable. As mentioned, this is now proved for $F(p)$ for any non-trivial polynomial $F(x)$ dividing $x^n - 1$ if $n = O(1)$. Other examples are the following alternating sums when $l = O(1)$:

$$p^{2l} - p^{2l-1} + - \cdots - p + 1 \ .$$

5.3 Summary

Theorem 6 now implies the following.

Corollary 8 *Let G be a cyclic group with generator g, and let B be a smoothness bound, polynomial in $\log|G|$. Then there exists a list of expressions $A(p)$ in p with the following property: if for every prime factor p of $|G|$ greater than B, at least one of the expressions $A(p)$ is B-smooth, then breaking the Diffie-Hellman protocol in G with respect to g is polynomial-time equivalent to computing discrete logarithms in G to the base g. (In the case of a multiple large prime factor p of $|G|$, the equivalence holds with respect to breaking the DH protocol in one of a certain subset of subgroups of G, or if an algorithm for computing p-th roots in G is given.) The list contains the following expressions:*

$$p - 1, \ p + 1,$$

$$p + 1 \pm 2a, \ p + 1 \pm 2b,$$

if $p \equiv 1 \pmod 4$, *where* $p = a^2 + b^2$ *and* $a + bi \equiv 1 \pmod{2 + 2i}$,

$$p + 1 \pm 2a,\ p + 1 \mp a \pm 2b,\ p + 1 \pm (a + b),$$

if $p \equiv 1 \pmod 3$, *where* $p = a^2 - ab + b^2$ *and* $a + b\omega \equiv 2 \pmod 3$,

$$\frac{(p^k)^l - 1}{p^k - 1} = (p^k)^{l-1} + \cdots + p^k + 1,$$

where $k, l = O((\log p)^c)$ *and* $c = O(1)$, *and*

$$\Phi_n(p),$$

where $n = O(1)$, *and* Φ_n *is the* n*-th cyclotomic polynomial.* □

6 Construction of groups for which the Diffie-Hellman problem is provably equivalent to the discrete logarithm problem

It appears desirable to use a group G in the DH protocol for which the equivalence to computing discrete logarithms provably holds. However, such reasoning should be used with care because it is conceivable that knowledge of the auxiliary groups makes computing discrete logarithms easier. There are three possible scenarios for such an equivalence:

1. When given G it is easy (also for the opponent) to find suitable auxiliary groups.
2. The designer of the group G knows suitable auxiliary groups but they are difficult to find for an opponent.
3. The designer of the group G knows that suitable auxiliary groups exist, without knowing them.

In the first case the equivalence holds, whereas in the other two cases breaking the DH protocol is at least as difficult as computing discrete logarithms when the auxiliary groups are known. Note that the second case can always be transformed into the first by publishing the suitable auxiliary groups. Of course, because this information can only help an opponent in breaking the Diffie-Hellman protocol, there is no reason for the designer of the group to make it public.

Constructing a group G of the third type is trivial: choose a (secret) arbitrary large smooth number m and search for a prime p in the interval $[m - 2\sqrt{m} + 1, m + 2\sqrt{m} + 1]$. A group G whose order contains only such large prime factors satisfies the third property. Note that it is easy to construct, for a given n, a DH-group G whose order is a multiple of n. One possibility is to find a multiple l of n such that $l + 1$ is prime and to use $G = GF(l + 1)^*$. An alternative, which may be more secure, is to use the construction of Lay and Zimmer [19] for finding an elliptic curve of order n.

The second case is somewhat more involved. Such a group G can be obtained by choosing a large smooth number m and using the method of Lay and Zimmer [19] for constructing a prime p together with an elliptic curve of order m.

We now consider efficient constructions for the first case. We generalize a method, presented in [37] by Vanstone and Zuccherato, for constructing a large prime p such that either a quarter of the curves $y^2 = x^3 - Dx$ or every sixth curve of the form $y^2 = x^3 + D$ have smooth order. First, we construct primes $p = a^2 + (k \pm 1)^2$ (for a fixed k with l digits) such that $a^2 + k^2$, which is then one of the possible orders of the curves $y^2 = x^3 - Dx$ over $\mathbf{F}_p$ (see (5)), is smooth.

l'-digit numbers x_1, x_2, y_1, and y_2 are chosen at random. Define

$$u + vi := (x_1 + y_1 i)(x_2 + y_2 i),$$

that is

$$u = x_1 x_2 - y_1 y_2, \quad v = x_1 y_2 + x_2 y_1.$$

Typically, u and v have approximately $2l'$ digits. If $\gcd(u, v)$ divides k (otherwise choose again), one can compute numbers c and d (of at most $2l' + l$ digits) such that

$$cv + du = k.$$

Define

$$a := cu - dv,$$

and restart the process if a is even. Then

$$a + ki = (c + di)(u + vi) = (c + di)(x_1 + y_1 i)(x_2 + y_2 i)$$

and

$$a^2 + k^2 = (c^2 + d^2)(x_1^2 + y_1^2)(x_2^2 + y_2^2).$$

The process is repeated until $a^2 + k^2$ is s-digit-smooth, which happens with probability approximately

$$\left(\frac{4l' + 2l}{s}\right)^{-\frac{4l'+2l}{s}} \cdot \left(\frac{2l'}{s}\right)^{-\frac{2l'}{s}} \cdot \left(\frac{2l'}{s}\right)^{-\frac{2l'}{s}}$$

(according to (2)), and smoothness can be tested with the elliptic curve factoring algorithm [20].

Because a and k are odd, exactly one of the expressions $a + (k \pm 1)i$ is congruent to 1 modulo $2 + 2i$. Let $\alpha := a + (k \pm 1)i$, respectively. Repeat the computations until

$$p := \alpha\overline{\alpha} = a^2 + (k \pm 1)^2$$

is prime. According to (5), a quarter of the curves $y^2 = x^3 - Dx$ over $\mathbf{F}_p$ have smooth order $a^2 + k^2$. Hence p is an $(8l' + 2l)$-digit prime such that an elliptic curve with s-digit-smooth order is constructable over $\mathbf{F}_p$. The expected number of trials is

$$O\left(\left(\frac{4l' + 2l}{s}\right)^{\frac{4l'+2l}{s}} \cdot \left(\frac{2l'}{s}\right)^{\frac{4l'}{s}} \cdot (8l' + 2l)\right) . \tag{9}$$

In a similar way, primes can be constructed such that curves of type $y^2 = x^3 + D$ have smooth order. More precisely, we generate primes

$$p = a^2 - a(k \pm 1) + (k \pm 1)^2$$

(where k is a fixed number of l digits and $a + (k \pm 1)\omega \equiv 2 \pmod 3$) such that

$$a^2 - ak + k^2,$$

which is one of the orders of the curves $y^2 = x^3 + D$ over $\mathbf{F}_p$ (see (6)), is s-digit-smooth. Again, we choose x_1, x_2, y_1, and y_2 (of l' digits) and compute u, v, c', d', c, d and a with

$$\begin{aligned} u + v\omega &= (x_1 + y_1\omega)(x_2 + y_2\omega) \\ c'v + d'u &= k \\ d &:= d', \\ c &:= c' + d' \\ a &:= cu + dv. \end{aligned}$$

Then

$$a + k\omega = (c + d\omega)(x_1 + y_1\omega)(x_2 + y_2\omega)$$

and

$$a^2 - ak + k^2 = (c^2 - cd + d^2)(x_1^2 - x_1y_1 + y_1^2)(x_2^2 - x_2y_2 + y_2^2).$$

(Note that $N(x + y\omega) = x^2 - xy + y^2$ is the norm of $\mathbf{Z}[\omega]$.) This is repeated until $a^2 - ak + k^2$ is s-digit-smooth and

$$p = a^2 - a(k \pm 1) + (k \pm 1)^2 \qquad (a + (k \pm 1)\omega \equiv 2 \pmod 3)$$

is prime. The expected number of repetitions is again given by (9).

In case of a small k, an L-digit prime p such that an s-digit-smooth curve is constructable over $\mathbf{F}_p$ can be found by

$$O\left(\left(\frac{L}{\sqrt{8}\cdot s}\right)^{\frac{L}{s}} \cdot L\right)$$

trials instead of

$$O\left(\left(\frac{L}{s}\right)^{\frac{L}{s}} \cdot L\right)$$

trials when varying p among L-digit numbers until p is prime and one of the considered curves is s-digit-smooth. For example, a 100-digit prime p such that a 10-digit-smooth curve over $\mathbf{F}_p$ is constructable can be found by approximately $3 \cdot 10^6$ trials (instead of about 10^{11} trials when using the straightforward strategy).

7 Concluding remarks

Our results imply that the DH problem is at least as difficult as the DL problem *with knowledge* of suitable auxiliary groups. Although it appears unlikely, it is possible that this knowledge helps computing discrete logarithms.

Throughout this paper, we have assumed to know the group order and its factorization. If $|G|$ is unknown, it is either computable from the side string S of Corollary 7, or, in case of efficiently constructable auxiliary groups (when given $|G|$ and its factorization), we can prove that breaking the DH protocol is at least as difficult as computing discrete logarithms in G when *given* $|G|$ and its factorization. It is conceivable that knowledge of $|G|$ could be of some help in computing discrete logarithms. For example, the algorithm of Pollard [31] requires knowledge of the group order. For the case of unknown factorization of the group order, note that no known discrete logarithm algorithm for general (not smooth) groups requires knowledge of the factors of the group order.

Let p be a large prime factor of $|G|$. If an appropriate multiplicative subgroup of an extension field of $\mathbf{F}_p$ has smooth order, then p can be found efficiently as a factor of $|G|$ (see [2]). The parameters of a smooth elliptic curve over $\mathbf{F}_p$ do generally not allow to find p efficiently by the method of [20], because no point can be generated on the curve modulo $|G|$.

The equivalence of the DH and DL problems holds in a strict sense for all cyclic groups with known factored group order (free of multiple large prime factors) if for any prime number p, an appropriate auxiliary group over $\mathbf{F}_p$ can be constructed, for example if the list of Corollary 8 is extended in such a way that it always contains a smooth number which can be found efficiently.

In Appendix D we describe a method, presented initially in [38] and independently considered in [4], for obtaining stronger results under the assumption of efficient DH-oracle *algorithms* using algebraic operations for certain groups. The idea is to execute the oracle algorithms on implicitly represented arguments. This allows to iterate the technique by using computation with multiply implicitly represented elements. It is then no more necessary that for every large prime factor p of $|G|$ a smooth auxiliary group H_p is known. For example, a cyclic auxiliary group H_p whose order contains a large prime factor q and a smooth auxiliary group H_q over $\mathbf{F}_q$ are sufficient under the assumption of a polynomial-time DH-oracle algorithm for H_p, using algebraic operations in $\mathbf{F}_p$.

Acknowledgment

We would like to thank Dan Boneh for interesting discussions.

Appendix A: Proof of Theorem 1

According to Lemma 1, one can construct a translation-invariant ε-DH-oracle which uses $O(\log|G|)$ group operations and one call to a time-invariant ε-DH-oracle per call. We proceed as follows: in a first step, independent of the oracle's input, we determine the error distribution of the translation-invariant ε-DH-oracle. More precisely, we determine all the errors which are approximately equally likely as the correct answer. In the second step, the distribution of the output with the oracle's input is compared with the reference distribution determined in the first step. In most cases, the correct answer is now uniquely determined. If not, which can happen in case of a symmetric error distribution, the correct answer can be found when the group order $|G|$ is known.

Let $\alpha := \beta\varepsilon/6$. An event with probability at least $1-\alpha$ will be called *almost certain*. Let $\delta := \varepsilon/10$ and $\delta' := \delta\varepsilon/100 = \varepsilon^2/1000$. If ε is not known, we take an estimate. (The proof does not depend on the choice of the constants (e.g., 1/10), which is somewhat arbitrary.) In order to determine the error distribution, one calls the translation-invariant oracle repeatedly with the input (g^0, g^0). Let ε' be the fraction of correct answers, and let t be the number of trials that are needed to assure that the true ε lies almost certainly in the interval $[\varepsilon'-\delta', \varepsilon'+\delta']$. It follows from the asymptotic behavior of the density of the standardized normal distribution that t is of order $O(\log(1/\alpha)/\delta'^2)$.

We now detect almost certainly all the errors e (including $e=0$) whose probability is in the interval $[\varepsilon'-\delta, \varepsilon'+\delta]$. (It could be helpful, but is not necessary, to detect the errors which have greater probability than the correct answer.) Call the oracle t times again, put the (implicitly represented) errors g^e into a sorted list and delete all the entries with a frequency not in the interval

$$[\varepsilon'-(\delta+\delta'), \varepsilon'+(\delta+\delta')],$$

such that those with probability in the interval $[\varepsilon' - \delta, \varepsilon' + \delta]$ are detected with probability at least $(1-\alpha)^{2/\varepsilon}$. The actual probabilities of all the remaining errors lie in

$$[\varepsilon' - (\delta + 2\delta'), \varepsilon' + (\delta + 2\delta')] \tag{10}$$

with probability at least $(1-\alpha)^{2/\varepsilon}$.

We now describe the second phase. Let g^u and g^v be given. The translation-invariant oracle is called t times with the input (g^u, g^v) and the answers are listed. Those with frequency not in $[\varepsilon' - (\delta + 3\delta'), \varepsilon' + (\delta + 3\delta')]$ are deleted. With probability at least $(1-\alpha)^{4/\varepsilon}$, all the errors that have a probability which lies in the interval (10) belong to the list, and all the errors in the list have their probability in $[\varepsilon' - (\delta + 4\delta'), \varepsilon' + (\delta + 4\delta')]$. The probability that the first list contains *all* the (implicitly represented) errors which have their probability in the interval $[\varepsilon' - \delta, \varepsilon' + \delta]$ and that all the errors of the first list also occur in the second list is at least

$$(1-\alpha)^{6/\varepsilon} \geq 1 - \frac{6\alpha}{\varepsilon} = 1 - \beta. \tag{11}$$

If this is fulfilled but the second list contains more elements than the first one, then almost certainly there is an error with probability in the set

$$[\varepsilon' - (\delta + 4\delta'), \varepsilon' - \delta] \cup [\varepsilon' + \delta, \varepsilon' + (\delta + 4\delta')]. \tag{12}$$

In this case replace δ by $\delta + 5\delta'$. Because the critical sets (12) are disjoint for different δ of distance $5\delta'$, and because there are at most $2/\varepsilon$ errors with probability at least $\varepsilon/2$, the second list can contain more elements than the first only for one out of 10 choices for δ.

If the lists have the same length c, they contain the same (implicitly represented) errors with probability at least $1-\beta$, but the entries of the second list are shifted by g^{uv}. In order to determine this shift, we multiply the first list with each of the entries of the second, sort it and compare it with the second list. In the (typical) case of a *unique* element of the second list for which equality holds, this element is g^{uv} with probability at least $1-\beta$. If there exists more than one such element, this means that the lists have a non-trivial translation symmetry, or, more precisely, that they are invariant under a multiplication with $|G|/c'$ for a divisor c' of c and $|G|$. Let c' be the maximal number with this property. Note that $|G|$ has a factor $c' \leq c \leq 1/(\varepsilon' - (\delta + 2\delta'))$ in this case. There are c' candidates for g^{uv}, namely

$$g^{uv},\ g^{uv + \frac{|G|}{c'}}, \ldots,\ g^{uv + (c'-1)\frac{|G|}{c'}}.$$

We show that if $|G|$ is known, the correct one of them can be determined. If $|G|$ is *not* known, but all the factors of $|G|$ are greater than $(1+s)/\varepsilon$, then one can execute the same procedure again, choosing a smaller δ, such that $1/(\varepsilon' - (\delta + 2\delta')) < (1+s)/\varepsilon$, which is satisfied if $\delta < s\varepsilon/2$.

Let $|G|$ be known, and let $p_1, \ldots, p_l$ be the distinct prime factors of c', i.e., $c' = \prod_{i=1}^{l} p_i^{f_i}$, and let $d := \prod_{i=1}^{l} p_i^{e_i}$ be the product of the maximal powers of the p_i dividing $|G|$. d is $2/\varepsilon$-smooth because $c' \leq c \leq 2/\varepsilon$, and u and v, and hence also uv, are computable modulo d from g^u and g^v by the Pohlig-Hellman discrete logarithm algorithm [30] with $O((\log |G|)^2 + \log |G|/\varepsilon)$ group operations. We can also compute the discrete logarithms of all the candidates modulo d.

The discrete logarithm of exactly one of the candidates has the correct remainder with respect to d. For every i, exactly every $p_i^{f_i}$-th candidate has the correct remainder with respect to $p_i^{e_i}$. The primes are distinct, thus relatively prime, and every $\prod_{i=1}^{l} p_i^{f_i}$-th candidate, that is *exactly one* of them, has the correct remainder. We have found g^{uv} with probability at least $1-\beta$. □

Appendix B: Finding generator sets of the auxiliary groups

We show how generator sets can be generated in a finite abelian auxiliary group H with B-smooth order $|H| = \prod_{i=1}^{l} q_i^{f_i}$. If H is cyclic, we determine a generator h by trial and error. The element h is a generator of H if and only if $(|H|/q_i)h \neq e$ for $i = 1, \ldots, l$, which can be tested by $O((\log |H|)^2)$ operations in H. The expected number of trials is $|H|/\varphi(|H|) = O(\log \log |H|)$ (see [15]), and the expected complexity is therefore $O((\log |H|)^2 \log \log |H|)$ group operations in H to find a generator. Additionally, an expected number of $O(\log \log |H|)$ random points of H have to be generated.

For the general case, suppose $H \cong \mathbf{Z}_{n_1} \times \cdots \times \mathbf{Z}_{n_r}$ (such that n_{j+1} divides n_j for $i = 1, \ldots, r-1$), where the numbers r and $n_1, \ldots, n_r$ are not known a priori. The maximal order of an element in H is $n_1 = \prod_{i=1}^{l} q_i^{g_i}$. We find such an element by trial and error. The expected number of trials is $n_1/\varphi(n_1) = O(\log \log n_1) = O(\log \log |H|)$.

Let π_l denote the canonical projections to the quotient groups. More exactly, $\pi_l(u)$ is the element of $H/\langle h_1, \ldots, h_l \rangle$ containing u. Suppose we have constructed the points h_i with order n_i in $H/\langle h_1, \ldots, h_{i-1} \rangle$ for $i = 1, \ldots, j-1$. For the construction of h_j such that $\pi_{j-1}(h_j)$ has maximal order in $H/\langle h_1, \ldots, h_{j-1} \rangle$, we choose the same number of points h in H as above at random and compute $\mathrm{ord}_{H/\langle h_1, \ldots, h_{j-1} \rangle} \pi_{j-1}(h)$ by comparing

$$\frac{n_1}{q_i^k} \pi_{j-1}(h) = \pi_{j-1}\left(\frac{n_1}{q_i^k} h\right) \quad (\text{for } i = 1, \ldots, l \text{ and } k = g_i, g_i - 1, \ldots, 0)$$

with the unity $e_{H/\langle h_1, \ldots, h_{j-1} \rangle}$ of the quotient group. Comparing $\pi_{j-1}(h')$ and $e_{H/\langle h_1, \ldots, h_{j-1} \rangle}$ is equivalent to deciding if $h' \in \langle h_1, \ldots, h_{j-1} \rangle$, which is done by the generalized PH discrete logarithm method described in Section 4. It is efficient because $|H|$ is smooth. If $\mathrm{ord}_{H/\langle h_1, \ldots, h_{j-1} \rangle} \pi_{j-1}(h_j)$ does not divide $\mathrm{ord}_{H/\langle h_1, \ldots, h_{j-2} \rangle} \pi_{j-2}(h_{j-1})$, the process must be restarted. One of the preceding points has not had maximal order. The same holds if $j > \max\{f_i\}$. The latter is a bound for the rank r of H. The process comes to an end if $\langle h_1, \ldots, h_r \rangle = H$, that is

$$H/\langle h_1, \ldots, h_r \rangle = \{e\},$$

and $\{h_1, \ldots, h_r\}$ is a generator set of H. An element c of H has a unique representation $c = \sum_{j=1}^{r} k_j h_j$ with $k_j \in \{0, \ldots, n_j - 1\}$. The expected number of operations in H to determine the generator set is

$$O\left(r^2 \log \log |H| \cdot \frac{\sqrt{B^r}}{\log B} (\log |H|)^3\right)$$

(using the time-memory tradeoff in the PH method.) The expected number of random points that have to be generated in H is $O(r \log \log |H|)$.

Appendix C: Using Gröbner bases for proving the applicability of certain subgroups of finite fields

We describe Gröbner bases and use them to show the applicability of the subgroup with order $\Phi_n(p)$ of $\mathbf{F}_{p^n}^*$ as an auxiliary group (if $n = O(1)$).

Let R be the ring $\mathbf{F}[x_1, \ldots, x_n]$ of multivariate polynomials, where $\mathbf{F}$ is a field. The elements of R are referred to as polynomials. Let

$$p_i = 0 \quad (i = 1, \ldots, l)$$

be a system of polynomial equations (we write $P = 0$, where $P := \{p_i\}$). Any different basis of the generated ideal $\langle P \rangle$ in the ring R is an equivalent system of equations. Gröbner bases with respect to the lexicographic term ordering are such that the system can be solved. The lexicographic term ordering is defined as follows:

$$\prod x_j^{i_j} < \prod x_j^{i'_j} \quad :\Longleftrightarrow \quad i_j = i'_j \text{ for } j = 1, \ldots, l-1 \text{ and } i_l < i'_l \text{ for some } l.$$

We motivate the definition of Gröbner bases. Let f and g be polynomials, and let t be the leading term of g. One can reduce f modulo g if any monomial of f is a multiple of t, $f = \alpha t + r$. The reduction of f modulo g is then

$$f - \frac{\alpha t}{\mathrm{M}(g)} \cdot g \ , \tag{13}$$

where $\mathrm{M}(\cdot)$ denotes the leading monomial. Let Q be a set of polynomials. The *reducer set* of the polynomial f with respect to Q are the polynomials g in Q such that the leading monomial of f can be reduced modulo g. There exists a simple algorithm for a maximal reduction of a polynomial f modulo a set Q of polynomials based on (13). Since R is not a principal ideal domain (if $n > 1$), the maximal reduction is not unique, and an element q of $\langle Q \rangle$ may be irreducible modulo Q. Gröbner bases (with respect to a term ordering) are defined and characterized by the following equivalent conditions:

1. Uniqueness of maximal reductions modulo G.
2. If $f \in \langle G \rangle$, then f reduces to 0 modulo G.
3. For all f and g in G,

$$\mathrm{lcm}\left(\mathrm{M}(f), \mathrm{M}(g)\right) \cdot \left(\frac{f}{\mathrm{M}(f)} - \frac{g}{\mathrm{M}(g)} \right) =: \text{s-poly}(f, g)$$

   reduces to 0 modulo G.

For given P, the third criterion leads to a simple algorithm, due to Buchberger, for the computation of a Gröbner basis G of $\langle P \rangle$ by extending P.

Algorithm (Buchberger) *Choose any pair (f_1, f_2) in $P \times P$ and compute a maximal P-reduction of* s-poly(f_1, f_2). *If it is different from zero, extend P by adding this polynomial. Repeat the process for all pairs, including the pairs with components added to P during the process.*

This algorithm can be improved by criteria deciding a priori whether s-poly(f, g) will reduce to 0, such that the number of s-polynomial reductions is decreased. The complexity of Gröbner basis computations is still a field of research. If the system $P = 0$ has only finitely many solutions over $\mathbf{C}$, the computation of a lexicographic Gröbner basis for $\langle P \rangle$ has complexity $O(d^{n^2})$, where n is a bound for the number of variables and polynomials and d is the maximal degree. The degrees of the polynomials in the Gröbner basis are bounded by $O(d')$, where

$$d' := (nd)^{(n+1)^{2^{s+1}}},$$

and s is the dimension of the ideal, $s \leq n$.

The following are key properties of Gröbner bases for solving equations. Let P be a set of polynomials and G a monic Gröbner basis for $\langle P \rangle$.

Property 1 *$P = 0$ has a solution if and only if $1 \notin G$.*

Property 2 *Let H be the set of all leading terms occuring in G. Then the following are equivalent:*

- *P has finitely many solutions over $\mathbf{C}$,*
- *For all i, there exists m_i such that $(x_i)^{m_i} \in H$.*

The first property is a criterion for solvability, and the second one implies, when using the lexicographic ordering, that a subset of the equations of the Gröbner basis has triangular form and can be solved if the same holds for univariate equations. The fact that $(x_i)^{m_i}$ is the leading term of a polynomial implies that the variables $x_1, \ldots, x_{i-1}$ do *not* occur in the polynomial. For example the polynomial with $(x_n)^{m_n}$ as leading term is *univariate* (with the only variable x_n), and similarly it follows that there is a polynomial containing x_{n-1} and x_n only, etc.

In order to solve our system of $n - \varphi(n)$ equations and unknowns, the Gröbner basis has to be computed with *implicitly represented* polynomial coefficients, because we have to assign the implicitly represented x to one of the variables before this computation, such that the system has a finite number of solutions over $\mathbf{C}$ only.

Buchberger's algorithm (which is built of polynomial reductions and comparisons) as well as the reduction of the resulting set of polynomials (see [13]) contain elementary polynomial arithmetic only and work with implicitly represented arguments.

The following are worst case bounds for the complexity of the Gröbner basis computation (with implicitly represented polynomial coefficients) for the system of equations characterizing the subgroup H of $\mathbf{F}_{p^n}^*$ with order $\Phi_n(p)$, where $\Phi_n(x) = \sum_{i=1}^{\varphi(n)} c_i x^i$ is the n-th cyclotomic polynomial:

$$O\left((\varphi(n) \cdot \max\{|c_i| : i = 0, \ldots, \varphi(n)\})^{(n-\varphi(n))^2} \cdot \log |G|\right) \text{ operations in } G \text{ and}$$

$$O\left((\varphi(n) \cdot \max\{|c_i| : i = 0, \ldots, \varphi(n)\})^{(n-\varphi(n))^2} \cdot \log p\right) \text{ calls to the oracle.}$$

If no solution exists (the first property above provides an efficient test), one replaces x by $x + d$ or varies the other coordinates.

We solve the implicitly represented polynomial equations for $n - \varphi(n)$ unknowns. According to the result of Gianni and Kalkbrener (see [13]), it suffices to solve a subset of $n - \varphi(n)$ equations (of triangular form and degree $O(d')$), which is done by Berlekamp's algorithm. The first polynomial has to be solved once, the second one $O(d')$ times (in the worst case, the first polynomial has $O(d')$ different solutions), the third one $O((d')^2)$ times, etc. This yields $O\left((d')^{n-\varphi(n)}\right)$ executions of Berlekamp's algorithm. The *effective* number of executions will be much smaller in a typical case, since only about one solution is expected. Hence the complexity of embedding (the implicitly represented) x into a group element c is then $O(\log p \log |G|)$ group operations in G and $O(\log p)$ calls to the oracle if $n = O(1)$. (Using the same method explicitly, points in H can efficiently be generated.)

Appendix D: DH-oracle algorithms using algebraic operations

Stronger equivalence results can be obtained under the assumption that not only a DH-oracle for the group $G = \langle g \rangle$, but also DH-oracle algorithms, using algebraic operations, for certain cyclic auxiliary groups are given.

Assume first that DH-oracles for the groups $\mathbf{F}_p^*$ are given by a polynomial-time algorithm using algebraic operations in $\mathbf{F}_p$. The idea is to perform the oracle algorithm with implicitly represented arguments. Let again B be a smoothness-bound, polynomial in $\log |G|$, p_0 a prime factor of $|G|$ greater than B, and let p_1 be the only prime factor of $p_0 - 1$ greater than B. Assume further that p_i is the only prime factor of $p_{i-1} - 1$ greater than B for all $i = 2, \ldots, k$, and that $p_k - 1 =: T = \prod_i r_i^{n_i}$ is B-smooth, where $k = O(1)$. Given $a = g^s$, it is now possible to compute $x_0 \equiv s \pmod{p_0}$, $x_0 \in \mathbf{F}_{p_0}$, in polynomial time when given a DH-oracle for G. Let

$$h_0 := \frac{|G|}{p_0}, \quad h_i := \frac{p_{i-1} - 1}{p_i} \text{ (for } i = 1, \ldots, k), \text{ and } \mathbf{F}_{p_i}^* = \langle c_i \rangle \text{ (for } i = 0, \ldots, k-1).$$

If $x_0 \neq 0$, then $x_0 = c_0^{w_0}$ (in $\mathbf{F}_{p_0}$), and g^s as an implicit representation of x_0. Since $p_0 - 1$ has a large prime factor p_1, w_0 modulo p_1 cannot be obtained directly. $x_1 \equiv w_0 \pmod{p_1}$ (with $x_1 \in \mathbf{F}_{p_1}$) can be written as $x_1 = c_1^{w_1}$ (in $\mathbf{F}_{p_1}$, if $x_1 \neq 0$), and g^s is a 'double-implicit' representation of x_1. Our assumptions allow efficient computation with these elements of $\mathbf{F}_{p_1}$ which are 'double-implicitly' represented. For example, an addition of two $\mathbf{F}_{p_1}$-elements requires multiplication of the corresponding implicitly represented $\mathbf{F}_{p_0}^*$-elements and can be obtained by a call to the DH-oracle for G. A multiplication in $\mathbf{F}_{p_1}^*$ is done by an oracle call for $\mathbf{F}_{p_0}^*$ with implicitly represented arguments and an implicitly represented answer. This works (in polynomial time) because of the stated properties of the DH-oracle algorithm for $\mathbf{F}_{p_0}^*$.

Finally computation with $(k+1)$-times implicitly represented arguments is possible in the smooth group $\mathbf{F}_{p_k}^*$. The index search problem for the set

$$\left\{ g^{h_0 c_0^{h_1 c_1^{\cdot^{\cdot^{\cdot^{h_k c_k^{\frac{T}{r_i} t}}}}}}} \; : \; t = 0, \ldots, r_i - 1 \right\}$$

and the element

$$g^{h_0 c_0^{h_1 c_1^{\cdot^{\cdot^{\cdot^{h_k c_k^{\frac{T}{r_i} w_k}}}}}}}$$

which can be obtained in polynomial time by computation with multiply implicitly represented arguments, is solved and leads to w_k modulo r_i. When this is done for all prime powers $r_i^{n_i}$, w_k is computable modulo T. Then $x_k = c_k^{w_k}$ (in $\mathbf{F}_{p^k}$), and one can get w_{k-1} modulo $p_{k-1} - 1$ in polynomial time because the other prime factors of $p_{k-1} - 1$ are smaller than B. Finally, we obtain w_0 modulo $p_0 - 1$ and x_0.

The process works in an analogous way if some of the used groups are cyclic elliptic curves or Jacobians, provided a DH-oracle algorithm with the same properties exists for these groups.

Corollary 9 *Let G be a cyclic group with the property that for all prime factors p_0 of $|G|$ greater than a smoothness bound B, polynomial in $\log |G|$, there exist k (bounded by some constant independent of p_0 and $|G|$) , numbers l_i (for $i = 1, \ldots, k+1$) and primes p_i ($i = 1, \ldots, k$) such that p_i is the only prime factor of l_i greater than B for $i = 1, \ldots, k$ and l_{k+1} is B-smooth, where l_i (for $i = 1, \ldots, k+1$) is either equal to $p_{i-1} - 1$ or to the order of a constructable cyclic elliptic curve or Jacobian over $\mathbf{F}_{p_{i-1}}$ such that a polynomial-time DH-oracle algorithm using group operations and algebraic operations in $\mathbf{F}_{p_{i-1}}$ is known for the constructable group over $\mathbf{F}_{p_{i-1}}$. Assume further that all the prime factors greater than B of the group orders are single.*

Then, breaking the Diffie-Hellman protocol and computing discrete logarithms are polynomial-time equivalent in G. □

References

[1] L.M. Adleman and M.A. Huang, Primality testing and abelian varieties over finite fields, *Lecture Notes in Mathematics*, vol. 1512, Springer-Verlag, 1992.

[2] E. Bach and J. Shallit, Factoring with cyclotomic polynomials, *Math. Comp.*, vol. 52, pp. 201-219, 1989.

[3] T. Beth, W. Geiselmann and F. Meyer, Finding (good) normal bases in finite fields, *Proc. ISSAC '91*, pp. 173-178, ACM press, 1991.

[4] D. Boneh and R. J. Lipton, Algorithms for black-box fields and their application to cryptography, preprint, 1995.

[5] J. Buchmann and V. Müller, Computing the number of points of elliptic curves over finite fields, *Proc. ISSAC '91*, pp. 179-182, ACM press, 1991.

[6] J. Buchmann and H.C. Williams, A key-exchange system based on imaginary quadratic fields, *Journal of Cryptology*, vol. 1, no. 2, pp. 107-118, 1988.

[7] E.R. Canfield, P. Erdös and C. Pomerance, On a problem of Oppenheim concerning "Factorisatio Numerorum", *J. Number Theory*, vol. 17, pp. 1-28, 1983.

[8] D.G. Cantor, Computing in the Jacobian of a hyperelliptic curve, *Math. Comp.*, vol. 48, No. 177, pp. 95-101, 1987.

[9] 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, Berlin: Springer-Verlag, 1989.

[10] W. Diffie and M.E. Hellman, New directions in cryptography, *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644-654, 1976.

[11] T. El-Gamal, A public key cryptosystem and a signature scheme based on the discrete logarithm, *IEEE Transactions on Information Theory*, vol. 31, no. 4, pp. 469-472, 1985.

[12] W. Feller, An introduction to probability theory and its applications, John Wiley & Sons, 1968.

[13] K.O. Geddes, S.R. Czapor and G. Labhan, Algorithms for computer algebra, Kluwer Academic Publisher, 1992.

[14] S. Goldwasser and J. Kilian, Almost all primes can be quickly certified, *Proc. of the 18th Annual ACM Symposium on the Theory of Computing*, pp. 316-329, 1986.

[15] G.H. Hardy and E.M. Wright, An introduction to the theory of numbers, University Press, Oxford, 1979.

[16] K. Ireland and M. Rosen, A classical introduction to modern number theory, Springer-Verlag, 1982.

[17] N. Koblitz, Hyperelliptic cryptosystems, *Journal of Cryptography*, vol. 1 (1989), pp. 139-150, 1989.

[18] S. Lang, Algebra, Addison-Wesley Publ. Comp., 1984.

[19] G.-J. Lay and H.G. Zimmer, Constructing elliptic curves with given group order over large finite fields, preprint, 1994.

[20] H.W. Lenstra, Jr., Factoring integers with elliptic curves, *Annals of Mathematics*, vol. 126, pp. 649-673, 1987.

[21] R. Lidl and H. Niederreiter, Introduction to finite fields and their application, Cambridge University Press, 1986.

[22] J.L. Massey, Advanced Technology Seminars Short Course Notes, pp. 6.66-6.68, Zürich, 1993.

[23] U.M. Maurer and Y. Yacobi, Non-interactive public-key cryptography, *Advances in Cryptology - EUROCRYPT '91*, Lecture Notes in Computer Science, Berlin: Springer-Verlag, vol. 547, pp. 498-507, 1991.

[24] U.M. Maurer, Towards the equivalence of breaking the Diffie-Hellman protocol and computing discrete logarithms, *Advances in Cryptology - CRYPTO '94*, Y. Desmedt (ed.), Lecture Notes in Computer Science, Berlin: Springer-Verlag, vol. 839, pp. 271-281, 1994.

[25] K.S. McCurley, A key distribution system equivalent to factoring, *Journal of Cryptology*, vol. 1, no. 2, pp. 95-105.

[26] 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.

[27] A.J. Menezes (ed.), Applications of finite fields, Kluwer Academic Publishers, 1992.

[28] A. Menezes, Elliptic curve public key cryptosystems, Kluwer Academic Publishers, 1993.

[29] R. Peralta, A simple and fast probabilistic algorithm for computing square roots modulo a prime number, *IEEE Trans. on Information Theory*, vol. 32, no. 6, pp. 846-847, 1986.

[30] 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.

[31] J.M. Pollard, Theorems on factorization and primality testing, *Proceedings of the Cambridge Philosophical Society*, vol. 76, pp. 521-528, 1974.

[32] R.L. Rivest, A. Shamir, and L. Adleman, A method for obtaining digital signatures and public-key cryptosystems, *Communications of the ACM*, vol. 21, no. 2, pp. 120-126, 1978.

[33] H. Rück, A note on elliptic curves over finite fields, *Math. Comp.*, vol. 49, pp. 301-304, 1987.

[34] K. Sakrai and H. Shizuya, Relationships among the computational powers of breaking discrete log cryptosystems, *Advances in Cryptology – EUROCRYPT '95*, L.C. Guillou and J.-J. Quiswater (Eds.), Springer-Verlag Berlin, 1995.

[35] C.P. Schnorr, Efficient identification and signatures for smart cards, *Advances in Cryptology – CRYPTO '89*, Lecture Notes in Computer Science, vol. 435, pp. 239-252, Berlin: Springer-Verlag, 1990.

[36] R. Schoof, Elliptic curves over finite fields and the computation of square roots mod p, *Math. Comp.*, vol. 44, No. 170, pp. 483-494, 1985.

[37] S.A. Vanstone and R.J. Zuccherato, Elliptic curve cryptosystems using curves of smooth order over the ring $\mathbf{Z}_n$, Preliminary version, 1994.

[38] S. Wolf, Diffie-Hellman and discrete logarithms, Diploma Thesis, Department of Computer Science, ETH Zürich, March 1995.