

# The Relationship Between Breaking the Diffie-Hellman Protocol and Computing Discrete Logarithms <sup>\*†</sup>

Ueli M. Maurer      Stefan Wolf

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

March 30, 1998

## Abstract

Both uniform and non-uniform results concerning the security of the Diffie-Hellman key-exchange protocol are proved. First, it is shown that in a cyclic group  $G$  of order  $|G| = \prod p_i^{e_i}$ , where all the multiple prime factors of  $|G|$  are polynomial in  $\log |G|$ , there exists an algorithm that reduces the computation of discrete logarithms in  $G$  to breaking the Diffie-Hellman protocol in  $G$  and has complexity  $\sqrt{\max\{\nu(p_i)\}} \cdot (\log |G|)^{O(1)}$ , where  $\nu(p)$  stands for the minimum of the set of largest prime factors of all the numbers  $d$  in the interval  $[p - 2\sqrt{p} + 1, p + 2\sqrt{p} + 1]$ . Under the unproven but plausible assumption that  $\nu(p)$  is polynomial in  $\log p$ , this reduction implies that the Diffie-Hellman problem and the discrete logarithm problem are polynomial-time equivalent in  $G$ . Second, it is proved that the Diffie-Hellman problem and the discrete logarithm problem are equivalent in a uniform sense for groups whose orders belong to certain classes: there exists a polynomial-time reduction algorithm that works for all those groups. Moreover, it is shown that breaking the Diffie-Hellman protocol for a small but non-negligible fraction of the instances is equally difficult as breaking it for

---

\*Some results of this paper have been presented at CRYPTO '94 [26] and at CRYPTO '96 [30].

†This work was supported by the Swiss National Science Foundation (SNF), grant No. 20-42105.94.

all instances. Finally, efficient constructions of groups are described for which the algorithm reducing the discrete logarithm problem to the Diffie-Hellman problem is efficiently constructible.

**Key words:** public-key cryptography, Diffie-Hellman protocol, discrete logarithms, finite fields, elliptic curves.

**AMS subject classification:** 11T71

## 1 Introduction

Two challenging open problems in cryptography are to prove or disprove that breaking the Diffie-Hellman protocol [13] is computationally equivalent to computing discrete logarithms in the underlying group and that breaking the RSA system [40] is computationally equivalent to factoring the modulus. This paper is concerned with the first of these problems.

### 1.1 The discrete logarithm problem

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

### 1.2 The Diffie-Hellman key-exchange protocol and the Diffie-Hellman problem

The Diffie-Hellman (DH) protocol [13] allows two parties Alice and Bob, connected by an authenticated but otherwise insecure channel (for instance an insecure telephone line over which Alice and Bob authenticate each other by speaker recognition), to generate a mutual secret key which appears to be computationally infeasible to determine for an 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 DL problem is believed to be hard. 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. Note that there exist efficient

algorithms for exponentiation in groups. Finally, they exchange these group elements over the insecure public channel and 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. Since  $a_{AB} = a_{BA}$ , 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.

It is unknown whether a group exists for which the DL problem is hard, but several candidate groups have been proposed. Examples are the multiplicative groups of large finite fields (prime fields [13] or extension fields), the multiplicative group of residues modulo a composite number [31],[32], elliptic curves over finite fields [36],[21], the Jacobian of a hyperelliptic curve over a finite field [20], and the class group of imaginary quadratic fields [7].

The security of the DH protocol is based on the assumptions that the DL problem is hard to solve in  $G$ , and that this implies that it is hard to compute  $g^{s_A s_B}$  from  $g^{s_A}$  and  $g^{s_B}$ . We will refer to the problem of computing  $g^{s_A s_B}$  from  $g^{s_A}$  and  $g^{s_B}$  as the *Diffie-Hellman (DH) problem*. This paper is mainly concerned with the relationship between the DH and DL problems. It is clear that the DH problem cannot be more difficult than the DL problem because exponentiation in a group is efficient. Conversely, even when using a group for which the DL problem is hard, this does not immediately imply that the DH protocol is secure when using this group. However, we will show that for every group whose order is not divisible by the square of a large prime number, the DH problem cannot be substantially easier than the DL problem. Moreover, for certain classes of groups an efficient algorithm reducing the DL problem to the DH problem does not only exist but is efficiently constructible.

### 1.3 Outline of the paper

The paper is organized as follows. In Section 2 a general index-search problem is defined and investigated, and some algorithms for computing discrete logarithms are described. In Sections 3 and 4 a technique for proving the equivalence of the DH and DL problems, using so-called auxiliary groups, is presented, and examples of suitable auxiliary groups, for instance elliptic curves or subgroups of the multiplicative group of a finite field, are described. These two sections contain the main results of this paper. More precisely, a generalization of the result of [26] is proved in Section 3 which states that the DH and DL problems are equivalent for groups  $G$  for which appropriate auxiliary groups are given. The first result of Section 4 is a non-uniform reduction of the DL problem to the DH problem: It is shown (under an unproven but plausible number-theoretic conjecture) that there exists, for

every group whose order does not contain a multiple large prime factor, a polynomial-time algorithm computing discrete logarithms and making calls to an oracle solving the DH problem. The second result of Section 4 is a list of smoothness conditions (depending on  $|G|$ ) which make the DH and DL problems equivalent in a uniform sense, i.e., an efficient reduction algorithm does not only exist but can also be found efficiently. In Section 5, several variants of the DH problem are defined, and it is shown that they are (almost) as hard as the original DH problem. For instance, breaking the DH problem with small probability is equally hard as breaking it with arbitrarily high probability.

In Appendix A an algorithm for finding generating sets of abelian groups is described. Appendix B contains some basic facts about Gröbner basis computations which are required in Section 4. In Appendix C we obtain results which are stronger than those of Sections 3 and 4 under the assumption that efficient *algorithms* exist for solving the DH problem in certain groups, and in Appendix D, we show how to construct DH groups for which the DH and DL problems are provably equivalent.

## 1.4 Related work

Considerations on related topics can be found in [4],[3],[45],[28],[11],[10], and [42]. In [4], the notion of a black-box field is introduced which makes more explicit the concept of computation with implicitly represented elements presented in [26]. Furthermore, the existence of a uniform reduction of the DL problem to the DH problem of subexponential complexity was proved in [4], using methods related to those of [26] and of Section 3 and Appendix C of this paper.

In [45], the hardness of the DH problem (and hence of the DL problem) is proved in the generic model, i.e., for general-purpose algorithms that do not exploit any special property of the representation of the group elements. However, it was shown in [28] that the DH and DL problems are *not* computationally equivalent in a generic sense if the group order contains multiple large prime factors. In [11], the hardness of the DL and DH problems modulo  $p$  is proved in special computational models. For example it was shown that the DH function cannot be interpolated by a low-degree polynomial.

An alternative construction to that of Section 5 for correcting a faulty oracle solving the DH problem is described in [45]. Finally, a comparison of the security of different DL based systems is given in [42].

## 2 The index-search problem and algorithms for computing discrete logarithms

### 2.1 The index-search and DL problems

Let  $A = (a_i)_{i=1, \dots, n-1}$  be a list of elements of some set  $S$  such that for a given  $i$  it is easy to compute  $a_i$ . We call the problem of computing for a given  $b \in S$  an index  $i$  such that  $b = a_i$  the *index-search problem*. It can trivially be solved by exhaustive search which requires at most  $n$  comparisons. If the list 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 trade-off known as *baby-step giant-step* algorithm. Using a table of size  $M$  to store the sorted list of values  $b, \sigma(b), \dots, \sigma^{M-1}(b)$ , one can compute the elements  $a_0, a_M, a_{2M}, \dots$  until one of them, say  $a_{iM}$ , equals an element  $\sigma^j(b)$  contained in the table. Then the index of  $b$  is  $iM - j$ . For the choice  $M := \lceil \sqrt{n} \rceil$ , the running time of the algorithm is  $O(\sqrt{n} \log n)$ .

The DL problem in a cyclic group  $H$  of order  $|H|$  with generator  $h$  is the index-search problem for the list  $(1, h, \dots, h^{|H|-1})$ . Multiplication with  $h$  corresponds to the above-mentioned permutation  $\sigma$ . Hence the baby-step giant-step algorithm is applicable for solving the DL problem. It is a general-purpose algorithm that uses no particular properties of the representation of the group elements other than the uniqueness of the representation.

### 2.2 The Pohlig-Hellman algorithm

We describe a generic algorithm due to Pohlig and Hellman [37] which reduces the computation of discrete logarithms to the same problem in the minimal non-trivial subgroups. It plays a central role in the paper.

**Theorem 1 [37]** *Let  $H = \langle h \rangle$  be a cyclic group with order  $|H| = \prod_{i=1}^r q_i^{f_i}$ , and let  $a = h^x \in H$  be given. The discrete logarithm  $x$  of  $a$  can be computed by  $O(\sum f_i(\log |H| + q_i))$  group operations and equality tests of group elements. If memory space for storing  $\lceil \sqrt{q} \rceil$  group elements (where  $q$  is the greatest prime factor of  $|H|$ ) is available, the running time can be reduced to  $O(\sum f_i(\log |H| + \sqrt{q_i} \log q_i))$ .*

*Proof.* To solve  $a = h^x$  for  $x$ , 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  $q_i^{f_i}$ ,

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

The number  $x_{i0}$  is the discrete logarithm of  $a^{|H|/q_i} = h^{x \cdot |H|/q_i} = h^{x_{i0} \cdot |H|/q_i}$  in the group  $H^{(i)} := \langle h^{|H|/q_i} \rangle$  of order  $q_i$ . Assume now that  $x_{i0}, \dots, x_{i,k-1}$  have already been computed. The number  $x_{ik}$  is the discrete logarithm of

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

in the same group  $H^{(i)}$ . The computation of a discrete logarithm in  $H^{(i)}$  has complexity  $O(q_i)$  with exhaustive search and can be sped up by a factor  $M$  when a table of size  $M$  is used (that can be sorted in time  $O(M \log M)$ ).

Given  $x$  modulo  $q_i^{f_i}$  for all  $i$ , Chinese remaindering yields the discrete logarithm  $x$  of  $a$  modulo  $|H|$ . The complexity of the entire 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 algorithm with  $M = \lceil \sqrt{q_i} \rceil$  is used.  $\square$

The algorithm is efficient only if  $|H|$  is smooth, i.e., if  $q_i \leq B$  for a small *smoothness bound*  $B$ . If this condition is satisfied we have in the worst case that  $q_i \approx B$  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 trade-off is used.

It is crucial in the following that the algorithm is generic, i.e., that it uses operations in  $H$  and equality tests of group elements only. Shoup showed in [45] that no general-purpose algorithm can solve the DL problem faster than the Pohlig-Hellman algorithm together with the baby-step giant-step trade-off. For special groups such as the multiplicative group of a finite field, more efficient algorithms are known. We refer to [33] for a detailed discussion of the DL problem and algorithms for solving it.

### 3 A general technique for reducing the DL problem to the DH problem

In this section we describe a technique that allows to reduce the DL problem to the DH problem efficiently in groups  $G$  (more precisely, in all groups of certain orders) which satisfy certain conditions.

In Section 3.1 we define the notion of a Diffie-Hellman oracle, and the subsequent sections deal with the problem of computing discrete logarithms in a group  $G$  when given such an oracle for  $G$ . As a preparation for this, it is investigated in Section 3.2 what kind of computations are possible in the exponents (i.e., in the unknown discrete logarithms) of group elements when given a Diffie-Hellman oracle. In Section 3.3, the concept of auxiliary groups is defined, and in Sections 3.4 and 3.5 it is shown that these auxiliary groups are a tool for reducing the DL problem to the DH problem.

#### 3.1 Computing discrete logarithms with an oracle solving the DH problem

In order to prove results concerning the equivalence of breaking the DH protocol and computing discrete logarithms we assume the availability of an oracle that solves the DH problem.

**Definition 1** A *Diffie-Hellman (DH) oracle* 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 the element  $g^{uv}$ .

In the following we describe a polynomial-time reduction of the DL problem to the DH problem for certain classes of groups. 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 we want to compute the discrete logarithm  $s$  using a DH oracle for  $G$ . It is sufficient to compute  $s$  modulo each prime factor of  $|G|$  (or modulo the prime powers if  $|G|$  contains multiple prime factors) and to combine 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  $|G|$ . Hence we can restrict our attention to the problem of computing  $s$  modulo  $p$  for a large prime factor  $p$  of  $|G|$ . We assume that  $p$  is a *single* prime factor of  $|G|$ ; the case of  $|G|$  having multiple large prime factors is discussed in Section 3.5. Let  $x$  be the element of  $GF(p)$  defined by  $s \equiv x \pmod{p}$ . In the following sections, the problem of computing  $x$  from the group element  $g^s$  is investigated.

### 3.2 Computing with implicit representations using a DH oracle

Every element  $y$  of the field  $GF(p)$  can be interpreted as corresponding to a set of elements of  $G$ , namely those whose discrete logarithm is congruent to  $y$  modulo  $p$ . Every element of this set is then a representation of the field element  $y$ .

**Definition 2** Let  $G$  be a cyclic group with a fixed generator  $g$ , and let  $p$  be a prime divisor of the group order. Then, a group element  $a = g^{y'}$  is called an *implicit representation* (with respect to  $G$  and  $g$ ) of the element  $y \in GF(p)$  if  $y \equiv y' \pmod{p}$ . We write  $y \rightsquigarrow a$ .

Note that the implicit representation of a field element is not unique if  $|G| \neq p$ .

The following operations on elements of  $GF(p)$  can be performed efficiently on implicit representations of these elements (i.e., by operating in the group  $G$ ), where the result is also in implicit form. 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

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

(where  $\text{DH}_g$  stands for the DH function with respect to the generator  $g$ ), and these implicitly executed operations on  $GF(p)$ -elements require a 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 an  $e$ -th-power-DH-oracle ( $\text{PDH}_{g,e}$  oracle) that computes an implicit representation of the  $e$ -th power of an implicitly represented element. A possible implementation of a  $\text{PDH}_{g,e}$  oracle is to use a “square and multiply” algorithm for obtaining an implicit representation of  $y^e$ , denoted by  $\text{PDH}_{g,e}(a)$ , 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

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

We call addition, subtraction, multiplication, division, and equality testing in  $GF(p)$  *algebraic* operations. Any efficient computation in  $GF(p)$  can be performed equally efficiently 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

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

or the computation of square roots using an algorithm of Massey [25]. We will crucially rely on the fact that algorithms based on exhaustive search (for example generic algorithms for solving the index-search problem, in particular the DL problem) can be executed on implicitly represented arguments and lead to explicit results.

### 3.3 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 exists. (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  $c$  of  $H$  and to compute the discrete logarithm of this element explicitly. We now define a first required property of the auxiliary group  $H$ .

**Definition 3** A finite group  $H$  is said to be *defined  $(m, \alpha)$ -algebraically over  $GF(p)$*  if the elements of  $H$  can be represented as  $m'$ -tuples (for some  $m' \leq m$ ) of elements of  $GF(p)$  such that the group operation in this representation can be carried out by at most  $\alpha$  algebraic operations in  $GF(p)$ .

We will need the following stronger property for auxiliary groups.

**Definition 4** A group  $H$  is *defined strongly  $(m, \alpha)$ -algebraically over  $GF(p)$*  if  $H$  is defined  $(m, \alpha)$ -algebraically over  $GF(p)$  and if there exist two algorithms, EMBED and EXTRACT, with the following properties.

1. For all  $(x, e) \in GF(p)^2$  the EMBED algorithm with input  $(x, e)$  either outputs a group element  $c$  of  $H$ , or reports failure.

2. If the EMBED algorithm is run with the input  $(x, e)$  for fixed  $x$  and randomly chosen  $e$  until the algorithm does not fail, then the expected running time until an element  $c \in H$  is computed is at most  $\alpha$  algebraic operations in  $GF(p)$ .
3. If the EMBED algorithm does not fail for the input  $(x, e)$ , then

$$\text{EXTRACT}(\text{EMBED}(x, e), e) = x .$$

4. The EXTRACT algorithm runs in time at most  $\alpha$ .

In the examples considered below, the EMBED algorithm computes a group element  $c$  that contains  $x + e$  as a coordinate, and the EXTRACT procedure outputs this particular coordinate minus  $e$ .

In the next section we show how an abelian group  $H$  with bounded rank, defined strongly algebraically over  $GF(p)$ , and with smooth order can be used as an auxiliary group in the reduction of the computation of discrete logarithms modulo  $p$  in  $G$  to breaking the DH protocol for  $G$ .

### 3.4 The reduction algorithm

First we extend the definition of implicit representations from elements of  $GF(p)$  to  $m$ -tuples over  $GF(p)$ .

**Definition 5** Let  $p$  and  $G$  be as above and let  $a_i \in G$  and  $y_i \in GF(p)$  (for  $i = 1, \dots, m$ ). We say that  $(a_1, \dots, a_m)$  is an implicit representation of  $(y_1, \dots, y_m)$  if  $y_i \rightsquigarrow a_i$  for  $1 \leq i \leq m$ .

**Theorem 2** *Let  $P$  be a fixed polynomial. Let  $G$  be a cyclic group with generator  $g$  such that  $|G|$  and its factorization  $|G| = \prod_{i=1}^s p_i^{e_i}$  are known. If there exist  $m$ ,  $\alpha$ , and  $B$ , all upper bounded by  $P(\log |G|)$ , such that every prime factor  $p$  of  $|G|$  greater than  $B$  is single, and for every such  $p$ , a finite abelian group  $H_p$  with rank  $r = O(1)$ , defined strongly  $(m, \alpha)$ -algebraically over  $GF(p)$ , is given whose order  $|H_p|$  is  $B$ -smooth and known, then breaking the DH protocol for  $G$  with respect to  $g$  is probabilistic polynomial-time equivalent to computing discrete logarithms in  $G$  to the base  $g$ .*

*The expected complexity of the computation of a discrete logarithm in  $G$  when given a DH oracle for  $G$  is  $O(m^2 B^r (\log |G|)^2 / \log B + m^2 \alpha (\log |G|)^3)$  group operations in  $G$ ,  $O(m^2 \alpha (\log |G|)^3)$  calls to the DH oracle for  $G$ , and  $O(m^2 \alpha (\log |G|)^3 + m \alpha B^r (\log |G|)^2 / \log B)$  field operations in  $GF(p)$  for  $p \leq |G|$ . The complexities can be reduced by a time-memory trade-off.*

*Proof.*<sup>1</sup> Let  $p$  be a single prime factor of  $|G|$  larger than  $B$ . Assume that an auxiliary group  $H$  is given that is defined strongly  $(m, \alpha)$ -algebraically over  $GF(p)$  with  $B$ -smooth order  $|H| = \prod q_i^{f_i}$ . It is clear that  $H$  has the property that when given an implicitly represented field element  $x \in GF(p)$ , then an implicitly represented *group* element  $c$  of  $H$  (and an explicit element  $e$  of  $GF(p)$ ) can be found efficiently with the property that from the *explicit* representation of  $c$  (and from  $e$ ), the EXTRACT algorithm leads to the element  $x$ . The reason is that because the EMBED procedure uses only algebraic operations, it works also on implicitly represented inputs (where the group element of the output is also implicitly represented). This fact allows to reduce the computation of discrete logarithms in  $G$  (modulo  $p$ ) to the same problem in the group  $H$ . The field element  $x$  is computed from an implicit representation of  $x$  in four steps.

*Step 1.* Use the EMBED algorithm to obtain, when given an implicit representation of  $x$  and a random  $e \in GF(p)$ , an implicit representation of a group element  $c$  of  $H$ .

*Step 2.* Compute the discrete logarithm of  $c$  in  $H$  (with respect to some generator set).

*Step 3.* Compute  $c$  explicitly.

*Step 4.* Use the EXTRACT algorithm to obtain  $x$  explicitly:

$$x = \text{EXTRACT}(c, e) .$$

We have to prove the stated complexity bounds for Step 2. The group  $H$  is abelian of rank  $r$ , i.e.,  $H$  is isomorphic to  $\mathbf{Z}_{n_1} \times \cdots \times \mathbf{Z}_{n_r}$  for some  $n_1, \dots, n_r$  satisfying  $\prod_{j=1}^r n_j = |H|$  and such that  $n_{j+1}$  divides  $n_j$  for  $j = 1, \dots, r-1$ . Let  $h_1, \dots, h_r$  be a set of generators of  $H$  such that  $|\langle h_j \rangle| = n_j$  and  $H$  is the internal product of the cyclic subgroups  $\langle h_1 \rangle, \dots, \langle h_r \rangle$ :

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

(If no generator set for  $H$  is known it can be computed by the method described in Appendix A.)

---

<sup>1</sup>The reader may wish to consult the survey paper [27], where a special case of this theorem is proved. More precisely, the proof is given under the assumption that all the auxiliary groups are cyclic elliptic curves over  $GF(p)$ .

The element  $c \in H$  has a unique representation

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

(the group  $H$  is written additively). We address the problem of computing the coefficients  $k_j$ . This can be done by a generalization of the Pohlig-Hellman algorithm (see Section 2), applied to implicitly represented group elements. 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  $j = 1, \dots, r$ . The algorithm uses  $v_j$  ( $j = 1, \dots, r$ ) as local variables (initialized by  $v_j \leftarrow 0$ ).

For the first step, let  $\alpha_1$  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  $j = 1, \dots, \alpha_1$ . Because  $H$  is defined algebraically over  $GF(p)$ , an implicit representation of

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

can be efficiently computed from an implicit representation of  $c$ . For all  $(t_1, \dots, t_{\alpha_1}) \in \{0, \dots, q-1\}^{\alpha_1}$ , we compute (explicitly)

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

transform the coordinates to an implicit representation, and test equality with  $(n_1/q)c$ . Equality indicates that the  $t_j$  are congruent to the coefficients  $k_j$  modulo  $q$ . We set  $v_j \leftarrow t_j$  for these  $t_j$ , and for  $1 \leq j \leq \alpha_1$ .

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

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

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

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

holds. Then assign

$$v_j \leftarrow t_j q + v_j \quad (j = 1, \dots, \alpha_1) ,$$

$$v_j \leftarrow t_j \quad (j = \alpha_1 + 1, \dots, \alpha_2) .$$

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  $n_j = \text{ord } h_j$  for  $j = 1, \dots, r$ .

After running the algorithm for all primes  $q$  dividing  $|H|$ , one can compute the coefficients  $k_j$  modulo  $\text{ord } h_j$  by Chinese remaindering. The complexity of the algorithm is

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

$O\left(m \frac{B^r}{r \log B} \log |H| \log |G| + \alpha \log |G|\right)$  operations in  $G$ ,

$O(\alpha \log |G|)$  calls to the DH oracle for  $G$ , and

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

The first part of the number of group operations comes from the comparisons of implicitly represented elements of  $H$ . Note that  $|H| \leq p^m$  because  $H$  is defined  $(m, \alpha)$ -algebraically over  $GF(p)$ . The implicit and explicit operations in  $H$  can be further reduced to operations and DH oracle calls in  $G$  and operations in  $GF(p)$ . Then, one obtains the following complexities.

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

$O(m^2 (\log p)^2 \alpha \log |G|)$  calls to the DH oracle for  $G$ , and

$O\left(\left(m^2 (\log p)^2 + m \log p \frac{B^r}{r \log B}\right) \cdot \alpha \log p\right)$  field operations in  $GF(p)$ .

The complexities can be reduced by a time-memory trade-off if memory space is available. The running time is polynomial in  $\log |G|$  because  $m$ ,  $\alpha$ , and  $B$  are polynomial in  $\log |G|$ , and because  $r = O(1)$ .  $\square$

### 3.5 The case of multiple large prime factors in $|G|$

In the previous sections we assumed that all the large prime factors of  $|G|$  are single. Under certain additional conditions one can also treat the 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 if either an additional DH oracle for a certain subgroup of  $G$  is given (Case 1), or if  $p$ -th roots can efficiently be computed in  $G$  (Case 2).

*Case 1.* Assume that a DH oracle for the group  $\langle g^{|G|/p} \rangle$  is given. We write

$$x \equiv \sum_{i=0}^{e-1} x_i p^i \pmod{p^e}$$

with  $x_i \in GF(p)$  for  $i = 0, \dots, e-1$ . Let  $k \leq e-1$ , assume that  $x_0, \dots, x_{k-1}$  are already computed (note that  $x_0$  can be computed as described in the previous section), and consider the problem of computing  $x_k$ . Let  $a' := a \cdot g^{-x_0 - \dots - x_{k-1} p^{k-1}}$ . Then

$$\begin{aligned} a' &= g^{x_0 + x_1 p + \dots + x_{e-1} p^{e-1}} \cdot g^{-x_0 - \dots - x_{k-1} p^{k-1}} \\ &= g^{x_k p^k + x_{k+1} p^{k+1} + \dots + x_{e-1} p^{e-1}} = \left( g^{p^k} \right)^{x_k + p \cdot l} \end{aligned}$$

for some  $l$ . From  $a'$ , compute

$$a'' := (a')^{|G|/p^{k+1}} = \left( g^{|G|/p} \right)^{x_k},$$

and from  $a''$ ,  $x_k$  can be computed as described in the previous section by using the DH oracle for  $\langle g^{|G|/p} \rangle$ . More generally, this also works when a DH oracle for any group  $\langle g^{d \cdot p^{e-1}} \rangle$ , where  $d \cdot p^{e-1}$  divides  $|G|/p$ , is given.

*Case 2.* Assume that  $a'$  (see Case 1) is computed. If an element  $a'''$  of the form

$$a''' = g^{x_k + p \cdot l'}$$

for some  $l'$  is computed,  $x_k$  can again be obtained as in Section 3.4, with the DH oracle for  $\langle g \rangle$ . Such an element  $a'''$  can be obtained by computing a  $p^k$ -th root, i.e.,  $k$  times the  $p$ -th root, of  $a'$ . Any  $p^k$ -th root of  $a'$  is of the required form because  $p$  divides  $|G|/p^k$ .

However, it has been shown that in the model of *generic* algorithms, it is *not* possible to compute discrete logarithms in a group  $G$  more efficiently than in time  $\Omega(\sqrt{p})$  with a DH oracle for  $G$ , if  $p$  is a *multiple* prime factor of  $|G|$  [28]. The model of generic algorithms was introduced by Shoup [45].

Intuitively, a generic algorithm is a general-purpose algorithm that works for all groups of a certain order, and that does not make use of any particular property of the representation of the group elements. Of course this result implies that in the generic model, a DH oracle cannot be efficiently used to construct the required subgroup oracles of Case 1 (a result which was proved already in [45]), and that for large  $p$ ,  $p$ -th roots cannot be computed efficiently by a generic algorithm in a group of which the order is divisible by  $p^2$ , even when a DH oracle is given for this group (Case 2) [28].

## 4 Applicable auxiliary groups over $GF(p)$

In this section, two classes of possible auxiliary groups satisfying the requirements specified in the previous section are described: elliptic curves over finite fields and subgroups of the multiplicative groups of finite fields. The applicability of Jacobians of hyperelliptic curves (see [20] and [9]) as auxiliary groups was demonstrated in [48].

Two types of results are derived as a consequence of the applicability of these classes of groups as auxiliary groups. First, a non-uniform reduction of the DL to the DH problem is shown. Under an unproven assumption on the existence of smooth numbers in small intervals, the complexity of this reduction is polynomial in  $\log|G|$ , i.e., for every group (if no squares of large primes divide the order) there exists an algorithm for computing discrete logarithms in polynomial time if it is allowed to make calls to a DH oracle for this group. As mentioned already, such a reduction does not exist (in the model of generic algorithms) if the group order contains multiple large prime factors.

Moreover, we give a list of expressions  $A(p)$  in  $p$  with the property that an auxiliary group  $H_p$  with order  $A(p)$  over  $GF(p)$  can efficiently be constructed. Theorem 2 then 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). The equivalence of the DH and DL problems holds in a uniform sense for these groups because an efficient reduction algorithm, whose existence is guaranteed by the non-uniform result, can even be found efficiently.

## 4.1 Elliptic curves

### 4.1.1 Applicability as auxiliary groups

Let  $\mathbf{F}$  be a field (whose characteristic is not 2 or 3) and let  $A, B \in \mathbf{F}$  with  $4A^3 + 27B^2 \neq 0$  (in  $\mathbf{F}$ ). 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  $\mathcal{O}$  is called “point at infinity”). Together with a certain operation on the set of points,  $E_{A,B}(\mathbf{F})$  forms an abelian group of rank at most 2. We refer to [35] for an introduction to elliptic curves.

We show that an elliptic curve  $E$  over the field  $GF(p)$  is defined strongly  $(2, O((\log p)^2))$ -algebraically over  $GF(p)$ . Therefore it can be used as an auxiliary group if it has smooth order. Note that the order of an elliptic curve can be computed in polynomial time [44],[6]. The points of  $E$  can be represented as pairs of  $GF(p)$ -elements, and the group operation can be executed in this representation by a constant number of additions, multiplications, divisions, and equality tests in  $GF(p)$ . We describe the EMBED algorithm. Let  $x, e \in GF(p)$  be given. First the expression  $D = (x + e)^3 + A(x + e) + B$  is computed and its quadratic residuosity is tested. If  $D$  is not a quadratic residue, the algorithm reports failure (and a new value for  $e$  is chosen). If  $D$  is a quadratic residue, then a square root  $y$  of  $D$  is computed by an algorithm due to Massey [25] (see Lemma 3). Then the EMBED algorithm outputs  $c = (x + e, y)$ . The necessary executions of the EMBED algorithm require  $O((\log p)^2)$  algebraic operations in  $GF(p)$ .

One can show in a completely analogous manner that an elliptic curve over an extension field  $GF(p^n)$  of  $GF(p)$ , where  $n$  is polynomial in  $\log p$ , can also be used as an auxiliary group.

### 4.1.2 Existence

It is well-known that for any  $A, B \in GF(p)$

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

and that for every  $d \in [p - 2\sqrt{p} + 1, p + 2\sqrt{p} + 1]$ , there exists a cyclic elliptic curve over  $GF(p)$  with order  $d$  [41]. This implies the following non-uniform reduction of the DL problem to the DH problem.

**Definition 6** For a number  $n$ , let  $\nu(n)$  be the minimum, taken over all  $d$  in the interval  $[n - 2\sqrt{n} + 1, n + 2\sqrt{n} + 1]$ , of the largest prime factor of  $d$ .

**Theorem 3** *Let  $P$  be a fixed polynomial. For every finite cyclic group  $G$  with order  $|G| = \prod p_i^{e_i}$  and such that all multiple prime factors  $p_i$  of  $|G|$  are smaller than  $B := P(\log |G|)$ , there exists an algorithm that makes calls to a DH oracle for  $G$  and computes discrete logarithms of elements of  $G$  in time*

$$\max\{\nu(p_i)\} \cdot (\log |G|)^{O(1)} .$$

The quantity  $\nu(p)$  is directly linked with the existence of a smooth number in the interval  $[p - 2\sqrt{p} + 1, p + 2\sqrt{p} + 1]$ . Unfortunately, very little is known about smooth numbers in such intervals. However, it is known [8] that for every fixed  $u$ ,

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

where  $\psi(n, y)$  denotes the number of  $y$ -smooth integers  $\leq n$ . This fact suggests that  $\nu(n)$  is polynomial in  $\log n$ .

**Smoothness Assumption.**  $\nu(n) = (\log n)^{O(1)}$ .

This assumption implies that the algorithms of Theorem 3 run in time polynomial in  $\log |G|$ , and this yields a polynomial-time non-uniform reduction of the DL problem to the DH problem for all groups whose orders are free of multiple large prime factors. Moreover, the reduction algorithms are generic, i.e., they depend only on the group order  $|G|$  of  $G$ , and they have a description of length linear in  $\log |G|$ , namely the large prime factors of  $|G|$  and parameters of suitable elliptic curves.

**Corollary 4** *Let  $P$  be a fixed polynomial. If the Smoothness Assumption is true, then for every group  $G = \langle g \rangle$  whose order is free of multiple prime factors greater than  $B := P(\log |G|)$ , there exists a side-information string  $S$  of length at most  $3 \log |G|$  such that when given  $S$ , breaking the DH protocol for  $G$  is polynomial-time equivalent to computing discrete logarithms in  $G$ .*

*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 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 unproven assumption. The reason is that in [1] the existence of Jacobians with *prime* order is proved, whereas Jacobians with *smooth* order are required for our purpose.

In the model of generic algorithms the results described in Section 3.5 (see [28],[45]) and in this section imply the following complete characterization of group orders  $n$  for which there exists an efficient generic algorithm computing discrete logarithms, making calls to a DH oracle for the same group.

**Corollary 5** *If the Smoothness Assumption is true, then there exists a polynomial-time generic algorithm computing discrete logarithms in cyclic groups of order  $n$ , making calls to a DH oracle for the same group, if and only if all the multiple prime factors of  $n$  are of order  $(\log n)^{O(1)}$ .*

### 4.1.3 Construction of elliptic curves

For certain expressions  $A(p)$ , elliptic curves over  $GF(p)$  with order  $A(p)$  can explicitly be constructed. The curve over  $GF(p)$  defined by the equation

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

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

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

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 constructible over  $GF(p)$ . (We will show in the next section that the subgroup of order  $p + 1$  of  $GF(p^2)^*$  is a useful auxiliary group for all  $p$ .) The following statements about the orders of curves of the form (2) or (3) in the case they are *not*  $p + 1$  are proved in [19].

If  $p \equiv 1 \pmod{4}$ , then  $p$  can uniquely be represented as the sum of two squares, i.e.,  $p = a^2 + b^2$ . Then the curves  $y^2 = x^3 - Dx$  have the orders

$$p + 1 \pm 2a, \quad p + 1 \pm 2b, \tag{4}$$

and the four orders occur equally often over the choices of  $D$ .

If  $p \equiv 1 \pmod{3}$ , then  $p$  can uniquely be represented as  $p = a^2 - ab + b^2$  with  $a \equiv 2 \pmod{3}$  and  $b \equiv 0 \pmod{3}$ . Then the curves  $y^2 = x^3 + D$  have the orders

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

and the six orders occur equally often over the choices of  $D$ .

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

## 4.2 Subgroups of the multiplicative group of an extension field of $GF(p)$

In this section we investigate under what conditions a subgroup  $H$  of  $GF(p^n)^*$  satisfies the properties of an auxiliary group in the technique for reducing the DL problem to the DH problem.

### 4.2.1 Representation with normal bases

We refer to [24] or [34] for an introduction to finite fields. For a prime power  $q$ , the field  $GF(q^n)$  is an  $n$ -dimensional vector space over  $GF(q)$  and hence its elements can be represented as  $n$ -tuples of  $GF(q)$ -elements with respect to some basis. Let  $\alpha$  be an element of  $GF(q^n)$ , and let  $\alpha_i := \alpha^{q^i}$  for  $i = 0, \dots, n-1$ . Then  $\{\alpha_0, \dots, \alpha_{n-1}\}$  is called a *normal basis* if it is linearly independent in which case  $\alpha$  is called a *normal element*. Let  $\vec{\alpha} := (\alpha_0, \dots, \alpha_{n-1})$ . The matrix  $T$  in  $(GF(q))^{n \times n}$  satisfying  $\alpha_0 \vec{\alpha} = \vec{\alpha} T$  is called the *multiplication table* of the basis.

Normal elements can be found efficiently by trial and error, and when given  $q$ ,  $n$ , and a normal element  $\alpha \in GF(q^n)$ , the multiplication table can be determined by solving a system of linear equations over  $GF(q)$ .

### 4.2.2 The use of subgroups $H$ of $GF(p^n)^*$ as auxiliary groups

Let  $H$  be a subgroup of  $GF(p^n)^*$ . We derive conditions under which such subgroups are defined strongly algebraically over  $GF(p)$ . The group operation of  $H$  is a multiplication in  $GF(p^n)^*$  and requires, in a normal basis representation,  $O(n^3)$  multiplications in  $GF(p)$ . We conclude that every subgroup of  $GF(p^n)^*$  (for  $n$  polynomial in  $\log p$ ) is defined  $(n, (\log p)^{O(1)})$ -algebraically over  $GF(p)$ . For all  $n$ ,  $GF(p^n)^*$  is a cyclic group. This implies that a subgroup of  $GF(p^n)^*$  is uniquely determined by its order  $|H|$ , or more precisely, for every divisor  $d$  of  $|H|$  there exists exactly *one* subgroup of  $GF(p^n)^*$  with  $|H| = d$ . Furthermore, all these subgroups are cyclic.

The next theorem states conditions on  $n$  and  $|H|$  under which  $H$  is defined *strongly* algebraically over  $GF(p)$ .

**Theorem 6** *Let  $P$  be a fixed polynomial and  $c$  be a fixed constant. Let  $H$  be the subgroup of  $GF(p^n)^*$  of order  $|H|$ . Then  $H$  is defined strongly  $(m, \alpha)$ -algebraically over  $GF(p)$  for  $m, \alpha = (\log p)^{O(1)}$  if one of the following two conditions is satisfied.*

*Condition 1.*  $n \leq P(\log p)$ , and there exists a divisor  $k < n$  of  $n$  such that

$$|H| = \frac{p^n - 1}{p^k - 1} = p^{n-k} + p^{n-2k} + \dots + p^k + 1 .$$

*Condition 2.*  $n \leq c$ , and there exists a non-constant polynomial  $f(x)$  (with integer coefficients) dividing  $x^n - 1$  such that  $|H| = f(p)$ .

*Remark.* An alternative formulation of Condition 2 is that  $|H|$  is a multiple of  $\Phi_n(p)$  for some  $n = O(1)$ , where  $\Phi_n$  stands for the  $n$ -th cyclotomic polynomial (see [24] and Appendix B). Examples are

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

The alternating sums

$$p^{2l} - p^{2l-1} + - \dots - p + 1$$

also satisfy Condition 2 for  $l = O(1)$ .

*Proof.* We show that if one of the conditions is satisfied there exists an EMBED algorithm that takes as input two elements  $x$  and  $e$  of  $GF(p)$  and computes coordinates  $\beta_1, \dots, \beta_{n-1}$  (still in the normal basis representation) in  $GF(p)$  by a polynomial number of algebraic operations such that  $\beta = (x + e, \beta_1, \dots, \beta_{n-1}) \in H$ .

One possibility of designing the EMBED algorithm is to express membership of an element  $\beta = (\beta_0, \dots, \beta_{n-1})$  to the subgroup  $H$  by an equation (or a system of equations) in the coordinates. Then, the element  $x + e$  can be assigned to one of the coordinates, say  $\beta_0$ , and the equation is solved for the remaining coordinates (by using only algebraic operations in the field  $GF(p)$ ).

For an element  $\beta$  of  $GF(p^n)^*$ , we have that  $\beta \in H$  if and only if  $\beta^{|H|} = 1$ . Clearly, this equation corresponds to a set of polynomial equations (with coefficients in  $GF(p)$ ) in the coordinates  $\beta_i$ .

We will show that if the first condition is satisfied, then it is sufficient to solve one univariate polynomial equation over a subfield  $GF(p^k)$  of  $GF(p^n)$  for finding such a  $\beta$ , and that this can be reduced to a polynomial number of algebraic operations in the field  $GF(p)$ . The situation when the second

condition is satisfied is more difficult. Here, a *system of multivariate* polynomial equations over  $GF(p)$  has to be solved by algebraic operations. This can be achieved by Gröbner basis computations. The proof that Condition 2 is sufficient is given in Appendix B.

*Proof that Condition 1 is sufficient.* The EMBED algorithm works as follows in this situation. Let  $x, e \in GF(p)$  be given. For  $l := n/k$  let  $\{\alpha_0, \dots, \alpha_{k-1}\}$  and  $\{\alpha'_0, \dots, \alpha'_{l-1}\}$  be normal bases of  $GF(p^k)$  over  $GF(p)$  and of  $GF(p^n)$  over  $GF(p^k)$ , respectively. For an element  $\beta = (\beta_0, \dots, \beta_{l-1}) \in GF(p^n)$  (with  $\beta_i \in GF(p^k)$ ), we have that  $\beta \in H$  is equivalent to

$$\beta^{(p^n-1)/(p^k-1)} = 1. \quad (6)$$

Equation (6) is equivalent to

$$\left( \sum_{i=0}^{l-1} \beta_i \alpha'_i \right)^{p^{(l-1)k} + p^{(l-2)k} + \dots + p^k + 1} = 1. \quad (7)$$

Now, we have  $(\beta_i)^{p^{jk}} = 1$  (because  $\beta_i \in GF(p^k)$ ) and  $(\alpha'_i)^{p^{jk}} = \alpha'_{i+j}$  (where the index is reduced modulo  $l$ ) by the definition of the normal basis. Hence (7) is equivalent to

$$\left( \sum_{i=0}^{l-1} \beta_i \alpha'_{i+l-1} \right) \cdot \left( \sum_{i=0}^{l-1} \beta_i \alpha'_{i+l-2} \right) \cdots \left( \sum_{i=0}^{l-1} \beta_i \alpha'_i \right) = 1 \quad (8)$$

(where the indices are reduced modulo  $l$ ).

Because  $(\beta^{(p^n-1)/(p^k-1)})^{p^k-1} = \beta^{p^n-1} = 1$ ,  $\beta^{(p^n-1)/(p^k-1)}$  is an element of the subfield  $GF(p^k)$  of  $GF(p^n)$ . Such elements are easy to characterize in terms of their coordinates. An element  $(\gamma_0, \dots, \gamma_{l-1})$  is an element of  $GF(p^k) \subset GF(p^n)$  if and only if  $\gamma_0 = \gamma_1 = \dots = \gamma_{l-1}$ . The reason for this fact is that  $\alpha'_0 + \alpha'_1 + \dots + \alpha'_{l-1}$  (the trace  $\text{Tr}(\alpha'_0)$  of  $\alpha'_0$ ) is also an element of  $GF(p^k)$ . Because both  $\beta^{(p^n-1)/(p^k-1)}$  and 1 are elements of  $GF(p^k)$ , they are equal if and only if their first coordinates are equal. The equation coming from (8), restricted to the first coordinate, is equivalent to

$$g(\beta_0, \dots, \beta_{l-1}) = 1/\text{Tr}(\alpha'_0) \quad (9)$$

for some  $l$ -degree polynomial  $g$  with  $GF(p^k)$ -coefficients.

The construction of a group element  $\beta$  of  $H$  with the desired property now works as follows. Let the first coordinate  $\beta_{0,0}$  of  $\beta_0$  (note that  $\beta_i$

corresponds to a  $k$ -tuple  $(\beta_{i,0}, \dots, \beta_{i,k-1})$  of  $GF(p)$ -elements with respect to the normal basis  $\alpha_0, \dots, \alpha_{k-1}$  be equal to  $x + e$ . Choose the coefficients  $\beta_{0,1}, \dots, \beta_{0,k-1}$  and the coefficients  $\beta_1, \dots, \beta_{l-2}$  randomly in  $GF(p)$  and in  $GF(p^k)$ , respectively. Then (9) is equivalent to a polynomial equation for  $\beta_{l-1}$  with coefficients in  $GF(p^k)$ .

The roots of a polynomial  $f(\gamma)$  over a finite field  $GF(p^k)$  can be computed in probabilistic polynomial time by the Cantor-Zassenhaus algorithm (see [34],[24]). The key idea of this algorithm 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 random  $\delta \in GF(p^k)$ . This is repeated with different  $\delta$  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  $GF(p^k)$ , and the latter can be reduced to algebraic operations (and equality tests) in  $GF(p)$  (with respect to the normal basis representation). The expected number of solutions for  $\beta_{l-1}$  is roughly 1 because  $|H|/p^n \approx 1/p^k$ . If no solution is found, then failure is reported.

Because the Cantor-Zassenhaus algorithm has probabilistic running time polynomial in  $n$  and  $\log p$  and uses only algebraic operations in  $GF(p)$ , the required executions of the EMBED procedure run in a probabilistic polynomial (in  $\log |G|$ ) number of algebraic operations if  $n$  is polynomial in  $\log p$ .  $\square$

### 4.3 Summary

The following corollary is an immediate consequence of Theorem 2, combined with the results of this section.

**Corollary 7** *Let  $P$  be a fixed polynomial, let  $G$  be a cyclic group with generator  $g$ , and let  $B := P(\log |G|)$ . Then there exists a list of expressions  $A(p)$  in  $p$  with the following property: if every prime factor  $p$  of  $|G|$  greater than  $B$  is single and if for every such prime factor at least one of the expressions  $A(p)$  is  $B$ -smooth, then breaking the DH protocol in  $G$  with respect to  $g$  is polynomial-time equivalent to computing discrete logarithms in  $G$  to the base  $g$ . The list contains the following expressions:*

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

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

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

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

if  $p \equiv 1 \pmod{3}$ , where  $p = a^2 - ab + b^2$ ,  $a \equiv 2 \pmod{3}$ , and  $b \equiv 0 \pmod{3}$ ,

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

where  $k, l = (\log p)^{O(1)}$ , and

$$f(p) ,$$

where  $f(x) \in \mathbf{Z}[x]$  is a non-constant polynomial dividing  $x^n - 1$  for some  $n = O(1)$ .  $\square$

## 5 Equivalence between variants of the DH problem

### 5.1 Introduction

In the previous sections we have proved results concerning the relationship between the security of the DH protocol and the hardness of the DL problem. However, in order to prove that the DH protocol is secure for a group in which the DL problem is hard, one has to show that the DH problem cannot be solved efficiently even with small probability of, say, 1%. Motivated by this, we show in this section that the assumption of a *perfect* DH oracle for the reduction process is unnecessarily strong and can be relaxed in many ways. In 5.2 we prove that a (probabilistic) DH oracle answering correctly with small probability is virtually as strong as a perfect DH oracle. For example, an oracle answering correctly with probability 1% can efficiently be transformed into an oracle that answers correctly with arbitrarily high probability.

In Section 5.3, it is shown that the same holds for a DH oracle that answers correctly for the input  $(g^u, g^v)$  only if  $u = v$ . Finally, the relationship between the DH problem in  $G$  and in subgroups of  $G$  is investigated in Section 5.4.

### 5.2 $\varepsilon$ -DH-oracles

This section deals with DH oracles that answer correctly only with small (but non-negligible) probability. It is shown that such oracles are virtually

as strong as perfect DH oracles. The problem of correcting faulty DH oracles was considered independently by Shoup [45], who described a quite different approach. We introduce the notion of an  $\varepsilon$ -DH-oracle for a cyclic group  $G$  with respect to a generator  $g$ . Note that such an “oracle” is probabilistic in general rather than deterministic.

**Definition 7** For  $\varepsilon > 0$ , an  $\varepsilon$ -DH-oracle is a probabilistic oracle which returns for an input  $(g^u, g^v)$  the correct answer  $g^{uv}$  with probability at least  $\varepsilon$  if the input is uniformly distributed over  $G \times G$ .

The *offset* of the oracle’s answer  $g^t$  to the input  $(g^u, g^v)$  is defined as  $t - uv \pmod{|G|}$ . A *translation-invariant*  $\varepsilon$ -DH-oracle is an  $\varepsilon$ -DH-oracle whose offset distribution is the same for every input  $(g^u, g^v)$ .

A special case of (non-translation-invariant)  $\varepsilon$ -DH-oracles are *deterministic* oracles answering correctly for a fraction  $\varepsilon$  of all inputs. We proceed in two steps to prove that an  $\varepsilon$ -DH-oracle can be transformed into a virtually perfect DH oracle. First, the oracle is made translation-invariant by randomization of the input, and then, the translation-invariant oracle is “amplified” to an (almost) perfect oracle.

**Lemma 1** *An  $\varepsilon$ -DH-oracle for a cyclic group  $G$  with order  $|G|$  can efficiently be transformed into a translation-invariant  $\varepsilon$ -DH-oracle. More precisely, implementing one call to 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  $\varepsilon$ -DH-oracle with randomized input is thus a translation-invariant  $\varepsilon$ -DH-oracle.  $\square$

*Remark.* If  $|G|$  is unknown the input can also be randomized, where the random numbers are chosen from a larger interval. The resulting  $\varepsilon$ -DH-oracle is then “almost translation-invariant” and applicable in the proof of Theorem 8 if the interval is of size at least  $2 \cdot |G|/(\varepsilon^2 \cdot \min\{s, 0.1\})$  (where  $s$  is as in Theorem 8). This is the reason for the greater number of group operations for this case in Theorem 8.

In the proof of Theorem 8 it is shown that a translation-invariant  $\varepsilon$ -DH-oracle can be transformed into an almost-perfect DH oracle. The straightforward approach to using a translation-invariant  $\varepsilon$ -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 must be used. (The Diffie-Hellman decision problem, which was first mentioned in [5], 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 the DH problem.)

**Theorem 8** *For every cyclic group  $G$  with generator  $g$  and known order  $|G|$  and for every  $\beta > 0$  there exists an algorithm for solving the DH problem in  $G$  which makes calls to an  $\varepsilon$ -DH-oracle and whose answer is correct with probability at least  $1 - \beta$ . The number of required oracle calls is  $O(\log(1/\beta\varepsilon)/\varepsilon^4)$ . If the order of  $G$  is unknown, then the reduction is also possible if all the prime factors of  $|G|$  are greater than  $(1 + s)/\varepsilon$  for some  $s > 0$ . The number of required calls to the  $\varepsilon$ -DH-oracle is then*

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

*The number of required group operations is  $O(\log |G|)$  times the number of oracle calls if  $|G|$  is known and  $O(\log(|G|/(\varepsilon^2 \cdot \min\{s, 1\})))$  times this number if  $|G|$  is not known, respectively.*

For the proof of Theorem 8 we need the following lemma.

**Lemma 2** *Let  $X_1, X_2, X_3, \dots$  be independent binary random variables with identical distribution  $P_{X_i}$  with  $P_{X_i}(1) = p$ . Let further  $\alpha, \delta' > 0$ . If  $t$  is the smallest number such that the event*

$$\frac{X_1 + \dots + X_t}{t} \in [p - \delta', p + \delta']$$

*has probability at least  $1 - \alpha$ , then  $t = O(\log(1/\alpha)/\delta'^2)$ .*

*Proof.* Since the random variables  $X_i$  are independent, we have

$$\frac{\text{Var}(X_1 + \dots + X_t)}{t} = \frac{t \cdot \text{Var}(X_i)}{t^2} = \Theta(1/t).$$

Hence the number of standard deviations corresponding to  $\delta'$  is of order  $\Theta(\delta'\sqrt{t})$ . The normal approximation of the binary distribution (see for example [15]) leads to  $\delta'\sqrt{t} = \Theta((\log(1/\alpha))^{1/2})$  or  $t = \Theta(\log(1/\alpha)/\delta'^2)$ .  $\square$

*Proof of Theorem 8.* The basic idea of the amplification of the DH oracle is as follows. In a precomputation phase, which is independent of the actual input, the oracle's offset distribution is determined. Then, the oracle is called with the given input to compute the correct solution with overwhelming probability.

More precisely, the reduction from an  $\varepsilon$ -DH-oracle to an oracle answering correctly with high probability consists of the following steps which we first describe intuitively.

*Step 1.* The  $\varepsilon$ -DH-oracle is transformed into a *translation-invariant*  $\varepsilon$ -DH-oracle.

*Step 2.* We compute an estimate  $\varepsilon'$  for the probability that the (translation-invariant) oracle answers correctly.

*Step 3.* A list  $L_1$  of group elements  $g^e$  is computed with the property that  $g^e$  is contained in  $L_1$  if and only if the probability of the offset  $e$  is close to  $\varepsilon'$ .

*Step 4.* A second list  $L_2$  of group elements is generated which contains exactly those group elements that occur with frequency close to  $\varepsilon'$  when the oracle is called with the input  $(g^u, g^v)$ .

*Step 5.* The lists  $L_1$  and  $L_2$  have (with high probability) the property that the elements of  $L_2$  are exactly the elements of  $L_1$  multiplied by the group element  $g^{uv}$  (which is itself contained in  $L_2$ ). In order to determine this switch element, the lists  $aL_1$  are generated for all elements  $a$  in  $L_2$  (the list  $aL_1$  contains exactly the elements  $al_1$ , where  $l_1$  is contained in  $L_1$ ). The list  $L_2$  is compared to all the lists  $aL_1$ , and equality yields a candidate  $a$  for  $g^{uv}$ .

*Step 6.* In case of one single candidate  $a$  for  $g^{uv}$ , this is the output of the algorithm. In the case of *several* candidates and if the group order  $|G|$  is known, the discrete logarithms of all the candidates and of  $g^u$  and  $g^v$  are determined modulo the smooth part of  $|G|$ . This yields the correct candidate for  $g^{uv}$ , which is then the output of the algorithm.

Note that the first three steps are a precomputation which is independent of the particular input  $(g^u, g^v)$ . The list  $L_1$  which is generated in these

steps is a reference list describing the offset behavior of the faulty oracle. We describe the steps in detail and analyze their correctness and efficiency.

*Step 1.* According to Lemma 1, one can construct a translation-invariant  $\varepsilon$ -DH-oracle which uses  $O(\log |G|)$  group operations and one call to an  $\varepsilon$ -DH-oracle per call if  $|G|$  is known. If  $|G|$  is unknown, the number of group operations is  $O(\log(|G|/(\varepsilon^2 \cdot \min\{s, 1\})))$ .

*Step 2.* Let  $\alpha := \beta\varepsilon/8$ . An event with probability at least  $1 - \alpha$  will be called *almost certain*. Let  $\delta := \varepsilon/10$  and  $\delta' := \delta\varepsilon/100 = \varepsilon^2/1000$ .<sup>2</sup> If  $\varepsilon$  is not known, we take a lower bound. In order to determine the probability of a correct answer, the translation-invariant oracle is called repeatedly with the input  $(g^0, g^0)$ , and  $\varepsilon'$  is the fraction of correct answers  $g^0$ . The number  $t$  of oracle calls is such that the true probability of a correct answer lies almost certainly in the interval  $[\varepsilon' - \delta', \varepsilon' + \delta']$ . It follows from Lemma 2 that  $t = O(\log(1/\alpha)/\delta'^2)$ .

*Step 3.* In this step the reference list  $L_1$  is generated as follows. The faulty oracle is called  $t$  times (for the same value of  $t$  as in the previous step), and all the occurring group elements are stored. Let the list  $L_1$  consist of those group elements whose fraction in the set of all answers lies in the interval  $[\varepsilon' - (\delta + \delta'), \varepsilon' + (\delta + \delta')]$ . According to Lemma 2, and because  $2/\varepsilon$  is an upper bound on the number of offsets occurring with probability at least  $\varepsilon/2$ , with probability  $(1 - \alpha)^{4/\varepsilon}$  the following two statements are both true.

1. If  $e$  is an offset with probability in  $[\varepsilon' - \delta, \varepsilon' + \delta]$ , then  $g^e$  is contained in  $L_1$ .
2. If  $g^e$  is in  $L_1$ , then the offset  $e$  has probability in  $[\varepsilon' - (\delta + 2\delta'), \varepsilon' + (\delta + 2\delta')]$ .

*Step 4.* The translation-invariant faulty oracle is called repeatedly with the input  $(g^u, g^v)$ , where  $(g^u, g^v)$  is the input to the DH algorithm for  $G$ . Let the list  $L_2$  then consist of those group elements which occur as answers of the oracle with a frequency in  $[\varepsilon' - (\delta + 3\delta'), \varepsilon' + (\delta + 3\delta')]$ . Then, for the same number of trials  $t$  as in the previous step, with probability at least  $(1 - \alpha)^{4/\varepsilon}$  the following statements are true.

---

<sup>2</sup>The proof does not depend on the choice of the constants (e.g.,  $1/10$ ), which is somewhat arbitrary. Intuitively, we need that  $\delta \ll \varepsilon$  and  $\delta' \ll \varepsilon\delta$ .

1. If  $e$  is an offset with probability in  $[\varepsilon' - (\delta + 2\delta'), \varepsilon' + (\delta + 2\delta')]$ , then  $g^{uv+e}$  is contained in  $L_2$ .
2. If  $g^{uv+e}$  is in  $L_2$ , then the offset  $e$  has probability in  $[\varepsilon' - (\delta + 4\delta'), \varepsilon' + (\delta + 4\delta')]$ .

*Step 5.* With high probability, the list  $L_2$  is equal to  $L_1$ , switched by  $g^{uv}$  (which is itself in  $L_2$ ). This allows to determine  $g^{uv}$ . More precisely, it follows from the above that the probability that  $L_1$  contains *all* the offsets which have their probability in the interval  $[\varepsilon' - \delta, \varepsilon' + \delta]$ , that all the offsets of  $L_1$  also occur in  $L_2$ , and that all the offsets of  $L_2$  have probability in  $[\varepsilon' - (\delta + 4\delta'), \varepsilon' + (\delta + 4\delta')]$  is at least

$$(1 - \alpha)^{8/\varepsilon} \geq 1 - \frac{8\alpha}{\varepsilon} = 1 - \beta. \quad (10)$$

If this is fulfilled, then  $L_2$  contains more elements than  $L_1$  only if there exists an offset whose probability is in the set

$$[\varepsilon' - (\delta + 4\delta'), \varepsilon' - \delta] \cup [\varepsilon' + \delta, \varepsilon' + (\delta + 4\delta')]. \quad (11)$$

In this case we replace  $\delta$  by  $\delta + i \cdot 5\delta'$  (for an integer  $i$  randomly chosen in  $[-2/\varepsilon, 2/\varepsilon]$ ), leave  $\delta'$  unchanged, and run the entire algorithm (except Steps 1 and 2) again. Because the sets (11) are disjoint for different  $i$ , and because there can be at most  $2/\varepsilon$  offsets with probability at least  $\varepsilon/2$ ,  $L_1$  and  $L_2$  contain the same number of elements for at least half of the possible choices for  $i$ .

If the lists  $L_1$  and  $L_2$  have equal length, then with probability at least  $1 - \beta$  we have that  $g^{uv}$  is contained in  $L_2$  and  $L_2 = g^{uv}L_1$ , i.e.,  $L_2$  contains exactly the elements  $g^{uv}l_1$  for  $l_1$  in  $L_1$ . The lists  $aL_1$  are computed for all elements  $a$  of  $L_2$  and compared to  $L_2$ . If equality holds, then  $a$  is a candidate for  $g^{uv}$ .

*Step 6.* Let  $c$  be the number of elements of  $L_1$  and  $L_2$ . If there exists only *one* candidate for  $g^{uv}$ , then this group element is the output of the algorithm. If there exist *several* such elements, this means that the lists have a non-trivial translation symmetry, or more precisely, that they are invariant under a multiplication with  $g^{|G|/c'}$  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

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

We show that if  $|G|$  is known, the correct one of them can be determined. Let  $p_1, \dots, p_l$  be the distinct prime factors of  $c'$  (they can be found in time  $O((\log(1/\varepsilon))^2/\varepsilon)$  because  $c' = O(1/\varepsilon)$ ), i.e.,  $c' = \prod_{i=1}^l p_i^{f_i}$ . Let further  $d = \prod_{i=1}^l p_i^{e_i}$  be the product of the maximal powers of the  $p_i$  dividing  $|G|$ . The number  $d$  can be computed in time  $O((\log |G|)^3)$  and is  $(2/\varepsilon)$ -smooth because  $c' \leq c \leq 2/\varepsilon$ . Hence  $u$  and  $v$  (and consequently  $uv$ ) are computable modulo  $d$  from  $g^u$  and  $g^v$  by the Pohlig-Hellman algorithm by  $O((\log |G|)^2 + \log |G|/\varepsilon)$  group operations. Analogously, 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  $d$ . This is true because 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 the  $p_i^{f_i}$  are relatively prime, and hence every  $\prod_{i=1}^l p_i^{f_i}$ -th candidate, that is *exactly one* of them, has the correct remainder. This group element is the output of the algorithm.

In the case where  $|G|$  is *not* known, this last step of finding the correct candidate does not work. The only possibility is to choose a smaller value for  $\delta$ . This is always successful if all the prime factors of  $|G|$  are greater than  $(1+s)/\varepsilon$  for some positive  $s$ . Then  $\delta$  must be chosen smaller than  $s\varepsilon/2$ , such that  $1/(\varepsilon' - (\delta + 2\delta')) < (1+s)/\varepsilon$  holds. The last inequality implies that such a symmetry of the lists  $L_1$  and  $L_2$  (this symmetry is a necessary condition for the case of more than one candidate for  $g^{uv}$ ) is not possible.  $\square$

*Remark.* Examples of  $\varepsilon$ -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.

Note that a DH oracle as obtained in Theorem 8 is virtually equivalent to a perfect DH oracle in a polynomial-time (or subexponential-time) reduction of the DL to the DH problem because the correctness of the output of a probabilistic algorithm computing discrete logarithms can be tested, and because only a polynomial (or subexponential) number of oracle calls is required for the computation of a discrete logarithm.

### 5.3 The squaring oracle

We describe an example of an oracle that is weaker than an  $\varepsilon$ -DH-oracle with respect to the fraction of correctly answered inputs. 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*.

From  $g^u$  and  $g^v$  one can compute  $g^{u+v} = g^u \cdot g^v$ , and with the squaring-DH-oracle

$$g^{(u+v)^2} \cdot \left(g^{(u^2)}\right)^{-1} \cdot \left(g^{(v^2)}\right)^{-1} = g^{(u+v)^2 - u^2 - v^2} = g^{2uv} = (g^{uv})^2. \quad (12)$$

When given  $|G|$ , the square root  $g^{uv}$  of  $(g^{uv})^2$  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 efficiently (see Lemma 3). 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  $v$ , and hence  $uv$ , modulo  $2^e$  with  $O((\log |G|)^2)$  group operations by the Pohlig-Hellman algorithm. 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 root  $g^{uv}$  by computing the discrete logarithms of one of the 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  $\varepsilon$  (an  $\varepsilon$ -*squaring-DH-oracle*) can be transformed into a translation-invariant  $\varepsilon^3$ -DH-oracle by randomizing the inputs in (12). The complexity is  $O((\log |G|)^2)$  group operations per call. This proves the following theorem.

**Theorem 9** *For every cyclic group  $G$  with generator  $g$  and known order  $|G|$  and for every  $\beta > 0$  there exists an algorithm solving the DH problem in  $G$  which makes calls to an  $\varepsilon$ -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  $O((\log |G|)^2)$  times the number of oracle calls.*

## 5.4 The security of subgroups

Throughout this section we assume that the order of  $G$  and its factorization are 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 12 below is very intuitive (and an analogous result

holds for the computation of discrete logarithms), the proofs of Theorems 10 and 11 are not trivial. First we give a criterion when a DH oracle for  $\langle g \rangle$  can be efficiently transformed into a DH oracle for  $\langle g^r \rangle$ . More precisely, we will show that a subgroup of  $G$  is at most as secure as  $G$  with respect to the DH protocol if every large prime factor of the index of the subgroup occurs with the same multiplicity in the index and in the group order. We need the following lemma on the computation of  $p$ -th roots in a cyclic group  $G$  if  $p$  is a multiple prime factor of  $|G|$ . Note that for *single* prime factors  $p$  of  $|G|$ , a  $p$ -th root can be obtained by computing the  $z$ -th power for  $z \equiv p^{-1} \pmod{|G|/p}$ .

**Lemma 3** *Let  $G$  be a cyclic group with generator  $g$ , and let  $p$  be a multiple 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 [25] can be generalized as follows. Let  $|G| = p^j s$  (where  $j \geq 2$  and  $(p, s) = 1$ ), and let  $h$  be a  $p$ -th power in  $G$ . By the Pohlig-Hellman algorithm 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  $G$ .  $\square$

*Remark.* When memory space is available, this algorithm can be sped up to  $O(\sqrt{p} \cdot (\log |G|)^{O(1)})$  by the baby-step giant-step trade-off in the Pohlig-Hellman algorithm. This running time is optimal: it was shown in [28] that no generic algorithm can compute  $p$ -th roots substantially faster in a group whose order is divisible by  $p^2$  (even when given a DH oracle for this group).

**Theorem 10** *Let  $P$  be a fixed polynomial. Let  $G$  be a cyclic group with generator  $g$ . If the number  $r$  is such that every prime factor of  $r$  is either smaller than  $B := P(\log |G|)$  or has at least the same multiplicity in  $r$  as in  $G$ , then there exists an algorithm solving the DH problem in the group  $\langle g^r \rangle$ , making one call to the DH oracle for  $\langle g \rangle$  and using a polynomial number of group operations per call.*

*Remark.* Again, the conditions of the theorem are optimal. Shoup [45] has shown that if the conditions are not satisfied, then the construction of a

subgroup oracle from an oracle for  $G$  is hard in the generic model.

*Proof.* Let  $|G| = \prod p_i^{e_i}$  and  $r = \prod p_i^{f_i}$  (where  $f_i > e_i$ ,  $e_i = 0$ , or  $f_i = 0$  is possible). The DH algorithm for the group  $\langle g^r \rangle$  takes as inputs two elements  $(g^r)^a$  and  $(g^r)^b$  and must output  $(g^r)^{ab}$ . Using the DH oracle for the group  $G = \langle g \rangle$  with the same input, one obtains  $g^{r^2 ab}$ , i.e., the  $r$ -th power of  $g^{r ab}$ . Now,  $g^{r ab}$  is computed from  $g^{r^2 ab}$  by computing the  $r$ -th root. More precisely, the  $p_i^{f_i}$ -th root of  $g^{r^2 ab}$  has to be computed for all  $i$  with  $f_i > 0$ , and the correct root, i.e., the particular root that is a power of  $g^{r ab}$ , must be determined. Assume that we have already computed

$$g^{p_1^{f_1} \dots p_{i-1}^{f_{i-1}} p_i^{2f_i} \dots p_s^{2f_s} ab} = g^{c p_i^{2f_i} ab} =: d_i ,$$

where  $c$  is explicitly known. We describe the computation of the correct  $p_i^{f_i}$ -th root of this group element separately for the cases  $f_i \geq e_i$  and  $p_i \leq B$ .

*Case 1:  $f_i \geq e_i$ .* We compute  $z$  with

$$z := (p_i^{f_i})^{-1} \pmod{|G|/p_i^{e_i}} ,$$

and  $d_i^z$ , which is the desired group element. First, it is a  $p_i^{f_i}$ -th root of  $d_i$ . Additionally, it is the only  $p_i^{f_i}$ -th root of this element which is a power of  $g^{p_i^{f_i}}$  (the  $p_i^{e_i} - 1$  different roots are

$$g^{c p_i^{2f_i} ab z + i |G|/p_i^{e_i}}$$

for  $i = 1, \dots, p_i^{e_i} - 1$ , and they are not even powers of  $g^{p_i^{e_i}}$ ).

*Case 2:  $p_i \leq B$  and  $f_i < e_i$ .* Here we repeat the following two steps  $f_i$  times.

*Step 1.* Compute the  $p_i$ -th roots of the group element.

*Step 2.* Decide which of the roots is a power of  $g^{r ab}$  and continue with this element.

Assume for some  $k = 2f_i - 1, 2f_i - 2, \dots, f_i$  that we have already computed

$$g^{p_1^{f_1} \dots p_{i-1}^{f_{i-1}} p_i^{k+1} p_{i+1}^{2f_{i+1}} \dots p_s^{2f_s} ab} = g^{c' p_i^{k+1} ab} ,$$

where  $c'$  is explicitly known. Then the two steps work as follows.

*Step 1.* According to Lemma 3 we can compute a  $p_i$ -th root of the group element in time  $O((\log |G|)^2 + p_i \log |G|)$ .

*Step 2.* Because  $a$  and  $b$  can be obtained modulo  $p_i^{e_i - f_i}$  directly from  $g^{ra}$  and  $g^{rb}$  by the Pohlig-Hellman algorithm and  $c'$  is explicitly known, and because  $k \geq f_i$ , we can compute  $c'p_i^k ab$  modulo  $p_i^{e_i}$ . From the root obtained in Step 1, all the roots

$$g^{c'p_i^k ab + j \cdot |G|/p_i} \quad (j = 0, \dots, p_i - 1)$$

can be computed. We have  $j \cdot |G|/p_i \equiv 0 \pmod{p_i^{e_i}}$  only for  $j = 0$ , and the correct group element  $g^{c'p_i^k ab}$  can be determined by computing the discrete logarithms of the candidates modulo  $p_i^{e_i}$ , using the Pohlig-Hellman algorithm.

The entire procedure, executed for all prime factors  $p_i$  of  $r$ , ends up with  $g^{rab}$ , and the running time of the algorithm is polynomial in  $\log |G|$ .  $\square$

*Remark.* It has been pointed out in a preliminary version of [4] that in case of a generator change, i.e., if  $(r, |G|) = 1$ , it is not even necessary to know  $r$ . Let  $h = g^r$ , and let  $\text{DH}_g$  and  $\text{DH}_h$  be the DH functions in  $G$  with respect to the generator  $g$  and  $h$ , respectively. Then

$$\begin{aligned} \text{DH}_h(h^a, h^b) &= h^{ab} = g^{rab} = \text{DH}_g(g^{r^2 ab}, g^{r^{-1}}) \\ &= \text{DH}_g(\text{DH}_g(h^a, h^b), \text{PDH}_{g, \varphi(|G|)-1}(h)) , \end{aligned}$$

and the last expression can be computed by  $O(\log |G|)$  applications of the oracle with respect to the basis  $g$ .

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.

**Theorem 11** *Let  $P$  be a fixed polynomial. Let  $G$  be a cyclic group with generator  $g$  and order  $|G| = \prod_{i=1}^t p_i^{e_i}$ , and let  $B := P(\log |G|)$  be a smoothness bound. If for all  $p_i > B$  a number  $s_i$ , where  $p_i$  does not divide  $s_i$ , and a DH oracle for the group  $\langle g^{s_i} \rangle$  is given, then there exists a polynomial-time algorithm solving the DH problem in  $G$  with respect to  $g$  which calls each oracle for such a subgroup once.*

*Proof.* Let  $g^u$  and  $g^v$  be given. We compute  $g^{uv}$  using the available oracles for subgroups. Let  $m_i := p_i^{e_i}$ ,  $M_i := |G|/m_i$ , and  $N_i := M_i^{-1} \pmod{m_i}$ .

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 algorithm. For a prime factor  $p_i > B$ , assume that a DH oracle for the subgroup  $\langle g^{s_i} \rangle$  is given, where  $p_i$  does not divide  $s_i$ . We apply the oracle for  $\langle g^{s_i} \rangle$  to  $(g^{s_i})^u = (g^u)^{s_i}$  and  $(g^{s_i})^v$  to obtain  $(g^{s_i})^{u \cdot v}$ , where  $u$ ,  $v$  and  $u \cdot v$  are modulo  $|G|/s_i$ . Let  $z_i := s_i^{-1} \pmod{m_i}$  and

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

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  $G$ :

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

□

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

**Corollary 12** *Consider a group  $G = \langle g \rangle$  and a subgroup  $H = \langle g^k \rangle$  of  $G$  with  $(\log |G|)^{O(1)}$ -smooth index. The DH problem for  $H$  is polynomial-time equivalent to the DH problem for  $G$ .*

## 6 Concluding remarks

We have presented a technique for reducing the DL problem in a group  $G$  to the DH problem in the same group efficiently when suitable auxiliary groups are given. One conclusion of this fact is that, under a plausible but unproven assumption on the existence of smooth numbers, for every group whose order does not contain a multiple large prime factor there exists a polynomial-time algorithm computing discrete logarithms and making calls to a DH oracle for the same group. In the generic model, it was proven that such a reduction cannot exist for groups whose order is divisible by the square of a large prime. A second conclusion is that solving the DH and DL problems is computationally equivalent for many classes of groups in a uniform sense. These are the groups for which suitable auxiliary groups can be efficiently constructed.

Throughout this paper, we have assumed to know the group order and its factorization. Let  $p$  be a large prime factor of  $|G|$ . If an appropriate auxiliary group over  $GF(p)$  such as a subgroup of the multiplicative group of a finite field or an elliptic curve is given that has smooth order, then  $p$  can be found efficiently as a factor of  $|G|$  (see [23] and [2]). This fact indicates

a close relationship between the problems of integer factoring and proving the equivalence between the DH and DL problems.

In Appendix C we describe a technique, presented in [48] and independently considered in [4], for obtaining stronger results under the assumption that efficient algorithms exist for solving the DH problem in certain groups, and which use only algebraic operations. The idea is to execute these algorithms on implicitly represented arguments. This allows to iterate the technique by computing with multiply implicitly represented elements. It is then no longer 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  $GF(q)$  are sufficient under the assumption that a polynomial-time DH algorithm exists for  $H_p$  which uses only algebraic operations in  $GF(p)$ .

## Acknowledgment

The authors thank Dan Boneh, Dima Grigoriev, Hendrik Lenstra, Markus Metzger, Victor Shoup, and Igor Shparlinsky for interesting discussions on the subject of this paper, and two anonymous referees for their very helpful comments for improving the presentation.

## 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] D. Boneh, *Studies in computational number theory with applications to cryptography*, Ph. D. Thesis, Princeton Univ., Nov. 1996.
- [4] D. Boneh and R. J. Lipton, Algorithms for black-box fields and their application to cryptography, *Advances in Cryptology - CRYPTO '96*, Lecture Notes in Computer Science, Vol. 1109, pp. 283–297, Springer-Verlag, 1996.
- [5] S. Brands, *An efficient off-line electronic cash system based on the representation problem*, Tech. Rep. CS-R9323, CWI, Amsterdam, 1993.
- [6] 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.

- [7] 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.
- [8] E. R. Canfield, P. Erdős, and C. Pomerance, On a problem of Oppenheim concerning “Factorisatio Numerorum”, *J. Number Theory*, Vol. 17, pp. 1–28, 1983.
- [9] D. G. Cantor, Computing in the Jacobian of a hyperelliptic curve, *Math. Comp.*, Vol. 48, No. 177, pp. 95–101, 1987.
- [10] M. A. Cherepnev, On the connection between discrete logarithms and the Diffie-Hellman problem, *Discrete Math. Appl.*, 1996.
- [11] D. Coppersmith and I. E. Shparlinsky, *On polynomial approximation and the parallel complexity of the discrete logarithm problem and breaking the Diffie-Hellman cryptosystem*, preprint, Nov. 1996.
- [12] B. den Boer, Diffie-Hellman is as strong as discrete log for certain primes, *Advances in Cryptology - CRYPTO '88*, Lecture Notes in Computer Science, Vol. 403, pp. 530–539, Springer-Verlag, 1989.
- [13] W. Diffie and M. E. Hellman, New directions in cryptography, *IEEE Transactions on Information Theory*, Vol. 22, No. 6, pp. 644–654, 1976.
- [14] 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.
- [15] W. Feller, *An introduction to probability theory and its applications*, John Wiley & Sons, 1968.
- [16] K. O. Geddes, S. R. Czapor, and G. Labhan, *Algorithms for computer algebra*, Kluwer Academic Publisher, 1992.
- [17] 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.
- [18] G. H. Hardy and E. M. Wright, *An introduction to the theory of numbers*, University Press, Oxford, 1979.
- [19] K. Ireland and M. Rosen, *A classical introduction to modern number theory*, Springer-Verlag, 1982.
- [20] N. Koblitz, Hyperelliptic cryptosystems, *Journal of Cryptology*, Vol. 1, pp. 139–150, 1989.
- [21] N. Koblitz, Elliptic curve cryptosystems, *Math. Comp.*, Vol. 48, pp. 203–209, 1987.

- [22] G.-J. Lay and H. G. Zimmer, Constructing elliptic curves with given group order over large finite fields, *Proc. of ANTS-I*, Lecture Notes in Computer Science, Vol. 877, pp. 250–263, Springer-Verlag, 1994.
- [23] H. W. Lenstra, Jr., Factoring integers with elliptic curves, *Annals of Mathematics*, Vol. 126, pp. 649–673, 1987.
- [24] R. Lidl and H. Niederreiter, *Introduction to finite fields and their application*, Cambridge University Press, 1986.
- [25] J. L. Massey, Advanced Technology Seminars Short Course Notes, pp. 6.66–6.68, Zürich, 1993.
- [26] U. M. Maurer, Towards the equivalence of breaking the Diffie-Hellman protocol and computing discrete logarithms, *Advances in Cryptology - CRYPTO '94*, Lecture Notes in Computer Science, Vol. 839, pp. 271–281, Springer-Verlag, 1994.
- [27] U. M. Maurer and S. Wolf, The Diffie-Hellman protocol, to appear in *Designs, Codes, and Cryptography*, Special Issue “20 Years of Public-Key Cryptography”, 1998.
- [28] U. M. Maurer and S. Wolf, Lower bounds on generic algorithms in groups, preprint, 1997.
- [29] U. M. Maurer and S. Wolf, On the complexity of breaking the Diffie-Hellman protocol, Tech. Rep. 244, Computer Science Department, ETH Zürich, April 1996.
- [30] U. M. Maurer and S. Wolf, Diffie-Hellman oracles, *Advances in Cryptology - CRYPTO '96*, Lecture Notes in Computer Science, Vol. 1109, pp. 268–282, Springer-Verlag, 1996.
- [31] U. M. Maurer and Y. Yacobi, Non-interactive public-key cryptography, *Designs, Codes, and Cryptography*, Vol. 9, pp. 305–316, 1996.
- [32] K. S. McCurley, A key distribution system equivalent to factoring, *Journal of Cryptology*, Vol. 1, No. 2, pp. 95–105, 1988.
- [33] 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.
- [34] A. J. Menezes (Ed.), *Applications of finite fields*, Kluwer Academic Publishers, 1992.
- [35] A. J. Menezes, *Elliptic curve public key cryptosystems*, Kluwer Academic Publishers, 1993.

- [36] V. Miller, Uses of elliptic curves in cryptography, *Advances in Cryptology - CRYPTO '85*, Lecture Notes in Computer Science, Vol. 218, pp. 417–426, Springer-Verlag, 1986.
- [37] 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.
- [38] J. M. Pollard, Monte-Carlo methods for index computation mod  $p$ , *Math. Comp.*, Vol. 32, pp. 918–924, 1978.
- [39] J. M. Pollard, Theorems on factorization and primality testing, *Proceedings of the Cambridge Philosophical Society*, Vol. 76, pp. 521–528, 1974.
- [40] 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.
- [41] H. Rück, A note on elliptic curves over finite fields, *Math. Comp.*, Vol. 49, pp. 301–304, 1987.
- [42] K. Sakurai and H. Shizuya, Relationships among the computational powers of breaking discrete log cryptosystems, *Advances in Cryptology - EUROCRYPT '95*, Lecture Notes in Computer Science, Vol. 921, pp. 341–355, Springer-Verlag, 1995.
- [43] 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, Springer-Verlag, 1990.
- [44] 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.
- [45] V. Shoup, Lower bounds for discrete logarithms and related problems, *Advances in Cryptology - EUROCRYPT '97*, Lecture Notes in Computer Science, Vol. 1233, pp. 256–266, Springer-Verlag, 1997.
- [46] I. E. Shparlinsky, *Computational problems in finite fields*, Kluwer Academic Publishers, 1992.
- [47] S. A. Vanstone and R. J. Zuccherato, Elliptic curve cryptosystems using curves of smooth order over the ring  $\mathbf{Z}_n$ , *IEEE Transactions on Information Theory*, 1997.
- [48] S. Wolf, *Diffie-Hellman and discrete logarithms*, Diploma Thesis, Department of Computer Science, ETH Zürich, March 1995.

## Appendix A: Finding generator sets of the auxiliary groups

We show how a generator set can be found efficiently in a (additively written) finite abelian group  $H$  of rank  $r$  and with  $B$ -smooth order  $|H| = \prod_{i=1}^l q_i^{f_i}$ . Suppose  $H \cong \mathbf{Z}_{n_1} \times \cdots \times \mathbf{Z}_{n_r}$  (such that  $n_{j+1}$  divides  $n_j$  for  $j = 1, \dots, r-1$ ), where the numbers  $r$  and  $n_1, \dots, n_r$  are not known a priori, and suppose that we have already found  $n_i$  and points  $h_i$  with order  $n_i$  in  $H/\langle h_1, \dots, h_{i-1} \rangle$  for  $i = 1, \dots, j-1$ . Let  $n_1 = \prod q_i^{g_i}$ , and let  $\pi_{j-1}$  denote the canonical projection to the quotient group  $H/\langle h_1, \dots, h_{j-1} \rangle$ , i.e.,  $\pi_{j-1}(u)$  is the element of  $H/\langle h_1, \dots, h_{j-1} \rangle$  containing  $u$ . For the construction of  $h_j$  such that  $\pi_{j-1}(h_j)$  has maximal order in  $H/\langle h_1, \dots, h_{j-1} \rangle$ , we choose  $O(\log \log |H|)$  points  $h$  in  $H$  at random and compute  $\text{ord}_{H/\langle h_1, \dots, h_{j-1} \rangle} \pi_{j-1}(h)$  by comparing

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

with the unity  $e_{H/\langle h_1, \dots, h_{j-1} \rangle}$  of the quotient group. Comparing  $\pi_{j-1}(h')$  and  $e_{H/\langle h_1, \dots, h_{j-1} \rangle}$  is equivalent to deciding if  $h' \in \langle h_1, \dots, h_{j-1} \rangle$ , which is done by the generalized Pohlig-Hellman DL algorithm described in Section 3. This leads to an element  $h_j$  with maximal order in  $H/\langle h_1, \dots, h_{j-1} \rangle$ .

It is possible that the algorithm makes a mistake here, i.e., that the generated element does not have maximal order. Such an error occurs only with probability exponentially small in the number of trials and can be detected as follows. In the case where  $\text{ord}_{H/\langle h_1, \dots, h_{j-1} \rangle} \pi_{j-1}(h_j)$  does not divide  $\text{ord}_{H/\langle h_1, \dots, h_{j-2} \rangle} \pi_{j-2}(h_{j-1})$ , the process must be restarted because 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 algorithm stops if  $\langle h_1, \dots, h_r \rangle = H$ , that is

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

and  $\{h_1, \dots, h_r\}$  is then a generator set of  $H$ . Every element  $c$  of  $H$  has a unique representation  $c = \sum_{j=1}^r k_j h_j$  with  $k_j \in \{0, \dots, n_j - 1\}$  with respect to this set. 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 trade-off in the Pohlig-Hellman algorithm).

## Appendix B: Gröbner basis computations and the completion of the proof of Theorem 6

The goal of this appendix is to complete the proof of Theorem 6, i.e., to show that the second condition also implies that  $H$  is defined strongly algebraically over  $GF(p)$ . In the first part of the proof of Theorem 6 (given in Section 4.2), the key argument was that the Cantor-Zassenhaus algorithm allows to solve a univariate polynomial equation over a finite field efficiently and with algebraic operations only. This led to an EMBED algorithm with the required properties.

For the second part of this proof the result is required that a *system* of such equations can be solved. In Section B.1 we shortly describe the concept of Gröbner bases, which are a tool for solving such systems. Section B.2 completes the proof of Theorem 6.

### B.1 Gröbner bases

Let  $\mathbf{F}$  be a field and  $R$  be the ring  $\mathbf{F}[x_1, \dots, x_n]$  of the polynomials in  $x_1, \dots, x_n$  over  $\mathbf{F}$ . Let further  $p_i = 0$  (where  $i = 1, \dots, l$  and  $p_i \in R$  for all  $i$ ) be a system of polynomial equations. We also write  $P = 0$ , where  $P := \{p_i\}$ . Every basis of the generated ideal  $\langle P \rangle$  in the ring  $R$  leads to an equivalent system of equations. Gröbner bases with respect to the lexicographic term ordering have the property that the system can be solved if univariate equations can be solved. The lexicographic term ordering is defined as follows:

$$\prod x_j^{i_j} <_L \prod x_j^{i'_j}$$

if and only if  $i_j = i'_j$  for  $j = 1, \dots, l-1$  and  $i_l < i'_l$  for some  $l$ .

We motivate the definition of Gröbner bases<sup>3</sup>. Let  $f$  and  $g$  be polynomials, and let  $t$  be the leading term of  $g$ . One can reduce  $f$  modulo  $g$  if a monomial of  $f$  is a multiple of  $t$ ,  $f = \alpha t + r$ . The reduction of  $f$  modulo  $g$  is then

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

where  $M(g)$  denotes the leading monomial of  $g$ . Let  $Q$  be a set of polynomials. The *reducer set* of the polynomial  $f$  with respect to  $Q$  are the polynomials  $g$  in  $Q$  with the property 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

---

<sup>3</sup>For an introduction to Gröbner bases, see for example [16].

$R$  is not a principal ideal domain (if  $n > 1$ ), the maximal reduction is not unique, and an element  $q$  of  $\langle Q \rangle$  can be irreducible modulo  $Q$ . A Gröbner basis  $G$  (with respect to a term ordering) is defined and characterized by the following equivalent conditions:

1. Maximal reductions modulo  $G$  are unique.
2. If  $f \in \langle G \rangle$ , then  $f$  reduces to 0 modulo  $G$ .
3. For all  $f$  and  $g$  in  $G$ ,

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

reduces to 0 modulo  $G$ .

For given  $P$ , the third criterion leads to a simple algorithm 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  $\text{s-poly}(f_1, f_2)$ . If it is different from zero, extend  $P$  by this polynomial. Repeat the process for all pairs, including the pairs with components added to  $P$  during the execution of the algorithm.*

This algorithm can be improved by criteria stating whether  $\text{s-poly}(f, g)$  reduces to 0, such that the number of s-polynomial reductions is decreased. The complexity of Gröbner basis computations is a subject of ongoing 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 of order  $O(D')$ , where

$$D' := (nD)^{(n+1)2^{s+1}}, \quad (14)$$

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

The following are key properties of Gröbner bases. Let  $P$  be a set of polynomials and  $G$  a monic Gröbner basis for  $\langle P \rangle$  (where monic means that the coefficients of the leading monomials of all the polynomials are 1).

*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 occurring in  $G$ . Then the following statements are equivalent:

1.  $P$  has finitely many solutions over  $\mathbf{C}$ ,
2. 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 property implies, when using the lexicographic ordering, that a subset of the equations coming from the polynomials of the Gröbner basis is a system of triangular form and can be solved if univariate polynomial equations can be solved. The fact that  $(x_i)^{m_i}$  is the leading term of a polynomial implies that the variables  $x_1, \dots, x_{i-1}$  do *not* occur in the polynomial. For example the polynomial with  $(x_n)^{m_n}$  as leading term is *univariate* (with the only variable  $x_n$ ). Analogously, there is a polynomial containing  $x_{n-1}$  and  $x_n$  only, etc.

## B.2 Completing the proof of Theorem 6

*Proof that Condition 2 is sufficient.* Let  $|H| = f(p)$  for some non-constant polynomial (with integer coefficients)  $f(x)$  dividing  $x^N - 1$ , where  $N = O(1)$ .

We show first that we can assume without loss of generality that  $f(x)$  equals a cyclotomic polynomial  $\Phi_n(x)$  for some  $n = O(1)$ . The cyclotomic polynomials are the irreducible factors of the polynomials  $x^N - 1$  over the ring  $\mathbf{Z}$  of integers (see for example [24]). More precisely, we have

$$x^N - 1 = \prod_{d|N} \Phi_d(x) ,$$

and the polynomials  $\Phi_d$  are irreducible over  $\mathbf{Z}$ . The degree of  $\Phi_n(x)$  is  $\varphi(N)$ , where  $\varphi$  is Euler's totient function. Because the cyclotomic polynomials are irreducible over  $\mathbf{Z}$ , the (non-constant) polynomial  $f(x)$  (that divides  $x^N - 1$ ) must be a multiple of at least one cyclotomic polynomial  $\Phi_n(x)$ .

We show that a subgroup  $H$  of  $GF(p^n)^*$  with  $|H| = \Phi_n(p)$  (for  $n = O(1)$ ) is defined strongly  $(n, \alpha)$ -algebraically over  $GF(p)$  for some  $\alpha = (\log p)^{O(1)}$ . This proves the second part of Theorem 6, because a group which has a subgroup with this property has the property itself (the same EMBED algorithm can be used). Let

$$\Phi_n(x) = \sum_{j=0}^{\varphi(n)-1} c_j x^j$$

(with  $c_j \in \mathbf{Z}$ ). Let further  $\alpha_0, \dots, \alpha_{n-1}$  be a normal basis of  $GF(p^n)$  over  $GF(p)$ .

We describe the EMBED algorithm for  $H$ . Let  $x, e \in GF(p)$  be given. We compute, by a polynomial number of algebraic operations in  $GF(p)$ , an element  $\beta = (\beta_0, \dots, \beta_{n-1})$  such that  $x + e$  is one of the coordinates of  $\beta$ , for instance  $x + e = \beta_0$ . Again, we need an alternative characterization of the fact that  $\beta \in H$  in terms of the  $GF(p)$ -coordinates of  $\beta$ . The following conditions are equivalent for  $\beta = \sum \beta_i \alpha_i$ .

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

In the fourth step, we have made use of  $\beta_i^{p^j} = \beta_i$  (because  $\beta_i \in GF(p)$ ) and  $\alpha_i^{p^j} = \alpha_{i+j}$  (by the definition of the normal basis). The last condition corresponds to a system of  $n$  polynomial equations (with  $GF(p)$ -coefficients) in the  $\beta_i$ , where the maximal degree  $D$  of the polynomials is bounded by

$$D \leq \max \left\{ \sum_{c_j > 0} c_j, \sum_{c_j < 0} |c_j| \right\} \leq \varphi(n) \cdot \max_j |c_j| .$$

As in the first part of the proof, the EMBED algorithm assigns  $x + e$  to one of the  $\beta_i$ 's, random values to some of the other  $\beta_i$ 's, and solves the arising equations over  $GF(p)$  for the remaining  $\beta_i$ 's. Because  $|H|/p^n \approx 1/p^{n-\varphi(n)}$ , i.e., approximately every  $p^{n-\varphi(n)}$ -th element  $\beta$  of  $GF(p^n)$  is also an element of  $H$ , we have to solve the equations for  $n - \varphi(n)$  different coordinates  $\beta_i$  simultaneously in order to have an expectation of one solution. (If no solution is found, the algorithm reports failure.)

Using Gröbner bases, this system of polynomial equations can now be transformed into an equivalent system of triangular form (see Section B.1). The computation of the Gröbner basis uses only algebraic operations in

$GF(p)$ , and its complexity is of order  $O(D^{n^2})$  (see [16]). The triangular system of equations can be solved by the Cantor-Zassenhaus algorithm for solving *univariate* polynomial equations (see in the first part of the proof of Theorem 6).

According to the result of Gianni and Kalkbrener (see [16]), it suffices to solve a subset of  $n - \varphi(n)$  equations. The first polynomial has to be solved once, the second one  $D'$  times (where  $D'$  is defined as in (14); the reason is that in the worst case, the first polynomial has  $D'$  different solutions), the third one  $(D')^2$  times, etc. This yields  $O((D')^n)$  executions of the Cantor-Zassenhaus algorithm. (The *effective* number of executions will be much smaller in a typical case, since only about one solution is expected.)

The expected complexity of the required executions of the EMBED algorithm is polynomial in  $\log p$  (and the algorithm uses only algebraic operations in  $GF(p)$ ) if  $n = O(1)$ .  $\square$

## Appendix C: Algebraic algorithms solving the DH problem

The results described in this appendix are based on the following observation. Assume that not only a DH oracle for a group  $G$ , but also an efficient *algorithm* which solves the DH problem in an entire class of groups, such as elliptic curves over a finite field or the groups  $GF(p)^*$ , is given. If this algorithm additionally has the property that it uses only algebraic operations in the underlying field, then it can be executed on inputs that are not explicitly known, but only implicitly represented (in the sense of Section 3). This allows to iterate the reduction algorithm described in Section 3, i.e., computing discrete logarithms in  $G$  is reduced to the same problem in a group  $GF(p)^*$ , which is further reduced to the DL problem of another group  $GF(q)^*$ , and so on.

We give an example. Assume that polynomial-time algorithms exist for solving the DH problem in all the groups  $GF(p)^*$ , and that these algorithms use algebraic operations in  $GF(p)$  only. Let again  $B = (\log |G|)^{O(1)}$  be a smoothness-bound,  $p_0$  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  $i = 2, \dots, k$ , and that  $p_k - 1 =: T = \prod_i r_i^{n_i}$  is  $B$ -smooth, and  $k = O(1)$ . Given  $a = g^s$ , it is possible to compute  $x_0 \equiv s \pmod{p_0}$ ,  $x_0 \in GF(p_0)$ , in polynomial time as

follows when given a DH oracle for  $G$ . Let

$$h_0 := \frac{|G|}{p_0}, \quad h_i := \frac{p_{i-1} - 1}{p_i} \quad (\text{for } i = 1, \dots, k),$$

and let

$$GF(p_i)^* = \langle c_i \rangle \quad (\text{for } i = 0, \dots, k-1).$$

If  $x_0 \neq 0$ , then  $x_0 = c_0^{w_0}$  (in  $GF(p_0)$ ), and  $g^s$  is 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. But  $x_1 \equiv w_0 \pmod{p_1}$  (with  $x_1 \in GF(p_1)$ ) can be written as  $x_1 = c_1^{w_1}$  (in  $GF(p_1)$ , 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  $GF(p_1)$  which are “double-implicitly” represented. For example, an addition of two  $GF(p_1)$ -elements requires multiplication of the corresponding implicitly represented  $GF(p_0)^*$ -elements and can be obtained by a call to the DH oracle for  $G$ . A multiplication in  $GF(p_1)^*$  is done by an oracle call for  $GF(p_0)^*$  with implicitly represented arguments and an implicitly represented answer. This works (in polynomial time) because of the stated properties of the DH algorithm for  $GF(p_0)^*$ .

Analogously, computation with  $(k+1)$ -times implicitly represented arguments is possible in the smooth group  $GF(p_k)^*$ . The index-search problem for the list

$$\left( g^{h_0 c_0^{h_1 c_1 \dots^{h_k c_k^{\frac{T}{r_i} t}}} \right)_{t=0, \dots, r_i-1}$$

and the element

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

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  $GF(p_k)$ ), 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  $x_0$ .

*Remark.* The reason for assuming that  $p_{i-1} - 1$  has *only one* large prime factor  $p_i$  is that otherwise it would not be possible to find the factors of  $p_{i-1} - 1$  in polynomial time. When these factors are given, then the condition is unnecessary.

**Theorem 13** *Let  $P$  be a fixed polynomial. Let  $G$  be a cyclic group with the property that all prime factors  $p_0$  of  $|G|$  greater than  $B := P(\log |G|)$  are single, and that for all such prime factors there exist  $k = O(1)$  and primes  $p_i$  ( $i = 1, \dots, k$ ) such that  $p_i$  is the only prime factor of  $p_{i-1} - 1$  greater than  $B$  for  $i = 1, \dots, k$ , and  $p_k - 1$  is  $B$ -smooth. Assume further that a polynomial-time algorithm is given which solves the DH problem in the groups  $GF(p)^*$  and uses algebraic operations in  $GF(p)$  only. Then, breaking the DH protocol and computing discrete logarithms are polynomial-time equivalent in  $G$ .  $\square$*

The process works in an analogous way if some of the used groups are cyclic elliptic curves or Jacobians, provided an efficient algebraic (with respect to the underlying field  $GF(p)$ ) DH algorithm is given for these groups.

## Appendix D: Construction of groups for which a reduction of the DL problem to the DH problem is efficiently constructible

It appears desirable to use a group  $G$  in the DH protocol for which the algorithm reducing the DL problem to the DH problem can easily be found. 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:

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.

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 DH protocol, there is no reason for the designer of the group to make it public.

Constructing a group  $G$  of the third type is not difficult: 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 group  $G$  for the DH protocol whose order is a multiple of  $n$ . One possibility is to find a multiple  $l$  of  $n$  (where  $l/n$  is small) such that  $l + 1$  is prime and to use  $G = GF(l + 1)^*$ . An alternative is to use the construction of Lay and Zimmer [22] for finding an elliptic curve of order  $n$ .

The second case is somewhat more involved. Primes  $p$  for which the designer knows an auxiliary group over  $GF(p)$  can be obtained by choosing a large smooth number  $m$  and using the method of Lay and Zimmer [22] for constructing a prime  $p$  together with an elliptic curve of order  $m$ . When given such prime factors of the group order, a group  $G$  can be found as described.

We now consider efficient constructions for the first case. We generalize an algorithm, presented in [47] 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  $GF(p)$  (see (4)), is smooth.

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

$$u + vi := (x_1 + y_1i)(x_2 + y_2i) ,$$

that is

$$u = x_1x_2 - y_1y_2, \quad v = x_1y_2 + x_2y_1 .$$

Then  $u$  and  $v$  have at most  $2l'$  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

$$a + ki = (c + di)(u + vi) = (c + di)(x_1 + y_1i)(x_2 + y_2i)$$

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 (1)), and smoothness can be tested with the elliptic curve factoring algorithm [23]. 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

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

is prime. According to (4), a quarter of the curves  $y^2 = x^3 - Dx$  over  $GF(p)$  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 constructible over  $GF(p)$ . The expected number of trials is

$$O\left(\left(\frac{4l' + 2l}{s}\right)^{\frac{4l'+2l}{s}} \cdot \left(\frac{2l'}{s}\right)^{\frac{4l'}{s}} \cdot (8l' + 2l)\right). \quad (15)$$

In a similar way, primes can be constructed such that curves of type  $y^2 = x^3 + D$  have smooth order. More precisely, one can generate primes  $p = a^2 - a(k \pm 1) + (k \pm 1)^2$  such that  $a^2 - ak + k^2$  is one of the possible orders of the curves  $y^2 = x^3 + D$  over  $GF(p)$  (see (5)) and  $s$ -digit-smooth.

In case of a small  $k$ , an  $L$ -digit prime  $p$  such that an  $s$ -digit-smooth curve is constructible over  $GF(p)$  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  $GF(p)$  is constructible can be found by approximately  $3 \cdot 10^6$  trials (instead of about  $10^{11}$  trials when using the straightforward strategy).