The Diffie-Hellman Protocol[*]

Ueli M. Maurer   Stefan Wolf

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

July 5, 1999

Abstract

The 1976 seminal paper of Diffie and Hellman is a landmark in the history of cryptography. They introduced the fundamental concepts of a trapdoor one-way function, a public-key cryptosystem, and a digital signature scheme. Moreover, they presented a protocol, the so-called Diffie-Hellman protocol, allowing two parties who share no secret information initially, to generate a mutual secret key. This paper summarizes the present knowledge on the security of this protocol.

1 Introduction

In 1976, Whitfield Diffie and Martin Hellman published their celebrated paper [16] which initiated a revolution in cryptography. Diffie and Hellman can be seen as the founders of modern cryptography. The theoretical concepts of a public-key cryptosystem and a digital signature have been realized only two years after Diffie and Hellman's paper by Rivest, Shamir, and Adleman in the RSA-system [50]. However, Diffie and Hellman presented the first protocol with public-key properties, the so-called Diffie-Hellman (DH) protocol for public key distribution. An earlier protocol due to Merkle, called *Merkle's puzzles*, achieved the same goals, but the DH protocol has the better ratio between security and efficiency. The security of the DH protocol is

[*] This work was supported by the Swiss National Science Foundation (SNF), grant No. 20–42105.94.

based on the hardness of a certain computational problem (see Section 2.1). The protocol allows two parties Alice and Bob, who are connected by an authenticated but otherwise insecure channel, to generate a secret key which is (believed to be) difficult to compute for an adversary Eve overhearing the communication between Alice and Bob.

The protocol works as follows. Let G be a finite cyclic group with order $|G|$ generated by g. 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 $[0, |G|-1]$.[1] Then they compute secretly $a_A = g^{s_A}$ and $a_B = g^{s_B}$, respectively, and exchange these group elements over the insecure public channel. Finally, Alice and Bob compute $a_{AB} = a_B^{s_A} = g^{s_A s_B}$ and $a_{BA} = a_A^{s_B} = g^{s_B s_A}$, respectively. Note that $a_{AB} = a_{BA}$, and hence this quantity can be used as a secret key shared by Alice and Bob. More precisely, they can apply a function mapping elements of G to the key space of a cryptosystem. For instance, they can use an appropriate block (e.g., the least significant bits of a_{AB}) as the secret key of a conventional block cipher. Figure 1 shows a mechanical analog of the Diffie-Hellman protocol.

Waldvogel and Massey [58] have studied the distribution of the resulting secret key under the assumption that the secret values s_A and s_B are chosen independently and uniformly in $[0, |G|-1]$. They showed that if the group order contains at least one large prime factor (the Diffie-Hellman protocol is insecure otherwise anyway), the distribution of the group element a_{AB} is close to uniform over G. Moreover, when s_A and s_B are chosen uniformly in $\mathbf{Z}_{|G|}^*$ (instead of $\mathbf{Z}_{|G|}$), then the resulting key is perfectly uniformly distributed in the set $\{g^c \ : \ c \in \mathbf{Z}_{|G|}^*\}$.

Specific groups that have been proposed for application in the DH protocol are the multiplicative groups of large finite fields (prime fields [16] or extension fields), the multiplicative group of residues modulo a composite number [37], [38], elliptic curves over finite fields [43], [24], the Jacobian of a hyperelliptic curve over a finite field [23], and the class group of imaginary quadratic fields [9].

This paper is organized as follows. In Section 2, some computational problems related to the DH protocol are discussed such as the Diffie-Hellman problem, the Diffie-Hellman decision problem, and the discrete logarithm problem. Section 3 is concerned with the relationship between the security

[1] It is not necessary that Alice and Bob know the order of the group G. If $|G|$ is not known, Alice and Bob choose their integers from a sufficiently large interval. Moreover, if no generator of G is known, one can find such a $g \in G$ by trial and error if the group order $|G|$ and its factorization $|G| = \prod q_i^{f_i}$ are known. However, it is sufficient if g is an element of G of high order.

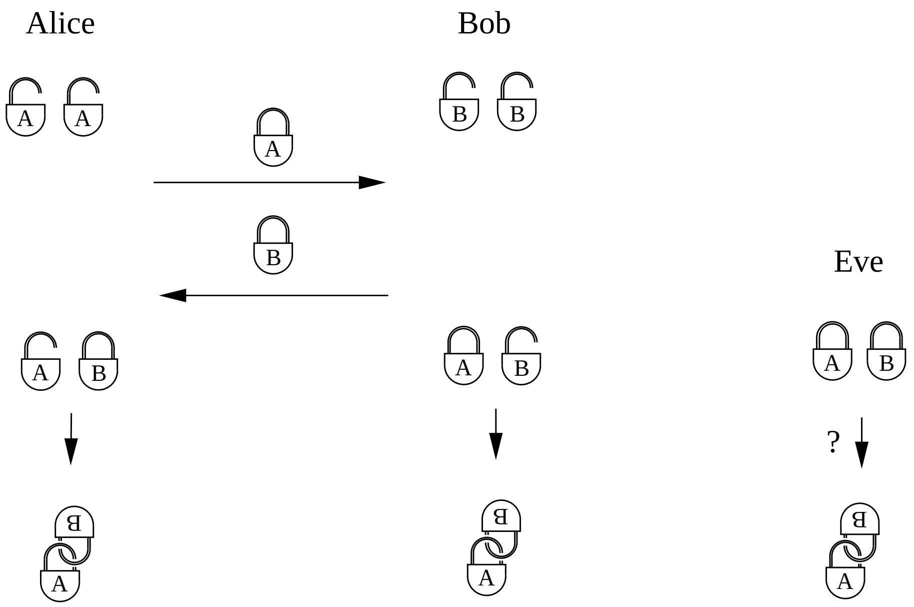

Figure 1: A mechanical analog of the Diffie-Hellman protocol. The padlocks have no keys. They are easy to lock, but hard to open. The two padlocks linked to each other are the secret key. The adversary Eve's task is to link two locked padlocks, which seems to require to open one of the locks first.

of the DH protocol and the hardness of the discrete logarithm problem. A technique is described that allows to reduce the discrete logarithm problem to the Diffie-Hellman problem efficiently for many groups. In Section 4, different definitions of breaking the DH protocol are discussed and compared. It is shown for example that breaking the protocol with a substantial probability is almost as hard as breaking it for all instances. Section 5 describes security proofs for the DH protocol under certain conditions on the knowledge and computational power of the adversary.

2 Computational problems related to the Diffie-Hellman protocol

2.1 The Diffie-Hellman problem

Definition 1 Let G be a finite cyclic group generated by g. The problem of computing $g^{s_A s_B}$ from g^{s_A} and g^{s_B} is called the the *Diffie-Hellman problem*

(*DH problem* for short) with respect to g.

One possibility to solve the DH problem is to compute s_A (or s_B) from g^{s_A} (g^{s_B}) first.

Definition 2 Let G be a finite cyclic group generated by g. The problem of computing from $a \in G$ a number s such that $g^s = a$ is called the *discrete logarithm problem* (*DL problem*) with respect to g.

(For a detailed discussion of the discrete logarithm problem, see [39] or Odlyzko's paper in this issue.) For many groups it is not known whether the most efficient way of solving the DH problem is by solving the DL problem first. It is also unknown whether there exist groups for which the DH problem is substantially easier than the DL problem. This question is addressed in Section 3 of this paper. Finally, it is an open question whether (though widely believed that) there are groups for which the DL problem is difficult. In Section 5.1 some evidence for this claim is given.

2.2 The Diffie-Hellman decision problem

For certain groups for which the DH problem is hard (note again that it is not known whether such groups exist), it is believed that even the problem of verifying the correctness of a solution of the DH problem is hard, i.e., given g^a and g^b, it is computationally infeasible to distinguish g^{ab} from a completely random group element. This implies that no partial information about g^{ab} can be efficiently extracted from g^a and g^b. The Diffie-Hellman decision problem is defined as follows. It was first explicitly formulated in [7].

Definition 3 Let G be a finite cyclic group with generator g. Let g^a, g^b, g^c be chosen independently and randomly in G according to the uniform distribution. Given the triples (g^a, g^b, g^{ab}) and (g^a, g^b, g^c) in random order, the *Diffie-Hellman decision problem* (DHD problem for short) is to decide, with probability substantially greater than 1/2, which of the triples is the correct DH triple (g^a, g^b, g^{ab}).

The DHD problem appears to be easier than the DH problem in general. For instance, consider a group G with order $|G| = 2p$ where p is a prime, and for which the DH problem is hard. When given the random triple (g^a, g^b, g^c) and the DH triple (g^a, g^b, g^{ab}), then with probability 3/4, the correct DH triple can be recognized. The reason is that from g^a, one can determine a

modulo 2 by computing $(g^a)^p$, which is equal to e if $a \equiv 0 \pmod 2$ and to $g^p (\neq e)$ if $a \equiv 1 \pmod 2$. With probability 1/2, $a \cdot b \not\equiv c \pmod 2$, in which case the correct DH triple can be determined. Otherwise, the success probability is 1/2.

Generally, the DH problem can be hard in a group G if the group order $|G|$ contains *at least one* large prime factor, whereas the DHD problem can only be hard if $|G|$ is *free of small prime factors* (see also Section 5.1).

Canetti [10] has described the following generalization of the DHD problem for a group G of prime order. Let f be an *uninvertible* function, i.e., a function for which it is hard to obtain x from $f(x)$ with non-negligible probability. (Note that an uninvertible function is not necessarily one-way, as the example $f(x) \equiv 0$ shows.) Then the generalized version of the DHD problem is to distinguish between the triples $(f(a), g^b, g^{ab})$ and $(f(a), g^b, g^c)$ with probability significantly greater than 1/2. As shown in [10], certain hash functions can be shown to hide all partial information about their input under the assumption that the generalized DHD problem is hard. Although this conjecture appears to be very strong, it has not yet been contradicted.

2.3 The discrete logarithm problem

Let $A = (a_i)_{i=0,\ldots,n-1}$ be a list of elements of some set such that it is easy to compute a_i for a given i. The *index search problem* for A is the problem of computing for a given b an index i such that $b = a_i$. This problem can trivially be solved by exhaustive search, which requires at most n comparisons. If the list A has the property that the permutation $\sigma: a_i \mapsto a_{i+1}$ (where the index is reduced modulo n) can efficiently be computed, then the search can be sped up by a time-memory tradeoff known as the *baby-step giant-step* algorithm. A table of size M is required for storing the sorted list of values $b, \sigma(b), \ldots, \sigma^{M-1}(b)$. The elements $a_0, a_M, a_{2M}, \ldots$ are computed until one of them, a_{iM}, is equal to one of the values $\sigma^j(b)$ in the table. Then the index of b is $iM - j$.

The discrete logarithm problem in a cyclic group H of order $|H|$ with generator h is the index search problem for the list $(h^0 = e, h^1 = h, \ldots, h^{|H|-1})$. Multiplication with h corresponds to the above-mentioned permutation σ. Hence the baby-step giant-step method is applicable for solving the DL problem.

Moreover, the computation of a discrete logarithm in a group H can be reduced to the same problem in the minimal non-trivial subgroups of H, i.e., the subgroups of H with prime order, by the following method which is often attributed to Pohlig and Hellman [47].

Let $a = h^x$. For a fixed prime factor q of $|H|$, consider the group element $a^{|H|/q} = h^{x \cdot |H|/q}$. The algorithm is based on the following two simple observations. Because $(a^{|H|/q})^q = a^{|H|} = e$, the group element $a^{|H|/q}$ can take q possible values, namely the q different q-th roots $h^0, h^{|H|/q}, \ldots, h^{(q-1)|H|/q}$ of the neutral element e. These group elements form a subgroup generated by $h^{|H|/q}$. Secondly, the modulus of x with respect to q determines *which* of these roots equals $a^{|H|/q}$. More precisely,

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

Hence x can be determined modulo q by solving the DL problem in the subgroup $\langle h^{|H|/q} \rangle$. If q is a prime factor of $|H|$ with multiplicity $f > 1$, then the coefficients $x_0, x_1, \ldots, x_{f-1}$ of the q-adic representation $x \equiv x_0 + x_1 q + \cdots + x_{f-1} q^{f-1} \pmod{q^f}$ can be computed as follows. Because $x \equiv x_0 \pmod{q}$, the first coefficient x_0 can be obtained as just described. When x_0 is known, we compute the group element $(a \cdot h^{-x_0})^{|H|/q^2}$. Because

$$(a \cdot h^{-x_0})^{|H|/q^2} = h^{x_1 \cdot |H|/q} \ ,$$

this group element is again equal to one of the q-th roots of the neutral element. The coefficient x_1 can be determined by computing a discrete logarithm in the group $\langle h^{|H|/q} \rangle$.

With this method, x can be computed modulo q^f for all prime factors q of $|H|$, and Chinese remaindering yields x modulo $|H|$, i.e., the discrete logarithm of a. The complexity of this algorithm for a group H with $|H| = \prod q_i^{f_i}$ is $O(\sum f_i(\log|H| + q_i))$. If memory space for storing $\sqrt{q}$ group elements (where q is the largest prime factor of $|H|$) is available, the running time reduces to $O(\sum f_i(\log|H| + \sqrt{q_i}\log q_i))$ when the baby-step giant-step method is applied for computing the discrete logarithms in the subgroups. The method is efficient only if $|H|$ is smooth, i.e., if $q_i \leq B$ for a small *smoothness bound* B. In the worst case we have $q_i \approx B$ for all i, i.e., the number of factors is $O(\log|H|/\log B)$, and the complexity is $O((\log|H|)^2 + B\log|H|/\log B)$ or $O((\log|H|)^2 + \sqrt{B}\log|H|)$ when the baby-step giant-step method is used.

An additional general-purpose discrete logarithm algorithm is Pollard's rho-method [48]. Heuristic arguments suggest that this algorithm has approximately the same running time as the baby-step giant-step method, but this has not been rigorously proved. The advantage of Pollard's rho-method is that it requires virtually no memory space.

Shoup showed [55] that no general-purpose discrete logarithm algorithm can be substantially faster than the combination of the Pohlig-Hellman decomposition and the baby-step giant-step method. For a description of these results see Section 5.1. For *particular* groups such as the multiplicative group of a finite field there exist more efficient algorithms for the computation of discrete logarithms. These so-called index calculus methods have subexponential running time. The index calculus method for the multiplicative group of a prime field for instance is based on the fact that the group elements of $\mathbf{Z}_p^*$ can be interpreted as integers, which can be easily factored when they consist only of small prime factors. For a description of these methods we refer to the survey article on the discrete logarithm problem by McCurley [39] and the references therein, and to Odlyzko's paper in this issue.

For certain groups however the fastest known algorithms for solving the DL problem are the general-purpose algorithms described above. Examples of such groups are non-supersingular elliptic curves and Jacobians of hyperelliptic curves, which were proposed by Miller [43] and Koblitz [24], [23] to be used in discrete-logarithm based cryptosystems such as the Diffie-Hellman protocol. They appear to have the advantage that shorter secret keys can be used for the same security level. Menezes *et. al.* [40] have shown that the DL problem in a *supersingular* elliptic curve over a finite field can be efficiently reduced to the same problem in the multiplicative group of an extension field of small degree.

Van Oorschot and Wiener [45] have studied the risk of choosing short exponents in the DH protocol. They presented a combination of Pollard's lambda-method and the Pohlig-Hellman decomposition.

Pollard's lambda-method [48] allows to find a discrete logarithm that is known to lie in a fixed interval $[A, B]$ of length $w = B - A$ in heuristic expected time $O(\sqrt{w})$ (instead of $O(\sqrt{w} \log w)$ with a simple generalization of the baby-step giant-step method). The idea is to compute two sequences of group elements, one starting with the upper limit B of the interval (the "trail of the tame kangaroo") and the other with the group element y of which the discrete logarithm should be computed (the "trail of the wild kangaroo"). The behavior of both sequences is given by $x_{i+1} = x_i \cdot h^{f(x_i)}$, where f is a "random-like" function taking integer values in a range R of mean m, where $m = \alpha \cdot w^{1/2}$ for some α depending on the tolerated failure probability. The starting point of the trail of the tame kangaroo (the sequence $x_0, x_1, \ldots$) is $x_0 = h^B$, and the group elements $x_0, x_1, \ldots, x_N$ are computed (for some fixed N). The trail of the wild kangaroo (the sequence $x'_0, x'_1, \ldots$) starts at $x'_0 = y$ and stops with x'_M if

$$\sum_{j=0}^{M-1} f(x'_j) > \sum_{i=0}^{N-1} f(x_i) + (B - A) \ ,$$

because the wild kangaroo has passed the tame kangaroo and escaped. Capture is indicated by $x'_m = x_N$ for some m. Then, the discrete logarithm of y is

$$x_N - f(x'_0) - \cdots - f(x'_{m-1}) \ .$$

Let now $y = h^x$, where x is known to be smaller than w for some w, and let $|H| = S \cdot N$, where S is the smooth part of the group order. By the Pohlig-Hellman decomposition, k can be computed such that $k \equiv x \pmod{S}$. Then $x = k + r \cdot S$ for some $r \leq w/S$ or equivalently, $y' = (h')^r$, where $y' := y \cdot h^{-k}$ and $h' := h^S$. The discrete logarithm r of y' with respect to the base h' can now be computed in probabilistic time $O(\sqrt{w/S})$ using Pollard's lambda method. In other words, the information obtained by the smooth part can be used to reduce the running time of the lambda-method by $\sqrt{S}$. The conclusion is that it is dangerous to use short exponents in the group $\mathbf{Z}_p^*$ with randomly chosen p, because $p-1$ is likely to have a substantial smooth part. This can be avoided by using a subgroup of prime order for the Diffie-Hellman protocol, for instance by selecting a prime p such that $(p-1)/2$ is also prime.

3 The relationship between the DH problem and the DL problem

As mentioned already, it is obvious that the DH problem is at most as hard as the DL problem. The strongest possible form of the converse statement would be that no more efficient way exists for solving the DH problem than to solve the DL problem first. In a strict sense, this would mean that given g^u and g^v, it is only possible to obtain g^{uv} when computing u or v first. However, it appears that such a statement can be proved only by giving an efficient algorithm that, when given g^u, g^v, and g^{uv}, computes u or v. Of course such an algorithm can only exist for groups for which it is easy to compute discrete logarithms because this algorithm itself can be used to compute the discrete logarithm of a group element a efficiently when giving as input a, g^s (in random order), and a^s.

A less strict version is that for groups for which the DH problem can be solved efficiently *for all instances* (or at least for a non-negligible fraction) it is possible to compute discrete logarithms efficiently. It was shown that this is true for certain classes of groups. In this section we describe a general technique for proving such equivalence results which was introduced by Maurer [32] as a generalization of an earlier result by den Boer [15], and was further developed by Wolf [60], Boneh and Lipton [5], Maurer and Wolf [36], and Cherepnev [13].

3.1 The Diffie-Hellman oracle

Definition 4 A *Diffie-Hellman oracle* (DH oracle for short) for a group G with respect to a given generator g takes as inputs two elements $a, b \in G$ (where $a = g^u$ and $b = g^v$) and returns (without computational cost) the element g^{uv}.

We will show that under a plausible but unproven number theoretic assumption, for every finite cyclic group whose order is not divided by a multiple large prime factor there exists a polynomial-time algorithm for computing discrete logarithms and that makes calls to a DH oracle for this group.

In Section 4 we consider different types of DH oracles, such as oracles that answer correctly only with a small probability. The reduction of the DL problem to the problem of breaking the DH protocol with small probability for instance leads to stronger equivalence results.

3.2 Computations on implicit representations using a DH oracle, and the black-box field problem

In the following, let the group order $|G|$ and its factorization $|G| = \prod p_i^{e_i}$ be known, let p be a fixed prime factor of this order, and let a DH oracle be given for the group G. Every element y of the field $GF(p)$ can be interpreted as corresponding to an equivalence class of elements of G, namely those whose discrete logarithm is congruent to y modulo p. Every element of this set is a representation of the field element y.

Definition 5 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 $y \in GF(p)$ if $y \equiv y' \pmod{p}$. We write $y \rightsquigarrow a$.

Note that this 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

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

and these implicitly performed operations on elements of $GF(p)$ 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 (PDH_e oracle) that computes an implicit representation of the e-th power of an implicitly represented element. A possible implementation of a PDH_e oracle is to use a "square and multiply" algorithm for obtaining an implicit representation of y^e, denoted by $\mathrm{PDH}_e(a)$, by $O(\log e)$ calls to a normal DH oracle (remember that $y \rightsquigarrow a$). In particular we can compute multiplicative inverses of implicitly represented elements because

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

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

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

or the computation of square roots using the algorithm of Peralta [46] or a faster method due to Massey [31]. Note that algorithms based on exhaustive search (for example to solve the index search problem, in particular the discrete logarithm problem) lead to explicit results even when executed on implicitly represented arguments.

In order to reduce the DL problem to the DH problem in G (with respect to a fixed generator g), we have to find algorithms that compute s from g^s, using the above technique of implicit computing. Because of the Chinese remainder theorem, it is sufficient to compute s modulo the maximal prime powers dividing the group order $|G|$. We first address the problem of computing s modulo a prime factor p of $|G|$.

Boneh and Lipton [5] have formalized this as the *black-box field problem.* Intuitively, a black-box field is a field $GF(p)$ of which the elements are represented by not necessarily unique arbitrary binary strings from which it is a priori difficult to determine the represented field element explicitly. The inverse problem of computing a black-box representation from an explicitly given field element on the other hand is easy in a black-box field.

The *black-box field problem* is to compute, from a black-box representation of an element x, denoted by $[x]$, the element x explicitly by an algorithm that can make use of oracles performing addition, multiplication, and equality tests of field elements in black-box representation. More precisely, these oracles take as inputs $[x]$ and $[y]$ and output $[x+y]$, $[x \cdot y]$, and δ_{xy} (the Kronecker symbol, i.e., $\delta_{xy} = 1$ if $x = y$ and $\delta_{xy} = 0$ otherwise), respectively. Efficient algorithms for solving the black-box field problem support the security of the Diffie-Hellman protocol because they allow to compute discrete logarithms in G modulo large prime factors of $|G|$ when given a DH oracle for G. A polynomial-time solution for instance would prove the computational equivalence of the DH problem and the DL problem for groups whose order is free of multiple large prime factors.

Note that the problem of finding generic algorithms for solving the DL problem is, following the above formalism, the *black-box group problem* for cyclic groups. (The generic DL problem considered in [55] is even a simpler problem because the representation is supposed to be unique. This allows the use of the sorting and searching techniques that are essential for the baby-step giant-step method.) Surprisingly enough, though the black-box *group* problem is provably hard, the additional structure offered by a field allows to solve the black-box *field* problem efficiently in many cases.

3.3 Reducing the computation of discrete logarithms to solving the DH problem

We describe a general method for computing discrete logarithms modulo a fixed large prime factor p of $|G|$ efficiently with a DH oracle for G. Let for simplicity $G = \langle g \rangle$ with $|G| = p$ prime. The basic idea is to reduce the size of the search spaces for the index search problem or more precisely, the size of the subgroups for computing discrete logarithms. Repeated application of exhaustive search in small spaces allows to obtain *explicit* information from *implicit* information efficiently. One such example was described by Boneh and Lipton [5]. From the implicit representation $a = g^x$ of $x \in GF(p)$ one can compute the implicit representation of the Legendre symbol

$$\left(\frac{x}{p}\right) \equiv x^{(p-1)/2} \pmod{p} .$$

Since only two values are possible (if $x \not\equiv 0 \pmod{p}$) the Legendre symbol can also be computed *explicitly* by comparing its implicit representation with g and g^{-1}. Analogously, the entire sequence

$$\left(\frac{x}{p}\right) , \left(\frac{x+1}{p}\right) , \left(\frac{x+2}{p}\right) , \ldots$$

of Legendre symbols can be computed. Unfortunately, no method is known for computing x modulo p efficiently from this explicit information about x.

We describe a different technique due to Maurer [32], which is based on the same idea, but which uses a so-called *auxiliary group*. In order to illustrate the method, let us assume that a cyclic elliptic curve $E = E_{a,b}(p)$ over $GF(p)$ with generator P is given with B-smooth order $|E| = \prod q_i^{f_i}$. We show that this allows to compute discrete logarithms in G in time

$$\sqrt{B} \cdot (\log p)^{O(1)}$$

when given a DH oracle for G.

The elliptic curve $E_{a,b}(p)$ (with parameters a and b in $GF(p)$) is the set

$$\{(x,y) \in (GF(p))^2 : y^2 = x^3 + ax + b\} \cup \{\mathcal{O}\}$$

(the additional point $\mathcal{O}$ is called the "point at infinity"). There exists an operation on the points of the elliptic curve (called addition) which can be expressed in a constant number of algebraic operations in the coordinates and such that $E_{a,b}(p)$ forms an abelian group with neutral element $\mathcal{O}$. We refer to [42] for an introduction to elliptic curves.

We describe how x can be computed from g^x. First, the group element

$$g^{x^3+ax+b}$$

can be computed from g^x by $O(\log p)$ group operations and two calls to the DH oracle for G. If $x^3 + ax + b$ is a quadratic residue mod p (which can be tested efficiently), then a group element g^y can be computed such that $y^2 \equiv x^3 + ax + b \pmod{p}$. (Otherwise, g^x can be replaced by g^{x+d} for a random offset d until the corresponding expression is a quadratic residue.) Hence we have computed

$$(g^x, g^y) = \left(g^x, g^{\sqrt{x^3+ax+b}}\right)$$

with $x, y \in GF(p)$, where $(x, y) := Q$ is a point of the elliptic curve E.

When given (g^{u_1}, g^{v_1}) and (g^{u_2}, g^{v_2}) with the property that $(u_i, v_i) \in E$, then (g^{u_3}, g^{v_3}) can be computed such that $(u_3, v_3) = (u_1, v_1) + (u_2, v_2)$ (in E). This computation requires $O(\log p)$ group operations in G and $O(\log p)$ calls to the DH oracle for G.

Let q be a prime factor of $|E|$. From (g^x, g^y) we compute (g^u, g^v) such that $(u, v) = (|E|/q) \cdot Q$ (in E). From the generator P of E, the points $(u_i, v_i) = i \cdot (|E|/q) \cdot P$ are computed for $i = 0, 1, \ldots, q-1$, and from (u_i, v_i) we obtain the group elements (g^{u_i}, g^{v_i}). For $Q = kP$, we have

$$(g^u, g^v) = (g^{u_i}, g^{v_i}) \quad \Longleftrightarrow \quad k \equiv i \pmod{q} \ .$$

In analogy to the Pohlig-Hellman algorithm, k can be computed modulo the prime powers of $|E|$, and hence modulo $|E|$. From k, compute $kP = Q$, and x is the first coordinate of this point.

The running time is $O(B \cdot (\log p)^2)$ group operations in G and field operations in $GF(p)$, and $O((\log p)^3)$ calls to the DH oracle for G. With the time-memory tradeoff, both complexities can be replaced by $O(\sqrt{B} \cdot (\log p)^3)$.

3.4 Generalizing the reduction method

From the previous section we can conclude that if each large prime factor p of $|G|$ is single and for every such p a cyclic elliptic curve over $GF(p)$ is known with smooth order, then breaking the DH protocol and computing discrete logarithms are equivalent for G. Maurer and Wolf [36], [33] have relaxed these conditions considerably by generalizing the reduction technique. The notion of an *auxiliary group* was introduced with the property that if for every large prime factor p of $|G|$ a "suitable" auxiliary group is known then the equivalence holds. The required property of such an auxiliary group H over $GF(p)$ is that the elements of H must have a representation by m-tuples of $GF(p)$-elements such that the group operation of H can be performed by a polynomial number of algebraic operations in the $GF(p)$-coordinates. Moreover, a so-called EMBED algorithm must exist that computes, in a polynomial number of algebraic operations in $GF(p)$, from a given $x \in GF(p)$ an element c of H with the property that x is one of the coordinates of c. If these conditions are satisfied, H is called *defined strongly algebraically over* $GF(p)$. Finally, H must be abelian of rank $r = O(1)$. The following theorem has been proved in [33].

Theorem 1 [33] *Let P be a fixed polynomial, and 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 every prime factor p of $|G|$ greater than $B := P(\log |G|)$ is single, and for every such p a finite abelian group H_p with rank $r = O(1)$ is given that is defined strongly algebraically over $GF(p)$ and whose order $|H_p|$ is B-smooth (and known), then breaking the Diffie-Hellman protocol for G with respect to g is probabilistic polynomial-time equivalent to computing discrete logarithms in G to the base g.*

Various types of groups have been proved to be useful auxiliary groups [60], [33], [59]. The use of the groups $H_p = GF(p)^*$ as auxiliary groups leads to the results of den Boer [15], who proved that the DH problem and the DL problem are equivalent for groups G for which $\varphi(|G|)$ is smooth, where φ is Euler's totient function. This condition is equivalent to the condition that large prime factors p of $|G|$ are single and have the property that $p-1$ is smooth.

The use of elliptic curves, as shown in the previous section, has been proposed by Maurer [32], and the application of Jacobians of higher-degree curves was investigated in [60].

Subgroups of $GF(p^n)^*$ as auxiliary groups have been examined in [60] and [36], where it has been shown that the subgroups of $GF(p^{kl})^*$ of order

$$\frac{p^{kl}-1}{p^k-1} = p^{k(l-1)} + p^{k(l-2)} + \cdots + p^k + 1 \qquad (1)$$

are suitable auxiliary groups for k and l polynomial in $\log |G|$, as well as the subgroups of $GF(p^n)^*$ of order $\Phi_n(p)$, i.e., the n-th cyclotomic polynomial evaluated in p, for $n = O(1)$. Without going into the details, we roughly describe the EMBED algorithm in this case. Membership of an arbitrary field element α in the subgroup H is characterized by the equation $\alpha^{|H|} = 1$, and in a normal basis representation of the extension field, this condition is equivalent to a polynomial equation in the $GF(p^k)$-coefficients of this element if H has order (1), and to a *system* of polynomial equations over $GF(p)$ if $|H| = \Phi_n(p)$. In the first case, the EMBED algorithm works as follows. The element $x \in GF(p)$ can be assigned to one of the coordinates, and one can solve the polynomial equation for one of the other coordinates in probabilistic polynomial time by the Cantor-Zassenhaus algorithm (see for example [56]). It is crucial that this algorithm only uses algebraic operations in the underlying field. The second case is treated analogously, but a system of polynomial equations has to be solved by the EMBED algorithm. As described in [60], this can be done with Gröbner bases computations, using algebraic operations only. The use of these groups as auxiliary groups leads to the following theorem.

Theorem 2 **[33]** *Let P be a fixed polynomial and let c be a fixed constant. Let G be a cyclic group with generator g. Assume that all the prime factors p of $|G|$ greater than $P(\log|G|)$ are single. Then there exists an algorithm that makes queries to a DH oracle for G and computes s from g^s in probabilistic time $\sqrt{B}\cdot(\log|G|)^{O(1)}$, where B is the minimum of the largest prime factors of the numbers $(p^{kl}-1)/(p^k-1)$ for $k,l \le P(\log|G|)$, and of the numbers $\Phi_n(p)$ for $n \le c$.*

For the case where the group order $|G|$ is divisible by a large multiple prime factor p a rather pessimistic result was proved. It follows from a result by Shoup [55] (see also [35], [34], [59]) that unless additional assumptions are made on G (i.e., on the representation of the group elements), an efficient reduction from the DL problem to the DH problem cannot exist for G. More precisely, every reduction has running time $\Omega(\sqrt{p})$. In [35], [59], additional lower bounds on the complexity of such reductions (and in particular on the number of required calls to the DH oracle) are proved.

3.5 Non-uniform equivalence

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

$$p - 2\sqrt{p} + 1 \le |E_{a,b}(GF(p))| \le p + 2\sqrt{p} + 1 \ ,$$

and that for each $d \in [p - 2\sqrt{p} + 1, p + 2\sqrt{p} + 1]$ there exists a cyclic elliptic curve over $GF(p)$ with order d [51]. This implies the following non-uniform reduction of the DL problem to the DH problem. For a number n, we define $\nu(n)$ to be the minimum of the set of largest prime factors of the numbers d in the interval $[n - 2\sqrt{n} + 1, n + 2\sqrt{n} + 1]$.

Theorem 3 **[33]** *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 $P(\log|G|)$, there exists an algorithm that makes calls to a DH oracle for G and computes discrete logarithms of elements of G in time*

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

Very little is known about the existence of smooth numbers in the interval of interest, i.e., about $\nu(p)$, for a given prime p. However, it is known [11] that for every fixed u,

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

where $\psi(n, y)$ denotes the number of integers $\le n$ with no prime divisor $\ge y$. This fact suggests that $\nu(n)$ is polynomial in $\log n$.

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

This assumption implies the existence of a $(\log p)^{O(1)}$-smooth cyclic elliptic curve over $GF(p)$ for each prime number p. Therefore for every cyclic group G there exists a small piece of information, which depends only on the order of G, that makes breaking the Diffie-Hellman protocol and computing discrete logarithms equivalent in G. This information is a string S consisting of the prime factors p_i of $|G|$ and appropriate elliptic curve parameters a_i and b_i for all p_i.

Corollary 4 [32] *Let P be a fixed polynomial. If Smoothness Assumption 1 is true, then for every group $G = \langle g \rangle$ whose order contains no multiple prime factors greater than $B := P(\log |G|)$, there exists a string S of length at most $3 \log |G|$ such that when given S, breaking the Diffie-Hellman protocol for G is polynomial-time equivalent to computing discrete logarithms in G.*

3.6 Uniform equivalence

Boneh and Lipton [5] used the presented technique to show that there exists an algorithm that reduces the DL problem to the DH problem in heuristic subexponential time for all groups. Such a result is of interest for groups for which no subexponential-time algorithm for the computation of discrete logarithms is known, such as non-supersingular elliptic curves or Jacobians of hyperelliptic curves.

The basic idea of the reduction is as follows. Like in Lenstra's elliptic curve integer factoring method [28], elliptic curve parameters are chosen at random until a curve with subexponentially-smooth order is generated. The running-time analysis of such an algorithm is based on the following conditions. First, the orders of elliptic curves with randomly chosen parameters should be (almost) uniformly distributed in the interval of interest (or a substantial sub-interval). Secondly, the density of smooth numbers in this interval must be high enough. The first problem was solved by Lenstra [28]. The orders of a non-negligible fraction of the elliptic curves with randomly chosen a, x, and y lie in the interval $(p+1-\sqrt{p}, p+1+\sqrt{p})$ and are distributed close to uniformly in this interval. The second problem, concerning smooth numbers in small intervals of type $[x, x+\Theta(\sqrt{x})]$ has not been solved, hence the running time of the algorithm is heuristic. As mentioned earlier, the density of smooth numbers in sufficiently large intervals satisfies (2). Motivated by this, the plausible assumption is often made that the statement concerning the density also holds for smaller intervals (see [28] or [5]). We write $L_\alpha(p)$ for $\exp((\log p)^\alpha (\log\log p)^{1-\alpha})$.

Smoothness Assumption 2 *Integers that are randomly chosen from the interval* $(p+1-\sqrt{p}, p+1+\sqrt{p})$ *are* $L_\alpha(p)$*-smooth with probability at least* $1/L_{1-\alpha}(p)^{1-\alpha+o(1)}$ *for any* α*.*

This statement for $\alpha = 1/2$ allows a running-time analysis of both the elliptic curve integer factoring algorithm and of the reduction of the DL problem to the DH problem considered in this section. The assumption for $\alpha = 2/3$ and $\alpha = 1/2$ is necessary for a reduction of algebraic algorithms which is described in Section 3.7.

By using the described method of choosing elliptic curves over $GF(p)$ at random, computing their order in polynomial time by a method due to Schoof [54], checking the smoothness of this order with the elliptic curve factoring algorithm (and choosing a new curve unless the order is $L_{1/2}(p)$-smooth), and applying the technique described in the previous sections, Boneh and Lipton showed in [5] that the black-box field problem in $GF(p)$ can be solved in time $L_{1/2}(p)^{2+o(1)}$. This immediately leads to the following result.

Theorem 5 **[5]** *Assume that Smoothness Assumption 2 is true for* $\alpha = 1/2$*. For groups* $G = \langle g \rangle$ *with the property that* $|G|$ *has no multiple prime factor greater than* $L_{1/2}(|G|)^2$ *there exists an algorithm that makes queries to a DH oracle for* G *with respect to* g *and that computes* s *from* g^s *in time* $L_{1/2}(|G|)^{2+o(1)}$*.*

Of course this result is useless for groups in which the DL problem can be solved in time $L_{1/2}(|G|)^{2+o(1)}$ such as the multiplicative group of a finite field. Non-supersingular elliptic curves have been proposed to be used in DL based systems because it is assumed that no subexponential algorithm for solving the DL problem exists for these groups. Theorem 5 implies that *if* elliptic curves have the advantage that no subexponential solution to the DL problem exists, *then* they have the *additional* advantage that this also leads to a lower bound of the complexity for breaking the DH protocol.

In the next section an even faster subexponential reduction will be described which works under the additional condition that a DH oracle for elliptic curves is given by a subexponential-time *algebraic* algorithm.

In order to obtain an integer factoring algorithm with rigorously proven running time, Lenstra, Pila, and Pomerance [27] have studied the use of hyperelliptic curves of genus 2 instead of elliptic curves. Jacobians of these curves have the advantage that the group order varies in an interval of size $[x, x+\Theta(x^{3/4})]$ instead of $[x, x+\Theta(\sqrt{x})]$. In a series of papers, the first of which has already been published [27], they prove that the factoring algorithm using Jacobians of hyperelliptic curves of genus 2 has running time at most

$$L_{2/3}(p)^{O(1)} \cdot (\log n)^2 ,$$

where p is the largest prime factor of n. In the mentioned first paper of the series a result on smooth numbers is shown which roughly corresponds to the statement of Smoothness Assumption 2 for $\alpha = 2/3$, and for larger intervals of the form $[x, x+y]$, where $x^{3/5} \leq y \leq x$.

Because of the similarities of the two methods, the analysis of the integer factoring algorithm also yields a *provable* subexponential solution of the black-box field problem, and hence a subexponential reduction of the DL problem to the DH problem.

Theorem 6 *Let c be a fixed constant. For groups $G = \langle g \rangle$ with the property that $|G|$ has no multiple prime factor greater than $L_{2/3}(|G|)^c$ there exists an algorithm that makes queries to a DH oracle for G with respect to g and that computes s from g^s in time $L_{2/3}(|G|)^{O(1)}$.*

As in the case of the factoring algorithms, this reduction is less efficient and less practical than the reduction using elliptic curves, but it is an improvement in the sense that it is so far the best rigorously proved complexity result.

3.7 Reducing the DL problem to algebraic algorithms solving the DH problem

Several results were obtained in [60], [33], [59], [5], and [13] under the assumption that *algorithms* with certain properties exist for solving the DH problem in certain classes of groups.

As an example, consider the following situation. Assume that the DH problem is easy in the groups $GF(p)^*$ or more precisely, that there exists a polynomial-time *algebraic* algorithm for solving the DH problem in $GF(p)^*$. Let further G be an arbitrary cyclic group with order $|G| = p$, where p is prime, $(p-1)/2 = q$ is prime, and $q-1$ is $(\log p)^{O(1)}$-smooth. Then there exists an efficient reduction of the DL problem to the DH problem in G. (Note that a direct application of the auxiliary group technique presented above does not lead to such a reduction.) The idea of this reduction is to apply the algorithm for solving the DH problem on elements x of $GF(p)^*$ that are only implicitly represented (i.e., by an element g^x of G), and where the oracle's answer is also implicit.

Let $a = g^x$, $x \equiv: \alpha^w \pmod{p}$, and $w \equiv: \beta^v \pmod{q}$ (where α and β are generators of $GF(p)^*$ and $GF(q)^*$, respectively). For the computation of the discrete logarithm x it is sufficient to compute v. Because $q-1$ is smooth, v *could* be computed (efficiently) from x when given a DH oracle for $GF(p)^*$. Unfortunately, x is not known. However, because the oracle for $GF(p)^*$ is given by an algebraic algorithm, this algorithm can be executed (with the same complexity) even on elements of $GF(p)^*$ given only implicitly by group elements of G, such as x (given by $a = g^x$). The reason is that the execution of the algorithm solving the DH problem can be reduced to a polynomial number of group operations, equality tests, and DH oracle calls in G. Hence a DH oracle for G allows to compute (from a) efficiently v, w, and finally x.

Of course it is not likely that an efficient algebraic algorithm exists for solving the DH problem in the groups $GF(p)^*$. Nevertheless, the result can be of some interest because it proves that if the DL problem is hard in G, then *either* the same holds for the DH problem in G, *or* the DH problem cannot be solved efficiently in the groups $GF(p)^*$ by an algebraic algorithm.

The above technique can be generalized in the following two ways. First, it was shown that the iteration depth can be increased [60], [33], [13]. The reduction complexity is then exponential in this depth and hence polynomial only as long as the depth is bounded by a constant. Furthermore, it was pointed out that it is possible to use different auxiliary groups (than multiplicative groups of finite fields) for the reduction process [60], [33], [5]. We briefly describe some of the results.

In [33], the technique was used to prove results like the following. Assume that there exists a sequence of groups of length $l = O(1)$, starting with G and such that the last group of the sequence has smooth order, such that all the other group orders are divided by precisely *one* large prime factor and such that each group is defined strongly algebraically over $GF(p_i)$, where p_i is the large prime factor of the order of the preceding group. If for all groups in the sequence (except for G and the smooth-order group) there exists a polynomial-time algebraic algorithm solving the DH problem, then the polynomial-time equivalence of the DH problem and the DL problem holds for G.

Boneh and Lipton [5] proved a uniform result by using the technique with an iteration depth 2, and with randomly chosen elliptic curves as auxiliary groups. They showed that under Smoothness Assumption 2 the black-box field problem can be solved in time $L_{1/3}(p)^{2+o(1)}$ under the additional assumption that the DH problem can be solved in time $L_{1/3}(q)$ in an elliptic curve over $GF(q)$ by an algebraic algorithm. Under this condition, the result leads to a reduction of the DL problem to the DH problem of complexity $L_{1/3}(p)^{2+o(1)}$, which is substantially faster than the $L_{1/2}(p)^{2+o(1)}$-reduction of Theorem 5. Here, an increase of the iteration depth does not lead to a more efficient reduction. The reason is that according to Smoothness Assumption 2 for $\alpha = 2/3$, a $L_{2/3}(p)$-smooth curve can be found in time $L_{1/3}(p)$, and that it takes roughly time $L_{1/2}(L_{2/3}(p)) \approx L_{1/3}(p)$ (under Smoothness Assumption 2 for $\alpha = 1/2$) to factor the order of this curve. Hence the running time of the first iteration step (and hence of the entire reduction) cannot be smaller than $L_{1/3}(p)$, regardless of the iteration depth.

Cherepnev [13] used the same method to construct an algorithm for solving the DL problem from a (general-purpose) algorithm for solving the DH problem. However, Shoup directly showed [55] that the DH problem is hard to solve by a general-purpose algorithm (see Section 5.1). Hence the reduction of [13] is not needed for proving a lower bound on the complexity of solving the DH problem in this model.

4 Equivalence between various types of DH oracles

4.1 Motivation

Security of a cryptographic protocol means that it is hard to break the scheme even with small (but non-negligible) probability. In this section we show that breaking the DH protocol with a small constant probability is (almost) as hard as breaking it for all instances. For example, an oracle answering correctly with probability 0.1% can efficiently be transformed into an oracle that answers correctly with probability 99.9% (see Section 4.2). This is not obvious in general because of the (assumed) hardness of the DHD problem for certain groups (see Section 2.2). In Section 4.3, the relationship between the DH problem in G and in subgroups of G is studied, and Section 4.4 is concerned with the most significant bits of the DH key.

4.2 ε-DH-oracles

First, we consider probabilistic oracles that answer correctly with a certain non-negligible probability $\varepsilon > 0$. In most cases, such an oracle can be efficiently transformed into a virtually perfect oracle. This problem was considered by Maurer and Wolf [36] and subsequently but independently by Shoup [55]. We briefly describe both approaches to solve this problem.

Definition 6 For $\varepsilon > 0$, an *ε-DH-oracle* is a probabilistic oracle which returns for an input (g^u, g^v) the correct answer g^{uv} with probability at least ε if the input is uniformly distributed over $G \times G$. The *offset* of the oracle's answer g^{uv+t} to the input (g^u, g^v) is defined as $t \pmod{|G|}$. A *translation-invariant* ε-DH-oracle is an ε-DH-oracle whose offset distribution is the same for every input (g^u, g^v).

A special case of (not translation-invariant) ε-DH-oracles are *deterministic* oracles answering correctly for a fraction ε of all inputs.

The first step in both reductions mentioned above, which was described in [32], is to transform the oracle into a translation-invariant oracle by randomizing the input.

Lemma 7 [32] *An ε-DH-oracle for a cyclic group G with order $|G|$ can be transformed into a translation-invariant ε-DH-oracle. One call of the latter requires one call to the former and $O(\log |G|)$ group operations.*

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

The straight-forward approach to using a translation-invariant ε-DH-oracle may at first sight appear to be to run it $O(1/\varepsilon)$ times until it produces the correct answer. However, because even the DHD problem is assumed to be hard in general (see Section 2.2), a more complicated approach must be used because the correct answer cannot be detected efficiently.

We describe Shoup's approach [55] to constructing an oracle that answers correctly with very high probability $1-\alpha$ from a translation-invariant ε-DH-oracle. Given the input (g^u, g^v), the faulty oracle is called $k = O(1/\varepsilon)$ times in order to obtain a list of group elements which contains the correct answer with probability greater than 7/8. In a second step, x and y are chosen at random, and the oracle is called with the input $((g^u)^x g^y, g^v)$. Again, a list is generated that contains the correct answer with probability greater than 7/8. Finally, for all pairs of elements (g_i, g'_j), where g_i and g'_j come from the first and second list, respectively, it is checked whether

$$g_i^x (g^v)^y = g'_j \tag{3}$$

holds. Because

$$(g^{uv})^x (g^v)^y = g^{(ux+y)v} ,$$

equation (3) is satisfied if both g_i and g'_j are the correct answers. If (3) is satisfied for exactly *one* pair, then g_i is the output of the algorithm, and otherwise, failure is reported. This algorithm is run $O(\log(1/\alpha))$ times in order to obtain an almost perfect oracle that answers correctly with probability at least $1-\alpha$, and the entire algorithm makes $O(1/\varepsilon \cdot \log(1/\alpha))$ queries to the translation-invariant ε-oracle.

The central argument in the proof of the correctness of the method is that (3) only holds with very small probability *unless* g_i and g'_j are correct answers. Unfortunately, this is true only if the group order contains no small prime factors. However, if the group order is known, then the "smooth" part of the group order can be treated separately. More precisely, $u \cdot v$ modulo the small prime factors can be computed directly from g^u and g^v by the Pohlig-Hellman method, and by Chinese remaindering in the exponent, the results of the smooth and the "non-smooth" part can be combined to the correct answer of the oracle. The only case in which this transformation is not necessarily successful is when the group order is unknown and contains at least one prime factor p smaller than some bound of order $1/\varepsilon^2$.

The method presented in [36] for correcting a faulty oracle is based on the following idea. First, the translation-invariant ε-oracle is called repeatedly with the input (g^0, g^0) in order to determine the distribution of the offset. In the second phase, the oracle is called repeatedly with the input (g^u, g^v). Comparison of the offset patterns leads to the correct answer g^{uv} with probability $1-\alpha$. This method is less efficient than the algorithm of Shoup. For instance, the faulty oracle must be called $O(\log(1/\alpha\varepsilon)/\varepsilon^4)$ times. On the other hand, the advantage of this method is that in the case of unknown group order, the condition concerning small prime factors is weaker: the method works if all the prime factors of $|G|$ are greater than $1/\varepsilon$. Examples of "malicious" oracles that *cannot* be transformed into perfect oracles with either of the methods 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.

Certain oracles answering correctly only for a negligible fraction of the inputs are as strong as a perfect oracle. An example considered in [36] is the "squaring-DH-oracle" that answers correctly on the input (g^u, g^v) only if $u = v$. The reduction exploits that $(g^{uv})^2 = g^{(u+v)^2}(g^{u^2})^{-1}(g^{v^2})^{-1}$.

A result related to the above has also been shown for the DHD problem by Naor and Reingold [44]. They proved that the DHD problem in a group

G with generator g and of prime order is as hard in the average case as it is in the worst case by giving a method for randomizing the input. When given the triple (g^a, g^b, g^c), choose randomly r, s, and t and compute the group elements $g^{a'} = (g^a)^r g^s$, $g^{b'} = g^b g^t$, and $g^{c'} = (g^c)^r (g^a)^{rt} (g^b)^s g^{st}$. It is not difficult to see that the triple $(g^{a'}, g^{b'}, g^{c'})$ has the property that it is a correct DH triple, i.e., $a' \cdot b' \equiv c' \pmod{|G|}$, if the same holds for the original triple, but that it is otherwise statistically independent of (g^a, g^b, g^c). This randomization is possible only if $|G|$ is a prime.

4.3 The security of subgroups and generator changes

In this section we address the questions whether a subgroup of G is more or less secure than G with respect to the Diffie-Hellman protocol, and whether changing the generator of the group can change the complexity of breaking the DH protocol. We assume here that the order of G is known.

Theorem 8 [33] *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. The conditions of Theorem 8 are optimal. Shoup [55] proved that if the conditions are not satisfied, then the construction of a DH oracle for $\langle g^r \rangle$ from a DH oracle for $\langle g \rangle$ is hard in the generic model.

Proof Sketch. 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 algorithm solving the DH problem in 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^{rab}. Now, g^{rab} is computed from $g^{r^2 ab}$ by computing the r-th root. More precisely, the $p_i^{f_i}$-th root 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^{rab}, must be determined.

For factors p_i with $f_i \geq e_i$, computing the $(p_i^{-f_i}$ modulo $|G|/p_i^{e_i})$-th power immediately yields the desired root. If $f_i < e_i$ and $p_i \leq B$, a p_i-th root can be computed by a generalization (see [36] or [33]) of a square root method of Massey [31]. The correct root can be determined by computing the discrete logarithms of all the roots modulo the smooth part of $|G|$. □

In case of a generator change, i.e., if $(r, |G|) = 1$, it is not even necessary to know r, as was pointed out by Boneh and Lipton in a preliminary version of [5]. Let $h = g^r$, and let DH_g and DH_h be the DH functions in G with respect to the generator g and h, respectively. Then

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

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

The problem of constructing an oracle for the entire group from a set of oracles for subgroups has been studied in [33] and [35]. It is not difficult to see that g^{uv} can be computed efficiently from $g^{uv|G|/p^e}$ for all the maximal powers p^e of $|G|$ of the large prime factors p. This works by Chinese remaindering in the exponent. The required group elements can be obtained from g^u and g^v by DH oracles for subgroups $\langle g^s \rangle$ if for each large prime factor p of $|G|$ there exists such an oracle for an s that is not a multiple of p. This directly yields a criterion for when a set of subgroup oracles can be used to construct an oracle for G efficiently. It was shown in [35] that this condition is not only sufficient but also necessary unless special assumptions on G are made.

The following intuitive statement follows from the above results. Of course an analogous result holds for the DL problem.

Theorem 9 [33] *Consider a group $G = \langle g \rangle$ and a subgroup $H = \langle g^k \rangle$ of G with smooth index k. Then (and without any assumptions on G only then) the DH problem for H is equivalent to the DH problem for G.*

4.4 The hardness of the most significant bits

A typical way of using the Diffie-Hellman protocol is to take a part of the generated secret key as the session key for encryption with a conventional block cipher. This key is usually shorter than the Diffie-Hellman key, and a natural method is to use the block consisting of its first k bits in a binary representation, the so-called most significant bits. However, it is conceivable that an adversary who is not able to break the Diffie-Hellman protocol can nevertheless compute these bits efficiently. Boneh and Venkatesan [6] investigated the security of the most significant bits in the Diffie-Hellman protocol (and other schemes) in the groups $\mathbf{Z}_p^*$ for prime numbers p. They considered the following two functions (where p and k are fixed).

Definition 7 For any $\alpha, h \in \mathbf{Z}_p^*$, let

$$N_{\alpha,h}^k(x) := \mathrm{msb}_k(\alpha \cdot h^x \bmod p)$$

and

$$D^k(g^a, g^b) := \mathrm{msb}_k(g^{ab}) ,$$

where msb_k denotes the k most significant bits.

They study the following *hidden number problem.* Given an oracle that computes the function $N_{\alpha,h}^k(x)$ for unknown α, find α. Note that the input to the function that computes $\mathrm{msb}_k(\alpha h^x \bmod p)$ is x, not h^x. The same problem when given an oracle for $\mathrm{msb}_k(\alpha t \bmod p)$ for chosen t is easy even for $k = 1$.

It is not difficult to see that a probabilistic polynomial-time solution of the hidden number problem proves that the k most significant bits together are equally hard as the entire key because

$$N_{g^{uv},g^v}^k(x) = D^k(g^{u+x}, g^v) ,$$

i.e., when given g^u and g^v, an oracle for the function D^k can immediately be used for computing N^k with $\alpha = g^{uv}$, and a solution of the hidden number problem hence yields g^{uv}.

The question remains for which k the hidden number problem can be solved in probabilistic polynomial time. Boneh and Venkatesan proved the following result by using rounding techniques in lattices, based on methods of Lenstra, Lenstra, and Lovasz [29] and Babai [2].

Theorem 10 [6] *Let p be prime, $n = \lceil \log p \rceil$, and let $G = \mathbf{Z}_p^*$. For $k = \lceil \sqrt{n} \rceil + \lceil \log n \rceil$, it is computationally equivalent to compute all the k most significant bits of the Diffie-Hellman key simultaneously and to solve the DH problem. For any $\varepsilon > 0$ and sufficiently large p, this holds for $k = \varepsilon \cdot \sqrt{\log p}$.*

A variant of the Diffie-Hellman protocol is described in [6] which is at most as secure as the original protocol, and for which the most significant bit is hard (i.e., $k = 1$). It is an open problem if the same result also holds for the original DH problem, or whether a *faulty* oracle for the k most significant bits also helps recovering the Diffie-Hellman secret efficiently.

5 Direct security results in restricted computational models

Under certain special conditions on the computational model, one can prove directly that it is hard to break the Diffie-Hellman protocol. Such results have been derived by Shoup [55] and by Coppersmith and Shparlinsky [14].

5.1 Generic algorithms

Shoup proved in [55] that no general-purpose (or generic) algorithm can break the DH protocol in G faster than the Pohlig-Hellman method (see Section 2.3). Intuitively, a generic algorithm is an algorithm that does not make use of any property of the representation of the group elements other than the fact that each group element has a *unique* representation (by some binary string). More precisely, a generic algorithm for the group $\mathbf{Z}_n$ is a probabilistic algorithm that takes as input a list $(\sigma(x_1), \ldots, \sigma(x_l))$, where the x_i are elements of $\mathbf{Z}_n$ and σ is a random encoding of the group elements, i.e., a random mapping $\mathbf{Z}_n \to S$ (S is a set of size n of binary strings). The generic algorithm is allowed to make calls to an oracle that can compute the functions add/sub: $S \times S \to S$ with $\text{add/sub}(\sigma(x), \sigma(y)) = \sigma(x \pm y)$. The following theorem implies that a generic algorithm which breaks the DH protocol with substantial probability cannot run considerably faster than in time $\Theta(\sqrt{p})$, where p is the largest prime factor of the group order. Note that the combination of the Pohlig-Hellman and the baby-step giant-step methods is a generic algorithm and matches this bound, which is hence tight.

Theorem 11 [55] *Let n and S be as above, and let p be a prime factor of n. Let further a generic algorithm for $\mathbf{Z}_n$ (and S) be given that makes at most m oracle queries. The probability that the algorithm answers the input $(\sigma(1), \sigma(x), \sigma(y))$ by $\sigma(xy)$ is at most $(m^2 + 5m + 10)/2p$ when x, y, and the encoding function σ are chosen randomly.*

Proof Sketch. Assume for simplicity that $n = p$. By the m oracle calls, the algorithm can compute the encodings $\sigma_1, \ldots, \sigma_{m+3}$ of linear expressions in x and y (including the input $\sigma(1)$, $\sigma(x)$, and $\sigma(y)$). Unless $\sigma_i = \sigma_j$ for some $i \neq j$, all the algorithm sees are distinct random elements of S. Hence the only information the algorithm obtains is that $\sigma_i \neq \sigma_j$ for all $i \neq j$. The probability of the events $\sigma_i = \sigma_j$ cannot exceed $1/p$. This holds because the probability that a (non-zero) linear expression $r + sX + tY \pmod{p}$

vanishes for random values (x, y) is $1/p$ (or 0). Thus the probability, taken over random (x, y) and random coin tosses of the algorithm, that $\sigma_i = \sigma_j$ for some $i \neq j$ is at most D/p, where $D := (m+3)(m+2)/2$ is the number of two-sets $\{i, j\}$. The probability of guessing $\sigma(xy)$ correctly if $\sigma_i \neq \sigma_j$ for all $i \neq j$ is small. More precisely, one can show that the algorithm answers correctly with probability at most $[(m+3)(m+2)/2 + 2]/p$. □

Clearly, this result also implies the same lower bound for generic algorithms solving the DL problem. Hence the DH problem and the DL problem have the same complexity, at least in the generic model. Shoup also proved that the DHD problem in a group G cannot be solved faster than in time $\Theta(\sqrt{q})$ by a generic algorithm, where here q stands for the *smallest* prime factor of $|G|$.

The methods of [55] can also be used to prove a lower bound on the complexity of generic *reductions* from the DL to the DH problem and in particular a lower bound on the number of required DH oracle calls in such a reduction [35], [59].

5.2 Approximation by polynomials and other classes of functions

Coppersmith and Shparlinsky [14] proved the impossibility of approximating the discrete logarithm function $g^x \mapsto x$ (modulo p) and the function $g^x \mapsto g^{x^2}$ (modulo p) by certain simple classes of functions such as low-degree polynomials. We state one of their results which claims that a polynomial that interpolates the function $g^x \mapsto g^{x^2}$ (note that computing this function and breaking the Diffie-Hellman protocol are equivalent) for a substantial fraction of the inputs must be of very high degree.

Theorem 12 **[14]** *Let g be a generator of $GF(p)^*$, and let $f(x)$ be a polynomial such that*

$$g^{x^2} = f(g^x)$$

for $x \in S$, where $S \subset \{N+1, \ldots, N+H\}$ has cardinality $|S| = H - s$ for some $N, H \leq p-1$ and s. Then $\deg f \geq H - 2s - 3$.

Proof. If $f(g^x) = g^{x^2}$ holds for $H - s$ different values x with $N + 1 \leq x \leq N + H$, then there must be at least $H - 1 - 2s$ values x such that both $f(g^x) = g^{x^2}$ and $f(g^{x+1}) = g^{(x+1)^2} = g^{x^2} \cdot (g^x)^2 \cdot g$ hold. Hence the polynomial

$$h(u) := f(g \cdot u) - f(u) \cdot u^2 \cdot g$$

has at least $H-1-2s$ different roots, is obviously not identical to zero, and has hence degree at least $H-1-2s$. Therefore, $\deg f \geq (H-1-2s)-2 = H-2s-3$. □

A variety of different results are proved in [14] concerning the difficulty of approximating the solutions of the DH problem and the DL problem by polynomials, algebraic functions, Boolean circuits, and linear recurring sequences.

Acknowledgments

The authors would like to thank Dan Boneh, Dima Grigoriev, Hendrik Lenstra, Markus Metzger, Victor Shoup, and Igor Shparlinsky for interesting discussions on the subject of this paper.

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] L. Babai, On Lovasz' lattice reduction and the nearest lattice point problem, *Combinatorica*, Vol. 6, pp. 1–13, 1986.

[3] E. Bach and J. Shallit, Factoring with cyclotomic polynomials, *Math. Comp.*, Vol. 52, pp. 201–219, 1989.

[4] D. Boneh, *Studies in computational number theory with applications to cryptography*, Ph. D. Thesis, Princeton Univ., Nov. 1996.

[5] 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.

[6] D. Boneh and R. Venkatesan, Hardness of computing the most significant bits of secret keys in Diffie-Hellman and related schemes, *Advances in Cryptology - CRYPTO '96*, Lecture Notes in Computer Science, Vol. 1109, pp. 129–142, Springer-Verlag, 1996.

[7] S. Brands, *An efficient off-line electronic cash system based on the representation problem*, Tech. Rep. CS-R9323, CWI, Amsterdam, 1993.

[8] 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.

[9] 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.

[10] R. Canetti, Towards realizing random oracles: hash functions that hide all partial information, *Advances in Cryptology - CRYPTO '97*, Lecture Notes in Computer Science, Vol. 1294, pp. 455–469, Springer-Verlag, 1997.

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

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

[13] M. A. Cherepnev, On the connection between discrete logarithms and the Diffie-Hellman problem, *Discrete Math. Appl.*, 1996.

[14] D. Coppersmith and I. Shparlinsky, *On polynomial approximation and the parallel complexity of the discrete logarithm problem and breaking the Diffie-Hellman cryptosystem*, preprint, Nov. 1996.

[15] 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.

[16] W. Diffie and M. E. Hellman, New directions in cryptography, *IEEE Transactions on Information Theory*, Vol. 22, No. 6, pp. 644–654, 1976.

[17] 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.

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

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

[20] 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.

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

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

[23] N. Koblitz, Hyperelliptic cryptosystems, *Journal of Cryptology*, Vol. 1, pp. 139–150, 1989.

[24] N. Koblitz, Elliptic curve cryptosystems, *Math. Comp.*, Vol. 48, pp. 203–209, 1987.

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

[26] 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.

[27] H. W. Lenstra, Jr., J. Pila, and C. Pomerance, A hyperelliptic smoothness test. I, *Philosophical Transactions of the Royal Society*, Series A, Vol. 345, No. 1676, pp. 397–408, London, 1993.

[28] H. W. Lenstra, Jr., Factoring integers with elliptic curves, *Annals of Mathematics*, Vol. 126, pp. 649–673, 1987.

[29] A. Lenstra, H. W. Lenstra, Jr., and L. Lovasz, Factoring polynomials with rational coefficients, *Mathematische Annalen*, Vol. 261, pp. 515–534, 1982.

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

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

[32] 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.

[33] U. M. Maurer and S. Wolf, The relationship between breaking the Diffie-Hellman protocol and computing discrete logarithms, *SIAM Journal on Computing*, Vol. 28, No. 5, pp. 1689–1721, 1999.

[34] U. M. Maurer and S. Wolf, Diffie-Hellman, decision Diffie-Hellman, and discrete logarithms, *Proc. of the 1998 IEEE Symp. on Information Theory*, Cambridge, U.S.A., p. 327, 1998.

[35] U. M. Maurer and S. Wolf, Lower bounds on generic algorithms in groups, *Proceedings of EUROCRYPT '98*, Lecture Notes in Computer Science, Vol. 1403, pp. 72–84, Springer-Verlag, 1998.

[36] 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.

[37] U. M. Maurer and Y. Yacobi, Non-interactive public-key cryptography, *Designs, Codes, and Cryptography*, Vol. 9, pp. 305–316, 1996.

[38] K. S. McCurley, A key distribution system equivalent to factoring, *Journal of Cryptology*, Vol. 1, No. 2, pp. 95–105, 1988.

[39] 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.

[40] A. J. Menezes, T. Okamoto, and S. A. Vanstone, Reducing elliptic curve logarithms to logarithms in a finite field, *IEEE Transactions on Information Theory*, Vol. 39, pp. 1639–1646, 1993.

[41] A. J. Menezes (Ed.), *Applications of finite fields*, Kluwer Academic Publishers, 1992.

[42] A. J. Menezes, *Elliptic curve public key cryptosystems*, Kluwer Academic Publishers, 1993.

[43] 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.

[44] M. Naor and O. Reingold, Number-theoretic constructions of efficient pseudorandom functions, preliminary version, 1997.

[45] P. C. van Oorschot and M. Wiener, On Diffie-Hellman key agreement with short exponents, *Advances in Cryptology - EUROCRYPT '96*, Lecture Notes in Computer Science, Vol. 1070, pp. 332–343, Springer-Verlag, 1996.

[46] R. Peralta, A simple and fast probabilistic algorithm for computing square roots modulo a prime number, *IEEE Transactions on Information Theory*, Vol. 32, No. 6, pp. 846–847, 1986.

[47] 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.

[48] J. M. Pollard, Monte-Carlo methods for index computation mod p, *Math. Comp.*, Vol. 32, pp. 918–924, 1978.

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

[50] 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.

[51] H. Rück, A note on elliptic curves over finite fields, *Math. Comp.*, Vol. 49, pp. 301–304, 1987.

[52] K. Sakrai 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.

[53] 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.

[54] 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.

[55] 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.

[56] I. E. Shparlinsky, *Computational problems in finite fields*, Kluwer Academic Publishers, 1992.

[57] 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.

[58] C. P. Waldvogel and J. L. Massey, The probability distribution of the Diffie-Hellman key, *Advances in Cryptology - AUSCRYPT '92*, Lecture Notes in Computer Science, Vol. 718, pp. 492–504, Springer-Verlag, 1993.

[59] S. Wolf, *Information-theoretically and computationally secure key agreement in cryptography*, ETH dissertation No. 13138, Swiss Federal Institute of Technology (ETH Zurich), May 1999.

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