Searching for Elements in Black Box Fields and Applications

Dan Boneh
dabo@cs.stanford.edu
Computer Science Dept.
Stanford University
Satnford, CA., 94305-9045

Richard Lipton*
rjl@cs.princeton.edu
Computer Science Dept.
Princeton University
Princeton, NJ, 08544

Abstract

We introduce the notion of a black box field and discuss the problem of explicitly exposing field elements given in a black box form. We present several sub-exponential algorithms for this problem using a technique due to Maurer. These algorithms make use of elliptic curves over finite fields in a crucial way. We present three applications for our results: (1) We show that any *algebraically homomorphic* encryption scheme can be broken in expected sub-exponential time. The existence of such schemes has been open for a number of years. (2) We give an expected sub-exponential time reduction from the problem of finding roots of polynomials over finite fields with low straight line complexity (e.g. sparse polynomials) to the problem of testing whether such polynomials have a root in the field. (3) We show that the hardness of computing discrete-log over elliptic curves implies the security of the Diffie-Hellman protocol over elliptic curves. Finally in the last section of the paper we prove the hardness of exposing black box field elements in a field of characteristic zero.

Mathematics Subject Classification: 11Y16, 11T71.

Keywords: Black box fields, Diffie-Hellman, Elliptic-Curves, Straight line complexity, Homomorphic Encryption, Sparse Polynomials.

1 Introduction

An algebraic structure is often defined as a set of operators acting on some universe. Usually there is no reference as to how the elements in the universe are represented. One can design algorithms for such an abstract algebraic structure by providing the algorithm with oracles for the various operators. We refer to such a representation of an algebraic structure as a *black box representation*. The most widely studied structure given in this fashion is the black box group [3].

In this paper we study *fields* given in a black box representation. We refer to such fields as black box fields, or BBF for short. The definition of black box fields will be given in Section 2. For now we give a high level description. Let K be a BBF. Intuitively speaking, the elements of K are represented as arbitrary binary strings. For an element $x \in K$ we denote by $[x]$ the binary string representing the element x. We refer to $[x]$ as the *black box representation* of x. A black box field algorithm has access to oracles that given $[x]$ and $[y]$ compute the black box representation of the sum $[x+y]$ and product $[xy]$. Similarly, there is an oracle that given $[x], [y]$ will output "true" if and only if $x = y$. Finally, the algorithm is provided with an oracle that given $x \in K$ will output $[x]$.

*Supported in part by NSF CCR-9633193

We will be most interested in the following problem: let p be a prime and $K = \mathbb{F}_p$ be a finite field given as a BBF. Find an algorithm that takes as input the black box representation of a field element $[\alpha]$ and outputs an integer $0 \leq a < p$ such that $a \equiv \alpha \pmod{p}$. We refer to this problem as the black box field problem, or BBFP for short. A trivial algorithm for this problem is to test all elements of $\mathbb{F}_p$ one by one by using the equality test oracle. The running time of this algorithm is $O(p)$. We are interested in finding algorithms for BBFP whose running time is substantially less than p.

In Section 3 we present two randomized algorithms for BBFP whose expected running time is sub-exponential in $\log p$. Our algorithms are based on a technique due to Maurer [23]. The existence of sub-exponential algorithms for BBFP is surprising when contrasted with a result of Nechaev [30] and Shoup [34]. They considered the equivalent problem to BBFP over groups, i.e. where elements can be added, but not multiplied. They show that for the group $G = \mathbb{Z}/p\mathbb{Z}$ the best algorithm for finding a hidden element must take time $\Omega(\sqrt{|G|})$.

Sub-exponential algorithms for BBFP have several applications. The first application shows that any *algebraically homomorphic* encryption scheme can be broken in sub-exponential time. Such cryptosystems are desirable since they enable non-interactive two-player secure function evaluation [1]. We define these concepts in Section 4.1. The existence of algebraically homomorphic encryption schemes was posed as an open problem in [16, pp. 6–7]. One may view this application as a general cryptanalytic tool: to show that a cryptosystem can be broken in sub-exponential time it suffices to show that it is algebraically homomorphic.

In Section 4.2 we show our algorithms for BBFP provide a sub-exponential time reduction from the problem of finding roots of polynomials over $\mathbb{F}_p$ given as straight line programs to testing whether such polynomials have a root in $\mathbb{F}_p$. Currently the best algorithms for finding roots of sparse polynomials in $\mathbb{F}_p$ take time polynomial in the *degree* of the polynomial [5]. Indeed, the best algorithms for finding roots of sparse polynomials of exponential degree (e.g. degree $p^{1/3}$) take exponential time in the size of their input. Consequently, our reduction suggests that just testing if a polynomial given as a straight line program has a root cannot be done in sub-exponential time (in the size of the program). Alternatively, to design a sub exponential algorithm for finding roots of sparse polynomials our results show that it suffices to design an algorithm for *testing* if a polynomial of low complexity has a root in $\mathbb{F}_p$.

An important motivation for studying the black box field problem is that algorithms for BBFP can be used to prove the security of the Diffie-Hellman secret key exchange protocol [14]. In Section 4.3 we show that efficient algorithms for BBFP imply that breaking the Diffie-Hellman protocol is equivalent to computing discrete log. Proving the equivalence of breaking the Diffie-Hellman protocol and computing discrete-log is one of the oldest problems in public key cryptography. The existence of a sub-exponential algorithm for BBFP has the following consequence: let G be a group in which the discrete log problem can not be solved in sub-exponential time. Then in the group G the Diffie-Hellman protocol can not be broken in sub-exponential time. The group generated by the points of an elliptic curve over a finite field is an example of a group for which there is no known sub-exponential algorithm for computing discrete log[1] (this is the main motivation for using elliptic curve cryptosystems [18, 28]). Hence, our results show that if computing discrete log in the group of points of an elliptic curve is hard then the Diffie-Hellman protocol in such groups is secure. The existence of such a reduction demonstrates yet another advantage of elliptic curve cryptosystems. These results complement the elegant results of Maurer and Wolf [23, 24, 37] who study the equivalence of Diffie-Hellman and discrete log in the group $\mathbb{Z}_p^*$ for various values of p.

Finally in Section 5 we consider an equivalent of BBFP over fields of characteristic zero. We show that solving BBFP over the rationals is as hard as factoring integers. This negative result suggests that

[1] A sub-exponential algorithm is known for the rare event when the curve is supersingular [27].

performing computations over black box fields of characteristic zero is much harder than over black box fields of positive characteristic.

2 Black-box fields

A black box field is an abstract algebraic construct motivated by various applications. We begin by giving a precise definition of black box fields (BBF) and the black box field problem (BBFP).

Definition 2.1 *A* black box field *is a six-tuple: (p, n, h, F, G, T) where p is a prime and n is a positive integer representing the encoding length. The functions h, F, G, T are defined as follows:*

1. *The function $h : \{0,1\}^n \to \mathbb{F}_p$ associates a field element with every n-bit binary string. The function h is surjective, i.e. every field element is represented by at least one binary string.*
2. *The functions $F, G : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ perform addition and multiplication. They satisfy the following relations: $h(F(x,y)) = h(x) + h(y)$ and $h(G(x,y)) = h(x)h(y)$.*
3. *The function $T : \{0,1\}^n \times \{0,1\}^n \to \{\texttt{true},\texttt{false}\}$ tests equality of two black box elements: $T(x,y) = \texttt{true}$ if and only if $h(x) = h(y)$.*

Notice that an element $x \in \mathbb{F}_p$ can be represented by many different n-bit binary strings since $h^{-1}(x)$ is a set of arbitrary cardinality. Throughout the paper we use $[x]$ to denote *some* binary string representing the field element x, i.e. $h([x]) = x$. The functions F and G compute $[x+y]$ and $[xy]$ given $[x]$ and $[y]$. This is consistent with the notation used in the introduction. We usually refrain from mentioning the functions h, F, G, T explicitly. When we say that $\mathbb{F}_p$ is given as a black box field we assume that these functions have already been agreed upon. As an abuse of notation we occasionally write $[x] \in \mathbb{F}_p$ which is to be understood as saying that $[x]$ is a black box representation of the field element $x \in \mathbb{F}_p$.

In Section 4 we give examples where black box fields arise naturally. To familiarize the reader with the concept of a black box field we present a few simple algorithms for such fields. The first is computing the inverse: given $[x] \in \mathbb{F}_p$ we wish to compute $[x^{-1}]$. This can easily be done by observing that $[x^{-1}] = [x^{p-2}]$. Using repeated squaring this requires $O(\log p)$ applications of the multiplication function. Another example is that of computing $[\sqrt{x}]$ given $[x]$ if it exists. An algorithm due to Shanks [9, pp. 32–33] can find square roots in finite fields using only operations which are supported in black box fields. Hence, the algorithm can be applied to $[x]$ and it will output $[\sqrt{x}]$. Notice that both examples require that the field characteristic p be known. As a final example we observe that one can manipulate polynomials over black box fields. A polynomial $p(x) \in \mathbb{F}_p[x]$ over a black box field is a formal polynomial of the form:

$$p(x) = [a_n]x^n + [a_{n-1}]x^{n-1} + \ldots + [a_1]x + [a_0] \ .$$

Two such polynomials $p(x), q(x)$ can be added, subtracted, multiplied and even divided in polynomial time. Consequently, one can manipulate elements in an algebraic extension of a black box fields.

Definition 2.2 *Let (p, n, h, F, G, T) be a black box field for some prime p. We denote the map sending x to some $[x]$ by $[]$. The* black box field problem *is the following: find an algorithm A that given p and oracles for $F, G, T, []$ and an element $[\alpha] \in \mathbb{F}_p$ finds α explicitly. Formally, $A^{F,G,T,[]}([\alpha]) = a$ where $a \equiv \alpha \pmod p$. The algorithm is said to run in polynomial time if it runs in time $\log^{O(1)} p$. The algorithm is sub-exponential if it runs in sub-exponential time in $\log p$.*

The main goal of this paper is to provide algorithms for solving the black box field problem. We first show that it is possible to obtain a small number of bits that uniquely define elements of the finite field $\mathbb{F}_p$. This will show that the black box field problem is information theoretically easy.

Conjecture 2.1 *Let p be a prime and set $k = \lceil 2\log^2 p \rceil$. For an element $x \in \mathbb{F}_p$ define the signature of x as the vector:*

$$sig(x) = \left(\left(\frac{x}{p}\right), \left(\frac{x+1}{p}\right), \ldots, \left(\frac{x+k}{p}\right) \right) \quad .$$

where $\left(\frac{x}{p}\right)$ is the Legendre symbol of x over p. Then we conjecture[2] *that for sufficiently large p, any two distinct elements x, y in $\mathbb{F}_p$ satisfy $sig(x) \neq sig(y)$.*

Problems similar to Conjecture 2.1 were studied by Davenport [11]. The identity $\left(\frac{x}{p}\right) \equiv x^{\frac{p-1}{2}} \pmod p$ shows that given $[x]$ one can compute $\mathrm{sig}(x)$ in a black box field using $O(\log^3 p)$ applications of the oracles. Assuming the conjecture is true, $\mathrm{sig}(x)$ provides enough information to recover x. Unfortunately, there is no known polynomial time algorithm for finding x given $\mathrm{sig}(x)$. In fact, Damgård [10] suggested using this sequence as a pseudo-random sequence.

The argument above shows that one can not hope to obtain an information theoretic lower bound on the number of oracle calls needed to solve BBFP. Assuming conjecture 2.1 holds, a polynomial number of oracle calls are sufficient to completely constrain the hidden element $[x]$. This situation is substantially different than the one which occurs in black box groups. As was mentioned in the introduction, Nechaev and Shoup showed that to solve the equivalent problem in the group $\mathbb{Z}/p\mathbb{Z}$ one must make at least $\Omega(\sqrt{p})$ oracle queries to obtain enough information about the hidden element.

3 Algorithms for black-box fields

In this section we describe two sub-exponential algorithms for BBFP. The results will depend on an assumption which is often used in computational number theory:

Smoothness assumption

Let $L_\alpha(p)$ be the function $L_\alpha(p) = \exp\left(\log^\alpha p \, \log\log^{1-\alpha} p\right)$. For an integer x let $d(x)$ be the largest prime divisor of x. If $d(x) < \alpha$ we say that x is α-smooth. The assumption we need is that integers chosen uniformly in the range $[p+1-2\sqrt{p}, p+1+2\sqrt{p}]$ satisfy

$$\Pr_x[d(x) < L_\alpha(p)] > \frac{1}{L_{1-\alpha}(p)^{1-\alpha+o(1)}}$$

for any fixed $\alpha > 0$. When the integer x is chosen in the range $[1, p]$ this assumption is known to be true (see [8, 12]). The assumption when $\alpha = \frac{1}{2}$ is necessary for the running time analysis of the elliptic curve factoring method of Lenstra [21].

3.1 A sub-exponential algorithm for BBFP

This section is devoted to the proof of the following theorem:

[2]Notice that the conjecture implies that $\mathrm{sig}(1) \neq \mathrm{sig}(2)$. This means that there must exist an $1 \leq a \leq k+2$ such that a is a quadratic non-residue modulo p. This is known to hold under the generalized Riemann hypothesis when $k > 2\log^2 p$ (see [4]). That explains our particular choice of k in the conjecture.

Theorem 3.1 *Let K be a finite field of size p given as a black box field. Under the smoothness assumption, BBFP can be solved in expected time*

$$L_{\frac{1}{2}}(p)^{2+o(1)} = \exp\left((2+o(1))\sqrt{\log p \ \log\log p}\right) \quad .$$

and $O(\log p)$ space.

The proof of Theorem 3.1 uses a technique due to Maurer [23]. Before proving Theorem 3.1 we state some simple facts regarding elliptic curves. We denote by $E_{a,b}(p)$ the set of points $(x, y) \in \mathbb{F}_p^2$ on the curve $y^2 = x^3 + ax + b$ plus a point O called the point at infinity. It is well known [35] that there is a natural Abelian group structure defined on the points in $E_{a,b}(p)$. Thus, $E_{a,b}(p)$ forms an Abelian group. Given two points $P_1, P_2 \in E_{a,b}(p)$ we denote their sum in this group by $P_1 + P_2$.

Fact 3.2 *Let $E_{a,b}$ be an elliptic curve over the black box field $\mathbb{F}_p$. Let $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ be two points on $E_{a,b}$. Let $P_3 = P_1 + P_2 = (x_3, y_3)$. Then given $[x_1], [y_1], [x_2], [y_2]$ it is possible to compute $[x_3], [y_3]$ in polynomial time in $\log p$.*

Proof The values x_3, y_3 can be expressed as algebraic expressions of x_1, y_1, x_2, y_2, a, b. See [35, pp. 58–59] for a list of these expressions. Thus, $[x_3], [y_3]$ can be computed from $[x_1], [y_1], [x_2], [y_2]$ using the addition and multiplication oracles. □

Throughout the section we adopt the following notation: for a point $P = (x, y) \in E_{a,b}(p)$ we denote by $[P]$ the point P whose coordinates are given as black box elements, i.e. $[P] = ([x], [y])$. Fact 3.2 shows that given points $[P_1] = ([x_1], [y_1])$ and $[P_2] = ([x_2], [y_2])$ it is possible to construct $[P_1 + P_2] = ([x_3], [y_3])$ in polynomial time. Similarly, using repeated doubling one can construct the point $[kP]$ given a point $[P]$ and an integer k.

Fact 3.3 *Let $E_{a,b}$ be an elliptic curve over $\mathbb{F}_p$. Then the group $E_{a,b}$ can be generated by two points. Furthermore, two random points P_1, P_2 generate the group $E_{a,b}$ with probability at least $\Omega(1/\log^2 p)$.*

Proof The structure of the group $E_{a,b}$ is known to be $E_{a,b} \simeq \mathbb{Z}_n \times \mathbb{Z}_m$ for some m, n where $m|n$ (see [35]). Thus $E_{a,b}$ can be generated by two points. The number of pairs which generate $\mathbb{Z}_n \times \mathbb{Z}_m$ is lower bounded by $\Omega(\varphi(nm)^2)$. The result now follows since always $\varphi(x)/x > \Omega(1/\log x)$. □

Proof of Theorem 3.1 The basic idea is to generate random elliptic curves $E_{a,b}$ over $\mathbb{F}_p$ until a curve with a smooth order is found. It is well known that the number of points on a random elliptic curve over $\mathbb{F}_p$ is approximately uniformly distributed in the range $[p + 1 - 2\sqrt{p},\ p + 1 + 2\sqrt{p}]$ (see [21]). Hence, by the smoothness assumption, after an expected $\exp\left((\frac{1}{2} + o(1))\sqrt{\log p \ \log\log p}\right)$ tries we will have generated a curve $E_{a,b}$ such that the largest prime divisor of $|E_{a,b}|$ is less than $\exp\left(\sqrt{\log p \ \log\log p}\right)$.

Let $E_{a,b}$ be an elliptic curve over $\mathbb{F}_p$ for which the largest prime divisor of $|E_{a,b}|$ is less than d for some $d > 0$. The number of points on the curve $E_{a,b}$ can be found in polynomial time using Schoof's algorithm [33]. We show that using this curve it is possible to solve the black box field problem over $\mathbb{F}_p$ in time $O(d^2 \log p)$. This will prove the theorem. Note that since the prime factors of $|E_{a,b}|$ are small, they can be found in the required time bound.

By Fact 3.3 we can find a pair of points P_1, P_2 which generate the group $E_{a,b}$ by picking them at random. If it so happens that P_1, P_2 do not generate $E_{a,b}$ then the algorithm will fail and we will know that P_1, P_2 were poorly chosen. Fact 3.3 shows that in an expected polynomial number of attempts the pair P_1, P_2 will work.

Given two points $P_1 = (x_1, y_1), P_2 = (x_2, y_2)$ which generate $E_{a,b}$ we show how to solve the black box field problem. Let $[x]$ be a black box representation of some field element $x \in \mathbb{F}_p$. Given $[x]$ our goal is to construct x explicitly, i.e. find an integer w such that $w \equiv x \pmod p$. This can be done in three steps:

1. Embed $[x]$ in the curve $E_{a,b}$, i.e. find $[y] \in \mathbb{F}_p$ so that the point $[P] = ([x],[y])$ is on $E_{a,b}$.
2. Find integers α and β such that $[P] = [\alpha P_1 + \beta P_2]$. Such integers exist since P_1, P_2 generate $E_{a,b}$.
3. Calculate $\alpha P_1 + \beta P_2$ explicitly. The x coordinate of the resulting point, which is an explicit field element, is the required value.

We now explain how to carry out steps 1 and 2. In step 1 we are looking for $[y] \in \mathbb{F}_p$ such that $y^2 = x^3 + ax + b$. Given $[x]$ one can clearly construct $[x^3 + ax + b]$. The required $[y]$ can be found by taking the square root of the element $[x^3 + ax + b]$. This is possible since in Section 2 we saw that given $[z] \in \mathbb{F}_p$ there is an algorithm for constructing $[\sqrt{z}]$ in random polynomial time. If the above element does not have a square root in $\mathbb{F}_p$ we can run through the entire computation using $[\hat{x}] = [x + r]$ for some randomly chosen r. After a constant number of tries it will be possible to embed $[\hat{x}]$ in the curve.

Next we explain how to find α and β as required in step 2. Recall that $E_{a,b} \simeq \mathbb{Z}_n \times \mathbb{Z}_m$ where $n = q_1^{\gamma_1} \cdots q_r^{\gamma_r}$, $m|n$ and all the primes q_i are less than d. To find α, β we use a simple generalization of the Pohlig-Hellman algorithm [31] for discrete log in groups of smooth order. For simplicity we assume that all the γ_i are equal to 1. The Pohlig-Hellman algorithm generalizes to the case where $\gamma_i > 1$ as well. The method for finding α, β is to construct $\alpha, \beta \pmod{q_i}$ for all $i = 1, \ldots, r$ and then use chinese remaindering to find α, β.

Let q_i be a prime dividing n and let $k = n/q_i$. Notice that the points kP_1, kP_2 have order at most q_i. Therefore, since the point kP is in the group generated by kP_1, kP_2 there must exist integers $0 \le \alpha_i, \beta_i < q_i$ such that

$$[kP] = [\alpha_i kP_1 + \beta_i kP_2]$$

These integers can be found in time $O(q_i^2)$ by trying all possible pairs α_i, β_i in the range $[0, q_i]$. Testing if the above condition holds is possible using the equality testing oracle.

As was stated above, the $(\alpha_i, \beta_i) \pmod{q_i}$ can be combined using chinese remaindering to obtain integers α, β such that $P = \alpha P_1 + \beta P_2$. Since $|E_{a,b}|$ has at most $O(\log p)$ prime factors and they are all less than d the running time of this procedure is at most $O(d^2 \log p)$. This completes the proof of the theorem. $\square$

We note that the algorithm easily extends to solve BBFP in any finite field $\mathbb{F}_{p^m}$ and not just prime fields. To simplify the exposition we chose to avoid the definition of BBFP and the description of the algorithm for non prime finite fields.

3.2 A two step algorithm

Before we present the results of this section we have to explain the notion of an *algebraic algorithm.* Intuitively, a function $f : \mathbb{F}^n \to \mathbb{F}$ can be computed by an algebraic algorithm if it can be computed efficiently even when the input is given as black box field elements. Essentially this means that the function can be computed by only applying elementary arithmetic operations and equality tests to its input.

Definition 3.1 *Let $\mathbb{F}_p$ be a finite field and $f : \mathbb{F}_p^n \to \mathbb{F}_p$ some function on n inputs. We say that f can be computed in time $T(p)$ by an* algebraic algorithm *if there exists a Turing machine M satisfying the following property: for any black box representation (p, n, h, F, G, T) of the field $\mathbb{F}_p$ the machine M*

given access to the oracles F, G, T and $[]$ will run in time at most $T(p)$ and will output $[f(x_1, \ldots, x_n)]$ on input $[x_1], \ldots, [x_n]$.

in Section 2 we noted that the functions $f(x) = x^{-1} \pmod p$ and $f(x) = \sqrt{x} \pmod p$ can be computed by random polynomial time *algebraic* algorithms.

We improve the algorithm of the previous section by improving the brute force search step. The improvement relies on a certain additional assumption. Namely, we assume that the Diffie-Hellman function over the group of points of an elliptic curve can be computed by an $L_{1/3}(p)$ *algebraic* algorithm. Though this assumption is likely to be false (there are no sub-exponential algebraic algorithms for computing discrete log) the contrapositive of the theorem is quite useful. It provides a more efficient reduction from computing discrete-log to breaking Diffie-Hellman (Theorem 4.5 part (2)).

Theorem 3.4 *Suppose $DH_g(P_1, P_2)$ can be computed in time $L_{1/3}(p)$ by an* algebraic *algorithm where $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$ and g are points on an elliptic curve $E_{a,b}(p)$. Then under the smoothness assumption, BBFP in a finite field of size p can be solved in expected time*

$$L_{\frac{1}{3}}(p)^{2+o(1)} = \exp\left((2 + o(1))\sqrt[3]{\log p\ \log\log^2 p}\right) \quad .$$

Note: Theorem 3.4 is a rare example where the elliptic curve method produces an algorithm with a running time of $L_{1/3}(p)$. Such running times are usually associated with the number field sieve [20].

Proof We improve the algorithm of Theorem 3.1 by using a two step algorithm. Let $\mathbb{F}_p$ be a black box field and let $[x] \in \mathbb{F}_p$. The idea is to generate an elliptic curve $E_{a,b}$ over $\mathbb{F}_p$ with n points such that the largest prime divisor of n is $L_{2/3}(p)$. By the smoothness assumption this can be done in expected $L_{1/3}(p)^{1/3+o(1)}$ time. An important point is that the prime factors of n can be found using the elliptic curve factoring algorithm [21] in time $L_{1/3}(p)^{1+o(1)}$.

As in the proof of Theorem 3.1 we can find, in expected polynomial time, two points P_1, P_2 that generate $E_{a,b}$. Similarly we can find a $[y] \in \mathbb{F}_p$ such that the point $[P] = ([x], [y])$ is in $E_{a,b}$. Recall that it may be necessary to shift $[x]$ by a random $r \in \mathbb{F}_p$ to make sure that $[y]$ exists. We now wish to find integers α, β such that $[P] = [\alpha P_1 + \beta P_2]$. Let q be a prime divisor of n and let $k = n/q$. We show how to find a pair α_q, β_q satisfying:

$$[kP] = [\alpha_q k P_1 + \beta_q k P_2]$$

for each prime divisor q of n. As in the proof of Theorem 3.1 this is sufficient for recovering the point P and finding x.

When $q < L_{1/3}(p)^{1/2}$ the values α_q, β_q can be found by a brute force search in time $L_{1/3}(p)$. When $q > L_{1/3}(p)^{1/2}$ we may assume that q^2 does not divide n. This follows from the fact that the number of integers less than p that are divisible by a square of a prime bigger than $L_{1/3}(p)^{1/2}$ is negligible. Hence, if n is divisible by q^2 we can look for a different curve. Within an expected constant number of tries a curve satisfying the above requirement will be found.

Let G_q be the group of points of order q in $E_{a,b}$. Since q^2 does not divide n we know that G_q is a cyclic group of size q. Furthermore, since q is prime, at least one of kP_1 or kP_2 must generate this group. Henceforth we assume that kP_1 generates G_q. We wish to find an integer α_q such that $[kP] = [\alpha_q \cdot kP_1]$.

To find α_q we use the algorithm recursively. Let $R = kP_1$. Then R generates G_q. We show that the group G_q may be regarded as the black box field $\mathbb{F}_q$. An element $u \in \mathbb{F}_q$ is represented in the following way: let $uR = (x, y) \in E_{a,b}$. Then $[u]_q$ is defined to be $[u]_q = [uR] = ([x], [y])$. Note that we use the notation $[u]_q$ for black box elements in $\mathbb{F}_q$ so as not to confuse with black box elements $[x]$ in $\mathbb{F}_p$.

We show that the addition, multiplication and equality testing oracles F, G, T for the above black box representation of $\mathbb{F}_q$ can be computed in time $L_{1/3}(p)$.

Addition Given $[u]_q = ([x_1],[y_1])$ and $[v]_q = ([x_2],[y_3])$ the sum $u+v$ is represented as $[u+v]_q = [uR+vR] = ([x_3],[y_3])$. By Fact 3.2 $[u+v]_q$ can indeed be derived from $[u]_q, [v]_q$ in polynomial time in $\log p$.

Multiplication Given $[u]_q = [uR] = ([x_1],[y_1])$ and $[v]_q = [vR] = ([x_2],[y_2])$ we wish to form $[uv]_q = [uvR] = ([x_3],[y_3])$. By assumption, the function $\mathrm{DH}_R(uR, vR) = uvR$ is computable in time $L_{1/3}(p)$ by an *algebraic* algorithm. Consequently, given $[x_1],[y_1],[x_2],[y_2]$ and R the algorithm will construct $[uv]_q = ([x_3],[y_3])$ in time $L_{1/3}(p)$.

Comparison Given $[u]_q = ([x_1],[y_1])$ and $[v]_q = ([x_2],[y_3])$ we simply use the equality testing oracle for $\mathbb{F}_p$ to test that $x_1 = x_2$ and $y_1 = y_2$.

The idea is now to solve the black box field problem in the smaller field $\mathbb{F}_q$. Since $kP \in G_q$ the point $[kP]$ is a valid black box representation of an element in $\mathbb{F}_q$. We can apply the algorithm of Theorem 3.1 to solve BBFP in $\mathbb{F}_q$. The output of the algorithm is an explicit element $\alpha_q \in \mathbb{F}_q$ satisfying $kP = \alpha_q R$ as required.

Since $q < L_{2/3}(p)$ solving BBFP in $\mathbb{F}_q$ using Theorem 3.1 takes time $L_{1/2}(q)^{2+o(1)} < L_{1/3}(p)^{2+o(1)}$. Thus, the entire running time of the algorithm is bounded by $L_{1/3}(p)^{2+o(1)}$. □

4 Applications

The black box field problem arises naturally in several domains. In this section we present three applications of the algorithms presented in the previous section.

4.1 Algebraically homomorphic encryption schemes

Informally a cryptosystem is algebraically homomorphic if given the encryption of two plain-texts x, y one can construct the encryption of the plain-texts $x+y$ and xy in polynomial time. This is captured in the following definition.

Definition 4.1 *Let d, e, k be positive integers denoting the plain-text length, cipher-text length and key-length respectively. Let $\mathbb{Z}_n$ be the ring of integers modulo n where $\lfloor \log_2 n \rfloor = d$. The ring $\mathbb{Z}_n$ constitutes the set of plain-texts. Let (E, D) be an encryption scheme, i.e.*

$$D : \mathbb{Z}_n \times \{0,1\}^k \to \{0,1\}^e \quad \textit{and} \quad E : \{0,1\}^e \times \{0,1\}^k \to \mathbb{Z}_n$$

where E, D are deterministic polynomial time computable functions. Furthermore, $D(E(x, K_e), K_d) = x$ for some key pair (K_e, K_d) generated by a probabilistic polynomial time key generation algorithm. The encryption scheme is said to be algebraically homomorphic *if there exist two probabilistic expected polynomial time algorithms $A, M : \{0,1\}^e \times \{0,1\}^e \to \{0,1\}^e$ such that for all $x, y \in \mathbb{Z}_n$ and encryption keys K_e:*

$$A(E(x, K_e), E(y, K_e)) = E(x+y, K_e) \quad \textit{and} \quad M(E(x, K_e), E(y, K_e)) = E(xy, K_e)$$

As an example we note that for the RSA cryptosystem, given the encryption of two plain-texts x and y one can easily construct the encryption of xy by simply multiplying the two given cipher-texts. RSA is not known to be algebraically homomorphic even though it supports one of the required operations.

Algebraically homomorphic encryption schemes have several applications which make them desirable. Most importantly, they enable two players two perform non-interactive secure function evaluation. See [1] for the appropriate definitions. The existence of such functions was posed as an open question in [16, pp. 6–7]. Unfortunately our sub-exponential algorithm for BBFP shows that any algebraically homomorphic encryption scheme can be broken in sub-exponential time.

Theorem 4.1 *Suppose that BBFP in a finite field of size p can be solved in time $T_{BBF}(p)$. Then any algebraically homomorphic encryption scheme (D, E) over a plain-text ring of size n can be broken in expected time*

$$O\left(T_{BBF}(n) + \exp((1 + o(1))\sqrt[3]{\log n \log^2 \log n})\right)$$

Proof To simplify the exposition we assume that n is square free. This restriction can be easily lifted using methods of Pohlig and Hellman [31] which will be discussed in section 4.3 (Lemma 4.3). Since one can factor integers in expected $\exp((1 + o(1))\sqrt[3]{\log n \log^2 \log n})$ time (see [20]) it is possible to factor the plain-text ring into a direct product of finite fields: $\mathbb{Z}_n = \prod_{i=1}^{s} \mathbb{F}_{p_i}$ where the p_i are distinct primes.

Let K_e, K_d be some encryption/decryption key pair. Given $E(x, K_e)$ we wish to find x in the required time bound. For each p_i we define the black box field (p_i, e, h, A, M, T) as follows: $h(x) = D(x, K_d) \pmod{p_i}$. Notice that for $a \in \mathbb{F}_{p_i}$ the string $[a]$ can be any string w satisfying $w = E(a', K_e)$ where $a \equiv a' \pmod{p_i}$. The addition and multiplication functions A, M are simply the corresponding functions used in Definition 4.1. Hence A, M can be computed in expected polynomial time. To implement the equality testing oracle T observe that $a' \equiv b' \pmod{p_i}$ if and only if $(a' - b') * n/p_i \equiv 0 \pmod{n}$. Thus, for $a, b \in \mathbb{F}_{p_i}$ given $[a] = E(a', K_e)$ and $[b] = E(b', K_e)$ testing if $a = b$ is done by testing if $E((a' - b') * n/p_i, K_e) = E(0, K_e)$. The string $E((a' - b') * n/p_i, K_e)$ can be computed from $[a], [b]$ in expected polynomial time. We have thus shown that (p_i, e, h, A, M, T) is a black box field and the functions A, M, T can be computed in probabilistic polynomial time.

Given $E(x, K_e)$ we use the algorithm for BBFP in each of the black box fields $\mathbb{F}_{p_i}$ to recover $x \bmod p_i$. Using chinese remaindering we can now recover x. The total expected running time of this procedure is $\sum_{i=0}^{s} O(T_{BBF}(p_i)) < O(T_{BBF}(n))$. □

Theorem 4.1 can be generalized to work for more general finite commutative rings (e.g. $\mathbb{Z}_n$). We do not pursue these generalizations here. An immediate corollary of Theorem 4.1 is that the sub-exponential algorithm for BBFP (Theorem 3.1) enables one to break any algebraically homomorphic encryption scheme in sub-exponential time. This fact may also be viewed as a general tool for cryptanalysis. A cryptanalyst which is faced with a new cryptosystem might try to prove that it is algebraically homomorphic. If he succeeds then our techniques immediately give a method for breaking the system[3].

4.2 Finding roots of polynomials with low complexity

In this section we consider polynomials over the field $\mathbb{F} = \mathbb{F}_p$ for some prime p. We show that testing if a polynomial $f(x)$ of low straightline complexity (e.g. a sparse polynomial) has a root in $\mathbb{F}$ is as hard as actually finding a root.

First we define the straightline complexity of a polynomial $f(x) \in \mathbb{F}[x]$. A straight line program for $f(x)$ is a sequence of polynomials $g_0, g_1, g_2, \ldots, g_m$ such that $g_0 = 1$; $g_1 = x$ and for all $i \geq 2$ each g_i is constructed as $g_i = g_k \circ g_l$ where $k, l < i$ and $\circ \in \{+, -, *, \div\}$. The straightline complexity of $f(x)$,

[3]This does not hold for probabilistic encryption schemes. The reason is that given two cipher-texts one can not test if they are the encryption of the same plain-text. As a result, the equality testing oracle can not be implemented.

denoted $L(f)$, is the length of the shortest straight line program for $f(x)$. Examples of polynomials with low straightline complexity are sparse polynomials (i.e. polynomials whose degree is much larger than the number of their terms).

Consider the following two problems:

Root testing: given a polynomial $f(x) \in \mathbb{F}[X]$ as a straight line program determine whether it has a root in $\mathbb{F}$.

Root finding: given a polynomial $f(x) \in \mathbb{F}[x]$ that has a unique root $x_0 \in \mathbb{F}$, find the root x_0. The polynomial is given as a straight line program.

We show that root finding is reducible in subexponential time (in the size of the input) to root testing. This is somewhat surprising since one does not expect the ability to test for the existence of a root to enable one to actually find a root.

Theorem 4.2 *Let $\mathbb{F} = \mathbb{F}_p$ for some prime $p > 2$. Let $\mathcal{A}$ be an oracle that given a straight line program for a polynomial $f(x) \in \mathbb{F}[x]$ returns true or false depending on whether $f(x)$ has a root in $\mathbb{F}$ or not. Then there exists an algorithm $\mathcal{B}$ that takes as input a straightline program of length L for a polynomial $g(x) \in \mathbb{F}[x]$ that has a unique root in $\mathbb{F}$ and outputs the root of g in expected time*

$$L \cdot \exp\left((2 + o(1))\sqrt{\log p \, \log\log p}\right) \ .$$

Furthermore, the algorithm queries the oracle on polynomials whose complexity is at most $L + (\log p)^{O(1)}$.

Proof Let x_0 be the unique root of $g(x)$ in $\mathbb{F}$. Consider the following black box representation of the field $\mathbb{F}$: an element $y \in \mathbb{F}$ is represented by a straight line program for a polynomial $h(x) \in \mathbb{F}[x]$ such that $h(x_0) = y$; in our notation $[y] = h$. First we show that the black box field operations can be efficiently computed.

Addition: Given $h_1 = [y_1]$ and $h_2 = [y_2]$ the black box representation of $y_1 + y_2$ is $[y_1 + y_2] = h_1 + h_2$. Clearly $(h_1 + h_2)(x_0) = y_1 + y_2$. A straight line program for $h_1 + h_2$ can be efficiently constructed from the programs for h_1 and h_2.

Multiplication: Given $h_1 = [y_1]$ and $h_2 = [y_2]$ then $[y_1 \cdot y_2] = h_1 \cdot h_2$. Clearly $(h_1 \cdot h_2)(x_0) = y_1 \cdot y_2$.

Equality: Given $h_1 = [y_1]$ and $h_2 = [y_2]$ define $f(y) = (h_1(y) - h_2(y))^{p-1} + g(y)^{p-1}$. The straight line complexity of $f(y)$ is longer by $O(\log p)$ steps than the sum of the lengths of the programs for h_1, h_2, g. To test if $y_1 = y_2$ we use the oracle $\mathcal{A}$ to test if $f(y)$ has a root in $\mathbb{F}$. If $y_1 = y_2$ then $h_1(x_0) - h_2(x_0) = 0$ and indeed x_0 is a root of $f()$. Otherwise, $h_1 - h_2$ and g have no common root (since x_0 is the unique root of g) and hence $f()$ always evaluates to one or two. Since the field characteristic is greater than two $f()$ has no root in $\mathbb{F}$ as required.

The [] mapping: Given a $t \in \mathbb{F}$ we construct the black box representation $[t]$ of t in a trivial way. Simply set $[t] = t$. Indeed, the constant polynomial t evaluates to t at x_0.

To find the root x_0 we use our algorithm for BBFP (Theorem 3.1) applied to to the polynomial $h(y) = y$. The polynomial $h(y) = y$ is a black box representation of x_0, i.e. $[x_0] = y$. Hence, on input y the BBFP algorithm of Theorem 3.1 will output x_0 in the required time.

Observe that the BBFP algorithm only performs equality tests on polynomials whose straight line complexity is at most $L + (\log p)^{O(1)}$ where L is the complexity of g. Therefore, the oracle $\mathcal{A}$ is only queried on polynomials whose complexity is $L + (\log p)^{O(1)}$ as required. $\square$

We apply Theorem 4.2 to polynomials given as straight line programs of length $(\log p)^{O(1)}$. There is no known sub-exponential algorithm for finding roots of such polynomials (their degree can be exponential in $\log p$). The reduction above explains why there is no known sub-exponential algorithm for just testing if such a polynomial has a root.

The reduction can also be viewed as an alternative approach for finding roots of sparse polynomials. Suppose one could find a sub-exponential time (in $\log p$) algorithm for *testing* if a polynomial given as a straight line program of length $(\log p)^{O(1)}$ has a root in $\mathbb{F}$. Then our reduction gives a sub-exponential time algorithm for *finding* roots of such polynomials. This will resolve the age old open problem of finding roots of sparse polynomials of very large degree in better than exponential time.

4.3 The Diffie-Hellman protocol

The Diffie-Hellman secret key exchange protocol [14] is one of the oldest public key protocols. The protocol enables two parties to perform a secret key exchange. We briefly describe the protocol using an arbitrary finite cyclic groups G. One can take as the group G the multiplicative group $\mathbb{Z}_p^*$ of residues modulo a prime p. This group contains $p-1$ elements.

Say Alice and Bob wish to perform a secret key exchange. They agree ahead of time on some generator $g \in G$ of the group G. The generator g is made public. Both Alice and Bob secretly pick random integers $0 < a, b < |G|$. Alice sends to Bob the value g^a and Bob sends to Alice the value g^b. Both Alice and Bob can now compute the value $g^{ab} = (g^a)^b = (g^b)^a$ which is used as their secret key.

A passive eavesdropper, Eve, who listens in on the conversation hears g^a and g^b. To discover the secret key, Eve has to compute g^{ab}. Thus, we define the Diffie-Hellman function as:

$$\mathrm{DH}_g(g^a, g^b) = g^{ab} \ .$$

Note that the cyclic group G is implicit in this notation. In their original paper, Diffie and Hellman claimed that for $G = \mathbb{Z}_p^*$ computing the function $\mathrm{DH}_g(x, y)$ is hard. Many other types of groups have been suggested by various authors. Examples include the multiplicative group of residues modulo a composite number [26, 25], elliptic curves over finite fields [18, 28], the Jacobian of a hyperelliptic curve over a finite field [19] and the class group of imaginary quadratic fields [7]. In all these groups the function $\mathrm{DH}_g(x, y)$ is believed to be hard to compute.

To stress the importance of demonstrating the hardness of the Diffie-Hellman function we note that many other cryptographic protocols rely on it for their security. Some examples are the ElGamal public key cryptosystem [15], Shamir's three pass protocol [17, pp. 96–97], and the Bellare-Micali non-interactive oblivious transfer scheme [6]. See [32] for proofs of the equivalence of breaking these protocols and computing the Diffie-Hellman function.

A long standing open problem in cryptography is the question of whether computing $\mathrm{DH}_g(x, y)$ is as hard as computing discrete-log for the group G. The discrete log-function is defined as $\mathrm{Dlog}_g(g^a) = a$ where a is an integer in the range 0 to $|G|-1$. It is not difficult to see that an oracle for $\mathrm{Dlog}_g(z)$ enables one to compute $\mathrm{DH}_g(x, y)$ in polynomial time. The hard question is whether the converse holds: given an oracle for computing $\mathrm{DH}_g(x, y)$ can one compute $\mathrm{Dlog}_g(z)$ in polynomial time (in $\log |G|$)? Surprisingly, there have been very few results on this problem [13, 23, 24, 37]. Maurer [23] obtained a beautiful result showing that given a polynomial number of advice bits that depend only on $|G|$ one can compute $\mathrm{Dlog}_g(x)$ in polynomial time given an oracle for $\mathrm{DH}_g(x, y)$. Unfortunately, computing these advice bits takes exponential time. Maurer and Wolf [24] showed that in the group $\mathbb{Z}_p^*$ for certain primes p of special form one can carry out the reduction without the need for advice bits.

We show how algorithms for BBFP can be used to reduce $\mathrm{Dlog}_g(z)$ to $\mathrm{DH}_g(x, y)$. First we state an observation due to Pohlig and Hellman [31] which shows that the ability to compute discrete log in groups of prime order is sufficient for computing discrete log in arbitrary groups.

Lemma 4.3 (Pohlig-Hellman) *Let G be a finite cyclic group for which the factorization of $|G|$ is known. Suppose that for any element $g \in G$ of prime order p one can compute $Dlog_g(x)$ in time $T(p)$. Then one can compute $Dlog_h(x)$ for any element $h \in G$ in time $T(|G|)\log^{O(1)}|G|$.*

Proof Let $|G| = \prod_{j=1}^{r} p_j^{\alpha_j}$ where p_j are distinct primes. We wish to compute $\mathrm{Dlog}_g(x)$ for an element $x = g^a$ of G. The plan is to determine $a \bmod p_j^{\alpha_j}$ for all j. The chinese remainder theorem will then enable us to reconstruct a.

Let p be some prime divisor of $|G|$ and p^α be the highest power of p that divides $|G|$. Suppose we already know $b = a \bmod p^k$ for some $0 \le k < \alpha$ (initially $b = k = 0$). We show how to determine $a \bmod p^{k+1}$.

First, define $h = g^{|G|/p}$. Note that the element h has prime order and hence generates a subgroup of G of prime order. Therefore, by assumption, we can compute $\mathrm{Dlog}_h(y)$ in time $T(p)$ (for y in the group generated by h). To find $a \bmod p^{k+1}$ compute the following value: $y = (xg^{-b})^{|G|/p^{k+1}}$.

Since $a - b$ is divisible by p^k the element y can be expressed as follows:

$$y = (xg^{-b})^{|G|/p^{k+1}} = g^{(a-b)|G|/p^{k+1}} = g^{\frac{|G|}{p}\frac{a-b}{p^k}} = h^{(a-b)/p^k} \ .$$

This shows that y is in the subgroup generated by h. As was stated above we can now compute $t = \mathrm{Dlog}_h(y)$. Clearly $t = (a - b)/p^k \bmod p$. This means that $a \equiv b + tp^k \pmod{p^{k+1}}$.

Thus, given $b = a \bmod p^k$ we were able to compute $b' = a \bmod p^{k+1}$. Iterating this procedure will eventually produce $a \bmod p^\alpha$. This shows that $a \bmod p_j^{\alpha_j}$ can be found for all j. Consequently a can be found by chinese remaindering. The total running time is $\sum_i \alpha_i T(p_i)\log^{O(1)} p \le T(|G|)\log^{O(1)}|G|$. The inequality follows from the fact that T must satisfy $T(x) + T(y) \le T(xy)$ since $T(p) > \Omega(\log p)$. □

The main connection between BBFP and the security of the Diffie-Hellman protocol is explained in the following theorem. The theorem shows that if BBFP can be solved quickly then an algorithm for breaking Diffie-Hellman will give an algorithm for computing discrete-log.

Theorem 4.4 *Suppose that BBFP over the field $\mathbb{F}_p$ can be solved in time $T_{BBF}(p)$. Let G be some finite cyclic group for which the factorization of $|G|$ is known. Suppose that $DH_g(x, y)$ can be evaluated in time $T_{DH}(|G|)$ for any $g \in G$ of prime order. Then for any $g \in G$ the function $Dlog_g(x)$ can be computed in time*

$$T_{BBF}(|G|) \cdot T_{DH}(|G|) \cdot \log^{O(1)}|G|$$

Proof Let g be a generator of G. We wish to compute $\mathrm{Dlog}_g(x)$ for some $x = g^a$. First we show that when G has prime order p the theorem is immediate. The ability to evaluate $\mathrm{DH}_g(x, y)$ transforms G into the black box field $\mathbb{F}_p$. An elements $b \in \mathbb{F}_p$ is represented as $[b] \doteq g^b$. Two elements can be added $[b + c] = g^b g^c$ and multiplied $[bc] = \mathrm{DH}_g(g^b, g^c)$. Observe that $[a] = x$. Hence, the algorithm for BBFP will output a on input x. By assumption, the black box field problem can be solved in time $T_{BBF}(|G|)$. The algorithm for solving BBFP will make at most $T_{BBF}(|G|)$ calls to the multiplication oracle. Each call requires a calculation of the Diffie-Hellman function which takes time $T_{DH}(|G|)$. Therefore, the total running time is $O(T_{BBF}(|G|) \cdot T_{DH}(|G|))$. Thus, the theorem is proved when G has prime order.

When $|G|$ is not prime we make use of Lemma 4.3. Let p be a prime dividing $|G|$ and let $h = g^{|G|/p}$. The group generated by h has prime order. By assumption, $\mathrm{DH}_h(x, y)$ can be evaluated in time $T_{DH}(p)$.

Hence, by the previous paragraph, $\mathrm{Dlog}_h(y)$ can be computed in time $T_{BBF}(p) \cdot T_{DH}(p)$. Lemma 4.3 can now be applied to show that $\mathrm{Dlog}_g(x)$ can be computed in time $T_{BBF}(|G|) \cdot T_{DH}(|G|) \cdot \log^{O(1)}|G|$ for any $g \in G$. $\square$

We note that when $|G|$ is square free the theorem can be strengthened to obtain a more direct reduction from computing discrete-log to breaking Diffie-Hellman. Namely, for a fixed g an algorithm for $\mathrm{DH}_g(x, y)$ can be converted to an algorithm for $Dlog_g(x)$.

4.3.1 Security of the Diffie-Hellman protocol

Theorem 4.4 shows that algorithms for BBFP give a reduction from computing discrete-log to Breaking the Diffie-Hellman protocol. In general we apply Theorem 4.4 in the following way: suppose discrete-log can not be computed in time $T_{DL}(|G|)$ in a group G. Then the Diffie-Hellman protocol can not be broken in time $\tilde{O}(T_{DL}(|G|)/T_{BBF}(|G|))$. Thus, the faster we can solve BBFP the more secure the Diffie-Hellman protocol is.

Since the algorithms for BBFP take sub-exponential time the above approach can only be used in groups for which there is no sub-exponential time algorithm for discrete log. There are several groups which are suspected to have such a property. The groups are all constructed using a prime modulus p.

1. Let g be an element of $\mathbb{Z}_p^*$ which generates a subgroup of order $L_{1/3}(p)$. Let H_g be this subgroup. Discrete log in H_g is not known to be solvable in sub-exponential time in $|H_g|$. Notice that the best general discrete log algorithm in $\mathbb{Z}_p^*$ takes time $O(|H_g|)$. The best known discrete log algorithm for such groups is Shank's baby-step-giant-step [9, pp.] which takes $O(\sqrt{|H_g|})$ time. We therefore say that H_g is a group for which there is no known sub-exponential algorithm for discrete log. The digital signature standard [29] relies on the exponential time hardness of discrete log in H_g for its security.

2. Let $E_{a,b}$ be the group of points on an elliptic curve modulo p. The best algorithm for discrete log in such groups is Shank's baby-step-giant-step algorithm. These groups are another example where there is no known sub-exponential algorithm for discrete-log.

3. Let J be the Jacobian of a hyper-elliptic curve of genus g. When the genus g is fixed, there is no known sub-exponential algorithm for discrete log in J. Adleman, DeMarrais and Huang [2] show that when the genus is at least $\log p$ the index calculus method can be adapted to give an $L_{\frac{1}{2}}(p^{2g+1})^{1+o(1)}$ algorithm for discrete log in J.

4. Finally we mention that all sub-exponential algorithms for group $\mathbb{Z}_p^*$ rely on the index calculus method and therefore require sub-exponential space. There is no known sub-exponential time and polynomial space algorithm for computing discrete log in $\mathbb{Z}_p^*$.

The algorithms for BBFP can be used to derive hardness results for breaking the Diffie-Hellman protocol for the groups mentioned above. These results are summarized in the following theorem.

Theorem 4.5 *Let p be a prime. Under the smoothness assumption we obtain the following results:*

1. *Let G be one of the groups $E_{a,b}, J, H_g$ over $\mathbb{F}_p$ and set $s = |G|$. If discrete log can not be computed in time $L_{1/2}(s)^{3+o(1)}$ in G then the Diffie-Hellman protocol can not be broken in time $L_{\frac{1}{2}}(s)$ in G.*

2. *If discrete log can not be computed in $L_{1/3}(p)^{3+o(1)}$ time in some group $E_{a,b}$ over $\mathbb{F}_p$ then there exists some group $E_{a,b}$ in which the Diffie-Hellman protocol can not be broken by an* algebraic *algorithm in time $L_{1/3}(p)$.*

3. *If discrete log can not be computed in time $L_{\frac{1}{2}}(p)^{3+o(1)}$ and polynomial space in the group $\mathbb{Z}_p^*$ then the Diffie-Hellman protocol can not be broken in time $L_{\frac{1}{2}}(p)$ and polynomial space in $\mathbb{Z}_p^*$.*

Proof First, we may assume that the factorization of $|G|$ is known. This factorization can be found using the number field sieve algorithm whose conjectured running time is $L_{1/3}(p)^{1+o(1)}$ [20]. Part (1) follows by combining Theorem 4.4 and Theorem 3.1. Part (2) follows by combining Theorem 4.4 and Theorem 3.4. Part (3) follows by observing that the algorithm of Theorem 3.1 and the reduction of Theorem 4.4 both require only polynomial space. □

Note that part (2) of the theorem relies on a weaker assumption than part (1). Unfortunately the conclusion is also weaker since we can only prove that $\mathrm{DH}_g(x, y)$ can not be computed by an *algebraic* algorithm.

5 Black-box fields in characteristic zero

We now turn our attention to black box fields over the field of rational numbers $\mathbb{Q}$. We are provided with a black box representation of the rationals that supports the axioms of Definition 2.1. The black box field problem over $\mathbb{Q}$ can be stated as follows: given a black box representation of an integer $[x]$ between 1 and N find the integer x. The algorithm is said to be polynomial if it runs in time polynomial in $\log N$. We prove the following negative result:

Theorem 5.1 *BBFP over $\mathbb{Q}$ can not be solved in polynomial time, unless factoring integers is easy.*

By "factoring integers is easy" we mean that a non-negligible fraction of the n-bit RSA composites can be factored in polynomial time. We refer to [22] for the precise definition. To prove Theorem 5.1 we use the notion of straight line complexity of a polynomial introduced in Section 4.2. The *straight line complexity* of $f(x)$, denoted by $L(f)$, is the length of the shortest straight line program for computing f. The following theorem can be easily derived from a result due to Lipton [22].

Theorem 5.2 *Let $\{f_k(x)\}$ be a sequence of polynomials over $\mathbb{Q}[x]$ such that for any k, the polynomial $f_k(x)$ has at least $2^k/k^e$ integer roots for some integer $e > 0$. Then for any $d > 0$ and sufficiently large k we have $L(f_k) > k^d$, unless factoring is easy.*

Proof of Theorem 5.1 In a black box field the only operations allowed are additions, multiplications and comparisons. Thus, a computation in a black box field can be viewed as an algebraic decision tree [36]. That is, given an input x_0 the computation proceeds as follows: at every internal node v in the tree some polynomial $f_v(x_0)$ is evaluated. If $f_v(x_0) = 0$ the computation branches to the left child of v otherwise the computation branches to the right child. The leaves of the tree are labeled with the output.

Assume towards a contradiction that for any N there is an algorithm which solves BBFP over $\mathbb{Q}$ in time $\log^{O(1)} N$. That is, given the black box representation of an integer $0 < x \leq N$ the algorithm will find x. The algorithm defines an algebraic decision tree with N leaves. Since the algorithm is polynomial, the depth of the tree is at most $\log^d N$ for some $d > 0$. Furthermore, the straight line complexity of every polynomial f_v in the tree must satisfy $L(f_v) < \log^e N$ for some $e > 0$.

Consider the path $v_0, \ldots, v_m$ from the root v_0 to a leaf v_m in which every v_i is the right child of v_{i-1}. For $i = 0, \ldots, m$ let Z_{v_i} be the set of integer roots of f_{v_i} in the range $[1, N]$. An input x will not reach the leaf v_m if it is contained in some Z_{v_i}. Since only one integer in the range $[1, N]$ is allowed to reach the leaf v_m we know that $|\bigcup_{i=1}^{m} Z_{v_i}| = N - 1$. Hence, there must exist an i such that $|Z_{v_i}| > (N-1)/m$. Since $m < \log^d N$ we conclude that the polynomial f_{v_i} has at least $N/\log^d N$ integer roots.

Applying this argument for $N = 2^k$ where $k = 1, 2, \ldots$ we obtain a sequence of polynomials f_k where each f_k has at least $2^k/k^d$ integer roots and $L(f_k) < k^e$. This sequence contradicts the statement of Theorem 5.2. □

The proof of Theorem 5.1 is based on the fact that an algorithm for BBFP over $\mathbb{Q}$ implies the existence of a family of polynomials with exponentially many roots. Naively one could argue that when these polynomials are evaluated at a point which is not a root the result requires exponential space. This does *not* imply a hardness result for BBFP over $\mathbb{Q}$ since the black box oracles are the ones who manipulate these exponentially long integers. They do so in constant time.

The theorem shows that the black box field problem over $\mathbb{Q}$ is as as hard as factoring. In light of the results of Section 4.1 the above hardness result suggests that cryptographers who wish to construct algebraically homomorphic cryptosystems may try to construct them over fields of characteristic zero.

6 Conclusions and open problems

We defined a new problem which we call the black box field problem or BBFP for short. We demonstrated several applications of BBFP: (1) Algorithms for BBFP can be used as a cryptanalytic tool to break algebraically homomorphic cryptosystems. (2) Algorithms for BBFP give a reduction from the problem of finding roots of polynomials over finite fields to the problem of testing whether such polynomials have a root in the field. (3) Efficient algorithms for BBFP provide reductions from breaking the Diffie-Hellman protocol to computing discrete-logs. These applications demonstrate the importance of this problem.

We described two sub-exponential algorithms for solving BBFP. The first works in expected time $L_{1/2}(p)$. The second solves works in expected time $L_{1/3}(p)$ under certain (strong) assumptions. These algorithms were used to show that over elliptic curves the hardness of computing discrete-log implies the security of the Diffie-Hellman protocol. These results demonstrate an advantage of elliptic-curve cryptosystems over conventional ones. In addition we noted that for small sub-groups of $\mathbb{Z}_p^*$, such as the ones used by DSS [29], our results show that the hardness of discrete-log implies the security of the Diffie-Hellman protocol. We have also shown that assuming factoring integers is hard, BBFP can not be solved in polynomial time over the rationals.

The problem of solving BBFP over a finite field in polynomial time is still open. We briefly describe a promising approach. Schoof's algorithm [33] for counting the number of points on elliptic curves over $\mathbb{F}_p$ can be made to work over black box fields. Given a curve $y^2 = x^3 + [a]x + [b]$ over $\mathbb{F}_p$ the algorithm will output an explicit integer which is the number of $\mathbb{F}_p$-points on the curve. Given a black box element $[j]$ we construct the curve $y^2 = x^3 + [a]x + [b]$ whose j-invariant is $[j]$. The number of points on this curve provides a lot of information about j. Currently it is not known how to reconstruct j from this information. Progress in this direction will be of great interest and may yield a polynomial time algorithm for BBFP.

Acknowledgments

Our BBFP algorithm is based on the pioneering work of Ueli Maurer. It is the goal of this paper to extend Maurer's ideas and present new applications for these techniques. The authors greatly acknowledge Maurer's contributions to this area and thank him for several stimulating conversations. The authors also thank Stefan Wolf and note that he independently came up with a similar notion to our black box fields which he calls *implicit representations* [37].

References

[1] M. Abadi and J. Feigenbaum. Secure circuit evaluation: a protocol based on hiding information from an oracle. *J. of Cryptology*, 3(2):1–12, 1990.

[2] L. Adleman, J. DeMarrais, and Ming-Deh Huang. A sub-exponential algorithm for discrete logarithm over the rational subgroup of the jacobian of large genus hyperelliptic curves over finite fields. In *First Algorithmic Number Theory Symposium*, 1994.

[3] L. Babai and E. Szemerédi. On the complexity of matrix group problems i. In *Proc. 25th IEEE Symp. on Foundations of Comp. Science*, pages 229–240, 1984.

[4] E. Bach. Explicit bounds for primality testing and related problems. *Math. Comp.*, 55:355–380, 1990.

[5] E. Bach and J. Shallit. *Algorithmic number theory.* MIT press, 1996.

[6] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM J. Computing*, 13:850–864, 1984.

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

[8] E. Canfield, P. Erdös, and C. Pomerance. On a problem of oppenheim concerning "factorisatio numerorum". *J. Number Theory*, 17:1–28, 1983.

[9] H. Cohen. *A course in computational algebraic number theory.* Springer-Verlag, 1991.

[10] I. Damgård. On the randomness of legendre and jacobi sequences. In *Proc. CRYPTO 88*, pages 163–172, 1988.

[11] H. Davenport. On the distribution of quadratic residues (mod p). *J. London Math. Soc.*, pages 46–52, 1933.

[12] N. DeBruijn. On the number of positive integers $\leq x$ and free of prime factors $> y$. *Indag. Math.*, 38:239–247, 1966.

[13] B. den Boer. Diffie-hellman is as strong as discrete log for certain primes. In *Proc. CRYPTO 88*, pages 530–539, 1988.

[14] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[15] T. El-Gamal. A public key cryptosystem and a signature scheme based on the discrete logarithm. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[16] J. Feigenbaum and N. Merrirr. Open questions and summary of discussions. In *Proceedings of DIMACS workshop on Distributed Computing and Cryptography*, volume 2, pages 1–32, 1989.

[17] N. Koblitz. *A course in number theory and cryptography.* Springer-Verlag, 1987.

[18] N. Koblitz. Elliptic curve cryptosystems. *Math. Comp.*, 48:203–209, 1987.

[19] N. Koblitz. A family of jacobians suitable for discrete log cryptosystems. In *Proc. CRYPTO 88*, pages 94–99, 1988.

[20] A. Lenstra and H.W. Lenstra, editors. *The development of the number field sieve*, volume 1554 of *Lecture Notes in Mathematics.* Springer-Verlag, 1994.

[21] H. W. Lenstra. Factoring integers with elliptic curves. *Annals of Math.*, 126:649–673, 1987.

[22] R. Lipton. Straight line complexity and integer factorization. In *First Algorithmic Number Theory Symposium*, 1994.

[23] U. Maurer. Towards the equivalence of breaking the diffie-hellman protocol and computing discrete logarithms. In *Proc. CRYPTO 94*, pages 271–281, 1994.

[24] U. Maurer and S. Wolf. Diffie-hellman oracles. In *Proc. CRYPTO 96*, pages 268–282, 1996.

[25] U. Maurer and Y. Yacobi. Non-interactive public-key cryptography. In *Proc. EUROCRYPT 91*, pages 498–507, 1991.

[26] K. McCurley. A key distribution system equivalent to factoring. *J. of Cryptology*, 1(2):95–105, 1988.

[27] A. Menezes and S. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. In *Proc. 23rd ACM Symp. on Theory of Computing*, pages 80–89, 1991.

[28] V. Miller. Use of elliptic curves in cryptography. In *Proc. CRYPTO 85*, pages 417–426, 1985.

[29] National Institute for Standards and Technology. *Specifications for the digital signature standard*, August 1991. Publication XX, draft.

[30] V. Nechaev. Complexity of a determinate algorithm for the discrete logarithm. *Mathematical Notes*, 55(2):165–172, 1994.

[31] S. Pohlig and M. Hellman. An improved algorithm for computing discrete logarithms over $gf(p)$ and its cryptographic significance. *IEEE Transactions on Information Theory*, 24:106–110, 1978.

[32] K. Sakurai and H Shizuya. Relationships among the computational powers of breaking discrete log cryptosystems. In *Proc. EUROCRYPT 95*, pages 341–355, May 1995.

[33] R. Schoof. Elliptic curves over finite fields and the computation of square roots mod p. *Math. Comp.*, 44(170):483–494, 1985.

[34] V. Shoup. Lower bounds for discrete logarithms and related problems. Manuscript, 1995.

[35] J. Silverman. *The arithmetic of elliptic curves.* Springer-Verlag, 1986.

[36] M. Steele and A. Yao. Lower bounds for algebraic decision trees. *J. of alg.*, 3:1–8, 1982.

[37] S. Wolf. Diffie-hellman and discrete logarithms. Master's thesis, ETH Zurich, 1995.