Evidence that the Diffie-Hellman Problem is as Hard as Computing Discrete Logs

Jonah Brown-Cohen

1 Introduction

The Diffie-Hellman protocol was one of the first methods discovered for two people, say Alice and Bob, to agree on a shared secret key despite the fact that any of the messages sent between the two of them might be intercepted. That is, the goal of the protocol is to make it impossible to determine the shared secret key for some third party, say Eve, who sees all the communication between Alice and Bob. Let $G = \langle g \rangle$ be a cyclic group with generator g. To perform the Diffie-Hellman protocol, first Alice chooses a secret number $a < |G|$, and Bob chooses a secret $b < |G|$. Then Alice sends g^a to Bob, and Bob sends g^b to Alice. Now, Alice computes $(g^b)^a = g^{ab}$, and Bob computes $(g^a)^b = g^{ab}$. Thus, the two of them have agreed on a shared secret group element $g^{ab} \in G$.

The question then is: is it possible for the evesdropper Eve to determine g^{ab} from the communications sent between Alice and Bob? Note that there are only two messages sent between Alice and Bob, namely the two group elements g^a and g^b. Thus the problem becomes: given $\{g^a, g^b\}$ compute g^{ab}. This is the Diffie-Hellman problem, and the assumption that it is hard (in the sense that no efficient algorithm exists) is central in many cryptographic protocols. One of the reasons for this assumption has to do with the relationship of the Diffie-Hellman problem to the problem of computing discrete logarithms in a cyclic group G. Note that if it were possible to efficiently compute the discrete logarithm a of g^a, then an attacker could easily solve the Diffie-Hellman problem by first computing a from g^a, and then calculating $(g^b)^a = g^{ab}$. Thus, the Discrete Log problem is at least as hard as the Diffie-Hellman problem.

The other direction of this relationship i.e. whether the Diffie-Hellman problem is as hard as the Discrete Log problem, is a fundamental open question in cryptography. Since the Discrete Log problem is generally thought to be hard, a reduction from Discrete Log to Diffie-Hellman would give strong evidence that the Diffie-Hellman protocol is secure. One of the first steps toward such a reduction was made by den Boer in [1]. He showed that for primes p satisfying a certain condition, there is a reduction from Discrete Log to Diffie-Hellman in the group Z_p^*. The condition required was that $\varphi(p-1)$ had only small prime factors. Here $\varphi(n)$ is Euler's Totient function which counts the number of positive integers less than n that are coprime to n. This result was later generalized to all groups by Maurer in [2] using elliptic curves, and requiring a different number-theoretic assumption. In this paper, we will present Maurer's main result, along with the necessary background from number theory.

2 Computation with A Diffie-Hellman Oracle

To give a reduction from the Discrete Log Problem to the Diffie-Hellman problem, we must show that an efficient algorithm that solves the Diffie-Hellman problem can be used to efficiently solve the Discrete Log Problem. We make the concept of an efficient algorithm that solves the Diffie-Hellman problem formal in the following definition.

Definition 2.1. A *Diffie-Hellman oracle (DH-oracle)* $\mathcal{O}$ for a cyclic group $G = \langle g \rangle$ is an algorithm that given $g^a, g^b \in G$ outputs g^{ab} in time polynomial in $\log |G|$. We will write $\mathcal{O}(g^a, g^b) = g^{ab}$.

Since the DH-oracle operates on the exponent of a generator of G, we introduce some extra notation to simplify the formulas for the remainder of the paper.

Definition 2.2. For a cyclic group $G = \langle g \rangle$ define $\exp_g(x) = g^x$.

The next step toward constructing a reduction from Discrete Log to Diffie-Hellman is to understand the computational power of a DH-oracle. The following lemma demonstrates what can be computed in the exponent of a generator g by a DH-oracle.

Lemma 2.3. *Let $G = \langle g \rangle$ be a cyclic group of prime order p and let $f(x)$ be any rational function over $\mathbb{F}_p$. Then given $g^x \in G$, a DH-oracle $\mathcal{O}$ can be used to compute $g^{f(x)}$ in time polynomial in* $\log |G|$ *and the size of the rational function f.*

Proof. Note clearly $g^a \cdot g^b = g^{a+b}$, so it is possible to add exponents in G. The inversion operation in the group can be used for subtraction of exponents because

$$g^a \cdot (g^b)^{-1} = g^a \cdot g^{-b} = g^{a-b}$$

Since $\mathcal{O}(g^a, g^b) = g^{ab}$ a DH-oracle can be used to multiply exponents. To achieve arbitrary powers of exponents in G, k calls to the oracle $\mathcal{O}$ can be used for repeated squaring of the exponent to compute $\exp_g(a^{2^k})$ for any k. Then by simply writing a number n in binary as

$$n = b_0 + b_1 \cdot 2 + b_2 \cdot 2^2 + ... + b_k \cdot 2^k$$

we can compute $\exp_g(a^n)$ by simply adding up the exponents $\exp_g(a^{2^i} b_i)$ for i from 0 to k. Note that since $k = O(\log n)$ this algorithm only requires $O(k^2) = O(\log^2 n)$ calls to the oracle. Since we can now compute subtraction, addition, multiplication and arbitrary powers of the exponent, we can compute any polynomial $\exp_g(p(x))$ given input g^x.

Next note that $x^{-1} \equiv x^{p-2} \pmod p$. Since G is cyclic of order p we have

$$\exp_g(x^{-1}) = \exp_g(x^{p-1})$$

Thus, we can compute inverses of exponents in G. Thus, for any rational function $f(x) = \frac{p(x)}{q(x)}$ where p and q are polynomials we can compute

$$\exp_g(f(x)) = \exp_g(p(x) \cdot q(x)^{-1})$$

Finally, note that all the above operations require a polynomial (in the size of f) number of group operations and calls to $\mathcal{O}$. Since by assumption $\mathcal{O}$ runs in polynomial time in $\log |G|$, the function $g^{f(x)}$ can be computed in time polynomial in the size of f and $\log |G|$. $\square$

An immediate consequence of the lemma is that any function $f(x)$ that can be computed by an algorithm that uses only addition, subtraction, multiplication, division and exponentiation can be used, along with a DH-oracle, to compute $g^{f(x)}$ from g^x. Below we state a corollary of this fact that will be useful in the reduction.

Corollary 2.4. *Let $G = \langle g \rangle$ be a group with prime order p. Given an element $g^x \in G$, a DH-oracle $\mathcal{O}$ can be used to efficiently compute g^z where $z^2 \equiv x \pmod{p}$.*

Proof. As noted in [2] there exist algorithms for computing square roots mod p that only use the aforementioned operations. So by Lemma 2.3 square roots can be computed efficiently. □

Since we will be exploiting elliptic curves to give our reduction, we will also need a corollary regarding computing in elliptic curve groups using a DH-oracle.

Corollary 2.5. *Let $E(\mathbb{F}_p)$ be an elliptic curve and let $(x_1, y_1), (x_2, y_2)$ be points on $E(\mathbb{F}_p)$. Let $(x_3, y_3) = (x_1, y_1) + (x_2, y_2)$. Then given the pairs (g^{x_1}, g^{y_1}) and (g^{x_2}, g^{y_2}) a DH-oracle can be used to efficiently compute (g^{x_3}, g^{y_3}).*

Proof. As before, the algorithm for adding points on an elliptic curve only requires computing rational functions over the input coordinates. Thus by Lemma 2.3 the coordinates of a sum over an elliptic curve can be computed efficiently in the exponent by a DH-oracle. □

We now move on in the next section where we will introduce the number-theoretic assumption that we will require for the reduction to work.

3 Smooth Numbers

The central assumption for our reduction has to do with the smoothness of the order of certain elliptic curve groups. First we give the formal definition of a smooth number.

Definition 3.1. An natural number n is said to be S-*smooth* if $p < S$ for every prime factor p of n.

The above definition says that a number n is smooth if all of its prime factors are sufficiently small. An important question, that is also relevant to our reduction, regards the distribution of smooth numbers in the natural. We will first give a known estimate of the density of smooth numbers in $\mathbb{N}$, followed by a discussion of the density condition that we require.

Definition 3.2. For $n, S \in \mathbb{N}$ we define $\psi(n, S)$ to be the number of S-smooth numbers $m < n$.

In [3] the authors give several well-known estimates on $\psi(n, S)$. We provide a representative example below to give an idea of the type of bounds that exist on the density of smooth numbers.

Proposition 3.3. *Let $u = \frac{\log n}{\log S}$. Then for $S \geq (\log n)^{1+\epsilon}$ the density of S-smooth numbers is given by*

$$\psi(n, S) = u^{-u+o(u)} n$$

Unfortunately, this is not the type of estimate that will be useful for our reduction. A key component of the reduction is the existence of an elliptic curve $E(\mathbb{F}_p)$ with smooth order. We know that for a curve over $\mathbb{F}_p$ the possible group orders are given by

$$p - 2\sqrt{p} + 1 \leq \#E(\mathbb{F}_p) \leq p + 2\sqrt{p} + 1$$

Further, as noted in [2], it has been shown that for each number n in the above interval, there is a cyclic elliptic curve of order n. Unfortunately it is not known how to construct such a curve.

In the next section, the reduction we give assumes that for each (large enough) prime p there exists a cyclic elliptic curve $E(\mathbb{F}_p)$ with smooth order. Further, it requires that we explicitly construct such a curve. Of course, this is not known to be possible. However, if we assume that there exists a number n in the interval

$$p - 2\sqrt{p} + 1 \leq n \leq p + 2\sqrt{p} + 1$$

such that n is S-smooth, then we know that there exists an elliptic curve that has the desired properties. Then, all that is required for the reduction to work is some sort of external advice which tells us the coefficients A and B of the desired curve.

4 The Reduction

The main result from this section is the reduction given in [2] from the Discrete Log Problem to the Diffie-Hellman problem under the assumption that there exists an elliptic curve over $\mathbb{F}_p$ with smooth order. The key point is that we transfer the question of computing discrete logs in G into a question about discrete logs over an elliptic curve $E(\mathbb{F}_p)$. We then are able to solve this new problem efficiently because $\#E(\mathbb{F}_p)$ is smooth.

Theorem 4.1. *Let $G = \langle p \rangle$ be a cyclic group of prime order p and let $\mathcal{O}$ be a DH-oracle for G. Suppose we are given a cyclic elliptic curve $E(\mathbb{F}_p)$, such that $\#E(\mathbb{F}_p)$ is S-smooth. Then $\mathcal{O}$ can be used to compute discrete logarithms in G in time polynomial in* $\log p$ *and S.*

Proof. Throughout the proof we will make extensive use of Lemma 2.3 to compute rational functions in the exponent using the DH-oracle. Suppose we are given $g^a \in G$ and wish to determine a. Let the elliptic curve E be given by the equation $y^2 = x^3 + Ax + B$. We first wish to map a to a point on $E(\mathbb{F}_p)$. To do so, let $\exp_g(x_0) = \exp_g(a + d)$ for a randomly selected integer d. Note that x_0 is the x-coordinate of a point on the curve if and only if $z = x_0^3 + Ax_0 + B$ is a quadratic residue mod p. To test if z is a quadratic residue we simply check if

$$z^{(p-1)/2} \equiv 1 \pmod{p}.$$

Note that z is a rational function of x_0, and so by Lemma 2.3 we can use $\mathcal{O}$ to compute $\exp_g(z)$. Then, since the test for a quadratic residue is also a rational function, we can compute that x_0 is the x-coordinate of a point on the curve if and only if

$$\exp_g(z^{(p-1)/2}) = \exp_g(1) = g.$$

If x_0 does not pass this test, then we randomly select a new d and try again. Since half the elements in $\mathbb{F}_p$ are quadratic residues mod p, the expected number of times that we must retry this test is 2.

Now that we have found a z which is a quadratic residue mod p, we know that x_0 is an x-coordinate on $E(\mathbb{F}_p)$. By Corollary 2.4 we can use $\mathcal{O}$ to compute $\exp_g(y_0)$ such that

$$y_0^2 \equiv z \equiv x_0^3 + Ax_0 + B \pmod{p}$$

This yields a pair of points (g^{x_0}, g^{y_0}) such that $(x_0, y_0) \in E(\mathbb{F}_p)$. Let $P \in E(\mathbb{F}_p)$ be a generator (recall $E(\mathbb{F}_p)$ is cyclic). Then we have that for some integer n

$$(x_0, y_0) = n \cdot P$$

Note that if we can compute n, then we will be able to compute x_0. Further, since $x_0 = a+d$, we will then be able to compute a because we know d. Thus, all that remains is to find n.

Now we use the fact that $m = \#E(\mathbb{F}_p)$ is S-smooth. Let $q_1, q_2, ..., q_k$ be the prime factors of m. For the sake of simplicity we will assume that the prime factors are distinct, though the proof can easily be extended to the case where each factor occurs with some multiplicity. Note that all we really need to know is the value of $n \pmod{m}$ because $E(\mathbb{F}_p)$ is cyclic of order m. Further, note that if we know $n_j \equiv n \pmod{q_j}$ for each j, then we can compute $n \pmod{m}$ via the Chinese Remainder Theorem.

For each j note that since $n_j \equiv n \pmod{q_j}$,

$$n\left(\frac{m}{q_j}\right) \equiv n_j\left(\frac{m}{q_j}\right) \pmod{m}$$

Further, since $E(\mathbb{F}_p)$ is cyclic of order m this implies that

$$n\left(\frac{m}{q_j}\right) \cdot P = n_j\left(\frac{m}{q_j}\right) \cdot P \tag{1}$$

Thus to determine n_j we first use Corollary 2.5 to compute (g^x, g^y) from (g^{x_0}, g^{y_0}) where $(x, y) = n\left(\frac{m}{q_j}\right) \cdot P$. In particular, since the corollary allows us to add points on an elliptic curve in the exponent, we can simply use the double-then-add algorithm on (g^{x_0}, g^{y_0}) to compute (g^x, g^y) in $O(\log \frac{m}{q_j}) = O(\log m)$ steps.

Next we simply brute force search for the value of n_j using (1) to check if we have found it. That is, let

$$(x_t, y_t) = t\left(\frac{m}{q_j}\right) \cdot P$$

for each positive integer t. By Corollary 2.5 we can compute (g^{x_t}, g^{y_t}) for each subsequent value of $t > 0$ and check if $(g^{x_t}, g^{y_t}) = (g^x, g^y)$. When we find a match, we know that $t = n_j$. Note computing (g^{x_t}, g^{y_t}) at each subsequent step amounts to doing just one elliptic curve addition in the exponent. Since m is S-smooth, we have that $q_j < S$, which implies that $n_j < S$. That is, for each j we must only search over at most S values to find n_j.

As noted above, once we have found each n_j we can use the Chinese Remainder Theorem to compute n, and once we have n we can compute the desired discrete log a. Finally, it is easy to check that all the steps described above use a polynomial (in S and $\log p$) number of group operations and calls to the oracle $\mathcal{O}$. $\square$

As mentioned in the previous section, this theorem relies on both the existence and the explicit construction of a cylic elliptic curve with smooth order. If we assume that for large enough p there exists an S-smooth number n in the interval $[p - 2\sqrt{p} + 1, p + 2\sqrt{p} + 1]$, then we know such a curve exists. It remains unkown how to construct the curve, so we also require external advice giving the curve explicitly in order for the reduction to hold.

From an alternate viewpoint, suppose there are certain values of p for which it is possible to explicitly construct an elliptic curve $E(\mathbb{F}_p)$ with known order (e.g. certain super-singular curves). Then, for values of p for which these explicitly constructible curves have smooth order, the reduction can be done without any external advice. Thus, for a group G with order p equal to one of these special values, the Diffie-Hellman problem is at least as hard as the Discrete Log Problem.

5 Conclusion

It is worthwhile to note that the proof given in the previous section is just a generalization using elliptic curves of the reduction given in [1]. In that paper, den Boer simply moves the discrete log calculation for a group G of prime order p into the field $\mathbb{F}_p$. Then, under the assumption that $p-1$ is smooth, an identical application of the Chinese Remainder Theorem can be used to efficiently compute the discrete log in $\mathbb{F}_p$. Of course this only works for certain primes p.

As we showed above, in [2] Maurer generalizes den Boers result simply by passing to the elliptic curve group $E(\mathbb{F}_p)$ instead of the finite field $\mathbb{F}_p$. The relative advantage of Maurer's method is that it seems as though it may work for all group orders. If, for example, a method were discovered for efficiently constructing a cyclic elliptic curve over $\mathbb{F}_p$ with smooth order, then Maurer's theorem would immediately imply that the Diffie-Hellman problem is as hard as the Discrete Log Problem in any group. Though neither paper gives a competely general reduction from Discrete Log to Diffie-Hellman, they both should be seen as strong evidence that the Diffie-Hellman problem is hard.

References

[1] den Boer, "Diffie-Hellman is as strong as discrete log for certain primes," Crypto 1988

[2] Maurer, "Towards the equivalence of breaking the Diffie-Hellman protocol and computing discrete logarithms," CRYPTO 1994

[3] David Naccache, Igor Shparlinski, "Divisibility, Smoothness and Cryptographic Applications", http://eprint.iacr.org/2008/437.pdf