Relationships Between Diffie-Hellman and "Index Oracles"

Adam Young[1] and Moti Yung[2]

[1] Cigital, Inc.
ayoung@cigital.com
[2] Dept. of Computer Science, Columbia University
moti@cs.columbia.edu

Abstract. The Computational Diffie-Hellman problem and its decisional variant are at the heart of many cryptographic applications. Yet, their exact computational power and their relationship to the Discrete Logarithm problem and the Decision Diffie-Hellman problem (DDH) is not fully understood in all settings. In order to extend the current understanding of the problem we introduce a new decision problem that we call the Jacobi Discrete Logarithm problem. We argue that this is a natural problem and we analyze it in groups in which Decision Diffie-Hellman (DDH) is believed to be intractable. In short, the JDL problem is to return the Jacobi symbol of the exponent x in g^x. We show that JDL is random self-reducible and that it lies in between the Computational Diffie-Hellman (CDH) problem and DDH. Our analysis involves the notion of a powering oracle. Maurer and Wolf showed that a squaring oracle that returns g^{u^2} on input g^u is actually equivalent to a DH oracle. It is weaker in the sense that it can be posed as a specialized DH oracle that need only respond correctly when $u = v$. In this paper we extend the study of the relationships between Diffie-Hellman and oracles for problems which manipulate or give partial information about the index of their input. We do so by presenting a reduction that shows that a powering oracle that responds with $g^{u^a} \ mod\ p$ when given $g^u \ mod\ p$ for an unknown a that is poly-logarithmic in p, is equivalent to DH. Technically, our reduction utilizes the inverse of a particular type of Vandermonde matrix. This inverse matrix has recursively defined entries. Implications for large values of a are also given.

Keywords: Diffie-Hellman (DH), Computational Diffie-Hellman, Decision Diffie-Hellman, Discrete-Log, Public Key Cryptography, Oracles, Black-Box Reductions, JDL, LDL.

1 Introduction

The Diffie-Hellman key exchange [DH76] paved the way for public key cryptography, and is based on the presumed intractability of the Computational Diffie-Hellman problem. A multitude of cryptosystems and protocols depend on the security of DH and its decision version. The ElGamal public key cryptosystem

C. Blundo and S. Cimato (Eds.): SCN 2004, LNCS 3352, pp. 16–32, 2005.

[ElG85] was the first discrete-log based cryptosystem and its semantic security is based on the security of DDH. Also, an efficient cryptosystem which is secure against chosen ciphertext attacks was proven secure under DDH [CS98]. In addition, many key exchange protocols [MvOV99] as well as numerous deployed protocols like IPSEC and SSL designs rely on DH.

The results in this paper are motivated by the following. In typical applications a cyclic group is chosen for which it is believed that solving the Discrete Logarithm (DL) problem is intractable, and then algorithms based on the DH or DDH problems are used within the group. The security therefore rests entirely on the intractability of these problems and so a full understanding of them is mandatory.

One approach to investigate the power of DH is by comparing it to the Discrete Logarithm (or index finding) problem. Much progress has been made in showing the equivalence between the DL problem and the CDH problem [Bo88, Ma94, MW96]. Among other things, these developments show that DH is equivalent to DL whenever Euler's totient function applied to the order of G results in a smooth number. The key idea behind these results is that a DH oracle allows exponents to be multiplied, and thus enables modular exponentiation to be computed in the exponent via a square-and-multiply algorithm. This paves the way for utilizing the Pohlig-Hellman [PH78] algorithm in the exponent.

Other progress has been made by investigating the relationship between the DH oracle and oracles for related problems. It has been shown that the Diffie-Hellman problem is random self-reducible. Hence, it has been shown that given an oracle that solves DH with non-negligible probability, an algorithm exists that solves DH with overwhelming probability. Randomized algorithms which succeed with overwhelming probability will be dubbed "perfect", though clearly they are not strictly perfect since for a randomly chosen input such an oracle fails to give the correct answer with non-zero probability. Thus, the resulting perfect algorithm is called *PerfectDH* (for Perfect-Diffie-Hellman). Another approach to investigating the security of Diffie-Hellman using oracles is to analyze the hardness of computing individual bits of the DH secret vs. computing the entire DH secret. In [BV96] it was shown that an oracle that computes the $O(\sqrt{log\ p})$ uppermost (or lower-most) bits of the DH secret can be used to compute all of the bits of the Diffie-Hellman secret. A third oracle result is the work of [MW96, MW98] where the notion of a squaring DH oracle was given. A squaring DH oracle, like a DH oracle, takes as input a value (g^u, g^v) chosen uniformly at random. However, unlike a DH oracle, on a successful call it returns g^{uv} only when $u = v$. The squaring oracle therefore succeeds on far fewer inputs. It was shown that this oracle is random self-reducible and is poly-time equivalent to a DH oracle. The solution is based on the observation that $g^{(u+v)^2}$ divided by g^{u^2} and then divided by g^{v^2} yields g^{2uv}. By taking the square root, the Diffie-Hellman key g^{uv} is obtained.

This paper attempts to extend our understanding of CDH and DDH by presenting a problem that gives partial information about the index. We call it the "Jacobi Discrete Logarithm problem" (which is investigated in groups in

which Decision Diffie-Hellman is believed to be intractable). On input g^u the partial information oracle returns the Jacobi character of the index u with non-negligible advantage over guessing. It is shown that this new decision problem resides between CDH and DDH.

In the case that the order of g is a prime q this is the "Legendre Discrete Logarithm problem." Since the Legendre is computed using Euler's criterion, evaluating the Legendre in the exponent amounts to "powering" the exponent (the "power" in this case is $(q-1)/2$). This paper is therefore geared towards the study of *powering oracles*.

We then investigate the notion of powering the exponent using small powers. A reduction is given that shows that an oracle that takes g^u as input and that answers with g^{u^a} with non-negligible probability is equivalent to a DH oracle. Here a is "an unknown constant" such that $1 < a \in \mathbb{Z}$. The novelty here is a reduction that uses a class of matrices with recursively defined entries. We also discuss oacles as above that answer with a large power.

We note that in an independent earlier work, Kiltz extended the notion of a squaring oracle [Ki01] and showed (among other things) that CDH is computationally equivalent to P-DH. Here P-DH is the problem of computing $g^{P(a,b)}$ given g^a and g^b where P is a non-linear polynomial in a and b. This work is related to our powering oracle results, though technically the works take somewhat different paths.

2 Definitions

Denote by $L(a/p)$ the Legendre symbol of a with respect to the prime p. Denote by $J(a/n)$ the Jacobi symbol of a with respect to the odd integer n. The notation $ord_p(g)$ is used to denote the order of element $g \in \mathbb{Z}_p$. When working in $\mathbb{Z}_p$, the notation $log_g(y)$ denotes x such that $y = g^x \ mod \ p$. The notation $Pr[E]$ is used to denote the probability that event E occurs.

The computational Diffie-Hellman problem will now be reviewed. Let p be prime, let $g \in \mathbb{Z}_p$ be an element having order q, and let G be the group generated by g. The value $\tau = (p, q)$ encodes the group parameters. Finally, let $\mathcal{IG}(\cdot)$ denote an *instance generator*. An instance generator for G is a randomized algorithm that when given an integer n (in unary), runs in polynomial time in n and outputs some random index τ and a generator g of G_τ. Observe that for each n, the instance generator induces a distribution on the set of indices τ. Let $\mathcal{G} = \{G_\tau\}$ be a group family. For one of the reductions a slightly different group parameter will be needed. Let τ' contain the information in τ but also include the "certified" group order, i.e. the factorization of q. The following definition of Computational Diffie-Hellman (CDH) is from [Bon98].

Definition 1. *A CDH algorithm $\mathcal{A}$ for $\mathcal{G}$ is a probabilistic polynomial time (in $|\tau|$) algorithm satisfying, for some fixed $\alpha > 0$ and sufficiently large n:*

$$Pr[\mathcal{A}(\tau, g, g^a, g^b) = g^{ab}] > \frac{1}{n^\alpha}$$

where g is a generator of G_τ. The probability is over the random choice of $<\tau, g>$ according to the distribution induced by $\mathcal{IG}(n)$, the random choice of a, b in the range $[1, |G_\tau|]$ and the random bits used by $\mathcal{A}$. The group family $\mathcal{G}$ satisfies the CDH assumption if there is no CDH algorithm for $\mathcal{G}$.

Let $PerfectDH(\tau, g, g^a, g^b)$ be the same as $\mathcal{A}$ above except that it succeeds with a probability that is overwhelming in α. The Decision Diffie-Hellman problem is as follows.

Definition 2. *A DDH algorithm $\mathcal{A}$ for $\mathcal{G}$ is a probabilistic poly-time algorithm satisfying, for some fixed $\alpha > 0$ and sufficiently large n:*

$$|Pr[\mathcal{A}(\tau, g, g^a, g^b, g^{ab}) = true] - Pr[\mathcal{A}(\tau, g, g^a, g^b, g^c) = true]| > 1/n^\alpha$$

where g is a generator of G_τ. The probability is over the random choice of $<\tau, g>$ according to the distribution induced by $\mathcal{IG}(n)$, the random choice of a, b, c in the range $[1, |G_\tau|]$, and the random bits used by $\mathcal{A}$. The group family $\mathcal{G}$ satisfies the DDH assumption if there is no DDH algorithm for $\mathcal{G}$.

The perfect DDH problem is the same as $\mathcal{A}$ above except that it succeeds with a probablity that is overwhelming in α. It was shown in Proposition 1 of [St96] (see also [NR97]) that DDH and perfect DDH are equivalent in prime order subgroups. An excellent overview of the Decision Diffie-Hellman problem was given in [Bon98]. Elliptic curve groups where DDH is easy and CDH is still believed to be hard were recently shown in [JN01].

3 The Jacobi Discrete Logarithm Problem

In this section a new computational problem is introduced called the Jacobi Discrete Logarithm (JDL) problem. The following is the formal definition of the Jacobi-Discrete-Log (JDL) problem. It is in the same vein as Section 2.

Definition 3. *A JDL algorithm $\mathcal{A}$ for $\mathcal{G}$ is a probabilistic poly-time algorithm satisfying, for some fixed $\alpha > 0$ and sufficiently large n:*

$$Pr[\mathcal{A}(\tau, g, g^a) = J(a/q)] > 1/2 + 1/n^\alpha$$

where g is a generator of G_τ. The probability is over the random choice of $<\tau, g>$ according to the distribution induced by $\mathcal{IG}(n)$, the random choice of a in the range $[1, |G_\tau|]$, and the random bits used by $\mathcal{A}$. The group family $\mathcal{G}$ satisfies the JDL assumption if there is no JDL algorithm for $\mathcal{G}$.

The Perfect Jacobi-Discrete-Log (Perfect-JDL) problem is to do the same as the above, but must succeed with overwhelming probability.

Clearly, when the order q of g is prime, the problem becomes that of computing the Legendre of the exponent (i.e., $L(a/q)$). By Euler's Criterion, $L(a/q) = a^{\frac{q-1}{2}} \ mod \ q$. Taking $a = (q-1)/2$ we see that the investigation of this problem is a natural extension to studying the aforementioned powering oracles.

For the remainder of this section n will be used to denote the order of g. Hence, $n = ord_p(g) = p_1^{t_1} p_2^{t_2} ... \ p_m^{t_m}$. The smallest prime is p_1, the next smallest prime is p_2, etc.

4 The JDL Assumption Implies the Perfect-CDH Assumption

The fact that the JDL assumption implies the Perfect-CDH assumption is not hard to see. The reduction algorithm uses an oracle that solves Perfect-CDH to compute Euler's Criterion (the Legendre symbol) in the exponent.

Observe that if we can compute $J(x/p_i)$ for $1 \leq i \leq m$ where $y = g^x \ mod \ p$, then we can compute $J(x/n)$ in the standard fashion.

$$J(x/n) = \prod_{i=1}^{m} L(x/p_i)^{t_i}$$

Consider the following algorithm which assumes the existence of an oracle *Perfect-CDH* which solves the Perfect Computational Diffie-Hellman problem. Here $y = g^x \bmod p$. The algorithm outputs $g^{x^a} \bmod p$ where $a \geq 0$. Let $a >> b$ denote the bit shift right operation, i.e., the operation of throwing away the b least significant bits of a. For example, $0110 >> 1 = 011$.

EXPSQMUL(τ, q, y, a):
1. let $L = \lceil log_2(a) \rceil$
2. set $SQ[0] = y$
3. for $i = 0$ to $L - 1$ do:
4. choose $r, r_1, r_2 \in_R \mathbb{Z}_q$
5. $t =$ Perfect-CDH$(\tau, g^r, SQ[i]^{rr_1}, SQ[i]^{rr_2})$
6. $SQ[i+1] = t^{(rr_1r_2)^{-1}} \bmod p$
7. let $t = g$ and $s = a$
8. for $i = 0$ to $L - 1$ do:
9. if (s mod 2) equals 1 then
10. choose $r, r_1, r_2 \in_R \mathbb{Z}_q$
11. $t =$ Perfect-CDH$(\tau, g^r, t^{rr_1}, SQ[i]^{rr_2})$
12. $t = t^{(rr_1r_2)^{-1}} \bmod p$
13. $s = s >> 1$
14. output t and halt

$\mathcal{A}(\tau, g, y)$:
1. set $\alpha = 1$
2. for $i = 1$ to m do:
3. if t_i is odd then
4. compute $w = n/p_i$
5. compute $g_i = g^w \ mod \ p$
6. compute $y' = y^w \ mod \ p$, and let $y' = g_i^{x'} \ mod \ p$
7. compute $b = EXPSQMUL(\tau, p_i, y', (p_i - 1)/2)$
8. if $b \neq g_i$ then set $\alpha = -\alpha$
9. output α and halt

Step 7 is computed using the Perfect-CDH oracle to compute $x'^{(p_i-1)/2}$ in the exponent in a square-and-multiply fashion.

Theorem 1. *If all calls to Perfect-CDH succeed then* $\mathcal{A}(\tau, g, y) = 1$ *iff* $L(x/n) = 1$.

Proof. Assume that every call to Perfect-CDH succeeds. Clearly $\mathcal{A}$ computes the Jacobi symbol in the standard fashion. Consider the operation of $\mathcal{A}$ when t_i is odd for prime p_i. If it can be shown that $b = g_i$ iff $L(x/p_i) = 1$ then we will be done. Note that $b = g_i \Rightarrow x'^{(p_i-1)/2} \equiv 1 \ mod \ p_i$ since g_i was raised to w in step 5. But, $x'^{(p_i-1)/2} \equiv 1 \ mod \ p_i \Rightarrow x^{(p_i-1)/2} \equiv 1 \ mod \ p_i$, since $x \equiv x' \ mod \ p_i$. Finally, from Euler's Criterion it follows that $x^{(p_i-1)/2} \equiv 1 \ mod \ p_i \Rightarrow L(x/p_i) = 1$. To prove the converse, namely that $L(x/p_i) = 1 \Rightarrow b = g_i$, the contrapositive will be proven. In other words, it will be shown that $b \neq g_i \Rightarrow L(x/p_i) = -1$. Clearly, $b \neq g_i$ implies that $x'^{(p_i-1)/2}$ is not congruent to 1 modulo p_i. But the order of $x'^{(p_i-1)/2} \ mod \ p_i$ is clearly 2 hence, $x'^{(p_i-1)/2} \equiv -1 \ mod \ p_i$. But, $x'^{(p_i-1)/2} \equiv -1 \ mod \ p_i \Rightarrow x^{(p_i-1)/2} \equiv -1 \ mod \ p_i$ since $x = x' \ mod \ p_i$. From Euler's Criterion it follows that $x^{(p_i-1)/2} \equiv -1 \ mod \ p_i \Rightarrow L(x/p_i) = -1$. □

Theorem 2. *With probability greater than* $1/2 + 1/n_2^{\alpha_2}$, $\mathcal{A}(\tau, g, y) = J(x/n)$.

Proof. It may be assumed that if one or more calls to Perfect-CDH fails then $\mathcal{A}$ outputs the wrong answer. Clearly, the worst-case is when all prime powers dividing n have a power of unity, since this is the case that requires the most invocations of Perfect-CDH. In this case algorithm $\mathcal{A}$ makes at most $k = \lceil log_2(n) \rceil$ calls to Perfect-CDH. Let γ_1 denote the probability that Perfect-CDH succeeds on a random input. Hence, γ_1 is overwhelming. It follows that $\mathcal{A}$ succeeds with probability at least ${\gamma_1}^k$. It can be shown that this quantity is at least $1/2+1/n_2^{\alpha_2}$ for fixed α_2 and sufficiently large n_2. □

5 The Perfect-JDL Assumption Implies the JDL Assumption

It is trivial to prove that the JDL assumption implies the Perfect-JDL assumption. In this section the other direction will be proven. The basic idea is to randomize the problem instance by exponentiating to a random value while taking into account the Jacobi symbol of this random vlaue. Let *JDLAlg* be an oracle solving the JDL problem. Now, consider the following algorithm.

$\mathcal{A}(\tau, g, y)$:
1. if $\sqrt{n} \in \mathbb{Z}$ output "1" and halt
2. for $\ell = 1$ to L do
3. choose $r_\ell \in_R \mathbb{Z}_n$
4. compute $x_\ell = JDLAlg(\tau, g, y^{r_\ell} \ mod \ p)$ and store $J(r_\ell/n) * x_\ell$ in list ω
5. output the majority answer in list ω and halt

Denote by property 1 the well known fact that $J(ab/n) = J(a/n)J(b/n)$.

Theorem 3. *With overwhelming probability,* $\mathcal{A}(\tau, g, y) = 1$ *iff* $J(x/n) = 1$.

Proof. If $\sqrt{n} \in \mathbb{Z}$ then the Jacobi symbol of all exponents for g is unity. Hence, step 1 always outputs the correct answer when it halts. It follows from property 1 that $J(log_g(y^{r_\ell})/n) = J(r_\ell/n)J(log_g(y)/n)$. By multiplying both sides by $J(r_\ell/n)$ it follows that $J(log_g(y)/n) = J(r_\ell/n)J(log_g(y^{r_\ell})/n)$ for $\ell = 1, 2, ..., L$. Therefore, $J(r_\ell/n) * x_\ell$ is the Jacobi of $log_g(y)$ with fixed probability s_1 in iteration ℓ where $s_1 \geq 1/2 + 1/{n_1}^{\alpha_1}$ (ineq. [1]) for some fixed α_1 and sufficiently large n_1. Observe that the loop in steps 2 through 4 constitutes a series of L Bernoulli trials. Theorem 1 (Chernoff Bound - see Appendix B) therefore applies. Let $\mu = s_1 L$ (eq. [2]) and take $L/2 = (1 - \delta)\mu$ (eq. [3]). Here the random variable X is a count of the number of successful trials. It follows that $Pr[X < L/2] < e^{-(s_1 L\delta^2)/2} \leq e^{-((1/2+1/{n_1}^{\alpha_1})L\delta^2)/2}$. By combining inequality [1] with equalities [2] and [3] it follows that $\delta \geq 2/(n_1^{\alpha_1} + 2)$. From this it can be shown that $Pr[X < L/2] < 2^{-L/({n_1}^{2\alpha_1} + 2{n_1}^{\alpha_1})}$. By taking $L = ({n_1}^{2\alpha_1} + 2{n_1}^{\alpha_1})n_1$ the theorem is proved. □

An open problem is whether or not CDH and JDL are equivalent and whether or not JDL and DDH are equivalent. The relationship between JDL and the bit hardness of DH is also interesting.

6 The DDH Assumption Implies the Perfect-JDL Assumption

We say that the group DDH *assumption* implies the Perfect-JDL *assumption* since solving DDH is intractable only if solving Perfect-JDL is intractable[1].

Observe that if $2 \mid t_1, t_2, ..., t_m$ then the Jacobi symbol of all elements in $\mathbb{Z}_n$ with respect to n is unity. A straightforward application of PerfectJDL will therefore not always suffice to distinguish DDH triples from random triples. It is not hard to see that as long as one of the t_i is odd, $J(x/n) = 1$ with probability $1/2$ for a randomly chosen x. Now, observe that n must be devoid of small prime factors, otherwise DDH triples can be distinguished based on residuosity (e.g., if $2 \mid n$ then DDH is broken based on testing for quadratic residuosity which can be done in poly-time). Hence, this implication applies to subgroups in which n is odd and free of small prime factors (in many cases the group where DDH is used is a prime order subgroup for a large prime).

Assume that an oracle *PerfectJDL* exists that solves the Perfect-JDL problem. Consider the following algorithm $\mathcal{A}$ which makes calls to PerfectJDL. It will be shown that $\mathcal{A}$ solves the DDH problem. Since PerfectJDL exists iff an algorithm solving JDL exists this proof will show that the DDH assumption implies the JDL assumption. We remark that the reduction can be easily extended to handle an order which is unknown. The problem instances can be transformed into triples that are statistically indistinguishable from DDH triples/random 3-

[1] Here we adopt the language of "one assumption implies another assumption," as in [Bon98].

tuples (see [Bon98] for this as well as the randomization technique that we use). Let $X = g^x$, $Y = g^y$, and $Z = g^z$.

$\mathcal{A}(\tau, g, X, Y, Z)$:
1. choose $r, u_1, u_2, v \in_R \mathbb{Z}_n$
2. construct the triplet $(x', y', z') = (X^{rv} g^{ru_1}, Y^r g^{ru_2}, Z^{rv} Y^{ru_1} X^{rvu_2} g^{ru_1 u_2})$
3. compute $s_1 = PerfectJDL(\tau, g^r, x')$, $s_2 = PerfectJDL(\tau, g^r, y')$, and
   $s_3 = PerfectJDL(\tau, g^r, z')$
4. if $s_3 = s_1 * s_2$ then output true else output false

Theorem 4. *If $\exists$ an algorithm $PerfectJDL$ solving Perfect-JDL then $\mathcal{A}$ breaks DDH.*

The above theorem can be seen from the following. The randomization of the problem instance has the following propertyies. If the input 3-tuple is a DH triple the (x', y', z') is a DH triple. If the input 3-tuple is not a DH triple then (x', y', z') is statisticallly indistinguishable from a random 3-tuple.

With overwhelming probability all three calls to $\mathcal{A}$ will succeed. So, when the input tuple is not a DH triple it will be "caught" with probability close to 1/2. This detection will arise when $s_3 \neq s_1 * s_2$. When the input tuple is a DH triple then with overwhelming probability $s_3 = s_1 * s_2$.

Since the DDH assumption holds iff the PerfectDDH assumption holds, it follows that any algorithm solving the JDL problem can be used as an oracle to solve PerfectDDH. It has therefore been shown that the JDL problem lies in between CDH and DDH. The potential equivalence of JDL and DDH is left as an open problem.

7 Powering Oracles

In this section we give a reduction that shows that a powering oracle that responds with g^{u^a} *mod* p when given g^u *mod* p for an unknown a that is polylogarithmic in p is equivalent to DH. It is a special case of the prior independent work of [Ki01]. Our approach involves the use of a special type of Vandermonde matrix. The reduction explicitly utilizes the factorized inverse of this type of Vandermonde matrix, an inverse matrix that has recursively defined entries. We also consider the case of unknown a and a that is very large.

Let p be a large prime and let g be an element with order q. For the moment we will consider the case that q is prime. The following is a formal definition of a powering oracle.

Definition 4. *A $PoweringDH_a$ algorithm $\mathcal{A}$ for $\mathcal{G}$ is a probabilistic polynomial time (in $|\tau|$) algorithm satisfying, for some fixed $a > 1, \alpha > 0$, and sufficiently large n:*

$$Pr[\mathcal{A}(\tau, g, g^u) = g^{u^a}] > \frac{1}{n^\alpha}$$

where g is a generator of G_τ and a is poly-logarithmic in p. The probability is over the random choice of $<\tau, g>$ according to the distribution induced by $\mathcal{IG}(n)$, the random choice of u in the range $[1, |G_\tau|]$ and the random bits used by $\mathcal{A}$. The group family $\mathcal{G}$ satisfies the $PoweringDH_a$ assumption if there is no $PoweringDH_a$ algorithm for $\mathcal{G}$.

The oracle $PerfectDH_a$ is the same as $PoweringDH_a$ except that it succeeds with a probability that is overwhelming in α. It was shown by Maurer and Wolf that $PoweringDH_2$ exists iff $PerfectDH_2$ exists. The following are a few simple facts. The problem of computing $s = g^{u^a}$ given (τ, g, g^u) when a is known is random self-reducible. To see this, consider the following algorithm $M(\cdot)$. First, M chooses $r \in_R \mathbb{Z}_q$. M then computes $t = PoweringDH_a(\tau, g, y^r)$. Finally, M outputs $t^{r^{-a}}$ and halts. It is easy to see that a perfect powering oracle for a exists provided a powering oracle for a exists that succeeds with non-negligible probability.

A powering oracle for a can be implemented given a perfect DH oracle. To see this, note that the DH oracle can be used to implement a square and multiply algorithm in the exponent. For example, to implement a powering oracle with $a = 5$, the value

$$PerfectDH(\tau, g, PerfectDH(\tau, g, PerfectDH(\tau, g, y, y), \\ PerfectDH(\tau, g, y, y)), y)$$

is computed, where $y = g^u$.

We will now motivate the general solution to the problem of showing that $PoweringDH \Leftrightarrow DH$ by considering powering oracles for $a = 3, 4$. Observe that $(x+1)^3 = x^3 + 3x^2 + 3x + 1$. From this equation it is clear that if we have access to a cubing oracle, we can isolate the $3x^2$ term. Since q is prime, 3 has an inverse mod q. So, x^2 can be isolated. The goal is therefore to utilize the cubing oracle to implement a squaring oracle.

$PerfectDH_2(\tau, g, y)$:
1. compute $t = PerfectDH_3(\tau, g, yg \ mod \ p)$
2. compute $t = t/(y^3 g) \ mod \ p$
3. compute $t = t/PerfectDH_3(\tau, g, y) \ mod \ p$
4. compute $t = t^{3^{-1}} \ mod \ p$
5. output t and halt

Since given a squaring oracle, we can implement a Diffie-Hellman oracle, the above algorithm proves that given a cubing oracle we can break Diffie-Hellman. Now consider $a = 4$. Again, the goal is to implement a squaring oracle given $PerfectDH_4$. The solution is based on the expansions $(x+1)^4 = x^4 + 4x^3 + 6x^2 + 4x + 1$ and $(x-1)^4 = x^4 - 4x^3 + 6x^2 - 4x + 1$. Observe that $(x+1)^4 + (x-1)^4 = 2x^4 + 12x^2 + 2$.

$PoweringDH_2(\tau, g, y)$:
1. compute $t = PerfectDH_4(\tau, g, yg \bmod p) PerfectDH_4(\tau, g, y/g \bmod p) \bmod p$
2. compute $t = t/(PerfectDH_4(\tau, g, y))^2 \bmod p$
3. compute $t = t/g^2 \bmod p$
4. output a twelfth root of t and halt

Given these two reductions it is only natural to ask whether or not there is a general reduction for $a > 2$. This is in fact the case, as will be shown in the sequel.

7.1 Inverse of the Vandermonde Matrix

In order to show that the reduction holds for larger values of a the form of the inverse of a specific class of Vandermonde Matrices will be explored. Recall that the following is an $a+1$ by $a+1$ square matrix $D(a+1)$ called a *Vandermonde Matrix*.

$$\begin{vmatrix} 1 & t_1 & t_1^2 & \cdots & t_1^a \\ 1 & t_2 & t_2^2 & \cdots & t_2^a \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & t_{a+1} & t_{a+1}^2 & \cdots & t_{a+1}^a \end{vmatrix}$$

It is well known from Linear Algebra that the Determinant of the Vandermonde Matrix is non-zero if all the t_i's are different [Ga59, Me01] and hence that it is non-singular. The inverse therefore exists, is unique, and can be found efficiently via Gauss-Jordan. Given the inverse, the solution to the matrix equation $D(n)\overrightarrow{x} = b$ can be easily solved by matrix multiplication since $D(n)^{-1}D(n)\overrightarrow{x} = \overrightarrow{x} = D(n)^{-1}b$. However, in this paper we will only be concerned with n by n Vandermonde Matrices $M(n)$ whose (i, j) entry is i^{j-1}.

$$[M(n)]_{i,j} = i^{j-1} \tag{1}$$

For example, the value $[M(4)]_{3,2}$ equals 3 and the entire matrix for $M(4)$ is given below.

$$\begin{vmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 \\ 1 & 3 & 9 & 27 \\ 1 & 4 & 16 & 64 \end{vmatrix}$$

The reduction algorithm given in the sequel requires the use of the inverse of such matrices. However, rather than having the reduction algorithm perform Gaussian Elimination as a procedural step and rather than relying on the fact that the matrix corresponds to an interpolation operation, we have opted to utilize a recent elegant method. In [BBM02] it was shown how to factor $M(n)^{-1}$ into two matrices in which only the rightmost matrix has recursively defined entries.

$$[M(n)]^{-1} = \frac{1}{(n-1)!}V(n) \qquad V(n) = T(n)U(n)$$

Here $U(n)$ has recursively defined entries. Thus, the authors give a recurrence describing each entry in $M(n)^{-1}$. Appendix A summarizes this approach. It also shows, by fiat, that all of the entries in $V(n)$ are integers for all n.

7.2 DH-Powering Equivalence When the Order is Prime

The goal in this section is to implement an algorithm that has access to a perfect powering oracle, and that outputs g^{x^2} on input $y = g^x$. It is assumed that q is prime and that a is known where $2 < a \in \mathbb{Z}$. These assumptions will be relaxed in the next section. Using the Binomial Theorem and the inverse of $M(n)$ the general reduction can be given for $a > 2$. Recall that the Binomial Theorem states that for a positive integer n, $(x+b)^n = \sum_{k=0}^{n} \binom{n}{k} x^k b^{n-k}$. The reduction uses $PerfectDH_a(\cdot,\cdot,\cdot)$ as an oracle in an algorithm that computes $PoweringDH_2(\cdot,\cdot,\cdot)$.

The key idea behind the reduction is the following. The powering oracle is used to compute $(x+1)^a$, $(x+2)^a$,..., $(x+a)^a$ in the exponent. For instance, $g^{(x+3)^a} = PerfectDH_a(\tau, g, yg^3 \ mod\ p)$. Using the Binomial Theorem the form of each of these binomial expansions can be found. For each power of x we can define a new variable that is the power of x times the corresponding binomial coefficient. It is then clear that under the new variables, the coefficients that remain form $M(a+1)$.

$PoweringDH_2(\tau, g, y)$:
1. set $z = g$ and $I = 1$
2. for $j = 1$ to $a+1$ do:
3. $\quad V(a+1)_{a-1,j} = \sum_{k=1}^{a+1} T(a+1)_{a-1,k} U(a+1)_{k,j}$
4. $\quad b_j = PerfectDH_a(\tau, g, yz \ mod\ p)$
5. $\quad I = Ib_j^{V(a+1)_{a-1,j}} \ mod\ p$
6. $\quad z = zg \ mod\ p$
7. compute $r = a!\frac{a(a-1)}{2} \ mod\ q$
8. compute $s = r^{-1} \ mod\ q$ using the Extended Euclidean Algorithm
9. output $I^s \ mod\ p$ and halt

Theorem 5. *If $a > 2$ and all calls to $PerfectDH_a$ succeed then $PoweringDH_2$ outputs g^{x^2}.*

Proof. The resulting values $V(a+1)_{a-1,j}$ for $j = 1, 2, ..., a+1$ computed in step 3 are equal to the row in $V(a+1)$ which is third from the bottom. The loop over step 5 which computes I effectively multiplies a $1 \times (a+1)$ matrix by an $(a+1) \times 1$ matrix, which yields a single value in the exponent of g in I. The difference is that the elements in $V(a+1)_{a-1,j}$ are in $\mathbb{Z}_q$ and the elements $b_1, b_2, ..., b_{a+1}$ are in $\mathbb{Z}_p$. By performing exponentiation, matrix operations are effectively performed in $\mathbb{Z}_q$. Using the Binomial Theorem and the fact that $[M(n)]^{-1} = \frac{1}{(n-1)!}V(n)$ it

can be shown that resulting value in the exponent of g in I is $a!\binom{a}{2}x^2$. Since it was assumed that the order q of g is prime, an inverse s of $r = a!\binom{a}{2}\ mod\ q$ exists and is unique. Hence, step 8 can be efficiently computed and correctly computes the inverse of r. Since I^s is output it follows that when all calls to $PerfectDH_a$ succeed the resulting output value is g^{x^2}. □

It is straightforward to compose $PoweringDH_2$ with Maurer and Wolf's squaring algorithm to yield the stated DH result. A small numerical example of the above reduction is given in the next subsection to illustrate this algorithm.

7.3 Small Example of the Reduction

It is instructive to analyze an example for $V(n)$. The following is an example of $V(n)$ where $n = 4$. It is straightforward to verify that $(\frac{1}{3!}V(4))M(4) = I$.

$$V(4) = \begin{vmatrix} 1 & -3 & 2 & -1 \\ 0 & 3 & -6 & 2 \\ 0 & 0 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} 7 & -2 & 1 & 0 \\ -4 & 7 & -4 & 1 \\ 2 & -5 & 4 & -1 \\ -1 & 3 & -3 & 1 \end{vmatrix} = \begin{vmatrix} 24 & -36 & 24 & -6 \\ -26 & 57 & -42 & 11 \\ 9 & -24 & 21 & -6 \\ -1 & 3 & -3 & 1 \end{vmatrix} \tag{2}$$

An example will go a long way to illustrate how and why the algorithm works. Suppose we are given a cubing oracle. We would like to show how to use this oracle to implement a squaring DH oracle, and hence an oracle solving computational DH. The loop over step 3 performs the following matrix multiplication.

$$\begin{vmatrix} 0 & 3 & -6 & 2 \end{vmatrix} \begin{vmatrix} 7 & -2 & 1 & 0 \\ -4 & 7 & -4 & 1 \\ 2 & -5 & 4 & -1 \\ -1 & 3 & -3 & 1 \end{vmatrix} = \begin{vmatrix} -26 & 57 & -42 & 11 \end{vmatrix} \tag{3}$$

The algorithm then computes I as shown below.

$$I = g^{3!3x^2} = g^{3!\binom{3}{2}x^2} = b_1^{-26} b_2^{57} b_3^{-42} b_4^{11}\ mod\ p \tag{4}$$

This is effectively the following matrix multiplication in the exponent.

$$\begin{vmatrix} -26 & 57 & -42 & 11 \end{vmatrix} \begin{vmatrix} (x+1)^3 \\ (x+2)^3 \\ (x+3)^3 \\ (x+4)^3 \end{vmatrix} = 18x^2 \tag{5}$$

7.4 Performance of the Reduction Algorithm

Theorem 6. *If a is poly-logarithmic in p then $PoweringDH_2$ halts in time polynomial in log p using $a+1$ oracle calls to $PerfectDH_a$.*

Proof. Computing $V(a+1)_{a-1,j}$ in step 3 requires $a+1$ multiplications. The loop over step 3 therefore constitutes $O(a^2)$ operations. Since $a = log_2^{O(1)}\ p$, it follows that this step constitutes a total of $log_2^{O(1)}\ p$ operations. The loop over steps 4 through 6 requires $a+1$ calls to $PerfectDH_a$ requries a polynomial number of operations in *log p*. Since a is poly-logarithmic in p, computing $a!$ *mod* q in step 7 requires a poly-logarithmic number of multiplications modulo q. From this it is not hard to see that the running time is as claimed. □

The correct termination of the algorithm is based on the fact that the probability that $PerfectDH_a$ succeeds is $\gamma(\alpha)$ which is overwhelming in some the security parameter α (typically the size of the modulus p) and only fails with negligible probability. Since $PoweringDH_2$ makes $a+1$ calls to the oracle $PerfectDH_a$ it holds that since further $a+1 \leq p(\alpha)$ where $p(\alpha)$ is a polynomial in α then $PoweringDH_2$ succeeds with non-negligible probability.

7.5 Generalizations

Note that if a is not known, a value for a for which $PerfectDH_a$ succeeds with non-negligible probability can be determined. To see this note that we can invoke the oracle with a randomly chosen index (exponent) x with x known. The values $a = 2, 3, 4, ...$ and so on can be tested. It is not hard to see that this process runs in time polynomial in *log p*. If $a = 2$, then Maurer and Wolf's algorithm for a squaring oracle is performed. Otherwise, our reduction is performed.

Now we will consider the same problem as in the previous section, but generalize it to allow the order of g to be composite. The value q will still be used to denote the order of g, but in this case q may be composite. Provided that $gcd(w, q)$ is not too large an algorithm can be used to compute g^{x^2} *mod* p. Care must be taken now since the existence of unique inverses modulo the composite q is not guaranteed. Recall that the value $g^{a!x_2} = g^{a!\binom{a}{2}x^2}$ *mod* p is readily obtained. By computing the r^{th} root where $r = a!\binom{a}{2}$ *mod* q, the answer is found. The work of Scott Lindhurst can be used to analyze the cases in which we can efficiently compute the r^{th} root mod p [SL97]. The following is Proposition 8 from his thesis.

Proposition 1. *Given a cyclic group G and a degree $r = O(log^2\ |G|)$, we can compute r^{th} roots in G deterministically (assuming we are given an r^{th} power residue) using $O(log^2\ |G|\ log\ log\ |G|)$ group operations.*

7.6 Equivalence to DH When the Power Is Large

Consider an oracle that on input g^x mod p returns $g^{x^{q-a}}$ mod p with overwhelming probability. Here $a > 1$ and the order of g is the prime q where q divides $p-1$ evenly. When a is polylogarithmic in p this oracle is equivalent to Diffie-Hellman.

To see this, observe that by applying the oracle twice in succession the value $g^{x^{(q-a)^2}} = g^{x^{q^2-2aq+a^2}}$ is computed with overwhelming probability. Suppose that x generates $\mathbb{Z}_q$. Then since $q^2 - 2aq + a^2$ divided by $q-1$ results in a remainder of $a^2 - 2a + 1$ it follows that $g^{x^{(q-a)^2}} = g^{x^{(a-1)^2}}$. This yields a powering oracle for a small exponent, which in this case is $(a-1)^2$, and this has been shown to be equivalent to Diffie-Hellman. Suppose that x does not generate $\mathbb{Z}_q$. To handle this issue it is possible to first randomize x using r to enable $xr \ mod \ q$ to be a generator of $\mathbb{Z}_q$ with non-negligible probability. This randomization factor can be removed in the final result by computing $(g^{(rx)^{(a-1)^2}})^{r^{-(a-1)^2}}$.

References

[BBM02] C. Bender, D. Brody, B. Meister. Inverse of a Vandermonde Matrix. Preprint 2002 (downloaded from http://theory.ic.ac.uk/∼brody/DCB/sa6.pdf).

[Bo88] B. Den Boer. Diffie-Hellman is as strong as discrete log for certain primes. In *Advances in Cryptology—Crypto '88*, LNCS 403, pages 530–539, 1988.

[Bon98] D. Boneh. The Decision Diffie-Hellman Problem. In *Third Algorithmic Number Theory Symposium*, LNCS 1423, pages 48–63, 1998.

[BV96] D. Boneh, R. Venkatesan. Hardness of Computing the Most Significant Bits of Secret Keys in Diffie-Hellman and Related Schemes. In *Advances in Cryptology—Crypto '96*, pages 129–142, 1996.

[CS98] R. Cramer, V. Shoup. A practical public key crypto system provably secure against adaptive chosen ciphertext attack. In *Advances in Cryptology—Crypto '98*, LNCS 1462, 1998.

[DH76] W. Diffie, M. Hellman. New Directions in Cryptography. In *IEEE Transactions on Information Theory*, 22(6), pages 644–654, 1976.

[ElG85] T. ElGamal. A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In *Advances in Cryptology—Crypto '84*, pages 10–18, 1985. Springer-Verlag.

[Ga59] F. R. Gantmacher. The Theory of Matrices vol. 1. AMS Chelsea Publishing, 1959.

[GKP] R. Graham, D. Knuth, O. Patashnik. Concrete Mathematics. Chapter 6 - Special Numbers, Second Edition, Addison-Wesley, 1994.

[JN01] A. Joux, K. Nguyen. Separating Decision Diffie-Hellman from Diffie-Hellman in Cryptographic Groups. Available at `http://eprint.iacr.org/2001/003/`.

[Ki01] E. Kiltz. A Tool Box of Cryptographic Functions Related to the Diffie-Hellman Function. In *Progress in Cryptology—Indocrypt '01*, pages 339–350, 2001.

[Ma94] U. Maurer. Towards proving the equivalence of breaking the Diffie-Hellman protocol and computing discrete logarithms. In *Advances in Cryptology—Crypto '94*, pages 271–281, 1994. Springer-Verlag.

[Me01] Alfred J. Menezes. Combinatorics and Optimization 331 - Coding Theory. Handout on Vandermonde Matrices. Downloaded by http from www.cacr.math.uwaterloo.ca/∼ajmeneze/co331/handouts/vandermonde.ps.

[MW96] U. Maurer, S. Wolf. Diffie-Hellman Oracles. In *Advances in Cryptology—Crypto '96*, pages 268–282, 1996.

[MW98] U. Maurer, S. Wolf. The Relationship Bewteen Breaking the Diffie-Hellman Protocol and Computing Discrete Logarithms. In *SIAM Journal of Computing*, vol. 28, pages 1689–1721, 1999.

[MvOV99] A. J. Menezes, P. C. van Oorschot, S. A. Vanstone. *Handbook of Applied Cryptography*, CRC Press, 1999.

[NR97] M. Naor, O. Reingold. Number theoretic constructions of efficient pseudo random functions. In *Proceedings of the* 38^{th} *Symposium on Foundations of Computer Science—FOCS '97*, pages 458–467.

[PH78] S. Pohlig, M. Hellman. An improved algorithm for computing logarithms over GF(p) and its cryptographic significance. In *IEEE Trans. on Information Theory*, vol. 24, no. 1, pages 106–110, 1978.

[SL97] Scott Lindhurst. Computing Roots in Finite Fields and Groups with a Jaunt through sums of Digits. Doctoral Dissertation (advisor - Eric Bach), Chapter 3 - Extensions of Shanks Algorithm, 1997 (downloaded from http://members.aol.com/SokobanMac/scott/papers/papers.html).

[St96] M. Stadler. Publicly verifiable secret sharing. In *Advances in Cryptology—Eurocrypt '96*, pages 190–199.

A Inverse of M(n)

We are interested in solving the matrix equation $M(n)\overrightarrow{x} = b$ once and for all by obtaining the solution for each element in $M(n)^{-1}$ for all n. Having such a solution is advantageous since it removes the need to do elimination for each n. Fortunately, one such solution which is recursive in nature was pointed out in [BBM02]. We summarize these findings below.

$$[M(n)]^{-1} = \frac{1}{(n-1)!}V(n) \tag{6}$$

The above relationship was shown, and it was noted that all of the entries in $V(n)$ are in $\mathbb{Z}$. However, a clever direct way to build it (which our application may benefit from) will be given explicitly here. The authors give the following factorization of $V(n)$,

$$V(n) = T(n)U(n) \tag{7}$$

where $T(n)$ is upper-triangular. The matrix $U(n)$ is given by the following inhomogeneous recursion relation.

$$[U(n)]_{i,j} = [U(n-1)]_{i-1,j-1} - [U(n-1)]_{i-1,j} + [W(n)]_{i,j} \tag{8}$$

If $i - 1 = 0$ then $[U(n-1)]_{i-1,j-1} = [U(n-1)]_{i-1,j} = 0$. If $j - 1 = 0$ then $[U(n-1)]_{i-1,j-1} = 0$. Finally, if $j = n$ then $[U(n-1)]_{i-1,j} = 0$. The following are the initial values for the recursion relation.

$$U(1) = \begin{vmatrix} 1 \end{vmatrix} \tag{9}$$

$$U(2) = \begin{vmatrix} 1 & 0 \\ -1 & 1 \end{vmatrix} \tag{10}$$

The matrix $W(n)$ is given by the following equations when $n > 2$,

$$[W(n)]_{1,j} = (-1)^j \frac{(n-1)!}{(n-j)!(j-1)!} + \sum_{k=1}^{j} \frac{(-1)^{k+j}(k+1)^{n-1}n!}{(n-j+k)!(j-k)!} \tag{11}$$

$$[W(n)]_{2,j} = (-1)^j \frac{(n-1)!}{(n-j)!(j-1)!} \tag{12}$$

$$[W(n)]_{i,j} = 0 \quad for \quad i > 2 \tag{13}$$

The matrices $T(1), T(2), ..., T(5)$ were given along with an explanation of their general form. Below we give the closed form of each entry in $T(n)$. The closed form equation for $[T(n)]_{i,j}$ utilizes Stirling numbers of the first kind. We adopt the notation $\begin{bmatrix} n \\ k \end{bmatrix}$ of [GKP] to represent these numbers.

$$[T(n)]_{i,j} = \begin{cases} 0 & \text{if } i > j \\ (-1)^{n-1} & \text{i=1, j=n} \\ (-1)^{j-i} \begin{pmatrix} n-j+i-1 \\ n-j \end{pmatrix} \begin{bmatrix} n-1 \\ n-j+i-1 \end{bmatrix} & \text{otherwise} \end{cases} \tag{14}$$

Table 1. Stirling numbers of the first kind

n	$\begin{bmatrix} n \\ 0 \end{bmatrix}$	$\begin{bmatrix} n \\ 1 \end{bmatrix}$	$\begin{bmatrix} n \\ 2 \end{bmatrix}$	$\begin{bmatrix} n \\ 3 \end{bmatrix}$	$\begin{bmatrix} n \\ 4 \end{bmatrix}$	$\begin{bmatrix} n \\ 5 \end{bmatrix}$	$\begin{bmatrix} n \\ 6 \end{bmatrix}$
0	1						
1	0	1					
2	0	1	1				
3	0	2	3	1			
4	0	6	11	6	1		
5	0	24	50	35	10	1	
6	0	120	274	225	85	15	1

Theorem 7. *All of the entries in $V(n)$ are integers.*

Proof. From (7) it follows that we need only show that all of the entries in $T(n)$ and all of the entries in $U(n)$ are integers. It is well known that binomial coefficients are contained in $\mathbb{Z}$. It follows that every entry in $T(n)$ is an integer

due to (14). It remains to consider $U(n)$. Since $n-j = n-1-(j-1)$ it follows that (12) can be rewritten as,

$$[W(n)]_{2,j} = (-1)^j \binom{n-1}{j-1} \tag{15}$$

Therefore, $[W(n)]_{2,j} \in \mathbb{Z}$. Note that this also shows that the term on the left of (11) is always an integer. Finally, observe that in (11),

$$\frac{(-1)^{k+j}(k+1)^{n-1}n!}{(n-j+k)!(j-k)!} = (-1)^{k+j}(k+1)^{n-1}\binom{n}{j-k} \tag{16}$$

which is clearly always an integer. □

The solution to $M(n)\overrightarrow{x} = b$ is therefore given by $\overrightarrow{x} = \frac{1}{(n-1)!}V(n)b$ where $V(n)$ has integer entries.

B Review of Chernoff Bounds

When n independent trials are conducted such that each trial results in success with fixed probability p, the trials are called *Bernoulli trials*. When the probability of success is p_i in each trial for $1 \leq i \leq n$ the trails are called *Poisson trials*. The following theorem is due to Chernoff.

Theorem 8. *Let $X_1, X_2, ..., X_n$ be independent Poisson trials such that, for $1 \leq i \leq n$, $Pr[X_i = 1] = p_i$, where $0 < p_i < 1$. Then, for $X = \sum_{i=1}^{n} X_i$, $\mu = E[X] = \sum_{i=1}^{n} p_i$, and $0 < \delta \leq 1$,*

$$Pr[X < (1-\delta)\mu] < e^{-(\mu\delta^2)/2}.$$