On Diffie-Hellman Key Agreement with Short Exponents *

Paul C. van Oorschot • Michael J. Wiener

Bell-Northern Research, Box 3511 Station C
Ottawa, Ontario K1Y 4H7, Canada
{paulv,wiener}@bnr.ca

Abstract. The difficulty of computing discrete logarithms known to be "short" is examined, motivated by recent practical interest in using Diffie-Hellman key agreement with short exponents (e.g. over Z_p with 160-bit exponents and 1024-bit primes p). A new divide-and-conquer algorithm for discrete logarithms is presented, combining Pollard's lambda method with a partial Pohlig-Hellman decomposition. For random Diffie-Hellman primes p, examination reveals this partial decomposition itself allows recovery of short exponents in many cases, while the new technique dramatically extends the range. Use of subgroups of large prime order precludes the attack at essentially no cost, and is the recommended solution. Using safe primes also precludes this particular attack and allows improved exponentiation performance, although parameter generation costs are dramatically higher.

1 Introduction

Diffie-Hellman key agreement [3] allows two parties A and B to derive a common secret by communications over an unsecured channel, while sharing no user-specific keying material a priori. A prime p and element $g \in Z_p^*$ of large multiplicative order are fixed. A chooses a random integer x, $1 \leq x \leq p-2$, and sends to B the value $g^x \bmod p$ (hereafter, reduction mod p is not explicitly noted). B responds by choosing a random y, $1 \leq y \leq p-2$, and sending to A the value g^y. A and B may then respectively compute a common secret key K as $K = (g^y)^x$ and $K = (g^x)^y$. Due to the intractability of the discrete logarithm problem [10] for appropriate p, an eavesdropper is unable to compute x or y from observation of g^x and g^y, and thus unable to compute K in the same manner as A or B. It is widely believed that computing discrete logs is required for any party other than A or B to compute K, although this remains an open question [12]. For protection from active adversaries, the technique must be augmented; various authenticated key agreement protocols are available [17].

Regarding appropriate choice of the prime p, two issues are size and structure. Regarding bitsize, taking current algorithms into account and depending on the security requirement, 512 bits is typically specified as a minimum, and 1024 bits (or more) is commonly recommended and generally considered safe for most

* Feb. 19, 1996. *Proc. of Eurocrypt'96* (to appear), Springer-Verlag LNCS, May 1996.

applications. Regarding structure, it is well-known that p must be such that $p-1$ contains a large prime factor, to preclude feasibility of the discrete log algorithm of Pohlig and Hellman [14].

Efficient implementation of Diffie-Hellman key agreement entails efficient modular exponentiation, which requires efficient modular multiplication. For a modulus p of bitlength m, mod p exponentiation with m-bit exponents using the basic square-and-multiply method [7] requires about m modular squarings and $c \cdot m$ (m-bit) modular multiplies, where naively, $c = 0.5$ (on average); improvements are possible. Standard techniques [2] for each of (m-bit) multiplication mod p and modular reduction require $O(m^2)$ bit operations.

Diffie-Hellman is a preferred mechanism for generating ephemeral keys. Performance issues arise as security considerations demand moduli well beyond 512 bits in some applications. Since the cost of computing g^x depends linearly on the bitlength of x, ensuing real-time costs have motivated use of exponents x of less than full (m-bit) length, e.g. for $\lg x$ as small as 128 or 160. This is somewhat analogous to the widespread use of short public exponents such as $e = 2^{16} + 1$ in the RSA [16] public-key operation. Care is necessary to ensure such optimizations do not introduce security weaknesses. For example, precautions are necessary in some applications when using $e = 3$ in RSA [4], and using a short RSA private exponent is known to be insecure [21].

This paper examines the security implications of using short Diffie-Hellman exponents. The major conclusions are that use of random primes p combined with short exponents is generally insecure, and that the use of large prime-order subgroups is highly recommended. §2 reviews standard techniques for computing discrete logs in cyclic groups, including the case of short exponents. §3 presents a new method combining Pollard's lambda method with partial recovery of a secret exponent by a (partial) Pohlig-Hellman decomposition, allowing an alarmingly effective attack when short exponents are used and the group in question has order n of arbitrary factorization (e.g. $n = p - 1$ for random prime moduli p). §4 examines the use of short exponents with safe primes, while §5 examines use of short exponents when computations are restricted to large subgroups of prime order. Both techniques, when used appropriately, preclude the known attacks; each offers different advantages. Safe primes may allow further computational savings during exponentiation if a generator such as $g = 2$ is used, while use of prime-order subgroups are dramatically (e.g. more than an order of magnitude) less costly with respect to parameter generation (namely p). Much of the discussion applies to exponentiation-based systems beyond Diffie-Hellman, i.e. more general discrete logarithm problems.

2 Background on Discrete Logarithm Techniques

In this section, the basic methods for computing discrete logs in cyclic groups are reviewed: Shanks' method, and Pollard's more practical rho and lambda methods. The Pohlig-Hellman decomposition technique, of use in conjunction with any of these, is also reviewed. Advanced readers may omit this section.

The discrete logarithm problem for cyclic groups is as follows: given a cyclic group G of order n (i.e. having n elements), generator $g \in G$, and element $y \in G$, find x such that $y = g^x$. One such group is the multiplicative group Z_p^* of integers modulo a prime p; this group has order $n = p - 1$.

Shanks' method. Aside from exhaustive search, the simplest idea for solving this problem is Shanks' "baby-step giant-step" method [8] (p.9 and 575-576). It requires $O(n^{1/2})$ steps and the same order of space, where a step is one group operation. Define $t = \lceil n^{1/2} \rceil$. Compute $(g^t)^i$ for $1 \le i \le t$, and store the ordered pair $((g^t)^i, i)$ in a table, sorted by first component (in constant time using conventional hashing). To find the logarithm x, compute $y \cdot g^j (= g^{x+j})$ for $j \ge 0$. Stop when finding a j yielding a value stored in the table; this is guaranteed for some $j \le t - 1$. At this point, $g^{ti} = g^{x+j}$, implying $ti \equiv x + j \pmod{n}$. Then $x = ti - j \bmod n$ is the desired log. The running time is $O(t)$ steps where $t = n^r$ and $r = 1/2$. The algorithm may be generalized using $0 \le r \le 1$, requiring a one-time precomputation of $O(n^r)$ time and space, and a per-logarithm computation of $O(n^{1-r})$ time; $r = 0$ is exhaustive search, while $r = 1$ is table lookup. For example, $r = 1/3$ uses less space and more time. As overall time cannot be reduced below $n^{1/2}$ and space is typically more expensive, $r > 1/2$ is not interesting unless computing a very large number of logarithms.

Pollard rho. In practice, Pollard's *rho method* for discrete logarithms [15] is preferable to Shanks. It has similar square-root running time (heuristic, whereas Shanks is deterministic), but entirely avoids the large space requirement. While further details are omitted here (as the main focus is the lambda method), it is noted that this memoryless rho method can be parallelized with perfect linear speedup [20]; that is, essentially an r-fold speedup is possible using r processors.

Pollard lambda. A lesser-known algorithm due to Pollard, the *lambda method* [15], more affectionately called the *method for catching kangaroos*, can be used when the pursued logarithm x is known to lie within a restricted interval of width w. Given a group G of order n and known integers b and w, it finds the logarithm x of an element $y = g^x \in G$ in time $O(w^{1/2})$ and space for $O(\log w)$ group elements, provided it is guaranteed that $b < x < b + w$. If x is not found on the first iteration, the probability of which can be controlled, subsequent iterations may be run as required. The technique is as follows.

The method involves computing two sequences (trails) of points T and W (representing paths travelled by tame and wild kangaroos). T computes the sequence $y'_0, y'_1, \ldots, y'_N$, where $y'_{i+1} = y'_i \cdot g^{f(y'_i)}$ using the "random" function $f(y'_i)$ whose output takes values from a set R. At the point y'_N, T halts and "sets a trap" hoping to catch W should W land at this point during its own trail $y = y_0, y_1, \ldots, y_N$; this will occur if W's trail hits any point y'_i. The tame trail begins at $y'_0 = g^{b+w}$, and proceeds to y'_N. Note $\log_g(y'_N) = \log_g(y'_0) + d'_N$, where $d'_N = \sum_{i=0}^{N'-1} f(y'_i) \bmod n$. The wild trail begins at $y_0 = y$, and concludes when $y_M = y'_N$ for some point y_M in W's trail, at which point the logarithm x of y is computed as $x = b + w + d'_N - d_M \bmod n$. If no collision $y_M = y'_N$ occurs before d_M exceeds $w + d'_N$, the hunt is terminated with failure (W has travelled beyond the trap). The failure probability for a single iteration is controlled by parameter θ (see below), which should be set to minimize the expected overall work (balancing expected number of iterations and work per iteration).

The sequences y'_i and y_i can be viewed as deterministic paths which are stepped by random values from an integer set R of mean m; by rough analysis, each of the N points in T's trail provides an independent chance with probability $1/m$ of catching W. For $\theta = N/m$ and m large, the probability of success is $p_S = 1-(1-m^{-1})^{\theta m} \approx 1-e^{-\theta}$. For optimum performance, set $m = \alpha \cdot w^{1/2}$ for α as optimized below. Then for example, for $\theta = 4$, $p_S = 0.98$ ($\theta = 1$ gives $p_S = 0.63$) and the total expected work, $N + M$, minimized by $\alpha = 1/4$, is $O(w^{1/2})$ steps. More generally, $2\sqrt{\theta w}$ is the expected work given θ, when minimized using $\alpha = 1/(2\sqrt{\theta})$ [15]. Pollard suggests computing and storing g^s for all $s \in R$ used, and therefore choosing $|R| \ll w^{1/2}$. $R = \{2^0, 2^1, 2^2, \ldots, 2^{L-1}\}$ is one suggestion (for an appropriate bound L), with $f(y_i) = 2^j$ where $j = y_i \bmod L$.

Pohlig-Hellman decomposition. Given the prime power factorization of n (with q_v the largest prime divisor), a divide-and-conquer technique known as *Pohlig-Hellman decomposition* [14] can be used to reduce the running time for each of the Shanks, rho and lambda methods, by decomposing the original large discrete log problem into a number of smaller such sub-problems. For Shanks, the reduction results in overall time and space $O(\sqrt{q_v})$; for rho and lambda, the reduction is to such time and negligible space. The original problem is to find x, given a group G of order n (e.g. $G = Z_p^*$, $n = p-1$), g, and y where $y = g^x$. This is reduced to one of finding c_i discrete logs in a subgroup of order q_i for each prime power $q_i^{c_i}$ dividing n. Let $n = \prod_{i=1}^{v} q_i^{c_i}$ where $q_i < q_{i+1}$, q_i prime. For simplicity, consider the case $c_i = 1$ for all i (distinct prime factors); the technique is easily generalized. For a fixed i, compute $y^{n/q_i} = (g^{n/q_i})^x$. The result takes on one of q_i values, defining the simpler sub-problem of finding the discrete logarithm x_i ($= x \bmod q_i$) in a group of q_i elements generated by g^{n/q_i}. Once x_i is found for all i, the Chinese Remainder Theorem allows solution of the simultaneous congruences $x \equiv x_i \bmod q_i$, yielding $x \bmod \prod q_i$, which is $x \bmod n$ as originally desired. The overall complexity is dominated by the cost of finding x_v, the logarithm for the largest prime factor q_v.

Computing Discrete Logs of Restricted Form. While sub-exponential time index-calculus techniques (see [9]) for computing discrete logarithms in groups with additional structure are in general more powerful than those of Shanks and Pollard, the latter are typically more effective when the order n of the group factors such that a Pohlig-Hellman decomposition is possible, or when the exponent is small. This paper restricts attention to methods applicable to arbitrary cyclic groups. For a cyclic group G of order n (e.g. $n = p-1$ for Z_p^*), it follows from the discussion above that:

(i) discrete logs can be found in $O(n^{1/2})$ time and negligible space; and
(ii) time can be reduced to $O(q^{1/2})$ if q is the largest prime divisor of n.

For an appropriate bound B, call a prime factor q_i of n *small* if $q_i < B$, and call n *smooth* (more specifically: B-smooth) if all of its prime factors are small in this sense. To be of interest, B is chosen depending on the computational resources available, and for the present purposes, defined such that when n is B-smooth, discrete logarithms may feasibly be computed using Pohlig-Hellman decomposition. For non-smooth n, finding x (given y where $y = g^x$) appears difficult for random x of approximate bitlength $\lg p$.[1] In the sections listed below, the following approaches for constraining the size of exponents are examined:

§3: restricting exponents to random integers x in the range $[1, w]$;
§4: using a safe prime $p = 2q + 1$ (q prime) along with the first option; and
§5: restricting computations to a subgroup of prime order q where $q \approx w$.

3 Restricting Exponents to the Range $[1, w]$

Pollard's lambda method allows extraction of logs with running time about the square-root of the size of the interval in question. The requirement of a fixed level of security, say 2^t (i.e. t bits), defines a (not necessarily greatest) lower bound on the size of Diffie-Hellman exponents. More specifically, if exponents are limited to random integers x of bitlength w, this imposes the constraint $w \geq 2^{2t}$. However, if attacks better than the basic square-root methods exist, this bound on w fails to provide t bits of security. Indeed, Lemma 2 implies a greater lower bound, resulting from such an improved attack.

To explore the security impact of short exponents, consider the amount of information which may be obtained from a Pohlig-Hellman decomposition in this case of short exponents, for a group G of order n (e.g. $n = p - 1$ for $G = Z_p^*$). As previously, assume $n = \prod_{i=1}^{v} q_i$ where $q_i < q_{i+1}$. The task is to find x given y $(= g^x)$. For each B-smooth prime factor q_i, it is feasible to compute $x_i = x \bmod q_i$. Suppose there are $r \leq v$ such small q_i. Combine these x_i using the Chinese Remainder Theorem to recover $x \bmod B_r$ where $B_r = \prod_{i=1}^{r} q_i$. If $B_r > x$, this yields x itself, while for $B_r \leq x$ it yields $k = \lg(B_r)$ bits of information about x. The bitlength of the product of all small prime factors of n is k, and each q_i essentially leaks $\lg q_i$ bits of x. This raises two important questions. Let $\lg x = u$.

Q1: For a random prime p, what is the expected number of bits k of x leaked?
Q2: Is there an attack finding the remaining $u - k$ bits of x in $O(\sqrt{2^{u-k}})$ time?
(If p leaks k bits of x, t bits of security would then require $\lg x \geq 2t + k$.)

Related to Q1 and of more direct practical interest is the question: for a random prime p, with what probability is x fully revealed ($B_r > x$)? Both this and Q1 depend on the size of x relative to the smoothness bound B. The expectation

[1] "lg" is used to denote base-2 logarithms; "ln" denotes natural logarithms.

is that $k \approx \lg B$ (see §3.1), and thus one may expect full compromise when $\lg x \approx \lg B$, although there is considerable variation. Perhaps more significant is the affirmative answer to Q2, using a new technique (see §3.2).

3.1 Expected Size of Product of Short Factors of $p - 1$

Again let B be an upper bound such that computing discrete logs is computationally feasible in groups of prime order $q \le B$. Consider first, for a random prime modulus p, the expected bitlength of the product B_r of all "small" primes $p_i < B$ which divide $p - 1$. (More generally, the question relates to divisors of a group order n.) Each such prime contributes $\lg p_i$ bits to $k = \lg B_r$, and p_i divides a random number with probability $1/p_i$ (refined further below). The expected bitlength of B_r can thus be approximated as:

$$k \;\approx\; \sum_{p_i \le B} \frac{\lg p_i}{p_i} \;.$$

This sum may be further approximated by summing, rather than over all primes p_i, over all integers $i \ge 2$ weighted by the probability i is prime (estimated as $1/\ln i$ by the Prime Number Theorem). Then, since $\sum_{i=2}^{B} i^{-1} \approx \ln B$,

$$k \;\approx \sum_{i=2}^{B} \frac{\lg i}{i} \cdot \frac{1}{\ln i} \;= \sum_{i=2}^{B} \frac{1}{i}\frac{1}{\ln 2} \;\approx \lg B \;.$$

A slightly more precise estimate results by replacing $1/p_i$ by $1/(p_i - 1)$, justified as follows. Of interest is whether $p_i | p - 1$. Since p_i does not divide p, $p \bmod p_i \ne 0$ and thus $p - 1 \bmod p_i \ne -1$, leaving only $p_i - 1$ (not p_i) possible residue classes. Moreover, $1/(p_i - 1)$ may itself be refined to $p_i/(p_i - 1)^2$, to account for divisors p_i of higher multiplicity. Numerical summation of this refinement, given in Lemma 1, supports the estimate above. (While such a result is known, and may be derived rigourously using advanced number theory, a heuristic derivation from first principles as given above is considered appealing.)

Lemma 1. *For a random prime p and a fixed bound B (see above), the expected bitlength $k = \lg(B_r)$ of the product of all prime divisors $p_i \le B$ of $p - 1$ is*

$$k \;\approx \sum_{p_i \le B} \lg p_i \left(\frac{p_i}{(p_i - 1)^2} \right) \;\approx \lg B + \; C_1$$

where $C_1 \approx 0.94$, and $C_1 < 1.0$ for $B > 2^{10}$.

This answers Q1 (but see also Table 2); note that k is independent of $\lg p$. Table 1 provides supporting results for a small sample of random primes p, indicating the average, minimum, and maximum bitlengths of the product of all B-smooth prime divisors of $p - 1$. Note the large deviation from the mean, and that values B in Table 1 are relatively small.

The cryptographic significance of Lemma 1 is as follows. An adversary with sufficient computational resources to compute discrete logs in cyclic groups of order up to 2^s implies $B \approx 2^s$. One then expects about $k = s$ bits of an exponent x are leaked by a "random" prime modulus p (revealing x entirely if $\lg x < s$).

B	$\lg(B_r)$ for 100 primes $p \approx 2^{1024}$		
	mean	min	max
2^{16}	16.4	1.0	47.2
2^{32}	31.3	1.0	112.2
2^{48}	52.1	2.6	194.4
2^{64}	65.0	2.6	243.1

Table 1. Smoothness of $p-1$ relative to B. $B_r = \prod q_i^{c_i}$ where $q_i \le B$, $q_i^{c_i} | p-1$.

3.2 Lambda Method Restricted to a Strategic Interval

A new technique for computing discrete logs is now presented, which is computationally feasible when exponents x are bounded to $x \approx 2^u$ for (in)appropriate u. The technique first recovers what information about x may be obtained from a partial Pohlig-Hellman decomposition (say k bits), and then employs the lambda method to recover the remaining bits. The combined technique runs in time $O(2^{(u-k)/2})$. This answers Q2 in the affirmative.

Let $y = g^x$ be an element of a group G of order n, with the usual task to determine x. Isolate the smooth factors of n and write $n = zQ$ where $z = B_r$ (see §3) is the product of smooth factors, and has bitlength approximately k. Compute V where $V = x \bmod z$, by a partial Pohlig-Hellman decomposition (see §2). Write $x = Az + V$, where $0 \le V < z$ with A as yet unknown. Then $y = g^x = g^{Az+V}$. Now $A \le x/z$ implies $A \in [0, 2^c]$, where $c \approx u - k$ bits of x remain unknown after finding V. Compute g^V and $y^* = y/g^V = g^{Az} = h^A$ where $h = g^z$ is known. Consider the new problem of finding the discrete logarithm A (relative to h) of y^*, given y^* and the base h (in place of g). Use the lambda algorithm (see §2) to find A. Using h in place of g here restricts the trail points to a subset of cardinality about 2^c. As required in the lambda method, the log to be found (here A in place of x) is known to lie in a restricted range $[b, b+w] = [0, 2^c]$ of width $w = 2^c$. The expected running time to find A is thus $O(\sqrt{2^c})$. Knowing z and V, this allows computation of $x = Az + V$.

An example with concrete parameters is given for clarity. Let $G = Z_p^*$ and $n = p - 1$ with a 1024-bit prime p and 160-bit x (so $u = 160$). Suppose $k = 100$ bits of x are recovered using Pohlig-Hellman (so $\lg z \approx 100$). Then $\lg A \approx 60$ ($c = 60$), and although h generates a subgroup of order $\approx 2^{1024-100}$, recovery of A by the lambda method is feasible, in $O(2^{30})$ steps, because A is known to lie in the interval $A \in [0, 2^{60}]$ of width $w = 2^{60}$. Note that while the Shanks method is also easily adapted to restricted intervals, it is much more costly than the lambda due to memory requirements. Furthermore, the rho method is not feasible – direct use results in running time which is square-root but in a group of cardinality 2^{924}, and adaptations remain far inferior to the lambda method.

In summary, the technique allows an attack on Diffie-Hellman (and similar exponentiation-based systems) which has running time (in number of group operations) significantly better than square root of the exponent space size (e.g.

$\sqrt{2^{160}} = 2^{80}$). Instead, it is the larger of the square root after this is first reduced to account for leaked bits recoverable by partial Pohlig-Hellman decomposition (e.g. $\sqrt{2^{160-100}} = 2^{30}$), and the square root of the largest prime factor of n used in this decomposition. This establishes the following result.

Lemma 2. *The security of discrete exponentiation in a group of order n using exponents $x \approx 2^u$ is at most $\frac{1}{2} \cdot \max(u-k,\ \lg(q_r))$ bits, where k is the number of bits of x feasibly recoverable by a partial Pohlig-Hellman decomposition (using a partial factorization of n), and q_r is the largest prime factor of n used therein.*

In other words, the running time is $O(\max(\sqrt{2^{u-k}},\ \sqrt{q_r}))$. By Lemma 1, for $G = Z_p^*$ with random primes p, one expects $k \approx \lg B$ for a smoothness bound B.

At present, it seems dangerous to assume other than that an analogous general result holds, when the exponent space is restricted to any arbitrary set S of cardinality 2^u. Related to this, the rho algorithm is adaptable to the case where exponents x are constrained by bounding their Hamming weight. For example, Heiman [5] describes how to do so using Shanks' method (the rho method would furthermore remove memory requirements). We are aware of no "faster than square-root" method on S in this case; the above combined method does not appear to apply directly. Of related interest, Yacobi [22] has sketched a technique for improving exponentiation performance (saving multiplications but not squarings) based on use of exponents which are compressible in the Ziv-Lempel sense.

Table 2 provides insight into the practical implications of Lemma 2. It was constructed by generating 100 random 1024-bit primes p (as per Table 1), and partially factoring each using an implementation of the elliptic curve factorization method, specifically tuned and run to find prime factors up to 85 bits in length (thus with high probability extracting all 80-bit factors). This allowed determination of k for use in the attack of Lemma 2.

Table 2 gives empirical values for the percentage of cases this attack succeeds for exponents $x \approx 2^u$, assuming attackers capable of 2^c total modular multiplications (roughly corresponding to an ability to take logs in groups of order $\approx 2^{2c}$). For example, for 32% of these 1024-bit primes, an adversary capable of 2^{40} multiplications mod p would find a 160-bit logarithm x. An ideal method accelerating exponentiation via short exponents would have, in Table 2, for the best attack, the entry "100%" only for (roughly) $u \le 2c$, and "0%" for all $u > 2c$; this appears to be the case for safe primes (§4) and prime-order subgroups (§5).

adversary	exponent bitlength u						
power 2^c	64	96	128	160	192	224	256
2^{24}	81%	45%	18%	10%	1%	0%	0%
2^{32}	100%	68%	38%	25%	12%	2%	1%
2^{40}	100%	84%	55%	32%	22%	7%	2%

Table 2. Sample success rate of discrete log attack in 2^c steps, for 1024-bit primes.

Prior to this result, it has been suggested that using a 160-bit exponent x provides 80 bits of security, for primes p which are not "Pohlig-Hellman weak". It is now seen that even "partial" Pohlig-Hellman weakness may significantly reduce attack times. Note the power of the divide-and-conquer nature of the attack: if $n = p - 1$ contains, for example, divisors whose bitlengths sum to $u/2$, a u-bit logarithm can be recovered by determining $u/2$ bits through a partial Pohlig-Hellman decomposition in $O(2^{u/4})$ (and typically less than $2^{u/4}$) steps, and $u/2$ bits through subsequent use of the lambda method on the appropriate subproblem in a further $O(2^{u/4})$ steps, giving an overall cost of $O(2^{u/4})$ steps. Previously, the best attack would require the minimum of the time to run either Pohlig-Hellman or a square-root method on the entire problem, i.e. $O(2^{u/2})$.

4 Use of Short Exponents with Safe Primes

Prior uncertainty about the security implications of using arbitrary prime moduli p with short exponents x has motivated (e.g. [6, 1]) the use of *safe primes* p, of the form $p = 2q + 1$ where q is also prime. This precludes the attack of Lemma 2 because a partial Pohlig-Hellman decomposition yields only a single bit of information about an exponent x. Against this attack, it appears safe to use exponents $x \approx 2^u$, with $u = 2t$ determined such that $O(2^t)$ operations are computationally infeasible (perhaps $t = 80$ for commercial security).

It should be emphasized, however, that when using short exponents, *it is not the single very large prime q that provides protection against Lemma 2*, but rather that the total bitlength of smooth factors is $k = 1$; this differs significantly from the situation for an ordinary Pohlig-Hellman attack against a full-length exponent. For example, $p = Rq + 1$ with q very large and R relatively small, e.g. $\lg R = r = 20$ bits, nonetheless leaks 20 bits of information about an exponent x (of bitlength $2t$ say). This reduces security from t to $t - (r/2)$ bits, which for short exponents (t small) may bring an attack per Lemma 2 within range.

An advantage of using safe primes $p = 2q + 1$ is that all group elements of Z_p^* other than ± 1 are then known to have order either q or $2q$. Consequently $g = 2$ is known to be a generator for the full group of $2q$ elements if 2 is a quadratic non-residue mod p (in this case one bit of a logarithm x is available by a partial Pohlig-Hellman decomposition), and a generator for the subgroup of q elements when 2 is a quadratic residue (in this case, $g = 2$ does not leak even one bit).[2] In modular exponentiation using base $g = 2$, the cost of the modular multiplies (but not squarings) effectively disappears. Ordinarily, multiplies are more costly than squarings, although the number of multiplies is substantially less. In efficient implementations, this number is reduced further. All points considered, one may expect use of $g = 2$ to result in modular exponentiation performance improved by 20% overall, vs. an arbitrary generator g. As the main motivation for use of short exponents is improved performance, this is significant.

[2] A more important point is that use of a generator for the subgroup of order q corresponds to a prime-order subgroup as per §5 (albeit the maximal proper subgroup), which avoids the attack detailed in the last paragraph of §4.

One drawback, however, is the computational cost of finding safe primes. An obvious (but inefficient) method to find an m-bit safe prime p is to generate and search some space of candidate primes $q = q_i$ of $m-1$ bits, verify that q is prime, and then verify the candidate $p_i = 2q_i + 1$ is also prime. While the density of primes in search sequences $2q_i + 1$ differs somewhat from that of the density of primes among all integers, the latter is $1/\ln p$ (e.g. 1 in $710 \approx \ln 2^{1024}$ for 1024-bit integers). A standard optimization is to filter candidate primes using small-prime sieve techniques. But even advanced techniques require some form of double search, generating many primes q_i before a safe prime p emerges. In contrast, the cost of parameter generation in the alternative of using prime-order subgroups (§5) is essentially that of generating a single candidate 1024-bit prime. Use of safe primes thus appears to introduce a substantial penalty, with parameter generation time increased by a factor roughly equal to the number of trials required to generate an $m-1$ bit prime q_i.

A drawback of more general concern for basic forms of Diffie-Hellman key agreement is a potentially fatal protocol attack (rather than an algorithmic attack on exponentiation itself) which, although observed by others (including S. Vanstone), appears to not yet be widely recognized. The attack applies also for safe primes, except when the large prime-order subgroup thereof is used (as noted above). Note that if α generates Z_p^* where $p = 2q + 1$, then $\beta = \alpha^q = \alpha^{(p-1)/2} = -1$ is an element of order 2. If A and B respectively send each other unauthenticated ephemeral exponentials α^x and α^y (as in §1), an active intruder may substitute $(\alpha^x)^q$ for the first, and $(\alpha^y)^q$ for the second. The shared key computed by both parties is then $K = \alpha^{xyq} = \beta^{xy}$, which is $+1$ or -1 depending on the parity of xy. (The attack generalizes directly for $p = Rq+1$, in which case K takes on one of R values from the group of order R generated by $\beta = \alpha^q$; for appropriately small R, K may thus be easily found by the intruder by exhaustive trial.) This again motivates the use of prime-order subgroups (§5).

5 Restricting Computations to Prime-order Subgroups

To meet the objective of improved running time through use of short exponents, the recommended alternative is to employ an idea used in Schnorr's signature scheme [18] and in DSA [19], to guarantee that computations are carried out in a large subgroup of cardinality $q \approx 2^u$ where q is prime (a prime-order subgroup). For t bits of security, set $u = 2t$ (e.g. $u = 160$ is used in DSA). If $g \in Z_p^*$ is a generator, and q divides $p-1$, then $\beta = g^{(p-1)/q}$ generates a subgroup of order q. Using β in place of g as a base for exponentiation, all exponents x may be reduced modulo q. Such an element β of order q may be found as follows. Select a random element $h \in Z_p^*$ and compute $\beta = h^{(p-1)/q}$; repeat until $\beta \neq 1$. (While it suffices to first find a generator g and then use $g = h$ with guaranteed success, finding such a g is not mandatory, and moreover requires knowledge of the factorization of $p-1$.)

The expected number of repetitions to find $\beta \neq 1$ is $q/(q-1) \approx 1$, established as follows. A trial fails if the order of h divides $Q = (p-1)/q$; otherwise $h^{(p-1)/q}$

has order q (success). Since exactly Q elements have order which divides Q, the proportion of elements h which fails is $Q/(p-1) = 1/q$.

The use of prime-order subgroups is by far preferable to using random primes (§3) and attempting to increase the exponent bitlength from $2t$ to $2t + t'$ to compensate for divisors of $p-1$. The latter not only increases exponentiation time, but an appropriate value of t' varies with p, can only be determined with certainty by factoring $p-1$, and with small probability may be quite large. The use of prime-order subgroups has the advantage over safe primes that prime generation does not require guaranteeing a large prime divisor $q = (p-1)/2$; only a prime divisor $q \approx 2^{2t}$ is required. Since typically $2t \ll \lg p$ (e.g. 160 vs. 1024), the cost of constructing, by well-known techniques [13], a prime p such that $p-1$ has a $2t$-bit prime divisor q is little more than for finding a random prime p. In addition, the use of prime-order subgroups precludes the attack discussed in the last paragraph of §4.

6 Concluding Remarks

Although the details differ, Pollard's lambda method can be parallelized analogously to the rho method [20]. Thus all aspects of the short-exponent discrete log attack of Lemma 2 can be parallelized, and the task can be distributed over large collections of computing platforms [11], with small memory requirements.

For a security requirement of t bits when using secret exponents x in Diffie-Hellman exponentiation (mod p) and similar systems, use of random primes p together with $\lg x \approx u$ is clearly insecure for $u = 2t$ (and even larger). Susceptibility to attacks based on a partial Pohlig-Hellman decomposition (stemming from divisors of $p-1$) is significant when using short exponents x. In this case, random primes p should not be used.

The results herein firmly establish that some measure must be taken to ensure security is preserved when using short exponents for Diffie-Hellman exponentiation. Aside from precluding strict Pohlig-Hellman attacks, this requirement was hitherto folklore, which has nonetheless apparently helped motivate the use of safe primes in practical protocols such as Photuris [6] and SKIP [1]. We are aware of no effective attacks against the combined use of (appropriately) short exponents with computations restricted to (sufficiently) large prime-order subgroups. Use of prime-order subgroups allows more efficient generation of ephemeral Diffie-Hellman parameters. In contrast, finding safe primes p introduces considerable expense, although pre-computing a safe prime off-line and using $g = 2$ as a base allows improved real-time performance. However, in this case, it is recommended that the safe prime be one for which $g = 2$ generates a large prime-order subgroup as well.

References

1. A. Aziz, "Simple Key Management for Internet Protocols (SKIP)", Internet Draft (work in progress), draft-ietf-ipsec-skip-04.txt, November 1995.

2. A. Bosselaers, R. Govaerts, J. Vandewalle, "Comparison of three modular reduction functions", *Crypto'93*, Springer-Verlag LNCS 773, pp.175-176.
3. W. Diffie, M. Hellman, "New directions in cryptography", *IEEE IT-22* (1976) pp.644-654.
4. J. Hastad, "On using RSA with low exponent in a public-key network", *Crypto'85*, Springer-Verlag LNCS 218, pp.403–408.
5. R. Heiman, "A note on discrete logarithms with special structure", *Eurocrypt'92*, Springer-Verlag LNCS 658, pp.454-457.
6. P. Karn, W.A. Simpson, "The Photuris session key management protocol", Internet Draft (work in progress), draft-ietf-ipsec-photuris-06.txt, October 1995.
7. D.E. Knuth, *The Art of Computer Programming, vol.2: Seminumerical Algorithms*, 2nd edition, Addison-Wesley, 1981.
8. D.E. Knuth, *The Art of Computer Programming, vol.3: Sorting and Searching*, Addison-Wesley, 1973.
9. B.A. LaMacchia, A.M. Odlyzko, "Computation of discrete logarithms in prime fields", *Designs, Codes and Cryptography*, vol.1 no.1 (May 1991), pp.47-62.
10. K.S. McCurley, "The discrete logarithm problem", pp.49-74 in: *Cryptology and Computational Number Theory – Proc. Symp. Applied Math.*, vol. 42 (1990), AMS.
11. A.K. Lenstra, M.S. Manasse, "Factoring by electronic mail", *Eurocrypt'89*, Springer-Verlag LNCS 434, pp.355-371.
12. U.M. Maurer, "Towards the equivalence of breaking the Diffie-Hellman protocol and computing discrete logarithms", *Crypto'94*, Springer-Verlag LNCS 839, pp.271-281.
13. U.M. Maurer, "Fast Generation of Prime Numbers and Secure Public-Key Cryptographic Parameters", *J. Cryptology*, vol.8 no.3 (1995), 123-156.
14. S.C. Pohlig, M.E. Hellman, "An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance", *IEEE Trans. Information Theory*, vol. IT-24, no.1 (Jan. 1978), pp.106-110.
15. J.M. Pollard, "Monte Carlo methods for index computation (mod p)", *Math. Comp.*, vol.32 no.143 (July 1978) pp.918-924.
16. R.L. Rivest, A. Shamir, L. Adleman, "A method for obtaining digital signatures and public key cryptosystems", *C.ACM*, vol.21 no.2 (Feb. 1978), pp.120–126.
17. R. Rueppel, P.C. van Oorschot, "Modern key agreement techniques", *Computer Communications*, vol.17 no.7 (July 1994), pp.458-465.
18. C.P. Schnorr, "Efficient signature generation by smart cards", *J. Cryptology* vol.4 (1991), pp.161-174.
19. U.S. Department of Commerce / N.I.S.T., *Digital Signature Standard*, FIPS 186, National Technical Information Service, Springfield, Virginia, May 1994.
20. P.C. van Oorschot, M.J. Wiener, "Parallel collision search with application to hash functions and discrete logarithms", *Proc. 2nd ACM Conference on Computer and Communications Security* (Nov. 1994), Fairfax, VA, pp.210-218.
21. M.J. Wiener, "Cryptanalysis of short RSA secret exponents", *IEEE Trans. on Info. Theory*, vol.36 no.3 (May 1990), pp.553–558.
22. Y. Yacobi, "Discrete-log with compressible exponents", *Crypto'90*, Springer-Verlag LNCS 537, pp.639-643.

This article was processed using the LaTeX macro package with LLNCS style