A Generic Algorithm for Small Weight Discrete Logarithms in Composite Groups

Alexander May* and Ilya Ozerov**

Horst Görtz Institute for IT-Security
Ruhr-University Bochum, Germany
Faculty of Mathematics
alex.may@rub.de, ilya.ozerov@rub.de

Abstract. Let $(\mathbb{G}, \cdot)$ be an arbitrary cyclic group of composite order N with $\mathbb{G} \simeq \mathbb{G}_1 \times \mathbb{G}_2$. We present a generic algorithm for solving the discrete logarithm problem in $\mathbb{G}$ with Hamming weight $\delta \log N$, $\delta \in (0,1)$, in time $\widetilde{O}(\sqrt{p} + \sqrt{|\mathbb{G}_2|}^{H(\delta)})$, where p is the largest prime divisor in $\mathbb{G}_1$ and $H(\cdot)$ is the binary entropy function.
Our algorithm improves on the running time of Silver-Pohlig-Hellman's algorithm whenever $\delta \neq \frac{1}{2}$. Moreover, it improves on the Meet-in-the-Middle type algorithms of Heiman, Odlyzko and Coppersmith with running time $\widetilde{O}(\sqrt{|\mathbb{G}|}^{H(\delta)})$ whenever $p < |\mathbb{G}|^{H(\delta)}$.

Keywords: cryptanalysis, generic discrete logarithm, small hamming weight, representations

1 Introduction

The hardness of the discrete logarithm problem on classical computers is one of the most central sources for constructing public key cryptography. In order to achieve minimal key-size and maximal performance, crypto designers usually choose cyclic groups $\mathbb{G}$ for which no group-specific algorithm, e.g. of index calculus type [1], is known. In these groups, the security analysis is based on the performance of generic algorithms.

However, very few generic algorithms for cyclic groups are known. Among them are Shanks' Baby-Step Giant-Step algorithm [10] and its low-memory variant, Pollard's Rho Method [9]. Both algorithms achieve a running time of $\sqrt{|\mathbb{G}|}$. Moreover, it is known by a result of Shoup [11] that generic algorithms in *prime* order groups cannot compute discrete logarithms faster than $\sqrt{|\mathbb{G}|}$.

The generic algorithm of Silver, Pohlig and Hellman [8] can be seen as a generalization of Shanks' algorithm to non-prime order groups. Let $\mathbb{G}$ be a cyclic group with $|\mathbb{G}| = N$ and prime factorization $N = \prod_{i=1}^{k} p_i^{e_i}$. Thus we have $\mathbb{G} \simeq \mathbb{G}_1 \times \ldots \times \mathbb{G}_k$ with cyclic groups $|\mathbb{G}_i| = p_i^{e_i}$. In the Silver-Pohlig-Hellman algorithm, the discrete logarithm is first computed in $\mathbb{G}_i$ modulo p_i, then lifted

* Supported by DFG as part of GRK 1817 Ubicrypt and SPP 1736 Big Data
** Supported by DFG as part of SPP 1307 Algorithm Engineering

modulo $p_i^{e_i}$ and afterwards composed by Chinese Remaindering to the full group order N. Since this process is dominated by the running time of an individual discrete logarithm computation in $\mathbb{G}_i$ modulo p_i, the total running time is dominated by $\max_i\{\sqrt{p_i}\}$.

If we do not further restrict the discrete logarithm problem, the generic algorithms of Pollard and Silver, Pohlig and Hellman are all that we have. If we limit our discrete logarithm to a certain interval $[a, b]$ then the discrete logarithm can be computed by Pollard's kangaroo method [9] in time $\widetilde{O}(\sqrt{b-a})$, which can be seen as another variant of Shanks' algorithm.

More generic algorithms are known when we limit our discrete logarithm to a small Hamming weight. Let α be a generator of $\mathbb{G}$ with an order of bit-size n. Let $\beta = \alpha^x$ with an n-bit integer x having Hamming weight δn, $\delta \in (0, 1)$, where we call δ the relative Hamming weight of x.

A brute-force enumeration of an n-bit number x with Hamming weight δn takes time $\binom{n}{\delta n} \approx 2^{H(\delta)n}$. The algorithms of Heiman-Odlyzko [5], Coppersmith [3] and Stinson [13] split x in two parts of length $\frac{n}{2}$ and Hamming weight $\delta\frac{n}{2}$ each. This is a classical Meet-in-the-Middle approach that achieves a square-root complexity of roughly $2^{\frac{H(\delta)}{2}n} = \widetilde{O}(\sqrt{|\mathbb{G}|}^{H(\delta)})$.

1.1 Our Contribution and Related Work

We present a new algorithm that can be seen as a generalization of the Meet-in-the-Middle algorithms of Heiman-Odlyzko and Coppersmith and the Silver-Pohlig-Hellman algorithm. In spirit, our approach is similar to an algorithm of van Oorschot and Wiener [15] for the discrete logarithm problem with *small* x, as opposed to small Hamming weight x in our case. The van Oorschot-Wiener algorithm computes the CRT-representation of x modulo a factor N_1 of the group order via Silver-Pohlig-Hellman. Thus, x can be expressed as $x = x_1N_1 + x_0$ for some known x_0. Then x_1 is easily computed via Pollard's kangaroo algorithm in time $\widetilde{O}(\sqrt{x_1})$. Thus, van Oorschot and Wiener proceed in a divide and conquer manner, where they split the computation of x in two parts.

Our algorithm also makes use of the Silver-Pohlig-Hellman algorithm as a subroutine. However, our computation of the second part is way more challenging than in the algorithm of van Oorschot and Wiener. Notice that the property of a small Hamming weight discrete logarithm does not transfer to its Chinese Remainder representation and vice versa. Nevertheless, we are able to show that parts of the Chinese Remainder representation automatically reduce the search space for small weight discrete logarithms.

In general, finding algorithms for small Hamming weight appears to be a harder problem than finding algorithms for small size, e.g. for polynomial equations there is an efficient algorithm that finds all small size integer roots due to Coppersmith [4], but there is no analogue known for small Hamming weight roots.

Let $\mathbb{G} \simeq \mathbb{G}'_1 \times \ldots \mathbb{G}'_k$ be our composite group. We write this in the form $\mathbb{G} \simeq \mathbb{G}_1 \times \mathbb{G}_2$, where we suitably combine groups. Our runtime will be dependent

on the size of $|\mathbb{G}_1|$, its prime factorization, and the relative Hamming weight δ of our discrete logarithm problem. So if $k > 2$, then for a given δ we have to form $\mathbb{G}_1$ in such a way that minimizes the running time. Let $\beta = \alpha^x$ be our discrete logarithm problem in $\mathbb{G}$.

Let us first describe a simple enumeration version of our algorithm. Assume $|\mathbb{G}| = N, |\mathbb{G}_1| = N_1, |\mathbb{G}_2| = N_2$ and let n, n_1, n_2 denote the bit-sizes of N, N_1, N_2, respectively. Notice that $N = N_1 N_2$ and thus (roughly) $n = n_1 + n_2$. Let us first compute $x \bmod N_1$, that is we compute the discrete logarithm in the smaller subgroup $\mathbb{G}_1 \times \{1\} \subset \mathbb{G}$. With the Silver-Pohlig-Hellman algorithm this can be done in time $\sqrt{p}$, where p is the largest prime factor of $|\mathbb{G}_1|$. Now we enumerate all natural numbers x' which have weight δ in the upper $n - n_1 = n_2$ most significant bits and which are consistent with the computed discrete logarithm in $\mathbb{G}_1$. We are able to show that the second restriction basically determines the remaining n_1 least significant bits uniquely. Since we do this enumeration as a Meet-in-the-Middle approach, we achieve complexity

$$\sqrt{\binom{n_2}{\delta n_2}} \approx \sqrt{2^{H(\delta) n_2}} \approx \sqrt{|\mathbb{G}_2|}^{H(\delta)}.$$

We want to stress that our algorithm is not designed to attack practical cryptographic schemes. Our main goal was to combine ideas of [6, 12, 7, 2] to obtain one of very few known generic algorithms for discrete logs. Our method is inspired by a recent subset sum algorithm of Howgrave-Graham and Joux [6] and our intention was to understand the full generality of their method in arbitrary groups. In the Howgrave-Graham-Joux algorithm the target vector $x \in \{0,1\}^n$ is represented as a sum of vectors $x_1, x_2 \in \{0,1\}^n$. But their sum is a vector sum in $\mathbb{Z}^n$, whereas our vectors represent integers and the addition $x_1 + x_2$ is in $\mathbb{Z}$, i.e. we allow carry bits that allow for new kinds of representations.

Our algorithm is also in the spirit of Stern's Information Set Decoding technique [12] for decoding random linear codes, where one part of the unknown error vector is obtained combinatorically, whereas the remaining bits are computed efficiently through simple linear algebra. Notice that like in [7, 2] it is possible to combine our technique with the classical technique of [6]. We leave as an open problem whether this leads to even better results.

2 Known Generic Algorithms

In this section, we quickly repeat some standard algorithms for discrete logarithms, since we will use variations of these as subroutines in our algorithm. We start by explaining the SORT-AND-MATCH algorithm that is the basis for all Meet-in-the-Middle approaches.

Let $\alpha^x = \beta$ be a discrete logarithm instance in some group $\mathbb{G}$ generated by α. Let us write $x = x_1 + x_2$, where $x_1 \in \mathcal{S}_1, x_2 \in \mathcal{S}_2$ for some sets $\mathcal{S}_1, \mathcal{S}_2 \subset \mathbb{Z}$. Then we obtain the identity

$$\alpha^{x_1} = \beta \cdot \alpha^{-x_2}.$$

We compute a list $\mathcal{L}$ that contains the elements (α^{x_1}, x_1) for all $x_1 \in \mathcal{S}_1$. Then, we compute $(\beta \cdot \alpha^{-x_2}, x_2)$ for all $x_2 \in \mathcal{S}_2$. Any element $(\beta \cdot \alpha^{-x_2}, x_2)$ that matches an element $(\alpha^{x_1}, x_1) \in \mathcal{L}$ in its first component yields a solution $x = x_1 + x_2$ to the discrete logarithm problem. This strategy leads to the algorithm SORT-AND-MATCH.

Algorithm 1

procedure SORT-AND-MATCH($\mathcal{S}_1, \mathcal{S}_2, \alpha, \beta$) ▷ $\mathcal{S}_1, \mathcal{S}_2 \subseteq \mathbb{Z}_N$
    Create a list $\mathcal{L}$ with entries (α^{x_1}, x_1) for all $x_1 \in \mathcal{S}_1$, sort by its first component
    for all $x_2 \in \mathcal{S}_2$ **do**
        Binary search for a $(\alpha^{x_1}, x_1) \in \mathcal{L}$ such that $\alpha^{x_1} = \beta/\alpha^{x_2}$
        return $x_1 + x_2$ if there is a match
    end for
    return no match
end procedure

It is not hard to see that both the time and space complexity of SORT-AND-MATCH are $\widetilde{O}(|\mathcal{S}_1| + |\mathcal{S}_2|)$, where the $\widetilde{O}(\cdot)$-notation suppresses logarithmic terms. So if $x \in \mathcal{S}$ and $x_1 \in \mathcal{S}_1$, $x_2 \in \mathcal{S}_2$ with $|\mathcal{S}_1| \approx |\mathcal{S}_2| \approx \sqrt{|\mathcal{S}|}$ then SORT-AND-MATCH achieves the square root of the time complexity that is required for simply enumerating all $x \in \mathcal{S}$. Hence our goal is to define $\mathcal{S}_1, \mathcal{S}_2$ in such a way that the maximum of their cardinalities roughly equals $\sqrt{|\mathcal{S}|}$, and that with high probability there always exist $(x_1, x_2) \in \mathcal{S}_1 \times \mathcal{S}_2$ with $x = x_1 + x_2$.

In the following, we illustrate how the selection of $\mathcal{S}_1, \mathcal{S}_2$ is done for Shanks' algorithm and for its variations due to Heiman-Odlyzko, Coppersmith and Stinson. Let $(x_{n-1}, \dots x_0) \in \{0,1\}^n$ be the binary representation of x, i.e. $x = \sum_{i=0}^{n-1} x_i 2^i$. Then we write $x = x_1 + x_2 = v_1 \cdot 2^{n/2} + v_2$ with $0 \le v_1, v_2 < 2^{n/2}$. Fig. 1 illustrates this splitting in form of the binary representations of x_1, x_2. Notice that for ease of writing throughout this work we ignore any complications that arise from rounding terms like $\frac{n}{2}$, since this is always easy to solve.

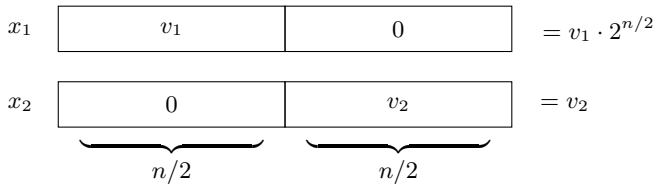

Fig. 1. Splitting

It is obvious that the search spaces $\mathcal{S}_1, \mathcal{S}_2$ both have cardinality $2^{\frac{n}{2}} = \sqrt{|\mathcal{S}|} = \sqrt{|\{0,1\}^n|}$ and that there always exists a pair $(x_1, x_2) \in \mathcal{S}_1 \times \mathcal{S}_2$ with $x = x_1 + x_2$.

This leads to a generic discrete logarithm algorithm in $\mathbb{G}$ with time and space complexity $\widetilde{O}(2^{\frac{n}{2}}) = \widetilde{O}(\sqrt{|\mathbb{G}|})$.

In a nutshell, when we move to discrete logarithms x whose binary representation $(x_{n-1}, \ldots, x_0)$ have Hamming weight δn, we can easily adapt the splitting of Fig. 1. Namely, we enumerate over all $v_1, v_2 \in \{0,1\}^{\frac{n}{2}}$ with Hamming weight $\delta\frac{n}{2}$. Hence, $\mathcal{S}$ consists of all numbers that can be represented as n-bit vectors with relative Hamming weight δ, whereas the numbers in $\mathcal{S}_1$, $\mathcal{S}_2$ can be represented as $\frac{n}{2}$-bit vectors with relative Hamming weight δ, where we append a $0^{\frac{n}{2}}$-string accordingly.

Let us first compare the cardinalities of $\mathcal{S}$ and $\mathcal{S}_1$, $\mathcal{S}_2$. Here, we use the well-known approximation $\binom{n}{\delta n} \approx 2^{H(\delta)n}$, that stems from Stirling's formula. This implies that $|\mathcal{S}_1| = |\mathcal{S}_2| \approx 2^{H(\delta)\frac{n}{2}} = \widetilde{O}(\sqrt{|\mathbb{G}|}^{H(\delta)})$ which is equal to the square root of $|\mathcal{S}|$. Thus, we obtain a Meet-in-the-Middle algorithm with square root time and space complexity.

Notice however, that as opposed to Shanks' algorithm not every x with Hamming weight δn admits a splitting in x_1, x_2 as above, where both x_i have Hamming weight $\delta\frac{n}{2}$. The probability that a random x splits in this way is $\binom{n/2}{\delta n/2}^2 / \binom{n}{\delta n} = \Theta(\frac{1}{\sqrt{n}})$. In the algorithms of Heiman, Odlyzko and Coppersmith this problem is solved by a combinatorial structure which is called a *splitting set*, which basically allows to re-randomize the coordinates for the splitting in x_1, x_2.

A different deterministic approach due to Coppersmith is a simple application of the intermediate value theorem. Assume wlog that the weight in the v_1-part is too high, and the weight in the v_2-part is too low. Let us rotate both parts cyclically until they change places. Since a rotation by one position changes the weight in each part by at most 1, there must exist one out of the $n/2$ rotations where both parts share the same weight.

In the following, we propose a slightly different solution for guaranteeing the existence of a valid splitting, which we use in our algorithm. Namely, we choose the non-zero parts of the binary representation of the element in $\mathcal{S}_1, \mathcal{S}_2$ from $\{0,1\}^{\frac{n}{2}}$, where their relative Hamming weight lies in the interval $(\delta - \varepsilon, \delta + \varepsilon)$ for some small $\varepsilon > 0$. In this way, we can ensure that a randomly chosen x splits in $x_1 + x_2$ with appropriate Hamming weights in this interval with a probability that is exponentially close to 1, while only slightly increasing the running time. The main reason for choosing such a weight interval is that it simplifies the description and analysis of our algorithm significantly.

3 Our New Generic Discrete Log Algorithm

Let α generate a composite order group $\mathbb{G} \simeq \mathbb{G}_1 \times \mathbb{G}_2$ with $|\mathbb{G}| = N$, $|\mathbb{G}_1| = N^\tau$ and $|\mathbb{G}_2| = N^{1-\tau}$ for some $\tau \in (0,1)$. In general, there might be several ways to decompose $\mathbb{G}$ as $\mathbb{G}_1 \times \mathbb{G}_2$. We will first describe our algorithm for a fixed decomposition. Afterwards, we will minimize the running time by adjusting the decomposition accordingly.

Our new algorithm combines the Silver-Pohlig-Hellman idea with a subsequent Meet-in-the-Middle approach for enumerating small weight vectors. As described in Section 2, we have to define the sets $\mathcal{S}_1, \mathcal{S}_2$ that describe how we split $x = x_1 + x_2$ with $(x_1, x_2) \in \mathcal{S}_1 \times \mathcal{S}_2$. We illustrate the binary representation for our candidates (x_1, x_2) in Fig. 2. Here, the v_i have relative weight δ, whereas the w_i may have arbitrary weight.

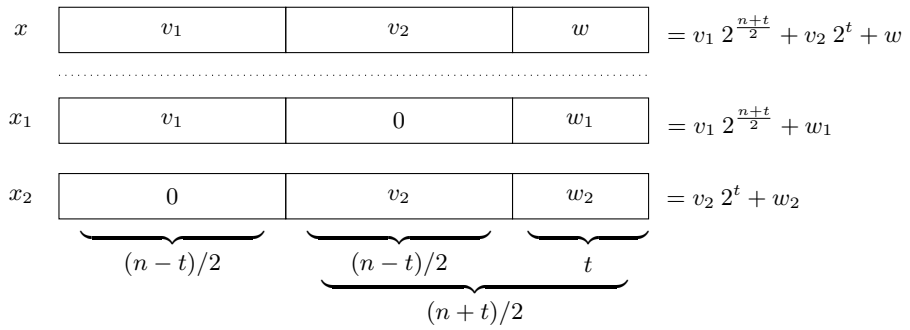

Fig. 2. New splitting

In the following, we always assume wlog that we know the factorization of the group order N. Notice that this does not limit the applicability of our generic algorithm, since our algorithm's running time is exponential in the bit-length of N anyway, whereas the factorization of N can be computed in sub-exponential time.

The main idea of our new algorithm DLOG is as follows. We first apply the algorithm of Silver, Pohlig and Hellman to the smaller subgroup $\mathbb{G}_1$ to obtain the discrete logarithm x' modulo $M := |\mathbb{G}_1|$. Afterwards, we apply a Meet-in-the-Middle technique on the bigger subgroup $\mathbb{G}_2$ where we cut down the search space by the amount of information that is provided by x'. More precisely, knowing only $n - t$ bits of the discrete logarithm x (i.e. v_1 and v_2 in Fig. 2), it is possible to compute the remaining t consecutive bits w in polynomial time with the help of $x' = x \bmod M$.

Let us fix some useful notation. We denote $x' = [x']_M = [x]_M$, where $[\cdot]_M$ describes the smallest non-negative representative of some number modulo M, i.e. in $[0, M)$. Let us choose t such that $2^{t-1} < M \leq 2^t$. Then w is either $[w]_M := [x' - v_1 \cdot 2^{(n+t)/2} - v_2 \cdot 2^t]_M$ or $[w]_M + M$.

For any $x \in \mathbb{N}$ we denote by $\mathrm{wt}(x)$ the Hamming weight of the binary representation of x.

In lines 6 through 9 a list $\mathcal{S}_1$ is computed by enumerating all values of v_1 (see Fig. 2). We show that the remaining t-bit value w_1 can be uniquely obtained from v_1. Similarly, in lines 10 through 15 we obtain only three possible values for w_2 for each value of v_2. In total, it is sufficient to perform a Meet-in-the-Middle

Algorithm 2

procedure DLOG($|\mathbb{G}_1|, |\mathbb{G}_2|, \alpha, \beta, \delta, \varepsilon$)
  $n \leftarrow \lceil \log_2(|\mathbb{G}_1| \cdot |\mathbb{G}_2|) \rceil$
  $t \leftarrow \lceil \log_2(|\mathbb{G}_1|) \rceil$
  $x' \leftarrow \text{SPH}(|\mathbb{G}_1|, \alpha^{|\mathbb{G}_2|}, \beta^{|\mathbb{G}_2|})$ ▷ Use Silver-Pohlig-Hellman to get x mod $|\mathbb{G}_1|$.
  $M \leftarrow |\mathbb{G}_1|$
  $\mathcal{S}_1 \leftarrow \{\}$
  for all $0 \leq v_1 < 2^{(n-t)/2}$ with $(\delta-\varepsilon)\frac{n-t}{2} \leq \text{wt}(v_1) \leq (\delta+\varepsilon)\frac{n-t}{2}$ **do**
    $\mathcal{S}_1 \leftarrow \mathcal{S}_1 \cup \{v_1 \cdot 2^{(n+t)/2} + [-v_1 \cdot 2^{(n+t)/2}]_M\}$
  end for
  $\mathcal{S}_2 \leftarrow \{\}$
  for all $0 \leq v_2 < 2^{(n-t)/2}$ with $(\delta-\varepsilon)\frac{n-t}{2} \leq \text{wt}(v_2) \leq (\delta+\varepsilon)\frac{n-t}{2}$ **do**
    $\mathcal{S}_2 \leftarrow \mathcal{S}_2 \cup \{v_2 \cdot 2^t + [x' - v_2 \cdot 2^t]_M\}$
    $\mathcal{S}_2 \leftarrow \mathcal{S}_2 \cup \{v_2 \cdot 2^t + [x' - v_2 \cdot 2^t]_M - M\}$
    $\mathcal{S}_2 \leftarrow \mathcal{S}_2 \cup \{v_2 \cdot 2^t + [x' - v_2 \cdot 2^t]_M + M\}$
  end for
  return SORT-AND-MATCH($\mathcal{S}_1, \mathcal{S}_2, \alpha, \beta$)
end procedure

attack on only $n-t$ bits instead of the full n bits of the binary representation of x. Eventually, the subroutine SORT-AND-MATCH finds the discrete logarithm x, since we show that with overwhelming probability there is always some $x_1 \in \mathcal{S}_1$ and $x_2 \in \mathcal{S}_2$ that sum to x.

Theorem 1. *Let α be a generator of an cyclic group $\mathbb{G} \simeq \mathbb{G}_1 \times \mathbb{G}_2$ of known order N, where N has bit-size n. Let $\delta \in (0, \frac{1}{2})$ and let x be sampled uniformly at random from all elements of $\mathbb{Z}_N$ with Hamming weight δn. Let $\beta := \alpha^x$ and p be the largest prime factor of $|\mathbb{G}_1|$. Then for any $\varepsilon > 0$ with $\delta + \varepsilon \leq \frac{1}{2}$ on input $(|\mathbb{G}_1|, |\mathbb{G}_2|, \alpha, \beta, \delta)$ algorithm* DLOG *outputs x with probability at least $1 - \frac{4(n+1)}{|\mathbb{G}_2|^{\varepsilon^2}}$ in time $\widetilde{O}\left(\sqrt{p} + \sqrt{|\mathbb{G}_2|}^{H(\delta+\varepsilon)}\right)$ and space $\widetilde{O}\left(\sqrt{|\mathbb{G}_2|}^{H(\delta+\varepsilon)}\right)$.*

Proof. Let us first define $M := |\mathbb{G}_1|$, $n := \lceil \log_2(N) \rceil$, $t := \lceil \log_2(M) \rceil$ and $x' := [x]_M$ as in DLOG. Recall that $[\cdot]_M$ denotes the least non-negative representative modulo M, and $\text{wt}(\cdot)$ denotes the Hamming weight of the binary representation. For simplicity, we ignore rounding problems like with $(n-t)/2$, since they can easily be resolved without affecting the asymptotic running time.

Similar to the standard Meet-in-the-Middle approach from Section 2, in DLOG we decompose the unknown $x = v_1 \cdot 2^{(n+t)/2} + v_2 \cdot 2^t + w$ for some $0 \leq v_1, v_2 < 2^{(n-t)/2}$ and $0 \leq w < 2^t$, as illustrated in Fig. 2. Moreover, we require that both v_1, v_2 have some Hamming weight in the interval $[(\delta-\varepsilon)(n-t)/2, (\delta+\varepsilon)(n-t)/2]$. The proof is organized as follows. Firstly, we show that any random x possesses the correct weights for v_1, v_2 with overwhelming probability. Secondly, we show that for the correct weights, DLOG always outputs x. Thus, DLOG is of Las Vegas type. Its output is always correct, but DLOG fails on an exponentially small fraction of all input instances.

Let $x \in \mathbb{Z}_N$ be chosen uniformly at random with Hamming weight $\mathrm{wt}(x) = \delta n$. We show that $(\delta-\varepsilon)(n-t)/2 \leq \mathrm{wt}(v_1), \mathrm{wt}(v_2) \leq (\delta+\varepsilon)(n-t)/2$ holds with a probability that is at least $1 - 4(n+1)/|\mathbb{G}_2|^{\varepsilon^2}$. Let $(x_{n-1}, \ldots, x_0)$ denote the binary representation of x, and let X_i be a random variable for x_i. For simplifying our proof, we assume that x was sampled by n independent Bernoulli trials with $\mathbb{P}[X_i = 1] = \delta$ for all bits $i = 0, \ldots, n-1$. Notice that sampling x in this manner and rejecting all x that have an incorrect Hamming weight gives the same distribution as sampling x uniformly at random from all x with Hamming weight δn.

Let $I \subseteq \{0, \ldots, n-1\}$ with $|I| = (n-t)/2$ be some index set. Let $X = \sum_{i=0}^{n-1} X_i$ be the Hamming weight of x, and let $Y = \sum_{i \in I} X_i$ be the Hamming weight of coordinates I. In order to estimate DLOG's failure probability, we compute

$$\mathbb{P}\left[|Y - \delta(n-t)/2| > \varepsilon(n-t)/2 \;\mid\; X = \delta n\right],$$

which is the probability that the Hamming weight on the I-bits of x is *not* in the range between $(\delta-\varepsilon)\cdot(n-t)/2$ and $(\delta+\varepsilon)\cdot(n-t)/2$, under the condition that x has the correct Hamming weight. Notice that $\mathbb{P}[X = \delta n] \geq \mathbb{P}[X = i]$ for any $i \neq \delta n$. Since $0 \leq X \leq n$, we get $1 = \sum_{i=0}^{n} \mathbb{P}[X = i] \leq (n+1) \cdot \mathbb{P}[X = \delta n]$ and thus $\mathbb{P}[X = \delta n] \geq \frac{1}{n+1}$. This implies

$$\begin{aligned}&\mathbb{P}\left[|Y - \delta(n-t)/2| > \varepsilon(n-t)/2 \;\mid\; X = \delta n\right] \\ &\leq (n+1) \cdot \mathbb{P}\left[|Y - \delta(n-t)/2| > \varepsilon(n-t)/2\right].\end{aligned}$$

An application of Hoeffding's inequality yields

$$(n+1) \cdot \mathbb{P}\left[|Y - \delta(n-t)/2| > \varepsilon(n-t)/2\right] \leq 2(n+1)2^{-\varepsilon^2(n-t)} \leq \frac{2(n+1)}{|\mathbb{G}_2|^{\varepsilon^2}}.$$

Hence, we obtain a probability of at most $2(n+1)/|\mathbb{G}_2|^{\varepsilon^2}$ that the relative Hamming weight for *one* of v_1, v_2 is incorrect. By the union bound, the probability that v_1 or v_2 have incorrect weight is bounded by $4(n+1)/|\mathbb{G}_2|^{\varepsilon^2}$.

It remains to show that for correct Hamming weight of v_1, v_2 DLOG always succeeds in computing x. By the correctness of our SORT-AND-MATCH routine, it suffices to show the existence of $(x_1, x_2) \in \mathcal{S}_1 \times \mathcal{S}_2$ with $x_1 + x_2 = x$.

We split x in three parts v_1, v_2, w with $x = v_1 \cdot 2^{(n+t)/2} + v_2 \cdot 2^t + w$ (see Fig. 2). Denote

$$x_1 := v_1 \cdot 2^{(n+t)/2} + w_1,\; x_2 := v_2 \cdot 2^t + w_2 \text{ with } 0 \leq v_1, v_2 < 2^{(n-t)/2} \text{ and } 0 \leq w_1, w_2 < 2^t.$$

In DLOG we enumerate a first list $\mathcal{S}_1$ of all possible v_1 and compute for each v_1 a corresponding w_1. We proceed with v_2 and their corresponding w_2 analogously.

In $\mathcal{S}_1$ we choose to fix $x_1 = 0 \bmod M$ — the value 0 could be any constant in $\mathbb{Z}_M$. Therefore, we compute $w_1 = -v_1 \cdot 2^{(n+t)/2} \bmod M$ and store the corresponding integer

$$x_1 := v_1 \cdot 2^{(n+t)/2} + [-v_1 \cdot 2^{(n+t)/2}]_M.$$

Notice that there always exists a $0 \leq w_1 < 2^t$ with $w_1 = -v_1 \cdot 2^{(n+t)/2} \bmod M$, since $M \leq 2^t$ by the choice of t.

Since we have to ensure $x_1 + x_2 = x$, we require $x_1 + x_2 = x' \bmod M$ and thus $x_2 = x' \bmod M$ by our choice of x_1. This in turn implies $w_2 = x' - v_2 \cdot 2^t \bmod M$. By construction, we obtain

$$x_1 + x_2 = v_1 \cdot 2^{(n+t)/2} + [-v_1 \cdot 2^{(n+t)/2}]_M + v_2 \cdot 2^t + [x' - v_2 \cdot 2^t]_M = x \bmod M.$$

However, this does not necessarily imply $x = x_1 + x_2$ over $\mathbb{Z}$. Especially, we have to guarantee $w_1 + w_2 = w$. Notice that by definition $w < 2^t$ and $M \leq 2^t < 2M$. Since $0 \leq [w_1]_M, [w_2]_M < M$ we have $0 \leq [w_1]_M + [w_2]_M < 2M$.

If either $0 \leq [w_1]_M + [w_2]_M < M$ and $w < M$ (case I+I in Fig. 3) or $M \leq [w_1]_M + [w_2]_M < 2M$ and $M \leq w$ (case II+II in Fig. 3), we are done. If $0 \leq [w_1]_M + [w_2]_M < M$ and $M \leq w$ (case I+II in Fig. 3), we have to add M to $[w_1]_M + [w_2]_M$. In the remaining case II+I, we have to substract M from $[w_1]_M + [w_2]_M$.

Thus, $[w_1]_M + [w_2]_M + kM = w$ holds for some $k \in \{-1, 0, 1\}$. In DLOG we choose $x_1 = v_1 \cdot 2^{(n+t)/2} + [w_1]_M \in \mathcal{S}_1$ and $x_2 = v_2 \cdot 2^t + [w_2]_M + kM \in \mathcal{S}_2$ for all $k \in \{-1, 0, 1\}$. For the correct k, we obtain $x_1 + x_2 = x$, as desired. Thus, SORT-AND-MATCH succeeds, and DLOG outputs the discrete logarithm x.

It remains to show the time and space complexities. SPH takes time $\widetilde{O}(\sqrt{p})$ with only polynomial memory consumption, when using Pollard's Rho Method as a subroutine. Notice that the complexity of the for-loops in step 7 and 11 of DLOG are dominated by the time to enumerate and store those v_i with largest weight $(\delta + \varepsilon)\frac{n-t}{2}$. Thus, our Meet-in-the-Middle attack has time and space complexity $\widetilde{O}\left(\sqrt{|\mathbb{G}_2|}^{H(\delta+\varepsilon)}\right)$.

□

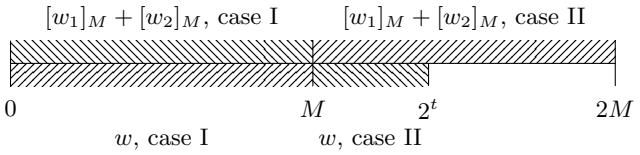

Fig. 3. $\pm M$

Remark 1 (Large weight). DLOG is by definition restricted to small Hamming weight $0 < \delta \leq \frac{1}{2}$. Symmetrically, DLOG can be applied to large Hamming weight $\frac{1}{2} \leq \delta < 1$ by transforming the discrete logarithm instance to $\widetilde{\beta} := \alpha^{2^n - 1}/\beta$. This transforms x to $\widetilde{x} = (2^n - 1) - x$ with Hamming weight $(1 - \delta) \cdot n$.

Remark 2 (Representations). Our algorithm can be interpreted in terms of the representation technique introduced by Howgrave-Graham and Joux [6] for solving the subset sum problem. Notice that we split $w = w_1 + w_2$ with $w_1, w_2 \in \mathbb{Z}_M$.

Thus, we obtain exactly M representations (w_1, w_2) of w as a sum. In our case, we use the fact that exactly one representation (w_1, w_2) ensures that $x_1 = 0 \bmod M$ and $x_2 = x \bmod M$, simultaneously. We can directly compute this representation in polynomial time – once x' is known – without any further assumption on the problem instance. This differs from [6], where the authors spent exponential time to compute a representation of the solution and only receive one representation on expectation, assuming a uniform distribution of the subset sum elements.

Remark 3 (Getting rid of ε). Recall that we introduced ε to ensure that the Hamming weight of v_1 and v_2 lies within some ε-strip around its expectation with overwhelming probability. If we set $\varepsilon = 0$, then DLOG finds the discrete logarithm x only for a polynomial fraction of all x that exactly match the expected Hamming weight on v_1, v_2.

One might be tempted to use cyclic rotations of the binary representation of x, just as described at the end of Section 2. However, some problems arise here. If we fully rotate, we obtain a bit vector for which the Hamming weight of v_1 and v_2 matches its expectation. In this case, it might however happen that w gets split into two parts by the cyclic rotations. In this case, we were not able to bound the number of w's by a polynomial. We could also consider the case where we do not fully rotate x, but restrict to $n - t$ left rotations only such that the w-part does not split. We conjecture that the number of pathological instances where DLOG does not succeed for at least one of the $n - t$ rotations is exponentially small in this case, but we were not able to prove that.

3.1 How to optimally split $\mathbb{G}$ into subgroups

It remains to show how to optimally choose the subgroups $\mathbb{G}_1$ and $\mathbb{G}_2$ dependent on the factorization of $|\mathbb{G}|$ and on the Hamming weight $\delta \cdot \log |\mathbb{G}|$ of x. Since we apply Silver-Pohlig-Hellman on $\mathbb{G}_1$, the group $\mathbb{G}_1$ should contain all prime subgroups of $\mathbb{G}$ that are smaller or as large as the largest prime subgroup of $\mathbb{G}_1$. In other words, if $N = \prod_{i=1}^{k} p_i$ is the factorization of the order of $\mathbb{G}$ and $p_1 \leq \ldots \leq p_k$, the only useful choices are $|\mathbb{G}_1| = \prod_{i=1}^{\ell} p_i$ and $|\mathbb{G}_2| = \prod_{i=\ell+1}^{k} p_i$ for $1 \leq \ell \leq k-1$. This is because we have to spend time $\sqrt{p}$ for the maximal prime divisor p of $|\mathbb{G}_1|$ anyway. Thus, our ordering of the p_i minimizes $|\mathbb{G}_2|$ and thus the overall running time.

Among the remaining $k-1$ choices, we have to find the best choice for ℓ. Fix an ℓ, $1 \leq \ell \leq k-1$, and define $\tau_i := \log_N p_i$ for each $1 \leq i \leq k$. From Theorem 1, the time complexity of DLOG is

$$\widetilde{O}\left(\sqrt{p_\ell} + \sqrt{p_{\ell+1} \cdots p_k}^{H(\delta+\varepsilon)}\right) = \widetilde{O}\left((2^{n/2})^{\tau_\ell} + (2^{n/2})^{(\tau_{\ell+1}+\ldots+\tau_k)\cdot H(\delta+\varepsilon)}\right).$$

Let us define $p_0 := 1$, and thus $\tau_0 = 0$. In the case $\ell = 0$ we obtain the time complexity of the standard small weight Meet-in-the-Middle algorithm without using SPH. In the case $\ell = k$ we obtain the standard SPH without any Meet-in-the-Middle approach. Thus, our algorithm perfectly interpolates between both cases and there exist τ_i such that our algorithm improves upon SPH and standard Meet-in-the-Middle for any $0 < \ell < k$ with $\delta < \frac{1}{2}$ (cf. Fig. 4).

Theorem 2. *Let $\tau_0 := 0$. Given δ, ε with $\delta + \varepsilon \in (0, \frac{1}{2}]$ and $\tau_1, \ldots, \tau_k$ as defined above, an optimal choice for* DLOG *is to pick $0 \leq \ell \leq k-1$ such that*

$$\frac{\tau_\ell}{\tau_\ell + \ldots + \tau_k} < H(\delta + \varepsilon) \leq \frac{\tau_{\ell+1}}{\tau_{\ell+1} + \ldots + \tau_k}$$

and to choose $|\mathbb{G}_1| = \prod_{i=1}^{\ell} p_i$ and $|\mathbb{G}_2| = \prod_{i=\ell+1}^{k} p_i$.

Proof. First notice that

$$\bigcup_{\ell=0}^{k-1} \left(\frac{\tau_\ell}{\tau_\ell + \ldots + \tau_k}, \frac{\tau_{\ell+1}}{\tau_{\ell+1} + \ldots + \tau_k} \right]$$

defines a disjoint partition of $(0, 1]$, due to the fact that $\tau_1 + \ldots + \tau_k = 1$. Thus each $\delta + \varepsilon$ with $0 < H(\delta + \varepsilon) \leq 1$ leads to a unique choice of ℓ.

Fix δ, ε and choose ℓ as defined above. We want to show that for each choice of $\ell' \neq \ell$, DLOG's complexity does not improve. Notice that there may be other choices for ℓ that achieve the same complexity.

If $\ell' < \ell$, then it is easy to see that $(\tau_{\ell+1} + \ldots + \tau_k) \cdot H(\delta+\varepsilon) \leq (\tau_{\ell'+1} + \ldots + \tau_k) \cdot H(\delta+\varepsilon)$. Since $\frac{\tau_\ell}{\tau_\ell + \ldots + \tau_k} < H(\delta+\varepsilon)$, we also have $\tau_\ell < (\tau_\ell + \ldots + \tau_k) \cdot H(\delta+\varepsilon) \leq (\tau_{\ell'+1} + \ldots + \tau_k) \cdot H(\delta + \varepsilon)$. Thus the complexity does not improve for $\ell' < \ell$.

If $\ell' > \ell$, obviously we have $\tau_\ell \leq \tau_{\ell'}$. Additionally, we have that $(\tau_{\ell+1} + \ldots + \tau_k) \cdot H(\delta + \varepsilon) \leq \tau_{\ell+1} \leq \tau_{\ell'}$, since $H(\delta + \varepsilon) \leq \frac{\tau_{\ell+1}}{\tau_{\ell+1} + \ldots + \tau_k}$. Therefore, the complexity also does not improve for any $\ell' > \ell$. □

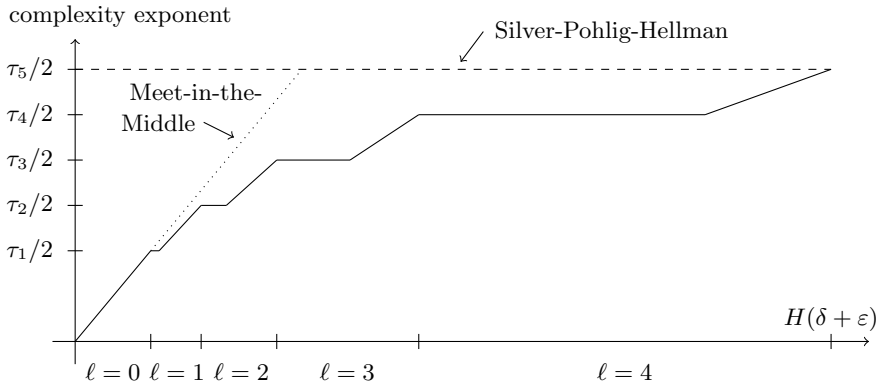

Fig. 4. time complexity for $k = 5$, $\tau_1 = 0.1, \tau_2 = 0.15, \tau_3 = 0.2, \tau_4 = 0.25, \tau_5 = 0.3$

Fig. 4 shows the time complexity for a fixed group with a size of N that is a product of 5 primes with sizes $N^{0.1}, N^{0.15}, N^{0.2}, N^{0.25}$ and $N^{0.3}$. In this case,

Silver-Pohlig-Hellman (dashed line in Fig. 4, corresponding to $\ell = k$) has a time complexity of $N^{0.15}$. As we can see, our algorithm's improvement (straight line) is defined piecewise for $1 \leq \ell \leq k - 1$. For small values of δ, a standard small weight Meet-in-the-Middle approach (dotted line, corresponding to $\ell = 0$) yields the optimal complexity.

References

1. L. M. Adleman. A subexponential algorithm for the discrete logarithm problem with applications to cryptography (abstract). In *FOCS*, pages 55–60. IEEE Computer Society, 1979.
2. A. Becker, A. Joux, A. May, and A. Meurer. Decoding random binary linear codes in 2 n/20: How 1 + 1 = 0 improves information set decoding. In *EUROCRYPT*, pages 520–536, 2012.
3. D. Coppersmith. In *Private communication to Scott Vanstone*, 1979.
4. D. Coppersmith. Small solutions to polynomial equations, and low exponent rsa vulnerabilities. *J. Cryptology*, 10(4):233–260, 1997.
5. R. Heiman. A note on discrete logorithms with special structure. In R. A. Rueppel, editor, *EUROCRYPT*, volume 658 of *Lecture Notes in Computer Science*, pages 454–457. Springer, 1992.
6. N. Howgrave-Graham and A. Joux. New generic algorithms for hard knapsacks. In H. Gilbert, editor, *EUROCRYPT*, volume 6110 of *Lecture Notes in Computer Science*, pages 235–256. Springer, 2010.
7. A. May, A. Meurer, and E. Thomae. Decoding random linear codes in 2 0.054n. In *ASIACRYPT*, pages 107–124, 2011.
8. S. Pohlig and M. Hellman. An improved algorithm for computing logarithms over gf(p) and its cryptographic significance. In *IEEE Trans. Information Theory 24*, pages 106–110, 1978.
9. J. M. Pollard. Monte carlo methods for index computation (mod p). In *Mathematics of Computation 32 (143)*, page 918924, 1978.
10. D. Shanks. Class number, a theory of factorization and genera. In *Proc. Symp. Pure Math. 20, AMS*, page 415440, 1971.
11. V. Shoup. Lower bounds for discrete logarithms and related problems. In W. Fumy, editor, *EUROCRYPT*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266. Springer, 1997.
12. J. Stern. A method for finding codewords of small weight. In *Coding Theory and Applications*, pages 106–113, 1988.
13. D. R. Stinson. Some baby-step giant-step algorithms for the low hamming weight discrete logarithm problem. *Math. Comput.*, 71(237):379–391, 2002.
14. P. C. van Oorschot and M. J. Wiener. Improving implementable meet-in-the-middle attacks by orders of magnitude. In *CRYPTO*, pages 229–236, 1996.
15. P. C. van Oorschot and M. J. Wiener. On diffie-hellman key agreement with short exponents. In U. M. Maurer, editor, *EUROCRYPT*, volume 1070 of *Lecture Notes in Computer Science*, pages 332–343. Springer, 1996.