Some baby-step giant-step algorithms for the low hamming weight discrete logarithm problem

D. R. Stinson
Department of Combinatorics and Optimization
University of Waterloo
Waterloo Ontario, N2L 3G1
Canada
`dstinson@cacr.math.uwaterloo.ca`

February 24, 1999

Abstract

In this paper, we present several baby-step giant-step algorithms for the low hamming weight discrete logarithm problem. In this version of the discrete log problem, we are required to find a discrete logarithm in a finite group, given that the unknown logarithm has a specified number of 1's in its binary representation. Heiman and Odlyzko presented the first algorithms for this problem. Unpublished improvements by Coppersmith include a deterministic algorithm with complexity $O\left(m\binom{\frac{m}{2}}{\frac{t}{2}}\right)$, and a Las Vegas algorithm with complexity $O\left(\sqrt{t}\binom{\frac{m}{2}}{\frac{t}{2}}\right)$.

We perform an average-case analysis of Coppersmith's deterministic algorithm. The average-case complexity achieves only a constant factor speed-up over the worst-case. Therefore, we present a generalized version of Coppersmith's algorithm, utilizing a combinatorial set system that we call a *splitting system.* Using probabilistic methods, we prove a new existence result for these systems that yields a deterministic algorithm with complexity $O\left(t^{3/2}\left(\log m\right)\binom{\frac{m}{2}}{\frac{t}{2}}\right)$. We also present some explicit consturctions for splitting systems that make use of perfect hash families.

Keywords: discrete logarithm problem, baby-step giant-step algorithm, splitting system.

1 Introduction: the Heiman-Odlyzko algorithm

Let G be an abelian group, written multiplicatively. Let $\alpha \in G$, and suppose $\beta \in \langle \alpha \rangle$. The *discrete logarithm* $\log_\alpha \beta$ is the unique integer x such that $0 \leq x \leq \mathsf{ord}(\alpha) - 1$ and $\alpha^x = \beta$. The *discrete logarithm problem* is to compute $\log_\alpha \beta$, given α and β.

Denote $m = \lceil \log_2(\mathsf{ord}(\alpha)) \rceil$. Then the binary representation of $x = \log_\alpha \beta$ requires at most m bits, so we can write

$$x = \sum_{i=0}^{m-1} x_i 2^i,$$

where $x_i \in \{0,1\}$ for $0 \leq i \leq m-1$. The *hamming weight* of an integer x, denoted $\mathsf{wt}(x)$, is the number of 1's in its binary representation. It is often advantageous to choose x such that $\mathsf{wt}(x)$ is "small" compared to m; this makes it faster to compute α^x using a typical square-and-multiply algorithm. However, if $\mathsf{wt}(x)$ is too small, then this fact can be exploited by an adversary who is trying to compute x.

Suppose $t < m$ is a positive integer. Given α and β, the *hamming weight t discrete logarithm problem* is to compute $\log_\alpha \beta$ whenever $\mathsf{wt}(\log_\alpha \beta) = t$. In this paper, we look at several algorithms for the hamming weight t discrete logarithm problem. The algorithms can be thought of as "baby-step giant-step algorithms" (see, e.g., [8, §3.6.2]).

For convenience, we will assume throughout this paper that m and t are both even integers (if this is not the case, then the algorithms we present can be altered in a straightforward manner).

The binary vector $(x_0, \ldots, x_{m-1})$ can be regarded as the characteristic vector of a subset of $\mathbb{Z}_m$ in an obvious way. This correspondence is made explicit by the two mappings

$$\mathsf{set} : \{0, \ldots, 2^m - 1\} \rightarrow 2^{\mathbb{Z}_m}$$

and

$$\mathsf{val} : 2^{\mathbb{Z}_m} \rightarrow \{0, \ldots, 2^m - 1\}$$

which are defined as follows:

$$\mathsf{set}(x) = \{i : x_i = 1\},$$

where $(x_0, \ldots, x_{m-1})$ is the binary representation of x; and

$$\mathsf{val}(Y) = \sum_{i \in Y} 2^i.$$

Clearly val and set are inverse functions, and

$$\mathsf{val}(Y_1 \cup Y_2) = \mathsf{val}(Y_1) + \mathsf{val}(Y_2)$$

if $Y_1 \cap Y_2 = \emptyset$. It is also clear that $\mathsf{wt}(x) = |\mathsf{set}(x)|$ for $0 \leq x \leq m-1$.

The following lemmas are easy.

Lemma 1.1 *Suppose that $Y_1, Y_2 \subseteq \mathbb{Z}_m$ and $\alpha^{\mathsf{val}(Y_1)} = \beta(\alpha^{\mathsf{val}(Y_2)})^{-1}$. Then*

$$\log_\alpha \beta = (\mathsf{val}(Y_1) + \mathsf{val}(Y_2)) \bmod \mathsf{ord}(\alpha).$$

Lemma 1.2 *Suppose that* $\mathsf{wt}(\log_\alpha \beta) = t$*, where* t *is an even positive integer. Then there exist subsets* $Y_1, Y_2 \subseteq \mathbb{Z}_m$ *such that* $Y_1 \cap Y_2 = \emptyset$, $|Y_1| = |Y_2| = t/2$ *and* $\alpha^{\mathsf{val}(Y_1)} = \beta(\alpha^{\mathsf{val}(Y_2)})^{-1}$.

Lemmas 1.1 and 1.2 are the basis of the following algorithm, independently due to Heiman and Odlyzko [4], which solves the hamming weight t discrete logarithm problem.

Algorithm 1

```
1. INPUT: $\alpha, \beta \in G$, and an even integer $t$
2. For all $Y \subseteq \mathbb{Z}_m$ such that $|Y| = t/2$, compute $\alpha^{\mathsf{val}(Y)}$
3. Sort the list of ordered pairs $(\mathsf{val}(Y), \alpha^{\mathsf{val}(Y)})$ by their second co-ordinates
4. For all $Y \subseteq \mathbb{Z}_m$ such that $|Y| = t/2$, compute $\beta(\alpha^{\mathsf{val}(Y)})^{-1}$
5. Sort the list of ordered pairs $(\mathsf{val}(Y), \beta(\alpha^{\mathsf{val}(Y)})^{-1})$ by their second co-ordinates
6. If possible, find $Y_1, Y_2$ such that $\alpha^{\mathsf{val}(Y_1)} = \beta(\alpha^{\mathsf{val}(Y_2)})^{-1}$.
7. If the previous step is successful, output $\log_\alpha \beta = (\mathsf{val}(Y_1) + \mathsf{val}(Y_2)) \bmod \mathsf{ord}(\alpha)$.
   Otherwise, output fail.
```

Here "fail" means that either $\beta \notin \langle\alpha\rangle$ or $\mathsf{wt}(\log_\alpha \beta) \neq t$. The complexity of Algorithm 1 (neglecting logarithmic factors) is $O\left(\binom{m}{\frac{t}{2}}\right)$. The space requirement is also $O\left(\binom{m}{\frac{t}{2}}\right)$.

2 The Coppersmith algorithms

2.1 Splitting families

Coppersmith's algorithm is summarized in [8, p. 128]. We describe a generalized version of algorithm in terms of a type of combinatorial set system that we define now. Suppose m and t are even integers, $0 < t < m$. An (m,t)*-splitting system* is a pair $(X, \mathcal{B})$ that satisfies the following properties:

1. $|X| = m$, and $\mathcal{B}$ is a set of $\frac{m}{2}$-subsets of X, called *blocks*
2. for every $Y \subseteq X$ such that $|Y| = t$, there exists a block $B \in \mathcal{B}$ such that $|B \cap Y| = t/2$.

We will use the notation $(N; m, t)$-SS to denote an (m,t)-splitting system having N blocks.

Here is a simple construction for splitting systems.

Lemma 2.1 (Coppersmith) *For all even integers* m *and* t *with* $0 < t < m$*, there exists an* $(\frac{m}{2}; m, t)$*-SS.*

Proof. Let $X = \mathbb{Z}_m$ and define

$$B_i = \{i + j \bmod m : 0 \leq j \leq m/2 - 1\}$$

for $i \in \mathbb{Z}_m$. Let $\mathcal{B} = \{B_i : 0 \leq i \leq m/2 - 1\}$. We will show that $(X, \mathcal{B})$ is an (m,t)-splitting system.

Fix any subset $Y \subseteq X$ such that $|Y| = t$. For $i \in \mathbb{Z}_m$, define

$$\nu(i) = |B_i \cap Y| - |(\mathbb{Z}_m \backslash B_i) \cap Y|.$$

If $\nu(0) = 0$, then we are done, so assume that $\nu(0) \neq 0$. It is easy to see that $\nu(i)$ is even for all i, $\nu(m/2) = -\nu(0)$, and $|\nu(i+1) - \nu(i)| \in \{-2, 0, 2\}$ for all i. Therefore there exists some i such that $0 < i < m/2$ and $\nu(i) = 0$. $\square$

Splitting systems can be used to solve the hamming weight t discrete logarithm problem, as follows.

Algorithm 2

1. INPUT: $\alpha, \beta \in G$, an even integer t, and an $(N; m, t)$-SS, $(\mathbb{Z}_m, \mathcal{B})$, where $\mathcal{B} = \{B_i : 0 \leq i \leq N - 1\}$.
2. For $i = 0, \ldots, N-1$, perform the following steps:
   (a) For all $Y \subseteq B_i$ such that $|Y| = t/2$, compute $\alpha^{\mathsf{val}(Y)}$
   (b) Sort the list of ordered pairs $(\mathsf{val}(Y), \alpha^{\mathsf{val}(Y)})$ by their second co-ordinates
   (c) For all $Y \subseteq \mathbb{Z}_m \backslash B_i$ such that $|Y| = t/2$, compute $\beta(\alpha^{\mathsf{val}(Y)})^{-1}$
   (d) Sort the list of ordered pairs $(\mathsf{val}(Y), \beta(\alpha^{\mathsf{val}(Y)})^{-1})$ by their second co-ordinates
   (e) If possible, find Y_1, Y_2 such that $\alpha^{\mathsf{val}(Y_1)} = \beta(\alpha^{\mathsf{val}(Y_2)})^{-1}$.
   (f) If the previous step is successful, output $\log_\alpha \beta = \mathsf{val}(Y_1 \cup Y_2) \bmod \mathsf{ord}(\alpha)$ and QUIT. Otherwise, proceed to the next iteration of the FOR loop.

The complexity of Algorithm 2 is $O\left(N\binom{\frac{m}{2}}{\frac{t}{2}}\right)$. The space requirement is $O\left(\binom{m}{\frac{t}{2}}\right)$, which does not depend on N. Using the splitting systems from Lemma 2.1 yields an algorithm having complexity $O\left(m\binom{\frac{m}{2}}{\frac{t}{2}}\right)$; this is the algorithm that was presented in [8, p. 128].

2.2 A randomized algorithm

A Las Vegas algorithm with good average-case complexity is easy to construct. This algorithm is also due to Coppersmith [2].

Algorithm 3

```
1. INPUT: $\alpha, \beta \in G$, and an even integer $t$.
2. REPEAT the following steps:
    (a) Let $B$ be a random $\frac{m}{2}$-subset of $X$
    (b) For all $Y \subseteq B$ such that $|Y| = t/2$, compute $\alpha^{\mathsf{val}(Y)}$
    (c) Sort the list of ordered pairs $(\mathsf{val}(Y), \alpha^{\mathsf{val}(Y)})$ by their second co-ordinates
    (d) For all $Y \subseteq \mathbb{Z}_m \backslash B$ such that $|Y| = t/2$, compute $\beta(\alpha^{\mathsf{val}(Y)})^{-1}$
    (e) Sort the list of ordered pairs $(\mathsf{val}(Y), \beta(\alpha^{\mathsf{val}(Y)})^{-1})$ by their second co-ordinates
    (f) If possible, find $Y_1, Y_2$ such that $\alpha^{\mathsf{val}(Y_1)} = \beta(\alpha^{\mathsf{val}(Y_2)})^{-1}$.
    (g) If the previous step is successful, output $\log_\alpha \beta = \mathsf{val}(Y_1 \cup Y_2) \bmod \mathsf{ord}(\alpha)$ and
        QUIT. Otherwise, proceed to the next iteration of the REPEAT loop.
```

The complexity of Algorithim 3 is analyzed as follows. In any iteration, the algorithm is successful if $|B \cap \mathsf{set}(\log_\alpha \beta)| = t/2$. This happens with probability

$$p = \frac{\binom{t}{\frac{t}{2}}\binom{m-t}{\frac{m-t}{2}}}{\binom{m}{\frac{m}{2}}}.$$

We can compute a lower bound on p using the following lemma.

Lemma 2.2 *[6, p. 309] Suppose that n and λn are positive integers, where $0 < \lambda < 1$. Define*

$$H(\lambda) = -\lambda \log_2 \lambda - (1-\lambda)\log_2(1-\lambda).$$

Then

$$\frac{1}{\sqrt{8n\lambda(1-\lambda)}} 2^{nH(\lambda)} \leq \binom{n}{\lambda n} \leq \frac{1}{\sqrt{2\pi n\lambda(1-\lambda)}} 2^{nH(\lambda)}.$$

Now, applying Lemma 2.2, it is easy to see that

$$p \geq \sqrt{\frac{\pi}{8}}\sqrt{\frac{m}{t(m-t)}} > ct^{-1/2}. \tag{1}$$

Hence, the complexity Algorithm 3 is $O\left(\sqrt{t}\binom{\frac{m}{2}}{\frac{t}{2}}\right)$.

3 Average-case analysis of the deterministic algorithm

Suppose we use Algorithm 2 with the splitting systems from Lemma 2.1. We consider the average-case complexity of this algorithm, where the average is computed over all $\binom{m}{t}$ possible exponents having hamming weight t. For any integer x with $0 \leq x \leq 2^m - 1$,

$\mathsf{wt}(x) = t$, let $\psi(x)$ denote the minimum integer $i \geq 0$ such that $|B_i \cap \mathsf{set}(x)| = t/2$. It follows from Lemma 2.1 that $0 \leq \psi(x) \leq m/2 - 1$ for all x. Next, define

$$\delta(m,t) = \sum_{\{x : 0 \leq x \leq 2^m - 1, \mathsf{wt}(x) = t\}} \psi(x).$$

Then the average-case complexity of the algorithm is in fact $O\left(\frac{\delta(m,t)\binom{\frac{m}{2}}{\frac{t}{2}}}{\binom{m}{t}}\right)$.

We proceed to develop a formula for $\delta(m,t)$. For any integer h such that $0 \leq h \leq m/2-1$, we determine the value

$$\eta(h) = |\{x : 0 \leq x \leq 2^m - 1, \mathsf{wt}(x) = t, \psi(x) = h\}|.$$

Then it is clear that

$$\delta(m,t) = \sum_{h=1}^{m/2-1} h\,\eta(h).$$

First, it is easy to see that

$$\eta(0) = \binom{\frac{m}{2}}{\frac{t}{2}}^2.$$

Next, we have that $\psi(x) = 1$ if and only if

$$\begin{array}{l} x_0 = 0 \\ |\{1, \ldots, m/2-1\} \cap \mathsf{set}(x)| = t/2 - 1 \\ x_{m/2} = 1, \text{ and} \\ |\{m/2+1, \ldots, m-1\} \cap \mathsf{set}(x)| = t/2; \end{array}$$

or

$$\begin{array}{l} x_0 = 1 \\ |\{1, \ldots, m/2-1\} \cap \mathsf{set}(x)| = t/2 \\ x_{m/2} = 0, \text{ and} \\ |\{m/2+1, \ldots, m-1\} \cap \mathsf{set}(x)| = t/2 - 1. \end{array}$$

From this it follows that

$$\eta(1) = 2\binom{\frac{m}{2}-1}{\frac{t}{2}-1}\binom{\frac{m}{2}-1}{\frac{t}{2}}.$$

Now, let's look at computing $\eta(h)$ for general h. Suppose the bit-sequences $[x_0, \ldots, x_{h-1}]$ and $[x_{m/2}, \ldots, x_{m/2+h-1}]$ are fixed. Then it is clearly necessary that the following *sum conditions* hold for $0 \leq k \leq h-1$:

$$\sum_{j=h-1-k}^{h-1} x_j \neq \sum_{j=m/2+h-1-k}^{m/2+h-1} x_j. \tag{2}$$

Denote

$$s_1 = \sum_{j=0}^{h-1} x_j$$

and

$$s_2 = \sum_{j=m/2}^{m/2+h-1} x_j.$$

Then $s_1 \neq s_2$, and $\psi(x) = h$ if and only if

$$|\{h, \ldots, m/2-1\} \cap \mathsf{set}(x)| = t/2 - s_1$$

and

$$|\{m/2+h, \ldots, m-1\} \cap \mathsf{set}(x)| = t/2 - s_2.$$

Let $\zeta(h, s_1, s_2)$ denote the number of ways of choosing $x_0, \ldots, x_{h-1}$ and $x_{m/2}, \ldots, x_{m/2}+h-1$ such that the inequality (2) holds for $0 \leq k \leq h-1$. Then, by the discussion above, we have that

$$\eta(h) = \sum_{s_1=0}^{h} \sum_{s_2=0}^{h} \zeta(h, s_1, s_2) \binom{\frac{m}{2}-h}{\frac{t}{2}-s_1} \binom{\frac{m}{2}-h}{\frac{t}{2}-s_2}.$$

Thus, it remains to find a formula for $\zeta(h, s_1, s_2)$. We do this using the familiar "reflection" technique that can be used to determine a formula for the Catalan numbers (see, e.g., [5, §3.4]).

For $0 \leq i \leq h-1$, define $z_{h-i} = x_i - x_{m/2+i}$. Then $z_i \in \{0, 1, -1\}$ for all i. Inequality (2) can then be rewritten as follows:

$$\sum_{j=1}^{i} z_j \neq 0 \tag{3}$$

for $1 \leq i \leq h$.

Given the sequence $[z_1, \ldots, z_h]$, we define a path $P = [(0, y_0), (1, y_1), \ldots, (h, y_h)]$, where $y_0 = 0$ and $y_i - y_{i-1} = z_i$ for $1 \leq i \leq h$. Observe that $y_h = s_1 - s_2$. Also, inequality (3) can be interpreted as saying that the path P never hits the x-axis, except for the initial point, $(0, 0)$.

For $j_1, j_2 \in \{0, 1\}$, define

$$a_{j_1, j_2} = |\{i : (x_i, x_{m/2+i}) = (j_1, j_2)\}|.$$

Note that a type $(1, 0)$ pair correpsonds to an "up" edge in P, a type $(0, 1)$ pair correpsonds to a "down" edge in P, and type $(0, 0)$ and $(1, 1)$ pairs correpsond to "horizontal" edges in P. We will think of each edge of P as being labelled with an ordered pair in this manner; this will allow the sequences $[x_0, \ldots, x_{h-1}]$ and $[x_{m/2}, \ldots, x_{m/2+h-1}]$ to be recovered from P.

It is easy to see that the following equations hold:

$$\begin{aligned} a_{0,0} + a_{1,1} + a_{1,0} + a_{0,1} &= h, \\ a_{1,1} + a_{0,1} &= s_2, \quad \text{and} \\ a_{1,1} + a_{1,0} &= s_1. \end{aligned}$$

Then

$$(a_{0,0}, a_{1,1}, a_{1,0}, a_{0,1}) = (h + j - s_1 - s_2, j, s_1 - j, s_2 - j),$$

where j is an integer.

Let us now assume that $s_1 > s_2$ (the case $s_2 > s_1$ can be analyzed in a similar fashion). Then the first edge of P must be labelled $(1,0)$, otherwise (3) will be violated for $i = 1$. The total number of such paths P is given by the multinomial coefficient

$$\binom{h-1}{h+j-s_1-s_2,\, j,\, s_1-j-1,\, s_2-j}.$$

Of course, this total includes paths that do not satisfy (3). Now, suppose that (3) is violated for some $i > 1$; let i_0 be the smallest such i. Form a path P^* by reflecting the initial portion of P (from $(0,0)$ to $(i_0,0)$) in the x-axis. P^* is a path from $(0,0)$ to $(h, s_1 - s_2)$ in which the initial edge is labelled $(0,1)$. Also, the values $(a_{0,0}, a_{1,1}, a_{1,0}, a_{0,1})$ are the same in P^* as they are in P. The total number of such paths P^* is given by the multinomial coefficient

$$\binom{h-1}{h+j-s_1-s_2,\, j,\, s_1-j,\, s_2-j-1}.$$

Therefore, it follows that the number of paths P that satisfy all the inequalities (3) is

$$\binom{h-1}{h+j-s_1-s_2,\, j,\, s_1-j-1,\, s_2-j} - \binom{h-1}{h+j-s_1-s_2,\, j,\, s_1-j,\, s_2-j-1},$$

which simplifies to give

$$\frac{s_1-s_2}{h}\binom{h}{h+j-s_1-s_2,\, j,\, s_1-j,\, s_2-j}.$$

Thus, for $h \neq 0$, it holds that

$$\zeta(h, s_1, s_2) = \frac{|s_1-s_2|}{h} \sum_{j=\max\{s_1+s_2-h,0\}}^{\min\{s_1,s_2\}} \binom{h}{h+j-s_1-s_2,\, j,\, s_1-j,\, s_2-j}. \tag{4}$$

The sum in (4) can be simplified, as follows:

$$\begin{aligned}
\sum_j \binom{h}{h+j-s_1-s_2,\, j,\, s_1-j,\, s_2-j} &= \sum_j \binom{h-s_2}{s_1-j}\binom{h-j}{h-s_2}\binom{h}{h-j} \\
&= \sum_j \binom{h-s_2}{s_1-j}\binom{h}{s_2}\binom{s_2}{s_2-j} \\
&= \binom{h}{s_2}\sum_j \binom{h-s_2}{s_1-j}\binom{s_2}{j} \\
&= \binom{h}{s_2}\binom{h}{s_1}.
\end{aligned}$$

Combining everything, we get the following formula:

$$\delta(m,t) = \sum_{h=1}^{m/2-1} \sum_{s_1=1}^{\min\{h,t/2\}} \sum_{s_2=0}^{s_1-1} 2(s_1-s_2)\binom{\frac{m}{2}-h}{\frac{t}{2}-s_1}\binom{\frac{m}{2}-h}{\frac{t}{2}-s_2}\binom{h}{s_1}\binom{h}{s_2}. \tag{5}$$

We are unable to simplify (5) any further. However, computational evidence show that the speed-up is, at best, only a constant factor. In order to compare the average-case to the worst-case complexity, we compute the ratio

$$r(m,t) = \frac{2\,\delta(m,t)}{m\binom{m}{t}}$$

for various values of m and t. It is clear from the definition of the function δ that $\delta(m,t) = \delta(m, m-t)$, so it suffices to restrict t so that $2 \le t \le m/2$. We computed all these ratios $r(m,t)$ for even values of m and t such that $2 \le t \le m/2$ and $4 \le m \le 80$. We found that the values $r(m,2)$ decrease as m increases; the values $r(m, m/2)$ increase as m increases; and, for any fixed value of m, the values $r(m,t)$ increase as t increases from 2 to $m/2$.

The following table lists values of $\delta(m,t)$ and $r(m,t)$ for $m \le 16$ and $t \le m/2$; and for $m \in \{24, 32, 40, 48, 56, 64, 72, 80\}$ when $t = m/2$.

m	t	$\delta(m,t)$	$r(m,t)$
4	2	2	.166667
6	2	8	.177778
8	2	20	.178571
8	4	56	.200000
10	2	40	.177778
10	4	216	.205714
12	2	70	.176768
12	4	616	.207407
12	6	1188	.214286
14	2	112	.175824
14	4	1456	.207792
14	6	4576	.217687
16	2	168	.175000
16	4	3024	.207692
16	6	14040	.219156
16	8	22880	.222222
24	12	7488432	.230769
32	16	2262890880	.235294
40	20	656412042000	.238095
48	24	185746197214656	.240000
56	28	51694598543070560	.241379
64	32	14216720608524338688	.242424
72	36	3874974677018786931408	.243243
80	40	1048850816910596843528000	.243902

It is easy to see from equation (5) that $\delta(m,2) = (m^3 - 4m)/24$. Hence $r(m,2) \to 1/6$ as $m \to \infty$. It is an interesting open problem to compute $\lim_{m\to\infty} r(m, m/2)$.

4 Improved results concerning splitting systems

4.1 Probabilistic Methods

We can improve Algorithm 2 by constructing smaller splitting systems. We first provide a bound using probabilistic methods. Let m and t be even integers such that $0 < t < m$. Suppose that $\mathcal{B}$ a set of $\frac{m}{2}$-subsets of an m-set, X, and $|\mathcal{B}| = N$. For a subset $Y \subseteq X$ with $|Y| = t$, define $Z_Y(\mathcal{B}) = 0$ if there exists a block $B \in \mathcal{B}$ such that $|B \cap Y| = t/2$, and define $Z_Y(\mathcal{B}) = 1$, otherwise. Let Z_Y denote the random variable obtained by letting $\mathcal{B}$ a set of N randomly chosen $\frac{m}{2}$-subsets of X. Clearly, the expected value of Z_Y, denoted $E[Z_Y]$, is $(1-p)^N$, where

$$p = \frac{\binom{t}{\frac{t}{2}}\binom{m-t}{\frac{m-t}{2}}}{\binom{m}{\frac{m}{2}}}.$$

If we define the random variable

$$Z = \sum_{\{Y \subseteq X : |Y| = t\}} Z_Y,$$

then we have $E[Z] = \binom{m}{t}(1-p)^N$. It is clear that there exists an $(N; m, t)$-SS if $E[Z] < 1$. Since $\binom{m}{t} < m^t$, this will be true if

$$t \log m + N \log(1-p) < 0,$$

which is equivalent to

$$N > \frac{t \log m}{-\log(1-p)}.$$

Using elementary calculus, we have that $-\log(1-p) > p$; and $p \geq ct^{-1/2}$ was shown in Equation (1). Hence, an $(N; m, t)$-SS exists if

$$N > \frac{1}{c} t^{3/2} \log m.$$

Thus we have proven the following result.

Theorem 4.1 *For any even integers t and m with $0 < t < m$, there exists an $(N; m, t)$-SS with $N \approx c_0 t^{3/2} \log_2 m$, where c_0 is a constant.*

Thus, Theorem 4.1 yields a deterministic algorithm having complexity $O\left(t^{3/2} (\log m) \binom{\frac{m}{2}}{\frac{t}{2}}\right)$.

4.2 Explicit Constructions

In this section, we present a recursive construction for splitting families that uses perfect hash families. Perfect hash families were introduced by Mehlhorn (see, e.g., [7]) and have been studied extensively since then (for a recent survey, see [3]).

We require some definitions. Let $n \geq m$ be positive integers. An (n, m)-*hash function* is a function $h : A \to B$, where $|A| = n$ and $|B| = m$. The hash function h is said to be *balanced* provided that $|h^{-1}(y)| = n/m$ for all $y \in B$.

Let n, m and w be integers such that $n \geq m \geq w \geq 2$. An (n, m, w)*-perfect hash family* is a finite set $\mathcal{H}$ of (n, m)-hash functions such that $h : A \rightarrow B$ for each $h \in \mathcal{H}$, where $|A| = n$ and $|B| = m$, with the property that for any $X \subseteq A$ with $|X| = w$, there exists at least one $h \in \mathcal{H}$ such that $h|_X$ is one-to-one. $\mathcal{H}$ is said to be an (n, m, w)*-balanced perfect hash family* if $\mathcal{H}$ is an (n, m, w)-perfect hash family and h is balanced for every $h \in \mathcal{H}$.

We will use the notation BPHF$(N; n, m, w)$ to denote an (n, m, w)-balanced perfect hash family with $|\mathcal{H}| = N$. We can depict a BPHF$(N; n, m, w)$ as an $N \times n$ array on m symbols, say A, where each row of A corresponds to one of the functions in the family. This array has the property that, for any subset of w columns, there exists at least one row such that the entries in the w given columns of that row are distinct; and any row of A contains exactly n/m occurrences of each symbol.

Here is a recursive construction for splitting families.

Theorem 4.2 *Suppose there exist a BPHF$(N_0; n, m, t)$ and an SS$(N_1; m, t)$. Then there exists an SS$(N_0 N_1; n, t)$.*

Proof. Let M be the $N_1 \times m$ incidence matrix of an SS$(N_1; m, t)$, and denote the columns of M by $c_1, \ldots, c_m$. Let A be the array representation of the BPHF$(N_0; n, m, t)$, and replace each entry $y = A(i, j)$ by the column vector c_y. Call the resulting matrix M_1.

It is easy to see that M_1 is the incidence matrix of an SS$(N_0 N_1; n, t)$: The "balance" property of the hash family ensures that each block of the resulting splitting system has cardinality $n/2$. Also, given a t-subset of points, say B_1, there exists a hash function h such that $h|_{B_1}$ is injective. Restricting to the N_1 corresponding rows of M_1, property 2. of splitting families is inherited from M. $\square$

The following corollary is an immediate application of Lemma 1.1 and Theorem 4.2.

Corollary 4.3 *If there exists a BPHF$(N_0; n, m, t)$, then there exists an SS$(N_0 m/2; n, t)$.*

In order to apply Theorem 4.2 or Corollary 4.3. we need balanced perfect hash families. It is not difficult to verify that certain direct constructions for perfect hash families in the literature yield BPHF. We illustrate with an example.

Let q be a prime power. An (N, K, D, q)*-code* is a set $\mathcal{C}$ of K vectors in $(\mathbb{F}_q)^N$ such that the Hamming distance between any two distinct vectors in $\mathcal{C}$ is at least D. The code $\mathcal{C}$ is *linear* if it is a subspace of $(\mathbb{F}_q)^N$; in this case $K = q^k$, where $k = \dim(\mathcal{C})$.

Theorem 4.4 *If a linear (N, K, D, q)-code exists, then there exists a BPHF$(N; K, q, w)$, provided that*

$$\frac{D}{N} > 1 - \frac{1}{\binom{w}{2}}.$$

Proof. Construct an $N \times K$ array whose columns are the codewords in $\mathcal{C}$. It is shown in [1] that this array is a PHF$(N; K, q, w)$ provided that $D/N > 1 - (1/\binom{w}{2})$. Since $\mathcal{C}$ is linear, it follows that each hash function in the family is balanced, and the result follows. $\square$

Using Reed-Solomon codes, we obtain the following corollary of Theorem 4.4.

Corollary 4.5 *Suppose that q is a prime power, $0 < \ell < q$ is an integer, and $q > (\ell-1)\binom{w}{2}$. Then there exists a BPHF$(q; q^\ell, q, w)$.*

Proof. A q-ary dimension ℓ extended Reed-Solomon code of length q exists. This is a linear $(q, q^\ell, q-\ell+1, q)$-code. Apply Theorem 4.4. □

Combining Corollaries 4.3 and 4.5 allows us to prove an interesting asymptotic existence theorem. Suppose m and t are given, and we want to construct an SS$(N; m, t)$. Choose $q \approx t^2 \log m$ and $\ell \approx \log m / \log q$. Then all necessary conditions are satisfied, and we obtain an SS$(N; m, t)$ in which N is $O(t^4 (\log m)^2)$.

5 Conclusion

We have described several varaints of baby-step giant-step algorithms for the low hamming weight discrete logarithm problem. For practical use, Coppersmith's Las Vegas algorithm (Algorithm 3) would be preferred. If a deteminisitic algorithm is desired, then an algorithm based on the idea of splitting systems can be used. This is a generalization of another algorithm due to Coppersmith. We performed an average case analysis of the simplest of these algorithms and found that only a constant factor speedup is acheived, as compared to the worst case. Several alternative methods of constructing splitting systems were investigated. These permit construction of smaller splitting systems, and hence more efficient determinstic algorithms, at least asymptotically.

Acknowledgements

The author's research is supported by the Natural Sciences and Engineering Research Council of Canada through the following grants: NSERC-IRC #216431-96 and NSERC-RGPIN #203114-98.

I would like to thank Ruizhong Wei for his help with computations and for his useful comments, and Alfred Menezes for his assistance with references.

References

[1] N. Alon. Explicit construction of exponential sized families of k-independent sets, *Discrete Math.* **58** (1986), 191–193.

[2] D. Coppersmith. Private communcation to Scott Vanstone, December 1997.

[3] Z. J. Czech, G. Havas and B. S. Majewski. Perfect hashing, *Theoretical Computer Science* **182** (1997), 1–143.

[4] R. Heiman. A note on discrete logarithms with special structure. *Lecture Notes in Computer Science* **658**, 454–457 (Advances in Cryptology – EUROCRYPT '92).

[5] D. L. Kreher and D. R. Stinson. *Combinatorial Algorithms: Generation, Enumeration and Search*, CRC Press, 1999.

[6] F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes*, North-Holland, 1977.

[7] K. Mehlhorn. *Data Structures and Algorithms 1: Sorting and Searching*, Springer-Verlag, Berlin, 1984.

[8] A. J. Menezes, P. C. van Oorschot and S. A. Vanstone. *Handbook of Applied Cryptography*, CRC Press, 1996.