A Comparison of Three Algorithms for the Discrete Log Problem

Weston Kirk

December 2025

1 Introduction

The DLP is a foundational problem in cryptography, underlying applications such as the Diffie-Hellman key exchange protocol and Elgamal public-key cryptosystem. Given a group $G = \mathbb{F}_p^*$ with multiplication, $g \in \mathbb{F}_p$ for prime p, and $h \in \mathbb{F}_p^*$, the Discrete Log Problem (DLP) is finding x such that,

$$g^x \equiv h \pmod{p}, \tag{1.1}$$

where $x = \log_g(h) \in \{0, 1, ..., p-2\}$ is called the discrete logarithm of h to the base g. Furthermore, assume that the prime factorization of N,

$$N = p_1^{e_1} p_2^{e_2} \cdots p_t^{e_t},$$

is easily computable. Let $\text{ord}(g)$ denote the order of g. For the purpose of this paper, suppose $\text{ord}(g) = N > 2$, since the $\text{ord}(g) = 1, 2$ cases are trivial for the DLP.

In this paper, I describe three algorithms used to solve the DLP, in particular Shanks's Baby-step Giant-step Algorithm (Section 2), Pollard's Kangaroo Method (Section 3), and the Pohlig-Hellman Algorithm (Section 4). Currently, the Discrete Log Problem is considered computationally hard, and the security of said applications depends on this assumption.

For each algorithm, I describe its general strategy, then derive its respective time complexity. I will also discuss when each algorithm is used in practice with respect to real-world vulnerabilities and constraints (Section 5). Lastly, an example computation for each algorithm is provided (Section 6).

2 Shanks's Baby-step Giant-step Algorithm

Shanks's Baby-step Giant-step Algorithm is a collision algorithm used to solve for x. This presentation follows [HPS08] and [Som25]. To see how it

works, $m = \left\lceil \sqrt{N} \right\rceil$, then build off (1.1). By the Division Algorithm, it is true that there exists unique $q, r \in \mathbb{Z}$ such that,

$$x = qm + r \quad \text{and} \quad 0 \leq r < m. \tag{2.1}$$

Substituting (2.1) into (1.1), it is true that,

$$\begin{aligned} g^{qm+r} &\equiv h \pmod{p} \\ g^{qm} \cdot g^{r} &\equiv h \pmod{p} \\ g^{r} &\equiv h \cdot g^{-qm} \pmod{p}. \end{aligned} \tag{2.2}$$

Now, find $q, r \in \mathbb{Z}$ which satisfies (2.2). It is true that, $qm \leq x$ by definition and

$$m = \left\lceil \sqrt{N} \right\rceil \implies m^2 > N,$$

where $x < N$ since g is a primitive root. Combining these inequalities, it is true that

$$qm \leq x < N < m^2 \implies qm < m^2 \implies q < m. \tag{2.3}$$

From (2.1) and (2.3), $r < m$ and $q < m$, so there are m values to check, denoted i. This task is split into two lists,

r	0	1	2	...	$m-1$
g^r	e	g	g^2	...	g^{m-1}

q	0	1	2	...	$m-1$
hg^{-qm}	h	hg^{-m}	hg^{-2m}	...	hg^{-m^2+m}

where there will be some r, q such that $g^r \equiv hg^{-qm} \pmod{p}$. To find these r, q, a sorting algorithm can be used on one list, then a standard search algorithm may be used to find a match in the other [HPS08]. At this point, recall $x = qm + r$, where m is defined, and q, r have been found. For an example, see section 5.1.

For each list in Shanks's Baby-step Giant-step Algorithm, there are $m \approx \sqrt{N}$ computations made for each list. Then, the total number of operations to create the two lists can be written as,

$$2m \approx 2\sqrt{N}$$

along with one calculation to compute m and one to compute x. Once the two lists are generated, according to [HPS08], the standard sorting and searching algorithm used to find a match requires,

$$m \log m \approx \sqrt{N} \log \sqrt{N} = \sqrt{N} \log \sqrt{N} = \frac{1}{2}\sqrt{N} \log N$$

steps, which implies, $\mathcal{O}(\sqrt{N}\log N)$ time complexity. In terms of storage complexity, each of the $2m \approx 2\sqrt{N}$ must be stored for the matching process, so the storage complexity can be written as,

$$2\sqrt{N} \implies \mathcal{O}(\sqrt{N}).$$

3 Pollard's Kangaroo Method

Pollard's Kangaroo Method is a probabilistic collision algorithm used to solve the DLP when we know that $x \in [a, b]$. First, define a function $f : G \to J$ which randomly assigns a group element $x \in G$ to a step size $f(x) \in J$, where J is a set of possible step sizes. Standard implementations such as [MT09, VOW94] define J as successive powers of 2,

$$J = \{1, 2, 4, 8, ..., 2^j\}, \tag{3.1}$$

for some $j \in \mathbb{N}$. From (3.1), let m be the mean jump size, and $L = 2^j$ be the largest element. Now, consider two kangaroos, one tame, denoted T, and one wild, denoted W. Let $T_0 = g^b$, and define,

$$T_{i+1} = T_i \cdot g^{f(T_i)}.$$

The tame kangaroo will complete $k \in \mathbb{N}$ jumps. Similarly, let $W_0 = h$, and define

$$W_{i+1} = W_i \cdot g^{f(W_i)}.$$

Definition 3.1: To find the distance traveled over n steps, define,

$$d_{X,n} = \sum_{i=0}^{n-1} f(X_i),$$

where $X \in \{T, W\}$.

Note that in contrast to T, W takes an unknown number of steps, which we denote as $\ell \in \mathbb{N}$, terminating when one of two possible conditions is met,

1. $W_\ell \equiv T_k \pmod{p}$, meaning that the two kangaroos have collided at the tame kangaroo's endpoint
2. $d_{W,\ell} > b + d_{T,k} - a$, meaning that the wild kangaroo has gone beyond the tame kangaroo's endpoint

In case 1, W_ℓ and T_k can be expressed in terms of distance from the initial respective points,

$$g^{x+d_{W,\ell}} \equiv g^{b+d_{T,k}} \pmod{p}.$$

Then, since g is a primitive root,

$$x + d_{W,\ell} \equiv b + d_{T,k} \pmod{p-1},$$

meaning that x can be isolated,

$$x \equiv b + d_{T,k} - d_{W,\ell} \pmod{p-1}. \tag{3.2}$$

Thus, a collision at the tame endpoint ensures a solution to the DLP. Based on Pollard's original paper, defining $N = \theta m$ for some $\theta \in \mathbb{Z}^+$, then the probability of a collision is,

$$1 - \left(1 - \frac{1}{m}\right)^N,$$

which for $\theta = 4$ indicates a 98% chance of a collision.

When $x \in [a, b]$, then the time complexity of Pollard's Kangaroo is,

$$\mathcal{O}(\sqrt{b-a}),$$

according to [Pol78] [MT09]. Furthermore, the space complexity is $\mathcal{O}(1)$, since all that needs to be stored is the position of the wild and tame kangaroo, their respective distances, and the step function J.

4 Pohlig-Hellman Algorithm

The Pohlig-Hellman Algorithm solves the DLP via decomposition of $\text{ord}(g) = N$ into prime powers, following the presentation in [HPS08]. Since $\text{ord}(g) = N$, the solution x to $g^x = h$ will be determined modulo N. Namely N can be factorized as,

$$N = \prod_{i=1}^{k} p_i^{e_i}$$

where each p_i is prime. For each of the k prime powers, define $A_i = \frac{N}{p_i^{e_i}}$, and let

$$g_i = g^{A_i} \quad \text{and} \quad h_i = h^{A_i},$$

which yields a smaller DLP within the prime power subgroup, written as,

$$g_i^{y_i} = h_i.$$

Observe that $\text{ord}(g_i) = p_i^{e_i}$ [HPS08], so the solution y_i to $g_i^{y_i} = h_i$ will be determined modulo $p_i^{e_i}$. A key assumption of the Pohlig-Hellman algorithm is that for each of these smaller powers, the DLP can be solved in $\mathcal{O}\left(\sqrt{p_i^{e_i}}\right)$ steps using existing methods such as Shank's Baby-step Giant-step Algorithm.

Once each of the k smaller DLP problems are solved, the Chinese Remainder Theorem may be employed to find x such that,

$$\begin{aligned} x &\equiv y_1 \pmod{p_1^{e_1}} \\ &\vdots \\ x &\equiv y_k \pmod{p_k^{e_k}}, \end{aligned}$$

which solves the original DLP for x modulo N. The final step takes $\mathcal{O}(\log N)$ steps [HPS08]. According to [HPS08], the total runtime of Pohlig-Hellman is,

$$\mathcal{O}\left(\sum_{i=1}^{k} \sqrt{p_i^{e_i}} + \log N\right).$$

Furthermore, when Shank's Baby-step Giant-step Algorithm is used to solve the smaller DLP problems, the space complexity of Pohlig-Hellman is $\mathcal{O}\left(\sqrt{p_{\max}}\right)$ where $p_{\max} = \max(p_1, ..., p_k)$ [Sut17].

5 Tradeoffs and Cryptographic Connections

To summarize previous sections, below are the time and storage complexities for each of the three algorithms,

	Shanks's	Pollard's Kangaroo	Pohlig-Hellman
Time	$\mathcal{O}(\sqrt{N}\log N)$	$\mathcal{O}\left(\sqrt{b-a}\right)$	$\mathcal{O}\left(\sum_{i=1}^{k} \sqrt{p_i^{e_i}} + \log N\right)$
Storage	$\mathcal{O}(\sqrt{N})$	$\mathcal{O}(1)$	$\mathcal{O}\left(\sqrt{p_{\max}}\right)$

Each of the three algorithms covered exhibit different strengths based on group structure and known information. Shanks's Baby-step Giant-step Algorithm and the Pohlig-Hellman Algorithm are both deterministic algorithms, while Pollard's Kangaroo is a probabilistic algorithm, which is only useful in the specialized case when it is known that $x \in [a, b]$.

Shanks's Baby-step Giant-step Algorithm is a deterministic, generic algorithm, meaning that it depends only on N, and serves as an excellent general approach, and first attack, to the DLP.

Definition 5.1: An X-bit key has an approximated maximum security strength of X bits, which takes on average 2^{X-1} units of time, (in this case, operations) to attack successfully. [Bar20].

According to a 2020 Special report from the National Institute of Standards and Technology [Bar20], an example of acceptable security strength is 128, meaning that 2^{128} units of time are required to ensure its safety. In the version of Shanks's Baby-step Giant-step Algorithm presented in Section 2, I derived a time complexity of $\mathcal{O}(\sqrt{N}\log N)$, following the presentation in [HPS08].

However, alternative presentations derive $\mathcal{O}\sqrt{N}$ complexity, which is a more conservative estimate for cryptographic applications. So, to ensure a safe algorithm, it must be the case that,

$$\sqrt{N} > 2^{128} \implies \text{ord}(g) = N > 2^{256}.$$

Furthermore, Shanks's Baby-step Giant-step Algorithm is used to solve relatively small DLP problems, (prime powers of large N), due to its performance guarantees. In fact, [Sut17] demonstrates that any generic Las Vegas algorithm for the DLP (meaning any algorithm that only uses the group operation and equality tests, may use randomness) usea an expected $\Omega(\sqrt{N})$ group operations. In essence, Shanks's algorithm is optimal among generic methods.

Pollard's Kangaroo is a probabilistic collision algorithm which represents superior time and space complexity compared to Shanks's Baby-step Giant-step Algorithm and the Pohlig-Hellman Algorithm when $x \in [a, b]$ for a reasonably small $b - a$. The original [Pol78] version of Pollard's Kangaroo presented in (Section 3) only stores one possible collision point, which is the endpoint of the tame kangaroo. Modified versions of Pollard's Kangaroo [MT09, VOW94] define a set of distinguished points, which are elements of the group with a common condition, such as having a particular number of zero leading bits, thus increasing the chances for a collision. Including a set of distinguished points is in fact more efficient in terms of time complexity than the original implementation according to [MT09], though more distinguished points must be stored. In most cases, this will still be far more efficient than the $\mathcal{O}(\sqrt{N})$ and $\mathcal{O}\left(\sqrt{p_{\max}}\right)$ storage required in Shanks's Baby-step Giant-step Algorithm and the Pohlig-Hellman Algorithm, respectively.

Pohlig-Hellman's Algorithm is a deterministic algorithm which is is highly dependent on the factorization of N. Namely, how large $p_{\max}$ strongly determines time and space complexity. If $p_{\max}$ is very large, say a prime near N, then the efficiency of Pohlig-Hellman is signifiantly diminished. On the other hand, if $p_{\max}$ is quite small, as in Example 6.3 which was 3-smooth, then Pohlig-Hellman has very efficient time and space complexity. This efficiency equates to a threat for cryptosystems, however, recalling the earlier security discussion. In the original 1978 "An Improved Algorithm for Computing Logarithms over GP(p) and Its Cryptographic Significance" by Pohlig and Hellman [PH78], the algorithm is noted to be most efficient when $p - 1$ factors into small primes. To avoid this, so Pohlig-Hellman cannot break cryptosystems, then the prime p selected must be "safe" where $p = 2p_{\max} + 1$ according to [PH78] [MvOV96], which in practice means $p_{\max} \geq 2^{160}$ [MvOV96].

6 Worked Examples

Consider $G = \mathbb{F}_{19}^*$ with multiplication, so $N = 18$. For primitive root $g = 3$ and $h = 11$, then the DLP to find is $\log_3(11)$ in $\mathbb{F}_{19}$, or

$$3^x = 11 \pmod{19}.$$

6.1 Shank's Baby-step Giant-step Algorithm Worked Example

Let $m = \lceil \sqrt{18} \rceil = 5$. Recall, $x = qm + r$ with $0 \leq r < m$, meaning that the algorithm compares the baby steps, g^r to the giant steps, hg^{-mq}, searching for some r, q such that,

$$g^r \equiv hg^{-qm} \pmod{p}.$$

Each of the "possibilities" 0 through $m-1$ must be checked. For $g^r = 3^r$, this is straightforward. However, the term $hg^{-qm} = 11(3^{-5q})$ must be simplified prior to computation,

$$11 \cdot 3^{-5q} = 11 \cdot \left(3^{-1}\right)^{5q} = 11 \cdot 13^{5q},$$

where $3^{-1} \equiv 13$ is found by using the Extended Euclidean Algorithm.

r	0	1	2	3	4
3^r	1	3	9	8	5

q	0	1	2	3	4
$11 \cdot 13^{5q}$	11	2	9	12	16

From this, it is clear that $3^2 \equiv 11(3^{-5(2)}) \equiv 9 \pmod{19}$. Because $x = qm + r \implies x = 2(5) + 2 = 12$, the DLP is solved by,

$$3^{12} \equiv 531441 \equiv 11 \pmod{19},$$

as required.

6.2 Pollard's Kangaroo Method Worked Example

In order to demonstrate Pollard's Kangaroo method, suppose it is known that $x \in [11, 14]$. Furthermore, consider a relatively simple set of jump sizes, and define $f : G \to \{1, 2\}$, where,

$$f(x) = \begin{cases} 1, & \text{if } 1 \leq x \leq 6, \\ 2, & \text{if } 7 \leq x \leq 18, \end{cases}$$

for group element $x \in G$. Note that f is a random function, and this seemingly non-random function is for the sake of demonstrating a reasonably-scoped example.

The tame kangaroo jumps first, and begins at the upper bound. So,

$$T_0 = g^b = 3^{14} \equiv 4 \pmod{19}.$$

Then, the tame kangaroo only takes 1 jump, so its endpoint is T_1,

$$T_1 = T_0 \cdot 3^{f(T_0)} = 4 \cdot 3^1 = 12.$$

Furthermore, the tame kangaroo's distance is $d_{T,1} = f(T_0) = 1$.

Now, the wild kangaroo begins from position,

$$W_0 = h = 11,$$

and will jump until it either lands on T_1, or "goes beyond" based on the criteria for case 2. Then, the next steps for the wild kangaroo may be computed,

$$\begin{aligned} W_1 &= W_0 \cdot 3^{f(W_0)} = 11 \cdot 3^2 = 99 \equiv 4 \pmod{19} \\ W_2 &= W_1 \cdot 3^{f(W_1)} = 4 \cdot 3^1 = 12 \pmod{19}. \end{aligned}$$

At this point, a collision has occurred, since,

$$W_2 \equiv T_1 \equiv 12 \pmod{19}$$

The wild kangaroo's distance is, $d_{W,2} = f(W_0) + f(W_1) = 2 + 1 = 3$.

Finally, since the collision point is known, x is now easily computed from (3.2),

$$x \equiv 14 + d_{T,1} - d_{W,2} = 14 + 1 - 3 = 12 \pmod{N}.$$

Since $x = 12$, it is clear that the DLP is solved,

$$3^{12} \equiv 531441 \equiv 11 \pmod{19},$$

as required.

6.3 Pohlig-Hellman Algorithm Worked Example

Since $N = 18$, the first step is to factor 18 into prime powers,

$$18 = 2 \cdot 3^2.$$

Then, define $A_1 = \frac{18}{2} = 9$ and $A_2 = \frac{18}{3^2} = 2$, and let,

$$g_1 = 3^9 \equiv 18 \pmod{19} \quad \text{and} \quad h_1 = 11^9 \equiv 1 \pmod{19},$$

and

$$g_2 = 3^2 = 9 \pmod{19} \quad \text{and} \quad h_2 = 11^2 \equiv 7 \pmod{19},$$

Now, the original DLP can be broken into two smaller subproblems,

$$18^{y_1} = 1 \quad \text{and} \quad 9^{y_2} = 7.$$

Using Shanks's Baby-step Giant-step Algorithm yields $y_1 = 2$ and $y_2 = 3$. Now, using the CRT to set up a set of linear congruencies,

$$\begin{aligned} x &\equiv 2 \pmod{2} \\ x &\equiv 3 \pmod{9}, \end{aligned}$$

which yields $x = 12 \pmod{N}$. Since $x = 12$, it is clear that the DLP is solved,

$$3^{12} \equiv 531441 \equiv 11 \pmod{19},$$

as required.

References

[Bar20] Elaine Barker. Recommendation for key management: Part 1 – general. NIST Special Publication 800-57 Part 1 Revision 5, National Institute of Standards and Technology, Gaithersburg, MD, May 2020.

[HPS08] Jeffrey Hoffstein, Jill Pipher, and Joseph H Silverman. *An Introduction to Mathematical Cryptography*. Springer, 2008.

[MT09] Ravi Montenegro and Prasad Tetali. How long does it take to catch a wild kangaroo? In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 553–560, 2009.

[MvOV96] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Public-Key Parameters*, chapter 4. CRC Press, 1996.

[PH78] S. Pohlig and M. Hellman. An improved algorithm for computing logarithms overgf(p)and its cryptographic significance (corresp.). *IEEE Transactions on Information Theory*, 24(1):106–110, 1978.

[Pol78] John M Pollard. Monte carlo methods for index computation (). *Mathematics of computation*, 32(143):918–924, 1978.

[Som25] Rick Sommer. Math 110 lecture 9: Computing discrete log. Course Lecture Notes, Stanford University, October 2025. Math 110: Number Theory for Cryptography.

[Sut17] Andrew V. Sutherland. Lecture 10: Generic algorithms for the discrete logarithm problem. Lecture Notes for 18.783 Elliptic Curves, March 2017. Spring 2017 course lecture notes.

[VOW94] Paul C Van Oorschot and Michael J Wiener. Parallel collision search with application to hash functions and discrete logarithms. In *Proceedings of the 2nd ACM Conference on Computer and Communications Security*, pages 210–218, 1994.