The Pohlig-Hellman Algorithm

Joshua Ottoson

May 12, 2023

1 Introduction

This paper considers an attack on the Discrete Logarithm Problem that was not discussed in class called the Pohlig-Hellman Algorithm. We begin with a restatement of the problem.

Definition 1.1 (The Discrete Logarithm Problem (DLP))**.** [HPS14] Let g be a primitive root for $\mathbb{F}_p$ and let h be a nonzero element of $\mathbb{F}_p$. The DLP is the problem of finding an exponent x such that

$$g^x \equiv h \mod p.$$

The number x is called the *discrete logarithm of h to the base g* and is denoted by $\log_g(h)$.

I would like to write a few reminders for the reader before we begin. The notation $\mathbb{F}_p$ is used to denote a finite field with a prime number of elements, which can equivalently be written as $\mathbb{Z}/p\mathbb{Z}$. A primitive root of $\mathbb{F}_p$ can generate every nonzero element of $\mathbb{F}_p$. In other words, each nonzero element of $\mathbb{F}_p$ can be written as some power of a primitive root, say g. So

$$1, g, g^2, \ldots, g^{p-2}$$

are all the elements of $\mathbb{F}_p$. Recall $g^{p-1} \equiv 1 \mod p$, as primitive roots have order $p-1$. The list of numbers above are most likely *not* in numerical order.

A naive method to solve the DLP is to simply plug in values for x, starting with $x = 1, 2, 3, \ldots$ and so on, where the successive squaring algorithm discussed in class can be used to compute the value g^x modulo some prime p. This brute-force approach works well enough for small values of p, but is not practical when p is very large. We would like to find some "shortcut" method to find x that would require far fewer steps for large values of p.

Such a shortcut was first discovered by Roland Silver, but first published in a paper by Martin Hellman and Stephen Pohlig in 1978 [HP78]. They describe a procedure to significantly speed up the computation of the discrete logarithm, now known as the Pohlig-Hellman algorithm. In this paper, I will provide several basic examples of the procedure, as well as prove a theorem of the result that

holds true in general. I will then provide another example of a method used to speed up the computation significantly when we have repeated prime factors, and then provide a proof that this method works in general. Last, I will discuss some of the limitations of this procedure.

2 Historical Notes

Before I continue further, I would like to briefly provide a few historical details.

Martin Hellman was one of the architects of the Diffie-Hellman Key Exchange, the first public key cryptosystem and a system based on the DLP. He grew up in the Bronx in New York City as a second-generation immigrant. He credits his father, a high school Physics teacher and HAM radio operator, as one of the factors that stoked his interest in Electrical Engineering.

Hellman did his graduate studies at Stanford University and received both his Masters and PhD in Electrical Engineering. He decided not to go into teaching because, as he know from his father, "[Teachers] don't make a lot of money." He worked at IBM's Research Department in 1968 – 1969. Here, he would regularly have lunch with Horst Feistel — the main creator of the Data Encryption Standard (DES). It was during these informal lunches that Hellman first became interested in cryptography, and the rest as they say, is history.

After Hellman realized that "[...] you didn't have to be poor to be a University Professor," he joined the faculty of Stanford, where he met Stephen Pohlig. Pohlig was a student of Hellman, and it was there at Stanford where the pair did a lot of exciting mathematics, including the development of the Pohlig-Hellman algorithm for calculating discrete logarithms. For more information, see [Hel04].

3 The Pohlig-Hellman Algorithm

We begin with a definition from the textbook.

Definition 3.1. [HPS14] The *order of a modulo p* is the smallest exponent $k \geq 1$ such that $a^k \equiv 1 \mod p$.

In solving the discrete logarithm problem $g^x \equiv h \mod p$, we know that since g is a primitive root, it has order $p-1$ modulo p. But recall that as a corollary to Fermat's Little Theorem, the exponents in a congruence are related to each other modulo $p-1$. In other words, the solution to the discrete logarithm problem, x, exists modulo $p-1$.

Now p obviously cannot be factored (as it is prime), but $p-1$ is not prime, and so can be written as a product of prime numbers $p-1 = p_1^{e_1} \cdot p_2^{e^2} \cdots p_n^{e_n}$. Then x is the solution modulo $p-1 = p_1^{e_1} \cdot p_2^{e^2} \cdots p_n^{e_n}$. This can split into individual congruences

$$x_1 \equiv a_1 \mod p_1^{e^1}, \quad x_2 \equiv a_2 \mod p_2^{e^2}, \quad \ldots, \quad x_n \equiv a_n \mod p_n^{e^n},$$

where the details of each congruence will be explained later. Since the moduli are powers of *different* prime numbers, then they are all relatively prime. So the Chinese Remainder Theorem guarantees a unique solution for the system of congruences modulo $p_1^{e_1} \cdot p_2^{e^2} \cdots p_n^{e_n} = p - 1$. We can solve each individual congruence, then piece them all back together to find the the solution to the original discrete logarithm problem $g^x \equiv h \pmod{p}$! Before we state the general result, we look at an example.

Example 3.1. Solve $3^x \equiv 22 \pmod{31}$.

Since 3 is a primitive root modulo 31, we know that such an x will exist. We have $p = 31$, and we compute the prime factorization of $p - 1 = 30 = 2 \cdot 3 \cdot 5$. We then choose one of the prime factors, say 5, and compute

$$\begin{aligned}
(22)^{\frac{p-1}{5}} &\equiv (3^x)^{\frac{p-1}{5}} \mod p \\
22^6 &\equiv (3^{5q_1+r_1})^{\frac{p-1}{5}} \mod p && \text{(Division Algorithm)} \\
8 &\equiv (3^{5q_1})^{\frac{p-1}{5}} \cdot (3^{r_1})^{\frac{p-1}{5}} \mod p \\
8 &\equiv (3^{p-1})^{q_1} \cdot (3^{6r_1}) \mod p \\
8 &\equiv (1)^{q_1} \cdot (3^6)^{r_1} \mod p && \text{(Fermat's Little Theorem)} \\
8 &\equiv 16^{r_1} \mod p
\end{aligned}$$

When we used the division algorithm above, we divided x by 5, so $0 \leq r_1 < 5$. We must plug in $r_1 = 0, 1, \ldots, 4$ until we find the correct value. It turns out that $r_1 = 2$ satisfies the congruence $8 \equiv 16^{r_1} \mod p$. Then since $x = 5q_1 + r_1$, we have that $x \equiv r_1 \mod 5$. Thus $x \equiv 2 \mod 5$. We repeat with another prime factor, say 3. Some details will be omitted this time.

$$\begin{aligned}
(22)^{\frac{p-1}{3}} &\equiv (3^x)^{\frac{p-1}{3}} \mod p \\
22^{10} &\equiv (3^{3q_2+r_2})^{\frac{p-1}{3}} \mod p && \text{(Division Algorithm)} \\
5 &\equiv (3^{3q_2})^{\frac{p-1}{3}} \cdot (3^{r_2})^{\frac{p-1}{3}} \mod p \\
5 &\equiv (3^{10r_2}) \mod p && \text{(Fermat's Little Theorem)} \\
5 &\equiv 25^{r_2} \mod p
\end{aligned}$$

We have $0 \leq r_2 < 3$. By trial and error, $r_2 = 2$. Thus $x \equiv 2 \mod 3$.

We repeat with the last prime factor 2.

$$\begin{aligned}
(22)^{\frac{p-1}{2}} &\equiv (3^x)^{\frac{p-1}{2}} \mod p \\
22^{15} &\equiv (3^{2q_3+r_3})^{\frac{p-1}{2}} \mod p && \text{(Division Algorithm)} \\
30 &\equiv (3^{2q_3})^{\frac{p-1}{2}} \cdot (3^{r_3})^{\frac{p-1}{2}} \mod p \\
30 &\equiv (3^{15r_3}) \mod p && \text{(Fermat's Little Theorem)} \\
30 &\equiv 30^{r_3} \mod p
\end{aligned}$$

We have $0 \leq r_3 < 2$. By trial and error, $r_3 = 1$. Thus $x \equiv 1 \mod 2$.

We use the Chinese Remainder Theorem to solve the three congruences

$$x \equiv 2 \mod 5, \quad x \equiv 2 \mod 3, \quad x \equiv 1 \mod 2,$$

which produce the unique solution $x \equiv 17 \mod 30$. Note that $3^{17} \equiv 22 \mod 31$, so this solves the problem.

In the previous example, each prime factor had multiplicity 1. But what happens when a prime factor is repeated? We will demonstrate this case with one more example.

Example 3.2. Solve $6^x \equiv 30 \pmod{41}$.

6 is a primitive root modulo 41, so such an x will exist. Note that $p = 41$, so $p - 1 = 40 = 2^3 \cdot 5$. If we select prime factor 5, then it can be shown that $x \equiv 3 \mod 5$ using the method described in Example 3.1. The details are left as an exercise.

Next, we select prime factor 2. Since it is repeated three times, then we compute

$$\begin{aligned}
(30)^{\frac{p-1}{2^3}} &\equiv (6^x)^{\frac{p-1}{2^3}} \mod p \\
30^5 &\equiv (6^{8q_2+r_2})^{\frac{p-1}{8}} \mod p && \text{(Division Algorithm)} \\
38 &\equiv (6^{8q_2})^{\frac{p-1}{8}} \cdot (6^{r_2})^{\frac{p-1}{8}} \mod p \\
38 &\equiv (6^{5r_2}) \mod p && \text{(Fermat's Little Theorem)} \\
38 &\equiv 27^{r_2} \mod p
\end{aligned}$$

By the Division Algorithm, we know that $0 \leq r_2 < 8$. By trial and error, $r_2 = 7$, so $x_2 \equiv 7 \mod 8$.

Then we are left with $x \equiv 7 \mod 8$ and $x \equiv 3 \mod 5$, so by the Chinese Remainder Theorem we have that $x \equiv 23 \mod 40$ is the unique solution to the pair of congruences. Note that $3^{23} \equiv 22 \mod 41$, so this solves the problem.

Remark 3.2. In the examples above, trial and error was used to compute the solution for each congruence. We had to plug in values for each remainder $r_i = 0, \ldots, p_i - 1$. This was easy enough to do here but this suggests that if the prime factors of $p - 1$ are very large, we may want to use this procedure in tandem with an algorithm to solve each individual discrete logarithm problem, such as Shanks' Babystep-giantstep Algorithm. It also suggests that there may be a more clever way to deal with repeated prime factors, as they can become quite large. This will be discussed after the following theorem.

Theorem 3.3 (Pohlig-Hellman Algorithm)**.** [HPS14] *Let $\mathbb{F}_p$ be a finite field, and suppose that we have an algorithm to solve the discrete logarithm problem in $\mathbb{F}_p$ for an element whose order is a power of a prime. To be concrete, if $g \in \mathbb{F}_p$ has order q^e, suppose that we can solve $g^x = h$ in $\mathcal{O}(S_{q^e})$ steps.*

Now let $g \in \mathbb{F}_p$ be a primitive root, and suppose that $p-1$ factors into a product of prime powers as

$$p - 1 = p_1^{e_1} \cdot p_2^{e_2} \cdots p_t^{e_t}.$$

Then the discrete logarithm problem $g^x = h$ can be solved in

$$\mathcal{O}\left(\sum_{i=1}^{t} S_{p_i^{e_i}} + \log(p-1)\right) \quad \textit{steps}$$

using the following procedure:

(1) For each $1 \le i \le t$, let

$$g_i = g^{\frac{p-1}{p_i^{e_i}}} \qquad \textit{and} \qquad h_i = h^{\frac{p-1}{p_i^{e_i}}}.$$

Notice that g_i has prime power order $p_i^{e_i}$, so use the given algorithm to solve the discrete logarithm problem

$$g_i^r = h_i.$$

Let $r = r_i$ be a solution to the above equation,

(2) Use the Chinese Remainder Theorem to solve

$$x \equiv r_1 \mod p_1^{e_1}, \quad x \equiv r_2 \mod p_2^{e_2}, \quad \ldots, \quad x \equiv r_t \mod p_t^{e_t}.$$

Proof. First, we show that the algorithm described in steps (1) and (2) give a solution to $g^x \equiv h \mod p$. Suppose that x is the solution to the system of congruences in step (2) above. By the Chinese Remainder Theorem, this solution is unique modulo $p_1^{e_1} p_2^{e_2} \cdots p_t^{e_t}$. We would like to show that this x is the same solution to the original discrete logarithm problem.

For each congruence, we have $x \equiv r_i \mod p_i^{e_i}$ for some $r_i \in \mathbb{Z}/(p_i^{e_i})\mathbb{Z}$. In other words, $0 \le r_i < p_i^{e_i}$. We note that by definition, $p_i^{e_i} | (x - r_i) \implies x - r_i = p_i^{e_i} q_i \implies x = p_i^{e_i} q_i + r_i$ for some $q_i \in \mathbb{Z}$. We can then write

$$\begin{aligned}
h_i &= g_i^{r_i} && \text{(DLP for element } g_i \text{ of prime power order)} \\
h^{\frac{p-1}{p_i^{e_i}}} &= (g^{\frac{p-1}{p_i^{e_i}}})^{r_i} && \text{(from (1) in the theorem statement)} \\
h^{\frac{p-1}{p_i^{e_i}}} &= (g^{r_i})^{\frac{p-1}{p_i^{e_i}}}
\end{aligned}$$

Then we have that $h^{\frac{p-1}{p_i^{e_i}}} = g^{r_i \cdot \frac{p-1}{p_i^{e_i}}}$. By the one-to-one property of logarithms, this implies

$$\log_g(h^{\frac{p-1}{p_i^{e_i}}}) = \log_g(g^{r_i \cdot \frac{p-1}{p_i^{e_i}}}),$$

which we may rewrite with properties of logarithms as

$$\frac{p-1}{p_i^{e_i}} \cdot \log_g(h) = r_i \cdot \frac{p-1}{p_i^{e_i}}.$$

We have that $x \equiv r_i \mod p_i^{e_i}$, thus

$$\frac{p-1}{p_i^{e_i}} \cdot \log_g(h) \equiv x \cdot \frac{p-1}{p_i^{e_i}} \mod p_i^{e_i}.$$

Since $\gcd(\frac{p-1}{p_i^{e_i}}, p_i^{e_i}) = 1$, we may use cancellation to rewrite this as

$$x \equiv \log_g(h) \mod p_i^{e_i}$$

We may repeat this process for each of the i congruences, $1 \leq i \leq t$. We are left with

$$x \equiv \log_g(h) \mod p_1^{e_1}, \quad x \equiv \log_g(h) \mod p_2^{e_2}, \quad \ldots, \quad x \equiv \log_g(h) \mod p_t^{e_t}.$$

Then since x was the solution to the system of congruences by assumption, we have that $x \equiv \log_g(h) \mod p-1$ as claimed.

Last, we will show that this procedure runs in $\mathcal{O}(\sum_{i=1}^{t} S_{p_i^{e_i}} + \log(p-1))$ steps. By assumption, if $g \in G$ has order of prime power q^e, then we can solve $g^x = h$ in $\mathcal{O}(S_{q^e})$ steps. Since we must solve t congruences, the runtime of part (1) of the procedure is $\mathcal{O}(\sum_{i=1}^{t} S_{p_i^{e_i}})$. Next, it is a fact that the Chinese Remainder Theorem takes $\mathcal{O}(\log N)$ steps, where N is the size of the modulus. So step (2) takes $\mathcal{O}(\log(p-1))$ steps. The claim follows. □

Remark 3.4. Theorem 3.3 also holds true for any group G and $g \in G$ whose order is the power of a prime.

Remark 3.5. The procedure above works to manipulate the discrete logarithm problem, changing the problem from elements in G of order $p-1$ to several simpler discrete logarithm problems with elements of order $p_i^{e_i}$. In theory this, simplifies the problem. In practice, the procedure does not necessarily guarantee a more efficient approach when the elements have large prime power factors. Consider the modulus $p = 11251$. Then $p - 1 = 11250 = 5^4$. This "simpler" discrete logarithm problem is not much simpler than the original one!

In the following example, we will demonstrate an improvement to the procedure above. It will essentially reduce the problem to elements of prime order p_i, instead of elements of prime power order $p_i^{e_i}$ as before. Since this makes the elements either have the same order or smaller order, then it will effectively reduce the runtime. We revisit Example 3.2 with a new method.

Example 3.3. Solve $6^x \equiv 30 \pmod{41}$.

We first find the following prime factorization: $p-1 = 2^3 \cdot 5$. We choose one of the prime factors, say 5, and compute:

$$\begin{aligned}(30)^{\frac{p-1}{5}} &\equiv (6^{x_0})^{\frac{p-1}{5}} \mod 41 \\ 30^8 &\equiv (6^8)^{x_0} \mod 41 \\ 16 &\equiv 10^{x_0} \mod 41\end{aligned}$$

By trial and error, we let $x_0 = 0, 1, 2, \ldots$ until we arrive at the solution $x_0 = 3$. We then compute $x \equiv 5^0 \cdot x_0 \mod 5 \implies x \equiv 3 \mod 5^1 = 5$.

We then choose the prime factor $p = 2$, which has multiplicity 3. Here is where things are different. We make 3 separate computations to find x_0, x_1, and x_2. Afterwards, we will compute $x = 2^0x_0 + 2^1x_1 + 2^2x_2 = x_0 + 2x_1 + 4x_2$. First, x_0 :

$$\begin{aligned}(30)^{\frac{p-1}{2}} &\equiv (6^{x_0})^{\frac{p-1}{2}} \mod 41 \\ 30^{20} &\equiv (6^{20})^{x_0} \mod 41 \\ 40 &\equiv 40^{x_0} \mod 41\end{aligned}$$

We plug in values $x_0 = 0, 1, 2, \ldots$ to find the solution. In this case, clearly $x_0 = 1$. Next, we find x_1 :

$$\begin{aligned}(30 \cdot 6^{-x_0})^{\frac{p-1}{2^2}} &\equiv (6^{x_1})^{\frac{p-1}{2}} \mod 41 \\ (30 \cdot (6^{-1})^{x_0})^{10} &\equiv (6^{20})^{x_1} \mod 41 \\ (30 \cdot (7)^1)^{10} &\equiv 40^{x_1} \mod 41 \qquad (6^{-1} \equiv 7 \mod 41) \\ 40 &\equiv 40^{x_1} \mod 41\end{aligned}$$

Then $x_1 = 1$. Next, we find x_2 :

$$\begin{aligned}(30 \cdot 6^{-x_0-2x_1})^{\frac{p-1}{2^3}} &\equiv (6^{x_2})^{\frac{p-1}{2}} \mod 41 \\ (30 \cdot (6^{-1})^{x_0+2x_1})^5 &\equiv (6^{20})^{x_2} \mod 41 \\ (30 \cdot (7)^3)^5 &\equiv 40^{x_2} \mod 41 \qquad (6^{-1} \equiv 7 \mod 41) \\ 40 &\equiv 40^{x_2} \mod 41\end{aligned}$$

We find $x_2 = 1$. Then $x = x_0 + 2x_1 + 4x_2 = 7$, so $x \equiv 7 \mod 2^3$. We can then use the Chinese Remainder Theorem to solve the pair of congruences $x \equiv 3 \mod 5$ and $x \equiv 7 \mod 8$, giving use the solution $x \equiv 23 \mod 40$. Indeed this is the correct solution to $6^x \equiv 30 \mod 41$.

It remains to prove that this result holds in general. The following theorem does just that.

Theorem 3.6. [HPS14] *Let G be a group. Suppose that q is a prime, and suppose that we know an algorithm that takes S_q steps to solve the discrete logarithm problem $g^x = h$ for all $g \in G$ whose order is q. Now let $g \in G$ be an element of order q^e with $e \geq 1$. Then we can solve the discrete logarithm problem $g^x = h$ in $\mathcal{O}(eS_q)$ steps.*

Proof. We may write the base q expansion of x as

$$x = x_0 q^0 + x_1 q^1 + \cdots + x_{e-1} q^{e-1} \qquad 0 \leq x_i < q.$$

Note that this is similar to how we might write the binary (base 2) expansion of an integer. For example, the number 9 can be written in base 2 as

$$9 = 1 \cdot 2^0 + 0 \cdot 2^1 + 0 \cdot 2^2 + 1 \cdot 2^3.$$

We will solve multiple discrete logarithm problems to find each value of x_i, $0 \leq i \leq e-1$. We will use the fact that since g has order q by assumption, then $g^q = 1$. It follows from this that any element of the form g^{q^i} is also of order q for all i, since $g^{q^i} = (g^q)^{q^{i-1}} = 1^{q^{i-1}} = 1$. In particular, $g^{q^{e-1}}$ is of order q. We can then make the following calculation

$$\begin{aligned}
h^{\frac{q^e}{q}} &= (g^x)^{\frac{q^e}{q}} \\
h^{q^{e-1}} &= (g^x)^{q^{e-1}} \\
h^{q^{e-1}} &= (g^{x_0 q^0 + x_1 q^1 + x_2 q^2 + \cdots x_{e-1} q^{e-1}})^{q^{e-1}} && \text{(Base } q \text{ expansion of } x\text{)} \\
h^{q^{e-1}} &= (g^{x_0})^{q^{e-1}} \cdot (g^{x_1 q + x_2 q^2 + \cdots x_{e-1} q^{e-1}})^{q^{e-1}} \\
h^{q^{e-1}} &= (g^{q^{e-1}})^{x_0} \cdot (g^{x_1 q^e + x_2 q^{e+1} + \cdots x_{e-1} q^{2e-2}}) \\
h^{q^{e-1}} &= (g^{q^{e-1}})^{x_0} \cdot (g^{q_e})^{x_1 + x_2 q + \cdots x_{e-1} q^{e-2}} \\
h^{q^{e-1}} &= (g^{q^{e-1}})^{x_0} && (g^{q^e} = 1)
\end{aligned}$$

So we are left with $h^{q^{e-1}} = (g^{q^{e-1}})^{x_0}$, which is a discrete logarithm problem whose base is the element $g^{q^{e-1}}$ with order q. This can be solved to find the missing exponent x_0 in S_q steps by assumption.

We next make a similar calculation, this time dividing exponent q^e by q^2.

$$\begin{aligned}
h^{\frac{q^e}{q^2}} &= (g^x)^{\frac{q^e}{q^2}} \\
h^{q^{e-2}} &= (g^x)^{q^{e-2}} \\
h^{q^{e-2}} &= (g^{x_0 q^0 + x_1 q^1 + x_2 q^2 + \cdots x_{e-1} q^{e-1}})^{q^{e-2}} \\
h^{q^{e-2}} &= g^{x_0 q^{e-2}} \cdot (g^{x_1 q})^{q^{e-2}} \cdot (g^{x_2 q^2 + \cdots x_{e-1} q^{e-1}})^{q^{e-2}} \\
h^{q^{e-2}} &= g^{x_0 q^{e-2}} \cdot g^{x_1 q^{e-1}} \cdot (g^{x_2 q^e + \cdots x_{e-1} q^{2e-3}}) \\
h^{q^{e-2}} &= g^{x_0 q^{e-2}} \cdot (g^{q^{e-1}})^{x_1} \cdot (g^{q_e})^{x_2 + \cdots x_{e-1} q^{e-3}} \\
h^{q^{e-2}} &= g^{x_0 q^{e-2}} \cdot (g^{q^{e-1}})^{x_1}
\end{aligned}$$

Since we already have found x_0, then we may multiply by $(g^{x_0 q^{e-2}})^{-1}$ on each side to obtain the following

$$h^{q^{e-2}} \cdot (g^{x_0 q^{e-2}})^{-1} = (g^{q^{e-1}})^{x_1}$$
$$(h \cdot g^{-x_0})^{q^{e-2}} = (g^{q^{e-1}})^{x_1}$$

We are again left with a discrete logarithm problem whose base is the element $g^{q^{e-1}}$ with order q. This can be solved to find the missing exponent x_1 in S_q steps by assumption. Combining the first two steps together, we now have the values of x_0 and x_1 such that

$$h^{q^{e-2}} = (g^{(x_0 + x_1 q)})^{q^{e-2}}$$

Next, we repeat the process above to find x_2. The details will be omitted this time.

$$h^{q^{e-3}} \cdot (g^{x_0 q^{e-3}})^{-1} \cdot (g^{x_1 q \cdot q^{e-3}})^{-1} = (g^{q^{e-1}})^{x_2}$$
$$(h \cdot g^{-x_0 - x_1 q})^{q^{e-3}} = (g^{q^{e-1}})^{x_2}$$

We iterate in this fashion, obtaining the value of x_i every time by solving the discrete logarithm problem

$$(h \cdot g^{-k})^{q^{e-(i+1)}} = (g^{q^{e-1}})^{x_i},$$

where $k = \sum_{n=0}^{i-1} x_n \cdot q^n$. This continues until we have found the value of each x_i on the right hand side of the equality $x = x_0 + \cdots x_{e-1} q^{e-1}$. Since x is a finite number, then eventually this process will terminate. Computing x solves the original discrete logarithm problem $g^x = h$.

Note that each individual discrete logarithm problem we solved gave us the value x_i, $0 \leq i \leq e-1$. Since each problem had a base with order q, then each can be solved in S_q steps by assumption. We calculated e different values of x_i, so the overall procedure took $\mathcal{O}(eS_q)$ steps as claimed. □

Remark 3.7. This theorem extends the procedure to the case when $p-1$ has a single prime factor that is repeated, i.e. the case when $p-1 = q^i$, where q is prime and $i > 1$. A concrete example of this used above is if $p = 11251$, then $p - 1 = 11250 = 5^4$. The discrete logarithm problem can be solved using the method in Theorem 3.6, and still works if there are multiple prime factors that are repeated, as demonstrated in 3.3.

Remark 3.8. Theorem 3.3 describes a procedure that solves the discrete logarithm problem for elements of order q^e in $\mathcal{O}(S_{q^e})$ time, where S_{q^e} is the number of steps required. Theorem 3.6 allows us to reduce the running time for elements of order q^e to just $\mathcal{O}(eS_q)$. Then given the discrete logarithm problem $g^x \equiv h \mod p$, we can solve it in time $\mathcal{O}(\sum_{i=1}^{t} eS_q + \log(p-1))$. If we use Shanks' Babystep-giantstep algorithm, then we may take S_q to be $\sqrt{q}\log(q)$ steps.

4 Limitations

The Pohlig-Hellman algorithm is effective for the discrete logarithm problem in $\mathbb{F}_p$ if $p-1$ factors into the product of small primes. However, if $p-1$ only contains very large prime factors, then this algorithm is not very efficient.

In practice, we may choose a very large prime q and use an element $g \in \mathbb{F}_p$ whose order is q. Then we can let $p = 2q+1$, so that $p-1 = 2q$. Since q is so large, then Pohlig-Hellman will not be efficient to solve the discrete logarithm problem modulo p.

References

[Hel04] Martin Hellman, *Oral history interview with Martin Hellman*, Charles Babbage Institute (2004), available at https://hdl.handle.net/11299/107353.

[HP78] Martin Hellman and Stephen Pohlig, *An improved algorithm for computing logarithms over GF(p) and its cryptographic significance*, IEEE Transactions on Information Theory **24** (1978), no. 1, 106-110.

[HPS14] Jeffrey Hoffstein, Jill Pipher, and Joseph Silverman, *An Introduction to Mathematical Cryptograpy*, 2nd ed., Springer, New York, 2014.