

## 1 The Diffie-Hellman Key Exchange System

### 1.1 Mechanism

**Goal:** Alice and Bob want to exchange a shared “secret key” via an insecure channel. (Incidentally, the shared secret key is usually used for encryption and decryption with a symmetric cipher.)

- Alice and Bob choose and publish a large prime  $p$  and an integer  $g$  with large prime order in  $\mathbb{F}_p^*$ .
- Alice chooses a secret  $a \in \mathbb{Z}$ , computes  $A := g^a \bmod p$ , and sends  $A \in \mathbb{F}_p^*$  to Bob via the insecure channel.
- Bob chooses a secret  $b \in \mathbb{Z}$ , computes  $B := g^b \bmod p$ , and sends  $B \in \mathbb{F}_p^*$  to Alice via the insecure channel.
- Bob computes  $A^b = (g^a)^b = g^{ab} \in \mathbb{F}_p^*$ .
- Alice computes  $B^a = (g^b)^a = g^{ab} \in \mathbb{F}_p^*$ .
- Alice and Bob have both arrived at  $g^{ab} \in \mathbb{F}_p^*$ , which is to be their shared secret.

### 1.2 Security of the Diffie-Hellman key exchange system

- The presumed security of the Diffie-Hellman key exchange system is based on the presumed difficulty in solving the

**The Diffie-Hellman Problem:**

Let  $p$  be a prime, and  $g \in \mathbb{F}_p^*$ . Given  $A, B \in \langle g \rangle \subset \mathbb{F}_p^*$ , find  $g^{ab} \in \mathbb{F}_p^*$ , where  $a, b \in \mathbb{Z}$  are determined by  $A = g^a$  and  $B = g^b$ .

Note that the solution of the Diffie-Hellman problem does NOT require the knowledge of  $a$  and  $b$ , but only that of  $g^{ab} \in \langle g \rangle$ .

- It is clear that an efficient algorithm for the Discrete Logarithm Problem will lead to an efficient algorithm for the Diffie-Hellman Problem. An efficient algorithm of the Discrete Logarithm Problem will break the Diffie-Hellman key exchange system.

**The Discrete Logarithm Problem:**

Let  $p$  be a prime, and  $g \in \mathbb{F}_p^*$  be a primitive element, i.e.  $\mathbb{F}_p^* = \langle g \rangle$ . Given any  $h \in \mathbb{F}_p^*$ , find  $x \in \mathbb{Z}$  such that  $h = g^x$ .

## 2 The ElGamal Public Key Cryptosystem

### 2.1 Mechanism

**Goal:** Bob wants to send Alice an encrypted message via an insecure channel.

- Alice chooses and publishes a large prime  $p$  and an integer  $g$  with large prime order in  $\mathbb{F}_p^*$ .
- Alice chooses a secret  $a \in \mathbb{Z}$ , with  $1 \leq a \leq p - 1$ , computes  $A := g^a \bmod p$ , and publishes  $A \in \mathbb{F}_p^*$  as her public key.
- Bob chooses a plaintext  $m \in \mathbb{F}_p^*$ , and a secret random ephemeral key  $k \in \mathbb{Z}$ , with  $1 \leq k \leq p - 1$ . Bob computes  $B_1 := g^k \bmod p$ , and  $B_2 := m \cdot A^k \in \mathbb{F}_p^*$ . Bob's ciphertext is  $(B_1, B_2)$  and Bob sends it to Alice via the insecure channel.
- Alice decrypts Bob's ciphertext  $(B_1, B_2)$  by computing  $B_2 \cdot (B_1^a)^{-1} \in \mathbb{F}_p^*$ . Note that

$$B_2 \cdot (B_1^a)^{-1} \equiv (m \cdot A^k) \cdot (g^{ak})^{-1} \equiv m \cdot (g^a)^k \cdot (g^{ak})^{-1} \equiv m \cdot (g^{ak}) \cdot (g^{ak})^{-1} \equiv m \bmod p$$

### 2.2 Security of the ElGamal public key cryptosystem

- The ElGamal public key encryption system is at least as secure as the difficulty of Diffie-Hellman problem, in the sense that an ElGamal oracle (efficient solver of the ElGamal cryptosystem problem) can be used to efficiently solve the Diffie-Hellman problem.

## 3 The RSA Public Key Cryptosystem

### 3.1 Mechanism

**Goal:** Alice wants to send Bob an encrypted message through an insecure channel.

- Bob chooses his public key  $(n, e) \in \mathbb{N}^2$  and private key  $d \in \mathbb{N}$ . Bob publishes his public key.
  - $n \in \mathbb{N}$  is called the modulus, with  $n = pq$ , where  $p$  and  $q$  are large distinct prime numbers. Note that Bob publishes  $n$  but keeps  $p$  and  $q$  secret.
  - $e \in \mathbb{N}$  is the called encryption exponent, and satisfies  $\gcd(e, (p-1)(q-1)) = 1$ .
  - $d \in \mathbb{N}$  is the called decryption exponent, and is determined by  $e$  and  $n = pq$  via  $d = e^{-1} \in \mathbb{Z}_{(p-1)(q-1)}$ . Note that  $e^{-1} \in \mathbb{Z}_{(p-1)(q-1)}$  exists since  $\gcd(e, (p-1)(q-1)) = 1$ .
- Alice
  - chooses plaintext  $m \in \mathbb{Z}_n = \mathbb{Z}/n\mathbb{Z}$ .
  - encrypts her plaintext  $m$  using Bob's public key  $(n, e)$  by **raising  $m \in \mathbb{Z}_n$  to the  $e^{\text{th}}$  power**. In other words, Alice computes her ciphertext  $c = m^e \in \mathbb{Z}_n$ .
  - sends to Bob through the insecure channel the ciphertext  $c \in \mathbb{Z}_n$ .
- Bob decrypts the ciphertext  $c \in \mathbb{Z}_n$  from Alice by **taking the  $e^{\text{th}}$  root** of  $c$  in  $\mathbb{Z}_n$  using his private key  $d \in \mathbb{N}$  as follows:

$$c^d = (m^e)^d = m^{ed} = m^{1+k(p-1)(q-1)} = m \cdot (m^{(p-1)(q-1)})^k = m \cdot (1)^k = m \in \mathbb{Z}_n$$

- The second last equality follows from  $m^{(p-1)(q-1)} \equiv 1 \pmod{n}$ , which follows immediately from Euler's Theorem. It can also be justified with Fermat's Little Theorem as follows:

$$\text{Fermat's Little Theorem} \implies \begin{cases} m^{(p-1)(q-1)} = (m^{p-1})^{q-1} \equiv 1 \pmod{p}, \text{ and} \\ m^{(p-1)(q-1)} = (m^{q-1})^{p-1} \equiv 1 \pmod{q}. \end{cases}$$

Hence,  $m^{(p-1)(q-1)} - 1$  is divisible by both  $p$  and  $q$ , and hence also by  $pq = n$  (since  $p$  and  $q$  are distinct primes). Thus,  $m^{(p-1)(q-1)} \equiv 1 \pmod{n = pq}$ .

### 3.2 Comments

- One-way function (easy): Exponentiation in  $\mathbb{Z}_n$ .
  - Repeating Squaring Algorithm
- (Difficult) inverse function: Taking roots in  $\mathbb{Z}_n$ , for  $n = pq$ , where  $p$  and  $q$  are large distinct prime numbers.
- Trapdoor: If the factorization of  $n = pq$  is known, then we can convert the inverse function (taking roots in  $\mathbb{Z}_n$ , which is slow) to an exponentiation in  $\mathbb{Z}_n$ , which is fast.

## 3.3 How to find large prime numbers?

- Generate a large  $N$ -bit (say  $N = 1024$ ) random number  $x$ , i.e.  $2^{N-1} < x < 2^N$ . Use an efficient primality test to check whether  $x$  is prime. If so, we are done. If not, repeat until we succeed.
- The Prime Number Theorem (from Analytic Number Theory) gives an estimate of how many times we need to try before succeeding. The Prime Number Theorem states that

$$\lim_{x \rightarrow \infty} \frac{\pi(x)/x}{1/\ln(x)} = 1,$$

where  $\pi(x)$  is the number of prime numbers less than or equal to  $x$ . Hence, it implies that, for large values of  $N$ , the probability that a randomly selected integer  $x \in (2^{N-1}, 2^N)$  is prime is approximately

$$\frac{1}{\ln(2^N)}$$

Conversely, this implies that, on average, out of every  $\ln(2^N) = N \cdot \ln(2) \approx 0.693 \cdot N$  randomly and independently selected integers from  $(2^{N-1}, 2^N)$ , one of them will be a prime number. For example, if  $N = 1024$ , then  $0.693 \cdot N \approx 709.78$ ; in other words, if we are selecting random integers from  $(2^{1023}, 2^{1024})$ , then on average, we expect repeating approximately 710 times before we succeed in selecting a prime number. Note that  $2^{1023} = 10^{1023 \times \log_{10}(2)} \approx 10^{1023 \times 0.301} \approx 10^{307.95}$ .

- The Miller-Rabin Primality test
  - **Proposition** Let  $p$  be an odd prime and write  $p - 1 = 2^k t$ , where  $t$  is odd. Then, for each  $a \in \mathbb{Z}$  with  $p \nmid a$ , one of the following is true:
    - $a^t \equiv 1 \pmod{p}$ , or
    - One of  $a^t, a^{2t}, a^{4t}, \dots, a^{2^{k-1}t}$  is congruent to  $-1 \pmod{p}$ .
  - **Corollary** Let  $n \in \mathbb{Z}$  be an odd number, with  $n - 1 = 2^k t$ ,  $t$  being odd. Then,  $n$  is composite, if any of the following is true:
    - There exists  $a \in \mathbb{Z}$  such that  $\gcd(a, n) > 1$ .
    - There exists  $a \in \mathbb{Z}$  such that  $\gcd(a, n) = 1$ , and  $a^t \not\equiv 1 \pmod{n}$ , and  $a^{2^i t} \not\equiv -1 \pmod{n}$ , for each  $i = 0, 1, 2, \dots, k - 1$ .
  - **Proposition** Let  $n$  be an odd composite number. Then, at least 75% of integers between 1 and  $n - 1$  are Miller-Rabin witnesses for  $n$ .

## 3.4 Factorization algorithms

- Pollard's  $p - 1$  factorization algorithm

This method “**probably**” works for producing a non-trivial factor for composite  $n \in \mathbb{N}$  admitting a prime factor  $p$  such that  $p - 1$  is a product of small primes.

- **Proposition:** Let  $n = pq$ , where  $p$  and  $q$  are distinct prime numbers. Then the following two statements hold:

- For each  $L \in \mathbb{N}$  and  $a$  with  $p \nmid a$ , we have the following implication:

$$(p-1) \mid L \implies L = (p-1)K \implies a^L = \left(a^{(p-1)}\right)^K \equiv 1 \pmod{p} \implies p \mid (a^L - 1)$$

- For any  $a \in \mathbb{N}$  with  $p \nmid a$  and  $q \nmid a$ , and any  $L \in \mathbb{N}$ ,

$$\left. \begin{array}{l} p \mid (a^L - 1) \\ q \nmid (a^L - 1) \end{array} \right\} \implies p = \gcd(a^L - 1, n)$$

– **Key observations:**

- If  $p-1$  is a product of small primes, then  $(p-1) \mid N!$ , for some not-too-large  $N$ .
- If  $(q-1) \nmid N!$ , then  $q \nmid (a^{N!} - 1)$  is “probably” true.
- If  $p-1$  is a product of small primes, and  $q-1$  is NOT so, then computing  $\gcd(a^{k!} - 1, n)$ , for  $k = 2, 3, \dots$ , will “probably” yield  $p$  as a non-trivial factor of  $n$ .

- Factorization via difference of squares

– **Key observations:**

Suppose we know that  $n \in \mathbb{N}$  is odd and composite. We want to find a non-trivial factor of  $n$ .

- If we can find  $a, b \in \mathbb{N}$  such that  $n$  is the difference of their squares, i.e.  $n = a^2 - b^2 = (a-b)(a+b)$ , then computing  $\gcd(a-b, n)$  will yield a non-trivial factor of  $n$ .
- Conversely, suppose  $n = cd$ . Since  $n$  is odd, both  $c$  and  $d$  must also be odd. Hence,  $a := \frac{1}{2}(c+d) \in \mathbb{Z}$  and  $b := \frac{1}{2}(c-d) \in \mathbb{Z}$ . And,  $a^2 - b^2 = \dots = cd = n$ . In other words, every composite odd integer can be written as the difference of two squares.
- If some multiple  $kn$  is a difference of squares, i.e.  $kn = a^2 - b^2 = (a-b)(a+b)$ , then computing  $\gcd(a-b, n)$  will “probably” yield a non-trivial factor of  $n$ , since it should be unlikely that  $n$  divides  $a-b$ .
- **In summary, if we could find  $a, b \in \mathbb{Z}$  such that  $a^2 \equiv b^2 \pmod{n}$ , then computing  $\gcd(a-b, n)$  will probably yield a non-trivial factor of  $n$ .**

– **Outline of general procedure:**

- (1) **Find  $B$ -smooth perfect squares in  $\mathbb{Z}_n$ .** Find many  $a_1, a_2, \dots, a_r \in \mathbb{Z}$  such that every prime factor of  $c_i \equiv a_i^2 \pmod{n}$  is less than or equal to  $B$ .
- (2) Find sub-collections  $c_{i_1}, c_{i_2}, \dots, c_{i_s}$  such that  $c_{i_1}c_{i_2} \dots c_{i_s} \equiv b^2 \pmod{n}$  are perfect squares in  $\mathbb{Z}_n$ .
- (3) Let  $a := a_{i_1}a_{i_2} \dots a_{i_s} \pmod{n}$ . Then, computing  $\gcd(a-b, n)$  will probably yield a non-trivial factor of  $n$ .

– Comments on the general procedure:

- Step (3) can be performed efficiently using the Euclidean Algorithm.
- Step (2) is equivalent to solving a homogeneous (sparse) system of linear equations over  $\mathbb{F}_2$ .
- The main challenge in difference-of-squares factorization is Step (1), namely, given  $n \in \mathbb{Z}$ , finding enough  $B$ -smooth perfect squares in  $\mathbb{Z}_n$ .

## 4 Methods for finding smooth numbers

### 4.1 The Quadratic Sieve

#### Problem

- Let  $N, B \in \mathbb{N}$  be given.
- Find a large number of integers of the form  $a^2 \bmod N$  that are  $B$ -smooth.

#### Theory

-

## 5 Algorithms for solving the Discrete Logarithm Problem

### The Discrete Logarithm Problem

- Let  $G$  be a finite cyclic group of order  $N \geq 2$ , and let  $g \in G$  be a generator of  $G$ .
- Let  $h \in G \setminus \{e\}$  be given.
- Find  $x \in \mathbb{N}$ ,  $1 \leq x < N$ , such that  $g^x = h$ .

#### 5.1 Shank's Baby-Step-Giant-Step Algorithm

**Theory:**

- Let  $n := 1 + \lfloor \sqrt{N} \rfloor$ , where  $\lfloor \sqrt{N} \rfloor$  is the round-down of  $\sqrt{N}$ . Note, trivially, that  $n^2 > N$ .
- Define:

$$\begin{aligned} \mathcal{S}_1 &:= \{ g^i \mid i = 0, 1, 2, \dots, n \} = \{ e, g, g^2, g^3, \dots, g^n \} \\ \mathcal{S}_2 &:= \{ h \cdot (g^{-n})^i \mid i = 0, 1, 2, \dots, n \} = \{ h, h \cdot (g^{-n}), h \cdot (g^{-n})^2, h \cdot (g^{-n})^3, \dots, h \cdot (g^{-n})^n \} \end{aligned}$$

- Then, it turns out that  $\mathcal{S}_1 \cap \mathcal{S}_2 \neq \emptyset$ .
- For any  $i, j \in \{0, 1, 2, \dots, n\}$  such that  $g^i = h \cdot (g^{-n})^j \in \mathcal{S}_1 \cap \mathcal{S}_2$ , the positive integer  $x := i + jn \in \mathbb{Z}$  is a solution to the Discrete Logarithm Problem.

**Outline of proof:**

- $h \in \langle g \rangle \setminus \{e\} \implies$  there exists  $x \in \mathbb{N}$ ,  $1 \leq x < N$ , such that  $g^x = h$ .
- Then,  $x = nq + r$ , with  $0 \leq r < n$  and  $q \geq 0$ . We claim that the quotient  $q$  in fact satisfies:  $q < n$ . Indeed,  $x < N$  implies

$$q = \frac{x - r}{n} \leq \frac{N}{n} < \frac{n^2}{n} = n.$$

- Hence,  $g^x = h$  can be rewritten as follows:

$$g^x = h \iff g^{nq+r} = h \iff g^r = h \cdot (g^{-n})^q \in \mathcal{S}_1 \cap \mathcal{S}_2, \quad \text{for some } 0 \leq r, q < n$$

**Pseudocode:**

- (1) Let  $n := 1 + \lfloor \sqrt{N} \rfloor$ , where  $\lfloor \sqrt{N} \rfloor$  is the round-down of  $\sqrt{N}$ .
- (2) Generate  $\mathcal{S}_1 = \{ g^i \mid i = 0, 1, 2, \dots, n \} = \{ e, g, g^2, g^3, \dots, g^n \}$ .
- (3) Form the hash table  $\mathcal{T}$ :

key $g^i$	$e$	$g$	$g^2$	$g^3$	$\dots$	$g^n$
value $i$	0	1	2	3	$\dots$	$n$

```

(4) gn := g^{-n};
    temp := h;
    for j = 0, 1, 2, ...
        if (T{temp} exists) {
            i := T{temp};
            return(i + n*j);
        } else {
            temp := temp * gn;
        }
    end

```

**Complexity:**  $\mathcal{O}(\sqrt{N})$

- The generation of  $\mathcal{S}_1$  takes approximately  $n$  multiplications.
- The generation of the hash table takes  $\mathcal{O}(1)$  computation steps.
- The loop takes  $\mathcal{O}(n)$  computational steps.
- Hence, the whole algorithm takes  $\mathcal{O}(n + n + 1) = \mathcal{O}(n) = \mathcal{O}(\sqrt{N})$  computational steps.



## 5.2 The Pohlig-Hellman Algorithm

### Summary:

- The Pohlig-Hellman Algorithm renders efficiently solvable the Discrete Logarithm Problem (DLP) in any finite cyclic group  $G$  whose order  $|G|$  is a product of powers of small primes.
- This is achieved with two results of Pohlig-Hellman:
  - The DLP in a finite cyclic group  $G$  with  $|G| = N = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_t^{\alpha_t}$  can be “converted” to  $t$  distinct associated DLP’s in groups of orders  $p_1^{\alpha_1}, \dots, p_t^{\alpha_t}$ , respectively. Here, “converted” means that the solutions of the associated DLP’s can be “combined”, via the Chinese Remainder Theorem, to give the solution of the original DLP.
  - The DLP in a finite cyclic group of prime power order  $p^\alpha$  can be “converted” to a series of DLP’s, each in a group of power  $p$ .
  - This last set of DLP’s in groups of prime power must then be solved by methods other than Pohlig-Hellman, for example, either by brute force, or by Shank’s Baby-Step-Giant-Step method. Thus, if each of the primer orders involved is relatively small (*e.g.*, efficiently solvable by brute force or Baby-Step-Giant-Step), then the original DLP can be efficiently solved.

### Theory:

- The assumption that  $h \in G = \langle g \rangle$  implies the solvability of  $g^x = h$  for  $x \in \{0, 1, 2, \dots, N-1\}$ .
- Let  $N = p_1^{\alpha_1} p_2^{\alpha_2} p_3^{\alpha_3} \cdots p_t^{\alpha_t}$  be the prime factorization of  $N$ . Then, it turns out that the following associated discrete logarithm problems (in finite cyclic groups with prime power order):

$$g_i^{y_i} = h_i, \quad \text{where } g_i := g^{N/p_i^{\alpha_i}}, \quad h_i := h^{N/p_i^{\alpha_i}}, \quad \text{and } i = 1, 2, \dots, t,$$

are also solvable for

$$y_i \in \{0, 1, 2, \dots, p_i^{\alpha_i} - 1\}.$$

If we also have their solutions  $y_i$ , then we can compute  $x$  using the Chinese Remainder Theorem, namely, by solving the following simultaneous congruences for  $x \in \{0, 1, 2, \dots, N-1\}$ :

$$x \equiv y_i \pmod{p_i^{\alpha_i}}, \quad \text{for } i = 1, 2, \dots, t.$$

### Outline of proof:

*The solvability of  $g^x = h$  implies the solvability of the auxiliary discrete logarithm problems  $g_i^{y_i} = h_i$  for  $y_i \in \{0, 1, 2, \dots, p_i^{\alpha_i} - 1\}$ , for each  $i = 1, 2, \dots, t$ , since*

$$g^x = h \implies (g^x)^{N/p_i^{\alpha_i}} = (h)^{N/p_i^{\alpha_i}} \implies \left(g^{N/p_i^{\alpha_i}}\right)^x = (h)^{N/p_i^{\alpha_i}} \implies (g_i)^x = h_i \implies (g_i)^{y_i} = h_i,$$

*where  $y_i := x \pmod{p_i^{\alpha_i}}$ . Here, note that  $\text{ord}(g_i) = p_i^{\alpha_i}$ , for each  $i = 1, 2, \dots, t$ . Next, observe that, for each  $i = 1, 2, \dots, t$ ,*

$$\begin{aligned} (g^{-x} \cdot h)^{N/p_i^{\alpha_i}} &= (g^{-x})^{N/p_i^{\alpha_i}} \cdot (h)^{N/p_i^{\alpha_i}} = \left(g^{N/p_i^{\alpha_i}}\right)^{-x} \cdot (h)^{N/p_i^{\alpha_i}} \\ &= (g_i)^{-x} \cdot h_i \\ &= (g_i)^{-y_i} \cdot h_i, \quad \text{since } x \equiv y_i \pmod{p_i^{\alpha_i}}, \text{ and } \text{ord}(g_i) = p_i^{\alpha_i} \\ &= 1_G, \quad \text{since } g_i^{y_i} = h_i \end{aligned}$$

Hence,  $\text{ord}(g^{-x} \cdot h)$  divides  $\frac{N}{p_i^{\alpha_i}}$ , for each  $i = 1, 2, \dots, t$ , which in turn implies that  $\text{ord}(g^{-x} \cdot h)$  divides  $\gcd\left(\frac{N}{p_1^{\alpha_1}}, \dots, \frac{N}{p_t^{\alpha_t}}\right)$ . However,  $\frac{N}{p_1^{\alpha_1}}, \dots, \frac{N}{p_t^{\alpha_t}}$  have no non-trivial common divisors, and hence  $\gcd\left(\frac{N}{p_1^{\alpha_1}}, \dots, \frac{N}{p_t^{\alpha_t}}\right) = 1$ . Thus,  $\text{ord}(g^x \cdot h) = 1$ ; hence,  $g^x = h$ , as desired.

- The discrete logarithm problem in a finite cyclic group of prime power order  $p^\alpha$  can be “converted” to a series of discrete logarithm problems in finite cyclic groups, each of order  $p$ .

### Outline of proof:

Let  $G = \langle g \rangle$  be a finite cyclic group, with  $|G| = p^\alpha$ , where  $p \in \mathbb{N}$  is a prime number,  $\alpha \in \mathbb{N}$ , and  $g \in G$  is a generator of  $G$ . Given  $h \in G$ , we wish to find  $x \in \{0, 1, 2, \dots, p^\alpha - 1\}$  such that  $g^x = h$ .

We make two key observations:

- $\text{ord}(g^{p^{\alpha-1}}) = p$ , and
- we may express  $x \in \{0, 1, 2, \dots, p^\alpha - 1\}$  as a linear combination of powers of  $p$ :

$$x = x_0 + x_1 p + x_2 p^2 + x_3 p^3 + \dots + x_{\alpha-1} p^{\alpha-1}, \quad \text{where } x_i \in \{0, 1, 2, \dots, p-1\}.$$

Assuming we have an “oracle” that can solve the discrete logarithm problem in a finite cyclic group of order  $p$ , we can then sequentially solve for  $x_0, x_1, x_2, \dots, x_{\alpha-1}$  as solutions to discrete logarithm problems in groups of order  $p$ , starting with  $x_0$ .

Raising both sides of  $g^x = h$  to the power of  $p^{\alpha-1}$  yields:

$$\begin{aligned} \left(g^{x_0 + \sum_{i=1}^{t-1} x_i p^i}\right)^{p^{\alpha-1}} &= (h)^{p^{\alpha-1}} \implies (g^{x_0})^{p^{\alpha-1}} \cdot \left(g^{\sum_{i=1}^{t-1} x_i p^i}\right)^{p^{\alpha-1}} = (h)^{p^{\alpha-1}} \\ &\implies \left(g^{p^{\alpha-1}}\right)^{x_0} \cdot (g^{p^\alpha})^{\sum_{i=1}^{t-1} x_i p^{i-1}} = (h)^{p^{\alpha-1}} \\ &\implies \left(g^{p^{\alpha-1}}\right)^{x_0} = (h)^{p^{\alpha-1}}, \quad \text{since } g^{p^\alpha} = 1_G. \end{aligned}$$

Since  $\text{ord}(g^{p^{\alpha-1}}) = p$ , we can use our oracle to obtain  $x_0 \in \{0, 1, 2, \dots, p-1\}$ .

Assuming we have successfully solved for  $x_0 \in \{0, 1, 2, \dots, p-1\}$ , we raise both sides of  $g^x = h$  to the power  $p^{\alpha-2}$  and obtain:

$$\begin{aligned} \left(g^{x_0 + x_1 p + \sum_{i=2}^{t-1} x_i p^i}\right)^{p^{\alpha-2}} &= (h)^{p^{\alpha-2}} \implies (g^{x_0})^{p^{\alpha-2}} \cdot (g^{x_1 p})^{p^{\alpha-2}} \cdot \left(g^{\sum_{i=2}^{t-1} x_i p^i}\right)^{p^{\alpha-2}} = (h)^{p^{\alpha-2}} \\ &\implies (g^{x_0})^{p^{\alpha-2}} \cdot \left(g^{p^{\alpha-1}}\right)^{x_1} \cdot (g^{p^\alpha})^{\sum_{i=2}^{t-1} x_i p^{i-2}} = (h)^{p^{\alpha-2}} \\ &\implies \left(g^{p^{\alpha-1}}\right)^{x_1} = (h \cdot g^{-x_0})^{p^{\alpha-2}}, \quad \text{since } g^{p^\alpha} = 1_G. \end{aligned}$$

We can now use our oracle to obtain  $x_1 \in \{0, 1, 2, \dots, p-1\}$ .

Assuming we have successfully solved for  $x_0, x_1 \in \{0, 1, 2, \dots, p-1\}$ , we raise both sides of  $g^x = h$  to the power  $p^{\alpha-3}$  and obtain:

$$\begin{aligned} & \left( g^{\sum_{i=0}^1 x_i p^i + x_2 p^2 + \sum_{i=3}^{t-1} x_i p^i} \right)^{p^{\alpha-3}} = (h)^{p^{\alpha-3}} \\ \implies & \left( g^{\sum_{i=0}^1 x_i p^i} \right)^{p^{\alpha-3}} \left( g^{x_2 p^2} \right)^{p^{\alpha-3}} \left( g^{\sum_{i=3}^{t-1} x_i p^i} \right)^{p^{\alpha-3}} = (h)^{p^{\alpha-3}} \\ \implies & \left( g^{\sum_{i=0}^1 x_i p^i} \right)^{p^{\alpha-3}} \left( g^{p^{\alpha-1}} \right)^{x_2} \left( g^{p^\alpha} \right)^{\sum_{i=3}^{t-1} x_i p^{i-3}} = (h)^{p^{\alpha-3}} \\ \implies & \left( g^{p^{\alpha-1}} \right)^{x_2} = \left( h \cdot g^{-\sum_{i=0}^{2-1} x_i p^i} \right)^{p^{\alpha-3}}, \quad \text{since } g^{p^\alpha} = 1_G. \end{aligned}$$

We can now use our oracle to obtain  $x_2 \in \{0, 1, 2, \dots, p-1\}$ .

Assuming we have successfully solved for  $x_0, x_1, \dots, x_{k-1} \in \{0, 1, 2, \dots, p-1\}$ , we raise both sides of  $g^x = h$  to the power  $p^{\alpha-k-1}$  and obtain:

$$\begin{aligned} & \left( g^{\sum_{i=0}^{k-1} x_i p^i + x_k p^k + \sum_{i=k+1}^{t-1} x_i p^i} \right)^{p^{\alpha-k-1}} = (h)^{p^{\alpha-k-1}} \\ \implies & \left( g^{\sum_{i=0}^{k-1} x_i p^i} \right)^{p^{\alpha-k-1}} \left( g^{x_k p^k} \right)^{p^{\alpha-k-1}} \left( g^{\sum_{i=k+1}^{t-1} x_i p^i} \right)^{p^{\alpha-k-1}} = (h)^{p^{\alpha-k-1}} \\ \implies & \left( g^{\sum_{i=0}^{k-1} x_i p^i} \right)^{p^{\alpha-k-1}} \left( g^{p^{\alpha-1}} \right)^{x_k} \left( g^{p^\alpha} \right)^{\sum_{i=k+1}^{t-1} x_i p^{i-k-1}} = (h)^{p^{\alpha-k-1}} \\ \implies & \left( g^{p^{\alpha-1}} \right)^{x_k} = \left( h \cdot g^{-\sum_{i=0}^{k-1} x_i p^i} \right)^{p^{\alpha-k-1}}, \quad \text{since } g^{p^\alpha} = 1_G. \end{aligned}$$

We can now use our oracle to obtain  $x_k \in \{0, 1, 2, \dots, p-1\}$ .

- The preceding two observations together imply that the original discrete logarithm problem in the group  $G$ , with  $|G| = N = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_t^{\alpha_t}$ , can be reduced to a number of associated DLP's in groups of order  $p_i$ ,  $i = 1, 2, \dots, t$ . If each of these associated DLP's can be solved sufficiently quickly in practice, then so can the original DLP in  $G$ . These associated DLP's in groups of order  $p_i$  will be solved in practice by methods other than Pohlig-Hellman, such as by brute force, or by Shank's Baby-Step-Giant-Step method (*i.e.* the "oracle" mentioned above is either brute-force, or Baby-Step-Giant-Step, *etc.*). In particular, if each  $p_i$  is "small" (relative to the capacity of the brute-force or the Baby-Step-Giant-Step methods), then the original DLP in  $G$  can be solved.

## 5.3 The Index Calculus Method