# Discrete Logarithms

Cryptanalysis

2016.09

# Contents





# Chapter 2





Section 2.2

# THE DISCRETE LOGARITHM PROBLEM

# The Discrete Logarithm Problem

- Let $p$ be a (large) prime
- By the primitive root theorem, there is a primitive element $g \in \mathbb{F}_p$
- The list of elements

$$1, g, g^2, g^3, \ldots, g^{p-2} \in \mathbb{F}_p^*$$

is a complete list of the elements in $\mathbb{F}_p^*$



# The Discrete Logarithm Problem

**Definition**. (Discrete Logarithm Problem, DLP)

- Let $g$ be a primitive root for $\mathbb{F}_p$ and let $h$ be a nonzero element of $\mathbb{F}_p$
- The *Discrete Logarithm Problem* (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 $\log_g h$



# The Discrete Logarithm Problem

Remark.

- Fermat's little theorem tells us that

$$g^{p-1} \equiv 1 \ (mod\ p)$$

- If $x$ is a solution to $g^x = h$, then $x + k(p-1)$ is also a solution for every value of $k$, because

$$g^{x+k(p-1)} = g^x \cdot (g^{p-1})^k \equiv h \cdot 1^k \equiv h \ (mod\ p)$$

- $\log_g h$ is uniquely defined up to modulo $p - 1$



# The Discrete Logarithm Problem

Example. $g = 627, p = 941$

- Powers

| $n$ | $g^n \bmod p$ |
|---|---|
| 1 | 627 |
| 2 | 732 |
| 3 | 697 |
| 4 | 395 |
| 5 | 182 |
| 6 | 253 |
| 7 | 543 |
| 8 | 760 |
| 9 | 374 |
| 10 | 189 |

| $n$ | $g^n \bmod p$ |
|---|---|
| 11 | 878 |
| 12 | 21 |
| 13 | 934 |
| 14 | 316 |
| 15 | 522 |
| 16 | 767 |
| 17 | 58 |
| 18 | 608 |
| 19 | 111 |
| 20 | 904 |

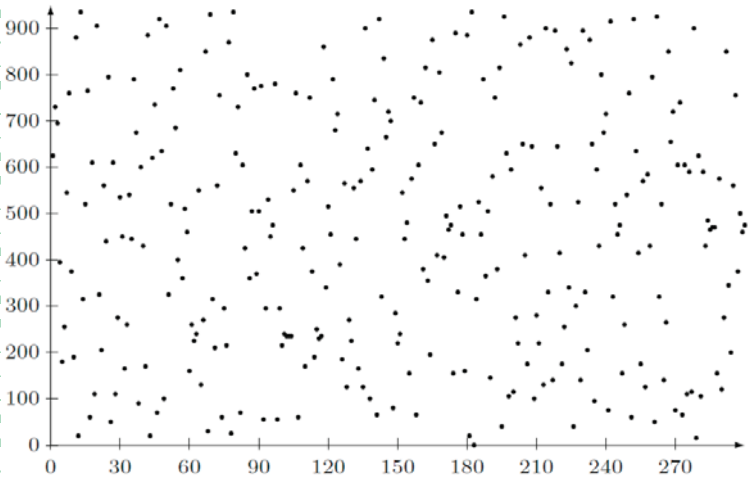
Discrete logarithms

| $h$ | $\log_g(h)$ |
|---|---|
| 1 | 0 |
| 2 | 183 |
| 3 | 469 |
| 4 | 366 |
| 5 | 356 |
| 6 | 652 |
| 7 | 483 |
| 8 | 549 |
| 9 | 938 |
| 10 | 539 |

| $h$ | $\log_g(h)$ |
|---|---|
| 11 | 429 |
| 12 | 835 |
| 13 | 279 |
| 14 | 666 |
| 15 | 825 |
| 16 | 732 |
| 17 | 337 |
| 18 | 181 |
| 19 | 43 |
| 20 | 722 |

# The Discrete Logarithm Problem

- Powers $627^i$ mod 941 for $i = 1, 2, 3, \ldots$






# The Discrete Logarithm Problem

<u>Remark.</u>

- It is not strictly necessary to assume that the base $g$ is a primitive root modulo $p$
- The DLP is the determination of an exponent $x$ satisfying $g^x \equiv h\ (mod\ p)$, assuming that such an $x$ exists
- More generally, we can take elements of any group instead of $\mathbb{F}_p^*$



# The Discrete Logarithm Problem

**<u>Definition</u>.**

- Let $G$ be a group whose group law we denote by the symbol $\star$
- The DLP for $G$ is to determine, for any two given elements $g$ and $h$ in $G$, an integer $x$ satisfying

$$\underbrace{g \star g \star g \star \cdots \star g}_{x \text{ times}} = h$$



Section 2.6

# HOW HARD IS THE DISCRETE LOGARITHM PROBLEM?

## Order Notation

- How can we quantify "hard"?
- A natural measure of hardness is the approximate number of operations necessary to solve the problem using the most efficient method currently known
- Order notation provides a handy way to get a grip on the magnitude of quantities



## Order Notation

**Definition.**

- Let $f(x)$ and $g(x)$ be functions of $x$ taking values that are positive. We say "$f$ is big-$\mathcal{O}$ of $g$" and write
$$f(x) = \mathcal{O}(g(x))$$
if there are positive constants $c$ and $C$ such that $f(x) \le cg(x)$ for all $x \ge C$.
- In particular, we write $f(x) = \mathcal{O}(1)$ if $f(x)$ is bounded for all $x \ge C$



## Order Notation

**Proposition.**

- If the limit
$$\lim_{x \to \infty} \frac{f(x)}{g(x)}$$
exists (and is finite), then $f(x) = \mathcal{O}(g(x))$



## Order Notation

*Proof.*

- Let $L$ be the limit. For any $\epsilon > 0$, there is a constant $C_\epsilon$ such that
$$\left| \frac{f(x)}{g(x)} - L \right| < \epsilon \qquad \text{for all } x > C_\epsilon$$
- In particular, taking $\epsilon = 1$, we find that
$$\frac{f(x)}{g(x)} < L + 1 \qquad \text{for all } x > C_1$$

# Order Notation

- Example. $2x^3 - 3x^2 + 7 = \mathcal{O}(x^3)$, since

$$\lim_{x\to\infty} \frac{2x^3 - 3x^2 + 7}{x^3} = 2$$

- Example – Exercise.

(a) $x^2 + \sqrt{x} = \mathcal{O}\left(x^2\right)$

(b) $5 + 6x^2 - 37x^5 = \mathcal{O}\left(x^5\right)$

(c) $k^{300} = \mathcal{O}\left(2^k\right)$

(d) $(\ln k)^{375} = \mathcal{O}\left(k^{0.001}\right)$

(e) $k^2 2^k = \mathcal{O}\left(e^{2k}\right)$.

(f) $N^{10} 2^N = \mathcal{O}\left(e^N\right)$.



# Order Notation

**Definition**.

- Suppose that there is a constant $A \geq 0$, independent of the size of the input, such that if the input is $\mathcal{O}(k)$ bits long, then it takes $\mathcal{O}(k^A)$ steps to solve the problem. Then the problem is said to be solvable in *polynomial time*.
- Polynomial-time algorithms are considered to be fast algorithms



# Order Notation

- On the other hand, if there is a constant $c > 0$ such that for inputs of size $\mathcal{O}(k)$ bits, there is an algorithm to solve the problem in $\mathcal{O}\left(e^{ck}\right)$ steps, then the problem is solvable in *exponential time*
- Exponential-time algorithms are considered to be slow algorithms



# Order Notation

- Intermediate between polynomial-time algorithms and exponential-time algorithms are *subexponential-time* algorithms
- These have the property that for every $\epsilon > 0$, they solve the problem in $\mathcal{O}_\epsilon\left(e^{\epsilon k}\right)$ steps
  - The notation $\mathcal{O}_\epsilon$ means that the constants $c$ and $C$ appearing in the definition of order notation are allowed to depend on $\epsilon$

## Difficulty of DLP

- Consider the discrete logarithm problem $g^x = h$ in $G = \mathbb{F}_p$
- Suppose $p$ is chosen between $2^k$ and $2^{k+1}$, then $g$, $h$, and $p$ all require at most $k$ bits, so the problem can be stated in $\mathcal{O}(k)$-bits
- It takes $\mathcal{O}(p) = \mathcal{O}(2^k)$ steps to solve the DLP using the trial-and-error method
- This algorithm takes exponential time



## Difficulty of DLP

- There are faster ways to solve the DLP in $\mathbb{F}_p^*$
  - Collision Algorithm (Babystep–Giantstep Algorithm)
  - Pohlig–Hellman Algorithm
  - Index Calculus
- The index calculus solves the DLP in

$$\mathcal{O}\left(e^{c\sqrt{(\log p)(\log\log p)}}\right)$$

steps, so it is a subexponential algorithm



## Difficulty of DLP

- The discrete logarithm problems in different groups may display different levels of difficulty for their solution
- The DLP in $\mathbb{F}_p$ with addition has a linear-time solution
- The best known general algorithm to solve the DLP in $\mathbb{F}_p^*$ with multiplication is subexponential



## Difficulty of DLP

- The discrete logarithm problem for elliptic curves is believed to be even more difficult
- If the elliptic curve group is chosen carefully and has $N$ elements, then the best known algorithm to solve the DLP requires $\mathcal{O}(\sqrt{N})$ steps
- Thus it currently takes exponential time to solve the elliptic curve discrete logarithm problem (ECDLP)

Section 2.7

# A COLLISION ALGORITHM FOR THE DLP



# Trivial Bound for DLP

**Proposition.**

- Let $G$ be a group and let $g \in G$ be an element of order $N$
- Then the discrete logarithm problem
$$g^x = h$$
can be solved in $\mathcal{O}(N)$ steps and $\mathcal{O}(1)$ storage, where each step consists of multiplication by $g$



# Trivial Bound for DLP

*Proof.*

- Simply compute $g, g^2, g^3, \dots$
- Only need to store two values $g, g^k$ at a time
- If a solution to $g^x = h$ exists, then $h$ will appear before we reach $g^N$



# Shanks's Babystep–Giantstep Algorithm

**Proposition.**

- Let $G$ be a group and let $g \in G$ be an element of order $N \geq 2$
- The following algorithm solves the DLP $g^x = h$ in $\mathcal{O}\left(\sqrt{N} \cdot \log N\right)$ steps and $\mathcal{O}\left(\sqrt{N}\right)$ storage

## Shanks's Babystep–Giantstep Algorithm

1. Let $n = 1 + \lfloor\sqrt{N}\rfloor$
2. Create two lists

   List 1: $e,\ g,\ g^2,\ g^3,\ \ldots,\ g^n,$

   List 2: $h,\ h \cdot g^{-n},\ h \cdot g^{-2n},\ h \cdot g^{-3n},\ \ldots,\ h \cdot g^{-n^2}$
3. Find a match between the two lists, say
$$g^i = hg^{-jn}$$
4. Then $x = i + jn$ is a solution to $g^x = h$



## Shanks's Babystep–Giantstep Algorithm

*Proof.*

- When creating List 2, we compute $u = g^{-n}$ first, then compile List 2 by computing
$$h, hu, hu^2, \ldots, hu^n$$
- Thus creating the two lists takes approximately $2n$ multiplications
- Assuming that a match exists, we can find a match in a small multiple of $n \log n$ steps using standard sorting and searching algorithms



## Shanks's Babystep–Giantstep Algorithm

- The total running time for the algorithm is
$$\mathcal{O}(n \log n) = \mathcal{O}\left(\sqrt{N} \cdot \log N\right)$$
- The lists in Step (2) have length $n$, so require $\mathcal{O}\left(\sqrt{N}\right)$ storage
- To prove that the algorithm works, we must show that Lists 1 and 2 always have a match



## Shanks's Babystep–Giantstep Algorithm

- Let $x$ be the unknown solution to $g^x = h$ and write $x$ as
$$x = nq + r$$
with $0 \le r < n$
- Since $1 \le x < N$ and $n > \sqrt{N}$,
$$q = \frac{x - r}{n} < \frac{N}{n} < n$$
- Thus $g^r = hg^{-qn}$ with $0 \le r, q < n$, where $g^r$ is in List 1 and $hg^{-qn}$ is in List 2

## Shanks's Babystep–Giantstep Algorithm

Example.

- Solve $g^x = h$ in $\mathbb{F}_p^*$ with $g = 9704, h = 13896,$ and $p = 17389$
- The number 9704 has order 1242 in $\mathbb{F}_p^*$
- Set $n = 1 + \lfloor\sqrt{1242}\rfloor = 36$ and $u = 9704^{-36} = 2494$
- Construct List 1 and List 2 in the following table



## Shanks's Babystep–Giantstep Algorithm

| $k$ | $g^k$ | $h \cdot u^k$ |
|---|---|---|
| 1 | 9704 | 347 |
| 2 | 6181 | 13357 |
| 3 | 5763 | 12423 |
| 4 | 1128 | 13153 |
| 5 | 8431 | 7928 |
| 6 | 16568 | 1139 |
| 7 | **14567** | 6259 |
| 8 | 2987 | 12013 |

| $k$ | $g^k$ | $h \cdot u^k$ |
|---|---|---|
| 9 | 15774 | 16564 |
| 10 | 12918 | 11741 |
| 11 | 16360 | 16367 |
| 12 | 13259 | 7315 |
| 13 | 4125 | 2549 |
| 14 | 16911 | 10221 |
| 15 | 4351 | 16289 |
| 16 | 1612 | 4062 |

| $k$ | $g^k$ | $h \cdot u^k$ |
|---|---|---|
| 17 | 10137 | 10230 |
| 18 | 17264 | 3957 |
| 19 | 4230 | 9195 |
| 20 | 9880 | 13628 |
| 21 | 9963 | 10126 |
| 22 | 15501 | 5416 |
| 23 | 6854 | 13640 |
| 24 | 15680 | 5276 |

| $k$ | $g^k$ | $h \cdot u^k$ |
|---|---|---|
| 25 | 4970 | 12260 |
| 26 | 9183 | 6578 |
| 27 | 10596 | 7705 |
| 28 | 2427 | 1425 |
| 29 | 6902 | 6594 |
| 30 | 11969 | 12831 |
| 31 | 6045 | 4754 |
| 32 | 7583 | **14567** |



## Shanks's Babystep–Giantstep Algorithm

- From the table we find the collision
$$9704^7 = 14567 = 13896 \cdot 2494^{32}$$
- Using the fact $2494 = 9704^{-36}$, we compute
$$13896 = 9704^7(9704^{36})^{32} = 9704^{1159}$$
- Hence $x = 1159$ solves the problem
$$9704^x = 13896 \text{ in } \mathbb{F}_{17389}^*$$



Section 2.9

# THE POHLIG–HELLMAN ALGORITHM

## Pohlig–Hellman Algorithm

**Theorem**.

- Let $G$ be a group, and suppose that we have an algorithm to solve the DLP in $G$ for any element whose order is a power of a prime
- To be concrete, if $g \in G$ has order $q^e$, suppose that we can solve $g^x = h$ in $\mathcal{O}(S_{q^e})$ steps



## Pohlig–Hellman Algorithm

- Now let $g \in G$ be an element of order $N$, and suppose that $N$ factors into a product of prime powers as $N = q_1^{e_1} q_2^{e_2} \cdots q_t^{e_t}$
- Then the DLP $g^x = h$ can be solved in

$$\mathcal{O}\left(\sum_{i=1}^{t} S_{q_i^{e_i}} + \log N\right)$$

steps



## Pohlig–Hellman Algorithm

1. For each $1 \leq i \leq t$, let

$$g_i = g^{\frac{N}{q_i^{e_i}}} \quad \text{and} \quad h_i = h^{\frac{N}{q_i^{e_i}}}$$

   - Notice that $g_i$ has prime power order $q_i^{e_i}$
   - Use the given algorithm to solve the DLP $g_i^y = h_i$ and let $y = y_i$ be such a solution
2. Use the Chinese remainder theorem to solve

$$x \equiv y_1 \pmod{q_1^{e_1}}, \quad x \equiv y_2 \pmod{q_2^{e_2}}, \quad \ldots, \quad x \equiv y_t \pmod{q_t^{e_t}}$$



## Pohlig–Hellman Algorithm

*Proof*.

- The running time is clear
  - Step (1) takes $\mathcal{O}\left(\sum_{i=1}^{t} S_{q_i^{e_i}}\right)$ steps
  - Step (2) takes $\mathcal{O}(\log N)$ steps
- In practice, the Chinese remainder theorem computation is usually negligible compared to the discrete logarithm computations

# Pohlig–Hellman Algorithm

- It remains to show that Steps (1) and (2) give a solution to $g^x = h$
- Let $x$ be a solution to the system of congruences, then for each $i$,

$$x = y_i + q_i^{e_i} z_i$$

for some $z_i$



# Pohlig–Hellman Algorithm

- This allows us to compute

$$\begin{aligned}
(g^x)^{N/q_i^{e_i}} &= \left(g^{y_i + q_i^{e_i} z_i}\right)^{N/q_i^{e_i}} \\
&= \left(g^{N/q_i^{e_i}}\right)^{y_i} \cdot g^{N z_i} \\
&= \left(g^{N/q_i^{e_i}}\right)^{y_i} \\
&= g_i^{y_i} \\
&= h_i \\
&= h^{N/q_i^{e_i}}
\end{aligned}$$



# Pohlig–Hellman Algorithm

- In terms of discrete logarithms to the base $g$, we can rewrite the above identity as

$$\frac{N}{q_i^{e_i}} \cdot x \equiv \frac{N}{q_i^{e_i}} \cdot \log_g(h) \pmod{N}$$

- Since the numbers

$$\frac{N}{q_1^{e_1}}, \frac{N}{q_2^{e_2}}, \ldots, \frac{N}{q_t^{e_t}}$$

have no nontrivial common factor, we can find integers $c_1, c_2, \ldots, c_t$ such that



# Pohlig–Hellman Algorithm

$$\frac{N}{q_1^{e_1}} \cdot c_1 + \frac{N}{q_2^{e_2}} \cdot c_2 + \cdots + \frac{N}{q_t^{e_t}} \cdot c_t = 1$$

by repeating application of the extended Euclidean theorem

- Therefore

$$\sum_{i=1}^{t} \frac{N}{q_i^{e_i}} \cdot c_i \cdot x \equiv \sum_{i=1}^{t} \frac{N}{q_i^{e_i}} \cdot c_i \cdot \log_g(h) \pmod{N}$$

- Hence $x = \log_g h \ (mod\ N)$

## Pohlig–Hellman Algorithm

**Proposition.**

- Let $G$ be a group and $q$ be a prime
- Suppose that we know an algorithm that takes $S_q$ steps to solve the discrete logarithm problem $g^x = h$ in $G$ whenever $g$ has order $q$
- Let $g \in G$ be an element of order $q^e$ with $e \geq 1$
- Then we can solve the DLP $g^x = h$ in $O(eS_q)$ steps



## Pohlig–Hellman Algorithm

*Proof.*

- The key idea to proving the proposition is to write the unknown exponent $x$ in the form

$$x = x_0 + x_1 q + x_2 q^2 + \cdots + x_{e-1} q^{e-1}$$

with $0 \leq x_i < q$, and then determine successively $x_0, x_1, x_2, \ldots$



## Pohlig–Hellman Algorithm

- The element $g^{q^{e-1}}$ is of order $q$. This allows us to compute

$$\begin{aligned} h^{q^{e-1}} &= (g^x)^{q^{e-1}} \\ &= \left(g^{x_0+x_1q+x_2q^2+\cdots+x_{e-1}q^{e-1}}\right)^{q^{e-1}} \\ &= g^{x_0 q^{e-1}} \cdot \left(g^{q^e}\right)^{x_1+x_2q+\cdots+x_{e-1}q^{e-2}} \\ &= \left(g^{q^{e-1}}\right)^{x_0} \end{aligned}$$



## Pohlig–Hellman Algorithm

- By assumption, we can solve the DLP $\left(g^{q^{e-1}}\right)^{x_0} = h^{q^{e-1}}$ in $S_q$ steps and obtain $x_0$
- We next do a similar computation

$$\begin{aligned} h^{q^{e-2}} &= (g^x)^{q^{e-2}} \\ &= \left(g^{x_0+x_1q+x_2q^2+\cdots+x_{e-1}q^{e-1}}\right)^{q^{e-2}} \\ &= g^{x_0 q^{e-2}} \cdot g^{x_1 q^{e-1}} \cdot \left(g^{q^e}\right)^{x_2+x_3q+\cdots+x_{e-1}q^{e-3}} \\ &= g^{x_0 q^{e-2}} \cdot g^{x_1 q^{e-1}}. \end{aligned}$$

# Pohlig–Hellman Algorithm

- In order to find $x_1$, we must solve the DLP

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

  - Keep in mind that $x_0$ is known
- Again applying the given algorithm, we can solve this in $S_q$ steps
- In $\mathcal{O}(2S_q)$ steps, $x_0$ and $x_1$ are determined



# Pohlig–Hellman Algorithm

- Similarly, we find $x_2$ by solving the DLP

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

- In general, after we have determined $x_0, x_1, \dots, x_{i-1}$, the value of $x_i$ is obtained by solving

$$\left(g^{q^{e-1}}\right)^{x_i} = \left(h \cdot g^{-x_0 - x_1 q - \cdots - x_{i-1} q^{i-1}}\right)^{q^{e-i-1}}$$



# Pohlig–Hellman Algorithm

- Each of these is a DLP whose base is of order $q$, so each of them can be solved in $S_q$ steps
- Hence after $\mathcal{O}\left(eS_q\right)$ steps, we obtain an exponent $x = x_0 + x_1 q + \cdots + x_{e-1} q^{e-1}$ satisfying $g^x = h$, thus solving the original DLP



# Pohlig–Hellman Algorithm

Example. Solve $5448^x = 6909$ in $\mathbb{F}_{11251}^*$

- $p = 11251$ is a prime and $5^4 | (p-1)$
- $5448$ has order $5^4$
- First, solve $\left(5448^{5^3}\right)^{x_0} = 6909^{5^3}$, which is in fact $11089^{x_0} = 11089$, hence $x_0 = 1$
- The next step is to solve

$$\left(5448^{5^3}\right)^{x_1} = (6909 \cdot 5448^{-x_0})^{5^2} = (6909 \cdot 5448^{-1})^{5^2}$$

which reduces to $11089^{x_1} = 3742$

# Pohlig–Hellman Algorithm

- Note that we only need to check values of $x_1$ between 1 and 4, and the solution is $x_1 = 2$
- Continuing, we next solve

$$\left(5448^{5^3}\right)^{x_2} = \left(6909 \cdot 5448^{-x_0 - x_1 \cdot 5}\right)^5 = \left(6909 \cdot 5448^{-11}\right)^5$$

which is $11089^{x_2} = 1$, $x_2 = 0$

- The final step is to solve

$$\left(5448^{5^3}\right)^{x_3} = 6909 \cdot 5448^{-x_0 - x_1 \cdot 5 - x_2 \cdot 5^2} = 6909 \cdot 5448^{-11}$$



# Pohlig–Hellman Algorithm

- This reduces to solving $11089^{x_3} = 6320$, which has the solution $x_3 = 4$
- Hence the final answer is

$$x = 511 = 1 + 2 \cdot 5 + 4 \cdot 5^3$$



# Pohlig–Hellman Algorithm

Example. $23^x = 9689$ in $\mathbb{F}_{11251}^*$

- $p = 11251, g = 23, h = 9689$
- $N = p - 1 = 11250 = 2 \times 3^2 \times 5^4$
- The first step is solve three subsidiary discrete logarithm problems

| $q$ | $e$ | $g^{(p-1)/q^e}$ | $h^{(p-1)/q^e}$ | Solve $\left(g^{(p-1)/q^e}\right)^x = h^{(p-1)/q^e}$ for $x$ |
|---|---|---|---|---|
| 2 | 1 | 11250 | 11250 | 1 |
| 3 | 2 | 5029 | 10724 | 4 |
| 5 | 4 | 5448 | 6909 | 511 |



# Pohlig–Hellman Algorithm

- The second step is to use the Chinese remainder theorem to solve the simultaneous congruences
  - $x \equiv 1 \pmod{2}$
  - $x \equiv 4 \pmod{3^2}$
  - $x \equiv 511 \pmod{5^4}$
- The smallest solution is $x = 4261$