5. Discrete logarithm problem.

A. Ushakov

MA503, October 5, 2022

Contents

Today we discuss methods to solve the discrete logarithm problem.

- Discrete logarithm problem.
- Diffie-Hellman key exchange. Example.
- Computational Diffie-Hellman problem.
- ElGamal public key cryptosystem. Example.
- ElGamal security.
- Babystep-giantstep algorithm.
- Pohlig–Hellman algorithm.
- Index calculus algorithm.
-

Discrete logarithm problem

Fix a modulus $n \in \mathbb{N}$ and $g, h \in U_n$.

Definition

$x \in \mathbb{Z}$ is the **discrete logarithm of h to the base g modulo n** if $g^x \% n = h$.

For $n = 11$ we get

$2^0 \equiv_{11} 1$	$\log_2(1) = 0$	$3^0 \equiv_{11} 1$	$\log_3(1) = 0$
$2^1 \equiv_{11} 2$	$\log_2(2) = 1$	$3^1 \equiv_{11} 3$	$\log_3(3) = 1$
$2^2 \equiv_{11} 4$	$\log_2(4) = 2$	$3^2 \equiv_{11} 9$	$\log_3(9) = 2$
$2^3 \equiv_{11} 8$	$\log_2(8) = 3$	$3^3 \equiv_{11} 5$	$\log_3(5) = 3$
$2^4 \equiv_{11} 5$	$\log_2(5) = 4$	$3^4 \equiv_{11} 4$	$\log_3(4) = 4$
$2^5 \equiv_{11} 10$	$\log_2(10) = 5$	$3^5 \equiv_{11} 1$	$\log_3(1) = 5???$
$2^6 \equiv_{11} 9$	$\log_2(9) = 6$		$\log_3(2) = -$
$2^7 \equiv_{11} 7$	$\log_2(7) = 7$		$\log_3(6) = -$
$2^8 \equiv_{11} 3$	$\log_2(3) = 8$		$\log_3(7) = -$
$2^9 \equiv_{11} 6$	$\log_2(6) = 9$		$\log_3(8) = -$
$2^{10} \equiv_{11} 1$	$\log_2(1) = 10???$		$\log_3(10) = -$

$\log_g(x)$ *is defined modulo* $|g|$.

Because the values of $g^x \% n$ repeat after $|g|$ steps.

Diffie-Hellman (DH) key exchange

The goal of a key exchange protocol is to allow two parties establish a common shared key.

Key generation (performed by Alice or by Bob):

- Choose a prime modulus p and a primitive root g modulo p.

Encryption step performed by Alice:

- Choose a random $a \in \mathbb{N}$; compute $A = g^a \ \% \ p$ and send it to Bob.

Encryption step performed by Bob:

- Choose a random $b \in \mathbb{N}$; compute $B = g^b \ \% \ p$ and send it to Alice.

Computing the shared key (performed by Alice): $K = B^a \ \% \ p$.

Computing the shared key (performed by Bob): $K = A^b \ \% \ p$.

It is easy to check that

$$B^a \ \% \ p = g^{ab} \ \% \ p = A^b \ \% \ p.$$

DH: example

Key generation:

- Choose a prime modulus $p = 13$ and $g = 2$.

Encryption step performed by Alice:

- Choose $a = 11$, compute $A = 2^{11} \ \% \ 13 = 7$, and send it to Bob.

Encryption step performed by Bob:

- Choose $b = 7$, compute $B = 2^{7} \ \% \ 13 = 11$, and send it to Alice.

Computing the shared key (performed by Alice): $K = 11^{11} \ \% \ 13 = 6$.
Computing the shared key (performed by Bob): $K = 7^{7} \ \% \ 13 = 6$.

DH: informal discussion of security

A passive adversary Eve collects the following information:

- A prime modulus p.
- A primitive root g modulo p.
- A number A constructed as $A = g^a \% p$ for some a secret a.
- A number B constructed as $B = g^b \% p$ for some a secret b.

Eve's goal is to compute $K = g^{ab} \% p$ using g, A, B.

(Computational Diffie-Hellman problem (CDH))

Given a triple $(g, g^a \% p, g^b \% p)$ compute $g^{ab} \% p$.

It is easy to see that CDH is not harder than DLP because if Eve can compute $a = \log_g(A)$ and $b = \log_g(B)$, then she can compute g^{ab} % p.

If we can solve DLP modulo p, then we can solve CDH.

The converse is not known to be true. Yet, computing the discrete logarithm is the only known method for solving CDH.

ElGamal public key cryptosystem

Key generation (performed by Alice):

- Pick a random prime p.
- Pick a primitive root g of p.
- Pick a random integer $a \in \{2, \ldots, p-2\}$ and compute $A = g^a \% p$.

Finally, Alice publishes the triple (p, g, A), called **Alice's public key**.

Encryption (performed by Bob):

To encrypt the message $1 \leq m \leq p-1$ Bob

- picks a (secret) random $j \in \{2, \ldots, p-2\}$;
- computes $c_1 = g^j \ \% \ p$ and $c_2 = mA^j \ \% \ p$;
- sends the pair (c_1, c_2) to Alice.

Decryption (performed by Alice):

- Alice computes $\frac{c_2}{c_1^a}$ modulo p. The obtained number is m.

It is easy to check that $m = \frac{c_2}{c_1^a} \ \% \ p$ because

$$\frac{c_2}{c_1^a} = \frac{mA^j}{(g^j)^a} = \frac{mA^j}{(g^a)^j} = \frac{mA^j}{A^j} = m.$$

Thus, Alice indeed obtains Bob's plaintext m.

ElGamal public key cryptosystem: example

Key generation (performed by Alice):

- Alice chooses $p = 17$, $g = 3$, and $a = 6$.
- Hence $A = 3^6 \ \% \ 17 = 15$.
- $(17, 3, 15)$ – public key.

Encryption: To encrypt $m = 2$, Bob picks a random $j = 4$ and computes

$$c_1 = 3^4 \ \% \ 17 = 13 \text{ and } c_2 = 2 \cdot (15)^4 \ \% \ 17 \equiv 2(-2)^4 = 32 \equiv 15.$$

The pair $c_1 = 13$,$c_2 = 15$ is sent to Alice.

Decryption: To decrypt Alice computes $15 \cdot 13^{-6}$:

$$13^6 = 4^6 = (16)^3 = -1$$

hence $15 \cdot 13^{-6} = -15 = 2$ which is the correct value of m.

ElGamal: informal discussion of security

A passive adversary Eve collects the following information:

- A prime modulus p.
- A primitive root g of p.
- A number A constructed as $A = g^a \% p$ for some a secret a.
- A pair $c_1 = g^j \ \% \ p$ and $c_2 = mA^j \ \% \ p$ constructed for secret m, j.

Alice can decrypt m since she knows $a = \log_g(A)$.

If Eve can efficiently compute $\log_g(A)$ modulo p, then she can find m.

Proposition

If Eve can decrypt arbitrary ElGamal ciphertexts, then she can solve CDH.

For a given instance $(g, A = g^a, B = g^b)$ and a prime modulus p, Eve decrypts ElGamal ciphertext

$$c_1 = B, \ c_2 = 1$$

that produces $\frac{c_2}{c_1^a} \equiv_p B^{-a} \equiv_p g^{-ab}$. Computing the inverse $\left(g^{-ab}\right)^{-1}$ we get the solution g^{ab} of the instance $(g, A = g^a, B = g^b)$ of CDH.

In this sense ElGamal is not weaker than DH.

(You will discuss chosen ciphertext attacks in a proper cryptography course.)

DLP: properties

$h = g^x \quad \Leftrightarrow \quad h = g^{x \pm |g|}$ *for any* $g \in U_n$.

Hence, $\log_g$ defines a number modulo $|g|$, i.e., $\log_g : \langle g \rangle \to \mathbb{Z}_{|g|}$.

$\log_g(ab) = \log_g(a) + \log_g(b)$.

$\log_g(a^z) = z \log_g(a)$.

(Straightforward enumeration of powers)

To compute $\log_g(h)$ *we can compute one-by one powers of* g

$$g^0, g, g^2, g^3, \ldots, g^{|g|-1}$$

until we get h*. In the worst case that requires* $O(|g|)$ *multiplications.*

DLP: Shanks' babystep-giantstep algorithm

(To compute $\log_g(h)$ modulo n)

- *Compute $N = |g|$ and $l = 1 + \lfloor\sqrt{N}\rfloor$ (notice that $l > \sqrt{N}$).*
- *Compute two sequences*

$$\textbf{(babysteps)}\ 1, g, g^2, g^3, \ldots, g^l,$$
$$\textbf{(giantsteps)}\ h, hg^{-l}, hg^{-2l}, hg^{-3l}, \ldots, hg^{-l^2};$$

- *Find a match $g^i = hg^{-jl}$ and output $jl + i$.*

The sequences have a matching pair $\Leftrightarrow$ $\log_g(h)$ is defined.

$$\begin{aligned} \log_g(h) \text{ is defined} \quad &\Leftrightarrow \quad h = g^x \text{ for some } 0 \le x < N \\ &\Leftrightarrow \quad h = g^{jl+i} \text{ for some } 0 \le i, j < l \qquad \text{(divided } x \text{ by } l\text{)} \\ &\Leftrightarrow \quad hg^{-jl} = g^i \text{ for some } 0 \le i, j < l. \end{aligned}$$

It requires $O(l) = O(\sqrt{N})$ multiplications to construct the sequences.

It is easy to find a match in two lists.

Shanks' babystep-giantstep algorithm: example

Compute $\log_2(50)$ *modulo* $n = 67$ *using babystep-giantstep algorithm.*

- Compute $\varphi(67) = 66 = 2 \cdot 3 \cdot 11$.
- Realize that $N = |2| = 66$ because

$$2^{33} \equiv_{67} 66 \not\equiv_{67} 1 \qquad 2^{22} \equiv_{67} 36 \not\equiv_{67} 1 \qquad 2^{6} \equiv_{67} 64 \not\equiv_{67} 1.$$

- Compute $l = 1 + \lfloor\sqrt{66}\rfloor = 9$.
- Compute babystep sequence:

$$\begin{array}{ccccc} 2^0 \equiv_{67} 1 & 2^1 \equiv_{67} 2 & 2^2 \equiv_{67} 4 & 2^3 \equiv_{67} 8 & 2^4 \equiv_{67} \color{red}{16} \\ 2^5 \equiv_{67} 32 & 2^6 \equiv_{67} 64 & 2^7 \equiv_{67} 61 & 2^8 \equiv_{67} 55 & 2^9 \equiv_{67} 43 \end{array}$$

- Compute giantstep sequence (precompute $g^{-n} = 2^{-9} \equiv_{67} 2^{57} \equiv 53$).

$$\begin{array}{ccccc} 50 \equiv_{67} 50 & 50 \cdot 2^{-9} \equiv_{67} 37 & 50 \cdot 2^{-9\cdot 2} \equiv_{67} 18 & 50 \cdot 2^{-9\cdot 3} \equiv_{67} \color{red}{16} & 50 \cdot 2^{-9\cdot 4} \equiv_{67} 44 \\ 50 \cdot 2^{-9\cdot 5} \equiv_{67} 54 & 50 \cdot 2^{-9\cdot 6} \equiv_{67} 48 & 50 \cdot 2^{-9\cdot 7} \equiv_{67} 65 & 50 \cdot 2^{-9\cdot 8} \equiv_{67} 28 & 50 \cdot 2^{-9\cdot 9} \equiv_{67} 10 \end{array}$$

- Find a matching pair $2^4 \equiv_{67} h2^{-9\cdot 3} = h2^{-27}$ and conclude that $h = 2^{31}$ and, hence, $\log_2(50) = 31$.

DLP: Pohlig–Hellman algorithm

Let $x = \log_g(h)$ modulo n and $|g| = N = p_1^{a_1} \dots p_k^{a_k}$. For each $i = 1, \dots, k$ define

$$\mathrm{N_i} = \frac{N}{p_i^{a_i}}, \ \mathrm{g_i} = g^{N_i}, \text{ and } \mathrm{h_i} = h^{N_i}.$$

Lemma

Define $x_i = x \,\%\, p_i^{a_i}$. Then $x_i = \log_{g_i}(h_i)$ modulo n.

$$\begin{aligned}
x_i = x \,\%\, p_i^{a_i} \quad &\Rightarrow \quad x = q_i p_i^{a_i} + x_i \\
&\Rightarrow \quad h_i = h^{N_i} = g^{xN_i} = g^{(q_i p_i^{a_i} + x_i)N_i} = g^{x_i N_i}\left(g^{p_i^{a_i} N_i}\right)^{q_i} \\
&\qquad\quad = g^{x_i N_i}\left(g^{N}\right)^{q_i} \equiv_N g^{x_i N_i} = \left(g^{N_i}\right)^{x_i} = g_i^{x_i} \\
&\Rightarrow \quad x_i = \log_{g_i}(h_i).
\end{aligned}$$

(The Pohlig–Hellman algorithm to compute $\log_g(h)$)

(1) *For each i compute $N_i = \frac{N}{p_i^{a_i}}$, $g_i = g^{N_i}$ and $h_i = h^{N_i}$.*

(2) *For each i compute $x_i = \log_{g_i}(h_i)$.*

(3) *Use CRT to solve the system on the right for x.*

$$\begin{cases} x \equiv_{p_1^{a_1}} x_1 \\ \dots \\ x \equiv_{p_k^{a_k}} x_k \end{cases}$$

Pohlig–Hellman algorithm: complexity

$|g_i| = p_i^{a_i}$.

Because $g_i^{p_i^{a_i}} = \left(g^{N_i}\right)^{p_i^{a_i}} = g^N = 1$.

(Pohlig–Hellman theorem: assumption)

Suppose that we have a collection of algorithms

$$\{\, \mathcal{A}_{p^a} \mid p \text{ is prime and } a \in \mathbb{N} \,\},$$

where each $\mathcal{A}_{p^a}$ solves $\log_g(h)$ for $|g| = p^a$ in time $O(S(p^a))$.

For instance, $\mathcal{A}_{p^a}$ can be the babystep-giantstep algorithm, in which case $S(p^a) = \sqrt{p^a}$.

(Pohlig–Hellman theorem: conclusion)

If $|g| = N = p_1^{a_1} \dots p_k^{a_k}$, then using algorithms $\{\mathcal{A}_{p^a}\}$ we can solve $\log_g(h)$ in time

$$O\left(\sum_{i=1}^{k} S(p^a) + \text{“small CRT overhead”}\right).$$

Pohlig–Hellman algorithm is efficient if $|g|$ is a product of small powers $p_i^{a_i}$.

If $p - 1$ is a product of small powers of primes, then DLP modulo p is easy.

Pohlig–Hellman algorithm: example

Compute $\log_3(24)$ *modulo* 31.

$g = 3$ is a primitive root modulo 31 and $|3| = 30 = 2 \cdot 3 \cdot 5$ in U_{31}. Let $h = 24$.

$$\begin{array}{llll}
N_1 = 15 & g_1 = 3^{15} \equiv_{31} -1 & h_1 = 24^{15} \equiv_{31} -1 & \log_{-1}(-1) = 1 = x_1 \\
N_2 = 10 & g_2 = 3^{10} \equiv_{31} 25 & h_2 = 24^{10} \equiv_{31} 25 & \log_{25}(25) = 1 = x_2 \\
N_3 = 6 & g_3 = 3^{6} \equiv_{31} 16 & h_3 = 24^{6} \equiv_{31} 4 & \log_{16}(4) = 3 = x_3.
\end{array}$$

Finally, solve the system

$$\begin{cases} x \equiv_2 1 \\ x \equiv_3 1 \\ x \equiv_5 3 \end{cases}$$

to get $x = 13$.

Index calculus method

Fix a relatively small B. To compute $\log_g(h)$ modulo p we can compute numbers

$$h,\ hg^{-1},\ hg^{-2},\ hg^{-3},\ hg^{-4},\ hg^{-5},\ \ldots \mod p.$$

If some hg^{-k} is B-smooth, then $hg^{-k} = 2^{a_2}3^{a_3}5^{a_5}\ldots$. But then

$$\begin{aligned}\log_g(h) &= k + \log_g(2^{a_2}3^{a_3}5^{a_5}\ldots) \\ &= k + a_2\log_g(2) + a_3\log_g(3) + a_5\log_g(5) + \ldots,\end{aligned}$$

which can be computed if we know the values of $\log_g(p)$ for each prime $2 \leq p \leq B$.

Q. *Is it easy to compute* $\log_g(p)$ *for small primes* p*?*
A. *No, it is not! Yet, we can try to generate and use some random data.*

Index calculus: example

For a randomly generated i check if $g^i \% p$ is B-smooth. If not, then discard. If $g^i \equiv_p 2^{a_2} 3^{a_3} 5^{a_5} \dots$, then call it a "**relation**" and remember it. After we collect sufficiently many relations, we might be able to compute $\log_g(p)$ for small primes p.

For instance, for $n = 18443$ and $g = 37$ we can choose $B = 5$ and randomly generate

$$
\begin{aligned}
g^{2731} &\equiv_n 2^3 \cdot 3 \cdot 5^4 &\Rightarrow\quad 2731 &\equiv_{|g|} 3\log_g(2) + \log_g(3) + 4\log_g(5)\\
g^{11311} &\equiv_n 2^3 \cdot 5^2 &\Rightarrow\quad 11311 &\equiv_{|g|} 3\log_g(2) + 2\log_g(5)\\
g^{12708} &\equiv_n 2^3 \cdot 3^4 \cdot 5 &\Rightarrow\quad 12708 &\equiv_{|g|} 3\log_g(2) + 4\log_g(3) + \log_g(5)\\
g^{15400} &\equiv_n 2^3 \cdot 3^3 \cdot 5 &\Rightarrow\quad 15400 &\equiv_{|g|} 3\log_g(2) + 3\log_g(3) + \log_g(5).
\end{aligned}
$$

where $|g| = 18442$. Denote $\log_g(p)$ by l_p and combine the equivalences above

$$
\begin{cases}
3l_2 + l_3 + 4l_5 \equiv 2731 \mod 18443\\
3l_2 + 2l_5 \equiv 11311 \mod 18443\\
3l_2 + 4l_3 + l_5 \equiv 12708 \mod 18443\\
3l_2 + 2l_3 + l_5 \equiv 15400 \mod 18443
\end{cases}
$$

Finally solve the system and compute $\log_g(2), \log_g(3), \log_g(5)$

Index calculus: a simple example

For $n = 41$ and $g = 7$ compute $|g| = 40$, choose $B = 5$ and randomly generate

$$\begin{array}{lcl}
g^2 \equiv_{41} 2^3 & \Rightarrow & 2 \equiv_{40} 3\log_g(2) \\
g^3 \equiv_{41} 3 \cdot 5 & \Rightarrow & 3 \equiv_{40} \log_g(3) + \log_g(5) \\
g^4 \equiv_{41} 23 & \Rightarrow & \text{discard} \\
g^5 \equiv_{41} 2 \cdot 19 & \Rightarrow & \text{discard} \\
g^6 \equiv_{41} 2^2 \cdot 5 & \Rightarrow & 6 \equiv_{40} 2\log_g(2) + \log_g(5) \\
g^7 \equiv_{41} 17 & \Rightarrow & \text{discard} \\
g^8 \equiv_{41} 37 & \Rightarrow & \text{discard} \\
g^9 \equiv_{41} 13 & \Rightarrow & \text{discard} \\
g^{10} \equiv_{41} 3^2 & \Rightarrow & 10 \equiv_{40} 2\log_g(3).
\end{array}$$

- $2 \equiv_{40} 3\log_g(2) \quad \Rightarrow \quad \log_g(2) = \frac{2}{3} \equiv_{40} 54 \equiv_{40} 14$.
- We cannot divide $10 \equiv_{40} 2\log_g(3)$ by 2, because 2 is not a unit modulo 40. You can check that $5 \equiv_{40} \log_g(3)$ is wrong!
- $6 \equiv_{40} 2\log_g(2) + \log_g(5) \equiv_{40} 2 \cdot 14 + \log_g(5)$. Hence, $\log_g(5) \equiv_{40} -22 \equiv_{40} 18$.
- $3 \equiv_{40} \log_g(3) + \log_g(5) \equiv_{40} \log_g(3) + 18$. Hence, $\log_g(3) \equiv_{40} -15 \equiv_{40} 25$.