Discrete logarithm

Mathematical Cryptography, Lectures 13 - 14

Contents

1. **Diffie-Hellman and ElGamal protocols**
   - Discrete logarithm
   - Diffie-Hellman and ElGamal protocols
   - Security of both protocols
2. **Computing discrete logarithms**
   - Baby step/giant step algorithm
   - Pohling-Hellman algorithm
3. **Finding a generator for $\mathbb{Z}_p^*$**

Discrete logarithm

Definition

Let $G = \langle a \rangle$ be a cyclic group of order n with a generator a.
Each element $b \in G$ has a form $b = a^x$ for unique $x \in \mathbb{Z}_n$.
This x is called the *discrete logarithm* of b to the base a in the group G. It is denoted by $\mathrm{dlog}_a(b)$.

Example

$\mathbb{Z}_9^* = \langle 2 \rangle$, $|\mathbb{Z}_9^*| = 6$.

$b \in \mathbb{Z}_9^*$	1	2	4	5	7	8
$\mathrm{dlog}_2(b)$	0	1	2	5	4	3

Discrete logarithm

Proposition

Let $G = \langle a \rangle$ be a cyclic group of order n.
The exponentiation map $\exp_a : (Z_n, +) \to (G, \cdot) : x \mapsto a^x$
is a group isomorphism.
The discrete logarithm $\mathrm{dlog}_a : (G, \cdot) \to (Z_n, +) : g = a^x \mapsto x$
is its inverse mapping, so it is a group isomorphism too.

For $b, c \in G$, $k \in \mathbb{Z}$ these formulas hold:

- $\mathrm{dlog}_a(b \cdot c) = \mathrm{dlog}_a(b) + \mathrm{dlog}_a(c)$
- $\mathrm{dlog}_a(1) = 0$
- $\mathrm{dlog}_a(b^k) = k \,\mathrm{dlog}_a(b),\ \mathrm{dlog}_a(b^{-1}) = -\mathrm{dlog}_a(b)$

Discrete logarithm

Note

Sometimes the discrete logarithm is introduced in a more general meaning:

Let G be an abelian group, $a, b \in G$.
If $b \in \langle a \rangle$, then $\mathrm{dlog}_a(b)$ is defined as any $x \in \mathbb{Z}$ for which $a^x = b$.
Such x is uniquely determined modulo the order of the element a.
If $b \notin \langle a \rangle$, then $\mathrm{dlog}_a(b)$ is not defined.

If G is a cyclic group and $\mathrm{r}(a) = r$ (here $r \leq |G|$),
then $b \in \langle a \rangle$ if and only if b solves the equation $x^r = 1$.
Therefore, $\mathrm{dlog}_a(b)$ is defined only if $b^{\mathrm{r}(a)} = 1$ in G.

Discrete logarithm

Discrete logarithm problem (DLP)

For most groups computation of discrete logarithms is assumed to be an exponential or subexponential problem - for groups $\mathbb{Z}_p^*$ and their subgroups, or for groups of points on an elliptic curve.
(In the following protocols these groups play the role of G.)

For the group $(\mathbb{Z}_n, +)$ it is not hard to compute the discrete logarithm. Here $\mathrm{dlog}_a(b) = x \in \mathbb{Z}_n$ such that $b = x \cdot a$ in $\mathbb{Z}_n$, because it is an additive group. This linear equation can be solved by the extended Euclidean algorithm in time $O(\mathrm{len}(n)^2)$.

Discrete logarithm in cryptography

- The Diffie-Hellman protocol is a public arrangement of a common secret key for symmetric encryption. Its security relies on the discrete logarithm problem.
- Invented by Whitfield Diffie and Martin Hellman, USA, 1976.
- ElGamal encryption is a non-symmetric public-key encryption that relies on the same idea as the Diffie-Hellman key establishment.
- Invented by Taher ElGamal, USA (Egypt), 1985.

Diffie-Hellman key establishment

Diffie-Hellman protocol

Alice chooses a cyclic group G of order n and its generator a.
Then she chooses $x \in \mathbb{Z}_n$ and computes $b = a^x$ in the group G.
Alice sends to Bob the element b and group informations (G, n, a).

Bob chooses $y \in \mathbb{Z}_n$ and computes $c = a^y$ in the group G.
Bob sends to Alice the element c.

Alice computes $s_A = c^x$ and Bob computes $s_B = b^y$ in G.
Both of them obtain the same secret key $s = s_A = s_B = a^{xy}$.

Diffie-Hellman key establishment

Diffie-Hellman protocol

Alternatively, the third part chooses a cyclic group G of order n and its generator a. Informations (G, n, a) are published in a "phone book", where Alice inserts her public part of key, $b = a^x$ under her name, and Bob inserts his public part of key, $c = a^y$ under his name. Both secret parts of key, x and y will not be revealed to anyone.

Encryption

Alice and Bob can use the established secret key s for a symmetric encryption.

Diffie-Hellman key establishment solves key distribution problem that troubles any symmetric encryption.

ElGamal encryption

ElGamal protocol

Alice chooses a cyclic group G of order n and its generator a.
Then she chooses $x \in \mathbb{Z}_n$ and computes $b = a^x$ in the group G.

Alice posts the element b and the group informations (G, n, a) in a "phone book".
Element b is Alice's public key, while x is Alice's secret key.

ElGamal encryption

Encryption

Bob chooses $y \in \mathbb{Z}_n$, a so-called jepices key.
He computes $c = a^y$, $s = b^y$ (again $s = a^{xy}$) in the group G.

Bob encrypts the message $m \in G$: $\bar{m} = m \cdot s$ in G.
Bob sends to Alice a pair $(c, \bar{m})$.

Decryption

Alice computes $s = c^x$, respectively $s^{-1} = c^{n-x}$ in the group G.

Alice decrypts the message $\bar{m}$: $m = \bar{m} \cdot s^{-1}$ in G.

ElGamal encryption

Example

Alice's public key is $b = 7$ for the group $\mathbb{Z}_{37}^* = \langle 2 \rangle$ of order $n = 36$.
Alice's private key is $x = 32$.

Bob wants to encrypt the message $m = 10$ for Alice, and he uses a jepices key $y = 8$.
He computes $c = 2^8 = 34$, $s = 7^8 = 16$, $\bar{m} = 10 \cdot 16 = 12$ in $\mathbb{Z}_{37}^*$.
Bob sends $(c, \bar{m}) = (34, 12)$.

Alice wants to decrypt the message $\bar{m}$.
She calculates $s^{-1} = 34^{36-32} = 7$, $m = 12 \cdot 7 = 10$ in $\mathbb{Z}_{37}^*$.

ElGamal encryption

For encryption and decryption in ElGamal protocol we need to do:

- exponentiation in the group G, by the repeated squaring algorithm it requires $O(\mathrm{len}(n))$ multiplications in the group G, where $n = |G|$.
- choosing different jepices keys for each message (by a random number generator in $\mathbb{Z}_n$).
  This is necessary for security: if Eve were to decrypt one message $\bar{m}$, she would compute the key $s = \bar{m}m^{-1}$ and then decrypt all messages.
- A disadvantage of the protocol is that the encrypted message is twice as long as the open message. ElGamal encryption is less used than RSA encryption because of this.

ElGamal encryption

To create both protocols, we need a cyclic group and its generator. In practice it is used (see Shoup, 2005):

- G is a subgroup in $\mathbb{Z}_p^*$,
  where p is a prime of 1024 bits - it is sufficient with respect to subexponential algorithms for discrete logarithm,
  $|G| = q$ is a prime of 160 bits - it is sufficient with respect to exponential algorithms for discrete logarithm.
  Messages can be chosen $m \in \mathbb{Z}_p^*$ (and it works), and the exponentiation is faster since exponents are smaller than q.
- G is a group of points on an elliptic curve,
  an order $|G|$ has 160 bits - in these groups, there is no subexponential algorithm for discrete logarithm.
  Messages must be converted to the group G.

Security of both protocols

Discrete logarithm problem

Let $G = \langle a \rangle$ be a group of order n. Discrete logarithm problem is for given $b \in G$ to find $x \in \mathbb{Z}_n$ such that $b = a^x$ in G.

Security of Diffie-Helman and ElGamal protocols relies on a complexity of the discrete logarithm problem in a given group. The following algorithms are known so far:

- exponential algorithms requiring $O(\sqrt{n}) = O(2^{\frac{1}{2}\,\mathrm{len}(n)})$ multiplications in G, which work in every cyclic group (subgroups of $\mathbb{Z}_p^*$, groups of points on elliptic curves);
- a subexponential probabilistic algorithm operating in time $O(e^{(2\sqrt{2}+o(1))\sqrt{\ln(p)\ln(\ln(p))}})$ works only in subgroups of $\mathbb{Z}_p^*$;

The exponential mapping in these groups is an *one-way function*.

Security of both protocols

Diffie-Hellman problem

Let $G = \langle a \rangle$ be a group. From knowledge of $b = a^x$ and $c = a^y$ compute $s = a^{xy}$. The exponents x, y are unknown.

The Diffie-Hellman problem can only be solved via the discrete logarithm so far, i.e. by computing x and y. The Diffie-Hellman problem is believed to have the exponential complexity.

Diffie-Hellman decisional problem

Decide whether a triple (b, c, d) is of the form (a^x, a^y, a^{xy}) in the group $G = \langle a \rangle$.

So far, it is believed that Diffie-Hellman triples (a^x, a^y, a^{xy}) are not distinguishable from random triples (a^x, a^y, a^z) even by probabilistic methods.

Security of both protocols

Notes

- If $|G| = n$ is divisible by small primes, then the discrete logarithm can be computed much faster (in exponential time with much smaller exponent by Pohling-Hellman algorithm).
- If $|G| = n$ is divisible by small primes, then there is a probabilistic algorithm that recognizes the Diffie-Hellman triples in polynomial time $(q + \text{len}(p))^{O(1)}$ with probability $\frac{1}{q}$, where q is the smallest prime that divides n (in case G is a subgroup of $\mathbb{Z}_p^*$).
- Therefore, it is used to choose $|G| = q$, with q prime.

Computing discrete logarithms

Let $G = \langle a \rangle$ be a cyclic group of order n with a generator a. For an element $b \in G$ we want to compute $\mathrm{dlog}_a(b)$, so we look for $x \in \mathbb{Z}_n$ such that $b = a^x$ in G.

Following algorithms work in any cyclic group. For the time complexity only the number of multiplications will be specified, because the time for doing a multiplication differs in each group.

Brute force algorithm

We compute a^i for every $0 \leq i < n$ by successive multiplying by a, until the element b is the result.

Time complexity - we perform at most n multiplications in G, so the complexity is $O(n) = O(2^{\mathrm{len}(n)})$ multiplications in G.

Computing discrete logarithms

Baby step/giant step algorithm

We choose an approximation $m \doteq \sqrt{n}$, then also $m' = [\frac{n}{m}] \doteq \sqrt{n}$. We write $x = \mathrm{dlog}_a(b) = vm + u$, then $0 \leq u < m$, $0 \leq v \leq m'$ because $0 \leq x < n$. We search for the corresponding u, v.

$$b = a^x = a^{vm+u}, \quad \text{thus} \quad b(a^{-m})^v = a^u.$$

Baby steps: We calculate a^i for all $0 \leq i < m$ and we store them in a computer memory.
Suitable implementation - (balanced binary) search tree T, where $T(g) = i$ in case $a^i = g$, and $T(g) = \perp$ for others $g \in G$.
The space complexity is $O(\sqrt{n})$ and each searching takes $O(\mathrm{len}(n))$ time.

Computing discrete logarithms

Baby step/giant step algorithm

Giant steps: We compute $b(a^{n-m})^j$ for $0 \leq j < m'$, until the result equals to one of the baby steps results.
If the same results occurre, then $i = u$ and $j = v$, so we get $\mathrm{dlog}_a(b) = x = vm + u$.

Time complexity - we perform at most $2\sqrt{n}$ multiplications in the group G and $\sqrt{n}$ searches in the tree T. (The time $O(\mathrm{len}(n))$ for searching is shorter than the time for multipling in any group.)
The time complexity is $O(\sqrt{n}) = O(2^{\frac{1}{2}\,\mathrm{len}(n)})$ multiplications in G, so exponential with half of the exponent compared to brute force.

The space complexity is also exponential $O(\sqrt{n}) = O(2^{\frac{1}{2}\,\mathrm{len}(n)})$, which a is much bigger problem.

Computing discrete logarithms

Example

The group $\mathbb{Z}_{37}^*$ is cyclic of order $n = 36$ with a generator $a = 2$.
Calculate $\mathrm{dlog}_2(7)$ in the group $\mathbb{Z}_{37}^*$.

We put $m = \sqrt{36} = 6$, then $m' = \frac{36}{6} = 6$.
Thus $x = \mathrm{dlog}_2(7) = 6v + u$, where $0 \leq u, v \leq 5$.

$$7 = 2^x = 2^{6v+u}, \text{ hence, } 2^u = 7(2^{-6})^v = 7(2^{30})^v = 7 \cdot 11^v$$

Baby steps: $2^0 = 1$, $2^1 = 2$, $2^2 = 4$, $2^3 = 8$, $2^4 = 16$, $2^5 = 32$
Giant steps: $7 \cdot 11^0 = 7$, $7 \cdot 11^1 = 3$, $7 \cdot 11^2 = 33$, $7 \cdot 11^3 = 30$, $7 \cdot 11^4 = 34$, $7 \cdot 11^5 = 4$ (we are multiplying by 11 succesively)
The equality occured for $2^2 = 4 = 7 \cdot 11^5$, so $u = 2$, $v = 5$ and $\mathrm{dlog}_2(7) = 6 \cdot 5 + 2 = 32$.

Computing discrete logarithms

Remarks

- Pollard's ρ-method for computing discrete logarithms is a probabilistic algorithm operating in expected time of $O(\sqrt{n}) = O(2^{\frac{1}{2}\,\mathrm{len}(n)})$ multiplications in G. An advantage of it is the polynomial space complexity.
  Published by John Pollard in 1978.
- The index calculus is a subexponential algorithm for discrete logarithms that works only for subgroups of $\mathbb{Z}_p^*$.
  Its time complexity is $O(e^{(2\sqrt{2}+o(1))\sqrt{\ln(p)\ln(\ln(p))}})$.
  We will introduce it as the SEDL algorithm later.

Computing discrete logarithms

Pohling-Hellman algorithm

Let G be a cyclic group of order n with a generator a, let $b \in G$. If the factorization of n is known, then the computation of $\mathrm{dlog}_a(b)$ could be sped up.

- If $|G| = n = \prod_i^k q_i^{e_i}$, then we can compute discrete logarithms in subgroups of orders $q_i^{e_i}$ and use the Chinese reminder theorem.

  If $a^x = b$ in the group G, then $(a^{n_i})^x = b^{n_i}$, where $n_i = n/q_i^{e_i}$. We compute $x_i = \mathrm{dlog}_{a^{n_i}}(b^{n_i})$ in the subgroup $H_i = \langle a^{n_i} \rangle$ of order $q_i^{e_i}$ for each $1 \leq i \leq k$. So we obtain a residual system of k equations $x \equiv x_i \pmod{q_i^{e_i}}$ whose solution is $x = \mathrm{dlog}_a(b)$.
- If $|G| = q$ is prime, we use the Baby step/giant step algorithm. The required time is $O(q^{\frac{1}{2}})$ multiplications in G.

Computing discrete logarithms

Pohling-Hellman algorithm

- If $|G| = q^e$ is a power of a prime, then computation of a discrete logarithm can be converted recursively to calculations of e discrete logarithms in the subgroup of order q. The time required for it is $O(eq^{\frac{1}{2}} + e\,\mathrm{len}(q))$ multiplications in G.

  If $a^x = b$ in the group G, then $x = x_{e-1}q^{e-1} + \cdots + x_1 q + x_0 < q^e$. We will compute the digits $0 \leq x_i < q$ successively from x_0 till x_{e-1} as discrete logarithms in the subgroup $H = \langle a^{(q^{e-1})} \rangle$ of order q. Powering the equation $a^x = b$ to q^{e-1} and using $\mathrm{r}(a) = q^e$ we get $(a^{(q^{e-1})})^{x_0} = b^{(q^{e-1})}$ and we calculate x_0. Then we power the equation $a^x = b$ to q^{e-2} and we calculate x_1. Continuing to x_{e-1} we get the q-ary expansion for x.

  Each call involves counting two powers with exponent less than q^e. A recursion is called $e-$times.

Computing discrete logarithms

Pohling-Hellman algorithm

- If q is the largest prime in the factorization $n = |G|$, the time required to compute a discrete logarithm in the group G is determined by computation in the subgroup of order q.
  The time complexity is roughly $O(q^{\frac{1}{2}})$ multiplications in G.
- The algorithm was published by Stephen Pohling and Martin Hellman in 1978 as a Pohling-Hellman attack on the Diffie-Hellman key establishment. If large $n = |G|$ is a product of small primes, then the protocol is not secure.

Computing discrete logarithms

Example

The group $G = \mathbb{Z}_{37}^*$ is cyclic of order $n = 36$ with a generator $a = 2$. Calculate $x = \mathrm{dlog}_2(7)$ in the group $\mathbb{Z}_{37}^*$.

$n = 36 = 2^2 \cdot 3^3 = 4 \cdot 9$.

Let $x \in \mathbb{Z}_{36}$ have residues $\theta(x) = (x', x'') \in \mathbb{Z}_4 \times \mathbb{Z}_9$.

The equality $7 = 2^x$ in the group G implies equalities:

- $7^9 = (2^9)^x = (2^9)^{x'}$ in the subgroup H' of order 4 with a generator 2^9, where $2^9 = 31$, $7^9 = 1$, so $x' = \mathrm{dlog}_{31}(1) = 0$
- $7^4 = (2^4)^x = (2^4)^{x''}$ in the subgroup H'' of order 9 with a generator 2^4, where $2^4 = 16$, $7^4 = 33$, so $x'' = \mathrm{dlog}_{16}(33) = 5$ (we use the baby step/giant step algorithm for H'').

From the Chinese reminder theorem, $x = \theta^{-1}(0, 5) = 32$.

Computing discrete logarithms

Example

$x'' = \mathrm{dlog}_{16}(33) = 5$ in the subgroup H'' of order $9 = 3^2$ with a generator 16 can be calculated recursively:
Let's denote $x'' = 3x_1 + x_0$, where $0 \leq x_0, x_1 < 3$.

- The third power of the equation $33 = 16^{x''} = 16^{3x_1+x_0}$ gives $33^3 = 16^{9x_1+3x_0} = 16^{3x_0}$ (we count modulo 9 in exponent). Now $16^3 = 26$, $33^3 = 10$, so $x_0 = \mathrm{dlog}_{26}(10) = 2$.
- We put x_0 into the original equation and get $33 = 16^{3x_1+2}$. Hence $16^{3x_1} = 33 \cdot 16^{-2} = 26$, so $x_1 = \mathrm{dlog}_{26}(26) = 1$.

We have found $x'' = 3 \cdot 1 + 2 = 5$.

Finding a generator for $\mathbb{Z}_p^*$

An element a is a generator of a cyclic group G of order n in case $a^d \neq 1$ for each proper divisor d of n. Moreover, it is sufficient to check only the maximal proper divisors of n.

So to check if a is a generator of G, we need to know the factorization of $n = |G|$. Otherwise, we cannot do it.

In the following section we work with groups $\mathbb{Z}_p^*$, even if all the algorithms would work analogously for any cyclic group.

The time complexity is computed due to multiplication in $\mathbb{Z}_p$, where multiplying two numbers is estimated with time $O(\mathrm{len}(p)^2)$.

Finding a generator for $\mathbb{Z}_p^*$

Algorithm 1

Let p be a prime and $p-1 = \prod_{i=1}^{k} q_i^{e_i}$ be factorization of $|\mathbb{Z}_p^*|$.
All calculations are done in $\mathbb{Z}_p$.

- repeat
  - select $a \in \mathbb{Z}_p^*$ at random
  - $GEN \leftarrow true$, $i \leftarrow 1$ [the element a can be a generator]
  - repeat
    - if $a^{\frac{p-1}{q_i}} = 1$ then $GEN \leftarrow false$ endif
    - $i \leftarrow i+1$
  - until *not* GEN or $i > k$
- until GEN
- output a

Finding a generator for $\mathbb{Z}_p^*$

Correctness of the algorithm 1

- The algorithm 1 stops because $\mathbb{Z}_p^*$ is a cyclic group.
- The element a in the output is a generator of $\mathbb{Z}_p^*$ due to the following statement.

Claim

An element a is a generator of a cyclic group G of order n if and only if $a^{\frac{n}{p}} \neq 1$ for each prime p, where $p \mid n$.

Finding a generator for $\mathbb{Z}_p^*$

Time complexity of the algorithm 1

- A cyclic group of order n has a $\varphi(n)$ possibilities how to choose a generator. We compute the probability that a randomly chosen element from $\mathbb{Z}_p^*$ is a generator:

$$P[a \text{ is gen}] = \frac{\varphi(p-1)}{p-1} = \prod_{i=1}^{k} \frac{q_i - 1}{q_i} \geq \prod_{i=2}^{k+1} \frac{i-1}{i} = \frac{1}{k+1}$$

- The average number of random selections will therefore be at most $k+1$. (Later on, we will derive this more precisely, introducing a random variable $L =$ the number of repeat-until cycles, and the expectation value of this random variable will be $E(L) \leq k+1$.)

Finding a generator for $\mathbb{Z}_p^*$

Time complexity of the algorithm 1

- In each cycle, we compute at most k powers with exponent less than p using the repeated squaring algorithm, so we need time $O(k \operatorname{len}(p)^3)$.
- The total expected time is $O(k^2 \operatorname{len}(p)^3)$, where k is the number of distinct primes in the factorization of $p-1$. The expected time is also $O(\operatorname{len}(p)^5)$, since $k < \operatorname{len}(p)$.

Finding a generator for $\mathbb{Z}_p^*$

Algorithm 2

Let p be a prime and $p-1 = \prod_{i=1}^{k} q_i^{e_i}$ be a factorization of $|\mathbb{Z}_p^*|$. All calculations are done in $\mathbb{Z}_p$.

- for $i \leftarrow 1$ to k do
  - repeat
    - select $b \in \mathbb{Z}_p^*$ at random
    - $b_i \leftarrow b^{\frac{p-1}{q_i}}$
  - until $b_i \neq 1$
  - $a_i \leftarrow b^{\frac{p-1}{(q_i^{e_i})}}$
- $a \leftarrow \prod_{i=1}^{k} a_i$
- output a

Finding a generator for $\mathbb{Z}_p^*$

Correctness of the algorithm 2

- The algorithm 2 stops because $\mathbb{Z}_p^*$ is a cyclic group, so it contains elements of all orders which divide the group order.
- The element a_i has order $q_i^{e_i}$ (see the following statement), and due to the relatively primeness of orders of different a_i's, the output element a has order $\prod_{i=1}^{k} q_i^{e_i} = p - 1$, so it is a generator of $\mathbb{Z}_p^*$.

Claim

Let q be a prime and $e \geq 1$ a natural number.
Let an element c of an abelian group satisfies $c^{(q^e)} = 1$ and $c^{(q^{e-1})} \neq 1$, then the order of the element c is q^e.

Finding a generator for $\mathbb{Z}_p^*$

Time complexity of the algorithm 2

- The element $b_i = b^{\frac{p-1}{q_i}} \neq 1$ has order q_i.
  The probability that for a randomly chosen element $b \in \mathbb{Z}_p^*$ its $\frac{p-1}{q_i}$'th power differs from 1 is

$$P[b_i \neq 1] = \frac{q_i - 1}{q_i} \geq \frac{1}{2}.$$

- On average, each of the k repeat-until cycles is repeated twice, calculating always one power by the repeated squaring algorithm.
- The total expected time is $O(2k\,\mathrm{len}(p)^3)$, where k is the number of distinct primes in the factorization of $p-1$.
  The expected time is also $O(\mathrm{len}(p)^4)$, since $k < \mathrm{len}(p)$.

Finding an element of order q in $\mathbb{Z}_p^*$

Algorithm 3

Let p be a prime and q be a prime such that $q \mid p-1$.
All calculations are performed in $\mathbb{Z}_p$.

- repeat
  - choose randomly $b \in \mathbb{Z}_p^*$
  - $c \leftarrow b^{\frac{p-1}{q}}$
- until $c \neq 1$
- output c

Correctness and time complexity

The algorithm is correct, the expected running time is $O(\mathrm{len}(p)^3)$.
Every element of order q can be found with equal probability.

Finding an element of order q in $\mathbb{Z}_p^*$

Example

For $p = 317$, $p - 1 = 2^2 \cdot 79$.
We are looking for an element of order 79 in $\mathbb{Z}_{317}^*$.
We choose $b = 2$. Since $2^4 = 16 \neq 1$, $c = 16$ has order 79.

We have a subgroup $G = \langle 16 \rangle$ of order 79 in the group $\mathbb{Z}_{317}^*$ and we can use it for ElGamal encryption.

Group informations in the "phone book": $(p, n, a) = (317, 79, 16)$

- multiply modulo $p = 317$, messages $0 < m < 317$
- in the exponent count modulo $n = 79$, jepices keys $0 < y < 79$
- a group generator is the element $a = 16$

Discrete logarithm

Literature

- Shoup: A Computational Introduction to Number Theory and Algebra. Chapter 11.
  http://shoup.net/ntb/