

Figure: The inverse isomorphism  $d\exp$  and  $d\log$ .

We take  $G = \mathbb{Z}_d$  with addition (modulo  $d$ ) as group operation.

“ $x = g^a$ ” now means  $x = a \cdot g$  in  $\mathbb{Z}_d$ , and for  $g \in \mathbb{Z}$  we have by the Extended Euclidean Algorithm:

$$g \text{ generates } \mathbb{Z}_d \iff \gcd(g, d) = 1 \iff \exists s, t \in \mathbb{Z} \quad sg + td = 1.$$

We claim that for any integer  $x$ , the value  $a = xs$  in  $\mathbb{Z}_d$  is the discrete logarithm  $a$  of  $x$  in basis  $g$ . Indeed,

$x = x(sg + td) = xsg + xtd = a \cdot g$  in  $\mathbb{Z}_d$ . This computation can be done in polynomial time, and therefore  $\mathbb{Z}_d$  is an *easy group* and useless for group cryptography.

1. *baby-step giant-step attack*: works for any  $G$  and takes time and space  $\sqrt{d}$ .
2. *Pollard rho*: works for any  $G$  and takes time  $\sqrt{d}$  and constant space.
3. *Chinese distribution*: reduces from general  $d$  to prime power  $d$ .
4. *Pohlig–Hellman algorithm*: reduces from prime power  $d$  to prime  $d$ .
5. *index calculus*: works for  $G = \mathbb{Z}_p^\times$ ,  $p$  a prime (and can be generalized slightly, say to  $\mathbb{F}_q^\times$  with  $q$  a prime power).

## CRYPTOSYSTEM 1. ElGamal encryption scheme.

General setup:

Input:  $1^n$ .

1. Choose a finite cyclic group  $G = \langle g \rangle$  with  $d$  elements, where  $d$  is an  $n$ -bit integer, and a generating element  $g$ . These data, including a description of  $G$ , are made public.

Key generation:

Output:  $(sk, pk)$ .

2. Choose a secret key  $sk = b \xleftarrow{\$} \mathbb{Z}_d$  at random. The public key is  $pk = B = g^b \in G$ , and  $sk = b$  is the secret key.

## CRYPTOSYSTEM 2. ElGamal encryption scheme.

Encryption:

Input: Plaintext  $x \in G$ .

Output: Ciphertext  $\text{enc}_{\text{pk}}(x) \in G^2$ .

1. Choose a secret session key  $a \xleftarrow{\$} \mathbb{Z}_d$  at random.
2. Public session key  $A \xleftarrow{\$} g^a \in G$ , and common session key  $k \leftarrow B^a = g^{ab} = A^b$ .
3.  $y \leftarrow x \cdot k \in G$ , Return  $\text{enc}_{\text{pk}}(x) = (y, A)$ .

Decryption:

Input: Arbitrary  $(y, A) \in G^2$ .

Output: Decryption  $\text{dec}_{\text{sk}}(y, A) \in G$ .

4. Common session key  $k \leftarrow A^b$ , inverse  $k^{-1} \in G$  of the common key.
5.  $z \leftarrow y \cdot k^{-1}$ , Return  $\text{dec}_{\text{sk}}(y, A) = z$ .

Let  $G = \mathbb{Z}_{2579}^\times$  and  $g = 2$  again. Bob has published his public key  $B = g^b = 949$ . Alice wants to encrypt the plaintext MY SECRET and writes it in the familiar numerical notation (12, 24, 18, 4, 2, 17, 4, 19), where A corresponds to 0, etc. Spaces are often ignored in cryptography. Alice sends two letters  $b_1, b_2$  as one unit by computing  $26b_1 + b_2$ . She uses different secret session keys to avoid statistical attacks, and sends  $(y, A)$  as her message.

plaintext		encryption				decryption		
letters	$x$	$a$	$A = g^a$	$k = B^a$	$y = x \cdot k$	$k = A^b$	$k^{-1}$	$k^{-1}y$
MY	336	2111	1617	2430	1516	2430	1402	336
SE	472	367	734	2474	2020	2474	2186	472
CR	69	351	832	203	1112	203	559	69
ET	123	161	1355	1483	1879	1483	873	123

## Theorem

For any group  $G$ , breaking the ElGamal cryptosystem and solving the Diffie–Hellman Problem can be reduced to each other in polynomial time. The reductions can be done using at most  $O(n^2)$  bit operations, where  $n$  is the bit size of  $G$ .

ALGORITHM 3. Birthday algorithm for discrete logarithm.

Input: A cyclic group  $G = \langle g \rangle$  with  $d$  elements, and a group element  $x \in G$ .

Output:  $\text{dlog}_g x$ .

1.  $X, Y \leftarrow \emptyset$ .
2. Do step 3 until a collision of  $X$  and  $Y$  occurs.
3. Choose uniformly at random a bit  $b \xleftarrow{\$} \{0, 1\}$  and  $i \xleftarrow{\$} \{0, \dots, d-1\}$ . Add  $xg^i$  to  $X$  if  $b = 0$  and  $g^i$  to  $Y$  if  $b = 1$ , and remember the index  $i$ .
4. If  $xg^i = g^j$  for some  $xg^i \in X$  and  $g^j \in Y$ , then return  $j - i$  in  $\mathbb{Z}_d$ .

## Theorem

The algorithm works correctly as specified. Its expected time is  $O(\sqrt{d} \log d)$  multiplications in  $G$ , with expected space for  $O(\sqrt{d})$  elements of  $G$ .

## Theorem

Let  $G$  be a cyclic group of order  $d$ . Then the Pollard's rho algorithm, with Floyd's trick, finds a discrete logarithm in  $G$  with an expected number of  $O(\sqrt{d})$  group operations, provided that the sequence  $x_0, x_1, x_2, \dots$  behaves randomly. Storage is required for two elements of  $G$  and four elements of  $\mathbb{Z}_d$ .

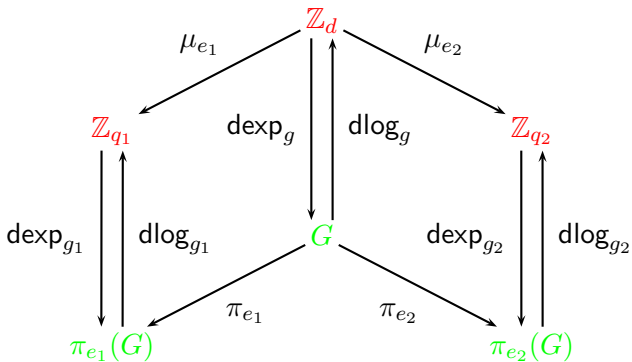


Figure: Chinese remainder computation for  $DL_G$  with  $d = q_1q_2$ .

We have  $G = \mathbb{Z}_{25}^\times = \langle 2 \rangle$  with  $d = \#G = 20 = 4 \cdot 5$ , so that  $q_1 = 4$  and  $q_2 = 5$ , and  $x = 17 \in G$ . Then  $g_1 = 2^{20/4} = 7$  and  $g_2 = 2^{20/5} = 16$ , and the two subgroups

$$S_1 = \langle 2^{20/4} \rangle = \{1, 7, 24, 18\} \text{ and } S_2 = \langle 2^{20/5} \rangle = \{1, 16, 6, 21, 11\}$$

have 4 and 5 elements, respectively. The Chinese remainder algorithm for the discrete logarithm of 17 first computes the two constituents of  $x$  in  $S_1$  and  $S_2$ :  $x_1 = 17^{20/4} = 7$  and  $x_2 = 17^{20/5} = 21$ . We can read off the discrete logarithms in  $S_1$  and  $S_2$ :  $a_1 = \text{dlog}_{g_1} x_1 = 1$  and  $a_2 = \text{dlog}_{g_2} x_2 = 3$ . With the Chinese Remainder Algorithm, we find  $a = 13$ , which satisfies  $a = 1$  in  $\mathbb{Z}_4$  and  $a = 3$  in  $\mathbb{Z}_5$ .

## Lemma

Let  $d = q_1 \cdots q_r$  be a factorization of  $d = \#G$  into pairwise coprime factors, with  $G = \langle g \rangle$  a cyclic group as above, let  $x \in G$  and for  $1 \leq i \leq r$ , let  $S_i = \{x^{d/q_i} : x \in G\}$  and  $T_i = \{x \in G : x^{q_i} = 1\}$ . Then the following hold.

1.  $S_i = T_i$  is a subgroup with  $q_i$  elements, generated by  $g^{d/q_i}$ , and the map

$$\begin{aligned} G &\rightarrow S_1 \times S_2 \times \cdots \times S_r, \\ y &\mapsto (y^{d/q_1}, \dots, y^{d/q_r}). \end{aligned}$$

is an isomorphism.

2. If  $x = g^a$ ,  $1 \leq i \leq r$ , and  $a = a_i$  in  $\mathbb{Z}_{q_i}$ , then  $x^{d/q_i} = (g^{d/q_i})^{a_i}$ .
3. If  $a_i = \text{dlog}_{g^{d/q_i}} x^{d/q_i}$  and  $a = a_i$  in  $\mathbb{Z}_{q_i}$  for all  $1 \leq i \leq r$ , then  $a = \text{dlog}_g x$ .

ALGORITHM 4. Chinese remaindering for discrete logarithms.

Input: A cyclic group  $G = \langle g \rangle$  of order  $d = \#G$ , and  $x \in G$ .

Output:  $a = \text{dlog}_g x$ .

1. Compute the prime power factorization of  $d$ .
2. For each  $i \leq r$ , do steps 2 and 3.
3. Compute  $g_i = g^{d/q_i}$  and  $x_i = x^{d/q_i}$ , with repeated squaring.
4. Compute the discrete logarithm  $a_i = \text{dlog}_{g_i} x_i$  in  $S_i = \langle g_i \rangle$ .
5. Combine these “small” discrete logarithms via Chinese remaindering to find the unique  $a \in \mathbb{Z}_d$  so that  $a = a_i$  in  $\mathbb{Z}_{q_i}$  for all  $i \leq r$ .

## Theorem

Let  $G$  be a cyclic group of  $n$ -bit order  $d$ . Then *Chinese remaindering for discrete logarithms* computes discrete logarithms in  $G$  at the following cost:

1. factoring the integer  $d$ ,
2. one discrete logarithm in each of the groups  $S_1, \dots, S_r$ ,
3.  $O(n^2)$  operations in  $G$ ,
4.  $O(n^2)$  bit operations.

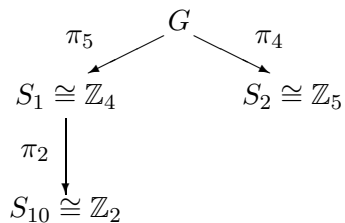


Figure: Discrete logarithm in a group with 20 elements

We illustrate the Pohlig–Hellman Algorithm in an example with  $p^e = 3^4 = 81$ . The group  $G$  is the subgroup  $G = \langle 4 \rangle \subseteq \mathbb{Z}_{163}^\times$  generated by  $g = 4$ . We note that 163 is prime and  $\#G_{163}^\times = \phi(163) = 162 = 2 \cdot 81$ . Furthermore,  $2^2 = 4$  and  $2^{81} = -1$  in  $\mathbb{Z}_{163}$ . Thus 2 is a generator of  $\mathbb{Z}_{163}^\times$ , so that the order of 4 in  $\mathbb{Z}_{163}^\times$  is 81.

We have  $p = 3$ ,  $e = 4$ , and  $H = \langle 4^{27} \rangle = \langle h \rangle = \{1, 104, 58\}$  with  $h = 104$ . We trace the computation of the discrete logarithm  $a = \text{dlog}_4 60 = 2 \cdot 3^3 + 0 \cdot 3^2 + 1 \cdot 3 + 2 = 59$  of  $x = 60 = 4^{59}$ . Discrete logarithms in  $H$  are found by inspection.

$$\begin{aligned} x_0 = x^{p^3} = x^{27} = 58 \in H, a_0 = \text{dlog}_h x_0 = 2, y_0 = y_{-1} \cdot g^{-a_0} = 51, \\ x_1 = (xy_0)^9 = 104 \in H, a_1 = \text{dlog}_h x_1 = 1, y_1 = y_0 \cdot g^{-a_1 \cdot 3} = 39, \\ x_2 = (xy_1)^3 = 1 \in H, a_2 = \text{dlog}_h x_2 = 0, y_2 = y_1 \cdot g^{-a_2 \cdot 9} = 39, \\ x_3 = xy_2 = 58 \in H, a_3 = \text{dlog}_h x_3 = 2. \end{aligned}$$

We now have computed

$$a = a_3 p^3 + a_2 p^2 + a_1 p + a_0 = 2 \cdot 27 + 0 \cdot 9 + 1 \cdot 3 + 2 \cdot 1 = 59. \text{ As a check, we find } 4^{59} = 60 \text{ in } \mathbb{Z}_{163}^\times.$$

ALGORITHM 5. Discrete logarithm in a cyclic group.

Input: A finite cyclic group  $G = \langle g \rangle$  of order  $d$ , and  $x \in G$ .

Output:  $a = \text{dlog}_g x$ .

1. Factor  $d = p_1^{e_1} \cdots p_r^{e_r}$  into powers of distinct primes, with positive integer exponents  $e_1, \dots, e_r$ .
2. For  $1 \leq i \leq r$  do the following.
3. Compute the image  $x^{d/q_i}$  of  $x$  in  $\langle g^{d/q_i} \rangle$ , where  $q_i = p_i^{e_i}$ .
4. Compute the discrete logarithm  $a_i$  of  $x^{d/q_i}$  in base  $g^{d/q_i}$ , using the Pohlig–Hellman. For the discrete logarithm subroutine in  $H = \langle g^{d/p_i} \rangle$ , with  $\#H = p_i$ , use the Pollard rho.
5. Combine the individual logarithms  $a_i$  into  $a$  by Chinese remaindering.

## Theorem

Let  $G$  be a cyclic group with  $d$  elements,  $p$  the largest prime factor of  $d$ , and  $n$  the bit length of  $d$ . We assume that the sequences in the Pollard rho algorithm behave sufficiently randomly. Then discrete logarithms in  $G$  can be randomly computed at the following cost:

- ▶ factoring  $d$ ,
- ▶ an expected number of  $O(n\sqrt{p} + n^2)$  operations in  $G$ ,
- ▶  $O(n^2)$  bit operations.

## Index calculus

We have a generator  $g$  for the multiplicative group  $\mathbb{Z}_p^\times = \langle g \rangle$  of units modulo  $p$ , of order  $d = p - 1$ , and want to compute  $\text{dlog}_g x$  for some given  $x \in \mathbb{Z}_p^\times$ . As in the random squares method, we choose a *factor base*  $\{p_1, \dots, p_h\}$  consisting of the primes up to some bound  $B = p_h$ .

In a preprocessing step, which does not depend on  $x$ , we choose random exponents  $e$  and check if  $g^e$  in  $\mathbb{Z}_p = \{0, \dots, p - 1\}$  is  $B$ -smooth. If it is, we find integers  $\alpha_1, \dots, \alpha_h$  with

$$\begin{aligned} g^e &= p_1^{\alpha_1} \cdots p_h^{\alpha_h} \text{ in } \mathbb{Z}_p, \\ e &= \alpha_1 \text{dlog}_g p_1 + \cdots + \alpha_h \text{dlog}_g p_h \text{ in } \mathbb{Z}_{p-1}. \end{aligned} \tag{1}$$

We collect enough of these data until we can solve for the  $\text{dlog}_g p_i$ . Typically, a little more than  $h$  relations (1) will be enough, say  $h + 10$ .

## Index calculus

To solve the system of linear equations, we may factor  $p - 1$ , solve modulo each prime power factor of  $p - 1$ , and piece the solutions together via the Chinese Remainder Theorem.

At this point, we know  $\text{dlog}_g p_1, \dots, \text{dlog}_g p_h$ . Now on input  $x$ , we choose random exponents  $e$  until some  $xg^e$  in  $\mathbb{Z}_p$  is  $B$ -smooth, say  $xg^e = p_1^{\beta_1} \cdots p_h^{\beta_h}$  in  $\mathbb{Z}_p$ . Then

$$\text{dlog}_g x = -e + \beta_1 \text{dlog}_g p_1 + \cdots + \beta_h \text{dlog}_g p_h \text{ in } \mathbb{Z}_{p-1}.$$

Choose an elliptic curve  $G$  with a large prime factor  $p$  of its order, and then work in the subgroup  $H$  with  $p$  elements as above for all group-based protocols. Ideally, we can take  $H = G$ .

**Figure:** Basic recommendation for group-based cryptography