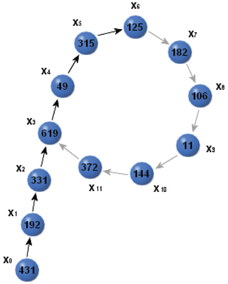
Discrete Logarithm Problem

Çetin Kaya Koç

koc@cs.ucsb.edu



Exponentiation and Logarithms in a General Group

- In a multiplicative group $(S, \otimes)$ with a primitive element $g \in S$, the exponentiation operation for a positive x is the computation of y in

$$y = g^x = \overbrace{g \otimes g \otimes \cdots \otimes g}^{x \text{ terms}}$$

- On the other hand, in an additive group $(S, \oplus)$ with a primitive element $g \in S$, the point multiplication operation is the computation of y in

$$y = [x]g = \overbrace{g \oplus g \oplus \cdots \oplus g}^{x \text{ terms}}$$

- In both cases, the discrete logarithm problem (DLP) is defined as:

$$\boxed{\text{Given } y \text{ and } g\text{, Compute } x}$$

Discrete Logarithms in Public-Key Cryptography

- If the DLP is difficult in a given group, we can use it to implement several public-key cryptographic algorithms, for example, Diffie-Hellman key exchange method, ElGamal public-key encryption method, and the Digital Signature Algorithm
- Two types of groups are noteworthy:
  - The multiplicative group $\mathcal{Z}_p^*$ of integers modulo a prime p
  - The additive group of elliptic curves defined over finite fields
- The DLP problem in these groups are known to be difficult

Discrete Logarithm in $(Z_n, + \bmod n)$

- There may also be other groups worth considering
- However, the DLP is trivial in many groups
- For example, the DLP in additive mod p group is trivial
- "Exponentiation" in this group is defined as

$$y = [x]g = \overbrace{g + g + \cdots + g}^{x \text{ terms}}$$

where g is a primitive element in the group, x is an integer, y is an element of the group (an integer in Z_n), and the $+$ operation is the addition mod n

Discrete Logarithm in $(Z_n, + \bmod n)$

- x is easily solvable from the above since

$$x = g^{-1} \cdot y \pmod{n}$$

where y^{-1} is the multiplicative inverse of y mod n

- Consider $(Z_{11}, + \bmod 11)$ where any nonzero element is primitive
- Any DLP in $(Z_{11}, + \bmod 11)$ is easily solvable, for example,

$$2 = [x]3 \pmod{11}$$

is solved as

$$\begin{aligned} x &= 3^{-1} \cdot 2 \pmod{11} \\ &= 4 \cdot 2 \pmod{11} \\ &= 8 \end{aligned}$$

Discrete Logarithms in GF(2^k)

- On the other hand, the DLP in the multiplicative group of GF(2^k) is also known to be rather easy (but not trivial)
- The multiplicative group of GF(2^k) consists of
  - The set $S = \text{GF}(2^k) - \{0\}$
  - The group operation multiplication mod $p(x)$
  - $p(x)$ is the irreducible polynomial generating the field GF(2^k)
  - The group order is $2^k - 1$
  - The group order is prime, when $2^k - 1$ is prime,

Discrete Logarithms in GF(2^k)

- Consider the multiplicative group of GF(2^3)
- The set is $S = \{1, x, x+1, x^2, x^2+1, x^2+x, x^2+x+1\}$
- The operation is the multiplication mod $p(x) = x^3 + x + 1$
- The group order is 7, which happens to be prime
- Thus, all elements of the set is primitive, except 1
- Let us take $g = x$
- The powers x^i for $i = 1, 2, \ldots, 7$ generates all elements of the set S

$$\{x^i \pmod{p(x)} \mid i = 1, 2, \ldots, 7\} =$$

$$\{x, x^2, x+1, x^2+x, x^2+x+1, x^2+1, 1\}$$

Discrete Logarithms in GF(2^k)

- Consider the DLP in GF(2^3)

$$x^a = x^2 + x \pmod{x^3 + x + 1}$$

  where a is the unknown, to be computed (the DL)
- Which power of x is equal to $x^2 + x \pmod{x^3 + x + 1}$?
- We can solve this particular DLP using exhaustive search
- There are 7 candidates for a, and we find it as $a = 4$

- The general DLP seems difficult
- Don Coppersmith proved that it is easy (but not trivial):
  http://cs.ucsb.edu/~koc/ecc/docx/Coppersmith84.pdf

Exponentiation and Discrete Logarithms in $\mathcal{Z}_p^*$

- Consider the multiplicative group $\mathcal{Z}_p^*$ of integers modulo a prime p and a primitive element $g \in \mathcal{Z}_p^*$
- The exponentiation operation is the computation of y in

$$y = g^x = \overbrace{g \cdot g \cdots g}^{x \text{ terms}} \pmod{p}$$

  for a positive integer x
- The discrete logarithm problem in this group is defined to be the computation of x, given y, g, and p
- Example: Given $p = 23$ and $g = 5$, find x such that

$$11 = 5^x \pmod{23}$$

  Answer: $x = 9$

Discrete Logarithms in $\mathcal{Z}_p^*$

- Given $p = 158(2^{800} + 25) + 1 =$

  1053546280395016975304616582933958731948871814925913489342
  6087342587178835751858673003862877377055779373829258737624
  5199045043066135085968269741025626827114728303489756321430
  0237166369174066615907176472549470083113107138189921280884
  003892629359

  and $g = 3$, find $x \in \mathcal{Z}_p^*$ such that

  $$2 = 3^x \pmod{p}$$

  Answer: ?

Discrete Logarithm Notation

- The computation of x in $y = g^x \pmod{p}$ is called the DLP
- Here x is equal to the discrete analogue of the logarithm

$$x = \log_g y \pmod{p-1}$$

- The modulus is $p-1$ since the powers are added and multiplied mod $p-1$ according to Fermat's Theorem
- The logarithm notation is particularly useful
- For example, $2^{15} = 27 \pmod{29}$ implies

$$\log_2 27 = 15 \pmod{28}$$

Discrete Logarithm Notation

- The logarithm notation allows us to compute new discrete logarithms
- For example

$$\log_g(a \cdot b \bmod p) = \log_g a + \log_g b \pmod{p-1}$$

- Similarly

$$\log_g(a^e \cdot b^f \bmod p) = e \cdot \log_g a + f \cdot \log_g b \pmod{p-1}$$

Discrete Logarithm Notation

- For example,

$$\begin{aligned} 2^{15} &= 27 \pmod{29} \\ 2^{22} &= 5 \pmod{29} \end{aligned}$$

which implies

$$\begin{aligned} \log_2 27 &= 15 \pmod{28} \\ \log_2 5 &= 22 \pmod{28} \end{aligned}$$

- Therefore, we can write

$$\begin{aligned} \log_2 27 + \log_2 5 &= \log_2(27 \cdot 5 \bmod 29) \pmod{28} \\ &= \log_2 19 \pmod{28} \\ 15 + 22 &= 9 \pmod{28} \end{aligned}$$

which implies $\log_2 19 = 9 \pmod{28}$ or $2^9 = 19 \pmod{29}$

Exhaustive Search

- Since $x \in \mathcal{Z}_p^*$, we can perform search, and try all possible values of x:

$$
\begin{array}{l}
\texttt{for } i = 1 \texttt{ to } p-1 \\
\quad z = g^i \pmod{p} \\
\quad \texttt{if } y = z \\
\qquad\quad \texttt{return } x = i
\end{array}
$$

- This algorithm requires the computation of ith power of g mod p at each step

Exhaustive Search

- However, ith power of g need not be computed from scratch

$$
\begin{array}{l}
z = g \\
\texttt{for } i = 2 \texttt{ to } p-1 \\
\quad z = g \cdot z \pmod{p} \\
\quad \texttt{if } y = z \\
\qquad\quad \texttt{return } x = i
\end{array}
$$

- This algorithm requires $p - 2$ multiplications
- Since multiplications of k-bit operands are of order $O(k^2)$, the search is exponential in k

$$O(pk^2) = O(2^k k^2)$$

Shanks' Baby-Step-Giant-Step

- In 1973, Shanks described an algorithm for computing discrete logarithms that runs in $O(\sqrt{p})$ time and requires $O(\sqrt{p})$ space
- Let $y = g^x \pmod{p}$, with $m = \lceil \sqrt{p} \rceil$ and $p < 2^k$
- Shanks' method is a deterministic algorithm and requires the construction of two tables S and T, which contains pairs of integers
- The construction of S is called the giant-steps:

$$S = \{(i, g^{i \cdot m}) \mid i = 0, 1, \ldots, m\}$$

- The construction of T is called the baby-steps:

$$T = \{(j, y \cdot g^j) \mid j = 0, 1, \ldots, m\}$$

Shanks' Baby-Step-Giant-Step

- The existence of the same group element in both tables implies

$$g^{i \cdot m} = y \cdot g^j = g^x \cdot g^j \pmod{p}$$

- We get the indices i and j, and write the equality of the powers as

$$i \cdot m = x + j \pmod{p-1}$$

  and thus find $x = i \cdot m - j \pmod{p-1}$

- To use this method in practice, one would typically only store the giant-steps array and the lookup each successive group element from the baby-steps array until a match is found
- However, the algorithm requires enormous amount of space, and thus, it is rarely used in practice

Shanks' Baby-Step-Giant-Step

- Consider the solution of $y = 44 = 3^x \pmod{101}$
- $m = \lceil \sqrt{101} \rceil = 11$, therefore, the giant-steps and baby-steps tables:

$$S = \{(i, 3^{11i}) \mid i = 0, 1, \ldots, 11\}$$

i	0	1	2	3	4	5	6	7	8	9	10	11
3^{11i}	1	94	49	61	**78**	60	85	11	24	34	65	50

$$T = \{(j, 44 \cdot 3^j) \mid j = 0, 1, \ldots, 11\}$$

j	0	1	2	3	4	5	6	7	8	9	10	11
$44 \cdot 3^j$	44	31	93	77	29	87	59	76	26	**78**	32	96

- The solution $x = 4 \cdot 11 - 9 = 35$, i.e., $3^{35} = 44 \pmod{101}$

Correctness of Shanks' Algorithm

- Solving for x in $y = g^x \pmod{p}$ requires creation of 2 tables of $O(\sqrt{p})$ size
- However, x can be any one of the numbers in the set $[2, p-2]$, which is of size $O(p)$
- How does it work that by searching in 2 tables of size $O(\sqrt{p})$ we can find an element x that belongs to a set of size $O(p)$?

Proof of Correctness of Shanks' Algorithm

- Since $m = \lceil \sqrt{p} \rceil$, we can write x in base-m as

$$x = i \cdot m + j$$

  such that $i, j \in [0, m-1]$
- For example, for $p = 101$, $m = 11$, and $x = 35$, we can write:

$$35 = 3 \cdot 11 + 2$$

- Instead of searching for $x \in [2, p-2]$, we can search for $i, j \in [0, m-1]$

Proof of Correctness of Shanks' Algorithm

- Therefore, we would be performing 2 searches in two sets of size $O(m) = O(\sqrt{p})$, one search for i and the other for j
- The exponentiation equality is given as

$$y = g^{i \cdot m + j} \pmod{p}$$

- This implies

$$y \cdot g^{-j} = g^{i \cdot m} \pmod{p}$$

Proof of Correctness of Shanks' Algorithm

- We would create one table (S) of values $(i, g^{i \cdot m})$, and another table (T) of values $(j, y \cdot g^{-j})$
- An equality of the form

$$g^{i \cdot m} = y \cdot g^{-j} = g^{x} \cdot g^{-j} \pmod{p}$$

for particular values of i, j implies that

$$i \cdot m = x - j \pmod{p-1}$$

which allows us to compute x by creating 2 tables of size $O(\sqrt{p})$

Pollard Rho Algorithm for DLP

- Pollard Rho algorithm is also of $O(\sqrt{p})$ time complexity, however, it does not require a large table
- It forms a pseudorandom sequence of elements from the group, and searches for a cycle to appear in the sequence
- The sequence is defined deterministically and each successive element is a function of only the previous element
- If a group element appears a second time, every element of the sequence after that will be a repeat of elements in the sequence
- According to the birthday problem, a cycle should appear after $O(\sqrt{p})$ elements of the sequence have been computed

Pollard Rho Algorithm for DLP

- The Pollard Rho algorithm defines the sequence

$$a_{i+1} = \begin{cases} y \cdot a_i & \text{for} \quad a_i \in S_0 \\ a_i^2 & \text{for} \quad a_i \in S_1 \\ g \cdot a_i & \text{for} \quad a_i \in S_2 \end{cases}$$

  where S_0, S_1, and S_2 are disjoint partitions of the group elements, that are approximately the same size
- The initial term is taken as $a_0 = g^\alpha$ for a random α
- Apparently, there is no need to keep all of the group elements; we compute the sequences from a_i to a_{2i} until an equality is discovered
- The equality of two terms in the sequence implies equality on exponents mod $(p-1)$, from which we solve for x

Pollard Rho Algorithm for DLP

- Consider the solution of $y = 44 = 3^x \pmod{101}$
- We divide the set $S = \{1, 2, \ldots, 100\}$ into 3 sets such that $S_0 = \{1, 2, \ldots, 33\}$, $S_1 = \{34, 35, \ldots, 66\}$, and $S_2 = \{67, 68, \ldots, 100\}$
- Starting with the first term $a_0 = g^\alpha$ for a random $\alpha = 15$, we get $a_0 = 3^{15} = 39$, and first few following terms of the iteration as

	a_i	S_0	S_1	S_2	a_i
$i = 0$	$a_0 = 39$		39		g^{15}
$i = 1$	$a_1 = a_0^2 = 39^2 = 6$	6			g^{30}
$i = 2$	$a_2 = y \cdot a_1 = 44 \cdot 6 = 62$		62		$y \cdot g^{30}$
$i = 3$	$a_3 = a_2^2 = 62^2 = 6$	6			$y^2 \cdot g^{60}$
$i = 4$	$a_4 = y \cdot a_3 = 44 \cdot 6 = 62$		62		$y^3 \cdot g^{60}$
$i = 5$	$a_5 = a_4^2 = 62^2 = 6$	6			$y^6 \cdot g^{120}$

Pollard Rho Algorithm for DLP

- Therefore, we find $a_1 = a_3$ (also $a_2 = a_4$ and $a_3 = a_5$)
- The discovery of an equality in the sequence implies that we found a relationship between the exponent x and known powers of g
- The equality $a_1 = a_3$ implies

$$g^{30} = y^2 \cdot g^{60} = (g^x)^2 \cdot g^{60} = g^{2x+60}$$

- We have an equality over the exponents

$$30 = 2x + 60 \pmod{100} \ \rightarrow \ 2x = 70 \pmod{100}$$

- Since $\gcd(2, 100) \neq 1$, this equation has two solutions: $x = \{35, 85\}$

Pollard Rho Algorithm for DLP

- We can check each candidate to verify:

$$3^x \stackrel{?}{=} 44 \pmod{100}$$

- We see that $x = 35$ is a solution since $3^{35} = 44 \pmod{101}$
- Similarly, the equality of $a_2 = a_4$ gives the same equation:

$$x + 30 = 3x + 60 \pmod{100} \;\rightarrow\; 2x = 70 \pmod{100}$$

- On the other hand, the equality of $a_3 = a_5$ implies

$$2x + 60 = 6x + 120 \pmod{100} \;\rightarrow\; 4x = 40 \pmod{100}$$

We find 4 candidates: $x = \{10, 35, 60, 85\}$

Pollard Rho Algorithm for DLP

- Another way to divide the set $S = \{1, 2, \ldots, p-1\}$:
  $S_0 = \{i \mid i = 0 \pmod 3\}$
  $S_1 = \{i \mid i = 1 \pmod 3\}$
  $S_2 = \{i \mid i = 2 \pmod 3\}$

- For $p = 101$, we get
  $S_0 = \{3, 6, 9, 12, \ldots, 99\}$
  $S_1 = \{1, 4, 7, 10, 13, \ldots, 100\}$
  $S_2 = \{2, 5, 8, 11, 14, \ldots, 98\}$

Pollard Rho Algorithm for DLP

- Solving for $48 = 3^x \pmod{101}$
- Starting with the first term $a_0 = g^\alpha$ for a random $\alpha = 10$, we get $a_0 = 3^{10} = 65$, and first few following terms of the iteration as

	a_i	S_0	S_1	S_2	a_i
$i = 0$	$a_0 = 65$			65	g^{10}
$i = 1$	$a_1 = g \cdot a_0 = 3 \cdot 65 = 94$		94		g^{11}
$i = 2$	$a_2 = a_1^2 = 94^2 = 49$		$\boxed{49}$		g^{22}
$i = 3$	$a_3 = a_2^2 = 49^2 = 78$	78			g^{44}
$i = 4$	$a_4 = y \cdot a_3 = 48 \cdot 78 = 7$		7		$y \cdot g^{44}$
$i = 5$	$a_5 = a_4^2 = 7^2 = 49$		$\boxed{49}$		$y^2 \cdot g^{88}$

Pollard Rho Algorithm for DLP

- The equality of $a_2 = a_5$ implies

$$g^{22} = y^2 \cdot g^{88} = g^{2x} \cdot g^{88} \pmod{101}$$

- From which, we write

$$\begin{aligned} 2x + 88 &= 22 \pmod{100} \\ 2x &= 34 \pmod{100} \end{aligned}$$

- We find candidates for the solution as $\{17, 67\}$
- Trying both, we find $x = 17$ as the solution in $3^x = 48 \pmod{101}$

The Complexity of Pollard Rho Algorithm

- The Pollard Rho algorithm generates a sequence in order to find a match, due to the birthday problem
- Its time complexity is $O(\sqrt{p})$ which is exponential in terms of the input size in bits: $O(2^{k/2})$
- However, there are subexponential algorithms, for example the index calculus method for the group $\mathcal{Z}_p^*$ has subexponential time complexity
- Before that, let us study the Pohlig-Hellman algorithm which converts on order-ab DL into an order-a DL, an order-b DL, and a few exponentiations

Pohlig-Hellman Algorithm

- The group order in $(Z_p^*, * \bmod p)$ is $p-1$
- There are $p-1$ elements in the set $Z_p^* = \{1, 2, \ldots, p-1\}$
- Assume $p-1 = a \cdot b$, that is, g has order $a \cdot b$
- Given y, which is a power of g, we deduce that:
  g^a has order b since $(g^a)^b = g^{ab} = g^{p-1} = 1$
  g^b has order a since $(g^b)^a = g^{ab} = g^{p-1} = 1$
  y^a is a power of g^a since $y^a = (g^x)^a = (g^a)^x$

Pohlig-Hellman Algorithm

- Step 1: Solve for r in the DLP:

$$(g^a)^r = y^a \pmod{p}$$

- Step 2: Solve for s in the DLP:

$$(g^b)^s = y \cdot g^{-r} \pmod{p}$$

- Step 3: Compute $x = r + s \cdot b$
- Correctness proof:

$$\begin{aligned} g^{r+s\cdot b} &= g^r \cdot (g^s)^b \pmod{p} \\ &= g^r \cdot y \cdot g^{-r} \pmod{p} \\ &= y \end{aligned}$$

Pohlig-Hellman Algorithm Example $p = 1259$

- Consider $p = 1259$ and $g = 2$, and the DLP

$$y = 338 = g^x \pmod{p}$$

- $p - 1 = 34 \cdot 37 = a \cdot b$
- Step 1: Solve for r in

$$\begin{aligned} (g^a)^r &= y^a \pmod{p} \\ (2^{34})^r &= 338^{34} \pmod{p} \\ 870^r &= 463 \pmod{p} \end{aligned}$$

- Since 870 is of order $b = 37$, we solve a smaller DLP
- The solution r is in the set $[0, b - 1] = [0, 36]$
- This DLP gives $r = 27$ since $870^{27} = 463 \bmod p$

Pohlig-Hellman Algorithm

- Step 2: Solve for s in the DLP:

$$\begin{aligned}
(g^b)^s &= y \cdot g^{-r} \pmod{p} \\
(2^{37})^s &= 338 \cdot 2^{-27} \pmod{p} \\
665^s &= 338 \cdot 880 \pmod{p} \\
665^s &= 316 \pmod{p}
\end{aligned}$$

- This is also a smaller DLP, since s is in the set $[0, a-1] = [0, 33]$
- We find $s = 2$, since $665^2 = 316 \pmod{p}$
- Step 3: We find x as

$$x = r + s \cdot b = 27 + 2 \cdot 37 = 101$$

- This is indeed the solution of DLP:

$$g^x = 2^{101} = 338 = y \pmod{p}$$

Complexity of the Pohlig-Hellman Algorithm

- The Pohlig-Hellman algorithm requires two independent DLPs which are order $\sqrt{a}$ and $\sqrt{b}$ when $p = a \cdot b$
- These can be solved using exhaustive search, requiring $O(\sqrt{a})$ and $O(\sqrt{b})$ multiplications
- It works better if a or b factors even further
- The means, we can apply Pohlig-Hellman recursively
- If the largest prime divisor of $p - 1$ is much smaller than p, then Pohlig-Hellman computes DL more quickly

Applications to ECDLP

- The exhaustive search, Shank's, Pollard Rho, and Pohlig-Hellman algorithms are applicable to any group
- They do not require particular properties from the group, and perform only group operations to solve for the DLP
- They are applicable to the ECDLP
- They all require exponential time

- Given the DLP $y = g^x \pmod{p}$, these algorithms require:
  Exhaustive Search: $O(p)$ operations
  Shank's Baby-Step-Giant-Step: $O(\sqrt{p})$ operations and $O(\sqrt{p})$ space
  Pollard Rho: $O(\sqrt{p})$ operations and probably less space
  Pohlig-Hellman: $O(\sqrt{a} + \sqrt{b})$ for $p - 1 = a \cdot b$

Index Calculus Algorithm

- The Index Calculus algorithm is asymptotically faster than the previous algorithms
- The Index Calculus algorithm generates group elements $g^{a \cdot n+b}$
- It then deduces equations for n from random collisions
- However, the Index Calculus algorithm obtains discrete-logarithm equations in a different way

Index Calculus Algorithm

- We are attempting to solve the DLP $y = g^x \pmod{p}$
- Consider the set S, called the factor base, the set of all primes less than or equal to some bound b
- For example, $S = \{2, 3, 5\}$ where $b = 5$
- An element of Z_p^* is called smooth with respect to b, if all of its factors are contained in S
- For example, these elements of Z_{19}^* are smooth wrt $b = 5$:

$$\{2, 3, 4 = 2^2, 5, 6 = 2{\cdot}3, 8 = 2^3, 9 = 3^2, 10 = 2{\cdot}5, 12 = 2^2{\cdot}3, 15 = 3{\cdot}5, 16 = 2^4, 18 = 2{\cdot}3^2\}$$

- These elements of Z_{19}^* are not smooth wrt $b = 5$:

$$\{7, 11, 13, 14 = 2 \cdot 7, 17\}$$

Index Calculus Algorithm

- The Index Calculus algorithm has 3 steps
- Step 1: Take a random α, and compute $g^\alpha \pmod{p}$, and see if it is smooth, that is

$$g^\alpha = \prod_{p_i \in S} p_i^{\alpha_i}$$

- If it is, we obtain its discrete logarithm base g, where all α_i are known

$$\alpha = \sum_{p_i \in S} \alpha_i \log_g p_i \pmod{p-1}$$

- Continue this process until more than $|S|$ equations are known

Index Calculus Algorithm

- Step 2: Solve this system of equations to find the unique values for each of $\log_g p_i \pmod{p-1}$
- Step 3: In this step we find the solution to the DLP $y = g^x \pmod{p}$, that is we compute $\log_g y \pmod{p-1}$
- Select a random α so that $y \cdot g^\alpha \pmod{p}$ is smooth
- When we find such an α, we can compute x using

$$x = \log_g y = -\alpha + \sum_{p_i \in S} \alpha_i \log_g p_i \pmod{p-1}$$

where everything on the right hand side is known

Index Calculus Algorithm Example for $p = 37$

- Consider $p = 37$, $g = 5$ and $S = \{2, 3\}$
- Step 1: Try $\alpha = 6$: $g^\alpha = 5^6 = 11 \pmod{37}$: Not smooth
- Try $\alpha = 7$: $g^\alpha = 5^7 = 18 = 2^1 \cdot 3^2 \pmod{37}$: Smooth
- Thus, we find

$$1 \cdot \log_5 2 + 2 \cdot \log_5 3 = 7 \pmod{36}$$

- Try $\alpha = 14$: $g^\alpha = 5^{14} = 28 = 2^2 \cdot 7 \pmod{37}$: Not smooth
- Try $\alpha = 31$: $g^\alpha = 5^{31} = 24 = 2^3 \cdot 3^1 \pmod{37}$: Smooth
- Thus, we find

$$3 \cdot \log_5 2 + 1 \cdot \log_5 3 = 31 \pmod{36}$$

Index Calculus Algorithm

- Step 2: We solve these two equations

$$\begin{aligned} 1 \cdot \log_5 2 + 2 \cdot \log_5 3 &= 7 \pmod{36} \\ 3 \cdot \log_5 2 + 1 \cdot \log_5 3 &= 31 \pmod{36} \end{aligned}$$

- Expressed in matrix form as

$$\begin{bmatrix} 1 & 2 \\ 3 & 1 \end{bmatrix} \begin{bmatrix} \log_5 2 \\ \log_5 3 \end{bmatrix} = \begin{bmatrix} 7 \\ 31 \end{bmatrix} \pmod{36}$$

- We find the solutions as $\log_5 2 = 11$ and $\log_5 3 = 34$
- These are verified as $5^{11} = 2 \pmod{37}$ and $5^{34} = 3 \pmod{37}$

Index Calculus Algorithm Example for $y = 17$

- Step 3: Suppose we want to find $\log_5 17 \pmod{36}$
- We are trying to solve the DLP: $y = 17 = 5^x \pmod{37}$
- Try $\alpha = 24$: $y \cdot g^\alpha = 17 \cdot 5^{24} = 35 \pmod{37}$: Not smooth
- Try $\alpha = 15$: $y \cdot g^\alpha = 17 \cdot 5^{15} = 12 \pmod{37}$: Smooth
- This number factors as $12 = 2^2 \cdot 3^1$, thus, we find

$$\begin{aligned}
\log_g y &= -\alpha + \sum_{p_i \in S} \alpha_i \log_g p_i \pmod{p-1} \\
\log_5 17 &= -15 + 2 \cdot \log_5 2 + 1 \cdot \log_5 3 \pmod{36} \\
&= -15 + 2 \cdot 11 + 1 \cdot 34 \pmod{36} \\
&= 41 \pmod{36} \\
&= 5
\end{aligned}$$

- The solution is $x = 5$ in $17 = 5^x \pmod{37}$, since $5^5 = 17 \pmod{37}$

Index Calculus Algorithm Example for $y = 19$

- Step 3: Suppose we want to find $\log_5 19 \pmod{36}$
- We are trying to solve the DLP: $y = 19 = 5^x \pmod{37}$
- Try $\alpha = 5$: $y \cdot g^\alpha = 19 \cdot 5^5 = 27 \pmod{37}$: Smooth
- This number factors as $27 = 3^3$, thus, we find

$$\begin{aligned}
\log_g y &= -\alpha + \sum_{p_i \in S} \alpha_i \log_g p_i \pmod{p-1} \\
\log_5 19 &= -5 + 3 \cdot \log_5 3 \pmod{36} \\
&= -5 + 3 \cdot 34 \pmod{36} \\
&= 97 \pmod{36} \\
&= 25
\end{aligned}$$

- Thus $x = 25$ in $19 = 5^x \pmod{37}$, since $5^{25} = 19 \pmod{37}$

Complexity of the Index Calculus Algorithm

- Analysis of the Index Calculus algorithm depends on several factors:
  - How likely is it that $g^\alpha \pmod{p}$ and $y \cdot g^\alpha \pmod{p}$ smooth ?
  - What is the chance that $|S|$ equations are linearly independent $\pmod{p-1}$?
  - How do we solve linear equations mod $p - 1$?
- It is shown that the time complexity is

$$e^{(\frac{1}{\sqrt{2}}+o(1))\sqrt{\log p \log \log p}}$$

- Furthermore, as $p \to \infty$, the Index Calculus algorithm scales very well: the cost becomes p^ϵ where $\epsilon \to 0$
- Compare this to the Pollard Rho algorithm: $\approx \sqrt{p}$