The discrete logarithm problem: theory and cryptographic applications

D. R. Stinson
Department of Combinatorics and Optimization
University of Waterloo
Waterloo Ontario, N2L 3G1

dstinson@uwaterloo.ca

http://cacr.math.uwaterloo.ca/~dstinson/index.html

The Second Lethbridge Workshop on Designs, Codes, Cryptography and Graph Theory
University of Lethbridge, Lethbridge, Alberta
July 9–14, 2001

Definitions

- Let G be a multiplicative abelian group.
- let $\alpha \in G$ have order n
- define $\langle\alpha\rangle = \{\alpha^i : 0 \leq i \leq n-1\}$
- $\langle\alpha\rangle$ is a cyclic subgroup of G

Problem 1: Discrete Logarithm (DLP)	
Instance:	a multiplicative group $(G, \cdot)$ an element $\alpha \in G$ having order n an element $\beta \in \langle\alpha\rangle$
Question:	find the unique integer a, $0 \leq a \leq n-1$, such that $\alpha^a = \beta$ (this integer a is denoted by $\log_\alpha \beta$)

A One-way Function

- finding discrete logarithms in certain (sub)groups is (probably) difficult: there exist groups in which the best known algorithms require time $\Omega\left(\sqrt{n}\right)$

- the inverse operation of exponentiation can be computed efficiently by using the square-and-multiply method

- the function $a \mapsto \alpha^a$ can be computed using $O(\log a)$ group operations

- since we can impose the restriction $0 \leq a \leq n-1$, we have that exponentiation requires $O(\log n)$ group operations

- hence, exponentiation is a one-way function in suitable (sub)groups

Shanks' Algorithm

Algorithm 1: *Shanks*(G, n, α, β)

1. $m \leftarrow \lceil \sqrt{n} \rceil$
2. **for** $j \leftarrow 0$ **to** $m - 1$
   do compute α^{mj}
3. Sort the m ordered pairs (j, α^{mj}) with respect to their second coordinates, obtaining a list L_1
4. **for** $i \leftarrow 0$ **to** $m - 1$
   do compute $\beta\alpha^{-i}$
5. Sort the m ordered pairs $(i, \beta\alpha^{-i})$ with respect to their second coordinates, obtaining a list L_2
6. Find a pair $(j, y) \in L_1$ and a pair $(i, y) \in L_2$ (i.e., find two pairs having identical second coordinates)
7. $\log_\alpha \beta \leftarrow (mj + i) \bmod n$

An Example

Suppose we wish to find $\log_3 525$ in $(\mathbb{Z}_{809}{}^*, \cdot)$. Note that 3 is a primitive element in $\mathbb{Z}_{809}{}^*$, so we have $\alpha = 3$, $n = 808$, $\beta = 525$ and $m = \lceil\sqrt{808}\rceil = 29$. Then

$$\alpha^{29} \bmod 809 = 99.$$

First, we compute the ordered pairs

$$(j, 99^j \bmod 809)$$

for $0 \le j \le 28$. We obtain the list

$(0,1)$	$(1,99)$	$(2,93)$	$(3,308)$	$(4,559)$
$(5,329)$	$(6,211)$	$(7,664)$	$(8,207)$	$(9,268)$
$(10,644)$	$(11,654)$	$(12,26)$	$(13,147)$	$(14,800)$
$(15,727)$	$(16,781)$	$(17,464)$	$(18,632)$	$(19,275)$
$(20,528)$	$(21,496)$	$(22,564)$	$(23,15)$	$(24,676)$
$(25,586)$	$(26,575)$	$(27,295)$	$(28,81)$	

which is then sorted to produce L_1.

An Example (cont.)

The second list contains the ordered pairs

$$(i, 525 \times (3^i)^{-1} \bmod 809),$$

$0 \leq j \leq 28$. It is as follows:

$$\begin{array}{ccccc}
(0,525) & (1,175) & (2,328) & (3,379) & (4,396) \\
(5,132) & (6,44) & (7,554) & (8,724) & (9,511) \\
(10,440) & (11,686) & (12,768) & (13,256) & (14,355) \\
(15,388) & (16,399) & (17,133) & (18,314) & (19,644) \\
(20,754) & (21,521) & (22,713) & (23,777) & (24,259) \\
(25,356) & (26,658) & (27,489) & (28,163) &
\end{array}$$

After sorting this list, we get L_2.

Now, if we proceed simultaneously through the two sorted lists, we find that $(10, 644)$ is in L_1 and $(19, 644)$ is in L_2. Hence, we can compute

$$\begin{aligned}
\log_3 525 &= (29 \times 10 + 19) \bmod 808 \\
&= 309.
\end{aligned}$$

Comments and Analysis

- if $(j, y) \in L_1$ and $(i, y) \in L_2$, then
$$\alpha^{mj} = y = \beta\alpha^{-i},$$
so
$$\alpha^{mj+i} = \beta$$

- we can express $\log_\alpha \beta$ in the form
$$\log_\alpha \beta = mj + i,$$
where $0 \le j, i \le m - 1$

- hence, the search in step 6 will be successful

- steps 3 and 5 can be done in time $O(m \log m)$ using an efficient sorting algorithm

- neglecting logarithmic factors, the algorithm runs in $O(m)$ time with $O(m)$ memory

Pollard Rho Algorithm

- we form a sequence $x_1, x_2, \ldots$ by iteratively applying a random-looking function f

- once we obtain two elements x_i and x_j in the sequence such that $x_i = x_j$ and $i < j$, we can (hopefully) compute $\log_\alpha \beta$

- let $S_1 \cup S_2 \cup S_3$ be a partition of G into three subsets of roughly equal size.

- define $f : \langle \alpha \rangle \times \mathbb{Z}_n \times \mathbb{Z}_n \to \langle \alpha \rangle \times \mathbb{Z}_n \times \mathbb{Z}_n$ as follows:

$$f(x, a, b) = \begin{cases} (\beta x, a, b+1) & \text{if } x \in S_1 \\ (x^2, 2a, 2b) & \text{if } x \in S_2 \\ (\alpha x, a+1, b) & \text{if } x \in S_3. \end{cases}$$

- now define

$$(x_i, a_i, b_i) = \begin{cases} (1, 0, 0) & \text{if } i = 0 \\ f(x_{i-1}, a_{i-1}, b_{i-1}) & \text{if } i \geq 1. \end{cases}$$

Pollard Rho Algorithm (cont.)

- for all $i \geq 0$, it holds that $x_i = \alpha^{a_i}\beta^{b_i}$

- suppose that $x_j = x_i$

- then

$$\alpha^{a_j}\beta^{b_j} = \alpha^{a_i}\beta^{b_i}$$

- if we denote $c = \log_\alpha \beta$, then it must be the case that

$$\alpha^{a_j + cb_j} = \alpha^{a_i + cb_i}$$

- since α has order n, it follows that

$$a_j + cb_j \equiv a_i + cb_i \pmod{n}$$

- if $\gcd(b_j - b_i, n) = 1$, then we can solve for c as follows:

$$c = (a_i - a_j)(b_j - b_i)^{-1} \bmod n$$

Pollard Rho Algorithm (cont.)

- consider the sequence $x_0, \ldots, x_{t-1}$
- assume the values x_i are random (clearly they are not)
- if $t > \sqrt{n}$, then the "Birthday paradox" asserts that there is a good probability that there is at least one collision, $x_i = x_j$ with $i < j$
- storing $\sqrt{n}$ triples requires a lot of memory
- the digraph $x_0 \to x_1 \to \cdots$ consists of a "tail" and an infinitely repeated cycle of length ℓ, say
- it looks like the Greek letter ρ
- if $x_i = x_j$, then $x_{i+\delta} = x_{j+\delta}$ for all integers $\delta \geq 0$

Pollard Rho Algorithm (cont.)

- if $x_i = x_j$, then it is also the case that $x_{i'} = x_{2i'}$ for all i' such that $i' \equiv 0 \pmod{\ell}$ and $i' \geq i$

- among the ℓ consecutive integers $i, \ldots, j-1$, there must be one that is divisible by ℓ

- there exists a value $i' \leq j-1$ that satisfies the two conditions above

- we can impose the restriction $j = 2i$ and expect to find $x_i = x_{2i}$ for $i = O(\sqrt{n})$

- this restriction allows a memory-free implementation of the algorithm

Pollard Rho Algorithm

Algorithm 2: *Pollard rho*(G, n, α, β)

procedure $f(x, a, b)$
  if $x \in S_1$
    then $f \leftarrow (\beta \cdot x, a, (b + 1) \bmod n)$
    else if $x \in S_2$
    then $f \leftarrow (x^2, 2a \bmod n, 2b \bmod n)$
    else $f \leftarrow (\alpha \cdot x, (a + 1) \bmod n, b)$
  return f

main
  define the partition $G = S_1 \cup S_2 \cup S_3$
  $(x, a, b) \leftarrow f(1, 0, 0)$
  $(x', a', b') \leftarrow f(x, a, b)$
  while $x \neq x'$
    do $\begin{cases} (x, a, b) \leftarrow f(x, a, b) \\ (x', a', b') \leftarrow f(x', a', b') \\ (x', a', b') \leftarrow f(x', a', b') \end{cases}$
  if $\gcd(b' - b, n) \neq 1$
    then return "failure"
    else return $(a - a')(b' - b)^{-1} \bmod n$

An Example

The integer $p = 809$ is prime, and it can be verified that the element $\alpha = 89$ has order $n = 101$ in $\mathbb{Z}_{809}{}^*$. The element $\beta = 618$ is in the subgroup $\langle\alpha\rangle$; we will compute $\log_\alpha \beta$.

Suppose we define the sets S_1, S_2 and S_3 as follows:

$$\begin{aligned} S_1 &= \{x \in \mathbb{Z}_{809} : x \equiv 1 \pmod 3\} \\ S_2 &= \{x \in \mathbb{Z}_{809} : x \equiv 0 \pmod 3\} \\ S_3 &= \{x \in \mathbb{Z}_{809} : x \equiv 2 \pmod 3\}. \end{aligned}$$

For $i = 1, 2, \ldots,$ we obtain triples (x_{2i}, a_{2i}, b_{2i}) and (x_i, a_i, b_i) as follows:

An Example (cont.)

i	(x_i, a_i, b_i)	(x_{2i}, a_{2i}, b_{2i})
1	$(618, 0, 1)$	$(76, 0, 2)$
2	$(76, 0, 2)$	$(113, 0, 4)$
3	$(46, 0, 3)$	$(488, 1, 5)$
4	$(113, 0, 4)$	$(605, 4, 10)$
5	$(349, 1, 4)$	$(422, 5, 11)$
6	$(488, 1, 5)$	$(683, 7, 11)$
7	$(555, 2, 5)$	$(451, 8, 12)$
8	$(605, 4, 10)$	$(344, 9, 13)$
9	$(451, 5, 10)$	$(112, 11, 13)$
10	$(422, 5, 11)$	$(422, 11, 15)$

The first collision is $x_{10} = x_{20} = 422$. The equation to be solved is

$$\begin{aligned} c &= (11 - 5)(11 - 15)^{-1} \bmod 101 \\ &= (6 \times 25) \bmod 101 \\ &= 49. \end{aligned}$$

Therefore, $\log_{89} 618 = 49$ in the group $\mathbb{Z}_{809}{}^*$.

Pohlig-Hellman Algorithm

- suppose $n = \prod {p_i}^{c_i}$, where the p_i's are distinct primes

- $a = \log_\alpha \beta \in \mathbb{Z}_n$

- if we can compute $a \bmod {p_i}^{c_i}$ for each i, then we can compute $a \bmod n$ using the Chinese remainder theorem

- suppose q is prime, $n \equiv 0 \pmod{q^c}$ and $n \not\equiv 0 \pmod{q^{c+1}}$

- we will compute $x = a \bmod q^c$

- denote $x = \sum_{i=0}^{c-1} a_i q^i$, where $0 \leq a_i \leq q-1$ for $0 \leq i \leq c-1$

Pohlig-Hellman Algorithm (cont.)

- $a = x + sq^c$ for some integer s

- hence, $a = \sum_{i=0}^{c-1} a_i q^i + sq^c$

- we have the following:

$$\begin{aligned} \beta^{n/q} &= (\alpha^a)^{n/q} \\ &= (\alpha^{a_0 + a_1 q + \cdots + a_{c-1} q^{c-1} + sq^c})^{n/q} \\ &= (\alpha^{a_0 + Kq})^{n/q} \quad (K \in \mathbb{Z}) \\ &= \alpha^{a_0 n/q} \alpha^{Kn} \\ &= \alpha^{a_0 n/q} \end{aligned}$$

- hence, we can determine the correct value of a_0 in time $O(q)$

- once a_0 is found, $a_1, a_2, \ldots$ can be computed in a similar fashion

Pohlig-Hellman Algorithm

Algorithm 3: *Pohlig-Hellman*$(G, n, \alpha, \beta, q, c)$

$j \leftarrow 0$
$\beta_j \leftarrow \beta$
while $j \leq c - 1$

$$\textbf{do} \begin{cases} \delta \leftarrow {\beta_j}^{n/q^{j+1}} \\ \text{find } i \text{ such that } \delta = \alpha^{in/q} \\ a_j \leftarrow i \\ \beta_{j+1} \leftarrow \beta_j \alpha^{-a_j q^j} \\ j \leftarrow j + 1 \end{cases}$$

return $a_0, \ldots, a_{c-1}$

An Example

- suppose $p = 29$ and $\alpha = 2$; then

$$n = p - 1 = 28 = 2^2 7^1$$

- suppose $\beta = 18$, so $a = \log_2 18$.

- we compute $a \bmod 4$ and $a \bmod 7$.

- first, set $q = 2$ and $c = 2$ and apply Algorithm 3.

- $a_0 = 1$ and $a_1 = 1$; hence, $a \equiv 3 \pmod 4$

- next, set $q = 7$ and $c = 1$ and apply Algorithm 3

- $a_0 = 4$, so $a \equiv 4 \pmod 7$

An Example (cont.)

- finally, solve the system

$$\begin{aligned} a &\equiv 3 \pmod{4} \\ a &\equiv 4 \pmod{7} \end{aligned}$$

  using the Chinese remainder theorem

- $a \equiv 11 \pmod{28}$, so $\log_2 18 = 11$ in $\mathbb{Z}_{29}$

Analysis

- a straightforward implementation of Algorithm 3 runs in time $O(cq)$.

- however, each computation of a value i such that $\delta = \alpha^{in/q}$ can be viewed as the solution of a particular instance of the **Discrete Logarithm** problem

- $\delta = \alpha^{in/q}$ if and only if

$$i = \log_{\alpha^{n/q}} \delta$$

- the element $\alpha^{n/q}$ has order q, and therefore each i can be computed (using *Shanks* algorithm, for example) in time $O(\sqrt{q})$

- the complexity of Algorithm 3 can therefore be reduced to $O(c\sqrt{q})$

Index Calculus Algorithm

- all the previous algorithms are "generic" algorithms: they can be applied to any group

- later, we will show that any generic algorithm has complexity $\Omega(\sqrt{n})$

- therefore, the *Shanks* and *Pollard Rho* algorithms have optimal complexity within the class of generic algorithms

- the *Index Calculus* algorithm applies to the particular situation of finding discrete logarithms in ${\mathbb{Z}_p}^*$ when p is prime and α is a primitive element modulo p

- the *Index Calculus* algorithm is much faster (asymptotically) than the algorithms previously considered

Index Calculus Algorithm (cont.)

- a $\boxed{\text{factor base}}$ is a set $\mathcal{FB} = \{p_1, p_2, \ldots, p_B\}$ of "small" primes (e.g., all primes $< 10^{10}$
- the *Index Calculus* algorithm consists of two steps:
    1. (preprocessing step) find the logarithms of the B primes in $\mathcal{FB}$
    2. compute the discrete logarithm of a desired element β, using the knowledge of the discrete logarithms of the elements in $\mathcal{FB}$

Index Calculus Algorithm (step 1)

- take a random value x, compute $\alpha^x \bmod p$, and then determine if $\alpha^x \bmod p$ has all its factors in $\mathcal{FB}$ (using trial division or sieving, for example)

- if this holds, then

$$\alpha^x \equiv {p_1}^{a_1}{p_2}^{a_2} \dots {p_B}^{a_B} \pmod{p}$$

- equivalently,

$$x \equiv a_1 \log_\alpha p_1 + \cdots + a_B \log_\alpha p_B \pmod{p-1}$$

- eventually, we construct B or more congruences of this form

- then we solve a system of linear congruences in the B unknowns, $\log_\alpha p_i$ $(1 \le i \le B)$

Index Calculus Algorithm (step 2)

- in order to compute a desired logarithm $a = \log_\alpha \beta$, choose a random integer s (and compute

$$\gamma = \beta\alpha^s \bmod p$$

- attempt to factor γ over the factor base $\mathcal{FB}$

- if this can be done, then we obtain a congruence of the form

$$\beta\alpha^s \equiv {p_1}^{c_1}{p_2}^{c_2}\ldots{p_B}^{c_B} \pmod{p}.$$

- equivalently,

$$a + s \equiv c_1 \log_\alpha p_1 + \cdots + c_B \log_\alpha p_B \pmod{p-1}.$$

- since all terms in the above congruence are now known except for a, we can easily solve for a

An Example

Suppose $p = 10007$, $\alpha = 5$, and $\mathcal{B} = \{2, 3, 5, 7\}$ $\log_5 5 = 1$, so there are three logs of factor base elements to be determined. Some examples of "lucky" exponents that might be chosen are $x = 4063$, 5136 and 9865:

$$\begin{aligned} 5^{4063} \bmod 10007 &= 42 = 2 \times 3 \times 7 \\ 5^{5136} \bmod 10007 &= 54 = 2 \times 3^3 \\ 5^{9865} \bmod 10007 &= 189 = 3^3 \times 7 \end{aligned}$$

This yields the system of congruences:

$$\begin{aligned} \log_5 2 + \log_5 3 + \log_5 7 &\equiv 4063 \pmod{10006} \\ \log_5 2 + 3 \log_5 3 &\equiv 5136 \pmod{10006} \\ 3 \log_5 3 + \log_5 7 &\equiv 9865 \pmod{10006}. \end{aligned}$$

We obtain $\log_5 2 = 6578$, $\log_5 3 = 6190$, $\log_5 5 = 1$ and $\log_5 7 = 1301$.

An Example (cont.)

Now, let's suppose that we wish to find $\log_5 9451$. Suppose we choose the "random" exponent $s = 7736$, and compute

$$9451 \times 5^{7736} \bmod 10007 = 8400.$$

Since

$$8400 = 2^4 3^1 5^2 7^1,$$

we obtain the following (modulo 10006):

$$\begin{aligned}
&\log_5 9451 \\
&\quad = \; 4\log_5 2 + \log_5 3 + 2\log_5 5 + \log_5 7 - s \\
&\quad = \; 4 \times 6578 + 6190 + 2 \times 1 + 1301 - 7736 \\
&\quad = \; 6057.
\end{aligned}$$

To verify, we can check that

$$5^{6057} \equiv 9451 \pmod{10007}.$$

Analysis

- under reasonable assumptions, the asymptotic running time of the *Index Calculus* algorithmis

$$O\left(e^{(c+o(1))\sqrt{\ln p \ln\ln p}}\right),$$

where c is a small constant

- more sophisticated versions, using the number field sieve, have (conjectured) running time

$$O\left(e^{(C+o(1))(\ln p)^{1/3}(\ln\ln p)^{2/3}}\right),$$

where C is a small constant

- currently implemented algorithms can compute discrete logs when $p \approx 2^{300}$ (p a general prime) and when $p \approx 2^{430}$ (p a special prime)

The subgroup DLP in ${\mathbb{Z}_p}^*$

- suppose p is prime and α_0 is a primitive element modulo p

- suppose q is prime and $q \mid (p-1)$

- $({\mathbb{Z}_p}^*, \cdot)$ has a unique subgroup H of order q, and $\alpha = {\alpha_0}^{(p-1)/q}$ is a generator of H

- the **Discrete Logarithm** problem in H can be attacked via the *Pollard rho algorithm* in H, having complexity $O(\sqrt{q})$; or the *Index calculus algorithm* in ${\mathbb{Z}_p}^*$, having complexity

$$O\left(e^{(C+o(1))(\ln p)^{1/3}(\ln \ln p)^{2/3}}\right)$$

- one should choose p and q so that *both* these potential attacks are infeasible

Using Group Isomorphisms

- can we solve the **Discrete Logarithm** problem in any subgroup $\langle\alpha\rangle$ of order n of any group G by "reducing" the problem to the the easily solved formulation in $(\mathbb{Z}_n, +)$?

- $\langle\alpha\rangle$ is isomorphic to $(\mathbb{Z}_n, +)$, so there is a bijection $\phi : \langle\alpha\rangle \to \mathbb{Z}_n$ such that

$$\phi(xy) = (\phi(x) + \phi(y)) \bmod n$$

  for all $x, y \in \langle\alpha\rangle$

- $\phi(\alpha^a) = a\phi(\alpha) \bmod n$, so we have that

$$\beta = \alpha^a \Leftrightarrow a\phi(\alpha) \equiv \phi(\beta) \pmod{n}$$

- solving for a (using the *Extended Euclidean algorithm*), we have that

$$\log_\alpha \beta = \phi(\beta)(\phi(\alpha))^{-1} \bmod n$$

Using Group Isomorphisms (cont.)

- if we have an efficient method of computing the isomorphism ϕ, then we would have an efficient algorithm to compute discrete logarithms in $\langle \alpha \rangle$

- computing discrete logarithms in $\langle \alpha \rangle$ is equivalent to finding an explicit isomorphism between $\langle \alpha \rangle$ and $(\mathbb{Z}_n, +)$

- for example, the MOV reduction uses the Weil pairing to find an explicit isomorphism between supersingular elliptic curves over $\mathbb{F}_q$ and a subgroup of the multiplicative group $(\mathbb{F}_{q^k}{}^*, \cdot)$, where k is a small positive integer

Elliptic Curves over the Reals

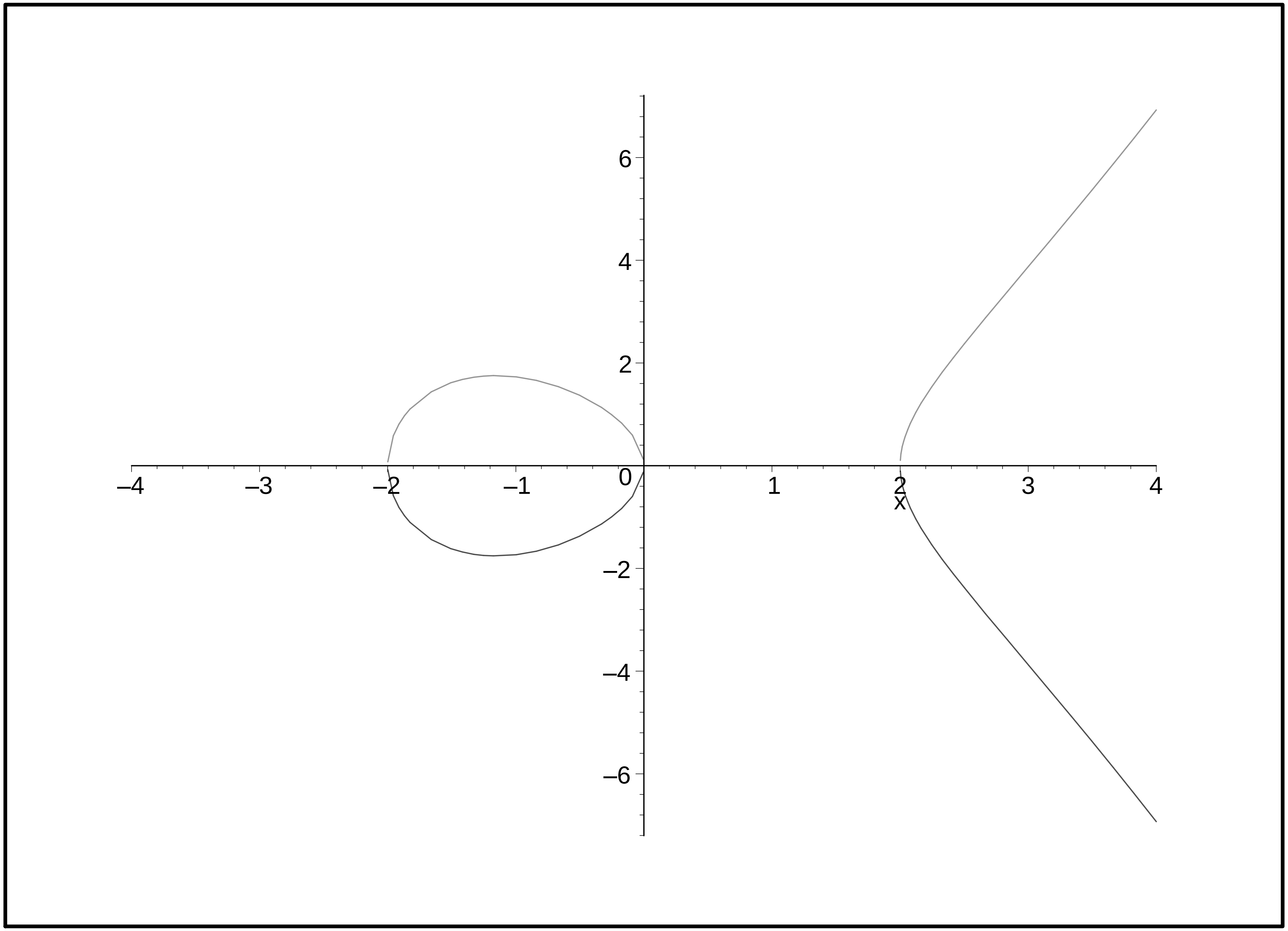
- let $a, b \in \mathbb{R}$, $4a^3 + 27b^2 \neq 0$

- a $\boxed{\text{non-singular elliptic curve}}$ is the set E of solutions $(x, y) \in \mathbb{R} \times \mathbb{R}$ to the equation
$$y^2 = x^3 + ax + b,$$
together with a special point, $\mathcal{O}$, which is called the $\boxed{\text{point at infinity}}$

- the condition $4a^3 + 27b^2 \neq 0$ implies that the equation $x^3 + ax + b = 0$ has three distinct roots

- we will define a binary operation, denoted "+", which makes E into an abelian group

- the point at infinity, $\mathcal{O}$, will be the identity element, so $P + \mathcal{O} = \mathcal{O} + \mathcal{P} = \mathcal{P}$ for all $P \in E$.


6
4
2
0
–2
–4
–6
–4
–3
–2
–1
1
2
3
4
x

The Addition Operation

- suppose $P, Q \in E$, $P = (x_1, y_1)$, $Q = (x_2, y_2)$

- we consider three cases:

  1. $x_1 \neq x_2$

  2. $x_1 = x_2$ and $y_1 = -y_2$

  3. $x_1 = x_2$ and $y_1 = y_2$

case 1 Define L to be the line through P and Q. L intersects E in P, Q, and in one further point, denoted R'. Reflect R' in the x-axis to get R. Then define $P + Q = R$.

case 2 Define $(x, y) + (x, -y) = \mathcal{O}$.

case 3 This case is similar to case 1, except that L is the tangent to E at the given point.

The Addition Operation (cont.)

- it is simple to derive algebraic formulas in each of the three cases

- it is easy to see that:

  1. "$+$" is closed on E;
  2. "$+$" is commutative;
  3. $\mathcal{O}$ is the identity element; and
  4. every point has an inverse

- it is also true (but not obvious) that "$+$" is associative

- therefore $(E, +)$ is an abelian group

Elliptic Curves over $\mathbb{Z}_p$

- let $p > 3$ be prime

- elliptic curves over $\mathbb{Z}_p$ can be defined exactly as they were over the reals provided that all operations over $\mathbb{R}$ are replaced by analogous operations in $\mathbb{Z}_p$

- $(E, +)$ is now a *finite* abelian group

- $p + 1 - 2\sqrt{p} \leq |E| \leq p + 1 + 2\sqrt{p}$

- there exist positive integers n_1 and n_2 such that

  1. $n_2 \mid n_1$;

  2. $n_2 \mid (p - 1)$; and

  3. $(E, +)$ is isomorphic to $\mathbb{Z}_{n_1} \times \mathbb{Z}_{n_2}$

Elliptic Curves over other Finite Fields

- elliptic curves can also be defined over finite fields of characteristic 2 and 3

- Koblitz curves are a special class of elliptic curves, defined over $\mathbb{F}_{2^m}$

- for every m, there are two Koblitz curves, defined by the equations

$$y^2 + xy = x^3 + 1 \quad \text{and}$$
$$y^2 + xy = x^3 + x^2 + 1$$

- Koblitz curves permit some optimizations of elliptic curve arithmetic, which makes them attractive for cryptographic applications

Elliptic Curve DLP

- in the elliptic curve DLP, it is necessary to specify the prime p, the constants a and b used to define the curve, the generating point α, the order n of α, and the point β in the subgroup generated by α

- often a curve E will be chosen so that $|E|$ is prime

- in this case, $(E, +)$ is necessarily cyclic, and any point $\neq \mathcal{O}$ is a generator of E

- there is no known version of the *Index Calculus* algorithm that works in the setting of elliptic curves

- the largest instances of elliptic curve DLPs that have been solved are "Certicom challenges": a random elliptic curve defined over $\mathbb{Z}_p$ where $p \approx 2^{97}$; and a Koblitz curve defined over $\mathbb{F}_{2^{109}}$

- solution of a challenge on a random elliptic curve defined over $\mathbb{Z}_p$, where $p \approx 2^{109}$, is on-going

The ECC2K-108 Challenge

- the ECC2K-108 challenge was solved on April 4th 2000 by a group led by Robert Harley, Damien Doligez, Daniel de Rauglaudre and Xavier Leroy of INRIA (France)

- for this achievement, Certicom awarded them a cash prize of $10,000

- the following is from the official Certicom press release:

  "The solution to the ECC2K-108 challenge is believed to be the largest effort ever expended in a public-key cryptography challenge. It took four months and involved approximately 9,500 machines and 1300 volunteers from 40 countries. The amount of work required to solve the ECC2K-108 challenge was about 50 times that required to solve the 512-bit RSA cryptosystem."

Equivalent Key Sizes

- the most popular cryptographic settings for the **Discrete Logarithm** problem are the subgroup DLP in $\mathbb{Z}_p{}^*$, and the elliptic curve DLP

- estimated equivalent (secure) key sizes have been suggested by Lenstra and Verheul:

year	subgroup DLP		elliptic curve DLP
	$\log_2 p$	$\log_2 q$	$\log_2 p$
2001	990	126	133
2011	1416	139	148
2021	1937	153	163

The Discrete Log Problem in Cryptography

- the security of many cryptographic protocols depend on the difficulty of the **discrete logarithm** problem

- the most important of these protocols are the following:
  - ElGamal signature scheme, DSA and ECDSA
  - Diffie-Hellman key exchange
  - ElGamal and elliptic curve public-key cryptosystems

- infeasibility of the **discrete logarithm** problem is necessary, but perhaps not sufficient for these protocols to be secure

ElGamal Signature Scheme

ElGamal Signature Scheme

Let p be a prime such that the discrete log problem in $\mathbb{Z}_p$ is intractable, and let $\alpha \in \mathbb{Z}_p{}^*$ be a primitive element. Let $\mathcal{M} = \mathbb{Z}_p{}^*$, $\mathcal{S} = \mathbb{Z}_p{}^* \times \mathbb{Z}_{p-1}$, and define

$$\mathcal{K} = \{(p, \alpha, a, \beta) : \beta \equiv \alpha^a \pmod{p}\}.$$

The values p, α and β are public, and a is secret.

For $K = (p, \alpha, a, \beta)$, and for a (secret) random number $k \in \mathbb{Z}_{p-1}{}^*$, define

$$sig_K(x, k) = (\gamma, \delta),$$

where

$$\gamma = \alpha^k \bmod p$$

and

$$\delta = (x - a\gamma)k^{-1} \bmod (p-1).$$

For $x, \gamma \in \mathbb{Z}_p{}^*$ and $\delta \in \mathbb{Z}_{p-1}$, define

$$ver_K(x, \gamma, \delta) = \text{true} \Leftrightarrow \beta^\gamma \gamma^\delta \equiv \alpha^x \pmod{p}.$$

The Verification Equation

- how does the signer construct a pair (γ, δ) such that

$$\beta^{\gamma}\gamma^{\delta} \equiv \alpha^{x} \text{ (mod } p\text{)?}$$

- recall that $\beta = \alpha^{a} \bmod p$ and $\gamma = \alpha^{k} \bmod p$

- we want

$$\alpha^{a\gamma}\alpha^{k\delta} \equiv \alpha^{x} \pmod{p}$$

- it is necessary and sufficient that

$$a\gamma + k\delta \equiv x \pmod{p-1}$$

Security Considerations

- in order for the *ElGamal Signature Scheme* in ${\mathbb{Z}_p}^*$ to be secure, we require $p \approx 2^{1024}$, say

- then a signature is 2048 bits in length

- the *Digital Signature Algorithm* (*DSA*) bases its security on the subgroup DLP, where the subgroup has size $\approx 2^{160}$

- the *DSA* has a much shorter signature of 320 bits, which is important for applications in constrained environments such as smart cards and wireless applications

- there is also an *Elliptic Curve Digital Signature Algorithm* (*ECDSA*) having a short signature length

Digital Signature Algorithm

Digital Signature Algorithm

Let p be a 1024-bit prime and let q be a 160-bit prime that divides $p-1$. Let $\alpha \in \mathbb{Z}_p{}^*$ have order q, such that the subgroup DLP in $\langle\alpha\rangle$ is intractable. Let $\mathcal{M} = \mathbb{Z}_q{}^*$, $\mathcal{S} = \mathbb{Z}_q{}^* \times \mathbb{Z}_q{}^*$, and define

$$\mathcal{K} = \{(p, q, \alpha, a, \beta) : \beta \equiv \alpha^a \pmod{p}\}.$$

p, q, α and β are public, and a is secret.
For $K = (p, q, \alpha, a, \beta)$, and for a (secret) random number k, $1 \leq k \leq q-1$, define

$$sig_K(x, k) = (\gamma, \delta),$$

where

$$\gamma = (\alpha^k \bmod p) \bmod q$$
$$\delta = (x + a\gamma)k^{-1} \bmod q.$$

Check that $\gamma, \delta \neq 0$.
For $x \in \mathbb{Z}_q{}^*$ and $\gamma, \delta \in \mathbb{Z}_q{}^*$, verification is done by performing the following computations:

$$e_1 = x\delta^{-1} \bmod q, \quad e_2 = \gamma\delta^{-1} \bmod q$$
$$ver_K(x, \gamma, \delta) = \text{true} \Leftrightarrow (\alpha^{e_1}\beta^{e_2} \bmod p) \bmod q = \gamma$$

Elliptic Curve Digital Signature Algorithm

Elliptic Curve Digital Signature Algorithm

Let p be prime and let E be an elliptic curve over $\mathbb{Z}_p$ such that $n = |E|$ is prime. Let $\alpha \in E$. Let $\mathcal{M} = \mathbb{Z}_n$, $\mathcal{S} = {\mathbb{Z}_n}^* \times {\mathbb{Z}_n}^*$, and define

$$\mathcal{K} = \{(p, E, n, \alpha, a, \beta) : \beta = a\alpha\}.$$

p, E, n, α and β are public, and a is secret.
For $K = (p, E, n, \alpha, a, \beta)$, and for a (secret) random number k, $1 \leq k \leq n-1$, define

$$sig_K(x, k) = (\gamma, \delta),$$

where

$$k\alpha = (x_1, y_1)$$
$$\gamma = x_1 \bmod n$$
$$\delta = (x + a\gamma)k^{-1} \bmod n.$$

Check that $\gamma, \delta \neq 0$.
For $x \in \mathbb{Z}_n$ and $\gamma, \delta \in {\mathbb{Z}_n}^*$, compute the following:

$$e_1 = x\delta^{-1} \bmod n, \quad e_2 = \gamma\delta^{-1} \bmod n$$
$$e_1\alpha + e_2\beta = (x_1, y_1)$$
$$ver_K(x, \gamma, \delta) = \text{true} \Leftrightarrow x_1 \bmod n = \gamma$$

Elliptic Curve Digital Signature Standard

- several elliptic curves for use in the *Elliptic Curve Digital Signature Algorithm* have been standardized by NIST

- these include the following:

  1. five "randomly" generated elliptic curves over $\mathbb{Z}_p$, for certain primes $p \approx 2^{192}$, 2^{224}, 2^{256}, 2^{384} and 2^{521}

  2. five Koblitz curves over $\mathbb{F}_{2^m}$, for $m = 163$, 233, 283, 409 and 571

  3. five "randomly" generated elliptic curves over $\mathbb{F}_{2^m}$, for $m = 163$, 233, 283, 409 and 571

Diffie-Hellman Key Exchange

Diffie-Hellman Key Exchange

Let $(G, \cdot)$ be a multiplicative group, and let $\alpha \in G$ have order n. Suppose that the **Discrete Logarithm** problem in $\langle \alpha \rangle$ is infeasible.

1. Alice chooses a random integer a, $0 \leq a \leq n-1$, computes $y_A = \alpha^a$, and sends y_A to Bob.
2. Bob chooses a random integer b, $0 \leq b \leq n-1$, computes $y_B = \alpha^b$, and sends y_B to Alice.
3. Alice computes $y_{AB} = (y_B)^a$.
4. Bob computes $y_{BA} = (y_A)^b$.
5. $y_{AB} = y_{BA} = \alpha^{ab}$ is the secret key shared by Alice and Bob.

The Diffie-Hellman Problems

Problem 2: Computational Diffie-Hellman (CDH)

Instance: a multiplicative group $(G, \cdot)$
an element $\alpha \in G$ having order n
elements $\beta, \gamma \in \langle \alpha \rangle$

Question: find the unique element $\delta \in \langle \alpha \rangle$ such that

$$\log_\alpha \beta \times \log_\alpha \gamma \equiv \log_\alpha \delta \pmod{n}$$

Problem 3: Decision Diffie-Hellman (DDH)

Instance: a multiplicative group $(G, \cdot)$
an element $\alpha \in G$ having order n
elements $\beta, \gamma, \delta \in \langle \alpha \rangle$

Question: is it the case that

$$\log_\alpha \beta \times \log_\alpha \gamma \equiv \log_\alpha \delta \pmod{n}?$$

Reductions

- there exist Turing reductions

$$\textbf{DDH} \propto_T \textbf{CDH} \quad \text{and} \quad \textbf{CDH} \propto_T \textbf{DLP}$$

- the first reduction is proven as follows:

  1. given $\alpha, \beta, \gamma, \delta$, use an algorithm that solves **CDH** to find the value δ' such that $\log_\alpha \delta' \equiv \log_\alpha \beta \times \log_\alpha \gamma \pmod{n}$

  2. then check to see if $\delta' \equiv \delta \pmod{n}$

- the second reduction is also very simple:

  1. given α, β, γ, use an algorithm that solves **DLP** to find $b = \log_\alpha \beta$ and $c = \log_\alpha \gamma$

  2. compute $d = bc \bmod n$ and $\delta = \alpha^d$

Security of Diffie-Hellman Key Exchange

- *Diffie-Hellman key exchange* is "secure" only if **CDH** is infeasible

- in view of the second Turing reduction, this is possibly a stronger assumption than the assumption that **DLP** is infeasible

- however, partial information may be leaked about the secret key y_{AB} even if **CDH** is infeasible

- for example, the quadratic residuosity of y_{AB} can easily be determined if the scheme is implemented in $(\mathbb{Z}_p{}^*, \cdot)$ and α is a primitive element

- y_{AB} is a quadratic residue iff at least one of α^a or α^b is a quadratic residue

- quadratic residuosity in $\mathbb{Z}_p{}^*$ can easily be checked using Euler's criterion

Security of Diffie-Hellman Key Exchange (cont.)

- therefore we have identified a setting in which **DDH** easily solved even though **CDH** is believed to be infeasible

- security of *Diffie-Hellman key exchange* requires that **DDH** be infeasible

- **DDH** is thought to be infeasible in prime order subgroups in which the **DLP** is infeasible

- for example, suppose $p = 2q+1$, where p and q are prime, and $\alpha \in \mathbb{Z}_p{}^*$ is a square of a primitive element

ElGamal Cryptosystem

ElGamal Public-key Cryptosystem

Let $(G, \cdot)$ be a multiplicative group, and let $\alpha \in G$ have order n. Suppose that the **Discrete Logarithm** problem in $H = \langle \alpha \rangle$ is infeasible. Let $\mathcal{P} = H$, $\mathcal{C} = H \times H$, and define

$$\mathcal{K} = \{(\alpha, a, \beta) : \beta = \alpha^a, 0 \leq a \leq n-1\}.$$

The values α and β are public, and a is secret. For $K = (\alpha, a, \beta)$, and for a (secret) random number r such that $0 \leq r \leq n-1$, define $e_K(x, r) = (y_1, y_2)$, where

$$y_1 = \alpha^r$$

and

$$y_2 = x\beta^r.$$

For $y_1, y_2 \in H$, define

$$d_K(y_1, y_2) = y_2({y_1}^a)^{-1}.$$

Security of the ElGamal Cryptosystem

- the *ElGamal Cryptosystem* is closely related to *Diffie-Hellman key exchange*: ElGamal decryption (without knowing the secret key) is equivalent to **CDH**

- the reduction in one direction is proven as follows:

  1. let A be an algorithm that solves **CDH**

  2. given α, β, y_1 and y_2, compute $A(\alpha, \beta, y_1) = \alpha^{ra}$

  3. then $x = (\alpha^{ra})^{-1} y_2$ is the decryption of (y_1, y_2)

- the reduction in the other direction is also easy:

  1. let B be an algorithm that decrypts ElGamal ciphertexts

  2. given α, β and γ, let $y_1 = \gamma$, $y_2 = 1$, and compute $x = B(\alpha, \beta, y_1, y_2)$

  3. then $\delta = x^{-1}$

Semantic Security of the ElGamal Cryptosystem

- a cryptosystem is $\boxed{\text{semantically secure}}$ if it is infeasible to distinguish encryptions of x_1 from encryptions of x_2, where x_1 and x_2 are arbitrary plaintexts

- if a public-key cryptosystem has a deterministic encryption scheme, it is always possible to encrypt x_1 and x_2 and see which one yields the ciphertext y

- therefore, a public-key cryptosystem can achieve semantic security only if encryption is randomized

- semantic security of the *ElGamal Cryptosystem* is equivalent to the infeasibility of **DDH**

Generic and Non-generic Algorithms

- the complexity of any algorithm depends on the way in which the input is "encoded"

- Garey and Johnson assert that "If we restrict ourselves to 'reasonable' encoding schemes, then the particular encoding scheme used should not affect the determination of whether a given problem is intractable"

- however, there are many reasonable ways in which groups can be encoded, and the encoding scheme appears to affect greatly the (in)tractability of the **discrete logarithm** problem

- recall that an algorithm that does not depend on the particular encoding used is called a generic algorithm

Generic and Non-generic Algorithms

- *Shank's algorithm*, the *Pollard rho algorithm* and the *Pohlig-Hellman algorithm* are all generic

- the *Index Calculus algorithm* is not generic; it involves treating elements of ${\mathbb{Z}_p}^*$ as integers, and then computing their factorizations

- in the additive group $(\mathbb{Z}_n, +)$, the **Discrete Logarithm** problem can be solved easily using the *Extended Euclidean algorithm* in time $O((\log n)^2)$

- suppose that $\gcd(\alpha, n) = 1$, so α is a generator of $\mathbb{Z}_n$

- then

$$\log_\alpha \beta = \beta\alpha^{-1} \bmod n$$

- this is much faster than a generic algorithm in a group of size n

Complexity of Generic Algorithms

- a result of Nechaev and Shoup provides a lower bound on the complexity of generic algorithms for the **Discrete Logarithm** problem

- the complexity of any generic algorithm for the **Discrete Logarithm** problem is $\Omega(\sqrt{n})$

- hence, the *Shanks* and *rho* algorithms are essentially optimal within the class of generic algorithms

- we will prove a version of this result using an elementary combinatorial argument

- we assume n is prime and consider generic algorithms in $(\mathbb{Z}_n, +)$

Encodings and Generic Algorithms

- an $\boxed{\text{encoding}}$ of $(\mathbb{Z}_n, +)$ is any injective mapping $\sigma : \mathbb{Z}_n \to S$, where S is a finite set

- there are many possible encodings of $(\mathbb{Z}_n, +)$

- for example, let α be a primitive element in ${\mathbb{Z}_p}^*$, let $n = p - 1$, and define $\sigma(i) = \alpha^i \bmod p$, $0 \leq i \leq n - 1$

- solving the **Discrete Logarithm** problem in $({\mathbb{Z}_p}^*, \cdot)$ with respect to the primitive element α is equivalent to solving the **Discrete Logarithm** problem in $(\mathbb{Z}_n, +)$ with generator 1 under the encoding σ

- a $\boxed{\text{generic algorithm}}$ is one that works correctly when the encoding function σ is a random injective function and group operations are performed by an $\boxed{\text{oracle}}$ (i.e., a black box)

Generic Algorithms

- suppose we have encodings of two (possibly unknown) group elements, say $\sigma(i)$ and $\sigma(j)$

- for any integers $c, d \in \mathbb{Z}_n$, the oracle can compute $\sigma((di + cj) \bmod n)$ in one unit of time

- the input to a generic algorithm, *GenLog*, consists of $\sigma_1 = \sigma(1)$ and $\sigma_2 = \sigma(a)$, where $a \in \mathbb{Z}_n$ is chosen uniformly at random

- the goal of *GenLog* is to output the unknown value a

- *GenLog* will use the oracle to generate a sequence of m, say, encodings of linear combinations of 1 and a

Generic Algorithms (cont.)

- the execution of *GenLog* can be specified by a list of m distinct ordered pairs $(c_i, d_i) \in \mathbb{Z}_n \times \mathbb{Z}_n$

- for each ordered pair (c_i, d_i), the oracle computes the encoding $\sigma_i = \sigma((d_i + c_i a) \bmod n)$

- we define $(c_1, d_1) = (0, 1)$ and $(c_2, d_2) = (1, 0)$ so our notation is consistent with the input to the algorithm *GenLog*

Computing the Discrete Log

- the encoding function σ is injective, so
$$d_i + c_i a \equiv d_j + c_j a \text{ (mod } p)$$
if and only if
$$\sigma_i = \sigma_j$$

- suppose that $\sigma_i = \sigma_j$ for two integers $i \neq j$

- if $c_i = c_j$, then $(c_i, d_i) = (c_j, d_j)$, which is a contradiction

- hence $c_i \neq c_j$ and *GenLog* can compute a as follows:
$$a = (d_i - d_j)(c_j - c_i)^{-1} \bmod p$$

Non-adaptive Algorithms

- suppose *GenLog* chooses a set

  $$S = \{(c_i, d_i) : 1 \leq i \leq w\} \subseteq \mathbb{Z}_p \times \mathbb{Z}_p$$

  of w distinct ordered pairs at the beginning of the algorithm

- i.e., *GenLog* is a $\boxed{\text{non-adaptive algorithm}}$

- then the list of w encodings is obtained from the oracle

- define

  $$\mathsf{Good}(S) = \{a \in \mathbb{Z}_p : a = (d_i - d_j)(c_j - c_i)^{-1} \bmod p\}$$

- the value of a can be computed by *Genlog* if $a \in \mathsf{Good}(S)$

- if $|\mathsf{Good}(S)| = p - 1$, then it must be the case that a is the unique element in $\mathbb{Z}_p \backslash \mathsf{Good}(S)$

Las Vegas Algorithms

- we assume that *GenLog* is a Las Vegas algorithm, σ is a random encoding and $a \in \mathbb{Z}_n$ is chosen randomly

- let $\epsilon \in \mathbb{R}$, $0 < \epsilon \leq 1$

- a $\boxed{\text{Las Vegas algorithm}}$ having success probability ϵ is a randomized algorithm, say A, which, given any problem instance I, will correctly solve the instance I with probability $\geq \epsilon$

- furthermore, if A does not correctly solve the instance I, then it will return the output "no answer"

- if $|\mathsf{Good}(S)| \leq p - 2$, then *Genlog* must return "no answer", since it cannot determine a unique value for a given the information that it has

Success Probability

- the success probability of *GenLog* is at most

$$\epsilon = \begin{cases} \frac{|\mathsf{Good}(S)|}{p} & \text{if } |\mathsf{Good}(S)| \leq p-2 \\ 1 & \text{if } |\mathsf{Good}(S)| \geq p-1 \end{cases}$$

- clearly

$$|\mathsf{Good}(S)| \leq \binom{|S|}{2},$$

so

$$\epsilon \leq \binom{|S|}{2}/p$$

whenever $\binom{|S|}{2} \leq p-2$

- if $\epsilon = 1$, then $|S|$ is $\Omega(\sqrt{p})$

- identical results hold for adaptive Las Vegas algorithms

- this implies that any generic algorithm for the **DLP** in a (sub)group of prime order p has complexity $\Omega(\sqrt{p})$

Other Security Results in Generic Groups

- similar results hold in $(\mathbb{Z}_n, +)$ when n is composite

- it can also be shown that algorithms for **DDH** and **CDH** have complexity $\Omega(\sqrt{p})$ [Shoup]

- there is no efficient generic reduction from **CDH** to **DDH** [Maurer and Wolf]

- *ECDSA* is secure in the generic group model if the hash function satisfies certain natural security properties [Brown]

Low Hamming Weight DLP

- Let G be a multiplicative abelian group.
- Let $\alpha \in G$, and suppose $\beta \in \langle \alpha \rangle$.
- The discrete logarithm $\log_\alpha \beta$ is the unique integer x such that $0 \leq x \leq \mathrm{ord}(\alpha) - 1$ and $\alpha^x = \beta$.
- Denote $m = \lceil \log_2(\mathrm{ord}(\alpha)) \rceil$.
- Then
$$x = \sum_{i=0}^{m-1} x_i 2^i,$$
where $x_i \in \{0, 1\}$ for $0 \leq i \leq m - 1$.
- The $\boxed{\text{hamming weight}}$ of an integer x, denoted $\mathrm{wt}(x)$, is the number of 1's in its binary representation.

Low Hamming Weight DLP (cont.)

- Given α, β and an integer $t < m$, the **hamming weight t discrete logarithm problem** is to compute $x = \log_\alpha \beta$ whenever $\mathrm{wt}(\log_\alpha \beta) = t$.

- various cryptosystems may use secret keys in which $t \ll m$ since this will lead to more efficient implementations

- the number of possible weight t keys is only $\binom{m}{t}$, as compared to 2^m

- the *Shanks* algorithm reduces the complexity of a general search for a discrete logarithm to $2^{m/2}$

- we investigate modifications that can be used to search for a hamming weight t discrete logarithm

Preliminaries

- The binary vector $(x_0, \ldots, x_{m-1})$ can be regarded as the characteristic vector of a subset of $\mathbb{Z}_m$ in an obvious way.

- define
$$\mathsf{set}(x) = \{i : x_i = 1\},$$
where $(x_0, \ldots, x_{m-1})$ is the binary representation of x; and
$$\mathsf{val}(Y) = \sum_{i \in Y} 2^i,$$
where $Y \subseteq \mathbb{Z}_m$

- val and set are inverse functions, and
$$\mathsf{val}(Y_1 \cup Y_2) = \mathsf{val}(Y_1) + \mathsf{val}(Y_2)$$
if $Y_1 \cap Y_2 = \emptyset$.

Two Lemmas

Lemma 1
Suppose that $Y_1, Y_2 \subseteq \mathbb{Z}_m$ *and*

$$\alpha^{\mathsf{val}(Y_1)} = \beta(\alpha^{\mathsf{val}(Y_2)})^{-1}.$$

Then

$$\log_\alpha \beta = (\mathsf{val}(Y_1) + \mathsf{val}(Y_2)) \bmod \mathsf{ord}(\alpha).$$

Lemma 2
Suppose that $\mathsf{wt}(\log_\alpha \beta) = t$*, where* t *is an even positive integer. Then there exist subsets* $Y_1, Y_2 \subseteq \mathbb{Z}_m$ *such that the following hold:*

1. $Y_1 \cap Y_2 = \emptyset$,

2. $|Y_1| = |Y_2| = t/2$; *and*

3. $\alpha^{\mathsf{val}(Y_1)} = \beta(\alpha^{\mathsf{val}(Y_2)})^{-1}$.

Heiman-Odlyzko Algorithm

Algorithm 4: *Heiman-Odlyzko*(G, m, t, α, β)

1. For all $Y \subseteq \mathbb{Z}_m$ such that $|Y| = t/2$, compute $\alpha^{\mathsf{val}(Y)}$
2. Sort the ordered pairs $(\mathsf{val}(Y), \alpha^{\mathsf{val}(Y)})$ with respect to their second coordinates, obtaining a list L_1
3. For all $Y \subseteq \mathbb{Z}_m$ such that $|Y| = t/2$, compute $\beta(\alpha^{\mathsf{val}(Y)})^{-1}$
4. Sort the ordered pairs $(\mathsf{val}(Y), \beta(\alpha^{\mathsf{val}(Y)})^{-1})$ with respect to their second coordinates, obtaining a list L_2
5. If possible, find a pair $(j, y) \in L_1$ and a pair $(i, y) \in L_2$ (i.e., find two pairs having identical second coordinates)
6. If the previous step was successful, then $\log_\alpha \beta \leftarrow (i + j) \bmod n$
7. Otherwise, output fail.

Analysis

- "fail" means that either $\beta \notin \langle \alpha \rangle$ or $\mathrm{wt}(\log_\alpha \beta) \neq t$.

- The complexity of this algorithm (neglecting logarithmic factors) is $O\left(\binom{m}{t/2}\right)$.

- The space requirement is also $O\left(\binom{m}{t/2}\right)$.

Example

- Suppose $G = \mathbb{Z}_{31}{}^*$, $\alpha = 3$, $\beta = 11$ and $t = 4$

- $\mathsf{ord}(\alpha) = 30$, so $m = 5$

- we compute the following:

Y	$\mathsf{val}(Y)$	$\alpha^{\mathsf{val}(Y)}$	$\beta(\alpha^{\mathsf{val}(Y)})^{-1}$
$\{0,1\}$	3	27	5
$\{0,2\}$	5	26	4
$\{0,3\}$	9	29	10
$\{0,4\}$	17	22	16
$\{1,2\}$	6	16	22
$\{1,3\}$	10	25	24
$\{1,4\}$	18	4	26
$\{2,3\}$	12	8	13
$\{2,4\}$	20	5	27
$\{3,4\}$	24	2	21

- since (for example), $\alpha^5 = 26 = \beta(\alpha^{18})^{-1}$, we have that $\log_\alpha \beta = 5 + 18 = 23$

Coppersmith Randomized Algorithm

Algorithm 5: *Coppersmith*(G, m, t, α, β)

1. Let B be a random $\frac{m}{2}$-subset of X
2. For all $Y \subseteq B$ such that $|Y| = t/2$, compute $\alpha^{\mathsf{val}(Y)}$
3. Sort the ordered pairs $(\mathsf{val}(Y), \alpha^{\mathsf{val}(Y)})$ with respect to their second coordinates, obtaining a list L_1
4. For all $Y \subseteq \mathbb{Z}_m \backslash B$ such that $|Y| = t/2$, compute $\beta(\alpha^{\mathsf{val}(Y)})^{-1}$
5. Sort the ordered pairs $(\mathsf{val}(Y), \beta(\alpha^{\mathsf{val}(Y)})^{-1})$ with respect to their second coordinates, obtaining a list L_2
6. If possible, find a pair $(j, y) \in L_1$ and a pair $(i, y) \in L_2$ (i.e., find two pairs having identical second coordinates)
7. If the previous step was successful, then $\log_\alpha \beta \leftarrow (i + j) \bmod n$
8. Otherwise, output fail.

Analysis

- The algorithm is successful if $\beta \in \langle \alpha \rangle$ and
$$|B \cap \mathsf{set}(\log_\alpha \beta)| = \frac{t}{2}.$$

- This happens with probability
$$p = \frac{\binom{t}{t/2}\binom{m-t}{(m-t)/2}}{\binom{m}{m/2}}.$$

- We can compute a lower bound on p using the following lemma.

Lemma 3 *Suppose that n and λn are positive integers, where $0 < \lambda < 1$. Define*
$$H(\lambda) = -\lambda \log_2 \lambda - (1-\lambda)\log_2(1-\lambda).$$
Then
$$\frac{1}{\sqrt{8n\lambda(1-\lambda)}} 2^{nH(\lambda)} \leq \binom{n}{\lambda n} \leq \frac{1}{\sqrt{2\pi n\lambda(1-\lambda)}} 2^{nH(\lambda)}.$$

Analysis (cont.)

- Applying this lemma with $\lambda = 1/2$, it is easy to see that

$$p \geq \sqrt{\frac{\pi}{8}}\sqrt{\frac{m}{t(m-t)}} > ct^{-1/2}.$$

- the expected number of iterations of the algorithm required until the algorithm is successful is $1/p = O(\sqrt{t})$

- The expected complexity of the algorithm is $O\left(\sqrt{t}\binom{m/2}{t/2}\right)$.

- If the algorithm fails, it could because $\beta \notin \langle\alpha\rangle$, $\mathrm{wt}(\log_\alpha \beta) \neq t$, or because the choice of B is bad.

Splitting Systems

Definition 1
Suppose m and t are even integers, $0 < t < m$. An $\boxed{(m, t)\text{-splitting system}}$ *is a pair $(X, \mathcal{B})$ that satisfies the following properties:*

1. *$|X| = m$, and $\mathcal{B}$ is a set of $\frac{m}{2}$-subsets of X, called* $\boxed{\text{blocks}}$

2. *for every $Y \subseteq X$ such that $|Y| = t$, there exists a block $B \in \mathcal{B}$ such that $|B \cap Y| = t/2$.*

We will use the notation $(N; m, t)$-SS to denote an (m, t)-splitting system having N blocks.

A Construction

Lemma 4 (Coppersmith)
For all even integers m and t with $0 < t < m$, there exists an $(\frac{m}{2}; m, t)$-SS.

Proof. Let $X = \mathbb{Z}_m$ and define

$$B_i = \{i + j \bmod m : 0 \leq j \leq m/2 - 1\}$$

for $i \in \mathbb{Z}_m$. Let $\mathcal{B} = \{B_i : 0 \leq i \leq m/2 - 1\}$. It is easy to show that $(X, \mathcal{B})$ is an (m, t)-splitting system. $\square$

Example When $m = 10$, we have

$$\begin{aligned} B_0 &= \{0, 1, 2, 3, 4\} \\ B_1 &= \{1, 2, 3, 4, 5\} \\ B_2 &= \{2, 3, 4, 5, 6\} \\ B_3 &= \{3, 4, 5, 6, 7\} \\ B_4 &= \{4, 5, 6, 7, 8\} \end{aligned}$$

Coppersmith Deterministic Algorithm

Algorithm 6: *Splitting System*$(G, m, t, \alpha, \beta, \mathcal{B})$

1. $k \leftarrow 1$;
2. $B \leftarrow B_k$;
3. For all $Y \subseteq B$ such that $|Y| = t/2$, compute $\alpha^{\mathsf{val}(Y)}$
4. Sort the ordered pairs $(\mathsf{val}(Y), \alpha^{\mathsf{val}(Y)})$ with respect to their second coordinates, obtaining a list L_1
5. For all $Y \subseteq \mathbb{Z}_m \backslash B$ such that $|Y| = t/2$, compute $\beta(\alpha^{\mathsf{val}(Y)})^{-1}$
6. Sort the ordered pairs $(\mathsf{val}(Y), \beta(\alpha^{\mathsf{val}(Y)})^{-1})$ with respect to their second coordinates, obtaining a list L_2
7. If possible, find a pair $(j, y) \in L_1$ and a pair $(i, y) \in L_2$ (i.e., find two pairs having identical second coordinates)
8. If the previous step was successful, then set $\log_\alpha \beta \leftarrow (i + j) \bmod n$ and quit.
9. Otherwise, $k \leftarrow k + 1$.
10. If $k \leq N$, then go to step 2; otherwise, output fail.

Comments

- "fail" means that either $\beta \notin \langle \alpha \rangle$ or $\mathrm{wt}(\log_\alpha \beta) \neq t$.

- The complexity of this algorithm is $O\left(N\binom{m/2}{t/2}\right)$.

- The space requirement is $O\left(\binom{m/2}{t/2}\right)$.

- The splitting systems constructed above yield an algorithm having complexity $O\left(m\binom{m/2}{t/2}\right)$.

Improved results concerning splitting systems

- Using the probabilistic method, the following can be proved:

Theorem 5 *For any even integers t and m with $0 < t < m$, there exists an $(N; m, t)$-SS with $N \approx c_0\, t^{3/2} \log_2 m$, where c_0 is a constant.*

- this proves the existence of a (non-uniform) deterministic algorithm having complexity $O\left(t^{3/2} (\log m) \binom{m/2}{t/2}\right)$.

Explicit Constructions Using Perfect Hash Families

Definition 2
Let $n \geq m$. An $\boxed{(n,m)\text{-}\textit{hash function}}$ *is a function $h : A \rightarrow B$, where $|A| = n$ and $|B| = m$. h is said to be* $\boxed{\textit{balanced}}$ *provided that $|h^{-1}(y)| = n/m$ for all $y \in B$.*

Definition 3
Let $n \geq m \geq w$. An $\boxed{(n,m,w)\text{-}\textit{perfect hash family}}$ *is a finite set $\mathcal{H}$ of (n,m)-hash functions such that $h : A \rightarrow B$ for each $h \in \mathcal{H}$, where $|A| = n$ and $|B| = m$, with the property that for any $X \subseteq A$ with $|X| = w$, there exists at least one $h \in \mathcal{H}$ such that $h|_X$ is one-to-one. $\mathcal{H}$ is said to be an* $\boxed{(n,m,w)\text{-}\textit{balanced perfect hash family}}$ *if $\mathcal{H}$ is an (n,m,w)-perfect hash family and h is balanced for every $h \in \mathcal{H}$.*

We will use the notation BPHF$(N; n, m, w)$ to denote an (n,m,w)-balanced perfect hash family with $|\mathcal{H}| = N$.

Explicit Constructions (cont.)

- We can depict a BPHF($N; n, m, w$) as an $N \times n$ array on m symbols, say A, where each row of A corresponds to one of the functions in the family.

- This array has the property that, for any subset of w columns, there exists at least one row such that the entries in the w given columns of that row are distinct; and any row of A contains exactly n/m occurrences of each symbol.

- we present a BPHF($2; 16, 4, 2$):

$$\begin{pmatrix} 1 & 1 & 1 & 1 & \dots & 4 & 4 & 4 & 4 \\ 1 & 2 & 3 & 4 & \dots & 1 & 2 & 3 & 4 \end{pmatrix}$$

Explicit Constructions (cont.)

Theorem 6
Suppose there exist a BPHF$(N_0; n, m, t)$ *and an SS*$(N_1; m, t)$*. Then there exists an SS*$(N_0N_1; n, t)$*.*

Proof. Let M be the $N_1 \times m$ incidence matrix of an SS$(N_1; m, t)$, and denote the columns of M by $c_1, \ldots, c_m$. Let A be the array representation of the BPHF$(N_0; n, m, t)$, and replace each entry $y = A(i, j)$ by the column vector c_y. The resulting matrix M_1 is the incidence matrix of an SS$(N_0N_1; n, t)$.

Corollary 7
If there exists a BPHF$(N_0; n, m, t)$*, then there exists an SS*$(N_0m/2; n, t)$*.*

- With a suitable choice of input designs, we can construct SS$(N; m, t)$ in which N is $O(t^4(\log m)^2)$.

Example

A BPHF$(2;16,4,2)$ and an SS$(2;4,2)$ both exist, so there exists an SS$(4;16,2)$.

$$\begin{array}{ccc}
\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix} & \leftarrow & \begin{array}{c} \{1,2\} \\ \{2,3\} \end{array} \\
+ & & \\
\begin{pmatrix} 1 & 1 & 1 & \ldots & 4 & 4 \\ 1 & 2 & 3 & \ldots & 3 & 4 \end{pmatrix} & & \\
\Downarrow & & \\
\begin{pmatrix} 1 & 1 & 1 & \ldots & 0 & 0 \\ 0 & 0 & 0 & \ldots & 0 & 0 \\ 1 & 1 & 0 & \ldots & 0 & 0 \\ 0 & 1 & 1 & \ldots & 1 & 0 \end{pmatrix} & \rightarrow & \begin{array}{c} \{1,2,3,4,5,6,7,8\} \\ \{5,6,7,8,9,10,11,12\} \\ \{1,2,5,6,9,10,13,14\} \\ \{2,3,6,7,10,11,14,15\} \end{array}
\end{array}$$

Constructions for BPHF

- Let q be a prime power. An $\boxed{(N, K, D, q)\text{-code}}$ is a set $\mathcal{C}$ of K vectors (called $\boxed{\text{codewords}}$) in $(\mathbb{F}_q)^N$ such that the Hamming distance between any two distinct vectors in $\mathcal{C}$ is at least D.

- WLOG, we can assume that there does not exist any coordinate in which every codeword has the entry "0"

- $\mathcal{C}$ is $\boxed{\text{linear}}$ if it is a subspace of $(\mathbb{F}_q)^N$; in this case $K = q^k$, where $k = \dim(\mathcal{C})$.

Theorem 8
If a linear (N, K, D, q)-code exists, then there exists a BPHF$(N; K, q, w)$, provided that

$$\frac{D}{N} > 1 - \frac{1}{\binom{w}{2}}.$$

Constructions for BPHF

- Using Reed-Solomon codes, we obtain the following corollary.

Corollary 9
Suppose that q is a prime power, $0 < \ell < q$ is an integer, and $q > (\ell - 1)\binom{w}{2}$. Then there exists a BPHF$(q; q^\ell, q, w)$.

Further Reading

- a recent survey paper on the **DLP**:

  A.M. Odlyzko. Discrete logarithms: the past and the future. *Designs, Codes, and Cryptography* **19** (2000), 129–145.

- a recent survey of square-root algorithms for the **DLP**:

  E. Teske. Square-root algorithms for the discrete logarithm problem (a survey). Technical report CORR 01-07, Department of C&O, University of Waterloo. To appear in *Public Key Cryptography and Computational Number Theory*. Available from

  `http://www.cacr.math.uwaterloo.ca/`

Further Reading (cont.)

- the Certicom elliptic curve DLP challenges:

  Certicom ECC challenge. Available from

  `http://www.certicom.com/resources/ecc_chall`

- a complete description of ECDSA:

  D. Johnson and A. Menezes. The elliptic curve digitial signature algorithm (ECDSA). Technical report CORR 99-34, Department of C&O, University of Waterloo. Available from

  `http://www.cacr.math.uwaterloo.ca/`

- the analysis of security levels of various cryptosystems with respect to key length is from:

  A.K. Lenstra and E.R. Verheul. Selecting cryptographic key sizes. To appear in *Journal of Cryptology*. Available from

  `http://www.cryptosavvy.com/`

Further Reading (cont.)

- a recent survey paper on the **DDH**:

  D. Boneh. The decision Diffie-Hellman problem. *Lecture Notes in Computer Science*, **1423** (1998), 48–63. (Proceedings of the Third Algorithmic Number Theory Symposium.)

- the discussion of generic algorithms for **DLP** is from:

  M. Chateauneuf and D.R. Stinson. Slope packings and coverings, and generic algorithms for the discrete logarithm problem. Preprint.

- results on the low hamming weight **DLP**:

  D.R. Stinson. Some baby-step giant-step algorithms for the low hamming weight discrete logarithm problem. To appear in *Mathematics of Computation*. Posted on the AMS web site, March 7, 2001. Available from

  `http://www.ams.org/mcom/`