CSE107: Intro to Modern Cryptography

https://cseweb.ucsd.edu/classes/sp22/cse107-a/

Emmanuel Thomé

May 3, 2022

Lecture 9b

Computational Number Theory (end of previous lecture)

Algorithms on numbers

Plan

Algorithms on numbers

Measuring Running Time of Algorithms on Numbers

In an algorithms course, the cost of arithmetic is often assumed to be $\mathcal{O}(1)$, because numbers are small. In cryptography numbers are

very, very BIG!

Typical numbers are 2^{512}, 2^{1024}, 2^{2048}: hundreds or thousands of bits.

Numbers are provided to algorithms in binary. The length of a, denoted $|a|$, is the number of bits in the binary encoding of a.

Example: $|7| = 3$ because 7 is 111 in binary.

Running time is measured as a function of the lengths of the inputs.

Algorithms on numbers

The straightforward algorithms have the following complexities:

Algorithm	Input	Output	Time
ADD	$a,\ b$	$a+b$	$\mathcal{O}(\lvert a\rvert+\lvert b\rvert)$
MULT	$a,\ b$	ab	$\mathcal{O}(\lvert a\rvert\cdot\lvert b\rvert)$
INT-DIV	$a,\ N$	q,r	$\mathcal{O}(\lvert a\rvert\cdot\lvert N\rvert)$
MOD	$a,\ N$	$a \bmod N$	$\mathcal{O}(\lvert a\rvert\cdot\lvert N\rvert)$
EXT-GCD	$a,\ N$	(d, a', N')	$\mathcal{O}(\lvert a\rvert\cdot\lvert N\rvert)$
MOD-INV	$a\in\mathbb{Z}_N^*,\ N$	$a^{-1} \bmod N$	$\mathcal{O}(\lvert N\rvert^2)$
MOD-EXP	$a\in\mathbb{Z}_N,\ n,\ N$	$a^n \bmod N$	$\mathcal{O}(\lvert n\rvert\cdot\lvert N\rvert^2)$
EXP_G	$a\in G,\ n$	$a^n\in G$	$\mathcal{O}(\lvert n\rvert)$ G-ops

Plan

Algorithms on numbers

- (Extended) gcd
- Exponentiation

Extended gcd

Definition (EXT-GCD)

EXT-GCD(a, N) returns (r, u, v) such that

$$r = \gcd(a, N) = a \cdot u + N \cdot v \ .$$

Example: EXT-GCD$(12, 20) =$

Extended gcd

Definition (EXT-GCD)

EXT-GCD(a, N) returns (r, u, v) such that

$$r = \gcd(a, N) = a \cdot u + N \cdot v \ .$$

Example: EXT-GCD$(12, 20) = (4, 2, -1)$ because

$$4 = \gcd(12, 20) = 12 \cdot 2 + 20 \cdot (-1) \ .$$

The (extended) Euclidean algorithm

Algorithm for gcd

To compute the (extended) gcd, we use the (extended) Euclidean algorithm.

Extended gcd Algorithm: rough idea

Definition (EXT-GCD)

EXT-GCD(a, N) returns (r, u, v) such that

$$r = \gcd(a, N) = a \cdot u + N \cdot v \; .$$

Lemma

Let $(q, r) = \text{INT-DIV}(a, N)$. *Then,* $\gcd(a, N) = \gcd(N, r)$

We use this lemma repeatedly.

Extended gcd Algorithm: code

Alg EXT-GCD(a, N) $\quad$ // $(a, N) \neq (0, 0)$
$(r_0, u_0, v_0) \leftarrow (N, 0, 1)$ $\quad$ // $u_0 a + v_0 N = r_0$
$(r_1, u_1, v_1) \leftarrow (a, 1, 0)$ $\quad$ // $u_1 a + v_1 N = r_1$
while $r_1 \neq 0$
$\quad (q, r_2) \leftarrow$ INT-DIV(r_0, r_1); $\quad$ // $r_0 - q r_1 = r_2$
$\quad u_2 = u_0 - q u_1$
$\quad v_2 = v_0 - q v_1$ $\quad$ // now $u_2 a + v_2 N = r_2$
$\quad (r_0, u_0, v_0) \leftarrow (r_1, u_1, v_1)$
$\quad (r_1, u_1, v_1) \leftarrow (r_2, u_2, v_2)$
return (r_0, u_0, v_0) $\quad$ // $u_0 a + v_0 N = r_0 = \gcd(a, N)$

Running time is $\mathcal{O}(|a| \cdot |N|)$, so the extended gcd can be computed in quadratic time. If $0 < a < N$ then $\mathrm{abs}(u) \leq N$ and $\mathrm{abs}(v) \leq a$ where $\mathrm{abs}(\cdot)$ denotes the absolute value.

Analysis showing all this is non-trivial (worst case is Fibonacci numbers).

Modular Inverse

For a, N such that $\gcd(a, N) = 1$, we want to compute $a^{-1} \bmod N$, meaning the unique $a' \in \mathbb{Z}_N^*$ satisfying $aa' \equiv 1 \pmod{N}$.

But if we let $(d, a', N') \leftarrow \text{EXT-GCD}(a, N)$ then

$$d = 1 = \gcd(a, N) = a \cdot a' + N \cdot N'$$

But $N \cdot N' \equiv 0 \pmod{N}$ so $aa' \equiv 1 \pmod{N}$

Alg MOD-INV(a, N)
$(d, a', N') \leftarrow \text{EXT-GCD}(a, N)$
return $a' \bmod N$

Modular inverse can be computed in quadratic time.

Plan

Algorithms on numbers

- (Extended) gcd
- Exponentiation

Modular Exponentiation

Let G be a group and $a \in G$. For $n \in \mathbb{N}$, we want to compute $a^n \in G$.

We know that

$$a^n = \underbrace{a \cdot a \cdots a}_{n}$$

Consider:

$y \leftarrow 1$
for $i = 1, \ldots, n$ do $y \leftarrow y \cdot a$
return y

Question: Is this a good algorithm?

Modular Exponentiation

Let G be a group and $a \in G$. For $n \in \mathbb{N}$, we want to compute $a^n \in G$.

We know that

$$a^n = \underbrace{a \cdot a \cdots a}_{n}$$

Consider:

$y \leftarrow 1$
for $i = 1, \ldots, n$ do $y \leftarrow y \cdot a$
return y

Question: Is this a good algorithm?

Answer: It is correct but VERY SLOW. The number of group operations is $\mathcal{O}(n) = \mathcal{O}(2^{|n|})$ so it is exponential time. For $n \approx 2^{512}$ it is prohibitively expensive.

Fast exponentiation idea

We can compute

$$a \longrightarrow a^2 \longrightarrow a^4 \longrightarrow a^8 \longrightarrow a^{16} \longrightarrow a^{32}$$

in just 5 steps by repeated squaring. So we can compute a^n in i steps when $n = 2^i$.

But what if n is not a power of 2?

Square-and-Multiply Exponentiation Example

Suppose the binary length of n is 5, meaning the binary representation of n has the form $b_4 b_3 b_2 b_1 b_0$. (We sometimes write $n = (b_4 b_3 b_2 b_1 b_0)_2$.)

Then

$$\begin{aligned} n &= 2^4 b_4 + 2^3 b_3 + 2^2 b_2 + 2^1 b_1 + 2^0 b_0 \\ &= 16 b_4 + 8 b_3 + 4 b_2 + 2 b_1 + b_0 \ . \end{aligned}$$

We want to compute a^n. Our exponentiation algorithm will proceed to compute the values $y_5, y_4, y_3, y_2, y_1, y_0$ in turn, as follows:

$$\begin{array}{lllll} y_5 & = & \mathbf{id} & & \\ y_4 & = & y_5^2 \cdot a^{b_4} & = & a^{b_4} \\ y_3 & = & y_4^2 \cdot a^{b_3} & = & a^{2b_4+b_3} \\ y_2 & = & y_3^2 \cdot a^{b_2} & = & a^{4b_4+2b_3+b_2} \\ y_1 & = & y_2^2 \cdot a^{b_1} & = & a^{8b_4+4b_3+2b_2+b_1} \\ y_0 & = & y_1^2 \cdot a^{b_0} & = & a^{16b_4+8b_3+4b_2+2b_1+b_0} \ . \end{array}$$

Square-and-Multiply Exponentiation Example

Let $N = 131$, $G = \mathbb{Z}_N^*$, and $a = 2 \in \mathbb{Z}_N^*$.
We want to compute $a^{107} \bmod N$.

We start with $107 = 64 + 32 + 0 + 8 + 0 + 2 + 1 = (1101011)_2$.

$(1101011)_2$ $\qquad$ $y \leftarrow a = 2,$

Square-and-Multiply Exponentiation Example

Let $N = 131$, $G = \mathbb{Z}_N^*$, and $a = 2 \in \mathbb{Z}_N^*$.
We want to compute $a^{107} \bmod N$.

We start with $107 = 64 + 32 + 0 + 8 + 0 + 2 + 1 = (1101011)_2$.

$$(1101011)_2 \qquad y \leftarrow a = 2,$$
$$(1101011)_2 \qquad y \leftarrow y^2 a = a^3 = 8,$$

Square-and-Multiply Exponentiation Example

Let $N = 131$, $G = \mathbb{Z}_N^*$, and $a = 2 \in \mathbb{Z}_N^*$.
We want to compute $a^{107} \bmod N$.

We start with $107 = 64 + 32 + 0 + 8 + 0 + 2 + 1 = (1101011)_2$.

$$
\begin{aligned}
(1101011)_2 &\qquad y \leftarrow a = 2, \\
(1101011)_2 &\qquad y \leftarrow y^2 a = a^3 = 8, \\
(1101011)_2 &\qquad y \leftarrow y^2 = a^6 = 64,
\end{aligned}
$$

Square-and-Multiply Exponentiation Example

Let $N = 131$, $G = \mathbb{Z}_N^*$, and $a = 2 \in \mathbb{Z}_N^*$.
We want to compute $a^{107} \bmod N$.

We start with $107 = 64 + 32 + 0 + 8 + 0 + 2 + 1 = (1101011)_2$.

$$
\begin{array}{rc}
(1101011)_2 & y \leftarrow a = 2, \\
(1101011)_2 & y \leftarrow y^2 a = a^3 = 8, \\
(1101011)_2 & y \leftarrow y^2 = a^6 = 64, \\
(1101011)_2 & y \leftarrow y^2 a = a^{13} = 8192 \equiv 70,
\end{array}
$$

Square-and-Multiply Exponentiation Example

Let $N = 131$, $G = \mathbb{Z}_N^*$, and $a = 2 \in \mathbb{Z}_N^*$.
We want to compute $a^{107} \bmod N$.

We start with $107 = 64 + 32 + 0 + 8 + 0 + 2 + 1 = (1101011)_2$.

$$
\begin{aligned}
(1101011)_2 & \quad y \leftarrow a = 2, \\
(1101011)_2 & \quad y \leftarrow y^2 a = a^3 = 8, \\
(1101011)_2 & \quad y \leftarrow y^2 = a^6 = 64, \\
(1101011)_2 & \quad y \leftarrow y^2 a = a^{13} = 8192 \equiv 70, \\
(1101011)_2 & \quad y \leftarrow y^2 = a^{26} \equiv 53,
\end{aligned}
$$

Square-and-Multiply Exponentiation Example

Let $N = 131$, $G = \mathbb{Z}_N^*$, and $a = 2 \in \mathbb{Z}_N^*$.
We want to compute $a^{107} \bmod N$.

We start with $107 = 64 + 32 + 0 + 8 + 0 + 2 + 1 = (1101011)_2$.

$$
\begin{array}{lc}
(1101011)_2 & y \leftarrow a = 2, \\
(1101011)_2 & y \leftarrow y^2 a = a^3 = 8, \\
(1101011)_2 & y \leftarrow y^2 = a^6 = 64, \\
(1101011)_2 & y \leftarrow y^2 a = a^{13} = 8192 \equiv 70, \\
(1101011)_2 & y \leftarrow y^2 = a^{26} \equiv 53, \\
(1101011)_2 & y \leftarrow y^2 a = a^{53} \equiv 116,
\end{array}
$$

Square-and-Multiply Exponentiation Example

Let $N = 131$, $G = \mathbb{Z}_N^*$, and $a = 2 \in \mathbb{Z}_N^*$.
We want to compute $a^{107} \bmod N$.

We start with $107 = 64 + 32 + 0 + 8 + 0 + 2 + 1 = (1101011)_2$.

$$
\begin{array}{ll}
(1101011)_2 & y \leftarrow a = 2, \\
(1101011)_2 & y \leftarrow y^2 a = a^3 = 8, \\
(1101011)_2 & y \leftarrow y^2 = a^6 = 64, \\
(1101011)_2 & y \leftarrow y^2 a = a^{13} = 8192 \equiv 70, \\
(1101011)_2 & y \leftarrow y^2 = a^{26} \equiv 53, \\
(1101011)_2 & y \leftarrow y^2 a = a^{53} \equiv 116, \\
(1101011)_2 & y \leftarrow y^2 a = a^{107} \equiv 57,
\end{array}
$$

So $2^{107} \equiv 57 \pmod{131}$.

Square-and-Multiply Exponentiation Algorithm

Let $\text{bin}(n) = b_{k-1} \ldots b_0$ be the binary representation of n, meaning

$$n = \sum_{i=0}^{k-1} b_i 2^i$$

Alg $\text{EXP}_G(a, n)$ $\quad // \; a \in G, \; n \geq 1$
$b_{k-1} \ldots b_0 \leftarrow \text{bin}(n)$
$y \leftarrow 1$
for $i = k - 1$ downto 0 do $y \leftarrow y^2 \cdot a^{b_i}$
return y

The running time is $\mathcal{O}(|n|)$ group operations.

MOD-EXP(a, n, N) returns $a^n \bmod N$ in time $\mathcal{O}(|n| \cdot |N|^2)$, meaning is cubic time.

Variants of Square-and-Multiply

There are many variants of the Square-and-Multiply algorithm.

- Left-to-Right (a.k.a. most significant bit first), as we presented.
- Right-to-Left.
- Fixed-window.
- Sliding-window.
- And more.

Algorithms on numbers

Algorithm	Input	Output	Time
ADD	$a,\ b$	$a+b$	$\mathcal{O}(\lvert a\rvert + \lvert b\rvert)$
MULT	$a,\ b$	ab	$\mathcal{O}(\lvert a\rvert \cdot \lvert b\rvert)$
INT-DIV	$a,\ N$	q, r	$\mathcal{O}(\lvert a\rvert \cdot \lvert N\rvert)$
MOD	$a,\ N$	$a \bmod N$	$\mathcal{O}(\lvert a\rvert \cdot \lvert N\rvert)$
EXT-GCD	$a,\ N$	(d, a', N')	$\mathcal{O}(\lvert a\rvert \cdot \lvert N\rvert)$
MOD-INV	$a \in \mathbb{Z}_N^*,\ N$	$a^{-1} \bmod N$	$\mathcal{O}(\lvert N\rvert^2)$
MOD-EXP	$a \in \mathbb{Z}_N,\ n,\ N$	$a^n \bmod N$	$\mathcal{O}(\lvert n\rvert \cdot \lvert N\rvert^2)$
EXP_G	$a \in G,\ n$	$a^n \in G$	$\mathcal{O}(\lvert n\rvert)$ G-ops

CSE107: Intro to Modern Cryptography

https://cseweb.ucsd.edu/classes/sp22/cse107-a/

Emmanuel Thomé

May 3, 2022

Lecture 10a

Discrete logarithms and RSA

Plan

Cyclic groups and discrete logarithms

Finding cyclic groups

Plan

Generators and cyclic groups

Let G be a group of order m and let $g \in G$. We let

$$\langle g \rangle = \{\, g^i \;:\; i \in \mathbb{Z}_m \,\} \,.$$

The size $|\langle g \rangle|$ of the set $\langle g \rangle$ need not equal m. It could be smaller.

Fact: $|\langle g \rangle|$ is always a divisor of m.

Definition (order of an element; generator; cyclic groups)

The order of $g \in G$ is defined to be $|\langle g \rangle|$.

We say that $g \in G$ is a generator (or primitive element) of G if $\langle g \rangle = G$, meaning the order of g is m.

We say that G is cyclic if it has a generator, meaning there exists $g \in G$ such that g is a generator of G.

Generators and cyclic groups: Example

Let $G = \mathbb{Z}_{11}^* = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$, which has order $m = 10$.

i	0	1	2	3	4	5	6	7	8	9	10
$2^i \mod 11$	1	2	4	8							
$5^i \mod 11$											

so

Generators and cyclic groups: Example

Let $G = \mathbb{Z}_{11}^* = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$, which has order $m = 10$.

i	0	1	2	3	4	5	6	7	8	9	10
$2^i \mod 11$	1	2	4	8	5	10					
$5^i \mod 11$											

so

Generators and cyclic groups: Example

Let $G = \mathbb{Z}_{11}^* = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$, which has order $m = 10$.

i	0	1	2	3	4	5	6	7	8	9	10
$2^i \mod 11$	1	2	4	8	5	10	9				
$5^i \mod 11$											

so

Generators and cyclic groups: Example

Let $G = \mathbb{Z}_{11}^* = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$, which has order $m = 10$.

i	0	1	2	3	4	5	6	7	8	9	10
$2^i \mod 11$	1	2	4	8	5	10	9	7			
$5^i \mod 11$											

so

Generators and cyclic groups: Example

Let $G = \mathbb{Z}_{11}^* = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$, which has order $m = 10$.

i	0	1	2	3	4	5	6	7	8	9	10
$2^i \mod 11$	1	2	4	8	5	10	9	7	3	6	1
$5^i \mod 11$	1	5	3	4	9	1	5	3	4	9	1

so

$$\begin{aligned} \langle 2 \rangle &= \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\} \\ \langle 5 \rangle &= \{1, 3, 4, 5, 9\} \end{aligned}$$

- 2 a generator because $\langle 2 \rangle = \mathbb{Z}_{11}^*$.
- 5 is not a generator because $\langle 5 \rangle \neq \mathbb{Z}_{11}^*$.
- $\mathbb{Z}_{11}^*$ is cyclic because it has a generator.

Generators and cyclic groups: Example

Let $G = \mathbb{Z}_{12}^* = \{1,$

Generators and cyclic groups: Example

Let $G = \mathbb{Z}_{12}^* = \{1, 5,$

Generators and cyclic groups: Example

Let $G = \mathbb{Z}_{12}^* = \{1, 5, 7,$

Generators and cyclic groups: Example

Let $G = \mathbb{Z}_{12}^* = \{1, 5, 7, 11\}$, which has order $m = 4$.

i	0	1	2	3
$5^i \mod 12$	1	5	1	5
$7^i \mod 12$	1	7	1	7
$(11)^i \mod 12$	1	11	1	11

so

$$\begin{aligned} \langle 5 \rangle &= \{1, 5\} \\ \langle 7 \rangle &= \{1, 7\} \\ \langle 11 \rangle &= \{1, 11\} \end{aligned}$$

Is $\mathbb{Z}_{12}^*$ cyclic?

Generators and cyclic groups: Example

Let $G = \mathbb{Z}_{12}^* = \{1, 5, 7, 11\}$, which has order $m = 4$.

i	0	1	2	3
$5^i \mod 12$	1	5	1	5
$7^i \mod 12$	1	7	1	7
$(11)^i \mod 12$	1	11	1	11

so

$$\begin{aligned} \langle 5 \rangle &= \{1, 5\} \\ \langle 7 \rangle &= \{1, 7\} \\ \langle 11 \rangle &= \{1, 11\} \end{aligned}$$

Is $\mathbb{Z}_{12}^*$ cyclic? No it is not, because no element has order 4.

Plan

Discrete Logarithms

If $G = \langle g \rangle$ is a cyclic group of order m then for every $a \in G$ there is a unique exponent $i \in \mathbb{Z}_m$ such that $g^i = a$. We call i the discrete logarithm of a to base g and denote it by

$$\mathrm{DLog}_{G,g}(a)$$

The discrete log function is the inverse of the exponentiation function:

$$\begin{array}{rcll} \mathrm{DLog}_{G,g}(g^i) & = & i & \text{for all } i \in \mathbb{Z}_m \\ g^{\mathrm{DLog}_{G,g}(a)} & = & a & \text{for all } a \in G. \end{array}$$

Discrete Logarithms: Example

Let $G = \mathbb{Z}_{11}^* = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$, which is a cyclic group of order $m = 10$. We know that 2 is a generator, so $\mathrm{DLog}_{G,2}(a)$ is the exponent $i \in \mathbb{Z}_{10}$ such that $2^i \bmod 11 = a$.

i	0	1	2	3	4	5	6	7	8	9
$2^i \bmod 11$	1	2	4	8	5	10	9	7	3	6

a	1	2	3	4	5	6	7	8	9	10
$\mathrm{DLog}_{G,2}(a)$										

Discrete Logarithms: Example

Let $G = \mathbb{Z}_{11}^* = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$, which is a cyclic group of order $m = 10$. We know that 2 is a generator, so $\mathrm{DLog}_{G,2}(a)$ is the exponent $i \in \mathbb{Z}_{10}$ such that $2^i \bmod 11 = a$.

i	0	1	2	3	4	5	6	7	8	9
$2^i \bmod 11$	1	2	4	8	5	10	9	7	3	6

a	1	2	3	4	5	6	7	8	9	10
$\mathrm{DLog}_{G,2}(a)$	0	1								

Discrete Logarithms: Example

Let $G = \mathbb{Z}_{11}^* = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$, which is a cyclic group of order $m = 10$. We know that 2 is a generator, so $\mathrm{DLog}_{G,2}(a)$ is the exponent $i \in \mathbb{Z}_{10}$ such that $2^i \bmod 11 = a$.

i	0	1	2	3	4	5	6	7	8	9
$2^i \bmod 11$	1	2	4	8	5	10	9	7	3	6

a	1	2	3	4	5	6	7	8	9	10
$\mathrm{DLog}_{G,2}(a)$	0	1	8							

Discrete Logarithms: Example

Let $G = \mathbb{Z}_{11}^* = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$, which is a cyclic group of order $m = 10$. We know that 2 is a generator, so $\mathrm{DLog}_{G,2}(a)$ is the exponent $i \in \mathbb{Z}_{10}$ such that $2^i \bmod 11 = a$.

i	0	1	2	3	4	5	6	7	8	9
$2^i \mod 11$	1	2	4	8	5	10	9	7	3	6

a	1	2	3	4	5	6	7	8	9	10
$\mathrm{DLog}_{G,2}(a)$	0	1	8	2						

Discrete Logarithms: Example

Let $G = \mathbb{Z}_{11}^* = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$, which is a cyclic group of order $m = 10$. We know that 2 is a generator, so $\mathrm{DLog}_{G,2}(a)$ is the exponent $i \in \mathbb{Z}_{10}$ such that $2^i \bmod 11 = a$.

i	0	1	2	3	4	5	6	7	8	9
$2^i \bmod 11$	1	2	4	8	5	10	9	7	3	6

a	1	2	3	4	5	6	7	8	9	10
$\mathrm{DLog}_{G,2}(a)$	0	1	8	2	4	9	7	3	6	5

Computing Discrete Logs

Let $G = \langle g \rangle$ be a cyclic group of order m with generator $g \in G$.

Input: $X \in G$

Desired Output: $\mathrm{DLog}_{G,g}(X)$

That is, we want x such that $g^x = X$.

for $x = 0, \ldots, m-1$ do
  if $g^x = X$ then return x

Is this a good algorithm?

Computing Discrete Logs

Let $G = \langle g \rangle$ be a cyclic group of order m with generator $g \in G$.

Input: $X \in G$

Desired Output: $\mathrm{DLog}_{G,g}(X)$

That is, we want x such that $g^x = X$.

for $x = 0, \ldots, m-1$ do
  if $g^x = X$ then return x

Is this a good algorithm? It is

- Correct (always returns the right answer)

Computing Discrete Logs

Let $G = \langle g \rangle$ be a cyclic group of order m with generator $g \in G$.

Input: $X \in G$

Desired Output: $\mathrm{DLog}_{G,g}(X)$

That is, we want x such that $g^x = X$.

for $x = 0, \ldots, m-1$ do
  if $g^x = X$ then return x

Is this a good algorithm? It is

- Correct (always returns the right answer), but
- SLOW!

Run time is $O(m)$ exponentiations, which for $G = \mathbb{Z}_p^*$ is $O(p)$, which is exponential time and prohibitive for large p.

Plan

Cyclic groups and discrete logarithms

Finding cyclic groups

Plan

Finding cyclic groups

- Examples of groups
- DL and CDH games
- Choosing/Building groups of the form $\mathbb{Z}_p^*$

Finding Cyclic Groups

Fact 1: Let p be a prime. Then $\mathbb{Z}_p^*$ is cyclic.

Example: $\mathbb{Z}_{11}^*$ is cyclic.

Fact 2: Let G be any group whose order $m = |G|$ is a prime number. Then G is cyclic.

Note: $|\mathbb{Z}_p^*| = p - 1$ is not prime, so Fact 2 doesn't imply Fact 1.

Cyclic groups in cryptography

Cryptography knows two main providers of cyclic groups:

- Multiplicative groups of finite fields: $\mathbb{Z}_p^*$ is the easiest example.
- Elliptic curves over finite fields.

Computing Discrete Logs: Best known algorithms

Group	**Time to find discrete logarithms**
$\mathbb{Z}_p^*$	$e^{1.92(\ln p)^{1/3}(\ln\ln p)^{2/3}}$ (roughly) subexponential time
EC_p	$\sqrt{p} = e^{\ln(p)/2}$ exponential time

Here p is a prime and EC_p represents an elliptic curve group of order p.

In the first case, if the largest factor of $p-1$ is q, there is also a $O(\sqrt{q})$ algorithm to solve discrete log.

In neither case is a polynomial-time algorithm known.

This (apparent, conjectured) computational intractability of the discrete log problem makes it the basis for cryptographic schemes in which breaking the scheme requires a discrete log computation.

Discrete logarithm computation records

In $\mathbb{Z}_p^*$:

$\lvert p \rvert$ **in bits**	**When**
431	2005
530	2007
596	2014
768	2016
795	2019

For elliptic curves, current record seems to be for $|p|$ around 114.

Elliptic curve groups

Elliptic curve groups are commonly used for public-key cryptography now.

The mathematical details are a bit complex.

For now, think of an elliptic curve group as a cyclic group.

This means it has a generator, a group operation (typically written as $+$), an order, and one can define the analogue of discrete logarithm in this group.

The structure of elliptic curve groups does not seem to permit the same types of subexponential-time discrete logarithm algorithms as $\mathbb{Z}_p^*$.

Why Elliptic curve (EC) groups?

Say we want 80-bit security, meaning discrete log computation by the best known algorithm should take time 2^{80}. Then

- If we work in $\mathbb{Z}_p^*$ (p a prime) we need to set $|\mathbb{Z}_p^*| = p - 1 \approx 2^{1024}$
- But if we work on an elliptic curve group of prime order p then it suffices to set $p \approx 2^{160}$.

This is because

$$e^{1.92(\ln 2^{1024})^{1/3}(\ln\ln 2^{1024})^{2/3}} \approx \sqrt{2^{160}} = 2^{80}$$

But now:

Group Size	Cost of Exponentiation
2^{160}	$T \approx 160^3$
2^{1024}	$1024^3 \approx 260\,T$

Exponentiation will be 260 times faster in the smaller group.

Moore's law and discrete log hardness

If Moore's law holds, the computational power of (your preferred opponent) doubles every 1.5 years.

If you were to adapt your group size for DLOG as a function of time, you would make sure that:

$$\begin{pmatrix} \text{time it takes} \\ \text{to solve } \mathrm{DLog}_G \end{pmatrix} \geq \begin{pmatrix} \text{some wide} \\ \text{security margin} \end{pmatrix} \times \text{base value} \times 2^{\text{year}/1.5}.$$

- if the time it takes is $e^{(\ln p)/2}$, then $\ln p$ would grow linearly with time.
- if the time it takes is $e^{1.92(\ln p)^{1/3}(\ln \ln p)^{2/3}}$, then $\ln p$ would grow as a cubic function of time.

Plan

DL Formally

Let $G = \langle g \rangle$ be a cyclic group of order m.

Game $\mathrm{DL}_{G,g}$

procedure Initialize
$x \xleftarrow{\$} \mathbb{Z}_m ; X \leftarrow g^x$
return X

procedure Finalize(x')
return $(x = x')$

Definition (dl-advantage $\mathbf{Adv}^{\mathrm{dl}}$)

The dl-advantage of an adversary A is

$$\mathbf{Adv}^{\mathrm{dl}}_{G,g}(A) = \Pr\left[\mathrm{DL}^{A}_{G,g} \Rightarrow \mathsf{true}\right]$$

CDH: The Computational Diffie-Hellman Problem

Let $G = \langle g \rangle$ be a cyclic group of order m with generator $g \in G$. The CDH problem is:

Input: $X = g^x \in G$ and $Y = g^y \in G$

Desired Output: $g^{xy} \in G$

This underlies security of the DH Secret Key Exchange Protocol.

Obvious algorithm: $x \leftarrow \mathrm{DLog}_{G,g}(X)$; Return Y^x.

So if one can compute discrete logarithms
then one can solve the CDH problem.

The converse is an open question: are CDH and DL equivalent?

Should they not be equivalent, there would be a way to quickly solve CDH that avoids computing discrete logarithms. But no such way is known.

CDH Formally

Let $G = \langle g \rangle$ be a cyclic group of order m.

Game $\mathrm{CDH}_{G,g}$

procedure Initialize
$x, y \xleftarrow{\$} \mathbb{Z}_m$
$X \leftarrow g^x; Y \leftarrow g^y$
return X, Y

procedure Finalize(Z)
return $(Z = g^{xy})$

Definition (cdh-advantage $\mathbf{Adv}^{\mathrm{cdh}}$)

The cdh-advantage of an adversary A is

$$\mathbf{Adv}^{\mathrm{cdh}}_{G,g}(A) = \Pr\left[\mathrm{CDH}^{A}_{G,g} \Rightarrow \mathsf{true}\right]$$

Plan

Finding cyclic groups

- Examples of groups
- DL and CDH games
- Choosing/Building groups of the form $\mathbb{Z}_p^*$

Building cyclic groups

We will need to build (large) groups over which our cryptographic schemes can work, and find generators in these groups.

How do we do this efficiently?

Building cyclic groups

To find a suitable prime p and generator g of $\mathbb{Z}_p^*$:

- Pick numbers p at random until p is a prime of the desired form
- Pick elements g from $\mathbb{Z}_p^*$ at random until g is a generator

For this to work we need to know

- How to test if p is prime
- How many numbers in a given range are primes of the desired form
- How to test if g is a generator of $\mathbb{Z}_p^*$ when p is prime
- How many elements of $\mathbb{Z}_p^*$ are generators

Finding primes

Desired: An efficient algorithm that given an integer k returns a prime $p \in \{2^{k-1}, \ldots, 2^k - 1\}$ such that $q = (p-1)/2$ is also prime.

Alg Findprime(k)
do
  $p \xleftarrow{\$} \{2^{k-1}, \ldots, 2^k - 1\}$
until (p is prime and $(p-1)/2$ is prime)
return p

- How do we test primality?
- How many iterations do we need to succeed?

Primality Testing

Given: integer N
Output: TRUE if N is prime, FALSE otherwise.

for $i = 2, \ldots, \lceil \sqrt{N} \rceil$ do
  if $N \bmod i = 0$ then return false
return true

Primality Testing

Given: integer N
Output: TRUE if N is prime, FALSE otherwise.

for $i = 2, \ldots, \lceil \sqrt{N} \rceil$ do
  if $N \bmod i = 0$ then return false
return true

Correct but SLOW! $O(\sqrt{N})$ running time, exponential in $|N|$.

However, we have polynomial time algorithms, which is much better:

- $O(|N|^3)$ time randomized algorithms
- Even a $O(|N|^8)$ time deterministic algorithm

Finding cryptographic size prime numbers is not a difficult problem.
It's even less of a problem when it only has to be done once.

Density of primes

Let $\pi(N)$ be the number of primes in the range $1, \ldots, N$. So if $p \xleftarrow{\$} \{1, \ldots, N\}$ then

$$\Pr[p \text{ is a prime}] = \frac{\pi(N)}{N}$$

Fact: $\pi(N) \sim \dfrac{N}{\ln(N)}$

So

$$\Pr[p \text{ is a prime}] \sim \frac{1}{\ln(N)}$$

If $N = 2^{1024}$ this is about $0.001488 \approx 1/700$.

So the number of iterations taken by our algorithm to find a prime is not too big.

Recall DH Secret Key Exchange

The following are assumed to be public: A large prime p and a generator g of $\mathbb{Z}_p^*$.

Alice		**Bob**
$x \xleftarrow{\$} \mathbb{Z}_{p-1}$; $X \leftarrow g^x \bmod p$		
	$\xrightarrow{X}$	
		$y \xleftarrow{\$} \mathbb{Z}_{p-1}$; $Y \leftarrow g^y \bmod p$
	$\xleftarrow{Y}$	
$K_A \leftarrow Y^x \bmod p$		$K_B \leftarrow X^y \bmod p$

- $Y^x = (g^y)^x = g^{xy} = (g^x)^y = X^y$ modulo p, so $K_A = K_B$
- Adversary is faced with the CDH problem.

DH Secret Key Exchange: Questions

- How do we pick a large prime p, and how large is large enough?
- What does it mean for g to be a generator modulo p?
- How do we find a generator modulo p?
- How can Alice quickly compute $x \mapsto g^x \bmod p$?
- How can Bob quickly compute $y \mapsto g^y \bmod p$?
- Why is it hard to compute $(g^x \bmod p, g^y \bmod p) \mapsto g^{xy} \bmod p$?
- ...

The slides have sketched the answers to many of these questions.