Modular Exponentiation:

We will be given an integer N (modulus)

" " " " 'a' $\in \mathbb{Z}_N$ (base).

" " " " 'b', where b is an integer. (exponent).

We have to compute

$$a^b \bmod N.$$

- if $b = 0$, then the problem is trivial.
- if $b < 0$ $\quad a^b \bmod N = (\bar{a}^1)^{-b} \bmod N$

$$= x^{-b} \bmod N, \text{ where } x = \bar{a}^1 \bmod N$$

We design algorithms to compute $a^b \bmod N$ when $a \in \mathbb{Z}_N$ and $b > 0$ is an integer.

Trivial Algorithm:

Multiply a b many times and then take the modulo of a^b with respect to N.

The size of the intermediate result is $b \cdot \log a = \underbrace{O(b \cdot \|a\|)}$.

The alg for computing the modulo will read the bit string of length $O(b \cdot \|a\|)$, which is exponential to the bit size of b.

Algorithm:

Input: Modulus N, $a \in \mathbb{Z}_N$ and $b \in \mathbb{Z}_{>0}$

output: $a^b \bmod N$

```
x = 1
for i = 1 to b
    x = x · a mod N
return x.
```

Previous alg.

```
x = 1
for i = 1 to b.
    x = x · a
return x mod N.
```

Time complexity is exponential to the length of b.

Recurrence:

$$a^b \bmod N = a \cdot a^{b-1} \bmod N = a \cdot a \cdot a^{b-2} \bmod N = \cdots =$$

$$a^5 \bmod N = a \cdot a^4 \bmod N$$

$$\downarrow$$

$$a^2 \bmod N$$

$$\downarrow$$

$$a \bmod N.$$

```
ModExp(a, N, b)
if b = 1 return a
  a · Modexp(a, N, b-1)
```

Modified Recurrence relation:

$$a^b \bmod N = \begin{cases} (a^{b/2})^2 \bmod N & \text{if } b \text{ is even} \\ a \cdot (a^{\frac{b-1}{2}})^2 \bmod N & \text{if } b \text{ is odd.} \end{cases}$$

"Square and Multiplication" algorithm.

ModExp(a, b, N) → $a^b \bmod N$

if $b = 1$ return a

else

if b is even

$t \leftarrow$ ModExp(a, b/2, N)

return $t^2 \bmod N$

if b is odd

$t \leftarrow$ ModExp$(a, \frac{b-1}{2}, N)$.

return $a \cdot t^2 \bmod N$.

$a^5 \bmod N$

ModExp(a, 5, N)

a^2 ModExp(a, 2, N)

ModExp(a, 1, N)

Conclusion:

Let N be a fixed integer and $a \in \mathbb{Z}_N$

$$f_{a,N} : \mathbb{Z} \to \mathbb{Z}_N$$

$$f_{a,N}(b) = a^b \bmod N$$

* Discrete logarithm problem is hard to compute"

→ we do have a poly time algo for computing this fn

" Square and Multiplication"

Given a, N and $a^b \bmod N$, compute b.

$$b = \log_a a^b \bmod N$$

(Discrete logarithm problem)

Choosing a random Group element:

Algorithm:

Input: A description of a group G, a length parameter l, and a parameter t

$l \rightarrow$ the length of the elements in the group G in terms of bits.

Output: A random element of G.

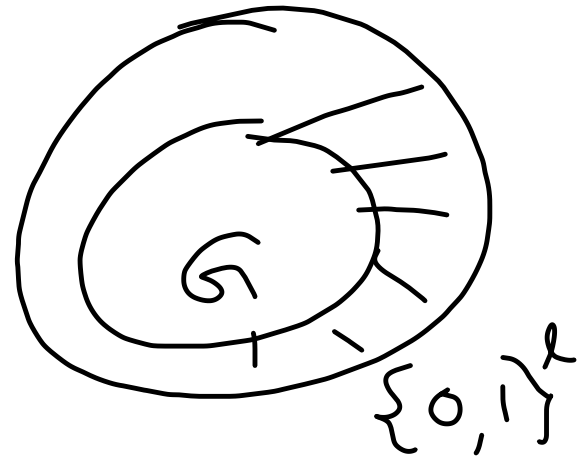
$|G| = 30, \quad G \subset \{0,1\}^5$

for $i = 1$ to t

$x \leftarrow \$ \{0,1\}^\ell$

if $x \in G$, return x

return "fail"



Membership test that should be done in polynomial time.

What is the prob. that the alg. fails.

At every iteration, the alg. fails with prob. $\frac{2^\ell - |G|}{2^\ell} = \left(1 - \frac{|G|}{2^\ell}\right)$

The alg. return "fail" with prob. $\left(1 - \frac{|G|}{2^\ell}\right)^t$ → failure probability.

If Fail denotes the alg. outputs fail
then for any element $g \in G$.

$$\Pr[\text{output of the alg. is } g \mid \overline{\text{Fail}}] = \frac{1}{|G|}.$$

Tradeoff:

$\uparrow t \longrightarrow \downarrow$ failure prob.

If Fail denotes the alg. outputs fail
then for any element $g \in G$.

$$P[\text{output of the alg. is } g \mid \overline{\text{Fail}}] = \frac{1}{|G|}.$$

Tradeoff:

$\uparrow t \longrightarrow \downarrow$ failure prob.

$\searrow \uparrow$ running time of the alg.

"For cryptographic application, we need alg. where the worst case running time is poly and the failure prob. is negligible"

If Fail denotes the alg. outputs fail
then for any element $g \in G$.

$$P[\text{output of the alg. is } g \mid \overline{\text{Fail}}] = \frac{1}{|G|}.$$

Tradeoff:

$\uparrow t \longrightarrow \downarrow$ failure prob.

$\searrow \uparrow$ running time of the alg.

"For cryptographic application, we need alg. where the worst case running time is poly and the failure prob. is negligible"

To achieve alg. of this sort, we require to fulfil the following two conditions:

① It should be possible to determine in polynomial time whether a $l(n)$ bit string $x \in G_2$.

$n \to$ Security parameter.

② the prob. that a random $l(n)$ bit string is an element of G_2 should be at least $1/\text{poly}(n)$.

To achieve alg. of this sort, we require to fulfil the following two conditions:

① It should be possible to determine in polynomial time whether a $l(n)$ bit string $x \in G$.

$n \to$ Security parameter.

② the prob. that a random $l(n)$ bit string is an element of G should be at least $1/\text{poly}(n)$.

The failure prob. is

$$\left(1 - \frac{|G|}{2^{\ell(n)}}\right)^{t(n)} \leq \left(1 - \frac{1}{poly(n)}\right)^{t(n)}$$

$$\frac{|G|}{2^{\ell(n)}} \geq \frac{1}{poly(n)}$$

Now you put $t(n) = n \cdot poly(n)$

$$= \left(1 - \frac{1}{poly(n)}\right)^{n\,poly(n)} \leq e^{-n}$$

Examples of some group:

① $\mathbb{Z}_N$, let $n = \|N\|$.

The failure prob. is

$$\left(1 - \frac{|G|}{2^{\ell(n)}}\right)^{t(n)} \leq \left(1 - \frac{1}{poly(n)}\right)^{t(n)}$$

$$\frac{|G|}{2^{\ell(n)}} \geq \frac{1}{poly(n)}$$

Now you put $t(n) = n \cdot poly(n)$

$$= \left(1 - \frac{1}{poly(n)}\right)^{n\,poly(n)} \leq e^{-n}$$

Examples of some group:

① $\mathbb{Z}_N$, let $n = \|N\|$.

$$\frac{|\mathbb{Z}_N|}{2^n} = \frac{N}{2^n} > \frac{2^{n-1}}{2^n} = \frac{1}{2}$$

$$\mathbb{Z}_N \subset \{0,1\}^n$$

$$N > 2^{n-1}$$

$$\frac{|\mathbb{Z}_N|}{2^n} > \frac{1}{2}.$$

$$2^{n-1} < N \leq 2^n$$

Example: Consider the group $\mathbb{Z}_N^*$, $N = \|N\|$

$$\frac{|\mathbb{Z}_N^*|}{2^n} = \frac{\phi(N)}{2^n} = \frac{\phi(N)}{N} \times \frac{N}{2^n}$$

Prop: For any $N \geq 3$ of length n,

$$\frac{\phi(N)}{N} > \frac{1}{2n}$$

$$\frac{|\mathbb{Z}_{4N}^*|}{2n} = \frac{\phi(N)}{2^n} = \underbrace{\frac{\phi(N)}{N}}_{\geq \frac{1}{2n}} \times \underbrace{\frac{N}{2^n}}_{> \frac{1}{2}} = \frac{1}{4n}.$$

Prop: For any $N \geq 3$ of length n,

$$\frac{\phi(N)}{N} > \frac{1}{2n}$$