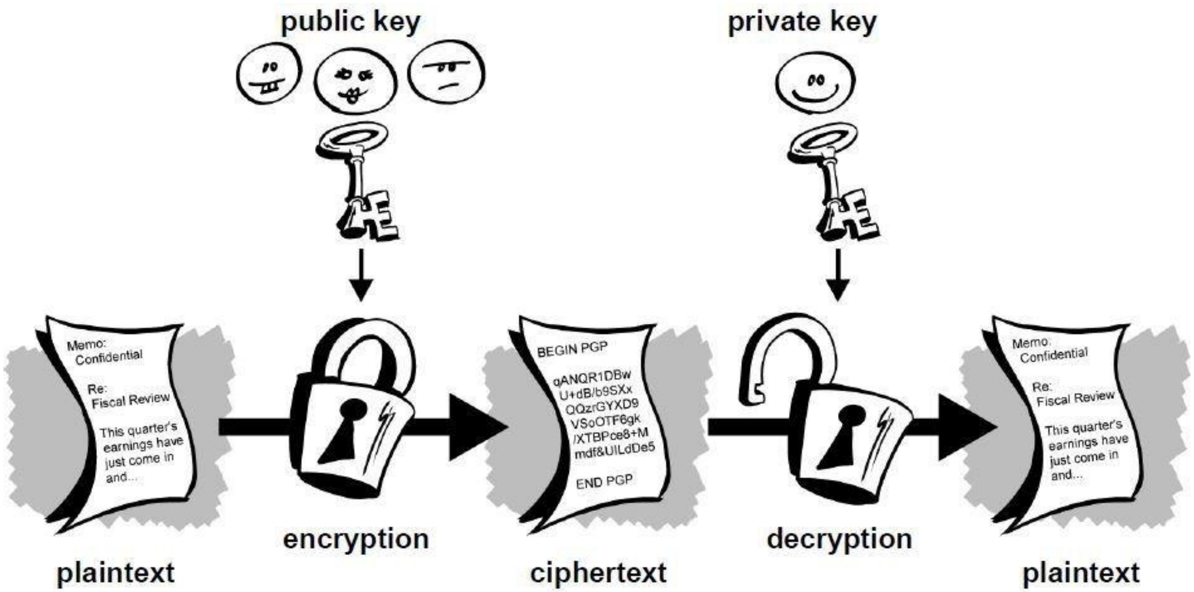
Introduction to Cryptography

By

Vipul Goyal



Modular Arithmetic and Hard Problems

Modular Arithmetic

Sometimes in arithmetic we "*work mod N*".

E.g., on a clock, the hours go mod 12.

"A and B are equivalent mod N",

" $A \equiv_N B$ ",

means A, B have same remainder mod N.

$2 \bmod 9 = 11 \bmod 9 \quad \Rightarrow 2 \equiv_9 11$

$11 \bmod 9 \neq 21 \bmod 9$

Modular Arithmetic

Keep in mind:

mod N, every integer is equivalent to exactly one of $0, 1, 2, 3, \ldots, N-1$.

Addition mod N

Addition, +, "plays nice" mod N:

$$A \equiv_N B$$

$$A' \equiv_N B'$$

$$\Rightarrow \quad A+A' \equiv_N B+B'$$

Addition mod 5

+	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

Subtraction mod N

"*What about* subtraction *mod N?*", you might ask

Given B, we define "$-B$" to be

"the positive number less than N such that $B + (-B) = 0$"

Note that $-B = N - B$

Negatives mod 5

+	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

$-2 = 3$

(the number we need to add to 2 to make it 0)

Similarly

$-4 = 1$

$-0 = 0$

Multiplication mod N

Multiplication, •, also "plays nice" mod N:

$$A \equiv_N B$$

$$A' \equiv_N B'$$

$$\Rightarrow \quad A \bullet A' \equiv_N B \bullet B'$$

Multiplication mod 5

•	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

Division mod N

"*What about* division *mod N?*", you might say

So given B, can we define "B^{-1}" to be
"the number less than N such that $B \cdot B^{-1} = 1$"?

Yes, but this is more tricky!

Inverses mod 6

•	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	1	2	3	4	5
2	0	2	4	0	2	4
3	0	3	0	3	0	3
4	0	4	2	0	4	2
5	0	5	4	3	2	1

$0^{-1} =$ undefined

$1^{-1} =$ 1

$2^{-1} =$ undefined!

$3^{-1} =$ undefined!

$4^{-1} =$ undefined!

$5^{-1} =$ 5

Huh. We only have two #'s with inverses

When does A have an inverse mod N?

GCD(A,N) = 1

Such A, N called “relatively prime”

Note: mod a prime, **all** nonzeros have inverses

Great! Now we can do all 4 basic operations under modular Arithmetic

Good practice: always work mod a prime unless necessary!!

Why Modular Arithmetic?

Friendly to large numbers: We can do arbitrary operations on very large numbers. Still the result will be at most N.

- **Exponentiation:** perhaps the most interesting example
- Normally: computing 2^x is very hard for large x. Say x = 100, even writing down 2^x is hard.
- But **can** compute and write $2^x \bmod N$ very efficiency!

Facts we will use

$$g^a . g^b = g^{a+b}$$

$$(g^a)^2 = g^{2a}$$

(also for modular arithmetic)

Modular Exponentiation

Example: Compute $2337^{32} \bmod 100$.

By hand.

Bad idea: $2337 \times 2337 = 5461569$

$2337 \times 5461569 = 12763686753$

$2337 \times 12763686753 = \cdots$

(30 more multiplications later…)

$= 62672756515215551165318888666866858831347582423666560739675500890577014623663553722821669603097061282892288\mathbf{1}$

Modular Exponentiation

Example: Compute $2337^{32} \bmod 100$.

By hand.

Smart idea 1:

Reduce mod 100 at every step.

$$37 \times 37 = 1369$$

$$37 \times 69 = 2553$$

Still need 32 multiplications, but smaller numbers

Modular Exponentiation

Smart idea 2:

Don’t multiply 32 times; square 5 times.

$$2337^{1} \rightarrow 2337^{2} \rightarrow 2337^{4} \rightarrow 2337^{8} \rightarrow 2337^{16} \rightarrow 2337^{32}$$

Lucky (?) that exponent was a power of 2.

Q: What if we had wanted 2337^{34} ?

A: Multiply together 2337^{32} and 2337^{2}.

Modular Exponentiation

Smart idea 2:

Don't multiply 32 times; square 5 times.

$$2337^1 \rightarrow 2337^2 \rightarrow 2337^4 \rightarrow 2337^8 \rightarrow 2337^{16} \rightarrow 2337^{32}$$

Lucky (?) that exponent was a power of 2.

Q: What if we had wanted 2337^{53} ?

A: Multiply powers: 32 + 16 + 4 + 1

Here I used that binary rep. of 53 is 110101

Modular Exponentiation

In general, to compute $g^x \bmod N$,
where g, x, N are $\leq n$ bits long:

1. Repeatedly square g, always mod N.
   Do this n times. (save all values)

2. Multiply together the powers of g
   corresponding to binary digits of x
   (again, always mod N).

A Simple Example

Need to compute $g^x \bmod N$, $g = 17$, $x = 38$, $N = 10$

1) Compute repeated squares of g mod N. We get

   $17 \bmod 10 = 7$

   $7^2 \bmod 10 = 9 = g^2 \bmod N$

   $9^2 \bmod 10 = 1 \bmod 10 = g^4 \bmod N$

   $1^2 \bmod 10 = 1 \bmod 10 = g^{16} \bmod N$

   $1^2 \bmod 10 = 1 \bmod 10 = g^{32} \bmod N$

2) Write x as powers of 2:

   $38 = 32 + 4 + 2$

3) Compute $(g^{32} \bmod N)(g^4 \bmod N)(g^2 \bmod N) = g^{38} \bmod N = 9$

Reversing Modular Exponentiation

You are given g, N and g^x mod N,

can you find x?

1. One option is, given g, N, compute g^x mod N for every number x. See which one matches. Too slow!

2. Can you do much faster? Answer is: people have tried for hundreds of years but failed!

3. Normally, given g^x, you can take its log base g to find x. But that doesn't work if you are given only g^x mod N.

Hard Problem 1: Discrete Log Problem

Discrete Log problem (DLP):

given g, N and y, output x s.t. $y = g^x \bmod N$

DLP considered to hard (for carefully chosen g, N)

Example: hard for even a supercomputer to solve the following problem

- Given g = 773, N = 62672756515…..242366667 (100 digits prime)
- Find x such that $g^x \bmod N$ = 427389323….2334739847 (100 digits)

Discrete Log Problem

We will see how use DLP to build:

- Public-key encryption
- Private-key encryption (with reusable short key)
- Digital Signatures

(Most useful hard problem in cryptography. Even more so than factoring.)

Questions and Discussion?