Introduction to Cryptography

PCMI 2022 - Undergraduate Summer School

Recall

We write $f << g$ if $\left|\frac{f}{g}\right|$ is bounded

as $k \to \infty$

- f grows polynomially if $\exists\, a, b > 0$ with

$$k^a << f << k^b$$

- f grows exponentially if $\exists\, a, b > 0$ with

$$2^{ak} << f << 2^{bk}$$

- f grows subexponentially if $\forall a,b>0$

$$k^a \ll f \ll 2^{bk}$$

Example: $f(k) = 2^{\sqrt{k}}$

Definition

An algorithm is <u>fast</u> if the number of steps, as a function of the size of the input, grows <u>polynomially</u>.

A problem is <u>easy</u> if the fastest known alg to solve it is <u>fast</u>.

Similarly

An alg is slow if # of steps grows exponentially

Problem is hard if the best known alg to solve it is slow.

What if # of steps grows subexponentially?

still (kind of) hard.

Recall the DLP

Given $G = \langle g \rangle$, $h \in G$, find x with

$0 \leq x < \# G$ such that

$$h = g^x$$

(think: $x = \log_g h$)

Depending on the specific group G, this problem <u>can</u> be hard.

Today: Assume we do have a cyclic group G such that

- multiplication and inversion in G is fast

  $(g,h) \mapsto gh$ $\qquad$ $g \mapsto g^{-1}$

- but the DLP is hard

Then: $g, x \mapsto h = g^x$ fast but $g, h \mapsto x = \log_g h$ slow

Our set up

$$A \longrightarrow B$$

1. B has to generate keys to receive messages
2. A can encrypt a message
3. B can decrypt the message

Elgamal key generation

B chooses G with known generator g

1. B generates a random secret number x
   (this is the secret key)

2. B computes $h = g^x$
   the public key is (G, g, h)

Elgamal encryption

Suppose that A wants to send a message $m \in G$ to B.

1. A generates a secret random number y.

2. A computes 2 ciphertexts

$$c_1 = g^y \qquad c_2 = m \cdot h^y$$

$$(g^y)^x = h^y$$

3. y is thrown out, (c_1, c_2) made public

Elgamal decryption

When B receives c_1 and c_2

B computes

$$c_1^{-x} \cdot c_2 = m.$$

$$c_1^{-x} = (c_1^{x})^{-1}$$

$c_1 = g^y$, $c_2 = m \cdot h^y$

$h = g^x$

Attacking Elgamal

- Certainly, solving the DLP is enough to break the encryption

- Actually "less" is necessary: It suffices to solve the Diffie-Hellman problem

Given $G = \langle g \rangle$, g, g^x, g^y

compute g^{xy}

Because we don't have a better way to solve DHP, we will try to solve the DLP.

To compute the "speed" of our solution, need to know the size of the input.

Input: group G

Size of the input $\quad k = $ size of $\#G$

number of digits / bits in the number $\#G$

$$k \approx \log(\#G)$$

Baby steps, giant steps due to Shanks

Best for generic group

That’s all for now!