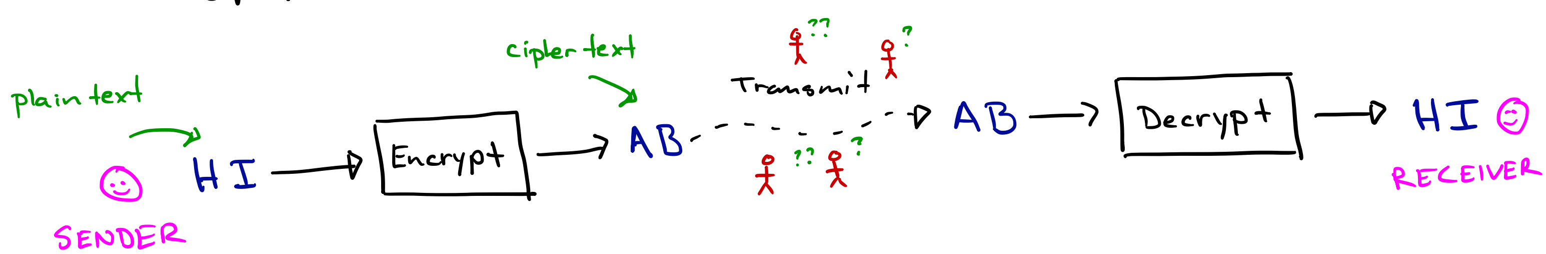
Section-Cryptography

Background

Suppose you want to have a private conversation with someone, but it is likely that your messages will be seen or heard by others. What to do?



So, how does one securely encrypt and decrypt? This is a problem studied for AGES!

For a good read on a bit of the history of cryptography, try "The Code Book" by Simon Singh.

<u>Def</u> Algorithms for encryption & decryption are called <u>ciphers</u>.

Keys and Key Exchange

Important fact: usually assume that <u>everyone</u> knows which cipher is being used. However most ciphers are based on a secret key that can be changed. An encrypted message should be "hard" to decrypt without the key. Both the sender and receiver will need the key, but in fact, it's possible to have different keys for encryption and decryption.

<u>Option 1</u>: Distribute keys physically, e.g. WW II with Enigma machines

<u>Option 2</u>: Public key encryption (private decryption key)

↩ Huge Revolution

Public Key Encryption (PKE)

- "Discovered" in 1976 by Diffie and Hellman (and Merkle)
  - known as Diffie–Hellman Key Exchange (DHKE)
  - predated by classified work of Ellis–Cocks–Williams GCHQ (Gov. Com. HQ – GB) 1969
  - based on discrete-log. problem.
- RSA Encryption (and key exchange)
  - developed by Rivest–Shamir–Adleman 1978
  - known to Cocks GCHQ 1973
  - based on problem of factoring large numbers

$\boxed{\text{Many Others}}$

- ElGamal encryption (and key exchange)
  - developed by Taher Elgamal 1985
  - can be used to send any message, but in

practice it's used to send the key for a (faster, symmetric) cipher.

— based on DHKE

Elgamal Encryption

For simplicity, our message will just be a number m.

Let's keep track of who knows what.

Sender	Public	Receiver
plain text → m, b ← encrypt key	$p, g, g^a, g^b,$ $\boxed{m \cdot (g^a)^b}$ ← cypher text (c)	a ← decrypt key

Step 1: $\boxed{\text{Setup}}$

- choose a prime p — this will be the modulus for some computations (need $p > m$)
- Choose a number g with $1 < g < p$.
- Make both public.

Step 2: $\boxed{\text{Generate Decrypt Key}}$

- receiver picks a random number a — this is the decryption key
- Receiver computes $g^a \pmod p$ and makes public.

Step 3: $\boxed{\text{Generate Encrypt Key}}$

- sender picks a random num b — the encryption key
- Sender computes $g^b \pmod p$ and makes public

Step 4: Encrypt + Send Message

- Sender computes $c = m \cdot (g^a)^b \pmod p$ — c is the cypher text.
- Sender sends c

Step 5: Decrypt — how?

cypher text: $c = m \cdot (g^a)^b = m \cdot g^{ab}$

public: p, g, g^a, g^b

Idea need to solve $c \equiv m g^{ab} \pmod p$ for m.

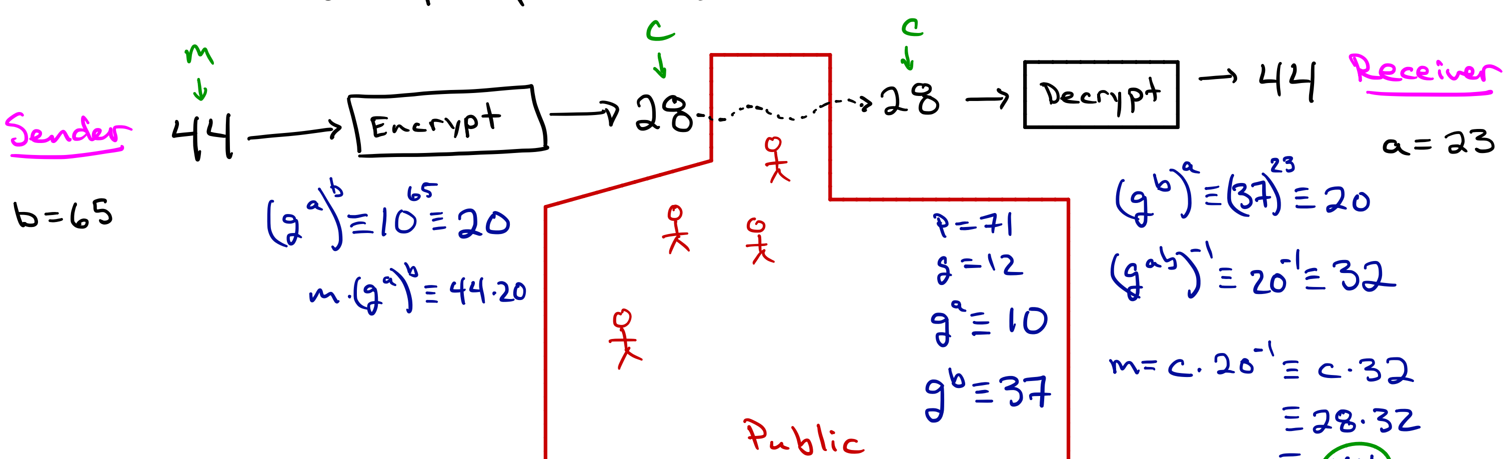
Receiver knows a!!

- Receiver computes $(g^b)^a = g^{ab} \pmod p$
- Receiver solves $g^{ab} \cdot x \equiv 1 \pmod p$ to find $(g^{ab})^{-1}$
- Receiver computes $c(g^{ab})^{-1} \equiv m g^{ab} g^{ab^{-1}} \equiv \boxed{m}$

message recovered.

Example

Setup: $p = 71$, $g = 12$



$b = 65$

$(g^a)^b \equiv 10^{65} \equiv 20$

$m \cdot (g^a)^b \equiv 44 \cdot 20$

$a = 23$

$(g^b)^a \equiv (37)^{23} \equiv 20$

$(g^{ab})^{-1} \equiv 20^{-1} \equiv 32$

$m = c \cdot 20^{-1} \equiv c \cdot 32$

$\equiv 28 \cdot 32$

$\equiv 44$

m — yay!

Final Notes on ElGamal:

- For security, don't just need p to be big, but really need k to be big where k is smallest positive integer such that
$$g^k \equiv 1 \pmod{p}.$$
(Recall that $g^{p-1} \equiv 1 \pmod{p}$ so $k \leq p-1$.)

- One way to ensure k is big is to choose p such that p is really big and $q = \frac{p-1}{2}$ is prime (i.e. p is a safe prime). Then q is also really big. And, for all $1 < g < p-1$, the associated k is either q or $2q$ — hence big enough.

Example

You are communcating with a friend using ElGamal encryption with $\boxed{p = 83 \text{ and } g = 7}$. Your friend wants to send you a message so you select a decryption key $a = \boxed{14}$ and send your friend the number $g^a = 7^{14} \equiv \boxed{40} \pmod{83}$.

A moment later your friend sends you the number $g^b \equiv \boxed{48}$, and then sends the encrypted message $c = \boxed{76}$. Decrypt the message. (You can use Wolfram Alpha.)

[ans. 55]

RSA ... maybe later.