Class notes

Cryptography

$$\boxed{\text{plaintext}} \underset{\text{encryption}}{\overset{\text{decryption}}{\rightleftarrows}} \boxed{\text{ciphertext}}$$

enemy intercepts & "guesses" secret key

$$\boxed{\text{plaintext}} \xrightarrow[\text{encryption}]{\text{(secret) key}} \boxed{\text{ciphertext}} \xRightarrow[\text{Transmission}]{} \boxed{\text{ciphertext}} \xrightarrow[\text{decryption}]{\text{(secret) key}} \boxed{\text{plaintext}}$$

Read Section 10.1 (1st half) for a discussion some old, & simple cryptographic techniques.

We are interested in public-key cryptography where $\exists$ two keys — encryption key & decryption key ("inverses" of each other). Encryption key is public but decryption key is only with the receiver — so everyone (consumers) can encrypt messages (cc numbers) to receiver (Bank) but only the recipient can decipher them.

In 1977, Rivest, Shamir, & Adelman (RSA) proposed a public-key cryptosystem using elementary ideas from NT. (still used in OpenSSH protocol)

Create a trapdoor or one-way function on a set X, $E: X \to X$, that is invertible & the receiver can easily compute E^{-1} but difficult for others.

~~Let~~ Say $\mathbb{Z}_n = \{0, 1, \ldots, n-1\}$, i.e., the message has been converted to an integer in $\mathbb{Z}_n$

(If the msg. is too long then break it into blocks)

Setup for RSA

Step 1 Pick two large primes p & q
& let $n = pq$

Step 2 Easily compute $\phi(n) = \phi(p)\,\phi(q) = (p-1)(q-1)$

usually small → **Step 3** Choose integer e with $1 < e < \phi(n)$
and $\gcd(e, \phi(n)) = 1$

Step 4 Solve $ex \equiv 1 \pmod{\phi(n)}$, i.e. find the multiplicative inverse of e modulo $\phi(n)$, call it d. (Use EA or some such)

Step 5 Define the function $E : \mathbb{Z}_n \to \mathbb{Z}_n$
by $E(x) = x^e \pmod{n}$
(Easy to compute by repeated squaring)

Recipients public key is (n, e) & any one can apply $E(x)$ to encrypt their message & send it to the recipient.

Only ~~[illegible]~~ the recipient knows d s.t.
$ed \equiv 1 \pmod{\phi(n)}$
& ~~[illegible]~~ the recipient decrypts the received message by $E^{-1}(y) = y^d$.

Claim: $(x^e)^d \equiv x \pmod{n}$.

<u>Theorem</u> (Decryption key)

Let $n = p_1 p_2 \cdots p_k$, product of distinct primes.

Let $d, e \in \mathbb{Z}^+$ s.t. $de \equiv 1 \pmod{\phi(n)}$

Then $a^{de} \equiv a \pmod{n}$ for all $a \in \mathbb{Z}$.

<u>Proof</u>

Since $n \mid a^{de} - a$ iff $p_i \mid a^{de} - a$ for each $i = 1, \ldots, k$

It is enough to show $a^{de} \equiv a \pmod{p_i}$ for any $p = p_i$

If $\gcd(a, p_i) \neq 1$, then $a \equiv 0 \pmod{p_i}$ so $a^{de} \equiv a \equiv 0 \pmod{p_i}$

If $\gcd(a, p_i) = 1$, then $a^{p_i - 1} \equiv 1 \pmod{p_i}$ (Fermat)

Since $\underset{(p_1-1)(p_2-1)\cdots(p_k-1)}{\underset{\parallel}{\phi(n)}} \mid de - 1$, $p - 1 \mid de - 1$.

This implies $a^{de-1} \equiv 1 \pmod{p}$

Multiplying by a, $a^{de} \equiv a \pmod{p}$.

So RSA is secure as long as its difficult to find d given the public key (n, e).

To find d we need $\phi(n)$, but the only way to do that is by finding p & q s.t. $n = pq$ & then $\phi(n) = (p-1)(q-1)$.

Typically, this prime factorization is very difficult (computationally).

<u>Careful</u> — p & q "close" to each other, then we can apply Fermat's factorization method.

e.g. 1. Choose p & q : $p=17$ $q=19$ so $n=pq=323$

2. Compute $\varphi(n)$: $\varphi(n)=(p-1)(q-1)=288$

3. Choose $e<288$ & $\gcd(e,288)=1$; Let $e=95$

4. Solve $95x\equiv 1 \pmod{288}$ to get $d=191$

Public key is $(323, 95)$

Encryption ftn. is $E(x)=x^{95} \pmod{323}$

& the Decryption ftn is $D(x)=x^{191} \pmod{323}$

e.g. "X" is encoded as 24

so, $E(24)=24^{95}\equiv 294 \pmod{323}$

& $E^{-1}(294)=(294)^{191}\equiv 24 \pmod{323}$

ElGamal public-key cryptosystem

When we are working with the real numbers $\log_b y$ is the value x, such that $b^x = y$.

We can define an analogous discrete logarithm.
Given integers b & n, with $b < n$, the disc. log. of an integer y to the base b is an integer x, s.t.

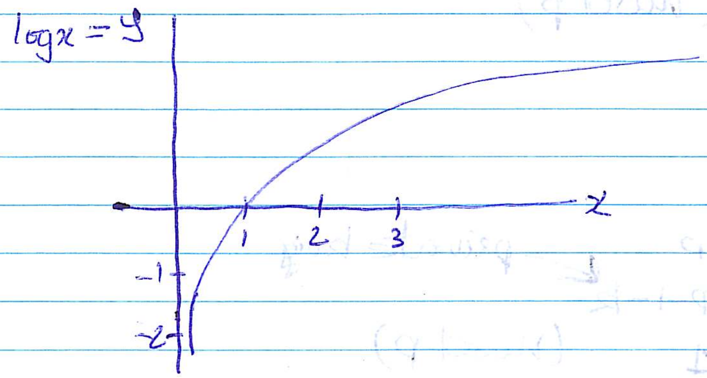
$$b^x \equiv y \pmod{n}$$

(Think of n as prime & b as primitive root of n)

written as

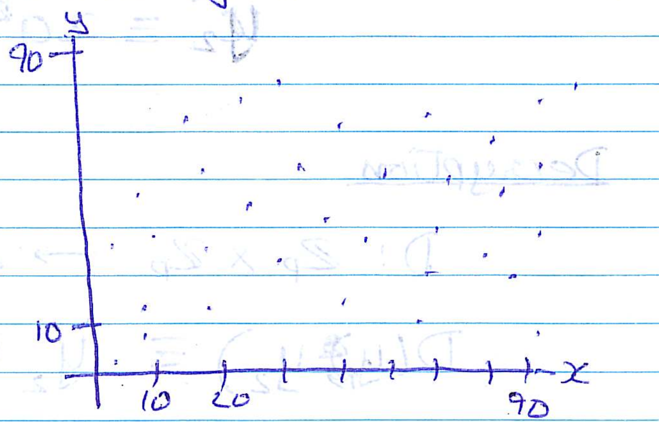
$$x = \mathrm{ind}_{b,n}\, y \quad \text{(index)}.$$

While it is quite efficient to raise numbers to large powers modulo p (repeated squaring algo), the inverse computation of discrete log is much harder.



log in reals

"continuous"



discrete log modulo 97

"random like"

The ElGamal cryptosystem relies on the intractability of the discrete log.

60

$\mathbb{Z}_p = \{0, 1, \ldots, p-1\}$

ElGamal ~~Encryption~~ Public-key & private-key generation

1. Pick a (large) prime p & r, a primitive root of p.
2. Pick an integer k, $2 \le k \le p-2$ (secret key)
3. Calculate $a \equiv r^k \pmod{p}$

Public-key is (p, r, a)

Private-key is k (note this is the discrete log of a to the base r modulo p)

Encryption

$E : \mathbb{Z}_p \to \mathbb{Z}_p \times \mathbb{Z}_p$

1. Pick any k', $2 \le k' \le p-2$ & note the public key (p, r, a)
2. $E(x) = (y_1, y_2)$

where $y_1 \equiv r^{k'} \pmod{p}$

$y_2 \equiv x a^{k'} \pmod{p}$

Decryption

$D : \mathbb{Z}_p \times \mathbb{Z}_p \to \mathbb{Z}_p$

private key

$D(y_1, y_2) \equiv y_2 y_1^{p-1-k} \pmod{p}$

Claim: $D(E(x)) = x$

$$D(y_1, y_2) \equiv y_2 y_1^{p-1-k} \equiv (x a^{k'})(r^{k'})^{p-1-k}$$
$$\equiv x (r^k)^{k'} (r^{k'(p-1)-k'k})$$
$$\equiv x (r^{k'(p-1)})$$
$$\equiv x (r^{p-1})^{k'}$$
$$\equiv x \pmod{p}, \text{ by Fermat.}$$

e.g. Key generation $p = 2357$ & $r = 2$

Choose $k = 1751$

$$a = r^k = 2^{1751} \equiv 1185 \pmod{2387}$$

public key $= (p = 2357,\ r = 2,\ a = 1185)$

private key $= k = 1751$

Encryption To encrypt $x = 2035$, note (p, r, a) & select $k' = 1520$ (say)

$$y_1 = r^{1520} \equiv 2^{1520} \equiv 1430 \pmod{2357}$$

$$y_2 = x\,a^{1520} \equiv 2035\cdot(1185)^{1520} \equiv 697 \pmod{2357}$$

Send $(y_1, y_2) = (1430, 697)$

Decryption

$$x = 697\,(1430)^{2356-1751}$$
$$\equiv 2035 \pmod{2357}$$

$\boxed{\text{Sender}} \xrightarrow[\text{yes sell stock}]{} \boxed{\text{Receiver}}$ ← customer ← stock broker

A variation of the above system is used for DRM (Digital rights management) & other such applications that involve "digital signatures" — protection against forgeries such as unauthorized copy of music or video files. It should be difficult to tamper with, but its authenticity should be easy to verify.

Given public-key (p, r, a) ← present hidden on your comp. (different from your friend)

& private-key k ← License bought from recording company

Generate a signature $S = (s_1, s_2)$

First pick an integer k', $1 \le k \le \varphi(p) = p-1$ s.t. $\gcd(k', p-1) = 1$.

$$s_1 \equiv r^{k'} \pmod{p}$$ private key

Solve $k' s_2 \equiv x - k s_1 \pmod{p-1}$ using EA, etc.

to get $s_2 = (k')^{-1}(x - k s_1) \pmod{p-1}$

multiplicative inverse of k' modulo $p-1$ (exists since $\gcd(k', p-1) = 1$

Question "Do you want to sell stock ATT?" encrypted using publickey of A

$A \longleftarrow B$

$A \longrightarrow B$

encrypted using publickey of B

Answer "Yes" with digital signature appended that can be verified using public key of A

Now, the receiver uses the sender's public key to to confirm the message.

Check: Calculate $v_1 \equiv a^{s_1} s_1^{s_2} \pmod p$

$v_2 \equiv r^{x} \pmod p$

Signature is legitimate if $v_1 = v_2$.

Claim $v_1 = v_2$

$$v_1 \equiv a^{s_1} s_1^{s_2} \equiv (r^{k})^{s_1} (r^{k'})^{s_2}$$

$$\equiv r^{ks_1 + k's_2} \underset{\text{by defn. of } s_2}{\equiv} r^{x + l(p-1)} \equiv r^{x}(r^{p-1})^{l}$$

$$\underset{\text{by Fermat}}{\equiv} r^{x} \equiv v_2 \pmod p$$

Again, to fake a signature you would need to know k (the private key). Difficult!

e.g. You want send sign & send message x.

any one could send this → Block1 = x encrypted using receivers public key

only you can send this → Block2 = your signature using your public & private key & Block1

eg. Let $(43, 3, 22)$ be the public key & $k=15$ be the private key.

Choose k' with $\gcd(k', 42) = 1$, say $k' = 25$

If Block1 = 13 (already encrypted).

Then $s_1 = 3^{25} \equiv 5 \pmod{43}$

$25 s_2 \equiv 13 - 5 \cdot 15 \pmod{42}$ gives $s_2 = 16 \pmod{42}$

$\therefore$ digital signature $s = (s_1, s_2) = (5, 16)$

To verify: $v_1 \equiv 22^{5} \cdot 5^{16} \equiv 39 \cdot 40 \equiv 12 \pmod{43}$

$v_2 \equiv 3^{13} \equiv 12 \pmod{43}$

Final
Not needed for 1 Exam →

Attacks on RSA

Recall public-key (n, e) s.t. $\gcd(e, \phi(n)) = 1$
private key d (multiplicative inverse of e modulo $\phi(n)$)

$$E(x) \equiv x^e \pmod{n}$$
$$D(y) \equiv y^d \pmod{n} \qquad D(E(x)) \equiv x \pmod{n}$$

Fact 1 If we know the factorization of $n = pq$
Then we can find $\phi(n)$
which will allow us to find d (since e is known)

Fact 2 If we know $\phi(n)$ then we can factor n

$n = pq$ & $\phi(n) = pq - (p+q) + 1$, i.e. $p+q = n+1-\phi(n)$

$\therefore$ $x^2 - (p+q)x + pq = (x-p)(x-q)$
can be found using quadratic formula.

Fact 3 Given d, we can efficiently factor n.

"Factorization of n"
"value of $\phi(n)$"
& "value of d" can each be found ~~from~~ The knowledge from one of the other two.

Attacks

(I) n can be factored if <u>p & q are close to each other</u>

- Use Fermat's factorization method

(II) <u>Common modulus</u> To avoid generating a different modulus n for each user, one may wish to fix N ~~once~~ for all.
A trusted central authority provides user i with unique pair e_i & d_i to form the public key (n, e_i) & ~~private~~ private key (n, d_i).

Now, ciphertext x^{e_i} meant for user i cannot be decrypted by user j because he doesn't have d_i. However, as noted above user j can use his own d_j & e_j to factor n & obtain d_i from the public key e_i (as its unique mult. inverse).

(III) Low private exponent:

Small d speeds up decryption. However –

Wiener's attack: Let $n = pq$, with $q < p < 2q$.

Let $d < \frac{1}{3} n^{1/4}$ ($\longleftrightarrow$ improved to $d < n^{0.29\ldots}$)

Given (n, e) with $ed \equiv 1 \pmod{\phi(n)}$, d can be found efficiently.

Proof depends on a property of "continued fractions".