CSCI 3100

Asymmetric Encryption

Today

- Posted issue w/ HW:
  - #1: use all 3 types
  - #2: if no soln, tell me why
- HW due Friday

Last time

AES:

(also a bunch of $\mathbb{Z}_n$)

Since working in $\mathbb{Z}_{256}$ or $\mathbb{Z}_{512}$, addition is XOR

Weakness:

Need a common key (secret)

More interesting:

How do we agree on a secret key?

Best way: Physically exchange

However, impractical for things like web traffic or email.

Public Key Cryptography:

Encryption function E, decryption D

Goals:

1. $D(E(M)) = M$ (and $E(D(M)) = M$)
2. E + D are fast
3. Given E, hard to derive D

Diffie-Hellman Key exchange

From "New directions in cryptography"
by Diffie + Hellman in 1976

Daily conspiracy tidbit:
Actually discovered by UK government in 1973!

Key exchange

- Start with $\mathbb{Z}_p$
  (p prime or power of a prime)

These groups have multiplicative inverses:

i.e. $2x = 1 \mod 5$

$x = 3$ is inverse

The protocol: Alice + Bob:

- p + $s < p$ $\;s \in \mathbb{Z}_p$ are both public
- Alice chooses secret $a < p$
  Bob chooses secret $b < \cancel{b}\, p$
- Alice ~~posts~~ sends $A = s^a \bmod p$
  Bob ~~posts~~ sends $B = s^b \bmod p$

Alice computes:

$$K = B^a \bmod p$$

Bob computes:

$$K = A^b \bmod p$$

$$\left.\begin{aligned} B^a &= (s^b)^a \\ A^b &= (s^a)^b \end{aligned}\right\} = s^{ab}$$

Example:

- $s = 2$, $p = 29$
- Alice picks $a = 3$

  Bob picks $b = 7$

$A = 2^3 \bmod 29$

$\equiv 8 \leftarrow$ Alice sends

$B = 2^7 \bmod 29$

$= 12 \leftarrow$ Bob sends

$$\left.\begin{array}{l} 8^7 \bmod 29 \\ 12^3 \bmod 29 \end{array}\right\} = 17$$

Why?

Common key is $k = s^{ab} \bmod p$

Public info: p, s, $A = s^{a} \bmod p$
and $B = s^{b} \bmod p$

What can an attacker try?

$$A \cdot B = (s^{a})(s^{b})$$
$$= s^{a+b} \quad \times$$

Attacker must find "hidden" exponent.

Hardness?

At its root, the key to why this is difficult is the discrete log problem:

Remember logarithms?

$$\log_{10} 1000 =$$

$$\log_2 1024 =$$

Here, discrete version:

Given A, find $\log_s A$

$$= \log_s s^A \quad \text{modulo } p$$

How hard?

This problem is connected to factoring

↳ NOT NP-Hard!

But no efficient algorithms are known. polynomial

Other key exchange algorithms work in other groups (like elliptic curves)

RSA: Rivest, Shamir & Adleman

Its hardness is tied to factoring

$\Rightarrow$ any fast algorithm to factor a number would break RSA.

First: more number theory!

Euler's totient function, $\Phi(n)$:

$\Phi(n) :=$ # of positive integers $\leq n$ that are relatively prime to n.

Ex: $\Phi(12) = \cancel{3}\ 4$ (1, 5, 7, 11)

If p is prime:

$\Phi(p) = p - 1$

$\Phi(7) = 6$ (1, 2, 3, 4,

What about non-primes?

Interesting special case when

$n = p \cdot q$, p & q prime

What is not relatively prime with n?

divisors: p & q, $q \cdot p$

$2p, 3p, \ldots, q \cdot p$

$2q, 3q, \ldots, qp$

So $\Phi(pq) =$

$$pq - p - (q-1)$$

$$\ldots = (q-1)(p-1)$$

Why we care:

Remember, a number k has to be relatively prime to n in order to have a multiplicative inverse in $\mathbb{Z}_n$.

Ex: 5 in $\mathbb{Z}_{22}$

$5 \cdot 9 = 45 = 1 \mod 22$

9 is 5's inverse

Euler's Thm: $n \in \mathbb{Z}$, and $x \in \mathbb{Z}_n$ s.t. $\gcd(x, n) = 1$.

Then $x^{\Phi(n)} \equiv 1 \mod n$.

So:

To compute inverses, need

- gcd algorithm
- $\Phi(n)$

Then Euler's thm:

$$x^{\Phi(n)} \equiv 1 \mod n$$

$$\Longrightarrow x \cdot \left(x^{\Phi(n)-1}\right) \equiv 1 \mod n$$

More generally, inverses in $\mathbb{Z}_n$ are a bit more complex...

Back to RSA (ie why we care!)

Steps: Select 2 large primes p & q

choose p & q

- Let $n = pq$

$\hookrightarrow \Phi(n) = (p-1)(q-1)$

- Select e & d s.t.
  - e and $\Phi(n)$ are relatively prime
  - $ed \equiv 1 \mod \Phi(n)$

$\rightarrow$ How to get d?

Computing inverses:

Remember Euclidean alg:

$$\gcd(\overset{e,}{x}, \overset{\Phi(n)}{n}) = d$$

gcd here = 1

Can augment EA to give

$$ix + jn = \gcd(x, n) = d$$

Now, if $\gcd(x,n) = 1$:

$$ix + j\underline{n} = 1$$

in $\mathbb{Z}_n$,

$$i \cdot x = 1 \bmod n$$

Extended Euclidean Algorithm:

Euclidean algorithm computed:

$$d = \gcd(a, b)$$

by doing

$$\gcd(a,b) = \gcd(b, \underbrace{a \bmod b})$$

Let $r = \underline{a \bmod b}$

$$\Rightarrow \quad a = bq + r$$

for some $q \in \mathbb{Z}$

We will modify Euc Alg so that each call returns not just the gcd, but also i & j

where $d = i \cdot b + j \cdot r$

$1 =$

inverse mod b

Some ugly math!

Goal:

Had $r = a \bmod b$
and $a = qb + r$

$\Rightarrow r = a - qb$

If $d = ib + jr$

$= ib + j(a - qb)$

$= j\,a + (i-q)\,b$

j here is
a's inverse mod b.

Extended EucAlg(a,b):

If $b = 0$

return $(a, 1, 0)$ $\Big\}$ $a = 1 \cdot a + 0 \cdot b$

i j

else

$r \leftarrow a \bmod b$

$q \leftarrow$ integer part of $\frac{a}{b}$

$(d, i, j) \leftarrow$ Extended EucAlg(b, r)

return $(d, j, i - jq)$

Runtime:

$O(\log n)$

So: RSA (finally!)

Bob: Selects 2 large primes p & q

- Let $n = pq$
  ↳ $\Phi(n) = (p-1)(q-1)$
- Select e & d s.t.
  - e and $\Phi(n)$ are relatively prime
  - $ed \equiv 1 \mod \Phi(n)$
    ↳ Extended Euc Alg

Now:
- (e, n) is public key
- d is private key (also p & q)

Encryting: Alice gets (e,n).

She takes a message M, with $0 < M < n$.

(chops into pieces)

Then:

$$C \leftarrow M^e \bmod n$$

(Remember public part: (e,n) was key)

Alice sends C to Bob

Decrypting: Bob gets C

$$C = M^e \bmod n$$

Bob calculate:

$$C^d \bmod n$$

Claim: $\stackrel{?}{=} M$

Why?

$$C^d \bmod n = (M^e)^d \bmod n$$

$$= M^{ed} \bmod n$$

Know $ed = 1 \bmod \Phi(n)$

$$M^{(ed)} = M^{(k\Phi(n)+1)} \bmod n$$

$$= \left(\underbrace{M^{\Phi(n)}}_{\text{Euler thm}}\right)^k \cdot M^1 \bmod n$$

$$= 1^k \cdot M \bmod n = M$$

<u>So</u>: Why secure?
Bob can decrypt!
He knows (secret) d.

Attacker Eve's goal:
Figure out d!

How? • Bob needed $\Phi(n)$, since
d is e's inverse mod n.
• Attacker knows n
(but <u>not</u> $\Phi(n)$).

How to find $\Phi(n)$?

So:

Whole thing is secure, as long as Eve can't get $\Phi(n)$, or $p + q$.

Bad news: Factoring is NOT NP-Hard.

Best algorithm:

Number field sieve:

$$O\left(e^{\left(\frac{64}{9}\log n\right)^{1/3}(\log\log n)^{2/3}}\right)$$

Some practical notes

- RSA can be used to encrypt entire message (but usually isn't)

- Slow (compared to XOR-ing)
- Easier to break than AES or other symmetric protocols

- Also: I was assuming $(M, n) = 1$!

  Here, saved since $n = pq$, & M <u>will</u> be relatively prime to p or q.