CS443 - More Crypto

Note Title 1/18/2013

Announcements

- Essay due
- Lab in a week - any issues?
- This week: encryption + authentication (on next homework)

Last time

Symmetric key cryptography:

- based on shared knowledge of a private key
- very secure: 128-bit key in AES would take roughly 10 billion billion (est. 1.02×10^{18}) years to crack on a 2012 supercomputer

↳ simple operations: XOR, etc...

Assymetric key Cryptography

- based on 1-way communication
- huge development, first published in "New directions in cryptography" by Diffe + Hellman, 1976

Daily Conspiracy:
Actually, secretly developed by government researchers in the U.K. in 1973.

General overview

(1) Alice looks up Bob's public key E_B in a directory.
Uses an algorithm to encode message m: written $E_B(m)$

(2) Bob then uses private key D_B to decode: $D_B(E_B(m)) = m$

Idea: Alice can give away m & $E_B(m)$, but no one else can decode without D_B.

Number Theory

The Euler phi function $\phi(n)$ is defined so that

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

↑ no common divisors > 1

Ex: $\phi(7) = 6$

5, 7

$\phi(6) = 2$

5, 6

↑ 1 & 5

$\phi(25) = 20$

↳ not relatively prime: 5, 10, 15, 20

Lemma: If p is prime, $\phi(p) = p-1$

Lemma: If p & q are prime, $p \neq q$,

$$\phi(p \cdot q) = (p-1)(q-1)$$
$$= \phi(p)\,\phi(q)$$

Why? not rel. prime #s:

$p, 2p, 3p, \ldots, (q-1)p$

$q, 2q, \ldots, (p-1)q$

Euler's thm: If n is a positive integer with $\gcd(a,n) = 1$, then $a^{\phi(n)} = 1 \mod n$

relatively prime

Why we care: inverses!

a & $a^{\phi(n)-1}$ are inverses under multiplication in $\mathbb{Z}_n$.

r r^a

6 + 5 are relatively prime.

$n = 6$

$\phi(n) = 2$

$a^{\phi(n)} \equiv 5^2 = 25 \equiv 1 \mod 6$

$pq = n$

Cor: If a is relatively prime to p & q (both primes), then

$$a^{(p-1)(q-1)} = 1 \pmod{pq}$$

$$\| $$

$$a^{\phi(p)\phi(q)}$$

Remember $\mathbb{Z}_n$?

Key: If a is relatively prime to n, then $\exists b$ with $ab = 1 \bmod n$ ← inverse!

Ex: $n = 8$ $\quad \phi(8) = \underline{4}$

$a = 2$	:	no inverse
$a = 3$	:	$3^3 = 27$
$a = 4$	:	no inverse
$a = 5$	:	$5^3 = 125$
$a = 6$	:	no inverse
$a = 7$	:	$7^3 = 343$

} ? x^2 ?

RSA: [Rivest - Shamir - Adelmann 1978]

① Bob generates 2 primes p & q and computes $n = p \cdot q$
$\phi(n) = (p-1)(q-1)$

② Bob picks e relatively prime to $\phi(n)$, & then finds d s.t. $ed = 1 \bmod \phi(n)$
(via Euclidean algorithm)

→ d is private key

→ (e, n) are public key

Side note: Euclidean Algorithm

```
Input: a, b
While b ≠ 0
    r ← a mod b
    a ← b
    b ← r
Return a
```

What does it do?

calculates gcd

So: given e & $\phi(n)$ relatively prime,
then $\gcd(e, \phi(n)) = 1$

By tracking variables in this algorithm, get value of d.

Key: d is inverse mod n!

Now: Alice has a message m, pubic key (e, n).
To send it to Bob:

1. Compute $c = m^e \bmod n$
2. Send to Bob

Bob decodes:

$$c^d = (m^e)^d \bmod n$$
$$= m^{ed} \bmod n$$
$$= m^1 \bmod n$$

Why it works:

- know $ed = 1 \mod \phi(n)$

  so $ed = 1 + k\phi(n)$ for some k

  $= 1 + k(p-1)(q-1)$

- Then $m^{ed} = m^{1+k\phi(n)} \mod n$

  $= m^1 \cdot m^{k\phi(n)} \mod n$

  $= m^1 \cdot \left(m^{\phi(n)}\right)^k \mod n$

  $= 1$

Public: (e, n)

How hard to get d?

- Easy if you know $\phi(n) = (p-1)(q-1)$
  - ↳ use euclidean alg

Otherwise: factor n

Notes

- Actually, in general no one will know p or q — use central certificate authority.

Public: (e, n)
Private: d

How do we get d again?

↳ $ed \equiv 1 \bmod \phi(n)$

This is why the effectiveness of RSA is based on factoring!

If we knew $\phi(n) = (p-1)(q-1)$, could break the system.

How hard is factoring?

No 512-bit number has (yet) been factored.

Diffie-Hellman key exchange [1976]

Most common use of public key cryptography is to exchange of private keys!

Why?

Symmetric encryption is more secure!

Diffie-Hellman basics:

Consider a prime number q
(or $q = p^k$, with p prime).

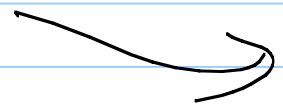
We saw last time that $\mathbb{Z}$ mod q is a finite field:
- nice $+$ $-$ $\times$ operation
- has multiplicative inverse:

Ex: $2 \cdot x = 1 \mod 5$

$\Rightarrow x = 3$

The protocol: p prime, $s < p$ (both public)

- Alice chooses $a < p$ (private)
  Bob chooses $b < p$
- Alice computes $\alpha = s^a \bmod p$ (public)
  Bob computes $\beta = s^b \bmod p$
- They exchange α and β
- Alice computes $\beta^a \bmod p$
  Bob computes $\alpha^b \bmod p$



- Alice computes $\beta^a \bmod p$
  Bob computes $\alpha^b \bmod p$

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

Ex: Let $s = 2, p = 29$

Alice likes $a = 3$

Bob likes $b = 7$

$\alpha = 2^3 \bmod 29 = 8$

$\beta = 2^7 \bmod 29 = 12$

Alice $= 12^3 \bmod 29 = 17$ ✓

Bob $= 8^7 \bmod 29 = 17$

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

Recap: Public info: • p and s
• $\alpha = s^a \bmod p$
• $\beta = s^b \bmod p$

Private: a (to Alice)
b (to Bob)
k

$\alpha\beta = (s^a)(s^b) = s^{a+b} \bmod p$

Why is it hard to break?

At its base, the key is logarithms.
(Remember those?)

$\log_{10} 1000 = 3$

$\log_2 1024 = 10$

We want discrete logs:
given α, find a s.t. $y = \log_a \alpha \pmod p$

The Discrete Log Problem

This is another one we "think" is hard.

Similar to factoring.

Note: There are ways to attack this! Not NP-Hard — just no known fast algorithms.

Stronger generalizations work in groups other than $\mathbb{Z}_p$.

(But elliptic curves are a bit beyond us now...)

Bigger Picture : NSA Suite B

The NSA has published a set of recommended algorithms (for both unclassified information as well as info up to SECRET).

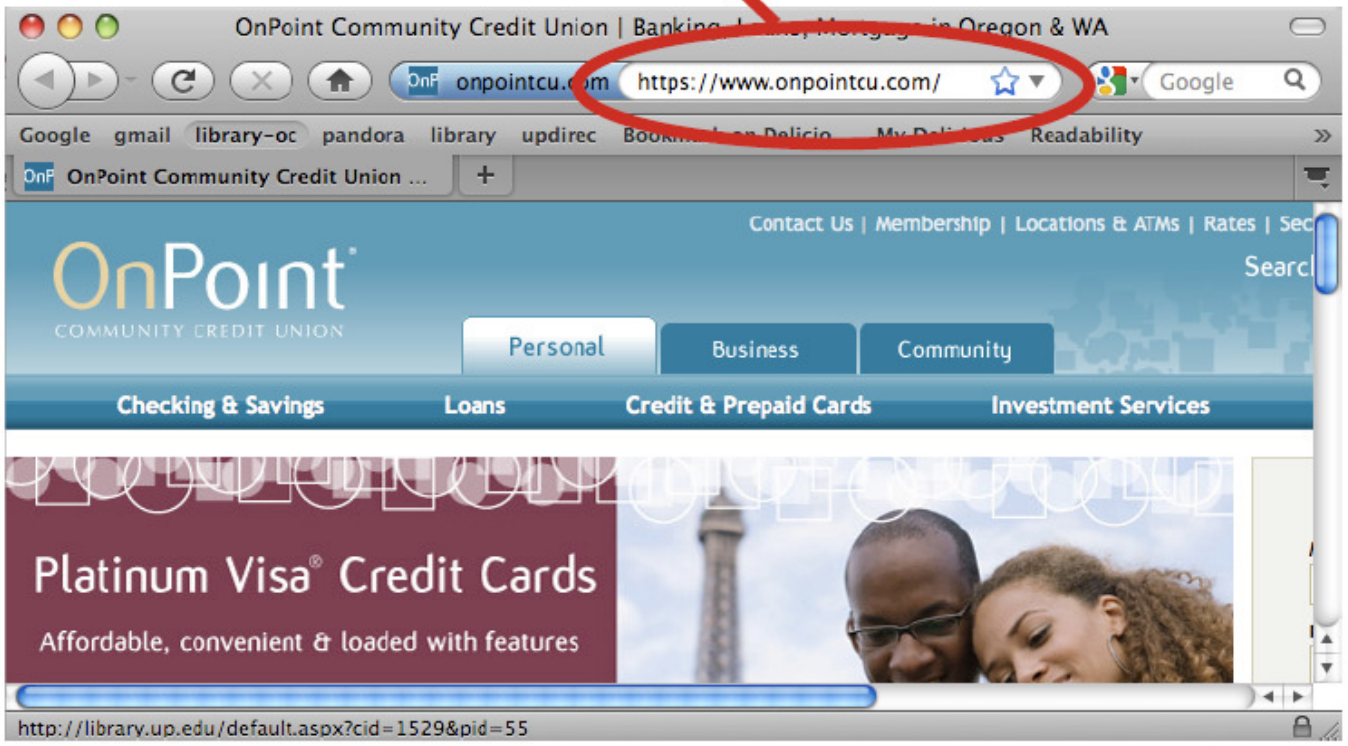
- Encryption : AES
- Signatures : Elliptic curve Diffe-Hellman
- Hashing : SHA (Secure hashing algorithm)

So- why study RSA??

Whenever you see "https", that's TLS at work.

Still in use!

(RSA is the basis of the Transport Layer Security in browsers.)



Why RSA: Cost

(Also used in smart cards, operating systems, etc.)

But why, if ECDH is better??

Most companies can't afford it, since the patents are still largely held by one company.