CS70: Lecture 9. Outline.

1. Public Key Cryptography
2. RSA system
   2.1 Efficiency: Repeated Squaring.
   2.2 Correctness: Fermat's Theorem.
   2.3 Construction.
3. Warnings.

Isomorphisms.

Bijection:

$f(x) = ax \pmod{m}$ if $gcd(a,m) = 1$.

Simplified Chinese Remainder Theorem:
There is a unique $x \pmod{mn}$ where $x = a \pmod{m}$ and $x = b \pmod{n}$ and $gcd(n,m) = 1$.

Bijection between $(a \pmod{n}, b \pmod{m})$ and $x \pmod{m}n$.

Consider $m = 5$, $n = 9$, then if $(a,b) = (3,7)$ then $x = 43 \pmod{45}$.

Consider $(a',b') = (2,4)$, then $x = 22 \pmod{45}$.

Now consider: $(a,b) + (a',b') = (0,2)$.

What is x where $x = 0 \pmod{5}$ and $x = 2 \pmod{9}$?

Try $43 + 22 = 65 = 20 \pmod{45}$.

Is it $0 \pmod{5}$? Yes! Is it $2 \pmod{9}$? Yes!

Isomorphism:
the actions under $\pmod{5}$, $\pmod{9}$
correspond to actions in $\pmod{45}$!

Xor

Computer Science:
 1 - True
 0 - False

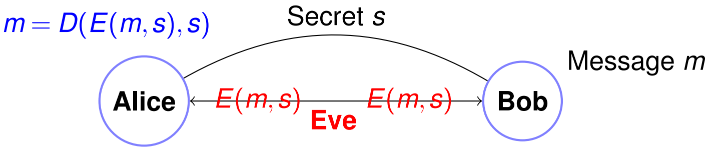
$1 \vee 1 = 1$
$1 \vee 0 = 1$
$0 \vee 1 = 1$
$0 \vee 0 = 0$

$A \oplus B$ - Exclusive or.
$1 \oplus 1 = 0$
$1 \oplus 0 = 1$
$0 \oplus 1 = 1$
$0 \oplus 0 = 0$

Note: Also modular addition modulo 2!
 $\{0, 1\}$ is set. Take remainder for 2.

Property: $A \oplus B \oplus B = A$.
By cases: $1 \oplus 1 \oplus 1 = 1$. ...

Cryptography ...



Example:
One-time Pad: secret s is string of length $|m|$.
$m = 1010101110101101$
$s = \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$
$E(m,s)$ – bitwise $m \oplus s$.
$D(x,s)$ – bitwise $x \oplus s$.
Works because $m \oplus s \oplus s = m$!
...and totally secure!
...given $E(m,s)$ any message m is equally likely.

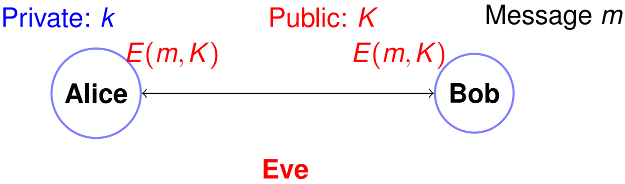
Disadvantages:

Shared secret!

Uses up one time pad..or less and less secure.

Public key crypography.

$m = D(E(m,K),k)$



Everyone knows key K!
Bob (and Eve and me and you and you ...) can encode.
Only Alice knows the secret key k for public key K.
(Only?) Alice can decode with k.

Is this even possible?

Is public key crypto possible?

We don't really know.
...but we do it every day!!!

RSA (Rivest, Shamir, and Adleman)
Pick two large primes p and q. Let $N = pq$.
Choose e relatively prime to $(p-1)(q-1)$.[1]
Compute $d = e^{-1} \mod (p-1)(q-1)$.
Announce $N (= p \cdot q)$ and e: $K = (N, e)$ is my public key!

Encoding: $\mod (x^e, N)$.

Decoding: $\mod (y^d, N)$.

Does $D(E(m)) = m^{ed} = m \mod N$?

Yes!

[1] Typically small, say $e = 3$.

Iterative Extended GCD.

Example: $p = 7$, $q = 11$.

$N = 77$.
$(p-1)(q-1) = 60$
Choose $e = 7$, since $\gcd(7,60) = 1$.
egcd(7,60).

$$\begin{aligned} 7(0)+60(1) &= 60 \\ 7(1)+60(0) &= 7 \\ 7(-8)+60(1) &= 4 \\ 7(9)+60(-1) &= 3 \\ 7(-17)+60(2) &= 1 \end{aligned}$$

Confirm: $-119+120 = 1$

$d = e^{-1} = -17 = 43 = \pmod{60}$

Encryption/Decryption Techniques.

Public Key: $(77,7)$
Message Choices: $\{0,\ldots,76\}$.

Message: 2!

$E(2) = 2^e = 2^7 \equiv 128 \pmod{77} = 51 \pmod{77}$
$D(51) = 51^{43} \pmod{77}$
uh oh!

Obvious way: 43 multiplications. Ouch.

In general, $O(N)$ or $O(2^n)$ multiplications!

Repeated squaring.

Notice: $43 = 32 + 8 + 2 + 1$. $51^{43} = 51^{32+8+2+1} = 51^{32} \cdot 51^{8} \cdot 51^{2} \cdot 51^{1} \pmod{77}$.

4 multiplications sort of...

Need to compute $51^{32} \ldots 51^{1}$.?

$51^{1} \equiv 51 \pmod{77}$

$51^{2} = (51) * (51) = 2601 \equiv 60 \pmod{77}$

$51^{4} = (51^{2}) * (51^{2}) = 60 * 60 = 3600 \equiv 58 \pmod{77}$

$51^{8} = (51^{4}) * (51^{4}) = 58 * 58 = 3364 \equiv 53 \pmod{77}$

$51^{16} = (51^{8}) * (51^{8}) = 53 * 53 = 2809 \equiv 37 \pmod{77}$

$51^{32} = (51^{16}) * (51^{16}) = 37 * 37 = 1369 \equiv 60 \pmod{77}$

5 more multiplications.

$51^{32} \cdot 51^{8} \cdot 51^{2} \cdot 51^{1} = (60) * (53) * (60) * (51) \equiv 2 \pmod{77}$.

Decoding got the message back!

Repeated Squaring took 9 multiplications versus 43.

Recursive version.

```
(define (power x y m)
  (if (= y 1)
    (mod x m)
    (let ((x-to-evened-y  (power (square x) (/ y 2) m)))
    (if (evenp y)
        x-to-evened-y
        (mod (* x x-to-evened-y) m )))))
```

Claim: Program correctly computes x^y.

Base: $x^1 = x \pmod{m}$.

$x^y = x^{2(y/2)+ \bmod (y,2)} = (x^2)^{y/2} x^{y \bmod 2} \pmod{m}$.

The program computes the last expression using a recursive call with x^2 and $y/2$.

Note: $y/2$ is integer division.

Repeated Squaring: x^y

Repeated squaring $O(\log y)$ multiplications versus y!!!

1. x^y: Compute $x^1, x^2, x^4, \ldots, x^{2^{\lfloor \log y \rfloor}}$.
2. Multiply together x^i where the $(\log(i))$th bit of y (in binary) is 1.
   Example: $43 = 101011$ in binary.
   $$x^{43} = x^{32} * x^8 * x^2 * x^1.$$

Modular Exponentiation: $x^y \mod N$. All n-bit numbers. Repeated Squaring:

$O(n)$ multiplications.
$O(n^2)$ time per multiplication.
$\implies O(n^3)$ time.

Conclusion: $x^y \mod N$ takes $O(n^3)$ time.

RSA is pretty fast.

Modular Exponentiation: $x^y \mod N$. All n-bit numbers.
$O(n^3)$ time.

Remember RSA encoding/decoding!

$E(m,(N,e)) = m^e \pmod{N}$.
$D(m,(N,d)) = m^d \pmod{N}$.

For 512 bits, a few hundred million operations.
Easy, peasey.

Decoding.

$E(m,(N,e)) = m^e \pmod N$.
$D(m,(N,d)) = m^d \pmod N$.

$N = pq$ and $d = e^{-1} \pmod{(p-1)(q-1)}$.

Want: $(m^e)^d = m^{ed} = m \pmod N$.

Always decode correctly?

$E(m,(N,e)) = m^e \pmod{N}$.
$D(m,(N,d)) = m^d \pmod{N}$.

$N = pq$ and $d = e^{-1} \pmod{(p-1)(q-1)}$.

Want: $(m^e)^d = m^{ed} = m \pmod{N}$.

Another view:
$d = e^{-1} \pmod{(p-1)(q-1)} \iff ed = k(p-1)(q-1)+1$.

Consider...

Fermat's Little Theorem: For prime p, and $a \not\equiv 0 \pmod{p}$,

$$a^{p-1} \equiv 1 \pmod{p}.$$

$$\implies a^{k(p-1)} \equiv 1 \pmod{p} \implies a^{k(p-1)+1} = a \pmod{p}$$

versus $a^{k(p-1)(q-1)+1} = a \pmod{pq}$.

Similar, not same, but useful.

Correct decoding...

Fermat's Little Theorem: For prime p, and $a \not\equiv 0 \pmod{p}$,

$a^{p-1} \equiv 1 \pmod{p}$.

Proof: Consider $S = \{a \cdot 1, \ldots, a \cdot (p-1)\}$.

All different modulo p since a has an inverse modulo p.
S contains representative of $\{1, \ldots, p-1\}$ modulo p.

$$(a \cdot 1) \cdot (a \cdot 2) \cdots (a \cdot (p-1)) \equiv 1 \cdot 2 \cdots (p-1) \mod p,$$

Since multiplication is commutative.

$$a^{(p-1)}(1 \cdots (p-1)) \equiv (1 \cdots (p-1)) \mod p.$$

Each of $2, \ldots (p-1)$ has an inverse modulo p, solve to get...

$$a^{(p-1)} \equiv 1 \mod p.$$

□

Always decode correctly? (cont.)

Fermat's Little Theorem: For prime p, and $a \not\equiv 0 \pmod{p}$,

$$a^{p-1} \equiv 1 \pmod{p}.$$

Lemma 1: For any prime p and any a, b,
$a^{1+b(p-1)} \equiv a \pmod{p}$
Proof: If $a \equiv 0 \pmod{p}$, of course.

Otherwise
$a^{1+b(p-1)} \equiv a^1 * (a^{p-1})^b \equiv a * (1)^b \equiv a \pmod{p}$

□

...Decoding correctness...

Lemma 1: For any prime p and any a, b,
$a^{1+b(p-1)} \equiv a \pmod{p}$

Lemma 2: For any two different primes p, q and any x, k,
$x^{1+k(p-1)(q-1)} \equiv x \pmod{pq}$

Let $a = x$, $b = k(p-1)$ and apply Lemma 1 with modulus q.

$$x^{1+k(p-1)(q-1)} \equiv x \pmod{q}$$

Let $a = x$, $b = k(q-1)$ and apply Lemma 1 with modulus p.

$$x^{1+k(p-1)(q-1)} \equiv x \pmod{p}$$

$x^{1+k(q-1)(p-1)} - x$ is multiple of p and q.

$$x^{1+k(q-1)(p-1)} - x \equiv 0 \mod (pq) \implies x^{1+k(q-1)(p-1)} = x \mod pq.$$

□

RSA decodes correctly..

Lemma 2: For any two different primes p, q and any x, k,
$x^{1+k(p-1)(q-1)} \equiv x \pmod{pq}$

Theorem: RSA correctly decodes!
Recall

$$D(E(x)) = (x^e)^d = x^{ed} \equiv x \pmod{pq},$$

where $ed \equiv 1 \mod (p-1)(q-1) \implies ed = 1 + k(p-1)(q-1)$

$$x^{ed} \equiv x^{k(p-1)(q-1)+1} \equiv x \pmod{pq}.$$

□

Construction of keys.. ..

1. Find large (100 digit) primes p and q?

   Prime Number Theorem: $\pi(N)$ number of primes less than N.For all $N \geq 17$

   $$\pi(N) \geq N/\ln N.$$

   Choosing randomly gives approximately $1/(\ln N)$ chance of number being a prime. (How do you tell if it is prime? ... cs170..Miller-Rabin test.. Primes in P).

   For 1024 bit number, 1 in 710 is prime.

2. Choose e with $\gcd(e,(p-1)(q-1)) = 1$.
   Use gcd algorithm to test.

3. Find inverse d of e modulo $(p-1)(q-1)$.
   Use extended gcd algorithm.

All steps are polynomial in $O(\log N)$, the number of bits.

Security of RSA.

Security?

1. Alice knows p and q.
2. Bob only knows, $N(= pq)$, and e.

   Does not know, for example, d or factorization of N.
3. I don't know how to break this scheme without factoring N.

No one I know or have heard of admits to knowing how to factor N.
Breaking in general sense $\implies$ factoring algorithm.

Much more to it.....

If Bobs sends a message (Credit Card Number) to Alice,

Eve sees it.

Eve can send credit card again!!

The protocols are built on RSA but more complicated;

For example, several rounds of challenge/response.

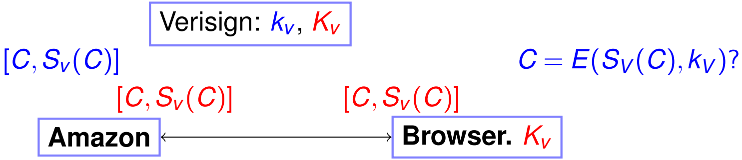
One trick:
  Bob encodes credit card number, c,
    concatenated with random k-bit number r.

Never sends just c.

Again, more work to do to get entire system.

CS161...

Signatures using RSA.



Certificate Authority: Verisign, GoDaddy, DigiNotar,...

Verisign's key: $K_V = (N, e)$ and $k_V = d$ ($N = pq$.)

Browser "knows" Verisign's public key: K_V.

Amazon Certificate: C = "I am Amazon. My public Key is K_A."

Versign signature of C: $S_v(C)$: $D(C, k_V) = C^d \mod N$.

Browser receives: $[C, y]$

Checks $E(y, K_V) = C$?

$$E(S_v(C), K_V) = (S_v(C))^e = (C^d)^e = C^{de} = C \pmod N$$

Valid signature of Amazon certificate C!

Security: Eve can't forge unless she "breaks" RSA scheme.

RSA

Public Key Cryptography:

$D(E(m, K), k) = (m^e)^d \mod N = m.$

Signature scheme:

$E(D(C, k), K) = (C^d)^e \mod N = C$

Other Eve.

Get CA to certify fake certificates: Microsoft Corporation.

2001..Doh.

... and August 28, 2011 announcement.

DigiNotar Certificate issued for Microsoft!!!

How does Microsoft get a CA to issue certificate to them ...

and only them?

Summary.

Public-Key Encryption.

RSA Scheme:
$N = pq$ and $d = e^{-1} \pmod{(p-1)(q-1)}$.
$E(x) = x^e \pmod{N}$.
$D(y) = y^d \pmod{N}$.

Repeated Squaring $\implies$ efficiency.

Fermat's Theorem $\implies$ correctness.

Good for Encryption and Signature Schemes.