CS70: Lecture 8. Outline.

1. Finish Up Extended Euclid.
2. Cryptography
3. Public Key Cryptography
4. RSA system
   4.1 Efficiency: Repeated Squaring.
   4.2 Correctness: Fermat’s Theorem.
   4.3 Construction.
5. Warnings.


ext-gcd(x,y)
if y = 0 then return(x, 1, 0)
else
   (d, a, b) := ext-gcd(y, mod(x,y))
   return (d, b, a − floor(x/y) * b)

Recursively: d = ay + bx − ⌊x/y⌋y
Returns (d, b, (a − ⌊x/y⌋b))


Extended GCD Algorithm.

ext-gcd(x,y)
if y = 0 then return(x, 1, 0)
else
   (d, a, b) := ext-gcd(y, mod(x,y))
   return (d, b, a − floor(x/y) * b)

Theorem: Returns (d,a,b), where d = gcd(a,b) and
   d = ax + by.

Wrap-up

Conclusion: Can find multiplicative inverses in O(n) time!
Very different from elementary school: try 1, try 2, try 3...

Inverse of 500,000,357 modulo 1,000,000,000,000? ≤ 80 divisions.
versus 1,000,000

Internet Security.
Public Key Cryptography: 512 digits.
512 divisions vs.
(1000000000000000000000000000000000000000000)

Correctness.

Proof: Strong Induction.1
Base: ext-gcd(x, 0) returns (d = x, 1, 0) with x = (1)x + (0)y.
Induction Step: Returns (d, a, b) with d = Ax + By
Ind hyp: ext-gcd(y, mod(x,y)) returns (d, a, b) with
   d = ay + bx + mod(x,y)
      ext-gcd(x,y) calls ext-gcd(y, mod(x,y)) so
      d = ay + bx + (a − ⌊x/y⌋b)y
      And ext-gcd returns (d, b, (a − ⌊x/y⌋b)) so theorem holds!

1Assume d is gcd(x,y) by previous proof.

Xor

Computer Science:
1 - True
0 - False
1 ∨ 1 = 1
1 ∨ 0 = 1
0 ∨ 1 = 1
0 ∨ 0 = 0
A ⊕ B - Exclusive or.
1 ⊕ 1 = 0
1 ⊕ 0 = 1
0 ⊕ 1 = 1
0 ⊕ 0 = 0
Note: Also modular addition modulo 2!
(0,1) is set. Take remainder for 2.
Property: A ⊕ B ⊕ B = A.
By cases: 1 ⊕ 1 ⊕ 1 = 1, ...
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).
Compute d = e−1 mod (p − 1)(q − 1).
Announce N(= p * 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^d = m mod N?
Yes!

Typically small, say e = 3.

Encryption/Decryption Techniques.

Public Key: (77, 7)
Message Choices: {0, ..., 76}.
Message: 2!
E(2) = 2^e = 2^7 ≡ 128 (mod 77) = 51 (mod 77)
D(51) = 51^d (mod 77)
uh oh!
Obvious way: 43 multiplications. Ouch.
In general, O(N) multiplications!

Repeated squaring.

Notice: 43 = 32 + 8 + 2 + 1. 51^43 = 51^{32} + 51^8 + 51^2 + 51^1 (mod 77)
4 multiplications sort of...
Need to compute 51^{32}...51^1, ?
51^1 = 51 (mod 77)
51^2 = (51^1) + (51^1) = 60 + 60 = 120 = 60 (mod 77)
51^3 = (51^2) + (51^2) = 60 + 60 = 360 = 58 (mod 77)
51^4 = (51^3) + (51^3) = 58 + 58 = 366 = 53 (mod 77)
51^5 = (51^4) + (51^4) = 35 + 35 = 280 = 37 (mod 77)
51^6 = (51^5) + (51^5) = 37 + 37 = 136 = 60 (mod 77)
5 more multiplications.
51^{12} = 1, 51^8 = (60) * (53) + (60) * (51) = 2 (mod 77).
Decoding got the message back!
Repeated Squaring took 9 multiplications versus 43.
Repeated Squaring: $x^y$

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

1. $x^1$: Compute $x, x^2, x^4, \ldots, x^{2\log y}$.
2. Multiply together $x^i$ where the $(\log i)$th bit of $y$ (in binary) is 1.
Example: $43 = 101011_2$ in binary.
$x^{43} = x^{32} \cdot x^8 \cdot x^2 \cdot x^1$.

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!
$E(m,(N,e)) = m^e \mod N$.
$D(m,(N,d)) = m^d \mod N$.
For 512 bits, a few hundred million operations.
Easy, please.

Always decode correctly?

$E(m,(N,e)) = m^e \mod N$.
$D(m,(N,d)) = m^d \mod N$.
$N = pq$ and $d = e^{-1} \mod (p-1)(q-1)$.
Want: $(m^e)^d = m \mod N$.
Another view:
$d = p^{-1} \mod (p-1)(q-1) \iff ed = k(p-1)(q-1) + 1$.
Consider...
Fermat’s Little Theorem: For prime $p$, and $a \not\equiv 0 \mod p$,
$a^{p-1} \equiv 1 \mod p$.
Proof: Consider $S = \{a, \ldots, a \cdot (p-1)\}$.
All different modulo $p$ since $a$ has an inverse modulo $p$.
$S$ contains representative of $\{1, \ldots, p-1\}$ modulo $p$.
$\therefore (a \cdot 1 \cdot (p-1)) = 1 \cdot (p-1) \mod p$.
Since multiplication is commutative.
$a^{p-1} \cdot (1 \cdot (p-1)) = (1 \cdot (p-1)) \mod p$.
Each of $2, \ldots, (p-1)$ has an inverse modulo $p$, solve to get...

Always decode correctly? (cont.)

Fermat’s Little Theorem: For prime $p$, and $a \not\equiv 0 \mod p$,
$a^{p-1} = 1 \mod p$.

Lemma 1: For any prime $p$ and any $a, b$,
$a^k \equiv a \mod p$.

Lemma 2: For any two different primes $p, q$ and any $x, k,
\begin{align*}
&x^{k(p-1)(q-1)} = x \mod pq
\end{align*}$

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

$x^{k(p-1)(q-1)} = x \mod q$.

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

$x^{k(p-1)(q-1)} = x \mod p$.

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

...Decoding correctness...

Lemma 1: For any prime $p$ and any $a, b$,
$a^k \equiv a \mod p$.

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

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

$x^{k(p-1)(q-1)} = x \mod q$.

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

$x^{k(p-1)(q-1)} = x \mod p$.

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

$\Box$
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
\[
D(E(x)) = (x^e)^d = x \pmod{pq},
\]
where \( ed \equiv 1 \pmod{(p-1)(q-1)} \),
\[
x^d = x^{k(p-1)(q-1)+1} \equiv x \pmod{pq}.
\]

---

**RSA**

---

**Public Key Cryptography:**

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

**Signature scheme:**

\[
E(D(C, k), K) = (C^d)^e = C \pmod{N}.
\]

**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 \( \Rightarrow \) factoring algorithm.

---

**Construction of keys.**

1. Find large (100 digit) primes \( p, q \).

   **Prime Number Theorem:** \( \pi(N) \) number of primes less than \( N \).

   \[
   \pi(N) \geq N/\ln N.
   \]

   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.

---

**Signatures using RSA.**

\[
[C, S_v(C)]
\]

Amazon

Browser. \( K_v \)

Certificate Authority: Verisign, GoDaddy, DigNotar,...

Verisign's key: \( K_v = (N, e) \) and \( k_v = d (N = pq) \).

Browser "knows" Verisign's public key: \( K_v \).

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

Verisign signature of \( C \): \( S_v(C) \).

\[
D(C, k_v) = C^{k_v} \pmod{N}.
\]

Browser receives: \( [C, v] \).

Checks \( E(v, K_v) = C \).

\[
E(S_v(C), K_v) = (S_v(C))^e = (C^{k_v})^e = C^{k_v} \pmod{N}.
\]

Valid signature of Amazon certificate \( C \).

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

---

**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...
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:
\[ N = pq \text{ and } d = e^{-1} \pmod{(p - 1)(q - 1)}. \]
\[ E(x) = x^e \pmod{N}. \]
\[ D(y) = y^d \pmod{N}. \]
Repeated Squaring \(\implies\) efficiency.
Fermat’s Theorem \(\implies\) correctness.
Good for Encryption and Signature Schemes.