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

**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\(^{-1}\) \text{mod} \((p - 1)(q - 1)\).
Announce N=(p\cdot q) and e: K = (N, e) is my public key!

Encoding: \text{mod} (x^e, N).
Decoding: \text{mod} (y^d, N).
Does \(D(E(m)) = m^d \equiv m \text{ mod } N\)?
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 $\text{gcd}(7, 60) = 1$.

\[
\text{gcd}(7, 60) = 7(0) + 60(1) = 60
\]

\[
7(1) + 60(0) = 7
\]

\[
7(-8) + 60(1) = 4
\]

\[
7(9) + 60(-1) = 3
\]

\[
7(-17) + 60(2) = 1
\]

Confirm: $-119 + 120 = 1$

\[d = e^{-1} = -17 = 43 \equiv (\text{mod } 60)\]

### Recursive version.

```lisp
(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:** $x^1 = x \pmod{m}$.

$$x^y = x^{2(\log_2 y)} \pmod{m} = (x^2)^{\log_2 y} \pmod{m}.$$

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

Note: $y/2$ is integer division.

---

**Encryption/Decryption Techniques.**

Public Key: $(77, 7)$

Message Choices: $\{0, \ldots, 76\}$.

Message: 2!

\[E(2) = 2^7 \equiv 128 \pmod{77} = 51 \pmod{77}\]

\[D(51) = 51^{63} \pmod{77}\]

uh oh!

Obvious way: 43 multiplications. Ouch.

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

---

**Repeated Squaring:** $x^y$

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

1. $x^y$: Compute $x^1, x^2, x^4,\ldots, x^{2^{\log y}}$.
2. Multiply together $x^i$ where the $(\log i)$th bit of $y$ (in binary) is 1.

Example: $43 = 101011$ in binary.

\[x^{11} = x^{2^5} x^3 x^1\]

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

\[O(n)\] multiplications.

\[O(n^2)\] time per multiplication.

Conclusion: $x^y \pmod{N}$ takes $O(n^2)$ time.

---

**RSA is pretty fast.**

Modular Exponentiation: $x^y \pmod{N}$. All $n$-bit numbers.

$O(n^2)$ 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.

---

Repeated squaring.

Notice: 43 = 32 + 8 + 2 + 1.

\[51^{43} = 51^{32} \cdot 51^8 \cdot 51^2 \cdot 51^1 \pmod{77}\]

4 multiplications sort of...

Need to compute $51^{32} \cdot 51^8 \cdot 51^2 \cdot 51^1$?

\[51^1 = 51 \pmod{77}\]

\[51^2 = (51^1)^2 = 2601 \pmod{77} \equiv 60 \pmod{77}\]

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

\[51^8 = 51^4 \cdot 51^4 = 3600 \cdot 3600 \pmod{77} \equiv 3664 \pmod{77} \equiv 53 \pmod{77}\]

\[51^{16} = (51^8)^2 \pmod{77} = 53 \cdot 53 = 2809 \pmod{77} \equiv 37 \pmod{77}\]

\[51^{32} = (51^{16})^2 \pmod{77} \equiv 37 \cdot 37 \equiv 1369 \pmod{77} \equiv 60 \pmod{77}\]

5 more multiplications.

\[51^{43} = 51^1 \cdot 51^2 \cdot 51^4 \cdot 51^8 \cdot 51^16 \cdot 51^{32} \pmod{77}\]

Decoding got the message back!

Repeated Squaring took 9 multiplications versus 43.
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
\[
    a^{1+b(p-1)} = a^1 \cdot (a^p)^b = a \cdot (1)^b = a \pmod{p}.
\]

...Decoding correctness...

Lemma 1: For any prime \( p \) and any \( a, b \),
\[
    a^{1+b(p-1)} = a \pmod{p}.
\]
Lemma 2: For any two different primes \( p, q \) and any \( x, k \),
\[
    x^{1+k(p-1)(q-1)} = x \pmod{pq}.
\]
Let \( a = x, b = k(p-1) \) and apply Lemma 1 with modulus \( q \).
\[
    x^{1+k(p-1)(q-1)} = x \pmod{q}.
\]
Let \( a = x, b = k(q-1) \) and apply Lemma 1 with modulus \( p \).
\[
    x^{1+k(p-1)(q-1)} = x \pmod{p}.
\]
\[
    x^{1+k(p-1)(q-1)} - x = \text{multiple of } p \text{ and } q.
\]
\[
    x^{1+k(p-1)(q-1)} - x \equiv 0 \pmod{pq} \implies x^{1+k(p-1)(q-1)} = x \pmod{pq}.
\]

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 \( S = \{a, 1 \ldots a \pmod{p-1}\} \).
All different modulo \( p \) since \( a \) has an inverse modulo \( p \).
\( S \) contains representative of \( \{1 \ldots p-1\} \) modulo \( p \).
\[
    (a \pmod{1} \cdot a \pmod{2} \cdot (a \pmod{p-1}) = 1 \ldots (p-1) \pmod{p}.
\]
Since multiplication is commutative.
\[
    a^{p-1} \cdot (1 \ldots (p-1)) \equiv (1 \ldots (p-1)) \pmod{p}.
\]
Each of \( 2 \ldots (p-1) \) has an inverse modulo \( p \), solve to get...
\[
    a^{p-1} \equiv 1 \pmod{p}.
\]

RSA decodes correctly...

Lemma 2: For any two different primes \( p, q \) and any \( x, k \),
\[
    x^{1+k(p-1)(q-1)} = x \pmod{pq}.
\]
Theorem: RSA correctly decodes!
Recall
\[
    D(E(x)) = (x^e)^d = x^{ed} \equiv x \pmod{pq}.
\]
where \( ed \equiv 1 \pmod{(p-1)(q-1)} \implies ed = 1 + k(p-1)(q-1) \)
\[
    x^{ed} = x^{k(p-1)(q-1)+1} = x \pmod{pq}.
\]
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? ...)

   For 1024 bit number, 1 in 710 is prime.

2. Choose \( e \) with gcd\((e, (p-1)(q-1))-1\) = 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.

\[
\begin{align*}
[&C, S_v(C)] \\
C &= E(S_v(C), k_v) \\
[C, S_v(C)] &= [C, S_v(C)]
\end{align*}
\]

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

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

Browser “knows” Verisign’s public key: \( K_v \).

Verisign signature of C: \( S_v(C) \): \( D(C, k_v) = C^{de} \mod N \).

Browser receives: \( [C, y] \)

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

\( E(S_v(C), k_v) = (S_v(C))^d = (C^{de})^d = C^{d^2} = C \mod N \)

Valid signature of Amazon certificate C!

Security: Eve can’t forge unless she “breaks” RSA scheme.

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.

Public Key Cryptography:

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

Signature scheme:

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

RSA

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 \(\Rightarrow\) efficiency.
Fermat's Theorem \(\Rightarrow\) correctness.
Good for Encryption and Signature Schemes.