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

Public key cryptography.

\[ m = D(E(m, K), k) \]

Private: \( k \)
Public: \( K \)
Message \( m \)

Example:
One-time Pad: secret \( s \) is string of length \( |m| \).
\[ m = 1010101110101101 \]
\[ s = \ldots \]
\[ E(m, s) = \text{bitwise } m \oplus s. \]
\[ D(x, s) = \text{bitwise } x \oplus s. \]
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.

Isomorphisms.

Bijection:
\[ f(x) = ax \pmod{m} \text{ if } \gcd(a, m) = 1. \]

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

Bijection between \( (a \pmod{m}, b \pmod{m}) \) and \( x \pmod{mn} \).
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\( \lor 1 = 1 \)
1\( \lor 0 = 1 \)
0\( \lor 1 = 1 \)
0\( \lor 0 = 0 \)
\( A \oplus B \) - 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: \( 1 \oplus 1 \oplus 1 = 1. \ldots \)

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 \( d = e^{-1} \pmod{(p - 1)(q - 1)}. \)
Announce \( N = (p \cdot q) \) and \( e: K = (N, e) \) is my public key!
Encoding: \( \pmod{x^e, N}. \)
Decoding: \( \pmod{y^d, N}. \)
Does \( D(E(m)) = m^d \equiv m \pmod{N} \)?
Yes!

\(^1\)Typically small, say \( e = 3. \)
Encryption/Decryption Techniques.

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

$$N = 77.$$  
$$(p-1)(q-1) = 60$$

Choose $e = 7$, since $\gcd(7, 60) = 1$.  
$e\gcd(7, 60)$.

$$7(0) + 60(1) = 60$$
$$7(1) + 60(0) = 7$$
$$7(-8) + 60(1) = 4$$
$$7(9) - 60(-1) = 3$$
$$7(-17) + 60(2) = 1$$

Confirm: $-119 + 120 = 1$

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

Claim: Program correctly computes $x^y$.

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

$$x^y = x^{2(y/2) \cdot \text{mod}(y/2)} \cdot (x^2)^{y/2 \cdot \text{mod}(y/2)} \pmod{m}.$$  
The program computes the last expression using a recursive call with $x^2$ and $y/2$.

Note: $y/2$ is integer division.

Encryption/Decryption Techniques.

Public Key: $(77, 7)$

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

Message: 2!

Notice: $43 = 32 + 8 + 2 + 1$.  
$51^{43} = 51^{32} \cdot 8^{2} \cdot 2^{1} = 51^{32} \cdot 51^{16} \cdot 51^{1}$  
$(\mod 77)$

4 multiplications sort of...

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

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

$$51^{16} = (51^{1})^{16} = 1201 = 60 \pmod{77}$$

$$51^{32} = (51^{16})^{2} = 60 \cdot 60 = 3600 = 58 \pmod{77}$$

$$51^{28} = (51^{16})^{2} = 58 \cdot 58 = 3364 = 53 \pmod{77}$$

5 more multiplications.

$$51^{32} = (51^{16})^{2} \cdot (51^{1})^{2} = 73 + 53 = 2809 = 37 \pmod{77}$$

$$51^{32} = (51^{16})^{2} \cdot (51^{1})^{2} = 37 \cdot 37 = 1369 \equiv 60 \pmod{77}$$

5 more multiplications.

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

Decoding got the message back!

Repeated Squaring took 9 multiplications versus 43.

Repeated squaring.

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

1. $x^2$: Compute $x^1, x^2, x^4, \ldots, x^{2^{\lceil \log_2 y \rceil}}$.

2. Multiply together $x$ where the $(\log_2(i))$th bit of $y$ (in binary) is 1.

Example: $43 = 101011$ in binary.

$$x^{43} = x^{32} \cdot x^8 \cdot x^4 \cdot x^2 \cdot x.$$  

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

Repeated Squaring: $O(n)$ multiplications.

$O(n^2)$ time per multiplication.

$O(n^2)$ time.

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.
Decoding.

Always decode correctly?

\[ E(m, (N,e)) = m^e \pmod N \]
\[ D(m, (N,d)) = m^d \pmod N \]
\[ N = pq \text{ and } d = e^{-1} \pmod{(p-1)(q-1)} \]
Want: \((m^d)^e \equiv m \pmod N\).

Correct decoding...

\textbf{Fermat's Little Theorem:} For prime \(p\), and \(a \not\equiv 0 \pmod p\), \n\[ a^{p-1} \equiv 1 \pmod p \]
\textbf{Proof:} Consider \(S = \{a \cdot 1 \cdot \ldots \cdot (p-1)\}\).
Each of 2, ..., \((p-1)\) has an inverse modulo \(p\), solve to get...
\[ a^{(p-1)(q-1)+1} \equiv a \pmod pq \]

...Decoding correctness...

\textbf{Lemma 1:} For any prime \(p\) and any \(a, b\),
\[ a^{p-1} = 1 \pmod p \]
\textbf{Lemma 2:} For any two different primes \(p, q\) and any \(x, k\),
\[ x^{k(p-1)(q-1)} \equiv x \pmod pq \]
\textbf{Proof:} If \(a \equiv 0 \pmod p\), of course.
Otherwise \(a^{p-1} = 1 \pmod p\)
\[ a^{1+k(p-1)(q-1)} = a^{1+b(p-1)} = a \pmod p \]

RSA decodes correctly.

\textbf{Lemma 2:} For any two different primes \(p, q\) and any \(x, k\),
\[ x^{k(p-1)(q-1)} \equiv x \pmod pq \]
\textbf{Theorem:} RSA correctly decodes!
\[ D(E(x)) = (x^e)^d = x^d \equiv x \pmod{pq} \]
Recall
\[ D(E(x)) = (x^e)^d = x^{ed} \equiv x \pmod{pq} \]
where \(ed = 1 \pmod{(p-1)(q-1)}\)
\[ x^{ed} = x^{k(p-1)(q-1)+1} \equiv x \pmod{pq} \]
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 $N \geq 17$
   \[ \pi(N) \approx N / \ln N. \]

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

   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.

Signatures using RSA.

Verisign: $k_v, K_v$

$[C, S_v(C)]$

$C = E(S_v(C), k_v)$

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 $C$: $S_v(C) = C^e \bmod N$.

Browser receives: $[C, y]$

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

$E(S_v(C), K_v) = (S_v(C))^e = (C^e)^e = C^{ed} = C \pmod N$

Valid signature of Amazon certificate $C$!

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

RSA

Public Key Cryptography:

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

Signature scheme:

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

Other Eve.

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...

Much more to it.....

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.