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:

\[ f(x) = ax \mod m \]
if \( \gcd(a, m) = 1 \).

Simplified Chinese Remainder Theorem:

There is a unique \( x \mod mn \) where \( x = a \mod m \) and \( x = b \mod n \) and \( \gcd(n, m) = 1 \).

Bijection between \((a \mod n, b \mod m)\) and \(x \mod mn\).

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

Consider \((a', b') = (2, 4)\), then \( x = 22 \mod 45 \).

Now consider:

\((a, b) + (a', b') = (0, 2)\).

What is \( x \) where \( x = 0 \mod 5 \) and \( x = 2 \mod 9 \)?

Try 43 + 22 = 65 = 20 \( \mod 45 \).

Is it 0 \( \mod 5 \)? Yes!

Is it 2 \( \mod 9 \)? Yes!

Isomorphism:

the actions under \( \mod 5 \), \( \mod 9 \) correspond to actions in \( \mod 45 \)!
Isomorphisms.

Bijection:

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

Bijection between \((a \pmod{n}, b \pmod{m})\) 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 \((\pmod{45})\)!
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 \( \gcd(n, m) = 1 \).
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 \( \gcd(n, m) = 1 \).

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

Consider \( m = 5, n = 9 \), then if \( (a, b) = (3, 7) \) then \( x = 43 \pmod{45} \).
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 \( \gcd(n, m) = 1. \)

Bijection between \((a \pmod{n}, b \pmod{m})\) 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 \( x = 22 \pmod{45} \).
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 \( \gcd(n, m) = 1 \).

Bijection between \( (a \pmod{n}, b \pmod{m}) \) 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:
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 \( \gcd(n, m) = 1. \)

Bijection between \( (a \pmod{n}, b \pmod{m}) \) 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: \( (a, b) + (a', b') = (0, 2). \)
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 \( x \pmod{mn} \).

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 \( x = 2 \pmod{9} \)?
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 \( \gcd(n, m) = 1. \)

Bijection between \( (a \pmod{n}, b \pmod{m}) \) 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 \( 43 + 22 = 65 \)
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 \( \gcd(n, m) = 1 \).

Bijection between \((a \pmod{n}, b \pmod{m})\) 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 \( 43 + 22 = 65 = 20 \pmod{45} \).
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 \(x \pmod{mn}\).

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 \( 0 \pmod{5} \)?
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 \( x \pmod{mn}. \)

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!
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 \( \gcd(n, m) = 1 \).

Bijection between \((a \pmod{n}, b \pmod{m})\) 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 \( 2 \pmod{9} \)?
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 \( \gcd(n, m) = 1 \).

Bijection between \( (a \pmod{n}, b \pmod{m}) \) 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!
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 \(x \pmod{mn}\).

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:
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(n, m) = 1.

Bijection between (a \pmod{n}, 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 \((\pmod{5}), (\pmod{9})\).
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 \( \gcd(n, m) = 1. \)

Bijection between \( (a \pmod{n}, b \pmod{m}) \) 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 \( \pmod{45} \)!
Computer Science:

**Xor**

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.
Xor

Computer Science:
1 - True
0 - False

\[ \begin{align*}
1 \lor 1 &= 1 \\
1 \lor 0 &= 1 \\
0 \lor 1 &= 1 \\
0 \lor 0 &= 0
\end{align*} \]

A \oplus B - Exclusive or.

\[ \begin{align*}
1 \oplus 1 &= 0 \\
1 \oplus 0 &= 1 \\
0 \oplus 1 &= 1 \\
0 \oplus 0 &= 0
\end{align*} \]

Note: Also modular addition modulo 2!

\{0, 1\} is set. Take remainder for 2.

Property:
\[ A \oplus B \oplus B = A. \]

By cases:
\[ 1 \oplus 1 \oplus 1 = 1. \]
Xor

Computer Science:
  1 - True
  0 - False

1 ∨ 1 = 1

Property:
A ⊕ B ⊕ B = A.
Xor

Computer Science:

1 - True
0 - False

1 ∨ 1 = 1
1 ∨ 0 = 1
0 ∨ 1 = 1
0 ∨ 0 = 0
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.
Xor

Computer Science:
  1 - True
  0 - False

\[ 1 \lor 1 = 1 \]
\[ 1 \lor 0 = 1 \]
\[ 0 \lor 1 = 1 \]
\[ 0 \lor 0 = 0 \]

\[ A \oplus B \text{ - Exclusive or.} \]
\[ 1 \oplus 1 = 0 \]

\{0, 1\} is set. Take remainder for 2.

Property:

\[ A \oplus B \oplus B = A \]

By cases:

\[ 1 \oplus 1 \oplus 1 = 1 \]
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: \[ 1 \oplus 1 \oplus 1 = 1 \]
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!
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.
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.
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: \[ A \oplus B \oplus B = A. \]
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.
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. ...
Cryptography...

Example: One-time Pad: secret $s$ is string of length $|m|$. $m = 10101011110101101$ $s = \ldots \ldots$ 

$E(m, s)$ – 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.
Cryptography ...

Example: One-time Pad: secret $s$ is string of length $|m|$.  
$m = 10101011110101101$
$s = ..................................$

$E(m,s)$ – 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..<br/>or less and less secure.
Cryptography ...

Bob

Alice

Secret s

Eve

Bob

Message m

Example: One-time Pad: secret s is string of length |m|.

\( m = 10101011110101101 \)

\( s = \ldots \ldots \) 

\( E(m, s) \) – 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.
Cryptography ...

\[ E(m, s) \]

\[ m = D(E(m, s), s) \]

Example: One-time Pad

\[ s \text{ is string of length } \|m\| \]

\[ m = 10101011110101101 \]

\[ s = \ldots \ldots \ldots \]

\[ E(m, s) - \text{bitwise } m \oplus s \]

\[ D(x, s) - \text{bitwise } x \oplus s \]

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.
Cryptography ...

Example: One-time Pad: secret $s$ is string of length $|m|$.  
$m = 10101011110101101$  
$s =$..................................

$E(m, s)$ – 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.
Cryptography ...

\[ m = D(E(m, s), s) \]

Example:

One-time Pad: secret \( s \) is string of length \( |m| \).

\[ m = 10101011110101101 \]

\[ s = \ldots \ldots \ldots \ldots \ldots \ldots \]

\[ E(m, s) \] – 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.
Cryptography...

Example:

One-time Pad: secret $s$ is string of length $|m|$.  

$m = D(E(m, s), s)$

Example: 

$m = 10101011110101101$

$s = \ldots \ldots$

$E(m, s)$ – 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.
Cryptography ...

\[ m = D(E(m, s), s) \]

Example:
One-time Pad: secret \( s \) is string of length \( |m| \).
Cryptography ...

\[ m = D(E(m, s), s) \]

Example:
One-time Pad: secret \( s \) is string of length \( |m| \).
\[ m = 10101011110101101 \]
Cryptography ...

\[ m = D(E(m, s), s) \]

Example:
One-time Pad: secret s is string of length \(|m|\).
\[ m = 10101011110101101 \]
\[ s = \ldots \]

\[ E(m, s) \] – 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.
Cryptography ...

\[ m = D(E(m, s), s) \]

Example:
One-time Pad: secret \( s \) is string of length \( |m| \).
\[ m = 10101011110101101 \]
\[ s = \ldots \]
\[ E(m, s) \) – bitwise \( m \oplus s \).
Cryptography ...

\[ m = D(E(m, s), s) \]

Example:
One-time Pad: secret \( s \) is string of length \( |m| \).
\[
\begin{align*}
m &= 10101011110101101 \\
s &= \ldots \ldots \ldots \ldots \ldots \ldots \ldots \\
E(m, s) &= \text{bitwise } m \oplus s. \\
D(x, s) &= \text{bitwise } x \oplus s.
\end{align*}
\]
Cryptography ...

\[ m = D(E(m, s), s) \]

Example:
One-time Pad: secret \( s \) is string of length \( |m| \).
\[ m = 10101011110101101 \]
\[ s = \ldots \]
\[ E(m, s) - \text{bitwise } m \oplus s. \]
\[ D(x, s) - \text{bitwise } x \oplus s. \]
Works because \( m \oplus s \oplus s = m! \)
Cryptography ...

\[ m = D(E(m, s), s) \]

**Example:**
One-time Pad: secret \( s \) is string of length \(|m|\).

\[ m = 10101011110101101 \]

\[ s = \ldots \]

\( E(m, s) \) – bitwise \( m \oplus s \).

\( D(x, s) \) – bitwise \( x \oplus s \).

Works because \( m \oplus s \oplus s = m! \)

...and totally secure!
Example:
One-time Pad: secret s is string of length $|m|$.

$m = 10101011110101101$

$s = \ldots\ldots\ldots\ldots\ldots$

$E(m, s)$ – 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.
Cryptography ...

\[ m = D(E(m,s),s) \]

Example:
One-time Pad: secret \( s \) is string of length \( |m| \).
\[ m = 10101011110101101 \]
\[ s = \ldots \]
\[ E(m,s) \text{ – bitwise } m \oplus s. \]
\[ D(x,s) \text{ – bitwise } x \oplus s. \]
Works because \( m \oplus s \oplus s = m! \)
...and totally secure!
...given \( E(m,s) \) any message \( m \) is equally likely.

Disadvantages:
Example:
One-time Pad: secret s is string of length $|m|$.
\[ m = 10101011110101101 \]
\[ s = \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \]
\[ E(m, s) \text{ – bitwise } m \oplus s. \]
\[ D(x, s) \text{ – bitwise } x \oplus s. \]
Works because $m \oplus s \oplus s = m$!
...and totally secure!
...given $E(m, s)$ any message $m$ is equally likely.

Disadvantages:
Shared secret!
Cryptography ...

\[ m = D(E(m, s), s) \]

Example:
One-time Pad: secret \( s \) is string of length \(|m|\).

\[ m = 10101011110101101 \]

\[ s = \ldots \ldots \ldots \]

\( E(m, s) \) – 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..
Example:
One-time Pad: secret $s$ is string of length $|m|$.

$m = 10101011110101101$
$s = \ldots \ldots \ldots$

$E(m, s)$ – 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 cryptography.

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?
Public key cryptography.

Public key cryptography allows for secure communication between two parties, Alice and Bob, without the need for a shared secret key. Alice uses a public key, denoted as $K$, to encrypt messages, while Bob uses this same public key to decrypt them. Only Alice knows the secret key $k$, which is used to decrypt messages encrypted with $K$. Eve, an eavesdropper, knows the public key $K$ and can intercept messages, but without the secret key $k$, she cannot decrypt them.

Mathematically, this can be represented as:

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

Where $E$ denotes encryption, $D$ denotes decryption, $m$ is the message, $K$ is the public key, and $k$ is the secret key.

Is this even possible?
Public key cryptography.

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?
Public key cryptography.

Bob

Alice

Private: $k$

Public: $K$

Message $m$

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?
Public key cryptography.

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?
Public key cryptography.

Private: $k$

Public: $K$

Message $m$

$E(m, K)$

Is this even possible?
Public key cryptography.

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

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

Alice \( \rightarrow \) Bob  
\( E(m, K) \)

Eve
Public key cryptography.

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

Everyone knows key \( K \)!

**Diagram:**
- **Alice**
  - Private: \( k \)
  - Public: \( K \)
  - \( E(m, K) \)

- **Bob**
  - Message \( m \)

- **Eve**

(Only?) Alice can decode with \( k \).
Public key cryptography.

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

Everyone knows key \( K \)!
Bob (and Eve) can encode.

Only Alice knows the secret key \( k \) for public key \( K \).
(Only?) Alice can decode with \( k \).
Is this even possible?
Public key cryptography.

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

Everyone knows key \( K \)!
Bob (and Eve and me) can encode.
Only Alice knows the secret key \( k \) for public key \( K \).
(Only?) Alice can decode with \( k \).
Is this even possible?
Public key cryptography.

Bob (and Eve and me 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?

$m = D(E(m, K), k)$
Public key cryptography.

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

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

Alice  
Bob

Eve

Everyone knows key \( K \)!
Bob (and Eve and me and you and you ...) can encode.
Public key cryptography.

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

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

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 \( K \).
Public key cryptography.

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

Private: $k$  
Public: $K$  
Message $m$

Alice  
Bob  
Eve

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 $k$. 
Public key cryptography.

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

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 \((p−1)(q−1)\).

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 \( D(E(m)) = med = m \mod N \)?

Yes!

\(^1\) Typically small, say \( e = 3 \).
Is public key crypto possible?

We don’t really know.

---

1 Typically small, say \( e = 3 \).
Is public key crypto possible?

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

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

\[ \text{Does } \text{D(E(m))} = \text{med} = \text{m mod N}? \]

\[ \text{Yes!} \]

\(^1\text{Typically small, say } e = 3.\]
Is public key crypto possible?

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

RSA (Rivest, Shamir, and Adleman)

---

Typically small, say $e = 3$.  

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 $N = pq$.

---

1 Typically small, say $e = 3$. 
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

¹Typically small, say $e = 3$. 
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 $d = e^{-1} \mod (p − 1)(q − 1)$.

\(^1\)Typically small, say $e = 3$. 
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!

---

$^1$Typically small, say $e = 3$. 
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 $N(= p \cdot q)$ and $e$: $K = (N, e)$ is my public key!

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

\[1\] Typically small, say $e = 3$. 
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: \( \mod (y^d, N) \).

\(^1\)Typically small, say \( e = 3 \).
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 \( D(E(m)) = m^{ed} = m \mod N \)?

---

\(^1\) Typically small, say \( e = 3 \).
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 $N(= p \cdot q)$ 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!

---

1Typically small, say $e = 3$. 
Iterative Extended GCD.

Example: \( p = 7, \ q = 11 \).
Iterative Extended GCD.

Example: \( p = 7, \ q = 11. \)

\( N = 77. \)
Iterative Extended GCD.

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

$N = 77$.

$(p - 1)(q - 1) = 60$
Iterative Extended GCD.

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

$N = 77$.

$(p - 1)(q - 1) = 60$

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

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

$N = 77$.

$(p - 1)(q - 1) = 60$

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

$\text{egcd}(7, 60)$. 

Iterative Extended GCD.

Example: \( p = 7, q = 11 \).

\( N = 77 \).

\((p - 1)(q - 1) = 60\)

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

\( \text{egcd}(7, 60) \).

\[
7(0) + 60(1) = 60
\]
Iterative Extended GCD.

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

$N = 77$.

$(p - 1)(q - 1) = 60$

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

$\text{egcd}(7,60)$.

\[
7(0) + 60(1) = 60
\]
\[
7(1) + 60(0) = 7
\]
Iterative Extended GCD.

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

$N = 77$.

$(p - 1)(q - 1) = 60$

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

\[
\text{egcd}(7, 60).
\]

\[
egin{align*}
7(0) + 60(1) & = 60 \\
7(1) + 60(0) & = 7 \\
7(-8) + 60(1) & = 4
\end{align*}
\]
Iterative Extended GCD.

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

$N = 77$.

$(p - 1)(q - 1) = 60$

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

$\text{egcd}(7, 60)$.

\[
\begin{align*}
7(0) + 60(1) &= 60 \\
7(1) + 60(0) &= 7 \\
7(-8) + 60(1) &= 4 \\
7(9) + 60(-1) &= 3
\end{align*}
\]
Iterative Extended GCD.

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

$N = 77$.

$(p - 1)(q - 1) = 60$

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

egcd$(7, 60)$.

\[
\begin{align*}
7(0) + 60(1) &= 60 \\
7(1) + 60(0) &= 7 \\
7(-8) + 60(1) &= 4 \\
7(9) + 60(-1) &= 3 \\
7(-17) + 60(2) &= 1 \\
\end{align*}
\]
Iterative Extended GCD.

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

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

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

$\text{egcd}(7, 60)$.

\[
\begin{align*}
7(0) + 60(1) &= 60 \\
7(1) + 60(0) &= 7 \\
7(-8) + 60(1) &= 4 \\
7(9) + 60(-1) &= 3 \\
7(-17) + 60(2) &= 1
\end{align*}
\]
Iterative Extended GCD.

Example: \( p = 7, \ q = 11 \).

\( N = 77 \).

\((p - 1)(q - 1) = 60\)

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

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

egcd(7,60).

\[
\begin{align*}
7(0) + 60(1) & = 60 \\
7(1) + 60(0) & = 7 \\
7(-8) + 60(1) & = 4 \\
7(9) + 60(-1) & = 3 \\
7(-17) + 60(2) & = 1
\end{align*}
\]

Confirm:
Iterative Extended GCD.

Example: \( p = 7, \ q = 11. \)

\[ N = 77. \]

\((p - 1)(q - 1) = 60\)

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

\( \text{egcd}(7, 60). \)

\[
7(0) + 60(1) = 60 \\
7(1) + 60(0) = 7 \\
7(-8) + 60(1) = 4 \\
7(9) + 60(-1) = 3 \\
7(-17) + 60(2) = 1
\]

Confirm: \(-119 + 120 = 1\)
Iterative Extended GCD.

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

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

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

\( \text{egcd}(7, 60) \).

\[
\begin{align*}
7(0) + 60(1) &= 60 \\
7(1) + 60(0) &= 7 \\
7(-8) + 60(1) &= 4 \\
7(9) + 60(-1) &= 3 \\
7(-17) + 60(2) &= 1 \\
\end{align*}
\]

Confirm: $-119 + 120 = 1$

\[d = e^{-1} = -17 = 43 \equiv (\text{mod } 60)\]
Encryption/Decryption Techniques.

Public Key: (77, 7)

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

Message: 2!

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

\[ D(51) = 51^43 \pmod{77} \]

uh oh!

Obvious way: 43 multiplications.

Ouch.

In general, \( O(N) \) or \( O(2^n) \) multiplications!
Encryption/Decryption Techniques.

Public Key: (77, 7)

Message: $2^2 = 2^e = 2^7 \equiv 128 \pmod{77} = 51 \pmod{77}$

uh oh!

Obvious way: 43 multiplications.

Ouch.

In general, $O(N)$ or $O(2^n)$ multiplications!
Encryption/Decryption Techniques.

Public Key: (77, 7)
Message Choices: {0, ..., 76}.
Public Key: (77, 7)
Message Choices: \{0, \ldots, 76\}.
Message: 2!

\begin{align*}
\text{Encrypt: } E(2) &= 2^e \equiv 128 \pmod{77} \\
\text{Decrypt: } D(51) &= 51^d \equiv 43 \pmod{77}
\end{align*}

Ugh!
Encryption/Decryption Techniques.

Public Key: (77, 7)
Message Choices: \{0, \ldots, 76\}.
Message: 2!
\[ E(2) \]
Public Key: (77, 7)
Message Choices: {0, . . . , 76}.
Message: 2!

\[ E(2) = 2^e \]
Encryption/Decryption Techniques.

Public Key: (77, 7)
Message Choices: \{0, \ldots, 76\}.
Message: 2!

\[ E(2) = 2^e = 2^7 \]
Encryption/Decryption Techniques.

Public Key: (77, 7)
Message Choices: {0, \ldots, 76}.
Message: 2!

\[ E(2) = 2^e = 2^7 \equiv 128 \pmod{77} \]
Encryption/Decryption Techniques.

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

Message: 2!

\[ E(2) = 2^e = 2^7 \equiv 128 \pmod{77} = 51 \pmod{77} \]
Public Key: (77, 7)
Message Choices: \{0, \ldots, 76\}.

Message: 2!

\[ E(2) = 2^e = 2^7 \equiv 128 \pmod{77} = 51 \pmod{77} \]
\[ D(51) = 51^{43} \pmod{77} \]
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!
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.
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.
Encryption/Decryption Techniques.

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

Message: 2!

\[ E(2) = 2^e = 2^7 \equiv 128 \pmod{77} \equiv 51 \pmod{77} \]
\[ D(51) = 51^{43} \pmod{77} \]

uh oh!

Obvious way: 43 multiplications. Ouch.

In general, \(O(N)\) or \(O(2^n)\) multiplications!
Repeated squaring.

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

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

Need to compute $51^{32}$.

$51^{2} = (51^{1}) \cdot (51^{1}) = 2601 \equiv 60 \pmod{77}$.

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

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

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

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

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

Decoding got the message back! 

Repeated squaring took 9 multiplications versus 43.
Repeated squaring.

Notice: $43 = 32 + 8 + 2 + 1$. 
Repeated squaring.

Notice: $43 = 32 + 8 + 2 + 1$. $51^{43}$
Repeated squaring.

Notice: \(43 = 32 + 8 + 2 + 1\). \(51^{43} = 51^{32+8+2+1}\)
Repeating squaring.

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

$51^1 \equiv 51 \pmod{77}$
Repeated squaring.

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

4 multiplications sort of...
Need to compute \(51^{32} \ldots 51^1\).?
\(51^1 \equiv 51\) (mod 77)
\(51^2 = \)
Repeated squaring.

\[
\text{Notice: } 43 = 32 + 8 + 2 + 1. \quad 51^{43} = 51^{32+8+2+1} = 51^{32} \cdot 51^8 \cdot 51^2 \cdot 51^1 \pmod{77}.
\]

4 multiplications sort of...

Need to compute \(51^{32} \cdots 51^1\).?

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

\[
51^2 = (51) \ast (51) = 2601 \equiv 60 \pmod{77}
\]
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 $51^{32} \ldots 51^1$?

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

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

$51^4 =$
Repeated squaring.

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

4 multiplications sort of...
Need to compute $51^{32} \ldots 51^1$.

$51^1 \equiv 51$ (mod 77)
$51^2 = (51) \ast (51) = 2601 \equiv 60$ (mod 77)
$51^4 = (51^2) \ast (51^2)$
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 \(51^{32} \ldots 51^1\).?

\[
51^1 \equiv 51 \pmod{77} \\
51^2 = (51) \cdot (51) = 2601 \equiv 60 \pmod{77} \\
51^4 = (51^2) \cdot (51^2) = 60 \cdot 60 = 3600 \equiv 58 \pmod{77}
\]
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 $51^{32} \ldots 51^1$.

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

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

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

$51^8 =$
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 $51^{32} \ldots 51^1$?

- $51^1 \equiv 51 \pmod{77}$
- $51^2 = (51) \cdot (51) = 2601 \equiv 60 \pmod{77}$
- $51^4 = (51^2) \cdot (51^2) = 60 \cdot 60 = 3600 \equiv 58 \pmod{77}$
- $51^8 = (51^4) \cdot (51^4)$
Repeated squaring.

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

4 multiplications sort of...
Need to compute \( 51^{32} \ldots 51^1 \)?

\( 51^1 \equiv 51 \) (mod 77)

\( 51^2 = (51) \cdot (51) = 2601 \equiv 60 \) (mod 77)

\( 51^4 = (51^2) \cdot (51^2) = 60 \cdot 60 = 3600 \equiv 58 \) (mod 77)

\( 51^8 = (51^4) \cdot (51^4) = 58 \cdot 58 = 3364 \equiv 53 \) (mod 77)
Repeated squaring.

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

4 multiplications sort of...

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

\[51^{1} \equiv 51 \pmod{77}\]
\[51^{2} = (51) \cdot (51) = 2601 \equiv 60 \pmod{77}\]
\[51^{4} = (51^{2}) \cdot (51^{2}) = 60 \cdot 60 = 3600 \equiv 58 \pmod{77}\]
\[51^{8} = (51^{4}) \cdot (51^{4}) = 58 \cdot 58 = 3364 \equiv 53 \pmod{77}\]
\[51^{16} = (51^{8}) \cdot (51^{8}) = 53 \cdot 53 = 2809 \equiv 37 \pmod{77}\]
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 $51^{32} \ldots 51^{1}$.

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

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

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

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

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

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

Decoding got the message back!
Repeated squaring.

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

4 multiplications sort of...

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

$51^{1} \equiv 51$ (mod 77)

$51^{2} = (51) \cdot (51) = 2601 \equiv 60$ (mod 77)

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

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

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

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

5 more multiplications.

Decoding got the message back!
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 $51^{32} \ldots 51^1$?

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

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

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

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

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

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

5 more multiplications.

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

\(51^{1} \equiv 51 \pmod{77}\)

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

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

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

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

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

5 more multiplications.

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

Decoding got the message back!
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 $51^{32} \ldots 51^1$.

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

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

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

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

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

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

5 more multiplications.

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

Decoding got the message back!

Repeated Squaring took 9 multiplications
Repeated squaring.

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

4 multiplications sort of...
Need to compute \( 51^{32} \ldots 51^1 \) ?

\[
\begin{align*}
51^1 & \equiv 51 \pmod{77} \\
51^2 &= (51) \cdot (51) = 2601 \equiv 60 \pmod{77} \\
51^4 &= (51^2) \cdot (51^2) = 60 \cdot 60 = 3600 \equiv 58 \pmod{77} \\
51^8 &= (51^4) \cdot (51^4) = 58 \cdot 58 = 3364 \equiv 53 \pmod{77} \\
51^{16} &= (51^8) \cdot (51^8) = 53 \cdot 53 = 2809 \equiv 37 \pmod{77} \\
51^{32} &= (51^{16}) \cdot (51^{16}) = 37 \cdot 37 = 1369 \equiv 60 \pmod{77}
\end{align*}
\]

5 more multiplications.

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

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 \(x^y\).
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: $x^1 = x \pmod{m}$. 
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: \(x^1 = x \pmod{m}\).

\[
x^y = x^{2(y/2) + \text{mod}(y,2)} = (x^2)^{y/2} x^y \pmod{2} \pmod{m}.
\]
(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: $x^1 = x \pmod{m}$.

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

The program computes the last expression using a recursive call with $x^2$ and $y/2$. 
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: $x^1 = x \pmod{m}$.

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

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

Note: $y/2$ is integer division.
Repeated Squaring: $x^y$

1. Compute $x^1, x^2, x^4, \ldots, x^{2^\left\lfloor \log y \right\rfloor}$.

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

Example: $43 = 101011$ in binary. $x^{43} = x^32 \times x^8 \times x^2 \times x^1$.

Modular Exponentiation: $x^y \mod N$.

All $n$-bit numbers. Repeated Squaring: $O(n)$ multiplications. $O(n^2)$ time per multiplication. $\Rightarrow O(n^3)$ time.

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

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

1. $x^y$: Compute $x^1$,
Repeated Squaring: $x^y$

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

1. $x^y$: Compute $x^1, x^2,$
Repeated Squaring: \( x^y \)

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

1. \( x^y \): Compute \( x^1, x^2, x^4 \),
Repeated Squaring: $x^y$

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

1. $x^y$: Compute $x^1, x^2, x^4, \ldots$, 

Modular Exponentiation: $x^y \mod N$.
Repeated Squaring: $x^y$

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

1. $x^y$: Compute $x^1, x^2, x^4, \ldots, x^{2^\lfloor \log y \rfloor}$. 

Modular Exponentiation: $x^y \mod N$. All $n$-bit numbers. Repeated Squaring: $O(n)$ multiplications. $O(n^2)$ time per multiplication. $\Rightarrow O(n^3)$ time. 

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

\[ x^{43} = x^{32} \cdot x^8 \cdot x^2 \cdot x^1. \]
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.
   \[ x^{43} = x^{32} \ast x^8 \ast x^2 \ast x^1. \]

Modular Exponentiation: $x^y \mod N$. 
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.
   $x^{43} = x^{32} \ast x^8 \ast x^2 \ast x^1$.

Modular Exponentiation: $x^y \mod N$. All $n$-bit numbers. Repeated Squaring:
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.
   \[ x^{43} = x^{32} \ast x^8 \ast x^2 \ast x^1. \]

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

\( O(n) \) multiplications.
Repeated Squaring: $x^y$

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

1. $x^y$: Compute $x^1, x^2, x^4, \ldots, x^{2^\lfloor \log y \rfloor}$.

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

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

$O(n)$ multiplications.
$O(n^2)$ time per multiplication.
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.
   
   
   $x^{43} = x^{32} \ast x^{8} \ast x^{2} \ast x^{1}$.

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: $x^y \mod N$
Repeated Squaring: $x^y$

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

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

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

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

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

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

Modular Exponentiation: $x^y \ mod \ N$. 
RSA is pretty fast.

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

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

Remember RSA encoding/decoding!
RSA is pretty fast.

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

Remember RSA encoding/decoding!

\[ E(m, (N, e)) = m^e \pmod{N}. \]
RSA is pretty fast.

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

Remember RSA encoding/decoding!

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

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

Remember RSA encoding/decoding!

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

\[ E(m, (N, e)) = m^e \pmod{N}. \]
Decoding.

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

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

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

\[ N = pq \]
Decoding.

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

\[ N = pq \text{ and } d = e^{-1} \pmod{(p-1)(q-1)}. \]
Decoding.

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

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

Want:
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?

\[ E(m, (N, e)) = m^e \pmod{N}. \]
Always decode correctly?

\[ E(m, (N, e)) = m^e \pmod{N}. \]
\[ D(m, (N, d)) = m^d \pmod{N}. \]
Always decode correctly?

\[ E(m, (N, e)) = m^e \pmod{N}. \]
\[ D(m, (N, d)) = m^d \pmod{N}. \]
Always decode correctly?

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

\[ N = pq \]
Always decode correctly?

\[ E(m, (N, e)) = m^e \pmod{N}. \]
\[ D(m, (N, d)) = m^d \pmod{N}. \]
\[ N = pq \text{ and } d = e^{-1} \pmod{(p-1)(q-1)}. \]
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:
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: \[ (m^e)^d = m^{ed} = m \pmod{N}. \]
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:
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:
\[ d = e^{-1} \pmod{(p-1)(q-1)} \iff ed = k(p-1)(q-1) + 1. \]
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...
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 \(a \not\equiv 0 \pmod{p}\),
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: \( (m^e)^d = m^{ed} = m \pmod{N} \).

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 \( a \not\equiv 0 \pmod{p} \),
\[ a^{p-1} \equiv 1 \pmod{p}. \]
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: \((m^e)^d = m^{ed} = m \pmod{N} \).

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 \( a \neq 0 \pmod{p} \),
\[ a^{p-1} \equiv 1 \pmod{p}. \]

\[ \implies a^{k(p-1)} \equiv 1 \pmod{p} \]
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 \( a \equiv 0 \pmod{p} \),
\[ a^{p-1} \equiv 1 \pmod{p}. \]

\[ \implies a^{k(p-1)} \equiv 1 \pmod{p} \implies \]
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: \((m^e)^d = m^{ed} = m \pmod{N}\).

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 \( a \not\equiv 0 \pmod{p} \),
\[ a^{p-1} \equiv 1 \pmod{p}. \]

\[ \implies a^{k(p-1)} \equiv 1 \pmod{p} \implies a^{k(p-1)+1} \]
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 \( a \not\equiv 0 \pmod{p} \),
\[ a^{p-1} \equiv 1 \pmod{p}. \]

\[ \implies a^{k(p-1)} \equiv 1 \pmod{p} \implies a^{k(p-1)+1} = a \pmod{p} \]
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: \( (m^e)^d = m^{ed} = m \pmod{N}. \)

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 \( a^{k(p-1)(q-1)+1} = a \pmod{pq} \).
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.
Fermat’s Little Theorem: For prime $p$, and $a \not\equiv 0 \pmod{p}$,
Fermat’s Little Theorem: For prime $p$, and $a \not\equiv 0 \pmod{p}$,
$$a^{p-1} \equiv 1 \pmod{p}.$$
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 $p$.

$$(a \cdot 1) \cdot (a \cdot 2) \cdots (a \cdot (p-1)) \equiv 1 \cdot 2 \cdots (p-1) \pmod{p},$$  

since multiplication is commutative.

$$a \cdot (p-1) \cdot (1 \cdots (p-1)) \equiv (1 \cdots (p-1)) \pmod{p}. $$  

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

$$a \cdot (p-1) \equiv 1 \pmod{p}.$$
Fermat’s Little Theorem: For prime $p$, and $a \not\equiv 0 \pmod{p}$,
\[ a^{p-1} \equiv 1 \pmod{p}. \]

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

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

$$(a \cdot 1) \cdot (a \cdot 2) \cdots (a \cdot (p-1)) \equiv 1 \cdot 2 \cdots (p-1) \pmod{p},$$
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 $p$. 

\[ (a \cdot 1) \cdot (a \cdot 2) \cdots (a \cdot (p - 1)) \equiv 1 \cdot 2 \cdots (p - 1) \pmod{p}, \]
Since multiplication is commutative.
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 \( p \).

\[
(a \cdot 1) \cdot (a \cdot 2) \cdots (a \cdot (p-1)) \equiv 1 \cdot 2 \cdots (p-1) \pmod{p},
\]
Since multiplication is commutative.

\[
a^{(p-1)}(1\cdots(p-1)) \equiv (1\cdots(p-1)) \pmod{p}.
\]
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 $p$.

\[ (a \cdot 1) \cdot (a \cdot 2) \cdots (a \cdot (p-1)) \equiv 1 \cdot 2 \cdots (p-1) \pmod{p}, \]

Since multiplication is commutative.

\[ a^{(p-1)}(1 \cdots (p-1)) \equiv (1 \cdots (p-1)) \pmod{p}. \]

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

\[(a \cdot 1) \cdot (a \cdot 2) \cdots (a \cdot (p-1)) \equiv 1 \cdot 2 \cdots (p-1) \pmod{p},\]

Since multiplication is commutative.

\[a^{(p-1)}(1\cdots (p-1)) \equiv (1\cdots (p-1)) \pmod{p}.

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

$$ (a \cdot 1) \cdot (a \cdot 2) \cdots (a \cdot (p-1)) \equiv 1 \cdot 2 \cdots (p-1) \pmod{p}, $$
Since multiplication is commutative.

$$ a^{(p-1)(1 \cdots (p-1))} \equiv (1 \cdots (p-1)) \pmod{p}. $$
Each of $2, \ldots (p-1)$ has an inverse modulo $p$, solve to get...

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

$$(a \cdot 1) \cdot (a \cdot 2) \cdots (a \cdot (p-1)) \equiv 1 \cdot 2 \cdots (p-1) \pmod{p},$$ 
Since multiplication is commutative. 

$$a^{(p-1)}(1 \cdots (p-1)) \equiv (1 \cdots (p-1)) \pmod{p}.$$ 
Each of $2, \ldots (p-1)$ has an inverse modulo $p$, solve to get... 

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


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

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

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

$$a^{1+b(p-1)} \equiv a^1 \ast (a^{p-1})^b$$
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

$$a^{1+b(p-1)} \equiv a^1 \cdot (a^{p-1})^b \equiv a \cdot (1)^b \equiv a \pmod{p}$$
Lemma 1: For any prime $p$ and any $a, b$, 
\[ a^{1+b(p-1)} \equiv a \pmod{p} \]
...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 $x, k$,
\[ x^{1+k(p-1)(q-1)} \equiv x \pmod{pq} \]
...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 $q.$
...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 $q$.
\[ x^{1+k(p-1)(q-1)} \equiv x \pmod{p} \]
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 $p.$
\[ x^{1+k(p-1)(q-1)} \equiv x \pmod{p} \]
...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 \( p \).
\[ x^{1+k(p-1)(q-1)} \equiv x \pmod{p} \]
...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 $q$.

\[ x^{1+k(q-1)(p-1)} - x \equiv 0 \pmod{(pq)} \]
...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 $q$.
\[ x^{1+k(q-1)(p-1)} - x \equiv 0 \pmod{(pq)} \implies x^{1+k(q-1)(p-1)} \equiv x \pmod{pq}. \]
...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 $q$.

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

**Lemma 2:** For any two different primes $p, q$ and any $x, k$,

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

$$D(E(x)) = (x^e)^d$$
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

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

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

where $ed \equiv 1 \pmod{(p - 1)(q - 1)} \implies ed = 1 + k(p - 1)(q - 1)$

\[ x^{ed} \equiv \]
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

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

where $ed \equiv 1 \pmod{(p-1)(q-1)} \implies ed = 1 + k(p-1)(q-1)$

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

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

\[ x^{ed} \equiv x^{k(p-1)(q-1)+1} \equiv x \pmod{pq}. \]
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.
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 $N \geq 17$

   $$\pi(N) \geq \frac{N}{\ln N}.$$
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 \( N \geq 17 \)

   \[
   \pi(N) \geq \frac{N}{\ln N}.
   \]

   Choosing randomly gives approximately \( \frac{1}{\ln N} \) chance of number being a prime. (How do you tell if it is prime?)
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 $N \geq 17$

   $$\pi(N) \geq \frac{N}{\ln N}.$$

   Choosing randomly gives approximately $1/(\ln N)$ chance of number being a prime. (How do you tell if it is prime? ... cs170.)
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 $N \geq 17$

   $$\pi(N) \geq \frac{N}{\ln N}.$$

   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..
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 \( N \geq 17 \)

\[
\pi(N) \geq \frac{N}{\ln N}.
\]

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 \( P \)).
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 $N \geq 17$

$$\pi(N) \geq \frac{N}{\ln N}.$$  

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.
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 $N \geq 17$
   
   \[
   \pi(N) \geq \frac{N}{\ln N}.
   \]

   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 $\gcd(e, (p-1)(q-1)) = 1$. 

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.
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 $N \geq 17$

   \[
   \pi(N) \geq \frac{N}{\ln N}.
   \]

   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.
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 \( N \geq 17 \)

   \[
   \pi(N) \geq \frac{N}{\ln N}.
   \]

   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 \( (p-1)(q-1) \).
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 $N \geq 17$

   $$\pi(N) \geq \frac{N}{\ln N}.$$

   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.
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 $N \geq 17$

   $\pi(N) \geq \frac{N}{\ln N}$.

   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.

1. Alice knows $p$ and $q$.
2. Bob only knows $N = pq$, and $e$.
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.
Security of RSA.

Security?

1. Alice knows $p$ and $q$.

2. Bob only knows, $N (= pq)$, and $e$. 
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 $N$. 
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 \( N \).
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 $N$. 
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.
If Bobs sends a message (Credit Card Number) to Alice,
If Bobs sends a message (Credit Card Number) to Alice, Eve sees it.
Much more to it.....

If Bobs sends a message (Credit Card Number) to Alice, Eve sees it.
Eve can send credit card again!!
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;
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.
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, $c$, 
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 $r$. 
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 $c$. 
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.
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.

Verisign:

Amazon \[ \rightarrow \] Browser.
Signatures using RSA.

Verisign:

Amazon → Browser.

Certificate Authority: Verisign, GoDaddy, DigiNotar,...
Signatures using RSA.

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

Verisign’s key: \( K_V = (N, e) \) and \( k_V = d \ (N = pq) \)

Amazon → Browser.

Verisign: \( k_V, K_V \)
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: \( K_v \).
Signatures using RSA.

Verisign: $k_V, K_V$

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

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

Browser “knows” Verisign’s public key: $K_V$.

Amazon Certificate: $C = \text{“I am Amazon. My public Key is } K_A.\text{”}$
Signatures using RSA.

Verisign: $k_v, K_v$

$[C, S_v(C)]$

Amazon $\leftrightarrow$ Browser. $K_v$

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

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

Browser “knows” Verisign’s public key: $K_v$.

Amazon Certificate: $C =$ “I am Amazon. My public Key is $K_A$.”

Versign signature of $C$: $S_v(C)$: $D(C, k_v) = C^d \mod N$. 
Signatures using RSA.

Verisign: $k_V, K_V$

$[C, S_V(C)]$

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 $C$: $S_V(C): D(C, k_V) = C^d \mod N$. 
Signatures using RSA.

Verisign: $k_V, K_V$

$[C, S_V(C)] ightarrow [C, S_V(C)]$

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: $[C, y]$
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: \( C = \text{“I am Amazon. My public Key is } K_A \text{.”} \)
Versign signature of \( C \): \( S_V(C) \): \( D(C, k_V) = C^d \mod N \).
Browser receives: \([C, y]\)
Checks \( E(y, K_V) = C? \)
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 $E(y, K_V) = C$?
$E(S_V(C), K_V)$
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: $C = \text{“I am Amazon. My public Key is } K_A \text{.”}$

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

Browser receives: $[C, y]$

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

$E(S_V(C), K_V) = (S_V(C))^e$
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: $C = \text{“I am Amazon. My public Key is } K_A.\text{”}$
Versign signature of $C$: $S_V(C): D(C, k_V) = C^d \mod N$.
Browser receives: $[C, y]$
Checks $E(y, K_V) = C$?
$E(S_V(C), K_V) = (S_V(C))^e = (C^d)^e$
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 $E(y, K_V) = C$?
$E(S_v(C), K_V) = (S_v(C))^e = (C^d)^e = C^{de}$
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 $E(y, K_V) = C$?

$E(S_V(C), K_V) = (S_V(C))^e = (C^d)^e = C^{de} = C \pmod{N}$
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: \( C = \text{“I am Amazon. My public Key is } K_A.\text{”} \)

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 \( C \)!
Signatures using RSA.

Verisign: $k_V, K_V$

$[C, S_V(C)]$

C = $E(S_V(C), k_V)$?

$[C, S_V(C)]$

Amazon ↔ Browser. $K_V$

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

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

Browser “knows” Verisign’s public key: $K_V$.

Amazon Certificate: $C = \text{“I am Amazon. My public Key is } K_A.\text{”}$

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:

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

Signature scheme:

\[E\left(D\left(C, k\right), K\right) = (C^d)^e \mod N = C.\]
RSA

Public Key Cryptography:

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

Signature scheme:

\[ E(D(C, k), K) = (C^d)^e \mod N = C. \]
Public Key Cryptography:

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

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

Signature scheme:
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?
Get CA to certify fake certificates: Microsoft Corporation.
Other Eve.

Get CA to certify fake certificates: Microsoft Corporation.
2001..Doh.
Get CA to certify fake certificates: Microsoft Corporation.
2001..Doh.
... and August 28, 2011 announcement.
Other Eve.

Get CA to certify fake certificates: Microsoft Corporation.
2001..Doh.
... and August 28, 2011 announcement.
DigiNotar Certificate issued for Microsoft!!!
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 ...
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 ⇒ efficiency.

Fermat’s Theorem ⇒ correctness.

Good for Encryption and Signature Schemes.
Summary.

Public-Key Encryption.

RSA Scheme:

\[ N = pq \]

\[ d = e^{-1} \pmod{(p-1)(q-1)} \]

\[ E(x) = x^e \pmod{N} \]

\[ D(y) = y^d \pmod{N} \]

Repeated Squaring ⇒ efficiency.

Fermat's Theorem ⇒ correctness.

Good for Encryption and Signature Schemes.
Public-Key Encryption.

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

Fermat’s Theorem \(\Rightarrow\) correctness.
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
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 \text{efficiency.}

Fermat’s Theorem \implies \text{correctness.}

Good for Encryption and Signature Schemes.