CS70: New Discussion Format

Small group:

- Three modes of working.
  - (A) Individual working.
  - (B) Pairs working together.
  - (C) Pairs: one works/one forces talking.

Supported by course staff and course volunteers.

Why?

- It works better for learning.
- Evidence:
  - (1) Experience. (years and years, faculty agree.)
  - (2) Literature.

- Students hate it.
- Students happy (in the moment): negatively correlated to learning.
- See marshmallow test. Delayed gratification.

- Our job is to have you learn.
- We would like you to be "happy" in the moment.
- But the result is what is important.

Be nice to the TA's. It's not them. It's the profs.
CS70: New Discussion Format

Small group:

Three modes of working.

(A) Individual working.

(B) Pairs working together.

(C) Pairs: one works/one forces talking.

Supported by course staff and course volunteers.

Why?

It works better for learning.

Evidence:

(1) Experience. (years and years, faculty agree.)

(2) Literature.

Students hate it.

Students happy (in the moment): negatively correlated to learning.

See marshmallow test. Delayed gratification.

Our job is to have you learn.

We would like you to be "happy" in the moment.

But the result is what is important.

Be nice to the TA's. It's not them. It's the profs.
CS70: New Discussion Format

Small group:

Three modes of working.

(A) Individual working.

Evidence:

1. Experience. (years and years, faculty agree.)
2. Literature.

Students hate it.

Students happy (in the moment): negatively correlated to learning.

See marshmallow test. Delayed gratification.

Our job is to have you learn. We would like you to be "happy" in the moment. But the result is what is important. Be nice to the TA's. It's not them. It's the profs.
CS70: New Discussion Format

Small group:

Three modes of working.
(A) Individual working.
(B) Pairs working together.
CS70: New Discussion Format

Small group:

Three modes of working.
  (A) Individual working.
  (B) Pairs working together.
  (C) Pairs: one works/one forces talking.
CS70: New Discussion Format

Small group:

Three modes of working.
(A) Individual working.
(B) Pairs working together.
(C) Pairs: one works/one forces talking.

Supported by course staff and course volunteers.
CS70: New Discussion Format

Small group:

Three modes of working.
(A) Individual working.
(B) Pairs working together.
(C) Pairs: one works/one forces talking.

Supported by course staff and course volunteers.

Why?
CS70: New Discussion Format

Small group:

Three modes of working.
(A) Individual working.
(B) Pairs working together.
(C) Pairs: one works/one forces talking.

Supported by course staff and course volunteers.

Why? It works better for learning.
CS70: New Discussion Format

Small group:

Three modes of working.
(A) Individual working.
(B) Pairs working together.
(C) Pairs: one works/one forces talking.

Supported by course staff and course volunteers.

Why? It works better for learning.

Evidence:
CS70: New Discussion Format

Small group:

Three modes of working.
(A) Individual working.
(B) Pairs working together.
(C) Pairs: one works/one forces talking.

Supported by course staff and course volunteers.

Why? It works better for learning.

Evidence:
(1) Experience. (years and years, faculty agree.)
CS70: New Discussion Format

Small group:

Three modes of working.
  (A) Individual working.
  (B) Pairs working together.
  (C) Pairs: one works/one forces talking.

Supported by course staff and course volunteers.

Why? It works better for learning.

Evidence:
  (1) Experience. (years and years, faculty agree.)
  (2) Literature.
CS70: New Discussion Format

Small group:

Three modes of working.
(A) Individual working.
(B) Pairs working together.
(C) Pairs: one works/one forces talking.

Supported by course staff and course volunteers.

Why? It works better for learning.

Evidence:
(1) Experience. (years and years, faculty agree.)
(2) Literature.
CS70: New Discussion Format

Small group:

Three modes of working.
   (A) Individual working.
   (B) Pairs working together.
   (C) Pairs: one works/one forces talking.

Supported by course staff and course volunteers.

Why?  It works better for learning.

Evidence:
   (1) Experience. (years and years, faculty agree.)
   (2) Literature.

Students hate it.
CS70: New Discussion Format

Small group:

Three modes of working.
(A) Individual working.
(B) Pairs working together.
(C) Pairs: one works/one forces talking.

Supported by course staff and course volunteers.

Why? It works better for learning.

Evidence:
(1) Experience. (years and years, faculty agree.)
(2) Literature.

Students hate it.
Students happy (in the moment):
CS70: New Discussion Format

Small group:

Three modes of working.

(A) Individual working.
(B) Pairs working together.
(C) Pairs: one works/one forces talking.

Supported by course staff and course volunteers.

Why? It works better for learning.

Evidence:

(1) Experience. (years and years, faculty agree.)
(2) Literature.

Students hate it.

Students happy (in the moment): negatively correlated to learning.
CS70: New Discussion Format

Small group:

Three modes of working.
(A) Individual working.
(B) Pairs working together.
(C) Pairs: one works/one forces talking.

Supported by course staff and course volunteers.

Why? It works better for learning.

Evidence:
(1) Experience. (years and years, faculty agree.)
(2) Literature.

Students hate it.
Students happy(in the moment): negatively correlated to learning.
See marshmallow test.
CS70: New Discussion Format

Small group:

Three modes of working.
  (A) Individual working.
  (B) Pairs working together.
  (C) Pairs: one works/one forces talking.

Supported by course staff and course volunteers.

Why?  It works better for learning.

Evidence:
  (1) Experience. (years and years, faculty agree.)
  (2) Literature.

Students hate it.
  Students happy(in the moment): negatively correlated to learning.
    See marshmallow test. Delayed gratification.
CS70: New Discussion Format

Small group:

Three modes of working.
(A) Individual working.
(B) Pairs working together.
(C) Pairs: one works/one forces talking.

Supported by course staff and course volunteers.

Why? It works better for learning.

Evidence:
(1) Experience. (years and years, faculty agree.)
(2) Literature.

Students hate it.
Students happy (in the moment): negatively correlated to learning.
See marshmallow test. Delayed gratification.

Our job is to have you learn.
CS70: New Discussion Format

Small group:

Three modes of working.
  (A) Individual working.
  (B) Pairs working together.
  (C) Pairs: one works/one forces talking.

Supported by course staff and course volunteers.

Why? It works better for learning.

Evidence:
  (1) Experience. (years and years, faculty agree.)
  (2) Literature.

Students hate it.
  Students happy (in the moment): negatively correlated to learning.
    See marshmallow test. Delayed gratification.

Our job is to have you learn.
  We would like you to be “happy”
CS70: New Discussion Format

Small group:

Three modes of working.
(A) Individual working.
(B) Pairs working together.
(C) Pairs: one works/one forces talking.

Supported by course staff and course volunteers.

Why? It works better for learning.

Evidence:
(1) Experience. (years and years, faculty agree.)
(2) Literature.

Students hate it.
Students happy (in the moment): negatively correlated to learning.
See marshmallow test. Delayed gratification.

Our job is to have you learn.
We would like you to be “happy” in the moment.
CS70: New Discussion Format

Small group:

Three modes of working.
(A) Individual working.
(B) Pairs working together.
(C) Pairs: one works/one forces talking.

Supported by course staff and course volunteers.

Why? It works better for learning.

Evidence:
(1) Experience. (years and years, faculty agree.)
(2) Literature.

Students hate it.
Students happy (in the moment): negatively correlated to learning.
See marshmallow test. Delayed gratification.

Our job is to have you learn.
We would like you to be “happy” in the moment.
But the result is what is important.
CS70: New Discussion Format

Small group:

Three modes of working.
(A) Individual working.
(B) Pairs working together.
(C) Pairs: one works/one forces talking.

Supported by course staff and course volunteers.

Why? It works better for learning.

Evidence:
(1) Experience. (years and years, faculty agree.)
(2) Literature.

Students hate it.
Students happy (in the moment): negatively correlated to learning.
See marshmallow test. Delayed gratification.

Our job is to have you learn.
We would like you to be “happy” in the moment.
But the result is what is important.
CS70: New Discussion Format

Small group:

Three modes of working.
(A) Individual working.
(B) Pairs working together.
(C) Pairs: one works/one forces talking.

Supported by course staff and course volunteers.

Why? It works better for learning.

Evidence:
(1) Experience. (years and years, faculty agree.)
(2) Literature.

Students hate it.
Students happy (in the moment): negatively correlated to learning.
See marshmallow test. Delayed gratification.

Our job is to have you learn.
We would like you to be “happy” in the moment.
But the result is what is important.

Be nice to the TA’s.
CS70: New Discussion Format

Small group:

Three modes of working.
(A) Individual working.
(B) Pairs working together.
(C) Pairs: one works/one forces talking.

Supported by course staff and course volunteers.

Why?  It works better for learning.

Evidence:
(1) Experience. (years and years, faculty agree.)
(2) Literature.

Students hate it.
Students happy (in the moment): negatively correlated to learning.
See marshmallow test. Delayed gratification.

Our job is to have you learn.
We would like you to be “happy” in the moment.
But the result is what is important.

Be nice to the TA’s. It’s not them.
CS70: New Discussion Format

Small group:

Three modes of working.
  (A) Individual working.
  (B) Pairs working together.
  (C) Pairs: one works/one forces talking.

Supported by course staff and course volunteers.

Why?  It works better for learning.

Evidence:
  (1) Experience. (years and years, faculty agree.)
  (2) Literature.

Students hate it.
  Students happy (in the moment): negatively correlated to learning.
    See marshmallow test. Delayed gratification.

Our job is to have you learn.
  We would like you to be “happy” in the moment.
  But the result is what is important.

Be nice to the TA’s. It’s not them. It’s the profs.
Do you remember the first lecture?
Do you remember the first lecture?

Veritassium on Khan
Do you remember the first lecture?

Veritassium on Khan

![Bar charts](attachment:image.png)

**Fig. 1.** Final recall (a) after repeatedly studying a text in four study periods (SSSS condition), reading a text in three study periods and then recalling it in one retrieval period (SSSR condition), or reading a text in one study period and then repeatedly recalling it in three retrieval periods (SRRR condition). Judgments of learning (b) were made on a 7-point scale, where 7 indicated that students believed they would remember material very well. The data presented in these graphs are adapted from Experiment 2 of Roediger and Karpicke (2006b). The pattern of students’ metacognitive judgments of learning (predicted recall) was exactly the opposite of the pattern of students’ actual long-term retention.
1. Public Key Cryptography

2. RSA system
   2.1 Efficiency: Repeated Squaring.
   2.2 Correctness: Fermat’s Theorem.
   2.3 Construction.

3. Warnings.
Simple Chinese Remainder Theorem.

My love is won.

Zero and One.

Nothing and nothing done.

Find $x = a \pmod{m}$ and $x = b \pmod{n}$ where $\gcd(m, n) = 1$.

**CRT Thm:** There is a unique solution $x \pmod{mn}$.

**Proof (solution exists):** Consider $u = n(n - 1 \pmod{m})$.

$u = 0 \pmod{n}$

$u = 1 \pmod{m}$

Consider $v = m(m - 1 \pmod{n})$.

$v = 1 \pmod{n}$

$v = 0 \pmod{m}$

Let $x = au + bv$.

$x = a \pmod{m}$ since $bv = 0 \pmod{m}$ and $au = a \pmod{m}$

$x = b \pmod{n}$ since $au = 0 \pmod{n}$ and $bv = b \pmod{n}$

This shows there is a solution.
Simple Chinese Remainder Theorem.

My love is won.
Simple Chinese Remainder Theorem.

My love is won. Zero and One.
Simple Chinese Remainder Theorem.

My love is won. Zero and One. Nothing and nothing done.
Simple Chinese Remainder Theorem.

My love is won. Zero and One. Nothing and nothing done.
Simple Chinese Remainder Theorem.

My love is won. Zero and One. Nothing and nothing done.

Find \( x = a \pmod{m} \) and \( x = b \pmod{n} \)
Simple Chinese Remainder Theorem.

My love is won. Zero and One. Nothing and nothing done.

Find $x = a \pmod{m}$ and $x = b \pmod{n}$ where $\gcd(m, n)=1$. 
Simple Chinese Remainder Theorem.

My love is won. Zero and One. Nothing and nothing done.

Find $x = a \pmod{m}$ and $x = b \pmod{n}$ where $\gcd(m,n)=1$.

**CRT Thm**: There is a unique solution $x \pmod{mn}$. 

Simple Chinese Remainder Theorem.

My love is won. Zero and One. Nothing and nothing done.

Find $x = a \pmod{m}$ and $x = b \pmod{n}$ where $\gcd(m, n) = 1$.

**CRT Thm:** There is a unique solution $x \pmod{mn}$.

**Proof (solution exists):**
Simple Chinese Remainder Theorem.

My love is won. Zero and One. Nothing and nothing done.

Find \( x = a \pmod{m} \) and \( x = b \pmod{n} \) where \( \gcd(m, n) = 1 \).

**CRT Thm:** There is a unique solution \( x \pmod{mn} \).

**Proof (solution exists):**
Consider \( u = n(n^{-1} \pmod{m}) \).
My love is won. Zero and One. Nothing and nothing done.

Find \( x = a \pmod{m} \) and \( x = b \pmod{n} \) where \( \gcd(m, n) = 1 \).

**CRT Thm:** There is a unique solution \( x \pmod{mn} \).

**Proof (solution exists):**
Consider \( u = n(n^{-1} \pmod{m}) \).

\[ u = 0 \pmod{n} \]
Simple Chinese Remainder Theorem.

My love is won. Zero and One. Nothing and nothing done.

Find \( x = a \pmod{m} \) and \( x = b \pmod{n} \) where \( \gcd(m, n) = 1 \).

**CRT Thm:** There is a unique solution \( x \pmod{mn} \).

**Proof (solution exists):**
Consider \( u = n(n^{-1} \pmod{m}) \).

\[
\begin{align*}
  u &= 0 \pmod{n} & u &= 1 \pmod{m}
\end{align*}
\]
Simple Chinese Remainder Theorem.

My love is won. Zero and One. Nothing and nothing done.

Find \( x = a \pmod{m} \) and \( x = b \pmod{n} \) where \( \gcd(m,n) = 1 \).

**CRT Thm:** There is a unique solution \( x \pmod{mn} \).

**Proof (solution exists):**
Consider \( u = n(n^{-1} \pmod{m}) \).

\[
\begin{align*}
\quad u & = 0 \pmod{n} \quad \quad u = 1 \pmod{m} \\
\end{align*}
\]
Consider \( v = m(m^{-1} \pmod{n}) \).
My love is won. Zero and One. Nothing and nothing done.

Find \( x = a \pmod{m} \) and \( x = b \pmod{n} \) where \( \gcd(m, n) = 1 \).

**CRT Thm:** There is a unique solution \( x \pmod{mn} \).

**Proof (solution exists):**
Consider \( u = n(n^{-1} \pmod{m}) \).
\[ u = 0 \pmod{n} \quad u = 1 \pmod{m} \]

Consider \( v = m(m^{-1} \pmod{n}) \).
\[ v = 1 \pmod{n} \]
Simple Chinese Remainder Theorem.

My love is won. Zero and One. Nothing and nothing done.

Find \( x = a \pmod{m} \) and \( x = b \pmod{n} \) where \( \gcd(m, n) = 1 \).

**CRT Thm:** There is a unique solution \( x \pmod{mn} \).

**Proof (solution exists):**
Consider \( u = n(n^{-1} \pmod{m}) \).

\[
\begin{align*}
u &= 0 \pmod{n} \\
u &= 1 \pmod{m}
\end{align*}
\]

Consider \( v = m(m^{-1} \pmod{n}) \).

\[
\begin{align*}
v &= 1 \pmod{n} \\
v &= 0 \pmod{m}
\end{align*}
\]
Simple Chinese Remainder Theorem.

My love is won. Zero and One. Nothing and nothing done.

Find \( x = a \pmod m \) and \( x = b \pmod n \) where \( \gcd(m, n) = 1 \).

**CRT Thm:** There is a unique solution \( x \pmod {mn} \).

**Proof (solution exists):**
Consider \( u = n(n^{-1} \pmod m) \).
\[
\begin{align*}
    u &= 0 \pmod n \quad u = 1 \pmod m \\
\end{align*}
\]
Consider \( v = m(m^{-1} \pmod n) \).
\[
\begin{align*}
    v &= 1 \pmod n \quad v = 0 \pmod m \\
\end{align*}
\]
Let \( x = au + bv \).
My love is won. Zero and One. Nothing and nothing done.

Find \( x = a \pmod{m} \) and \( x = b \pmod{n} \) where \( \gcd(m, n) = 1 \).

**CRT Thm:** There is a unique solution \( x \pmod{mn} \).

**Proof (solution exists):**

Consider \( u = n(n^{-1} \pmod{m}) \).

\[
\begin{align*}
    u & = 0 \pmod{n} & u & = 1 \pmod{m}
\end{align*}
\]

Consider \( v = m(m^{-1} \pmod{n}) \).

\[
\begin{align*}
    v & = 1 \pmod{n} & v & = 0 \pmod{m}
\end{align*}
\]

Let \( x = au + bv \).

\[
\begin{align*}
    x & = a \pmod{m}
\end{align*}
\]
Simple Chinese Remainder Theorem.

My love is won. Zero and One. Nothing and nothing done.

Find \( x = a \pmod{m} \) and \( x = b \pmod{n} \) where \( \gcd(m, n) = 1 \).

**CRT Thm:** There is a unique solution \( x \pmod{mn} \).

**Proof (solution exists):**

Consider \( u = n(n^{-1} \pmod{m}) \).

\[
\begin{align*}
  u &\equiv 0 \pmod{n} \\
  u &\equiv 1 \pmod{m}
\end{align*}
\]

Consider \( v = m(m^{-1} \pmod{n}) \).

\[
\begin{align*}
  v &\equiv 1 \pmod{n} \\
  v &\equiv 0 \pmod{m}
\end{align*}
\]

Let \( x = au + bv \).

\[
\begin{align*}
  x &\equiv a \pmod{m} \quad \text{since} \quad bv = 0 \pmod{m} \quad \text{and} \quad au = a \pmod{m}
\end{align*}
\]
My love is won. Zero and One. Nothing and nothing done.

Find $x = a \pmod{m}$ and $x = b \pmod{n}$ where $\gcd(m,n)=1$.

**CRT Thm:** There is a unique solution $x \pmod{mn}$.

**Proof (solution exists):**
Consider $u = n(n^{-1} \pmod{m})$.

- $u = 0 \pmod{n}$
- $u = 1 \pmod{m}$

Consider $v = m(m^{-1} \pmod{n})$.

- $v = 1 \pmod{n}$
- $v = 0 \pmod{m}$

Let $x = au + bv$.

- $x = a \pmod{m}$ since $bv = 0 \pmod{m}$ and $au = a \pmod{m}$
My love is won. Zero and One. Nothing and nothing done.

Find $x = a \pmod{m}$ and $x = b \pmod{n}$ where $\gcd(m, n)=1$.

**CRT Thm:** There is a unique solution $x \pmod{mn}$.

**Proof (solution exists):**
Consider $u = n(n^{-1} \pmod{m})$.
- $u = 0 \pmod{n}$         $u = 1 \pmod{m}$

Consider $v = m(m^{-1} \pmod{n})$.
- $v = 1 \pmod{n}$         $v = 0 \pmod{m}$

Let $x = au + bv$.
- $x = a \pmod{m}$ since $bv = 0 \pmod{m}$ and $au = a \pmod{m}$
- $x = b \pmod{n}$
My love is won. Zero and One. Nothing and nothing done.

Find \( x = a \pmod{m} \) and \( x = b \pmod{n} \) where \( \gcd(m, n)=1 \).

**CRT Thm:** There is a unique solution \( x \pmod{mn} \).

**Proof (solution exists):**
Consider \( u = n(n^{-1} \pmod{m}) \).
\[
\begin{align*}
  u &= 0 \pmod{n} & u &= 1 \pmod{m} \\
\end{align*}
\]
Consider \( v = m(m^{-1} \pmod{n}) \).
\[
\begin{align*}
  v &= 1 \pmod{n} & v &= 0 \pmod{m} \\
\end{align*}
\]
Let \( x = au + bv \).
\[
\begin{align*}
  x &= a \pmod{m} & \text{since } bv &= 0 \pmod{m} \text{ and } au = a \pmod{m} \\
  x &= b \pmod{n} & \text{since } au &= 0 \pmod{n} \text{ and } bv = b \pmod{n} \\
\end{align*}
\]
Simple Chinese Remainder Theorem.

My love is won. Zero and One. Nothing and nothing done.

Find \( x = a \pmod{m} \) and \( x = b \pmod{n} \) where \( \gcd(m, n)=1 \).

**CRT Thm:** There is a unique solution \( x \pmod{mn} \).

**Proof (solution exists):**
Consider \( u = n(n^{-1} \pmod{m}) \).
\[
  u = 0 \pmod{n} \quad \text{and} \quad u = 1 \pmod{m}
\]
Consider \( v = m(m^{-1} \pmod{n}) \).
\[
  v = 1 \pmod{n} \quad \text{and} \quad v = 0 \pmod{m}
\]
Let \( x = au + bv \).
\[
  x = a \pmod{m} \quad \text{since} \quad bv = 0 \pmod{m} \quad \text{and} \quad au = a \pmod{m}
\]
\[
  x = b \pmod{n} \quad \text{since} \quad au = 0 \pmod{n} \quad \text{and} \quad bv = b \pmod{n}
\]
This shows there is a solution.
Simple Chinese Remainder Theorem.

**CRT Thm:** There is a unique solution \( x \) (mod \( mn \)).
Simple Chinese Remainder Theorem.

**CRT Thm:** There is a unique solution $x \pmod{mn}$. 
Simple Chinese Remainder Theorem.

**CRT Thm:** There is a unique solution \( x \pmod{mn} \).

**Proof (uniqueness):**
Simple Chinese Remainder Theorem.

**CRT Thm:** There is a unique solution $x \pmod{mn}$.

**Proof (uniqueness):**
If not, two solutions, $x$ and $y$.
Simple Chinese Remainder Theorem.

**CRT Thm:** There is a unique solution \( x \pmod{mn} \).

**Proof (uniqueness):**
If not, two solutions, \( x \) and \( y \).
Simple Chinese Remainder Theorem.

**CRT Thm:** There is a unique solution $x \pmod{mn}$.

**Proof (uniqueness):**
If not, two solutions, $x$ and $y$.

$$(x - y) \equiv 0 \pmod{m} \text{ and } (x - y) \equiv 0 \pmod{n}.$$
CRT Thm: There is a unique solution $x \pmod{mn}$.

Proof (uniqueness):
If not, two solutions, $x$ and $y$.

$$(x - y) \equiv 0 \pmod{m} \text{ and } (x - y) \equiv 0 \pmod{n}.$$  

$\implies (x - y)$ is multiple of $m$ and $n$
**Simple Chinese Remainder Theorem.**

**CRT Thm:** There is a unique solution $x \pmod{mn}$.

**Proof (uniqueness):**
If not, two solutions, $x$ and $y$.

$$(x - y) \equiv 0 \pmod{m} \text{ and } (x - y) \equiv 0 \pmod{n}.$$  

$\implies (x - y)$ is multiple of $m$ and $n$ 

$\gcd(m, n) = 1 \implies$ no common primes in factorization $m$ and $n$
Simple Chinese Remainder Theorem.

**CRT Thm:** There is a unique solution \( x \) (mod \( mn \)).

**Proof (uniqueness):**
If not, two solutions, \( x \) and \( y \).

\[
(x - y) \equiv 0 \pmod{m} \quad \text{and} \quad (x - y) \equiv 0 \pmod{n}.
\]

\[
\implies (x - y) \text{ is multiple of } m \text{ and } n
\]

\[
\gcd(m, n) = 1 \implies \text{no common primes in factorization } m \text{ and } n
\]

\[
\implies mn|(x - y)
\]
Simple Chinese Remainder Theorem.

**CRT Thm:** There is a unique solution $x \pmod{mn}$.

**Proof (uniqueness):**
If not, two solutions, $x$ and $y$.

$$(x - y) \equiv 0 \pmod{m} \text{ and } (x - y) \equiv 0 \pmod{n}.$$  

$\implies (x - y)$ is multiple of $m$ and $n$

$\gcd(m, n) = 1 \implies$ no common primes in factorization $m$ and $n$

$\implies mn | (x - y)$

$\implies x - y \geq mn$
Simple Chinese Remainder Theorem.

**CRT Thm:** There is a unique solution $x \pmod{mn}$.

**Proof (uniqueness):**
If not, two solutions, $x$ and $y$.

$$(x - y) \equiv 0 \pmod{m} \text{ and } (x - y) \equiv 0 \pmod{n}.$$  

$\implies (x - y)$ is multiple of $m$ and $n$

$\gcd(m, n) = 1 \implies$ no common primes in factorization $m$ and $n$

$\implies mn|(x - y)$

$\implies x - y \geq mn \implies x, y \notin \{0, \ldots, mn - 1\}$. 

Simple Chinese Remainder Theorem.

**CRT Thm:** There is a unique solution $x \pmod{mn}$.

**Proof (uniqueness):**
If not, two solutions, $x$ and $y$.

$$(x - y) \equiv 0 \pmod{m} \text{ and } (x - y) \equiv 0 \pmod{n}.$$  

$\implies (x - y)$ is multiple of $m$ and $n$  

$m, n$ are coprime  

$\implies mn | (x - y)$  

$\implies x - y \geq mn \implies x, y \not\in \{0, \ldots, mn - 1\}.$

Thus, only one solution modulo $mn$. 
**Simple Chinese Remainder Theorem.**

**CRT Thm:** There is a unique solution $x \pmod{mn}$.

**Proof (uniqueness):**
If not, two solutions, $x$ and $y$.

$$(x - y) \equiv 0 \pmod{m} \text{ and } (x - y) \equiv 0 \pmod{n}.$$  

$\implies (x - y)$ is multiple of $m$ and $n$

$\gcd(m, n) = 1 \implies$ no common primes in factorization $m$ and $n$

$\implies mn|(x - y)$

$\implies x - y \geq mn \implies x, y \notin \{0, \ldots, mn - 1\}$.

Thus, only one solution modulo $mn$. $\square$
Isomorphisms.

Bijection:

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

Simplified Chinese Remainder Theorem:

If $\gcd(n, m) = 1$, there is unique $x \pmod{mn}$ where $x = a \pmod{m}$ and $x = b \pmod{n}$.

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:

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

Bijection:

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

**Simplified Chinese Remainder Theorem:**

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:

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

**Simplified Chinese Remainder Theorem:**

If \( \gcd(n, m) = 1 \), there is unique \( x \pmod{mn} \) where

\[ x = a \pmod{m} \] and \( x = b \pmod{n} \).

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

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

Now consider:

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

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

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

Is it \( 0 \pmod{5} \)?

Yes!

Is it \( 2 \pmod{9} \)?

Yes!

Isomorphism:

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

Bijection:

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

**Simplified Chinese Remainder Theorem:**

If \( \gcd(n, m) = 1 \), there is unique \( x \pmod{mn} \) where

\[ x = a \pmod{m} \] and \( x = b \pmod{n} \).

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

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

**Simplified Chinese Remainder Theorem:**
If \( \gcd(n, m) = 1 \), there is unique \( x \pmod{mn} \) where
\[ x = a \pmod{m} \text{ and } x = b \pmod{n}. \]

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:

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

**Simplified Chinese Remainder Theorem:**

If \( \gcd(n, m) = 1 \), there is unique \( x \pmod{mn} \) where
- \( x = a \pmod{m} \)
- \( x = b \pmod{n} \).

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:

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

**Simplified Chinese Remainder Theorem:**

If \( \gcd(n, m) = 1 \), there is unique \( x \pmod{mn} \) where
\[ x = a \pmod{m} \] and \( x = b \pmod{n} \).

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:**

If \( \gcd(n, m) = 1 \), there is unique \( x \pmod{mn} \) where
\[ x = a \pmod{m} \text{ and } x = b \pmod{n}. \]

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:

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

**Simplified Chinese Remainder Theorem:**

If \( \gcd(n, m) = 1 \), there is unique \( x \pmod{mn} \) where

\[ x = a \pmod{m} \text{ and } x = b \pmod{n}. \]

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:

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

**Simplified Chinese Remainder Theorem:**

If \( \gcd(n, m) = 1 \), there is unique \( x \pmod{mn} \) where

\[ x = a \pmod{m} \] and \( x = b \pmod{n} \).

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:**

If \( \gcd(n, m) = 1 \), there is a unique \( x \pmod{mn} \) where

\[ x = a \pmod{m} \text{ and } x = b \pmod{n}. \]

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:

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

**Simplified Chinese Remainder Theorem:**

If \( \gcd(n, m) = 1 \), there is unique \( x \pmod{mn} \) where

\[ x = a \pmod{m} \text{ and } x = b \pmod{n}. \]

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:

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

**Simplified Chinese Remainder Theorem:**

If \( \gcd(n, m) = 1 \), there is unique \( x \pmod{mn} \) where

\[ x = a \pmod{m} \text{ and } x = b \pmod{n}. \]

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:**

If \( \gcd(n, m) = 1 \), there is unique \( x \pmod{mn} \) where

\[ x = a \pmod{m} \text{ and } x = b \pmod{n}. \]

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

**Simplified Chinese Remainder Theorem:**

If \( \gcd(n, m) = 1 \), there is unique \( x \pmod{mn} \) where

\[ x = a \pmod{m} \quad \text{and} \quad x = b \pmod{n}. \]

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

**Simplified Chinese Remainder Theorem:**
If \( \gcd(n, m) = 1 \), there is unique \( x \pmod{mn} \) where
\[ x = a \pmod{m} \text{ and } x = b \pmod{n}. \]

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

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

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:**

If \( \gcd(n, m) = 1 \), there is unique \( x \pmod{mn} \) where
\[ x = a \pmod{m} \text{ and } x = b \pmod{n}. \]

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 \((\pmod{45})\)!
$x = 5 \mod 7$ and $x = 5 \mod 6$

$y = 4 \mod 7$ and $y = 3 \mod 6$
\( x = 5 \mod 7 \text{ and } x = 5 \mod 6 \)
\( y = 4 \mod 7 \text{ and } y = 3 \mod 6 \)

What’s true?
$x = 5 \mod 7 \text{ and } x = 5 \mod 6$
$y = 4 \mod 7 \text{ and } y = 3 \mod 6$

What’s true?

(A) $x + y = 2 \mod 7$
(B) $x + y = 2 \mod 6$
(C) $xy = 3 \mod 6$
(D) $xy = 6 \mod 7$
(E) $x = 5 \mod 42$
(F) $y = 39 \mod 42$
$x = 5 \mod 7 \text{ and } x = 5 \mod 6$
$y = 4 \mod 7 \text{ and } y = 3 \mod 6$

What’s true?

(A) $x + y = 2 \mod 7$
(B) $x + y = 2 \mod 6$
(C) $xy = 3 \mod 6$
(D) $xy = 6 \mod 7$
(E) $x = 5 \mod 42$
(F) $y = 39 \mod 42$

All true.
Xor

Computer Science:

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.
Xor

Computer Science:
- 1 - True
- 0 - False

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

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

Computer Science:
  1 - True
  0 - False

\[ 1 \lor 1 = 1 \]
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

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

\[ A \oplus B \] - Exclusive or.
\[ 1 \oplus 1 = 0 \]
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.
\[ 1 \oplus 1 = 0 \]
\[ 1 \oplus 0 = 1 \]
\[ 0 \oplus 1 = 1 \]
\[ 0 \oplus 0 = 0 \]
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!
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

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

\[A \oplus B\] - 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!
\[
\{0, 1\} \text{ is set. Take remainder for 2.}
\]
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.
Xor

Computer Science:
  1 - True
  0 - False

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

\[ A \oplus B \text{ - Exclusive or.} \]
\[ \begin{align*}
1 \oplus 1 &= 0 \\
1 \oplus 0 &= 1 \\
0 \oplus 1 &= 1 \\
0 \oplus 0 &= 0
\end{align*} \]

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
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 = ..................................

$E(m, s)$ – 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 ...

Cryptography ...

Alice \rightarrow Bob

Secret s

Eve

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

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

m = 10101011110101101

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

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

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 ...

\[ E(m, s) \]

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

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 ...

$E(m, s)$

Secret $s$

Message $m$

$E(m, s)$ – bitwise $m \oplus s$.

$D(x, s)$ – bitwise $x \oplus s$.

Works because $m \oplus s \oplus s = m$!

...given $E(m, s)$ any message $m$ is equally likely.

Disadvantages:

Shared secret!

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

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 $x \oplus s$.
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 \(|m|\).

\[ m = 10101011110101101 \]
\[ s = .................. \]

\[ E(m, s) \] – 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 a string of length \(|m|\).

\[
m = 10101011110101101
\]

\[
s = ..................................
\]

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

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

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 ...

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

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 \(|m|\).

\[ m = 10101011110101101 \]
\[ s = \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \]
Cryptography ...

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

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

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

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

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

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

Works because \( m \oplus s \oplus s = m! \)
...and totally secure!
Cryptography ...

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

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

Works because \( m \oplus s \oplus s = m \! \)
...and totally secure!
...given \( E(m, s) \) any message \( m \) is equally likely.
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:
**Cryptography** ...

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

Example:
One-time Pad: secret \( s \) is string of length \( |m| \).
\[ m = 1010101110101101 \]
\[ s = \ldots \]

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

$m = 10101011110101101$

$s = \ldots$ (omitted for brevity)

$E(m,s)$ – 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.
Cryptography...

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

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

\[ m = 10101011110101101 \]
\[ s = \ldots \ldots \ldots \]

\[ E(m, s) \text{ – bitwise } m \oplus s. \]
\[ D(x, s) \text{ – bitwise } x \oplus s. \]

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.
Public key cryptography.

Public key: $K$

Private key: $k$

Message: $m$

Encryption: $E(m, K)$

Decryption: $D(E(m, K), k)$

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?
Public key cryptography.

Public: $K$

Private: $k$

Message: $m$

$$E(m, K)$$

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

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?
Public key cryptography.

Private: $k$
Public: $K$

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?
Public key cryptography.

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

Alice  \( \rightarrow \) Bob  \( \leftarrow \) Eve

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 $m$

$E(m, K)$

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 $m$

$E(m, K)$
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?
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 \( K \). (Only?) Alice can decode with \( k \). Is this even possible?
Public key crypography.

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

Everyone knows key $K$!
Bob (and Eve...)

Only Alice knows the secret key $k$ for public key $K$. Only Alice can decode with $k$.

Is this even possible?
Public key cryptography.

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

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

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.

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

Private: \( k \)

Public: \( K \)

Message \( m \)

\( E(m, K) \)

Alice \( \rightarrow \) Bob

Eve

Everyone knows key \( K \)!
Bob (and Eve and me and you)
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.
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 \( K \).
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 \( k \).
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?

\[ \text{Announce } N \text{ and } e: K = (N, e) \text{ is my public key!} \]

\[ \text{Encoding: } \text{mod}(x^e, N) \]

\[ \text{Decoding: } \text{mod}(y^d, N) \]

\[ D(E(m)) = m \mod N? \]

\[ \text{Yes!} \]

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

No. In a sense. One can try every message to “break” system.

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

No. In a sense. One can try every message to “break” system. Too slow. Does it have to be slow?

\[\begin{aligned}
\text{Let } p \text{ and } q \text{ be large primes. Let } N &= pq. \\
\text{Choose } e \text{ relatively prime to } (p-1)(q-1). \\
\text{Compute } d &= e^{-1} \mod (p-1)(q-1). \\
\text{Announce } N (= p \cdot q) \text{ and } e; K = (N, e) \text{ is my public key!} \\
\text{Encoding: } \mod (x^e, N). \\
\text{Decoding: } \mod (y^d, N). \\
\text{Does } D(E(m)) = med = m \mod N? \\
\end{aligned}\]

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

No. In a sense. One can try every message to “break” system. Too slow. Does it have to be slow? We don’t really know.

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

No. In a sense. One can try every message to “break” system. Too slow. Does it have to be slow? We don’t really know. ...but we do public-key cryptography constantly!!!

---

¹ Typically small, say $e = 3$. 
Is public key crypto possible?

No. In a sense. One can try every message to “break” system. Too slow. Does it have to be slow? We don’t really know. ...but we do public-key cryptography constantly!!!

RSA (Rivest, Shamir, and Adleman)

---

1 Typically small, say $e = 3$. 
Is public key crypto possible?

No. In a sense. One can try every message to “break” system. Too slow. Does it have to be slow? We don’t really know. ...but we do public-key cryptography constantly!!!

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?

No. In a sense. One can try every message to “break” system. Too slow. Does it have to be slow? We don’t really know. ...but we do public-key cryptography constantly!!!

RSA (Rivest, Shamir, and Adleman)
Pick two large primes $p$ and $q$. Let $N = pq$.
Choose $e$ relatively prime to $(p - 1)(q - 1)$.\(^1\)

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

No. In a sense. One can try every message to “break” system. Too slow. Does it have to be slow? We don’t really know.
...but we do public-key cryptography constantly!!!

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 $d = e^{-1} \mod (p-1)(q-1)$.

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

No. In a sense. One can try every message to “break” system. Too slow. Does it have to be slow? We don’t really know. ...but we do public-key cryptography constantly!!!

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!

\footnote{Typically small, say $e = 3$.}
Is public key crypto possible?

No. In a sense. One can try every message to “break” system. Too slow. Does it have to be slow? We don’t really know. ...but we do public-key cryptography constantly!!

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

---

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

No. In a sense. One can try every message to “break” system. Too slow. Does it have to be slow? We don’t really know. ...but we do public-key cryptography constantly!!!

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

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

No. In a sense. One can try every message to “break” system. Too slow. Does it have to be slow? We don’t really know. ...but we do public-key cryptography constantly!!!

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

\footnote{Typically small, say $e = 3$.}
Is public key crypto possible?

No. In a sense. One can try every message to “break” system. Too slow. Does it have to be slow? We don’t really know. ...but we do public-key cryptography constantly!!!

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

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

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

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

Yes!

¹Typically small, say $e = 3$. 
What is a piece of RSA?

Bob has a key (N,e,d). Alice is good, Eve is evil.
What is a piece of RSA?

Bob has a key (N, e, d). Alice is good, Eve is evil.

(A) Eve knows e and N.
(B) Alice knows e and N.
(C) $ed = 1 \pmod{N−1}$
(D) Bob forgot $p$ and $q$ but can still decode?
(E) Bob knows $d$
(F) $ed = 1 \pmod{(p−1)(q−1)}$ if $N = pq$. 
What is a piece of RSA?

Bob has a key (N,e,d). Alice is good, Eve is evil.

(A) Eve knows e and N.
(B) Alice knows e and N.
(C) \( ed = 1 \pmod{N - 1} \)
(D) Bob forgot p and q but can still decode?
(E) Bob knows d
(F) \( ed = 1 \pmod{(p - 1)(q - 1)} \) if \( N = pq \).

(A), (B), (D), (E), (F)
Iterative Extended GCD.

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

\[ N = 77. \]
\[ (p - 1)(q - 1) = 60. \]

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

\[ \text{egcd}(7, 60) = 7(0) + 60(1) = 60. \]
\[ 7(1) + 60(0) = 7. \]
\[ 7(-8) + 60(1) = 4. \]
\[ 7(9) + 60(-1) = 3. \]
\[ 7(-17) + 60(2) = 1. \]

Confirm:
\[ -119 + 120 = 1. \]

\[ d = e - 1 = -17 = 43 \pmod{60}. \]
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 $\text{gcd}(7, 60) = 1$.

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

\[
\begin{align*}
7(0) + 60(1) &= 60 \\
7(1) + 60(0) &= 7
\end{align*}
\]
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).

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

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

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

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

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

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

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

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

Confirm:
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: \( -119 + 120 = 1 \)
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: $-119 + 120 = 1$

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

Public Key: (77, 7)

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

Message: 2!

E(2) = 2^e = 2^7 ≡ 128 = 51 (mod 77)

D(51) = 51^43 (mod 77)

uh oh! Obvious way: 43 multiplications.

Ouch.

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

Public Key: (77, 7)

Message: 2!

$E(2) = 2^e = 2^7 \equiv 128 \equiv 51 \pmod{77}$

$D(51) = 51^4 \equiv 43 \pmod{77}$

uh oh!

Obvious way: 43 multiplications.

Ouch.

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

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

Message: 2!
Encryption/Decryption Techniques.

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

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

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

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

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

Message: 2!

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

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

\[
E(2) = 2^e = 2^7 \equiv 128 = 51 \pmod{77}
\]
\[
D(51) = 51^{43} \pmod{77}
\]
Public Key: (77, 7)
Message Choices: \{0, \ldots, 76\}.

Message: 2!

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

uh oh!
Encryption/Decryption Techniques.

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

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

uh oh!

Obvious way: 43 multiplications.
Public Key: (77, 7)
Message Choices: {0, ..., 76}.

Message: 2!

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

uh oh!

Obvious way: 43 multiplications. Ouch.
Encryption/Decryption Techniques.

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

Message: 2!

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

uh oh!

Obvious way: 43 multiplications. Ouch.

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

Notice: 43 = 32 + 8 + 2 + 1 or 101011 in binary.

51
43
= (51
32
· 51
8
· 51
2
· 51
1
)(mod 77).

3 multiplications sort of...

Need to compute 51
32
· 51
1
.

51
1
≡ 51
(mod 77)

51
2
= (51
1
)∗ (51
1
) = 2601 ≡ 60 (mod 77)

51
4
= (51
2
)∗ (51
2
) = 60∗ 60 = 3600 ≡ 58 (mod 77)

51
8
= (51
4
)∗ (51
4
) = 58∗ 58 = 3364 ≡ 53 (mod 77)

51
16
= (51
8
)∗ (51
8
) = 53∗ 53 = 2809 ≡ 37 (mod 77)

51
32
= (51
16
)∗ (51
16
) = 37∗ 37 = 1369 ≡ 60 (mod 77)

5 more multiplications.

51
32
· 51
8
· 51
2
· 51
1
≡ 2 (mod 77).

Decoding got the message back!

Repeated Squaring took 8 multiplications versus 42.
Repeated squaring.

Notice: $43 = 32 + 8 + 2 + 1$ or $101011$ in binary.
Repeated squaring.

Notice: $43 = 32 + 8 + 2 + 1$ or $101011$ in binary.

$51^{43}$
Repeated squaring.

Notice: 43 = 32 + 8 + 2 + 1 or 101011 in binary.

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

Notice: $43 = 32 + 8 + 2 + 1$ or $101011$ in binary.

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

Notice: $43 = 32 + 8 + 2 + 1$ or $101011$ in binary.

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

3 multiplications sort of...
Repeated squaring.

Notice: $43 = 32 + 8 + 2 + 1$ or $101011$ in binary.

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

3 multiplications sort of...

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

Notice: $43 = 32 + 8 + 2 + 1$ or $101011$ in binary.

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

3 multiplications sort of...

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

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

Notice: \( 43 = 32 + 8 + 2 + 1 \) or \( 101011 \) in binary.
\[
51^{43} = 51^{32+8+2+1} = 51^{32} \cdot 51^8 \cdot 51^2 \cdot 51^1 \pmod{77}.
\]
3 multiplications sort of...
Need to compute \( 51^{32} \ldots 51^1 \)?
\[
51^1 \equiv 51 \pmod{77}
\]
\[
51^2 =
\]
Repeated squaring.

Notice: $43 = 32 + 8 + 2 + 1$ or $101011$ in binary.

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

3 multiplications sort of...

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

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

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

Notice: $43 = 32 + 8 + 2 + 1$ or $101011$ in binary.

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

3 multiplications sort of...

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

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

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

$51^4 =$
Repeated squaring.

Notice: $43 = 32 + 8 + 2 + 1$ or $101011$ in binary.

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

3 multiplications sort of...

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

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

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

$51^{4} = (51^{2}) \cdot (51^{2})$
Repeated squaring.

Notice: $43 = 32 + 8 + 2 + 1$ or $101011$ in binary.

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

3 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: $43 = 32 + 8 + 2 + 1$ or $101011$ in binary.

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

3 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: $43 = 32 + 8 + 2 + 1$ or $101011$ in binary.

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

3 multiplications sort of...

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

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

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

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

$51^8 = (51^4) \times (51^4)$

Decoding got the message back!

Repeating squaring took 8 multiplications versus 42.
Repeated squaring.

Notice: $43 = 32 + 8 + 2 + 1$ or $101011$ in binary.
$51^{43} = 51^{32+8+2+1} = 51^{32} \cdot 51^{8} \cdot 51^{2} \cdot 51^{1} \pmod{77}$.

3 multiplications sort of...

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

$51^{1} \equiv 51 \pmod{77}$
$51^{2} = (51) \cdot (51) = 2601 \equiv 60 \pmod{77}$
$51^{4} = (51^{2}) \cdot (51^{2}) = 60 \cdot 60 = 3600 \equiv 58 \pmod{77}$
$51^{8} = (51^{4}) \cdot (51^{4}) = 58 \cdot 58 = 3364 \equiv 53 \pmod{77}$

Decoding got the message back!

Repeated squaring took 8 multiplications versus 42.
Repeated squaring.

Notice: $43 = 32 + 8 + 2 + 1$ or $101011$ in binary.

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

3 multiplications sort of...

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

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

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

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

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

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

Decoding got the message back!

Repeated squaring took 8 multiplications versus 42.
Repeated squaring.

Notice: $43 = 32 + 8 + 2 + 1$ or $101011$ in binary.

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

3 multiplications sort of...

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

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

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

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

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

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

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

Notice: $43 = 32 + 8 + 2 + 1$ or $101011$ in binary.

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

3 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.
Repeated squaring.

Notice: \(43 = 32 + 8 + 2 + 1\) or \(101011\) in binary.
\[51^{43} = 51^{32+8+2+1} = 51^{32} \cdot 51^8 \cdot 51^2 \cdot 51^1 \pmod{77}.\]
3 multiplications sort of...
Need to compute \(51^{32} \ldots 51^1\)?
\[51^1 \equiv 51 \pmod{77}\]
\[51^2 = (51) \ast (51) = 2601 \equiv 60 \pmod{77}\]
\[51^4 = (51^2) \ast (51^2) = 60 \ast 60 = 3600 \equiv 58 \pmod{77}\]
\[51^8 = (51^4) \ast (51^4) = 58 \ast 58 = 3364 \equiv 53 \pmod{77}\]
\[51^{16} = (51^8) \ast (51^8) = 53 \ast 53 = 2809 \equiv 37 \pmod{77}\]
\[51^{32} = (51^{16}) \ast (51^{16}) = 37 \ast 37 = 1369 \equiv 60 \pmod{77}\]

5 more multiplications.
\[51^{32} \cdot 51^8 \cdot 51^2 \cdot 51^1 = (60) \ast (53) \ast (60) \ast (51) \equiv 2 \pmod{77}.\]
Repeated squaring.

Notice: \( 43 = 32 + 8 + 2 + 1 \) or \( 101011 \) in binary.

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

3 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: \(43 = 32 + 8 + 2 + 1\) or \(101011\) in binary.
\[51^{43} = 51^{32+8+2+1} = 51^{32} \cdot 51^8 \cdot 51^2 \cdot 51^1 \pmod{77}.
\]
3 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 took 8 multiplications.
Repeated squaring.

Notice: \(43 = 32 + 8 + 2 + 1\) or \(101011\) in binary.
\[51^{43} = 51^{32+8+2+1} = 51^{32} \cdot 51^8 \cdot 51^2 \cdot 51^1 \pmod{77}.\]
3 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 took 8 multiplications versus 42.
Repeated Squaring: $x^y$

1. 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.

$x^{43} = x^{32} \times x^8 \times x^2 \times x^1$.

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

All $n$-bit numbers. Repeated Squaring: $O(n)$ multiplications.

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

$\Rightarrow O(n^3)$ 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 $x^1$, 

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

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

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

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

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

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 $x^1, x^2, x^4, \ldots$, 

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

$$x^{43} = x^{32} \times x^8 \times x^2 \times x^1.$$
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 $x^1, x^2, x^4, \ldots, x^{2^{\lceil\log y\rceil}}$.

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

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

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

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:
\[
O(n) \text{ multiplications.}
\]
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.
   \[ x^{43} = x^{32} \cdot x^{8} \cdot x^{2} \cdot x^{1}. \]

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

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

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

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

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

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

\[ x^y. \]
Recursive.

Let $x^y$.

If $x$ is even, $x = 2k$, then $x^y = x^{2k} = (x^2)^k$.

If $x$ is odd, $x = 2k + 1$, then $x^y = x^{2k} = (x^2)^k$. 

Base case: $x^0 = 1$. 
Recursive.

\[ x^y. \]

\( x \) is even, \( x = 2k \), \( x^y = x^{2k} = (x^2)^k. \)

\[ \text{power} \ (x,y) = \text{power} \ (x^2, \frac{y}{2}). \]

\( x \) is odd, \( x = 2k + 1 \), \( x^y = (x^{2k+1}) = x^{2k} \cdot x. \)
Recursive.

\[ x^y. \]

\textit{xiseven, } \( x = 2k, \) \( x^y = x^{2k} = (x^2)^k. \)

\textit{power (x,y) = power (x^2, y/2).}

\textit{xisodd, } \( x = 2k+1, \) \( x^y = x^{2k} = (x^2)^k. \)
Recursive.

\[ x^y. \]

**xiseven,** \( x = 2k, \ x^y = x^{2^k} = (x^2)^k. \)

\[ \text{power} \ (x,y) = \text{power} \ (x^2, y/2). \]

**xisodd,** \( x = 2k+1, \ x^y = x^{2^k} = (x^2)^k. \)

\[ \text{power} \ (x,y) = x \times \text{power} \ (x^2, y/2). \]

Base case:

\[ x^0 = 1. \]
Recursive.

\[ x^y. \]

\textit{xiseven}, \( x = 2k \), \( x^y = x^{2k} = (x^2)^k \).

\text{power} (x,y) = \text{power} (x^2, y/2).

\textit{xisodd}, \( x = 2k+! \), \( x^y = x^{2k} = (x^2)^k \).

\text{power} (x,y) = x \times \text{power} (x^2, y/2).

\text{Base case:} \ x^0 = 1.
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.

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

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

Want:
Decoding.

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

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

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?

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

\( N = pq \) 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?

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

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

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?

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

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

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

\[
\implies a^{k(p-1)} \equiv 1 \pmod{p}
\]
Always decode correctly?

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

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

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 \( a \not\equiv 0 \pmod{p} \),

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

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

\[
\implies a^{k(p-1)} \equiv 1 \pmod{p} \implies a^{k(p-1)+1}
\]
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?

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

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

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

Proof:
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 $p$. 

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 $p$.

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

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

Mark what is true.

(A) \( 2 \neq 1 \mod 7 \)

(B) \( 2 = 1 \mod 7 \)

(C) \( 2, 2^2, 2^3, 2^4, 2^5, 2^6, 2^7 \) are distinct \( \mod 7 \).

(D) \( 2, 2^2, 2^3, 2^4, 2^5, 2^6 \) are distinct \( \mod 7 \)

(E) \( 2^{15} = 2 \mod 7 \)

(F) \( 2^{15} = 1 \mod 7 \)

(B), (F)
Poll
Mark what is true.

(A) $2^7 = 1 \mod 7$
(B) $2^6 = 1 \mod 7$
(C) $2^1, 2^2, 2^3, 2^4, 2^5, 2^6, 2^7$ are distinct mod 7.
(D) $2^1, 2^2, 2^3, 2^4, 2^5, 2^6$ are distinct mod 7.
(E) $2^{15} = 2 \mod 7$
(F) $2^{15} = 1 \mod 7$
Poll
Mark what is true.

(A) $2^7 = 1 \mod 7$
(B) $2^6 = 1 \mod 7$
(C) $2^1, 2^2, 2^3, 2^4, 2^5, 2^6, 2^7$ are distinct mod 7.
(D) $2^1, 2^2, 2^3, 2^4, 2^5, 2^6$ are distinct mod 7
(E) $2^{15} = 2 \mod 7$
(F) $2^{15} = 1 \mod 7$

(B), (F)
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.
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

$$a^{1+b(p-1)} \equiv a^{1}(a^{p-1})^{b}$$
Fermat’s Little Theorem: For prime \( p \), and \( a \neq 0 \pmod{p} \),

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

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 $a, b$, 
\[ a^{1+b(p-1)} \equiv a \pmod{p} \]
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 $x, k$,
$$x^{1+k(p-1)(q-1)} \equiv x \pmod{pq}$$

Proof:
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 $x, k$,

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

Proof:

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

Proof:
Let $a = x$, $b = k(p - 1)$ and apply Lemma 1 with modulus $q$.
\[ x^{1+k(p-1)(q-1)} \equiv x \pmod{q} \]
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 $x, k$, 
$$x^{1+k(p-1)(q-1)} \equiv x \pmod{pq}$$

Proof:

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 $p$. 

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

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

Proof:
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 $p$.
\[ x^{1+k(p-1)(q-1)} \equiv x \pmod{p} \]
\[ x^{1+k(q-1)(p-1)} - x \text{ is multiple of } p \text{ and } q. \]
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 $x, k$,
$$x^{1+k(p-1)(q-1)} \equiv x \pmod{pq}$$

Proof:
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 $p$.
$$x^{1+k(p-1)(q-1)} \equiv x \pmod{p} \quad x^{1+k(q-1)(p-1)} - x \text{ is multiple of } p \text{ and } q.$$ 
$$x^{1+k(q-1)(p-1)} - x \equiv 0 \pmod{(pq)}$$
...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 $x, k$,
\[ x^{1+k(p-1)(q-1)} \equiv x \pmod{pq} \]

**Proof:**

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

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

From CRT: $y = x \pmod{p}$ and $y = x \pmod{q} \implies y = x$. 

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

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

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

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

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

**Theorem:** RSA correctly decodes!
Recall
\[ D(E(x)) = (x^e)^d = x^{ed} \equiv x \pmod{pq}, \]
where $ed \equiv 1 \pmod{(p-1)(q-1)} \implies ed = 1 + k(p-1)(q-1)$

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

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 $N \geq 17$

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

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

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

   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 $N \geq 17$

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

   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 $\gcd(e, (p-1)(q-1)) = 1$. 
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.
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 $e$. 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.
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 $\implies$ factoring algorithm.
Much more to it.....

If Bobs sends a message (Credit Card Number) to Alice,
Much more to it.....

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;
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 $r$. 
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 $c$. 
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.
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.

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 $C$: $S_V(C) = D(C, k_V)$.

Browser receives: $[C, y]$ checks $E(y, K_V) = C$?

$E(S_V(C), K_V) = C \equiv (S_V(C))^e \equiv C^{de} \equiv C (mod N)$

Valid signature of Amazon certificate $C$!

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

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

Amazon

Browser.

Verisign:

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

Browser "knows" Verisign's public key: $k_V$.

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

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 $E(y, k_V) = C$?

$E(SV(C), K_V) = (SV(C))^e = (C_d)^e = C$ (mod $N$)

Valid signature of Amazon certificate $C$!

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

Verisign: $k_V, K_V$

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

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

Amazon $\rightarrow$ 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 $K_A$.”
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: $k_V, K_V$

$[C, S_V(C)]$

Amazon $\rightarrow$ 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.

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

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

Browser receives: $[C, y]$
Signatures using RSA.

Verisign: $k_V, K_V$

$[C, S_V(C)]$

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

$[C, S_V(C)]$

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

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

Verisign 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.

Verisign: $k_V, K_V$

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

$[C, S_V(C)]$

Amazon

Browser. $K_V$

$[C, S_V(C)]$

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

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

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

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

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

\( E(S_V(C), K_V) = (S_V(C))^e = (C^d)^e = C^{de} \)
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 $E(y, K_V) = C$?

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

$E(S_V(C), K_V) = (S_V(C))^e = (C^d)^e = C^{de} = C \mod N$
Valid signature of Amazon certificate $C$!
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: $C = \text{“I am Amazon. My public Key is } K_A.$”
Versign signature of $C$: $S_V(C)$: $D(C, k_V) = C^d \mod N$.
Browser receives: $[C, y]$ Checks $E(y, K_V) = C$?
$E(S_V(C), K_V) = (S_V(C))^e = (C^d)^e = C^{de} = C \pmod{N}$
Valid signature of Amazon certificate $C$!
Security: Eve can’t forge unless she “breaks” RSA scheme.
RSA

Public Key Cryptography:

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

Signature scheme:

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

RSA

Public Key Cryptography:
Public Key Cryptography:

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

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:

\[ E(D(C, k), K) = (C^d)^e \mod N = C \]
Signature authority has public key \((N,e)\).
Signature authority has public key \((N,e)\).

(A) Given message/signature \((x,y)\) : check \(y^d = x \pmod{N}\)

(B) Given message/signature \((x,y)\): check \(y^e = x \pmod{N}\)

(C) Signature of message \(x\) is \(x^e \pmod{N}\)

(D) Signature of message \(x\) is \(x^d \pmod{N}\)
Signature authority has public key \((N,e)\).

(A) Given message/signature \((x,y)\) : check \(y^d = x \pmod{N}\)
(B) Given message/signature \((x,y)\): check \(y^e = x \pmod{N}\)
(C) Signature of message \(x\) is \(x^e \pmod{N}\)
(D) Signature of message \(x\) is \(x^d \pmod{N}\)
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?
Other Eve.

Get CA to certify fake certificates: Microsoft Corporation.
Get CA to certify fake certificates: Microsoft Corporation.
2001..Doh.
Other Eve.

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:**

\[ N = pq \]

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

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

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

Repeated Squaring ⇒ efficiency.

Fermat's Theorem ⇒ correctness.

Good for Encryption and Signature Schemes.
Summary.

Public-Key Encryption.

RSA Scheme:

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

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

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

Repeated Squaring ⇒ efficiency.

Fermat's Theorem ⇒ correctness.

Good for Encryption and Signature Schemes.
Summary.

Public-Key Encryption.

RSA Scheme:

\[ N = pq \] 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.
Public-Key Encryption.

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

Repeated Squaring $\Rightarrow$ efficiency.
Public-Key Encryption.

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

Repeated Squaring \( \Rightarrow \) efficiency.

Fermat’s Theorem \( \Rightarrow \) correctness.
Summary.

Public-Key Encryption.

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

Repeated Squaring \(\Rightarrow\) efficiency.

Fermat’s Theorem \(\Rightarrow\) correctness.

Good for Encryption
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.

Good for Encryption and Signature Schemes.