Encrypting Communication

Puella Magi Madoka Magica Blu-ray 1

Blu-ray

38 customer reviews

Blu-ray from $44.99	DVD from $14.01

Additional Blu-ray options	Edition	Discs	Price	New from	Used from
Blu-ray (Jan 01, 2012)	—	—	—	$44.99	—
Blu-ray	—	1	—	$49.38	—
Blu-ray (May 03, 2011)	—	2	—	$52.94	$13.88

Report incorrect product information.

I want to buy my favorite show on Amazon.

I enter my credit card information online.

What if someone is trying to steal my credit card information?

Encrypting Communication

Puella Magi Madoka Magica Blu-ray 1

Blu-ray

38 customer reviews

Blu-ray from $44.99 | DVD from $14.01

Additional Blu-ray options	Edition	Discs	Price	New from	Used from
Blu-ray (Jan 01, 2012)	—	—	—	$44.99	—
Blu-ray	—	1	—	$49.38	—
Blu-ray (May 03, 2011)	—	2	—	$52.94	$13.88

Report incorrect product information.

I want to buy my favorite show on Amazon.

I enter my credit card information online.

What if someone is trying to steal my credit card information?

Today: Encrypt communication using RSA.

Review

$\mathbb{Z}/m\mathbb{Z} = \{0, 1, \ldots, m-1\}$ with operations of addition and multiplication modulo m.

Review

$\mathbb{Z}/m\mathbb{Z} = \{0, 1, \ldots, m-1\}$ with operations of addition and multiplication modulo m.

$(\mathbb{Z}/m\mathbb{Z})^\times$ is the set of elements in $\mathbb{Z}/m\mathbb{Z}$ which have multplicative inverses.

Review

$\mathbb{Z}/m\mathbb{Z} = \{0, 1, \ldots, m-1\}$ with operations of addition and multiplication modulo m.

$(\mathbb{Z}/m\mathbb{Z})^\times$ is the set of elements in $\mathbb{Z}/m\mathbb{Z}$ which have multplicative inverses.

- In other words, $a \in (\mathbb{Z}/m\mathbb{Z})^\times$ if and only if $\gcd(a, m) = 1$.

Review

$\mathbb{Z}/m\mathbb{Z} = \{0, 1, \ldots, m-1\}$ with operations of addition and multiplication modulo m.

$(\mathbb{Z}/m\mathbb{Z})^\times$ is the set of elements in $\mathbb{Z}/m\mathbb{Z}$ which have multplicative inverses.

- In other words, $a \in (\mathbb{Z}/m\mathbb{Z})^\times$ if and only if $\gcd(a, m) = 1$.

For $a \in (\mathbb{Z}/m\mathbb{Z})^\times$, we can compute a^{-1} efficiently.

Review

$\mathbb{Z}/m\mathbb{Z} = \{0, 1, \ldots, m-1\}$ with operations of addition and multiplication modulo m.

$(\mathbb{Z}/m\mathbb{Z})^\times$ is the set of elements in $\mathbb{Z}/m\mathbb{Z}$ which have multplicative inverses.

- In other words, $a \in (\mathbb{Z}/m\mathbb{Z})^\times$ if and only if $\gcd(a, m) = 1$.

For $a \in (\mathbb{Z}/m\mathbb{Z})^\times$, we can compute a^{-1} efficiently. (Extended Euclid's Algorithm)

Review

$\mathbb{Z}/m\mathbb{Z} = \{0, 1, \ldots, m-1\}$ with operations of addition and multiplication modulo m.

$(\mathbb{Z}/m\mathbb{Z})^\times$ is the set of elements in $\mathbb{Z}/m\mathbb{Z}$ which have multplicative inverses.

- In other words, $a \in (\mathbb{Z}/m\mathbb{Z})^\times$ if and only if $\gcd(a, m) = 1$.

For $a \in (\mathbb{Z}/m\mathbb{Z})^\times$, we can compute a^{-1} efficiently. (Extended Euclid's Algorithm)

If p is prime, then $(\mathbb{Z}/p\mathbb{Z})^\times = \{1, \ldots, p-1\}$.

Extended Euclid's Algorithm

Extended Euclid's Algorithm:

- If $b = 0$, then $\mathrm{egcd}(a, 0) = (a, 1, 0)$.
- Otherwise, let $(d', x', y') := \mathrm{egcd}(b, a \bmod b)$. Return $(d', y', x' - \lfloor a/b \rfloor y')$.

Extended Euclid's Algorithm

Extended Euclid's Algorithm:

- If $b = 0$, then $\mathrm{egcd}(a, 0) = (a, 1, 0)$.
- Otherwise, let $(d', x', y') := \mathrm{egcd}(b, a \bmod b)$. Return $(d', y', x' - \lfloor a/b \rfloor y')$.

Extended Euclid is just as fast as Euclid's Algorithm.

Extended Euclid's Algorithm

Extended Euclid's Algorithm:

- If $b = 0$, then $\mathrm{egcd}(a, 0) = (a, 1, 0)$.
- Otherwise, let $(d', x', y') := \mathrm{egcd}(b, a \bmod b)$. Return $(d', y', x' - \lfloor a/b \rfloor y')$.

Extended Euclid is just as fast as Euclid's Algorithm.

We have proved: we can express $\mathrm{gcd}(a, b)$ as an integer linear combination of a and b.

Extended Euclid's Algorithm

Extended Euclid's Algorithm:

- If $b = 0$, then $\mathrm{egcd}(a, 0) = (a, 1, 0)$.
- Otherwise, let $(d', x', y') := \mathrm{egcd}(b, a \bmod b)$. Return $(d', y', x' - \lfloor a/b \rfloor y')$.

Extended Euclid is just as fast as Euclid's Algorithm.

We have proved: we can express $\gcd(a, b)$ as an integer linear combination of a and b.

If $d = x \cdot a + y \cdot b$, then multiply both sides by k.

Extended Euclid's Algorithm

Extended Euclid's Algorithm:

- If $b = 0$, then $\mathrm{egcd}(a, 0) = (a, 1, 0)$.
- Otherwise, let $(d', x', y') := \mathrm{egcd}(b, a \bmod b)$. Return $(d', y', x' - \lfloor a/b \rfloor y')$.

Extended Euclid is just as fast as Euclid's Algorithm.

We have proved: we can express $\gcd(a, b)$ as an integer linear combination of a and b.

If $d = x \cdot a + y \cdot b$, then multiply both sides by k.
$kd = kx \cdot a + ky \cdot b$.

Extended Euclid's Algorithm

Extended Euclid's Algorithm:

- If $b = 0$, then $\text{egcd}(a, 0) = (a, 1, 0)$.
- Otherwise, let $(d', x', y') := \text{egcd}(b, a \bmod b)$. Return $(d', y', x' - \lfloor a/b \rfloor y')$.

Extended Euclid is just as fast as Euclid's Algorithm.

We have proved: we can express $\gcd(a, b)$ as an integer linear combination of a and b.

If $d = x \cdot a + y \cdot b$, then multiply both sides by k.
$kd = kx \cdot a + ky \cdot b$.

A number can be expressed as an integer linear combination of a and b *if and only if* it is a multiple of $\gcd(a, b)$.

Back to Multiplicative Inverses

Let $a \in \mathbb{Z}/m\mathbb{Z}$.

Back to Multiplicative Inverses

Let $a \in \mathbb{Z}/m\mathbb{Z}$.

- Run Extended Euclid on a, m, which gives $\gcd(a,m) = x \cdot a + y \cdot m$.

Back to Multiplicative Inverses

Let $a \in \mathbb{Z}/m\mathbb{Z}$.

- Run Extended Euclid on a, m, which gives $\gcd(a, m) = x \cdot a + y \cdot m$.
- If $\gcd(a, m) > 1$, then a^{-1} does not exist.

Back to Multiplicative Inverses

Let $a \in \mathbb{Z}/m\mathbb{Z}$.

- Run Extended Euclid on a, m, which gives $\gcd(a, m) = x \cdot a + y \cdot m$.
- If $\gcd(a, m) > 1$, then a^{-1} does not exist.
- Otherwise, we have $1 = x \cdot a + y \cdot m$.

Back to Multiplicative Inverses

Let $a \in \mathbb{Z}/m\mathbb{Z}$.

- Run Extended Euclid on a, m, which gives $\gcd(a, m) = x \cdot a + y \cdot m$.
- If $\gcd(a, m) > 1$, then a^{-1} does not exist.
- Otherwise, we have $1 = x \cdot a + y \cdot m$.
- Take both sides modulo m: $1 \equiv x \cdot a \pmod{m}$.

Back to Multiplicative Inverses

Let $a \in \mathbb{Z}/m\mathbb{Z}$.

- Run Extended Euclid on a, m, which gives $\gcd(a,m) = x \cdot a + y \cdot m$.
- If $\gcd(a,m) > 1$, then a^{-1} does not exist.
- Otherwise, we have $1 = x \cdot a + y \cdot m$.
- Take both sides modulo m: $1 \equiv x \cdot a \pmod{m}$.
- Thus, $a^{-1} \equiv x \pmod{m}$.

Back to Multiplicative Inverses

Let $a \in \mathbb{Z}/m\mathbb{Z}$.

- Run Extended Euclid on a, m, which gives $\gcd(a,m) = x \cdot a + y \cdot m$.
- If $\gcd(a,m) > 1$, then a^{-1} does not exist.
- Otherwise, we have $1 = x \cdot a + y \cdot m$.
- Take both sides modulo m: $1 \equiv x \cdot a \pmod{m}$.
- Thus, $a^{-1} \equiv x \pmod{m}$.

We can now efficiently compute multiplicative inverses!

Euler's Totient Function

We define $\varphi(1) := 1$, and for positive integers m, $\varphi(m) := |(\mathbb{Z}/m\mathbb{Z})^\times|$.

Euler's Totient Function

We define $\varphi(1) := 1$, and for positive integers m, $\varphi(m) := |(\mathbb{Z}/m\mathbb{Z})^\times|$.

In other words, $\varphi(m)$ is the number of elements with multiplicative inverses in $\mathbb{Z}/m\mathbb{Z}$.

Euler's Totient Function

We define $\varphi(1) := 1$, and for positive integers m, $\varphi(m) := |(\mathbb{Z}/m\mathbb{Z})^\times|$.

In other words, $\varphi(m)$ is the number of elements with multiplicative inverses in $\mathbb{Z}/m\mathbb{Z}$.

In other words, $\varphi(m)$ is the number of integers in $\{0, 1, \ldots, m-1\}$ which are relatively prime to m.

Euler's Totient Function

We define $\varphi(1) := 1$, and for positive integers m, $\varphi(m) := |(\mathbb{Z}/m\mathbb{Z})^\times|$.

In other words, $\varphi(m)$ is the number of elements with multiplicative inverses in $\mathbb{Z}/m\mathbb{Z}$.

In other words, $\varphi(m)$ is the number of integers in $\{0, 1, \ldots, m-1\}$ which are relatively prime to m.

Examples:

- $\varphi(2) = 1$.

Euler's Totient Function

We define $\varphi(1) := 1$, and for positive integers m, $\varphi(m) := |(\mathbb{Z}/m\mathbb{Z})^\times|$.

In other words, $\varphi(m)$ is the number of elements with multiplicative inverses in $\mathbb{Z}/m\mathbb{Z}$.

In other words, $\varphi(m)$ is the number of integers in $\{0, 1, \ldots, m-1\}$ which are relatively prime to m.

Examples:

- $\varphi(2) = 1$. $(\mathbb{Z}/2\mathbb{Z})^\times = \{1\}$.

Euler's Totient Function

We define $\varphi(1) := 1$, and for positive integers m, $\varphi(m) := |(\mathbb{Z}/m\mathbb{Z})^\times|$.

In other words, $\varphi(m)$ is the number of elements with multiplicative inverses in $\mathbb{Z}/m\mathbb{Z}$.

In other words, $\varphi(m)$ is the number of integers in $\{0, 1, \ldots, m-1\}$ which are relatively prime to m.

Examples:

- $\varphi(2) = 1$. $(\mathbb{Z}/2\mathbb{Z})^\times = \{1\}$.
- $\varphi(3) = 2$.

Euler's Totient Function

We define $\varphi(1) := 1$, and for positive integers m, $\varphi(m) := |(\mathbb{Z}/m\mathbb{Z})^{\times}|$.

In other words, $\varphi(m)$ is the number of elements with multiplicative inverses in $\mathbb{Z}/m\mathbb{Z}$.

In other words, $\varphi(m)$ is the number of integers in $\{0, 1, \ldots, m-1\}$ which are relatively prime to m.

Examples:

- $\varphi(2) = 1$. $(\mathbb{Z}/2\mathbb{Z})^{\times} = \{1\}$.
- $\varphi(3) = 2$. $(\mathbb{Z}/3\mathbb{Z})^{\times} = \{1, 2\}$.

Euler's Totient Function

We define $\varphi(1) := 1$, and for positive integers m, $\varphi(m) := |(\mathbb{Z}/m\mathbb{Z})^\times|$.

In other words, $\varphi(m)$ is the number of elements with multiplicative inverses in $\mathbb{Z}/m\mathbb{Z}$.

In other words, $\varphi(m)$ is the number of integers in $\{0, 1, \ldots, m-1\}$ which are relatively prime to m.

Examples:

- $\varphi(2) = 1$. $(\mathbb{Z}/2\mathbb{Z})^\times = \{1\}$.
- $\varphi(3) = 2$. $(\mathbb{Z}/3\mathbb{Z})^\times = \{1, 2\}$.
- $\varphi(4) = 2$.

Euler's Totient Function

We define $\varphi(1) := 1$, and for positive integers m, $\varphi(m) := |(\mathbb{Z}/m\mathbb{Z})^\times|$.

In other words, $\varphi(m)$ is the number of elements with multiplicative inverses in $\mathbb{Z}/m\mathbb{Z}$.

In other words, $\varphi(m)$ is the number of integers in $\{0, 1, \ldots, m-1\}$ which are relatively prime to m.

Examples:

- $\varphi(2) = 1$. $(\mathbb{Z}/2\mathbb{Z})^\times = \{1\}$.
- $\varphi(3) = 2$. $(\mathbb{Z}/3\mathbb{Z})^\times = \{1, 2\}$.
- $\varphi(4) = 2$. $(\mathbb{Z}/4\mathbb{Z})^\times = \{1, 3\}$.

Euler's Totient Function

We define $\varphi(1) := 1$, and for positive integers m, $\varphi(m) := |(\mathbb{Z}/m\mathbb{Z})^\times|$.

In other words, $\varphi(m)$ is the number of elements with multiplicative inverses in $\mathbb{Z}/m\mathbb{Z}$.

In other words, $\varphi(m)$ is the number of integers in $\{0, 1, \ldots, m-1\}$ which are relatively prime to m.

Examples:

- $\varphi(2) = 1$. $(\mathbb{Z}/2\mathbb{Z})^\times = \{1\}$.
- $\varphi(3) = 2$. $(\mathbb{Z}/3\mathbb{Z})^\times = \{1, 2\}$.
- $\varphi(4) = 2$. $(\mathbb{Z}/4\mathbb{Z})^\times = \{1, 3\}$.
- $\varphi(5) = 4$.

Euler's Totient Function

We define $\varphi(1) := 1$, and for positive integers m, $\varphi(m) := |(\mathbb{Z}/m\mathbb{Z})^\times|$.

In other words, $\varphi(m)$ is the number of elements with multiplicative inverses in $\mathbb{Z}/m\mathbb{Z}$.

In other words, $\varphi(m)$ is the number of integers in $\{0,1,\ldots,m-1\}$ which are relatively prime to m.

Examples:

- $\varphi(2) = 1$. $(\mathbb{Z}/2\mathbb{Z})^\times = \{1\}$.
- $\varphi(3) = 2$. $(\mathbb{Z}/3\mathbb{Z})^\times = \{1,2\}$.
- $\varphi(4) = 2$. $(\mathbb{Z}/4\mathbb{Z})^\times = \{1,3\}$.
- $\varphi(5) = 4$. $(\mathbb{Z}/5\mathbb{Z})^\times = \{1,2,3,4\}$.

Euler's Totient Function

We define $\varphi(1) := 1$, and for positive integers m, $\varphi(m) := |(\mathbb{Z}/m\mathbb{Z})^\times|$.

In other words, $\varphi(m)$ is the number of elements with multiplicative inverses in $\mathbb{Z}/m\mathbb{Z}$.

In other words, $\varphi(m)$ is the number of integers in $\{0, 1, \ldots, m-1\}$ which are relatively prime to m.

Examples:

- $\varphi(2) = 1$. $(\mathbb{Z}/2\mathbb{Z})^\times = \{1\}$.
- $\varphi(3) = 2$. $(\mathbb{Z}/3\mathbb{Z})^\times = \{1, 2\}$.
- $\varphi(4) = 2$. $(\mathbb{Z}/4\mathbb{Z})^\times = \{1, 3\}$.
- $\varphi(5) = 4$. $(\mathbb{Z}/5\mathbb{Z})^\times = \{1, 2, 3, 4\}$.
- $\varphi(6) = 2$.

Euler's Totient Function

We define $\varphi(1) := 1$, and for positive integers m, $\varphi(m) := |(\mathbb{Z}/m\mathbb{Z})^\times|$.

In other words, $\varphi(m)$ is the number of elements with multiplicative inverses in $\mathbb{Z}/m\mathbb{Z}$.

In other words, $\varphi(m)$ is the number of integers in $\{0, 1, \ldots, m-1\}$ which are relatively prime to m.

Examples:

- $\varphi(2) = 1$. $(\mathbb{Z}/2\mathbb{Z})^\times = \{1\}$.
- $\varphi(3) = 2$. $(\mathbb{Z}/3\mathbb{Z})^\times = \{1, 2\}$.
- $\varphi(4) = 2$. $(\mathbb{Z}/4\mathbb{Z})^\times = \{1, 3\}$.
- $\varphi(5) = 4$. $(\mathbb{Z}/5\mathbb{Z})^\times = \{1, 2, 3, 4\}$.
- $\varphi(6) = 2$. $(\mathbb{Z}/6\mathbb{Z})^\times = \{1, 5\}$.

Euler's Totient Function

We define $\varphi(1) := 1$, and for positive integers m, $\varphi(m) := |(\mathbb{Z}/m\mathbb{Z})^\times|$.

In other words, $\varphi(m)$ is the number of elements with multiplicative inverses in $\mathbb{Z}/m\mathbb{Z}$.

In other words, $\varphi(m)$ is the number of integers in $\{0, 1, \ldots, m-1\}$ which are relatively prime to m.

Examples:

- $\varphi(2) = 1$. $(\mathbb{Z}/2\mathbb{Z})^\times = \{1\}$.
- $\varphi(3) = 2$. $(\mathbb{Z}/3\mathbb{Z})^\times = \{1, 2\}$.
- $\varphi(4) = 2$. $(\mathbb{Z}/4\mathbb{Z})^\times = \{1, 3\}$.
- $\varphi(5) = 4$. $(\mathbb{Z}/5\mathbb{Z})^\times = \{1, 2, 3, 4\}$.
- $\varphi(6) = 2$. $(\mathbb{Z}/6\mathbb{Z})^\times = \{1, 5\}$.
- $\varphi(p)$ for p prime?

Euler's Totient Function

We define $\varphi(1) := 1$, and for positive integers m, $\varphi(m) := |(\mathbb{Z}/m\mathbb{Z})^{\times}|$.

In other words, $\varphi(m)$ is the number of elements with multiplicative inverses in $\mathbb{Z}/m\mathbb{Z}$.

In other words, $\varphi(m)$ is the number of integers in $\{0, 1, \ldots, m-1\}$ which are relatively prime to m.

Examples:

- $\varphi(2) = 1$. $(\mathbb{Z}/2\mathbb{Z})^{\times} = \{1\}$.
- $\varphi(3) = 2$. $(\mathbb{Z}/3\mathbb{Z})^{\times} = \{1, 2\}$.
- $\varphi(4) = 2$. $(\mathbb{Z}/4\mathbb{Z})^{\times} = \{1, 3\}$.
- $\varphi(5) = 4$. $(\mathbb{Z}/5\mathbb{Z})^{\times} = \{1, 2, 3, 4\}$.
- $\varphi(6) = 2$. $(\mathbb{Z}/6\mathbb{Z})^{\times} = \{1, 5\}$.
- $\varphi(p)$ for p prime? $\varphi(p) = p - 1$.

Bijections

Recall: Let $f(x) = ax \bmod m$.

Bijections

Recall: Let $f(x) = ax \bmod m$. The map f is a bijection if and only if $\gcd(a, m) = 1$.

Bijections

Recall: Let $f(x) = ax \bmod m$. The map f is a bijection if and only if $\gcd(a,m) = 1$.

So if $\gcd(a,m) = 1$, $\{0,1,2,\ldots,m-1\} = \{0,a,2a,\ldots,(m-1)a\}$.

Bijections

Recall: Let $f(x) = ax \bmod m$. The map f is a bijection if and only if $\gcd(a, m) = 1$.

So if $\gcd(a, m) = 1$, $\{0, 1, 2, \ldots, m-1\} = \{0, a, 2a, \ldots, (m-1)a\}$.

But what if you only apply f to elements in $(\mathbb{Z}/m\mathbb{Z})^\times$?

Bijections

Recall: Let $f(x) = ax \bmod m$. The map f is a bijection if and only if $\gcd(a, m) = 1$.

So if $\gcd(a, m) = 1$, $\{0, 1, 2, \ldots, m-1\} = \{0, a, 2a, \ldots, (m-1)a\}$.

But what if you only apply f to elements in $(\mathbb{Z}/m\mathbb{Z})^\times$?

Since a is coprime with m, and elements in $(\mathbb{Z}/m\mathbb{Z})^\times$ are coprime with m, the result is still coprime with m.

Bijections

Recall: Let $f(x) = ax \bmod m$. The map f is a bijection if and only if $\gcd(a, m) = 1$.

So if $\gcd(a, m) = 1$, $\{0, 1, 2, \ldots, m-1\} = \{0, a, 2a, \ldots, (m-1)a\}$.

But what if you only apply f to elements in $(\mathbb{Z}/m\mathbb{Z})^{\times}$?

Since a is coprime with m, and elements in $(\mathbb{Z}/m\mathbb{Z})^{\times}$ are coprime with m, the result is still coprime with m.

But we know f is one-to-one.

Bijections

Recall: Let $f(x) = ax \mod m$. The map f is a bijection if and only if $\gcd(a, m) = 1$.

So if $\gcd(a, m) = 1$, $\{0, 1, 2, \ldots, m-1\} = \{0, a, 2a, \ldots, (m-1)a\}$.

But what if you only apply f to elements in $(\mathbb{Z}/m\mathbb{Z})^\times$?

Since a is coprime with m, and elements in $(\mathbb{Z}/m\mathbb{Z})^\times$ are coprime with m, the result is still coprime with m.

But we know f is one-to-one.

Thus, f is also a bijection $(\mathbb{Z}/m\mathbb{Z})^\times \to (\mathbb{Z}/m\mathbb{Z})^\times$.

Euler's Theorem

If $\gcd(a, m) = 1$, then $f(x) = ax \bmod m$ is a bijection $(\mathbb{Z}/m\mathbb{Z})^\times \to (\mathbb{Z}/m\mathbb{Z})^\times$.

Euler's Theorem

If $\gcd(a, m) = 1$, then $f(x) = ax \bmod m$ is a bijection $(\mathbb{Z}/m\mathbb{Z})^\times \to (\mathbb{Z}/m\mathbb{Z})^\times$.

Example: $m = 5$, $a = 3$.

Euler's Theorem

If $\gcd(a,m)=1$, then $f(x)=ax \bmod m$ is a bijection $(\mathbb{Z}/m\mathbb{Z})^\times \to (\mathbb{Z}/m\mathbb{Z})^\times$.

Example: $m=5$, $a=3$.

- $(\mathbb{Z}/5\mathbb{Z})^\times = \{1,2,3,4\} = \{3,6,9,12\}$.

Euler's Theorem

If $\gcd(a,m)=1$, then $f(x)=ax \bmod m$ is a bijection $(\mathbb{Z}/m\mathbb{Z})^\times \to (\mathbb{Z}/m\mathbb{Z})^\times$.

Example: $m=5$, $a=3$.

- $(\mathbb{Z}/5\mathbb{Z})^\times = \{1,2,3,4\} = \{3,6,9,12\}$.

In general, $(\mathbb{Z}/m\mathbb{Z})^\times = \{ax : x \in (\mathbb{Z}/m\mathbb{Z})^\times\}$.

Euler's Theorem

If $\gcd(a, m) = 1$, then $f(x) = ax \bmod m$ is a bijection $(\mathbb{Z}/m\mathbb{Z})^\times \to (\mathbb{Z}/m\mathbb{Z})^\times$.

Example: $m = 5$, $a = 3$.

- $(\mathbb{Z}/5\mathbb{Z})^\times = \{1, 2, 3, 4\} = \{3, 6, 9, 12\}$.

In general, $(\mathbb{Z}/m\mathbb{Z})^\times = \{ax : x \in (\mathbb{Z}/m\mathbb{Z})^\times\}$.

Idea: Multiply all elements in both sides.

Euler's Theorem

If $\gcd(a,m)=1$, then $f(x)=ax \bmod m$ is a bijection $(\mathbb{Z}/m\mathbb{Z})^\times \to (\mathbb{Z}/m\mathbb{Z})^\times$.

Example: $m=5$, $a=3$.

- $(\mathbb{Z}/5\mathbb{Z})^\times = \{1,2,3,4\} = \{3,6,9,12\}$.

In general, $(\mathbb{Z}/m\mathbb{Z})^\times = \{ax : x \in (\mathbb{Z}/m\mathbb{Z})^\times\}$.

Idea: Multiply all elements in both sides.

$$\prod_{x\in(\mathbb{Z}/m\mathbb{Z})^\times} x \equiv \prod_{x\in(\mathbb{Z}/m\mathbb{Z})^\times} ax \pmod{m}.$$

Euler's Theorem

If $\gcd(a, m) = 1$, then $f(x) = ax \bmod m$ is a bijection $(\mathbb{Z}/m\mathbb{Z})^\times \to (\mathbb{Z}/m\mathbb{Z})^\times$.

Example: $m = 5$, $a = 3$.

- $(\mathbb{Z}/5\mathbb{Z})^\times = \{1, 2, 3, 4\} = \{3, 6, 9, 12\}$.

In general, $(\mathbb{Z}/m\mathbb{Z})^\times = \{ax : x \in (\mathbb{Z}/m\mathbb{Z})^\times\}$.

Idea: Multiply all elements in both sides.

$$\prod_{x \in (\mathbb{Z}/m\mathbb{Z})^\times} x \equiv \prod_{x \in (\mathbb{Z}/m\mathbb{Z})^\times} ax \pmod{m}.$$

Each $x \in (\mathbb{Z}/m\mathbb{Z})^\times$ has an inverse, so divide!

Euler's Theorem

If $\gcd(a, m) = 1$, then $f(x) = ax \bmod m$ is a bijection $(\mathbb{Z}/m\mathbb{Z})^\times \to (\mathbb{Z}/m\mathbb{Z})^\times$.

Example: $m = 5$, $a = 3$.

- $(\mathbb{Z}/5\mathbb{Z})^\times = \{1, 2, 3, 4\} = \{3, 6, 9, 12\}$.

In general, $(\mathbb{Z}/m\mathbb{Z})^\times = \{ax : x \in (\mathbb{Z}/m\mathbb{Z})^\times\}$.

Idea: Multiply all elements in both sides.

$$\prod_{x \in (\mathbb{Z}/m\mathbb{Z})^\times} x \equiv \prod_{x \in (\mathbb{Z}/m\mathbb{Z})^\times} ax \pmod{m}.$$

Each $x \in (\mathbb{Z}/m\mathbb{Z})^\times$ has an inverse, so divide! $\prod_{x \in (\mathbb{Z}/m\mathbb{Z})^\times} a \equiv 1 \pmod{m}$.

Euler's Theorem

If $\gcd(a, m) = 1$, then $f(x) = ax \bmod m$ is a bijection $(\mathbb{Z}/m\mathbb{Z})^\times \to (\mathbb{Z}/m\mathbb{Z})^\times$.

Example: $m = 5$, $a = 3$.

- $(\mathbb{Z}/5\mathbb{Z})^\times = \{1, 2, 3, 4\} = \{3, 6, 9, 12\}$.

In general, $(\mathbb{Z}/m\mathbb{Z})^\times = \{ax : x \in (\mathbb{Z}/m\mathbb{Z})^\times\}$.

Idea: Multiply all elements in both sides.

$$\prod_{x \in (\mathbb{Z}/m\mathbb{Z})^\times} x \equiv \prod_{x \in (\mathbb{Z}/m\mathbb{Z})^\times} ax \pmod{m}.$$

Each $x \in (\mathbb{Z}/m\mathbb{Z})^\times$ has an inverse, so divide! $\prod_{x \in (\mathbb{Z}/m\mathbb{Z})^\times} a \equiv 1 \pmod{m}$. How many elements in $(\mathbb{Z}/m\mathbb{Z})^\times$?

Euler's Theorem

If $\gcd(a, m) = 1$, then $f(x) = ax \bmod m$ is a bijection $(\mathbb{Z}/m\mathbb{Z})^\times \to (\mathbb{Z}/m\mathbb{Z})^\times$.

Example: $m = 5$, $a = 3$.

- $(\mathbb{Z}/5\mathbb{Z})^\times = \{1, 2, 3, 4\} = \{3, 6, 9, 12\}$.

In general, $(\mathbb{Z}/m\mathbb{Z})^\times = \{ax : x \in (\mathbb{Z}/m\mathbb{Z})^\times\}$.

Idea: Multiply all elements in both sides.

$$\prod_{x \in (\mathbb{Z}/m\mathbb{Z})^\times} x \equiv \prod_{x \in (\mathbb{Z}/m\mathbb{Z})^\times} ax \pmod{m}.$$

Each $x \in (\mathbb{Z}/m\mathbb{Z})^\times$ has an inverse, so divide! $\prod_{x \in (\mathbb{Z}/m\mathbb{Z})^\times} a \equiv 1 \pmod{m}$. How many elements in $(\mathbb{Z}/m\mathbb{Z})^\times$? $\varphi(m)$.

Euler's Theorem

Euler's Theorem: If $\gcd(a, m) = 1$, then $a^{\varphi(m)} \equiv 1 \pmod{m}$.

Euler's Theorem

Euler's Theorem: If $\gcd(a, m) = 1$, then $a^{\varphi(m)} \equiv 1 \pmod{m}$.

Consider the case when the modulus is a prime p.

Euler's Theorem

Euler's Theorem: If $\gcd(a, m) = 1$, then $a^{\varphi(m)} \equiv 1 \pmod{m}$.

Consider the case when the modulus is a prime p.

Corollary (Fermat's Little Theorem): If a is not a multiple of p, then $a^{p-1} \equiv 1 \pmod{p}$.

Euler's Theorem

Euler's Theorem: If $\gcd(a, m) = 1$, then $a^{\varphi(m)} \equiv 1 \pmod{m}$.

Consider the case when the modulus is a prime p.

Corollary (Fermat's Little Theorem): If a is not a multiple of p, then $a^{p-1} \equiv 1 \pmod{p}$.

Consider the equation $a^p \equiv a \pmod{p}$.

Euler's Theorem

Euler's Theorem: If $\gcd(a, m) = 1$, then $a^{\varphi(m)} \equiv 1 \pmod{m}$.

Consider the case when the modulus is a prime p.

Corollary (Fermat's Little Theorem): If a is not a multiple of p, then $a^{p-1} \equiv 1 \pmod{p}$.

Consider the equation $a^p \equiv a \pmod{p}$.

- If $a \equiv 0 \pmod{p}$, the equation is true.

Euler's Theorem

Euler's Theorem: If $\gcd(a, m) = 1$, then $a^{\varphi(m)} \equiv 1 \pmod{m}$.

Consider the case when the modulus is a prime p.

Corollary (Fermat's Little Theorem): If a is not a multiple of p, then $a^{p-1} \equiv 1 \pmod{p}$.

Consider the equation $a^p \equiv a \pmod{p}$.

- If $a \equiv 0 \pmod{p}$, the equation is true.
- If $a \not\equiv 0 \pmod{p}$, then the equation is true because of Fermat's Little Theorem.

Euler's Theorem

Euler's Theorem: If $\gcd(a, m) = 1$, then $a^{\varphi(m)} \equiv 1 \pmod{m}$.

Consider the case when the modulus is a prime p.

Corollary (Fermat's Little Theorem): If a is not a multiple of p, then $a^{p-1} \equiv 1 \pmod{p}$.

Consider the equation $a^p \equiv a \pmod{p}$.

- If $a \equiv 0 \pmod{p}$, the equation is true.
- If $a \not\equiv 0 \pmod{p}$, then the equation is true because of Fermat's Little Theorem.

Thus, for *all* $a \in \mathbb{Z}/p\mathbb{Z}$, $a^p \equiv a \pmod{p}$.

Exclusive OR

Remember XOR:

x	y	$x \oplus y$
1	1	0
1	0	1
0	1	1
0	0	0

Exclusive OR

Remember XOR:

x	y	$x \oplus y$
1	1	0
1	0	1
0	1	1
0	0	0

Notice: $x \oplus y = x + y \pmod 2$.

Exclusive OR

Remember XOR:

x	y	$x \oplus y$
1	1	0
1	0	1
0	1	1
0	0	0

Notice: $x \oplus y = x + y \pmod 2$.

Facts: $x \oplus x = 0$.

Exclusive OR

Remember XOR:

x	y	$x \oplus y$
1	1	0
1	0	1
0	1	1
0	0	0

Notice: $x \oplus y = x + y \pmod 2$.

Facts: $x \oplus x = 0$. Also, $x \oplus 0 = x$.

Exclusive OR

Remember XOR:

x	y	$x \oplus y$
1	1	0
1	0	1
0	1	1
0	0	0

Notice: $x \oplus y = x + y \pmod 2$.

Facts: $x \oplus x = 0$. Also, $x \oplus 0 = x$.

Consequence: $y \oplus x \oplus x = y \oplus 0 = y$.

Cryptosystems

Alice has a **message** (a bit string).

Cryptosystems

Alice has a **message** (a bit string).

- Pass it through an **encryption** function E.

Cryptosystems

Alice has a **message** (a bit string).

- Pass it through an **encryption** function E.
- Send encrypted message $E(m)$ to Bob.

Cryptosystems

Alice has a **message** (a bit string).

- Pass it through an **encryption** function E.
- Send encrypted message $E(m)$ to Bob.
- Bob passes message through **decryption** function D, so that $D(E(m)) = m$.

Cryptosystems

Alice has a **message** (a bit string).

- Pass it through an **encryption** function E.
- Send encrypted message $E(m)$ to Bob.
- Bob passes message through **decryption** function D, so that $D(E(m)) = m$.

We allow the encryption and decryption functions to depend on a **key** k: $D(E(m,k),k) = m$.

Cryptosystems

Alice has a **message** (a bit string).

- Pass it through an **encryption** function E.
- Send encrypted message $E(m)$ to Bob.
- Bob passes message through **decryption** function D, so that $D(E(m)) = m$.

We allow the encryption and decryption functions to depend on a **key** k: $D(E(m,k),k) = m$.

This implies that E must be one-to-one.

Cryptosystems

Alice has a **message** (a bit string).

- Pass it through an **encryption** function E.
- Send encrypted message $E(m)$ to Bob.
- Bob passes message through **decryption** function D, so that $D(E(m)) = m$.

We allow the encryption and decryption functions to depend on a **key** k: $D(E(m, k), k) = m$.

This implies that E must be one-to-one.

An eavesdropper Eve intercepts the message $E(m)$.

Cryptosystems

Alice has a **message** (a bit string).

- Pass it through an **encryption** function E.
- Send encrypted message $E(m)$ to Bob.
- Bob passes message through **decryption** function D, so that $D(E(m)) = m$.

We allow the encryption and decryption functions to depend on a **key** k: $D(E(m,k),k) = m$.

This implies that E must be one-to-one.

An eavesdropper Eve intercepts the message $E(m)$. We must make sure she cannot recover m.

One-Time Pad

One-Time Pad:

One-Time Pad

One-Time Pad:

- k is a bit string of the same length as m.

One-Time Pad

One-Time Pad:

- k is a bit string of the same length as m.
- Choose: $E(m,k) = D(m,k) = m \oplus k$.

One-Time Pad

One-Time Pad:

- k is a bit string of the same length as m.
- Choose: $E(m,k) = D(m,k) = m \oplus k$.
- This works since $D(E(m,k),k) = m \oplus k \oplus k = m$.

One-Time Pad

One-Time Pad:

- k is a bit string of the same length as m.
- Choose: $E(m,k) = D(m,k) = m \oplus k$.
- This works since $D(E(m,k),k) = m \oplus k \oplus k = m$.
- Advantage: If Eve does not know k, then communication is secure.

One-Time Pad

One-Time Pad:

- k is a bit string of the same length as m.
- Choose: $E(m,k) = D(m,k) = m \oplus k$.
- This works since $D(E(m,k),k) = m \oplus k \oplus k = m$.
- Advantage: If Eve does not know k, then communication is secure. All possible input messages m are possible.

One-Time Pad

One-Time Pad:

- k is a bit string of the same length as m.
- Choose: $E(m,k) = D(m,k) = m \oplus k$.
- This works since $D(E(m,k),k) = m \oplus k \oplus k = m$.
- Advantage: If Eve does not know k, then communication is secure. All possible input messages m are possible.
- Disadvantage: After one use, the pad should be discarded to maintain security.

One-Time Pad

One-Time Pad:

- k is a bit string of the same length as m.
- Choose: $E(m,k) = D(m,k) = m \oplus k$.
- This works since $D(E(m,k),k) = m \oplus k \oplus k = m$.
- Advantage: If Eve does not know k, then communication is secure. All possible input messages m are possible.
- Disadvantage: After one use, the pad should be discarded to maintain security. Annoying to use!

One-Time Pad

One-Time Pad:

- k is a bit string of the same length as m.
- Choose: $E(m,k) = D(m,k) = m \oplus k$.
- This works since $D(E(m,k),k) = m \oplus k \oplus k = m$.
- Advantage: If Eve does not know k, then communication is secure. All possible input messages m are possible.
- Disadvantage: After one use, the pad should be discarded to maintain security. Annoying to use!
- Disadvantage: Alice and Bob must agree upon the key k beforehand.

Public-Key Cryptography

In **public-key cryptography**:

Public-Key Cryptography

In **public-key cryptography**:

- There are two keys, a **public key** K, and a **private key** k.

Public-Key Cryptography

In **public-key cryptography**:

- There are two keys, a **public key** K, and a **private key** k.
- The encrypted message is $E(m, K)$ and the decryption is $D(E(m, K), k) = m$.

Public-Key Cryptography

In **public-key cryptography**:

- There are two keys, a **public key** K, and a **private key** k.
- The encrypted message is $E(m, K)$ and the decryption is $D(E(m, K), k) = m$.
- *Anyone* can send a message to Bob, since the encryption function and public key are revealed to the public.

Public-Key Cryptography

In **public-key cryptography**:

- There are two keys, a **public key** K, and a **private key** k.
- The encrypted message is $E(m, K)$ and the decryption is $D(E(m, K), k) = m$.
- *Anyone* can send a message to Bob, since the encryption function and public key are revealed to the public.
- *Only Bob* can decode the message, since only he has the private key.

Public-Key Cryptography

In **public-key cryptography**:

- There are two keys, a **public key** K, and a **private key** k.
- The encrypted message is $E(m, K)$ and the decryption is $D(E(m, K), k) = m$.
- *Anyone* can send a message to Bob, since the encryption function and public key are revealed to the public.
- *Only Bob* can decode the message, since only he has the private key.

Think of Bob as Amazon.

Public-Key Cryptography

In **public-key cryptography**:

- There are two keys, a **public key** K, and a **private key** k.
- The encrypted message is $E(m, K)$ and the decryption is $D(E(m, K), k) = m$.
- *Anyone* can send a message to Bob, since the encryption function and public key are revealed to the public.
- *Only Bob* can decode the message, since only he has the private key.

Think of Bob as Amazon. Anyone can encrypt credit card information and send it to Amazon.

Public-Key Cryptography

In **public-key cryptography**:

- There are two keys, a **public key** K, and a **private key** k.
- The encrypted message is $E(m, K)$ and the decryption is $D(E(m, K), k) = m$.
- *Anyone* can send a message to Bob, since the encryption function and public key are revealed to the public.
- *Only Bob* can decode the message, since only he has the private key.

Think of Bob as Amazon. Anyone can encrypt credit card information and send it to Amazon. Only Amazon can decrypt.

Public-Key Cryptography

In **public-key cryptography**:

- There are two keys, a **public key** K, and a **private key** k.
- The encrypted message is $E(m, K)$ and the decryption is $D(E(m, K), k) = m$.
- *Anyone* can send a message to Bob, since the encryption function and public key are revealed to the public.
- *Only Bob* can decode the message, since only he has the private key.

Think of Bob as Amazon. Anyone can encrypt credit card information and send it to Amazon. Only Amazon can decrypt.

Is public-key cryptography possible?

Public-Key Cryptography

In **public-key cryptography**:

- There are two keys, a **public key** K, and a **private key** k.
- The encrypted message is $E(m, K)$ and the decryption is $D(E(m, K), k) = m$.
- *Anyone* can send a message to Bob, since the encryption function and public key are revealed to the public.
- *Only Bob* can decode the message, since only he has the private key.

Think of Bob as Amazon. Anyone can encrypt credit card information and send it to Amazon. Only Amazon can decrypt.

Is public-key cryptography possible? Open question, but we can still try.

RSA

RSA Protocol (Rivest-Shamir-Adleman):

RSA

RSA Protocol (Rivest-Shamir-Adleman):

- Pick two large (2048-bit) distinct primes p and q.

RSA

RSA Protocol (Rivest-Shamir-Adleman):

- Pick two large (2048-bit) distinct primes p and q.
- Let $N := pq$.

RSA

RSA Protocol (Rivest-Shamir-Adleman):

- Pick two large (2048-bit) distinct primes p and q.
- Let $N := pq$. Pick an integer e.

RSA

RSA Protocol (Rivest-Shamir-Adleman):

- Pick two large (2048-bit) distinct primes p and q.
- Let $N := pq$. Pick an integer e. The public key is (N, e).

RSA

RSA Protocol (Rivest-Shamir-Adleman):

- Pick two large (2048-bit) distinct primes p and q.
- Let $N := pq$. Pick an integer e. The public key is (N, e).
- The decryption key is $d := e^{-1} \pmod{(p-1)(q-1)}$.

RSA

RSA Protocol (Rivest-Shamir-Adleman):

- Pick two large (2048-bit) distinct primes p and q.
- Let $N := pq$. Pick an integer e. The public key is (N, e).
- The decryption key is $d := e^{-1} \pmod{(p-1)(q-1)}$.
- Encryption function: $E(m) = m^e \bmod N$.

RSA

RSA Protocol (Rivest-Shamir-Adleman):

- Pick two large (2048-bit) distinct primes p and q.
- Let $N := pq$. Pick an integer e. The public key is (N, e).
- The decryption key is $d := e^{-1} \pmod{(p-1)(q-1)}$.
- Encryption function: $E(m) = m^e \bmod N$.
- Decryption function: $D(c) = c^d \bmod N$.

RSA

RSA Protocol (Rivest-Shamir-Adleman):

- Pick two large (2048-bit) distinct primes p and q.
- Let $N := pq$. Pick an integer e. The public key is (N, e).
- The decryption key is $d := e^{-1} \pmod{(p-1)(q-1)}$.
- Encryption function: $E(m) = m^e \bmod N$.
- Decryption function: $D(c) = c^d \bmod N$.

We have a lot of work to do.

RSA

RSA Protocol (Rivest-Shamir-Adleman):

- Pick two large (2048-bit) distinct primes p and q.
- Let $N := pq$. Pick an integer e. The public key is (N, e).
- The decryption key is $d := e^{-1} \pmod{(p-1)(q-1)}$.
- Encryption function: $E(m) = m^e \bmod N$.
- Decryption function: $D(c) = c^d \bmod N$.

We have a lot of work to do.

- Prove RSA works: $m^{ed} \equiv m \pmod{N}$.

RSA

RSA Protocol (Rivest-Shamir-Adleman):

- Pick two large (2048-bit) distinct primes p and q.
- Let $N := pq$. Pick an integer e. The public key is (N, e).
- The decryption key is $d := e^{-1} \pmod{(p-1)(q-1)}$.
- Encryption function: $E(m) = m^e \bmod N$.
- Decryption function: $D(c) = c^d \bmod N$.

We have a lot of work to do.

- Prove RSA works: $m^{ed} \equiv m \pmod{N}$.
- Explain why we can do the steps *efficiently*.

RSA

RSA Protocol (Rivest-Shamir-Adleman):

- Pick two large (2048-bit) distinct primes p and q.
- Let $N := pq$. Pick an integer e. The public key is (N, e).
- The decryption key is $d := e^{-1} \pmod{(p-1)(q-1)}$.
- Encryption function: $E(m) = m^e \bmod N$.
- Decryption function: $D(c) = c^d \bmod N$.

We have a lot of work to do.

- Prove RSA works: $m^{ed} \equiv m \pmod{N}$.
- Explain why we can do the steps *efficiently*.
- Explain why we think Eve cannot break it.

Correctness of RSA

Public: $(N = pq, e)$, private: $d = e^{-1} \pmod{(p-1)(q-1)}$.

Correctness of RSA

Public: $(N = pq, e)$, private: $d = e^{-1} \pmod{(p-1)(q-1)}$.

Theorem: For any $m \in \{0, 1, \ldots, N-1\}$, $m^{ed} \equiv m \pmod{N}$.

Correctness of RSA

Public: $(N = pq, e)$, private: $d = e^{-1} \pmod{(p-1)(q-1)}$.

Theorem: For any $m \in \{0, 1, \ldots, N-1\}$, $m^{ed} \equiv m \pmod{N}$.

Proof.

Correctness of RSA

Public: $(N = pq, e)$, private: $d = e^{-1} \pmod{(p-1)(q-1)}$.

Theorem: For any $m \in \{0, 1, \ldots, N-1\}$, $m^{ed} \equiv m \pmod{N}$.

Proof.

- By definition of d, $ed = 1 + k(p-1)(q-1)$ for some $k \in \mathbb{N}$.
- So, $m^{ed} = m \cdot m^{k(p-1)(q-1)}$.

Correctness of RSA

Public: $(N = pq, e)$, private: $d = e^{-1} \pmod{(p-1)(q-1)}$.

Theorem: For any $m \in \{0, 1, \ldots, N-1\}$, $m^{ed} \equiv m \pmod{N}$.

Proof.

- By definition of d, $ed = 1 + k(p-1)(q-1)$ for some $k \in \mathbb{N}$.
- So, $m^{ed} = m \cdot m^{k(p-1)(q-1)}$.
- If p divides m, then p divides $m^{ed} - m$.

Correctness of RSA

Public: $(N = pq, e)$, private: $d = e^{-1} \pmod{(p-1)(q-1)}$.

Theorem: For any $m \in \{0, 1, \ldots, N-1\}$, $m^{ed} \equiv m \pmod{N}$.

Proof.

- By definition of d, $ed = 1 + k(p-1)(q-1)$ for some $k \in \mathbb{N}$.
- So, $m^{ed} = m \cdot m^{k(p-1)(q-1)}$.
- If p divides m, then p divides $m^{ed} - m$.
- Otherwise, by Fermat's Little Theorem, $m^{p-1} \equiv 1 \pmod{p}$.

Correctness of RSA

Public: $(N = pq, e)$, private: $d = e^{-1} \pmod{(p-1)(q-1)}$.

Theorem: For any $m \in \{0, 1, \ldots, N-1\}$, $m^{ed} \equiv m \pmod{N}$.

Proof.

- By definition of d, $ed = 1 + k(p-1)(q-1)$ for some $k \in \mathbb{N}$.
- So, $m^{ed} = m \cdot m^{k(p-1)(q-1)}$.
- If p divides m, then p divides $m^{ed} - m$.
- Otherwise, by Fermat's Little Theorem, $m^{p-1} \equiv 1 \pmod{p}$. So, $m^{ed} - m = m(m^{k(p-1)(q-1)} - 1) \equiv 0 \pmod{p}$.

Correctness of RSA

Public: $(N = pq, e)$, private: $d = e^{-1} \pmod{(p-1)(q-1)}$.

Theorem: For any $m \in \{0, 1, \dots, N-1\}$, $m^{ed} \equiv m \pmod{N}$.

Proof.

- By definition of d, $ed = 1 + k(p-1)(q-1)$ for some $k \in \mathbb{N}$.
- So, $m^{ed} = m \cdot m^{k(p-1)(q-1)}$.
- If p divides m, then p divides $m^{ed} - m$.
- Otherwise, by Fermat's Little Theorem, $m^{p-1} \equiv 1 \pmod{p}$. So, $m^{ed} - m = m(m^{k(p-1)(q-1)} - 1) \equiv 0 \pmod{p}$.
- In both cases, $m^{ed} - m$ is divisible by p.

Correctness of RSA

Public: $(N = pq, e)$, private: $d = e^{-1} \pmod{(p-1)(q-1)}$.

Theorem: For any $m \in \{0, 1, \ldots, N-1\}$, $m^{ed} \equiv m \pmod{N}$.

Proof.

- By definition of d, $ed = 1 + k(p-1)(q-1)$ for some $k \in \mathbb{N}$.
- So, $m^{ed} = m \cdot m^{k(p-1)(q-1)}$.
- If p divides m, then p divides $m^{ed} - m$.
- Otherwise, by Fermat's Little Theorem, $m^{p-1} \equiv 1 \pmod{p}$.
  So, $m^{ed} - m = m(m^{k(p-1)(q-1)} - 1) \equiv 0 \pmod{p}$.
- In both cases, $m^{ed} - m$ is divisible by p.
- Similarly, $m^{ed} - m$ is divisible by q.

Correctness of RSA

Public: $(N = pq, e)$, private: $d = e^{-1} \pmod{(p-1)(q-1)}$.

Theorem: For any $m \in \{0, 1, \ldots, N-1\}$, $m^{ed} \equiv m \pmod{N}$.

Proof.

- By definition of d, $ed = 1 + k(p-1)(q-1)$ for some $k \in \mathbb{N}$.
- So, $m^{ed} = m \cdot m^{k(p-1)(q-1)}$.
- If p divides m, then p divides $m^{ed} - m$.
- Otherwise, by Fermat's Little Theorem, $m^{p-1} \equiv 1 \pmod{p}$.
  So, $m^{ed} - m = m(m^{k(p-1)(q-1)} - 1) \equiv 0 \pmod{p}$.
- In both cases, $m^{ed} - m$ is divisible by p.
- Similarly, $m^{ed} - m$ is divisible by q.
- Since $p \neq q$, then $m^{ed} - m$ is divisible by $pq = N$, i.e.,
  $m^{ed} \equiv m \pmod{N}$. $\square$

Another Look at Correctness

Given any $m \in \{0, 1, \ldots, N-1\}$,
$D(E(m)) = E(D(m)) = m^{ed} = m$.

Another Look at Correctness

Given any $m \in \{0, 1, \ldots, N-1\}$,
$D(E(m)) = E(D(m)) = m^{ed} = m$.

The maps E and D are bijections $\mathbb{Z}/N\mathbb{Z} \to \mathbb{Z}/N\mathbb{Z}$.

Another Look at Correctness

Given any $m \in \{0, 1, \ldots, N-1\}$,
$D(E(m)) = E(D(m)) = m^{ed} = m$.

The maps E and D are bijections $\mathbb{Z}/N\mathbb{Z} \to \mathbb{Z}/N\mathbb{Z}$.

The key idea behind cryptography is that E is easy to compute but hard to invert.

Implementing RSA Is Fast

Pick two 2048-bit prime numbers.

Implementing RSA Is Fast

Pick two 2048-bit prime numbers.

- How?

Implementing RSA Is Fast

Pick two 2048-bit prime numbers.

- How? By the Prime Number Theorem, the “probability” that a random number between 1 and N is prime is $\approx 1/\ln N$.

Implementing RSA Is Fast

Pick two 2048-bit prime numbers.

- How? By the Prime Number Theorem, the “probability” that a random number between 1 and N is prime is $\approx 1/\ln N$.
- We need to generate and check $O(\ln N)$ primes.

Implementing RSA Is Fast

Pick two 2048-bit prime numbers.

- How? By the Prime Number Theorem, the "probability" that a random number between 1 and N is prime is $\approx 1/\ln N$.
- We need to generate and check $O(\ln N)$ primes.
- This is linear in the number of bits!

Implementing RSA Is Fast

Pick two 2048-bit prime numbers.

- How? By the Prime Number Theorem, the "probability" that a random number between 1 and N is prime is $\approx 1/\ln N$.
- We need to generate and check $O(\ln N)$ primes.
- This is linear in the number of bits!
- Use a randomized primality test: test if N is prime in time which is polynomial in the number of bits of N.

Implementing RSA Is Fast

Pick two 2048-bit prime numbers.

- How? By the Prime Number Theorem, the "probability" that a random number between 1 and N is prime is $\approx 1/\ln N$.
- We need to generate and check $O(\ln N)$ primes.
- This is linear in the number of bits!
- Use a randomized primality test: test if N is prime in time which is polynomial in the number of bits of N.
- Works with very high probability.

Implementing RSA Is Fast

Pick two 2048-bit prime numbers.

- How? By the Prime Number Theorem, the "probability" that a random number between 1 and N is prime is $\approx 1/\ln N$.
- We need to generate and check $O(\ln N)$ primes.
- This is linear in the number of bits!
- Use a randomized primality test: test if N is prime in time which is polynomial in the number of bits of N.
- Works with very high probability. The probability of failure can be made as low as the probability of meteor crash!

Implementing RSA Is Fast

Pick two 2048-bit prime numbers.

- How? By the Prime Number Theorem, the "probability" that a random number between 1 and N is prime is $\approx 1/\ln N$.
- We need to generate and check $O(\ln N)$ primes.
- This is linear in the number of bits!
- Use a randomized primality test: test if N is prime in time which is polynomial in the number of bits of N.
- Works with very high probability. The probability of failure can be made as low as the probability of meteor crash!

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

Implementing RSA Is Fast

Pick two 2048-bit prime numbers.

- How? By the Prime Number Theorem, the "probability" that a random number between 1 and N is prime is $\approx 1/\ln N$.
- We need to generate and check $O(\ln N)$ primes.
- This is linear in the number of bits!
- Use a randomized primality test: test if N is prime in time which is polynomial in the number of bits of N.
- Works with very high probability. The probability of failure can be made as low as the probability of meteor crash!

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

- Extended Euclid is fast!

Implementing RSA Is Fast

Pick two 2048-bit prime numbers.

- How? By the Prime Number Theorem, the "probability" that a random number between 1 and N is prime is $\approx 1/\ln N$.
- We need to generate and check $O(\ln N)$ primes.
- This is linear in the number of bits!
- Use a randomized primality test: test if N is prime in time which is polynomial in the number of bits of N.
- Works with very high probability. The probability of failure can be made as low as the probability of meteor crash!

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

- Extended Euclid is fast!

Compute $m^e \bmod N$ and $(m^e)^d \bmod N$.

Implementing RSA Is Fast

Pick two 2048-bit prime numbers.

- How? By the Prime Number Theorem, the “probability” that a random number between 1 and N is prime is $\approx 1/\ln N$.
- We need to generate and check $O(\ln N)$ primes.
- This is linear in the number of bits!
- Use a randomized primality test: test if N is prime in time which is polynomial in the number of bits of N.
- Works with very high probability. The probability of failure can be made as low as the probability of meteor crash!

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

- Extended Euclid is fast!

Compute $m^e \bmod N$ and $(m^e)^d \bmod N$.

- Repeated squaring! (fast modular exponentiation)

Fast Modular Exponentiation

What is $2^{1000000} \pmod{12}$?

Fast Modular Exponentiation

What is $2^{1000000} \pmod{12}$?

Multiply 2 by itself, a million times.

Fast Modular Exponentiation

What is $2^{1000000} \pmod{12}$?

Multiply 2 by itself, a million times. Wait!

Fast Modular Exponentiation

What is $2^{1000000} \pmod{12}$?

Multiply 2 by itself, a million times. Wait! Use repeated squaring.

Fast Modular Exponentiation

What is $2^{1000000} \pmod{12}$?

Multiply 2 by itself, a million times. Wait! Use repeated squaring. $2^{1000000} = 4^{500000} = 16^{250000} = \cdots$

Fast Modular Exponentiation

What is $2^{1000000} \pmod{12}$?

Multiply 2 by itself, a million times. Wait! Use repeated squaring. $2^{1000000} = 4^{500000} = 16^{250000} = \cdots$

Insight: 16^{250000} is 250000 products of 16.

Fast Modular Exponentiation

What is $2^{1000000} \pmod{12}$?

Multiply 2 by itself, a million times. Wait! Use repeated squaring. $2^{1000000} = 4^{500000} = 16^{250000} = \cdots$

Insight: 16^{250000} is 250000 products of 16. But $16 \equiv 4 \pmod{12}$.

Fast Modular Exponentiation

What is $2^{1000000} \pmod{12}$?

Multiply 2 by itself, a million times. Wait! Use repeated squaring. $2^{1000000} = 4^{500000} = 16^{250000} = \cdots$

Insight: 16^{250000} is 250000 products of 16. But $16 \equiv 4 \pmod{12}$. So, $16 \cdot 16 \cdot 16 \cdots \pmod{12} \equiv 4 \cdot 4 \cdot 4 \cdots \pmod{12}$.

Fast Modular Exponentiation

What is $2^{1000000} \pmod{12}$?

Multiply 2 by itself, a million times. Wait! Use repeated squaring. $2^{1000000} = 4^{500000} = 16^{250000} = \cdots$

Insight: 16^{250000} is 250000 products of 16. But $16 \equiv 4 \pmod{12}$. So, $16 \cdot 16 \cdot 16 \cdots \pmod{12} \equiv 4 \cdot 4 \cdot 4 \cdots \pmod{12}$.

Continue: $4^{250000} \equiv 16^{125000}$.

Fast Modular Exponentiation

What is $2^{1000000} \pmod{12}$?

Multiply 2 by itself, a million times. Wait! Use repeated squaring. $2^{1000000} = 4^{500000} = 16^{250000} = \cdots$

Insight: 16^{250000} is 250000 products of 16. But $16 \equiv 4 \pmod{12}$. So, $16 \cdot 16 \cdot 16 \cdots \pmod{12} \equiv 4 \cdot 4 \cdot 4 \cdots \pmod{12}$.

Continue: $4^{250000} \equiv 16^{125000}$. Reduce modulo 12 again: 4^{125000}.

Fast Modular Exponentiation

What is $2^{1000000} \pmod{12}$?

Multiply 2 by itself, a million times. Wait! Use repeated squaring. $2^{1000000} = 4^{500000} = 16^{250000} = \cdots$

Insight: 16^{250000} is 250000 products of 16. But $16 \equiv 4 \pmod{12}$. So, $16 \cdot 16 \cdot 16 \cdots \pmod{12} \equiv 4 \cdot 4 \cdot 4 \cdots \pmod{12}$.

Continue: $4^{250000} \equiv 16^{125000}$. Reduce modulo 12 again: 4^{125000}.

Repeated squaring:

Fast Modular Exponentiation

What is $2^{1000000} \pmod{12}$?

Multiply 2 by itself, a million times. Wait! Use repeated squaring. $2^{1000000} = 4^{500000} = 16^{250000} = \cdots$

Insight: 16^{250000} is 250000 products of 16. But $16 \equiv 4 \pmod{12}$. So, $16 \cdot 16 \cdot 16 \cdots \pmod{12} \equiv 4 \cdot 4 \cdot 4 \cdots \pmod{12}$.

Continue: $4^{250000} \equiv 16^{125000}$. Reduce modulo 12 again: 4^{125000}.

Repeated squaring:

- Square the base and cut the exponent in half.

Fast Modular Exponentiation

What is $2^{1000000} \pmod{12}$?

Multiply 2 by itself, a million times. Wait! Use repeated squaring. $2^{1000000} = 4^{500000} = 16^{250000} = \cdots$

Insight: 16^{250000} is 250000 products of 16. But $16 \equiv 4 \pmod{12}$. So, $16 \cdot 16 \cdot 16 \cdots \pmod{12} \equiv 4 \cdot 4 \cdot 4 \cdots \pmod{12}$.

Continue: $4^{250000} \equiv 16^{125000}$. Reduce modulo 12 again: 4^{125000}.

Repeated squaring:

- Square the base and cut the exponent in half.
- If the base exceeds m, reduce the base modulo m.

Fast Modular Exponentiation

What is $2^{1000000} \pmod{12}$?

Multiply 2 by itself, a million times. Wait! Use repeated squaring. $2^{1000000} = 4^{500000} = 16^{250000} = \cdots$

Insight: 16^{250000} is 250000 products of 16. But $16 \equiv 4 \pmod{12}$. So, $16 \cdot 16 \cdot 16 \cdots \pmod{12} \equiv 4 \cdot 4 \cdot 4 \cdots \pmod{12}$.

Continue: $4^{250000} \equiv 16^{125000}$. Reduce modulo 12 again: 4^{125000}.

Repeated squaring:

- Square the base and cut the exponent in half.
- If the base exceeds m, reduce the base modulo m.

What if there is an odd exponent, 2^{17}?

Fast Modular Exponentiation

What is $2^{1000000} \pmod{12}$?

Multiply 2 by itself, a million times. Wait! Use repeated squaring. $2^{1000000} = 4^{500000} = 16^{250000} = \cdots$

Insight: 16^{250000} is 250000 products of 16. But $16 \equiv 4 \pmod{12}$. So, $16 \cdot 16 \cdot 16 \cdots \pmod{12} \equiv 4 \cdot 4 \cdot 4 \cdots \pmod{12}$.

Continue: $4^{250000} \equiv 16^{125000}$. Reduce modulo 12 again: 4^{125000}.

Repeated squaring:

- Square the base and cut the exponent in half.
- If the base exceeds m, reduce the base modulo m.

What if there is an odd exponent, 2^{17}? Write this as $2 \cdot 2^{16}$.

Breaking RSA Is Slow?

Cryptograph relies on *assumptions*.

Breaking RSA Is Slow?

Cryptograph relies on *assumptions*.

RSA Assumption: Given N, e, and $m^e \bmod N$, there is no efficient algorithm for finding m.

Breaking RSA Is Slow?

Cryptograph relies on *assumptions*.

RSA Assumption: Given N, e, and $m^e \bmod N$, there is no efficient algorithm for finding m.

In other words, we believe Eve cannot break RSA.

Breaking RSA Is Slow?

Cryptograph relies on *assumptions*.

RSA Assumption: Given N, e, and $m^e \bmod N$, there is no efficient algorithm for finding m.

In other words, we believe Eve cannot break RSA.

- Why do we believe this?

Breaking RSA Is Slow?

Cryptograph relies on *assumptions*.

RSA Assumption: Given N, e, and $m^e \bmod N$, there is no efficient algorithm for finding m.

In other words, we believe Eve cannot break RSA.

- Why do we believe this? One way to break RSA is to factor $N = pq$ to get $(p-1)(q-1)$ and compute d yourself.

Breaking RSA Is Slow?

Cryptograph relies on *assumptions*.

RSA Assumption: Given N, e, and $m^e \bmod N$, there is no efficient algorithm for finding m.

In other words, we believe Eve cannot break RSA.

- Why do we believe this? One way to break RSA is to factor $N = pq$ to get $(p-1)(q-1)$ and compute d yourself.
- How do we factor N?

Breaking RSA Is Slow?

Cryptograph relies on *assumptions*.

RSA Assumption: Given N, e, and $m^e \bmod N$, there is no efficient algorithm for finding m.

In other words, we believe Eve cannot break RSA.

- Why do we believe this? One way to break RSA is to factor $N = pq$ to get $(p-1)(q-1)$ and compute d yourself.
- How do we factor N? There are no good algorithms known!

Breaking RSA Is Slow?

Cryptograph relies on *assumptions*.

RSA Assumption: Given N, e, and $m^e \bmod N$, there is no efficient algorithm for finding m.

In other words, we believe Eve cannot break RSA.

- Why do we believe this? One way to break RSA is to factor $N = pq$ to get $(p-1)(q-1)$ and compute d yourself.
- How do we factor N? There are no good algorithms known!
- The naïve algorithms for factoring N (brute force) take time exponential in the number of bits.

Breaking RSA Is Slow?

Cryptograph relies on *assumptions*.

RSA Assumption: Given N, e, and $m^e \bmod N$, there is no efficient algorithm for finding m.

In other words, we believe Eve cannot break RSA.

- Why do we believe this? One way to break RSA is to factor $N = pq$ to get $(p-1)(q-1)$ and compute d yourself.
- How do we factor N? There are no good algorithms known!
- The naïve algorithms for factoring N (brute force) take time exponential in the number of bits.
- No one has ever factored a 2048-bit RSA key before (without knowing p and q beforehand).

Flipping RSA: Digital Signatures

Suppose I am Spiderman.

Flipping RSA: Digital Signatures

Suppose I am Spiderman.

- Spiderman has a private key.

Flipping RSA: Digital Signatures

Suppose I am Spiderman.

- Spiderman has a private key. (He accepts donations, so he needs to secure credit card transactions.)

Flipping RSA: Digital Signatures

Suppose I am Spiderman.

- Spiderman has a private key. (He accepts donations, so he needs to secure credit card transactions.)
- Now I want to reveal my identity to the world as Spiderman.

Flipping RSA: Digital Signatures

Suppose I am Spiderman.

- Spiderman has a private key. (He accepts donations, so he needs to secure credit card transactions.)
- Now I want to reveal my identity to the world as Spiderman.
- Videos can be faked.

Flipping RSA: Digital Signatures

Suppose I am Spiderman.

- Spiderman has a private key. (He accepts donations, so he needs to secure credit card transactions.)
- Now I want to reveal my identity to the world as Spiderman.
- Videos can be faked. The public wants proof.

Flipping RSA: Digital Signatures

Suppose I am Spiderman.

- Spiderman has a private key. (He accepts donations, so he needs to secure credit card transactions.)
- Now I want to reveal my identity to the world as Spiderman.
- Videos can be faked. The public wants proof.
- One suggestion: I could reveal my private key, then everyone will believe me.

Flipping RSA: Digital Signatures

Suppose I am Spiderman.

- Spiderman has a private key. (He accepts donations, so he needs to secure credit card transactions.)
- Now I want to reveal my identity to the world as Spiderman.
- Videos can be faked. The public wants proof.
- One suggestion: I could reveal my private key, then everyone will believe me.
- But what if I do not want to reveal my private key?

Flipping RSA: Digital Signatures

Suppose I am Spiderman.

- Spiderman has a private key. (He accepts donations, so he needs to secure credit card transactions.)
- Now I want to reveal my identity to the world as Spiderman.
- Videos can be faked. The public wants proof.
- One suggestion: I could reveal my private key, then everyone will believe me.
- But what if I do not want to reveal my private key?

Now introducing **digital signatures**.

Digital Signatures

The public chooses a message m, e.g., “Spiderman is cool.” (encode in binary)

Digital Signatures

The public chooses a message m, e.g., “Spiderman is cool.” (encode in binary)

The public asks Spiderman for $m^d \bmod N$.

Digital Signatures

The public chooses a message m, e.g., “Spiderman is cool.” (encode in binary)

The public asks Spiderman for $m^d \bmod N$. Then, they can verify that $(m^d)^e \equiv m \bmod N$.

Digital Signatures

The public chooses a message m, e.g., "Spiderman is cool." (encode in binary)

The public asks Spiderman for $m^d \bmod N$. Then, they can verify that $(m^d)^e \equiv m \bmod N$.

What if I really am Spiderman?

Digital Signatures

The public chooses a message m, e.g., “Spiderman is cool.” (encode in binary)

The public asks Spiderman for $m^d \bmod N$. Then, they can verify that $(m^d)^e \equiv m \bmod N$.

What if I really am Spiderman?

- Computing $m^d \bmod N$ is no problem for me.

Digital Signatures

The public chooses a message m, e.g., “Spiderman is cool.” (encode in binary)

The public asks Spiderman for $m^d \bmod N$. Then, they can verify that $(m^d)^e \equiv m \bmod N$.

What if I really am Spiderman?

- Computing $m^d \bmod N$ is no problem for me. I can sign the message.

Digital Signatures

The public chooses a message m, e.g., “Spiderman is cool.” (encode in binary)

The public asks Spiderman for $m^d \bmod N$. Then, they can verify that $(m^d)^e \equiv m \bmod N$.

What if I really am Spiderman?

- Computing $m^d \bmod N$ is no problem for me. I can sign the message.

What if I am a fraud?

Digital Signatures

The public chooses a message m, e.g., “Spiderman is cool.” (encode in binary)

The public asks Spiderman for $m^d \bmod N$. Then, they can verify that $(m^d)^e \equiv m \bmod N$.

What if I really am Spiderman?

- Computing $m^d \bmod N$ is no problem for me. I can sign the message.

What if I am a fraud?

- I do not know d.

Digital Signatures

The public chooses a message m, e.g., “Spiderman is cool.” (encode in binary)

The public asks Spiderman for $m^d \bmod N$. Then, they can verify that $(m^d)^e \equiv m \bmod N$.

What if I really am Spiderman?

- Computing $m^d \bmod N$ is no problem for me. I can sign the message.

What if I am a fraud?

- I do not know d.
- I spend the rest of my life exhaustively looking through all $x \in \mathbb{Z}/N\mathbb{Z}$ until I find something with $x^e \equiv m \pmod N$.

Digital Signatures

The public chooses a message m, e.g., “Spiderman is cool.” (encode in binary)

The public asks Spiderman for $m^d \bmod N$. Then, they can verify that $(m^d)^e \equiv m \bmod N$.

What if I really am Spiderman?

- Computing $m^d \bmod N$ is no problem for me. I can sign the message.

What if I am a fraud?

- I do not know d.
- I spend the rest of my life exhaustively looking through all $x \in \mathbb{Z}/N\mathbb{Z}$ until I find something with $x^e \equiv m \pmod{N}$.

Takeaway: No one but Spiderman can sign the message.

Breaking Textbook RSA

I make a purchase on Amazon.

Breaking Textbook RSA

I make a purchase on Amazon. Amazon's public key is (N, e).

Breaking Textbook RSA

I make a purchase on Amazon. Amazon's public key is (N, e).

- I take my credit card number m and encrypt it: $E(m) = m^e \pmod{N}$.

Breaking Textbook RSA

I make a purchase on Amazon. Amazon's public key is (N, e).

- I take my credit card number m and encrypt it: $E(m) = m^e \pmod{N}$.
- I send $E(m)$ to Amazon.

Breaking Textbook RSA

I make a purchase on Amazon. Amazon's public key is (N, e).

- I take my credit card number m and encrypt it: $E(m) = m^e \pmod{N}$.
- I send $E(m)$ to Amazon.
- Amazon decrypts my credit card number and completes my transaction.

Breaking Textbook RSA

I make a purchase on Amazon. Amazon's public key is (N, e).

- I take my credit card number m and encrypt it: $E(m) = m^e \pmod{N}$.
- I send $E(m)$ to Amazon.
- Amazon decrypts my credit card number and completes my transaction. I get my favorite show in BD format.

Breaking Textbook RSA

I make a purchase on Amazon. Amazon's public key is (N, e).

- I take my credit card number m and encrypt it: $E(m) = m^e \pmod{N}$.
- I send $E(m)$ to Amazon.
- Amazon decrypts my credit card number and completes my transaction. I get my favorite show in BD format.
- But Eve was listening to our communication and now she knows $E(m)$.

Breaking Textbook RSA

I make a purchase on Amazon. Amazon's public key is (N, e).

- I take my credit card number m and encrypt it: $E(m) = m^e \pmod{N}$.
- I send $E(m)$ to Amazon.
- Amazon decrypts my credit card number and completes my transaction. I get my favorite show in BD format.
- But Eve was listening to our communication and now she knows $E(m)$.
- Eve sends $E(m)$ to Amazon.

Breaking Textbook RSA

I make a purchase on Amazon. Amazon's public key is (N, e).

- I take my credit card number m and encrypt it: $E(m) = m^e \pmod{N}$.
- I send $E(m)$ to Amazon.
- Amazon decrypts my credit card number and completes my transaction. I get my favorite show in BD format.
- But Eve was listening to our communication and now she knows $E(m)$.
- Eve sends $E(m)$ to Amazon.
- Now Eve can use my credit card.

Breaking Textbook RSA

I make a purchase on Amazon. Amazon's public key is (N, e).

- I take my credit card number m and encrypt it: $E(m) = m^e \pmod{N}$.
- I send $E(m)$ to Amazon.
- Amazon decrypts my credit card number and completes my transaction. I get my favorite show in BD format.
- But Eve was listening to our communication and now she knows $E(m)$.
- Eve sends $E(m)$ to Amazon.
- Now Eve can use my credit card.
- Oops!

Another RSA Attack

- I send $E(m) = m^e \bmod N$ to Amazon.

Another RSA Attack

- I send $E(m) = m^e \bmod N$ to Amazon.
- Eve intercepts the message.

Another RSA Attack

- I send $E(m) = m^e \bmod N$ to Amazon.
- Eve intercepts the message.
- Eve chooses some number r and asks Amazon to decrypt $E(m)r^e$ for her.

Another RSA Attack

- I send $E(m) = m^e \bmod N$ to Amazon.
- Eve intercepts the message.
- Eve chooses some number r and asks Amazon to decrypt $E(m)r^e$ for her.
- $E(m)r^e$ does not look like a suspicious string…

Another RSA Attack

- I send $E(m) = m^e \bmod N$ to Amazon.
- Eve intercepts the message.
- Eve chooses some number r and asks Amazon to decrypt $E(m)r^e$ for her.
- $E(m)r^e$ does not look like a suspicious string. . . so Amazon says why not.

Another RSA Attack

- I send $E(m) = m^e \bmod N$ to Amazon.
- Eve intercepts the message.
- Eve chooses some number r and asks Amazon to decrypt $E(m)r^e$ for her.
- $E(m)r^e$ does not look like a suspicious string...so Amazon says why not.
- Amazon sends back $[E(m)r^e]^d \bmod N = mr \bmod N$.

Another RSA Attack

- I send $E(m) = m^e \bmod N$ to Amazon.
- Eve intercepts the message.
- Eve chooses some number r and asks Amazon to decrypt $E(m)r^e$ for her.
- $E(m)r^e$ does not look like a suspicious string...so Amazon says why not.
- Amazon sends back $[E(m)r^e]^d \bmod N = mr \bmod N$.
- Eve calculates $r^{-1} \pmod N$ and uses this to recover the message m.

Another RSA Attack

- I send $E(m) = m^e \bmod N$ to Amazon.
- Eve intercepts the message.
- Eve chooses some number r and asks Amazon to decrypt $E(m)r^e$ for her.
- $E(m)r^e$ does not look like a suspicious string…so Amazon says why not.
- Amazon sends back $[E(m)r^e]^d \bmod N = mr \bmod N$.
- Eve calculates $r^{-1} \pmod N$ and uses this to recover the message m.
- Now Eve knows my credit card number.

Another RSA Attack

- I send $E(m) = m^e \bmod N$ to Amazon.
- Eve intercepts the message.
- Eve chooses some number r and asks Amazon to decrypt $E(m)r^e$ for her.
- $E(m)r^e$ does not look like a suspicious string...so Amazon says why not.
- Amazon sends back $[E(m)r^e]^d \bmod N = mr \bmod N$.
- Eve calculates $r^{-1} \pmod{N}$ and uses this to recover the message m.
- Now Eve knows my credit card number.
- Double oops!

RSA with Padding

Simple idea: Before you encrypt the message m, *pad it with some randomly generated string* s.

RSA with Padding

Simple idea: Before you encrypt the message m, *pad it with some randomly generated string s*.

Send over $E(\text{concatenate}(m, s))$.

RSA with Padding

Simple idea: Before you encrypt the message m, *pad it with some randomly generated string* s.

Send over $E(\text{concatenate}(m, s))$.

Even if we send the same message twice, the encrypted messages are different.

RSA with Padding

Simple idea: Before you encrypt the message m, *pad it with some randomly generated string s*.

Send over $E(\mathtt{concatenate}(m, s))$.

Even if we send the same message twice, the encrypted messages are different.

- So, if Eve sends the same encrypted message as before, it looks suspicious!

RSA with Padding

Simple idea: Before you encrypt the message m, *pad it with some randomly generated string* s.

Send over $E(\texttt{concatenate}(m, s))$.

Even if we send the same message twice, the encrypted messages are different.

- So, if Eve sends the same encrypted message as before, it looks suspicious!

To avoid the second attack, be careful.

RSA with Padding

Simple idea: Before you encrypt the message m, *pad it with some randomly generated string* s.

Send over $E(\text{concatenate}(m, s))$.

Even if we send the same message twice, the encrypted messages are different.

- So, if Eve sends the same encrypted message as before, it looks suspicious!

To avoid the second attack, be careful. Amazon should give out as little information as possible.

Summary

- $\varphi(1) := 1$ and for $m \geq 2$, $\varphi(m) := |(\mathbb{Z}/m\mathbb{Z})^\times|$.
- Euler's Theorem: If $\gcd(a, m) = 1$, then $a^{\varphi(m)} \equiv 1 \pmod{m}$.
- RSA: Pick two large primes p and q and an integer e, encrypt by $m^e \bmod pq$, and decrypt by $m^{ed} \equiv m \pmod{pq}$.
- RSA can also be used for digital signatures.
- RSA is currently not breakable (use padding though).