BILL TAPE LECTURE

Diffie-Helman Key Exchange

Summary of Where We Are

Summary of Where We Are

1. Finding primes p such that $p - 1 = 2q$, q a prime, EASY

Summary of Where We Are

1. Finding primes p such that $p - 1 = 2q$, q a prime, EASY
2. Given such a p, finding generator g, EASY.

Summary of Where We Are

1. Finding primes p such that $p - 1 = 2q$, q a prime, EASY
2. Given such a p, finding generator g, EASY.
3. Given such a p, finding generator $g \in \{\frac{p}{3}, \ldots, \frac{2p}{3}\}$ EASY.

Summary of Where We Are

1. Finding primes p such that $p-1=2q$, q a prime, EASY
2. Given such a p, finding generator g, EASY.
3. Given such a p, finding generator $g \in \{\frac{p}{3}, \ldots, \frac{2p}{3}\}$ EASY.
4. Given p, g, a finding $g^a \pmod{p}$ EASY.

Summary of Where We Are

1. Finding primes p such that $p - 1 = 2q$, q a prime, EASY
2. Given such a p, finding generator g, EASY.
3. Given such a p, finding generator $g \in \{\frac{p}{3}, \ldots, \frac{2p}{3}\}$ EASY.
4. Given p, g, a finding $g^a \pmod{p}$ EASY.
5. The following problem thought to be hard:
   Input prime p, generator $g \in \{\frac{p}{3}, \ldots, \frac{2p}{3}\}$, and a.
   Output The x such that $g^x \equiv a \pmod{p}$

The Diffie-Hellman Key Exchange

Alice & Bob want to establish a secret s w/o meeting.

The Diffie-Hellman Key Exchange

Alice & Bob want to establish a secret s w/o meeting.
Security parameter L.

The Diffie-Hellman Key Exchange

Alice & Bob want to establish a secret s w/o meeting.
Security parameter L.

1. Alice finds a (p, g), p of length L, g gen for $\mathbb{Z}_p^*$.

The Diffie-Hellman Key Exchange

Alice & Bob want to establish a secret s w/o meeting.
Security parameter L.

1. Alice finds a (p, g), p of length L, g gen for $\mathbb{Z}_p^*$.
2. Alice sends (p, g) to Bob (Eve can see it).

The Diffie-Hellman Key Exchange

Alice & Bob want to establish a secret s w/o meeting.
Security parameter L.

1. Alice finds a (p, g), p of length L, g gen for $\mathbb{Z}_p^*$.
2. Alice sends (p, g) to Bob (Eve can see it).
3. Alice picks rand a. Alice computes $g^a \pmod{p}$ and sends it to Bob (Eve can see it).

The Diffie-Hellman Key Exchange

Alice & Bob want to establish a secret s w/o meeting.
Security parameter L.

1. Alice finds a (p, g), p of length L, g gen for $\mathbb{Z}_p^*$.
2. Alice sends (p, g) to Bob (Eve can see it).
3. Alice picks rand a. Alice computes $g^a \pmod{p}$ and sends it to Bob (Eve can see it).
4. Bob picks rand b. Bob computes $g^b \pmod{p}$ and sends it to Alice (Eve can see it).

The Diffie-Hellman Key Exchange

Alice & Bob want to establish a secret s w/o meeting.
Security parameter L.

1. Alice finds a (p, g), p of length L, g gen for $\mathbb{Z}_p^*$.
2. Alice sends (p, g) to Bob (Eve can see it).
3. Alice picks rand a. Alice computes $g^a \pmod{p}$ and sends it to Bob (Eve can see it).
4. Bob picks rand b. Bob computes $g^b \pmod{p}$ and sends it to Alice (Eve can see it).
5. Alice computes $(g^b)^a = g^{ab} \pmod{p}$.

The Diffie-Hellman Key Exchange

Alice & Bob want to establish a secret s w/o meeting.
Security parameter L.

1. Alice finds a (p, g), p of length L, g gen for $\mathbb{Z}_p^*$.
2. Alice sends (p, g) to Bob (Eve can see it).
3. Alice picks rand a. Alice computes $g^a \pmod{p}$ and sends it to Bob (Eve can see it).
4. Bob picks rand b. Bob computes $g^b \pmod{p}$ and sends it to Alice (Eve can see it).
5. Alice computes $(g^b)^a = g^{ab} \pmod{p}$.
6. Bob computes $(g^a)^b = g^{ab} \pmod{p}$.

The Diffie-Hellman Key Exchange

Alice & Bob want to establish a secret s w/o meeting.
Security parameter L.

1. Alice finds a (p, g), p of length L, g gen for $\mathbb{Z}_p^*$.
2. Alice sends (p, g) to Bob (Eve can see it).
3. Alice picks rand a. Alice computes $g^a \pmod{p}$ and sends it to Bob (Eve can see it).
4. Bob picks rand b. Bob computes $g^b \pmod{p}$ and sends it to Alice (Eve can see it).
5. Alice computes $(g^b)^a = g^{ab} \pmod{p}$.
6. Bob computes $(g^a)^b = g^{ab} \pmod{p}$.
7. $s = g^{ab}$ is the shared secret.

The Diffie-Hellman Key Exchange

Alice & Bob want to establish a secret s w/o meeting.
Security parameter L.

1. Alice finds a (p, g), p of length L, g gen for $\mathbb{Z}_p^*$.
2. Alice sends (p, g) to Bob (Eve can see it).
3. Alice picks rand a. Alice computes $g^a \pmod{p}$ and sends it to Bob (Eve can see it).
4. Bob picks rand b. Bob computes $g^b \pmod{p}$ and sends it to Alice (Eve can see it).
5. Alice computes $(g^b)^a = g^{ab} \pmod{p}$.
6. Bob computes $(g^a)^b = g^{ab} \pmod{p}$.
7. $s = g^{ab}$ is the shared secret.

PRO Alice and Bob can execute the protocol easily.

The Diffie-Hellman Key Exchange

Alice & Bob want to establish a secret s w/o meeting.
Security parameter L.

1. Alice finds a (p, g), p of length L, g gen for $\mathbb{Z}_p^*$.
2. Alice sends (p, g) to Bob (Eve can see it).
3. Alice picks rand a. Alice computes $g^a \pmod{p}$ and sends it to Bob (Eve can see it).
4. Bob picks rand b. Bob computes $g^b \pmod{p}$ and sends it to Alice (Eve can see it).
5. Alice computes $(g^b)^a = g^{ab} \pmod{p}$.
6. Bob computes $(g^a)^b = g^{ab} \pmod{p}$.
7. $s = g^{ab}$ is the shared secret.

PRO Alice and Bob can execute the protocol easily.
Biggest PRO Alice and Bob never had to meet!

The Diffie-Hellman Key Exchange

Alice & Bob want to establish a secret s w/o meeting.
Security parameter L.

1. Alice finds a (p, g), p of length L, g gen for $\mathbb{Z}_p^*$.
2. Alice sends (p, g) to Bob (Eve can see it).
3. Alice picks rand a. Alice computes $g^a \pmod{p}$ and sends it to Bob (Eve can see it).
4. Bob picks rand b. Bob computes $g^b \pmod{p}$ and sends it to Alice (Eve can see it).
5. Alice computes $(g^b)^a = g^{ab} \pmod{p}$.
6. Bob computes $(g^a)^b = g^{ab} \pmod{p}$.
7. $s = g^{ab}$ is the shared secret.

PRO Alice and Bob can execute the protocol easily.
Biggest PRO Alice and Bob never had to meet!
Question Can Eve find out s?

What Do We Really Know about Diffie-Hellman?

If Eve can compute Discrete Log quickly then she can crack DH:

What Do We Really Know about Diffie-Hellman?

If Eve can compute Discrete Log quickly then she can crack DH:

1. Eve sees g^a, g^b.

What Do We Really Know about Diffie-Hellman?

If Eve can compute Discrete Log quickly then she can crack DH:

1. Eve sees g^a, g^b.
2. Eve computes Discrete Log to find a, b.

What Do We Really Know about Diffie-Hellman?

If Eve can compute Discrete Log quickly then she can crack DH:

1. Eve sees g^a, g^b.
2. Eve computes Discrete Log to find a, b.
3. Eve computes $g^{ab} \pmod{p}$.

What Do We Really Know about Diffie-Hellman?

If Eve can compute Discrete Log quickly then she can crack DH:

1. Eve sees g^a, g^b.
2. Eve computes Discrete Log to find a, b.
3. Eve computes $g^{ab} \pmod{p}$.

Known If Eve can crack DH then Eve can compute Discrete Log.
Not Known If Eve can crack DH then Eve can compute.

Hardness Assumption

Definition Let DHF be the following function:
Inputs p, g, g^a, g^b (note that a, b are not the input)

Outputs g^{ab}.

Obvious Theorem If Alice can crack Diffie-Hellman quickly then Alice can compute DHF quickly.

Hardness Assumption

Definition Let DHF be the following function:
Inputs p, g, g^a, g^b (note that a, b are not the input)

Outputs g^{ab}.

Obvious Theorem If Alice can crack Diffie-Hellman quickly then Alice can compute DHF quickly.
Hardness assumption DHF is hard to compute.

Hardness Assumption

Definition Let DHF be the following function:
Inputs p, g, g^a, g^b (note that a, b are not the input)

Outputs g^{ab}.

Obvious Theorem If Alice can crack Diffie-Hellman quickly then Alice can compute DHF quickly.
Hardness assumption DHF is hard to compute.
What is Believed

1. DHF is hard.
2. DHF is not equivalent to DL.

How Can Alice and Bob Use DH Key Exchange?

DH is Not a crypto System

1. Alice finds a (p, g), p of length L, g gen for $\mathbb{Z}_p^*$.

DH is Not a crypto System

1. Alice finds a (p, g), p of length L, g gen for $\mathbb{Z}_p^*$.
2. Alice sends (p, g) to Bob (Eve can see it).

DH is Not a crypto System

1. Alice finds a (p, g), p of length L, g gen for $\mathbb{Z}_p^*$.
2. Alice sends (p, g) to Bob (Eve can see it).
3. Alice picks **rand** a. Alice computes g^a and broadcasts it.

DH is Not a crypto System

1. Alice finds a (p, g), p of length L, g gen for $\mathbb{Z}_p^*$.
2. Alice sends (p, g) to Bob (Eve can see it).
3. Alice picks **rand** a. Alice computes g^a and broadcasts it.
4. Bob picks **rand** b. Bob computes g^b and broadcasts it.

DH is Not a crypto System

1. Alice finds a (p, g), p of length L, g gen for $\mathbb{Z}_p^*$.
2. Alice sends (p, g) to Bob (Eve can see it).
3. Alice picks **rand** a. Alice computes g^a and broadcasts it.
4. Bob picks **rand** b. Bob computes g^b and broadcasts it.
5. Alice computes $(g^b)^a = g^{ab} \pmod{p}$.

DH is Not a crypto System

1. Alice finds a (p, g), p of length L, g gen for $\mathbb{Z}_p^*$.
2. Alice sends (p, g) to Bob (Eve can see it).
3. Alice picks **rand** a. Alice computes g^a and broadcasts it.
4. Bob picks **rand** b. Bob computes g^b and broadcasts it.
5. Alice computes $(g^b)^a = g^{ab} \pmod{p}$.
6. Bob computes $(g^a)^b = g^{ab} \pmod{p}$.

DH is Not a crypto System

1. Alice finds a (p, g), p of length L, g gen for $\mathbb{Z}_p^*$.
2. Alice sends (p, g) to Bob (Eve can see it).
3. Alice picks **rand** a. Alice computes g^a and broadcasts it.
4. Bob picks **rand** b. Bob computes g^b and broadcasts it.
5. Alice computes $(g^b)^a = g^{ab} \pmod{p}$.
6. Bob computes $(g^a)^b = g^{ab} \pmod{p}$.
7. $s = g^{ab}$ is the shared secret.

DH is Not a crypto System

1. Alice finds a (p, g), p of length L, g gen for $\mathbb{Z}_p^*$.
2. Alice sends (p, g) to Bob (Eve can see it).
3. Alice picks **rand** a. Alice computes g^a and broadcasts it.
4. Bob picks **rand** b. Bob computes g^b and broadcasts it.
5. Alice computes $(g^b)^a = g^{ab} \pmod{p}$.
6. Bob computes $(g^a)^b = g^{ab} \pmod{p}$.
7. $s = g^{ab}$ is the shared secret.

At the end Alice and Bob have $\boldsymbol{s}$

DH is Not a crypto System

1. Alice finds a (p, g), p of length L, g gen for $\mathbb{Z}_p^*$.
2. Alice sends (p, g) to Bob (Eve can see it).
3. Alice picks **rand** a. Alice computes g^a and broadcasts it.
4. Bob picks **rand** b. Bob computes g^b and broadcasts it.
5. Alice computes $(g^b)^a = g^{ab} \pmod{p}$.
6. Bob computes $(g^a)^b = g^{ab} \pmod{p}$.
7. $s = g^{ab}$ is the shared secret.

At the end Alice and Bob have s but **s has no meaning!**.

DH is Not a crypto System

1. Alice finds a (p, g), p of length L, g gen for $\mathbb{Z}_p^*$.
2. Alice sends (p, g) to Bob (Eve can see it).
3. Alice picks **rand** a. Alice computes g^a and broadcasts it.
4. Bob picks **rand** b. Bob computes g^b and broadcasts it.
5. Alice computes $(g^b)^a = g^{ab} \pmod{p}$.
6. Bob computes $(g^a)^b = g^{ab} \pmod{p}$.
7. $s = g^{ab}$ is the shared secret.

At the end Alice and Bob have $\boldsymbol{s}$ but **$\boldsymbol{s}$ has no meaning!**.
s is not going to be **Bounded Queries in Recursion Theory.**

DH is Not a crypto System

1. Alice finds a (p, g), p of length L, g gen for $\mathbb{Z}_p^*$.
2. Alice sends (p, g) to Bob (Eve can see it).
3. Alice picks **rand** a. Alice computes g^a and broadcasts it.
4. Bob picks **rand** b. Bob computes g^b and broadcasts it.
5. Alice computes $(g^b)^a = g^{ab} \pmod{p}$.
6. Bob computes $(g^a)^b = g^{ab} \pmod{p}$.
7. $s = g^{ab}$ is the shared secret.

At the end Alice and Bob have s but **s has no meaning!**.
s is not going to be **Bounded Queries in Recursion Theory.**
s is going to be some random number in $\{1, \ldots, p-1\}$.

How can Alice and Bob Use s?

s is random.

How can Alice and Bob Use s?

s is random. No meaning.

How can Alice and Bob Use s?

s is random. No meaning. Darn.

How can Alice and Bob Use s?

s is random. No meaning. Darn.

When life gives you lemons, make lemonade.

How can Alice and Bob Use s?

s is random. No meaning. Darn.

When life gives you lemons, make lemonade.

When life gives you a random string, use a one-time pad.

How can Alice and Bob Use s?

s is random. No meaning. Darn.

When life gives you lemons, make lemonade.

When life gives you a random string, use a one-time pad.

1. Alice and Bob do DH and have shared string s.

How can Alice and Bob Use s?

s is random. No meaning. Darn.

When life gives you lemons, make lemonade.

When life gives you a random string, use a one-time pad.

1. Alice and Bob do DH and have shared string s.
2. Alice uses s as the key for a 1-time pad to tell Bob the name of the Book for Book Cipher.

How can Alice and Bob Use s?

s is random. No meaning. Darn.

When life gives you lemons, make lemonade.

When life gives you a random string, use a one-time pad.

1. Alice and Bob do DH and have shared string s.
2. Alice uses s as the key for a 1-time pad to tell Bob the name of the Book for Book Cipher.

This is not quite what people do but its the idea. Next slide is **El Gamal Public Key Crypto Systems** which is what people do.

Note really 1-Time Pad

Usual 1-Time Pad messages are bit strings. Use $\oplus$.

Note really 1-Time Pad

Usual 1-Time Pad messages are bit strings. Use $\oplus$.
In Next Protocol messages are elements of $\mathbb{Z}_p^*$. Use Mult Mod p.

ElGamal is DH Made Into an Enc System

ElGamal is DH Made Into an Enc System

1. Alice and Bob do Diffie Hellman.

ElGamal is DH Made Into an Enc System

1. Alice and Bob do Diffie Hellman.
2. Alice and Bob share secret $s = g^{ab} \pmod{p}$.

ElGamal is DH Made Into an Enc System

1. Alice and Bob do Diffie Hellman.
2. Alice and Bob share secret $s = g^{ab} \pmod{p}$.
3. Alice and Bob compute $s^{-1} \pmod{p}$.

ElGamal is DH Made Into an Enc System

1. Alice and Bob do Diffie Hellman.
2. Alice and Bob share secret $s = g^{ab} \pmod{p}$.
3. Alice and Bob compute $s^{-1} \pmod{p}$.
4. To send m, Alice sends $c = ms \pmod{p}$.

ElGamal is DH Made Into an Enc System

1. Alice and Bob do Diffie Hellman.
2. Alice and Bob share secret $s = g^{ab} \pmod{p}$.
3. Alice and Bob compute $s^{-1} \pmod{p}$.
4. To send m, Alice sends $c = ms \pmod{p}$.
5. To decrypt, Bob computes $cs^{-1} \equiv mss^{-1} \equiv m \pmod{p}$.

We omit discussion of Hardness assumption (HW)

Public Key Cryptography: RSA

Recall that DH is not a crypto-system

Diffie Hellman allowed Alice and Bob to share a secret string.

Recall that DH is not a crypto-system

Diffie Hellman allowed Alice and Bob to share a secret string.

Diffie Hellman *is not* an encryption system.

Recall that DH is not a crypto-system

Diffie Hellman allowed Alice and Bob to share a secret string.

Diffie Hellman *is not* an encryption system.

El Gamal *is* an encryption system but hard to use since its a 1-shot. You need to keep on doing DH to use it.

Recall that DH is not a crypto-system

Diffie Hellman allowed Alice and Bob to share a secret string.

Diffie Hellman *is not* an encryption system.

El Gamal *is* an encryption system but hard to use since its a 1-shot. You need to keep on doing DH to use it.

RSA is an encryption system.

Theorem for Primes, Theorem for n

We restate and generalize.

Theorem for Primes, Theorem for n

We restate and generalize.
Fermat's Little Theorem If p is prime and $a \not\equiv 0 \pmod{p}$ then

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

Theorem for Primes, Theorem for n

We restate and generalize.

Fermat's Little Theorem If p is prime and $a \not\equiv 0 \pmod{p}$ then

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

Restate:

Fermat's Little Theorem If p is prime and a is rel prime to p then

$$a^m \equiv a^{m \bmod \phi(p)} \pmod{p}.$$

Theorem for Primes, Theorem for n

We restate and generalize.

Fermat's Little Theorem If p is prime and $a \not\equiv 0 \pmod{p}$ then

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

Restate:

Fermat's Little Theorem If p is prime and a is rel prime to p then

$$a^m \equiv a^{m \bmod \phi(p)} \pmod{p}.$$

Generalize:

Fermat-Euler Theorem If $n \in \mathbb{N}$ and a is rel prime to n then

$$a^m \equiv a^{m \bmod \phi(n)} \pmod{n}.$$

Examples

$$14^{999,999} \pmod{393}$$

Examples

$$14^{999{,}999} \pmod{393}$$

$$\phi(393) = \phi(3 \times 131) = \phi(3) \times \phi(131) = 2 \times 130 = 260.$$

Examples

$$14^{999,999} \pmod{393}$$

$$\phi(393) = \phi(3 \times 131) = \phi(3) \times \phi(131) = 2 \times 130 = 260.$$

$$14^{999,999} = 14^{999,999 \pmod{260}} \pmod{393} \equiv 14^{39} \pmod{393}$$

Examples

$$14^{999,999} \pmod{393}$$

$$\phi(393) = \phi(3 \times 131) = \phi(3) \times \phi(131) = 2 \times 130 = 260.$$

$$14^{999,999} = 14^{999,999 \pmod{260}} \pmod{393} \equiv 14^{39} \pmod{393}$$

Now just do repeated squaring.

Easy and Hard

Easy or Hard?

Easy and Hard

Easy or Hard?

1. Given L, generate two primes of length L: p, q.

Easy and Hard

Easy or Hard?

1. Given L, generate two primes of length L: p, q. **Easy.**

Easy and Hard

Easy or Hard?

1. Given L, generate two primes of length L: p, q. **Easy.**
2. Given p, q find $N = pq$ and $R = \phi(N) = (p-1)(q-1)$.

Easy and Hard

Easy or Hard?

1. Given L, generate two primes of length L: p, q. **Easy.**
2. Given p, q find $N = pq$ and $R = \phi(N) = (p-1)(q-1)$. **Easy.**

Easy and Hard

Easy or Hard?

1. Given L, generate two primes of length L: p, q. **Easy.**
2. Given p, q find $N = pq$ and $R = \phi(N) = (p-1)(q-1)$. **Easy.**
3. Given R find an e rel prime to R. (e for encrypt.)

Easy and Hard

Easy or Hard?

1. Given L, generate two primes of length L: p, q. **Easy.**
2. Given p, q find $N = pq$ and $R = \phi(N) = (p-1)(q-1)$. **Easy.**
3. Given R find an e rel prime to R. (e for encrypt.) **Easy.**

Easy and Hard

Easy or Hard?

1. Given L, generate two primes of length L: p, q. **Easy.**
2. Given p, q find $N = pq$ and $R = \phi(N) = (p-1)(q-1)$. **Easy.**
3. Given R find an e rel prime to R. (e for encrypt.) **Easy.**
4. Given R, e find d such that $ed \equiv 1 \pmod{R}$.

Easy and Hard

Easy or Hard?

1. Given L, generate two primes of length L: p, q. **Easy.**
2. Given p, q find $N = pq$ and $R = \phi(N) = (p-1)(q-1)$. **Easy.**
3. Given R find an e rel prime to R. (e for encrypt.) **Easy.**
4. Given R, e find d such that $ed \equiv 1 \pmod{R}$. **Easy.**

Easy and Hard

Easy or Hard?

1. Given L, generate two primes of length L: p, q. **Easy.**
2. Given p, q find $N = pq$ and $R = \phi(N) = (p-1)(q-1)$. **Easy.**
3. Given R find an e rel prime to R. (e for encrypt.) **Easy.**
4. Given R, e find d such that $ed \equiv 1 \pmod{R}$. **Easy.**
5. Given N, e find d such that $ed \equiv 1 \pmod{R}$.

Easy and Hard

Easy or Hard?

1. Given L, generate two primes of length L: p, q. **Easy.**
2. Given p, q find $N = pq$ and $R = \phi(N) = (p-1)(q-1)$. **Easy.**
3. Given R find an e rel prime to R. (e for encrypt.) **Easy.**
4. Given R, e find d such that $ed \equiv 1 \pmod{R}$. **Easy.**
5. Given N, e find d such that $ed \equiv 1 \pmod{R}$. **Hard.**

Easy and Hard

Easy or Hard?

1. Given L, generate two primes of length L: p, q. **Easy.**
2. Given p, q find $N = pq$ and $R = \phi(N) = (p-1)(q-1)$. **Easy.**
3. Given R find an e rel prime to R. (e for encrypt.) **Easy.**
4. Given R, e find d such that $ed \equiv 1 \pmod{R}$. **Easy.**
5. Given N, e find d such that $ed \equiv 1 \pmod{R}$. **Hard.**
6. Compute $m^e \pmod{N}$.

Easy and Hard

Easy or Hard?

1. Given L, generate two primes of length L: p, q. **Easy.**
2. Given p, q find $N = pq$ and $R = \phi(N) = (p-1)(q-1)$. **Easy.**
3. Given R find an e rel prime to R. (e for encrypt.) **Easy.**
4. Given R, e find d such that $ed \equiv 1 \pmod{R}$. **Easy.**
5. Given N, e find d such that $ed \equiv 1 \pmod{R}$. **Hard.**
6. Compute $m^e \pmod{N}$. **Easy.**

RSA

Let L be a security parameter

RSA

Let L be a security parameter

1. **Alice** picks two primes p, q of length L and computes $N = pq$.

RSA

Let L be a security parameter

1. **Alice** picks two primes p, q of length L and computes $N = pq$.
2. **Alice** computes $R = \phi(N) = \phi(pq) = (p-1)(q-1)$.

RSA

Let L be a security parameter

1. **Alice** picks two primes p, q of length L and computes $N = pq$.
2. **Alice** computes $R = \phi(N) = \phi(pq) = (p-1)(q-1)$.
3. **Alice** picks an $e \in \{\frac{R}{3}, \ldots, \frac{2R}{3}\}$ that is relatively prime to R.

RSA

Let L be a security parameter

1. **Alice** picks two primes p, q of length L and computes $N = pq$.
2. **Alice** computes $R = \phi(N) = \phi(pq) = (p-1)(q-1)$.
3. **Alice** picks an $e \in \{\frac{R}{3}, \ldots, \frac{2R}{3}\}$ that is relatively prime to R.
4. **Alice** finds d such that $ed \equiv 1 \pmod{R}$.

RSA

Let L be a security parameter

1. **Alice** picks two primes p, q of length L and computes $N = pq$.
2. **Alice** computes $R = \phi(N) = \phi(pq) = (p-1)(q-1)$.
3. **Alice** picks an $e \in \{\frac{R}{3}, \ldots, \frac{2R}{3}\}$ that is relatively prime to R.
4. **Alice** finds d such that $ed \equiv 1 \pmod{R}$.
5. **Alice** broadcasts (N, e). (Bob and Eve both see it.)

RSA

Let L be a security parameter

1. **Alice** picks two primes p, q of length L and computes $N = pq$.
2. **Alice** computes $R = \phi(N) = \phi(pq) = (p-1)(q-1)$.
3. **Alice** picks an $e \in \{\frac{R}{3}, \ldots, \frac{2R}{3}\}$ that is relatively prime to R.
4. **Alice** finds d such that $ed \equiv 1 \pmod{R}$.
5. **Alice** broadcasts (N, e). (Bob and Eve both see it.)
6. **Bob** To send $m \in \{1, \ldots, N-1\}$, broadcast $m^e \pmod{N}$.

RSA

Let L be a security parameter

1. **Alice** picks two primes p, q of length L and computes $N = pq$.
2. **Alice** computes $R = \phi(N) = \phi(pq) = (p-1)(q-1)$.
3. **Alice** picks an $e \in \{\frac{R}{3}, \ldots, \frac{2R}{3}\}$ that is relatively prime to R.
4. **Alice** finds d such that $ed \equiv 1 \pmod{R}$.
5. **Alice** broadcasts (N, e). (Bob and Eve both see it.)
6. **Bob** To send $m \in \{1, \ldots, N-1\}$, broadcast $m^e \pmod{N}$.
7. If **Alice** gets $m^e \pmod{N}$ she computes

$$(m^e)^d \equiv m^{ed} \equiv m^{ed \bmod R} \equiv m^{1 \bmod R} \equiv m \pmod{N}.$$

RSA

Let L be a security parameter

1. **Alice** picks two primes p, q of length L and computes $N = pq$.
2. **Alice** computes $R = \phi(N) = \phi(pq) = (p-1)(q-1)$.
3. **Alice** picks an $e \in \{\frac{R}{3}, \ldots, \frac{2R}{3}\}$ that is relatively prime to R.
4. **Alice** finds d such that $ed \equiv 1 \pmod{R}$.
5. **Alice** broadcasts (N, e). (Bob and Eve both see it.)
6. **Bob** To send $m \in \{1, \ldots, N-1\}$, broadcast $m^e \pmod{N}$.
7. If **Alice** gets $m^e \pmod{N}$ she computes

$$(m^e)^d \equiv m^{ed} \equiv m^{ed \bmod R} \equiv m^{1 \bmod R} \equiv m \pmod{N}.$$

PRO Alice and Bob can execute the protocol easily.

RSA

Let L be a security parameter

1. **Alice** picks two primes p, q of length L and computes $N = pq$.
2. **Alice** computes $R = \phi(N) = \phi(pq) = (p-1)(q-1)$.
3. **Alice** picks an $e \in \{\frac{R}{3}, \ldots, \frac{2R}{3}\}$ that is relatively prime to R.
4. **Alice** finds d such that $ed \equiv 1 \pmod{R}$.
5. **Alice** broadcasts (N, e). (Bob and Eve both see it.)
6. **Bob** To send $m \in \{1, \ldots, N-1\}$, broadcast $m^e \pmod{N}$.
7. If **Alice** gets $m^e \pmod{N}$ she computes

$$(m^e)^d \equiv m^{ed} \equiv m^{ed \bmod R} \equiv m^{1 \bmod R} \equiv m \pmod{N}.$$

PRO Alice and Bob can execute the protocol easily.
Biggest PRO Alice and Bob never had to meet!

RSA

Let L be a security parameter

1. **Alice** picks two primes p, q of length L and computes $N = pq$.
2. **Alice** computes $R = \phi(N) = \phi(pq) = (p-1)(q-1)$.
3. **Alice** picks an $e \in \{\frac{R}{3}, \ldots, \frac{2R}{3}\}$ that is relatively prime to R.
4. **Alice** finds d such that $ed \equiv 1 \pmod{R}$.
5. **Alice** broadcasts (N, e). (Bob and Eve both see it.)
6. **Bob** To send $m \in \{1, \ldots, N-1\}$, broadcast $m^e \pmod{N}$.
7. If **Alice** gets $m^e \pmod{N}$ she computes

$$(m^e)^d \equiv m^{ed} \equiv m^{ed \bmod R} \equiv m^{1 \bmod R} \equiv m \pmod{N}.$$

PRO Alice and Bob can execute the protocol easily.
Biggest PRO Alice and Bob never had to meet!
PRO Bob can control the message.

RSA

Let L be a security parameter

1. **Alice** picks two primes p, q of length L and computes $N = pq$.
2. **Alice** computes $R = \phi(N) = \phi(pq) = (p-1)(q-1)$.
3. **Alice** picks an $e \in \{\frac{R}{3}, \ldots, \frac{2R}{3}\}$ that is relatively prime to R.
4. **Alice** finds d such that $ed \equiv 1 \pmod{R}$.
5. **Alice** broadcasts (N, e). (Bob and Eve both see it.)
6. **Bob** To send $m \in \{1, \ldots, N-1\}$, broadcast $m^e \pmod{N}$.
7. If **Alice** gets $m^e \pmod{N}$ she computes

$$(m^e)^d \equiv m^{ed} \equiv m^{ed \bmod R} \equiv m^{1 \bmod R} \equiv m \pmod{N}.$$

PRO Alice and Bob can execute the protocol easily.
Biggest PRO Alice and Bob never had to meet!
PRO Bob can control the message.
Question Can Eve find out m?

Convention for RSA

Alice sends (N, e) to get the process started.

Convention for RSA

Alice sends (N, e) to get the process started.

Then Bob can send Alice messages.

Convention for RSA

Alice sends (N, e) to get the process started.

Then Bob can send Alice messages.

We don't have Alice sending Bob messages.

Convention for RSA

Alice sends (N, e) to get the process started.

Then Bob can send Alice messages.

We don't have Alice sending Bob messages.

In examples we do in slides and HW we might not have $e \in \{\frac{R}{3}, \ldots, \frac{2R}{3}\}$ since we want to have easy computations for educational purposes.

What Do We Really Know about RSA

If Eve can factor then she can crack RSA.

What Do We Really Know about RSA

If Eve can factor then she can crack RSA.

1. Input (N, e) where $N = pq$ and e is rel prime to $R = (p-1)(q-1)$. $(p, q, R$ are NOT part of the input.)

What Do We Really Know about RSA

If Eve can factor then she can crack RSA.

1. Input (N, e) where $N = pq$ and e is rel prime to $R = (p-1)(q-1)$. (p, q, R are NOT part of the input.)
2. Eve factors N to find p, q. Eve computes $R = (p-1)(q-1)$.

What Do We Really Know about RSA

If Eve can factor then she can crack RSA.

1. Input (N, e) where $N = pq$ and e is rel prime to $R = (p-1)(q-1)$. (p, q, R are NOT part of the input.)
2. Eve factors N to find p, q. Eve computes $R = (p-1)(q-1)$.
3. Eve finds d such that $ed \equiv 1 \pmod{R}$.

Open If RSA is crackable then Factoring is Easy.

Hardness Assumption

Definition Let $RSAF$ be the following function:
Input $N, e, m^e \pmod{N}$ (know $N = pq$ but don't know p, q).

Outputs m.

Hardness Assumption

Definition Let $RSAF$ be the following function:
Input $N, e, m^e \pmod{N}$ (know $N = pq$ but don't know p, q).

Outputs m.

Hardness assumption (HA) $RSAF$ is hard to compute.

Hardness Assumption

Definition Let $RSAF$ be the following function:
Input $N, e, m^e \pmod{N}$ (know $N = pq$ but don't know p, q).

Outputs m.

Hardness assumption (HA) $RSAF$ is hard to compute.

One can show, assuming HA that RSA is hard to crack.
Believed RSA is uncrackable but not equiv to factoring.

Making RSA More Efficient

Use $e = 2^{2^4} + 1$. But ...

Use $e = 2^{2^4} + 1$. But ...

In Practice: Want to use $e = 2^{2^4} + 1$ since:

Use $e = 2^{2^4} + 1$. But ...

In Practice: Want to use $e = 2^{2^4} + 1$ since:

1. Only 15 mults. ($2^{2^4} + 1$ has very few 1's in it.)

Use $e = 2^{2^4} + 1$. But ...

In Practice: Want to use $e = 2^{2^4} + 1$ since:

1. Only 15 mults. ($2^{2^4} + 1$ has very few 1's in it.)
2. $2^{2^4} + 1$ Big enough to ward off the low-e attacks (we will study those later).

Use $e = 2^{2^4} + 1$. But ...

In Practice: Want to use $e = 2^{2^4} + 1$ since:

1. Only 15 mults. ($2^{2^4} + 1$ has very few 1's in it.)
2. $2^{2^4} + 1$ Big enough to ward off the low-e attacks (we will study those later).
3. $2^{2^4} + 1$ is prime, so only way it fails to be rel prime to $R = (p-1)(q-1)$. is if it divides R. Unlikely and easily tested.

Use $e = 2^{2^4} + 1$. But ...

In Practice: Want to use $e = 2^{2^4} + 1$ since:

1. Only 15 mults. ($2^{2^4} + 1$ has very few 1's in it.)
2. $2^{2^4} + 1$ Big enough to ward off the low-e attacks (we will study those later).
3. $2^{2^4} + 1$ is prime, so only way it fails to be rel prime to $R = (p-1)(q-1)$. is if it divides R. Unlikely and easily tested.

In Theory: Do not want to use **the same** e over and over again for fear of this being exploited.

Who is Right: $e = 2^{16} + 1$ is used a lot.

Use $e = 2^{2^4} + 1$. But ...

In Practice: Want to use $e = 2^{2^4} + 1$ since:

1. Only 15 mults. ($2^{2^4} + 1$ has very few 1's in it.)
2. $2^{2^4} + 1$ Big enough to ward off the low-e attacks (we will study those later).
3. $2^{2^4} + 1$ is prime, so only way it fails to be rel prime to $R = (p-1)(q-1)$. is if it divides R. Unlikely and easily tested.

In Theory: Do not want to use **the same** e over and over again for fear of this being exploited.

Who is Right: $e = 2^{16} + 1$ is used a lot. Should it be?

Use $e = 2^{2^4} + 1$. But ...

In Practice: Want to use $e = 2^{2^4} + 1$ since:

1. Only 15 mults. ($2^{2^4} + 1$ has very few 1's in it.)
2. $2^{2^4} + 1$ Big enough to ward off the low-e attacks (we will study those later).
3. $2^{2^4} + 1$ is prime, so only way it fails to be rel prime to $R = (p-1)(q-1)$. is if it divides R. Unlikely and easily tested.

In Theory: Do not want to use **the same** e over and over again for fear of this being exploited.

Who is Right: $e = 2^{16} + 1$ is used a lot. Should it be?

- Nobody in academia has cracked RSA just using that $e = 2^{2^4} - 1$.

Use $e = 2^{2^4} + 1$. But ...

In Practice: Want to use $e = 2^{2^4} + 1$ since:

1. Only 15 mults. ($2^{2^4} + 1$ has very few 1's in it.)
2. $2^{2^4} + 1$ Big enough to ward off the low-e attacks (we will study those later).
3. $2^{2^4} + 1$ is prime, so only way it fails to be rel prime to $R = (p-1)(q-1)$. is if it divides R. Unlikely and easily tested.

In Theory: Do not want to use **the same** e over and over again for fear of this being exploited.

Who is Right: $e = 2^{16} + 1$ is used a lot. Should it be?

- Nobody in academia has cracked RSA just using that $e = 2^{2^4} - 1$.
- Nobody in the real world has cracked RSA just using that $e = 2^{2^4} - 1$.

Use $e = 2^{2^4} + 1$. But ...

In Practice: Want to use $e = 2^{2^4} + 1$ since:

1. Only 15 mults. ($2^{2^4} + 1$ has very few 1's in it.)
2. $2^{2^4} + 1$ Big enough to ward off the low-e attacks (we will study those later).
3. $2^{2^4} + 1$ is prime, so only way it fails to be rel prime to $R = (p-1)(q-1)$. is if it divides R. Unlikely and easily tested.

In Theory: Do not want to use **the same** e over and over again for fear of this being exploited.

Who is Right: $e = 2^{16} + 1$ is used a lot. Should it be?

- Nobody in academia has cracked RSA just using that $e = 2^{2^4} - 1$.
- Nobody in the real world has cracked RSA just using that $e = 2^{2^4} - 1$.
- Do we really know that?

RSA has NY,NY Problem. Will Fix

Plain RSA Bytes!

Scenario

Eve sees Bob send Alice c_1 (message is m_1).

Plain RSA Bytes!

Scenario

Eve sees Bob send Alice c_1 (message is m_1).
Later Eve sees Bob send Alice c_2 (message is m_2).

Plain RSA Bytes!

Scenario
Eve sees Bob send Alice c_1 (message is m_1).
Later Eve sees Bob send Alice c_2 (message is m_2).

What can Eve **easily** deduce?

Plain RSA Bytes!

Scenario

Eve sees Bob send Alice c_1 (message is m_1).
Later Eve sees Bob send Alice c_2 (message is m_2).

What can Eve **easily** deduce?

Eve can know if $c_1 = c_2$ or not. So what?

Plain RSA Bytes!

Scenario
Eve sees Bob send Alice c_1 (message is m_1).
Later Eve sees Bob send Alice c_2 (message is m_2).

What can Eve **easily** deduce?

Eve can know if $c_1 = c_2$ or not. So what?
Eve knows if $m_1 = m_2$ or not. Its the NY,NY problem!

Plain RSA Bytes!

Scenario
Eve sees Bob send Alice c_1 (message is m_1).
Later Eve sees Bob send Alice c_2 (message is m_2).

What can Eve **easily** deduce?

Eve can know if $c_1 = c_2$ or not. So what?
Eve knows if $m_1 = m_2$ or not. Its the NY,NY problem!

That alone makes it insecure.

Plain RSA Bytes!

Scenario
Eve sees Bob send Alice c_1 (message is m_1).
Later Eve sees Bob send Alice c_2 (message is m_2).

What can Eve **easily** deduce?

Eve can know if $c_1 = c_2$ or not. So what?
Eve knows if $m_1 = m_2$ or not. Its the NY,NY problem!

That alone makes it insecure.
Plain RSA is never used and should never be used!

PKCS-1.5 RSA

We need to change how Bob sends a message;
BAD To send $m \in \{1, \ldots, N-1\}$, send $m^e \pmod{N}$.

PKCS-1.5 RSA

We need to change how Bob sends a message;
BAD To send $m \in \{1, \ldots, N-1\}$, send $m^e \pmod{N}$.

FIX To send $m \in \{1, \ldots, N-1\}$, pick rand r, send $(rm)^e$. (NOTE- rm means r CONCAT with m here and elsewhere.) Alice and Bob agree on **length** of r ahead of time.

PKCS-1.5 RSA

We need to change how Bob sends a message;
BAD To send $m \in \{1, \ldots, N-1\}$, send $m^e \pmod{N}$.

FIX To send $m \in \{1, \ldots, N-1\}$, pick rand r, send $(rm)^e$.
(NOTE- rm means r CONCAT with m here and elsewhere.) Alice and Bob agree on **length** of r ahead of time.

Alice and Bob pick L_1 and L_2 such that $\lg N = L_1 + L_2$.
To send $m \in \{0,1\}^{L_2}$ pick random $r \in \{0,1\}^{L_1}$.
When Alice gets rm she will know that m is the last L_2 bits.

RSA Misc

RSA has other Problems

An encryption system is **malleable** if when Eve sees a message she can figure out a way to send a similar one, where she knows the similarity (she still does not know the message).

RSA has other Problems

An encryption system is **malleable** if when Eve sees a message she can figure out a way to send a similar one, where she knows the similarity (she still does not know the message).

1. The definition above is informal.

RSA has other Problems

An encryption system is **malleable** if when Eve sees a message she can figure out a way to send a similar one, where she knows the similarity (she still does not know the message).

1. The definition above is informal.
2. Can modify RSA so that it's probably not malleable.

RSA has other Problems

An encryption system is **malleable** if when Eve sees a message she can figure out a way to send a similar one, where she knows the similarity (she still does not know the message).

1. The definition above is informal.
2. Can modify RSA so that it's probably not malleable.
3. That way is called PKCS-2.0-RSA.

RSA has other Problems

An encryption system is **malleable** if when Eve sees a message she can figure out a way to send a similar one, where she knows the similarity (she still does not know the message).

1. The definition above is informal.
2. Can modify RSA so that it's probably not malleable.
3. That way is called PKCS-2.0-RSA.
4. Name BLAH-1.5 is hint that it's not final version.

RSA has other Problems

An encryption system is **malleable** if when Eve sees a message she can figure out a way to send a similar one, where she knows the similarity (she still does not know the message).

1. The definition above is informal.
2. Can modify RSA so that it's probably not malleable.
3. That way is called PKCS-2.0-RSA.
4. Name BLAH-1.5 is hint that it's not final version.
5. There are other issues that RSA needs to deal with and does, so the real RSA that is used adds more to what I've said here.

Rabin's Encryption System and its Variants

Rabin's Encryption System and its Variants

1. Rabin's enc equivalent to factoring pq.

Rabin's Encryption System and its Variants

1. Rabin's enc equivalent to factoring pq.
2. Rabin's enc is hard to use: messages do not decode uniquely.

Rabin's Encryption System and its Variants

1. Rabin's enc equivalent to factoring pq.
2. Rabin's enc is hard to use: messages do not decode uniquely.
3. Blum-Williams modified Rabin's Enc so that messages decode uniquely; but the set of messages you can send is small.

Rabin's Encryption System and its Variants

1. Rabin's enc equivalent to factoring pq.
2. Rabin's enc is hard to use: messages do not decode uniquely.
3. Blum-Williams modified Rabin's Enc so that messages decode uniquely; but the set of messages you can send is small.
4. Hard to combine Blum-Williams modification with the padding needed to solve NY,NY problem.

Rabin's Encryption System and its Variants

1. Rabin's enc equivalent to factoring pq.
2. Rabin's enc is hard to use: messages do not decode uniquely.
3. Blum-Williams modified Rabin's Enc so that messages decode uniquely; but the set of messages you can send is small.
4. Hard to combine Blum-Williams modification with the padding needed to solve NY,NY problem.
5. Cracking Rabin Enc EQUIV factoring: but this is only if Eve has no other information.

Rabin's Encryption System and its Variants

1. Rabin's enc equivalent to factoring pq.
2. Rabin's enc is hard to use: messages do not decode uniquely.
3. Blum-Williams modified Rabin's Enc so that messages decode uniquely; but the set of messages you can send is small.
4. Hard to combine Blum-Williams modification with the padding needed to solve NY,NY problem.
5. Cracking Rabin Enc EQUIV factoring: but this is only if Eve has no other information.
6. If Eve can trick Alice into sending a chosen message, she can crack Rabin. So **Chosen Plaintext Attack**-insecure.

Summary of RSA

Summary of RSA

1. PKCS-2.0-RSA is REALLY used!

Summary of RSA

1. PKCS-2.0-RSA is REALLY used!
2. There are many variants of RSA but all use the ideas above.

Summary of RSA

1. PKCS-2.0-RSA is REALLY used!
2. There are many variants of RSA but all use the ideas above.
3. Factoring easy implies RSA crackable. TRUE.

Summary of RSA

1. PKCS-2.0-RSA is REALLY used!
2. There are many variants of RSA but all use the ideas above.
3. Factoring easy implies RSA crackable. TRUE.
4. RSA crackable implies Factoring easy: UNKNOWN.

Summary of RSA

1. PKCS-2.0-RSA is REALLY used!
2. There are many variants of RSA but all use the ideas above.
3. Factoring easy implies RSA crackable. TRUE.
4. RSA crackable implies Factoring easy: UNKNOWN.
5. RSA crackable implies Factoring easy: Often stated in expositions of crypto. They are wrong!

Public Key Not Based on Factoring

What if Factoring can be done fast (quantum, fancy number theory, better hardware)?

Public Key Not Based on Factoring

What if Factoring can be done fast (quantum, fancy number theory, better hardware)?

1. Since 1960:

Public Key Not Based on Factoring

What if Factoring can be done fast (quantum, fancy number theory, better hardware)?

1. Since 1960:
   1.1 Math-advances have sped up factoring by 1000 times.

Public Key Not Based on Factoring

What if Factoring can be done fast (quantum, fancy number theory, better hardware)?

1. Since 1960:
   1.1 Math-advances have sped up factoring by 1000 times.
   1.2 Hardware-advances have sped up factoring by 1000 times.

Public Key Not Based on Factoring

What if Factoring can be done fast (quantum, fancy number theory, better hardware)?

1. Since 1960:
    1.1 Math-advances have sped up factoring by 1000 times.
    1.2 Hardware-advances have sped up factoring by 1000 times.
    1.3 So Factoring has been sped up 1,000,000 times.

Public Key Not Based on Factoring

What if Factoring can be done fast (quantum, fancy number theory, better hardware)?

1. Since 1960:
   1.1 Math-advances have sped up factoring by 1000 times.
   1.2 Hardware-advances have sped up factoring by 1000 times.
   1.3 So Factoring has been sped up 1,000,000 times.
2. Factoring is in Quantum P, though making that practical seems a ways off.

Public Key Not Based on Factoring

What if Factoring can be done fast (quantum, fancy number theory, better hardware)?

1. Since 1960:
   1.1 Math-advances have sped up factoring by 1000 times.
   1.2 Hardware-advances have sped up factoring by 1000 times.
   1.3 So Factoring has been sped up 1,000,000 times.
2. Factoring is in Quantum P, though making that practical seems a ways off.
3. There are now several Public Key Systems based on **other** hardness assumptions. They are not used yet as they need to be tested. Chicken-and-Egg Problem.

BILL, STOP RECORDING LECTURE!!!!

BILL STOP RECORDING LECTURE!!!