BILL, RECORD LECTURE!!!!

BILL RECORD LECTURE!!!

Public Key Crypto: Math Needed and Diffie-Hellman

Private-Key Ciphers

What do the following all have in common?

1. Shift Cipher
2. Affine Cipher
3. Vig Cipher
4. General Sub
5. General 2-char sub
6. Matrix Cipher
7. One-time Pad
8. Other ciphers we studied

Private-Key Ciphers

What do the following all have in common?

1. Shift Cipher
2. Affine Cipher
3. Vig Cipher
4. General Sub
5. General 2-char sub
6. Matrix Cipher
7. One-time Pad
8. Other ciphers we studied

Alice and Bob need to **meet!** (Hence **Private-Key.**)

Private-Key Ciphers

What do the following all have in common?

1. Shift Cipher
2. Affine Cipher
3. Vig Cipher
4. General Sub
5. General 2-char sub
6. Matrix Cipher
7. One-time Pad
8. Other ciphers we studied

Alice and Bob need to **meet!** (Hence **Private-Key.**)
Can Alice and Bob establish a key without meeting?

Private-Key Ciphers

What do the following all have in common?

1. Shift Cipher
2. Affine Cipher
3. Vig Cipher
4. General Sub
5. General 2-char sub
6. Matrix Cipher
7. One-time Pad
8. Other ciphers we studied

Alice and Bob need to **meet!** (Hence **Private-Key.**)
Can Alice and Bob establish a key without meeting?
Yes! And that is the **key** to public-**key** cryptography.

General Philosophy

A good crypto system is such that:

1. The computational task to **encrypt** and **decrypt** is **easy.**
2. The computational task to **crack** is **hard.**

General Philosophy

A good crypto system is such that:

1. The computational task to **encrypt** and **decrypt** is **easy.**
2. The computational task to **crack** is **hard.**

Caveats

General Philosophy

A good crypto system is such that:

1. The computational task to **encrypt** and **decrypt** is **easy.**
2. The computational task to **crack** is **hard.**

Caveats

1. Hard to achieve info-theoretic hardness (One-time pad).

General Philosophy

A good crypto system is such that:

1. The computational task to **encrypt** and **decrypt** is **easy.**
2. The computational task to **crack** is **hard.**

Caveats

1. Hard to achieve info-theoretic hardness (One-time pad).
2. Hard to achieve comp-hardness. Few problems provably hard.

General Philosophy

A good crypto system is such that:

1. The computational task to **encrypt** and **decrypt** is **easy.**
2. The computational task to **crack** is **hard.**

Caveats

1. Hard to achieve info-theoretic hardness (One-time pad).
2. Hard to achieve comp-hardness. Few problems provably hard.
3. Can use hardness assumptions (e.g. factoring is hard).

Difficulty of Problems Based on Length of Input

Hardness of a problem is measured by time-to-solve as a function of **length of input**.

Difficulty of Problems Based on Length of Input

Hardness of a problem is measured by time-to-solve as a function of **length of input**.

Examples

Difficulty of Problems Based on Length of Input

Hardness of a problem is measured by time-to-solve as a function of **length of input**.

Examples

1. Given a Boolean formula $\phi(x_1, \ldots, x_n)$, is there a satisfying assignment? Seems to require $2^{\Omega(n)}$ steps.

Difficulty of Problems Based on Length of Input

Hardness of a problem is measured by time-to-solve as a function of **length of input**.

Examples

1. Given a Boolean formula $\phi(x_1, \ldots, x_n)$, is there a satisfying assignment? Seems to require $2^{\Omega(n)}$ steps.
2. Polynomial vs Exp time is our notion of easy vs hard.

Difficulty of Problems Based on Length of Input

Hardness of a problem is measured by time-to-solve as a function of **length of input**.

Examples

1. Given a Boolean formula $\phi(x_1, \ldots, x_n)$, is there a satisfying assignment? Seems to require $2^{\Omega(n)}$ steps.
2. Polynomial vs Exp time is our notion of easy vs hard.
3. Factoring n can be done in $O(\sqrt{n})$ time: **Discuss**. Easy!

Difficulty of Problems Based on Length of Input

Hardness of a problem is measured by time-to-solve as a function of **length of input**.

Examples

1. Given a Boolean formula $\phi(x_1, \ldots, x_n)$, is there a satisfying assignment? Seems to require $2^{\Omega(n)}$ steps.
2. Polynomial vs Exp time is our notion of easy vs hard.
3. Factoring n can be done in $O(\sqrt{n})$ time: **Discuss**. Easy! **NO!!**: n is of **length** $\lg n + O(1)$ (henceforth just $\lg n$). $\sqrt{n} = 2^{(0.5)\lg n}$. Exponential. Better (but still exp) algs known.

Difficulty of Problems Based on Length of Input

Hardness of a problem is measured by time-to-solve as a function of **length of input**.

Examples

1. Given a Boolean formula $\phi(x_1, \ldots, x_n)$, is there a satisfying assignment? Seems to require $2^{\Omega(n)}$ steps.
2. Polynomial vs Exp time is our notion of easy vs hard.
3. Factoring n can be done in $O(\sqrt{n})$ time: **Discuss**. Easy! **NO!!**: n is of **length** $\lg n + O(1)$ (henceforth just $\lg n$). $\sqrt{n} = 2^{(0.5)\lg n}$. Exponential. Better (but still exp) algs known.

Upshot For numeric problems length is $\lg n$. Encryption requires:

- Alice and Bob can Enc and Dec in time $\le (\log n)^{O(1)}$.
- Eve needs time $\ge c^{O(\log n)}$ to crack.

Difficulty of Problems Based on Length of Input

Hardness of a problem is measured by time-to-solve as a function of **length of input**.

Examples

1. Given a Boolean formula $\phi(x_1, \ldots, x_n)$, is there a satisfying assignment? Seems to require $2^{\Omega(n)}$ steps.
2. Polynomial vs Exp time is our notion of easy vs hard.
3. Factoring n can be done in $O(\sqrt{n})$ time: **Discuss**. Easy! **NO!!**: n is of **length** $\lg n + O(1)$ (henceforth just $\lg n$). $\sqrt{n} = 2^{(0.5)\lg n}$. Exponential. Better (but still exp) algs known.

Upshot For numeric problems length is **$\lg n$.** Encryption requires:

- Alice and Bob can Enc and Dec in time $\le (\log n)^{O(1)}$.
- Eve needs time $\ge c^{O(\log n)}$ to crack.

What Counts We count math operations as taking 1 step. This could be an issue with enormous numbers. We will work with mods so not a problem.

Math Needed for Both Diffie-Hellman and RSA

Notation

Let p be a prime.

1. $\mathbb{Z}_p$ is the numbers $\{0, \ldots, p-1\}$ with mod add and mult.
2. $\mathbb{Z}_p^*$ is the numbers $\{1, \ldots, p-1\}$ with mod mult.

Convention By **prime** we will always mean a large prime, so in particular, NOT 2. Hence we can assume $\frac{p-1}{2}$ is in $\mathbb{N}$.

Exponentiation Mod p

Exponentiation Mod p

Problem Given a, n, p find $a^n \pmod p$

Exponentiation Mod p

Problem Given a, n, p find $a^n \pmod{p}$

Even though we use p and p is always prime, our algorithm works for any natural p.

Exponentiation Mod p: First Attempt

Problem Given a, n, p find $a^n \pmod p$

1. $x_0 = a^0 = 1$
2. For $i = 1$ to n, $x_i = ax_{i-1} \pmod p$
3. Let $x = x_n$
4. Output x

Is this a good idea?

Exponentiation Mod p: First Attempt

Problem Given a, n, p find $a^n \pmod{p}$

1. $x_0 = a^0 = 1$
2. For $i = 1$ to n, $x_i = ax_{i-1} \pmod{p}$
3. Let $x = x_n$
4. Output x

Is this a good idea? I called it **First Attempt,** so no.

Exponentiation Mod p: First Attempt

Problem Given a, n, p find $a^n \pmod{p}$

1. $x_0 = a^0 = 1$
2. For $i = 1$ to n, $x_i = ax_{i-1} \pmod{p}$
3. Let $x = x_n$
4. Output x

Is this a good idea? I called it **First Attempt,** so no.
Discuss How many steps were used to compute $a^n \pmod{p}$?

Exponentiation Mod p: First Attempt

Problem Given a, n, p find $a^n \pmod{p}$

1. $x_0 = a^0 = 1$
2. For $i = 1$ to n, $x_i = ax_{i-1} \pmod{p}$
3. Let $x = x_n$
4. Output x

Is this a good idea? I called it **First Attempt,** so no.
Discuss How many steps were used to compute $a^n \pmod{p}$?
Answer $\sim n$.

Exponentiation Mod p: Example of a Good Alg

Want $3^{64} \pmod{101}$. All math is mod 101.

Exponentiation Mod p: Example of a Good Alg

Want $3^{64} \pmod{101}$. All math is mod 101.

$x_0 = 3$

Exponentiation Mod p: Example of a Good Alg

Want $3^{64} \pmod{101}$. All math is mod 101.

$x_0 = 3$

$x_1 = x_0^2 \equiv 9$. This is $3^2 \pmod{101}$.

Exponentiation Mod p: Example of a Good Alg

Want $3^{64} \pmod{101}$. All math is mod 101.

$x_0 = 3$

$x_1 = x_0^2 \equiv 9$. This is $3^2 \pmod{101}$.

$x_2 = x_1^2 \equiv 9^2 \equiv 81$. This is $3^4 \pmod{101}$.

Exponentiation Mod p: Example of a Good Alg

Want $3^{64} \pmod{101}$. All math is mod 101.

$x_0 = 3$

$x_1 = x_0^2 \equiv 9$. This is $3^2 \pmod{101}$.

$x_2 = x_1^2 \equiv 9^2 \equiv 81$. This is $3^4 \pmod{101}$.

$x_3 = x_2^2 \equiv 81^2 \equiv 97$. This is $3^8 \pmod{101}$.

Exponentiation Mod p: Example of a Good Alg

Want $3^{64} \pmod{101}$. All math is mod 101.

$x_0 = 3$

$x_1 = x_0^2 \equiv 9$. This is $3^2 \pmod{101}$.

$x_2 = x_1^2 \equiv 9^2 \equiv 81$. This is $3^4 \pmod{101}$.

$x_3 = x_2^2 \equiv 81^2 \equiv 97$. This is $3^8 \pmod{101}$.

$x_4 = x_3^2 \equiv 97^2 \equiv 16$. This is $3^{16} \pmod{101}$.

Exponentiation Mod p: Example of a Good Alg

Want $3^{64} \pmod{101}$. All math is mod 101.

$x_0 = 3$

$x_1 = x_0^2 \equiv 9$. This is $3^2 \pmod{101}$.

$x_2 = x_1^2 \equiv 9^2 \equiv 81$. This is $3^4 \pmod{101}$.

$x_3 = x_2^2 \equiv 81^2 \equiv 97$. This is $3^8 \pmod{101}$.

$x_4 = x_3^2 \equiv 97^2 \equiv 16$. This is $3^{16} \pmod{101}$.

$x_5 = x_4^2 \equiv 16^2 \equiv 54$. This is $3^{32} \pmod{101}$.

Exponentiation Mod p: Example of a Good Alg

Want $3^{64} \pmod{101}$. All math is mod 101.

$x_0 = 3$

$x_1 = x_0^2 \equiv 9$. This is $3^2 \pmod{101}$.

$x_2 = x_1^2 \equiv 9^2 \equiv 81$. This is $3^4 \pmod{101}$.

$x_3 = x_2^2 \equiv 81^2 \equiv 97$. This is $3^8 \pmod{101}$.

$x_4 = x_3^2 \equiv 97^2 \equiv 16$. This is $3^{16} \pmod{101}$.

$x_5 = x_4^2 \equiv 16^2 \equiv 54$. This is $3^{32} \pmod{101}$.

$x_6 = x_5^2 \equiv 54^2 \equiv 88$. This is $3^{64} \pmod{101}$.

So in 6 steps we got the answer!

Exponentiation Mod p: Example of a Good Alg

Want $3^{64} \pmod{101}$. All math is mod 101.

$x_0 = 3$

$x_1 = x_0^2 \equiv 9$. This is $3^2 \pmod{101}$.

$x_2 = x_1^2 \equiv 9^2 \equiv 81$. This is $3^4 \pmod{101}$.

$x_3 = x_2^2 \equiv 81^2 \equiv 97$. This is $3^8 \pmod{101}$.

$x_4 = x_3^2 \equiv 97^2 \equiv 16$. This is $3^{16} \pmod{101}$.

$x_5 = x_4^2 \equiv 16^2 \equiv 54$. This is $3^{32} \pmod{101}$.

$x_6 = x_5^2 \equiv 54^2 \equiv 88$. This is $3^{64} \pmod{101}$.

So in 6 steps we got the answer!

Discuss How many steps are used to compute $a^n \pmod{p}$?

Exponentiation Mod p: Example of a Good Alg

Want $3^{64} \pmod{101}$. All math is mod 101.

$x_0 = 3$

$x_1 = x_0^2 \equiv 9$. This is $3^2 \pmod{101}$.

$x_2 = x_1^2 \equiv 9^2 \equiv 81$. This is $3^4 \pmod{101}$.

$x_3 = x_2^2 \equiv 81^2 \equiv 97$. This is $3^8 \pmod{101}$.

$x_4 = x_3^2 \equiv 97^2 \equiv 16$. This is $3^{16} \pmod{101}$.

$x_5 = x_4^2 \equiv 16^2 \equiv 54$. This is $3^{32} \pmod{101}$.

$x_6 = x_5^2 \equiv 54^2 \equiv 88$. This is $3^{64} \pmod{101}$.

So in 6 steps we got the answer!

Discuss How many steps are used to compute $a^n \pmod{p}$?

$\sim \lg n$.

Exponentiation Mod p: Example of a Good Alg

Want $3^{64} \pmod{101}$. All math is mod 101.

$x_0 = 3$

$x_1 = x_0^2 \equiv 9$. This is $3^2 \pmod{101}$.

$x_2 = x_1^2 \equiv 9^2 \equiv 81$. This is $3^4 \pmod{101}$.

$x_3 = x_2^2 \equiv 81^2 \equiv 97$. This is $3^8 \pmod{101}$.

$x_4 = x_3^2 \equiv 97^2 \equiv 16$. This is $3^{16} \pmod{101}$.

$x_5 = x_4^2 \equiv 16^2 \equiv 54$. This is $3^{32} \pmod{101}$.

$x_6 = x_5^2 \equiv 54^2 \equiv 88$. This is $3^{64} \pmod{101}$.

So in 6 steps we got the answer!

Discuss How many steps are used to compute $a^n \pmod{p}$?

$\sim \lg n$.

But the above algorithm only seems to work if n is a power of 2.

Exponentiation Mod p: Example of a Good Alg

Want $3^{64} \pmod{101}$. All math is mod 101.

$x_0 = 3$

$x_1 = x_0^2 \equiv 9$. This is $3^2 \pmod{101}$.

$x_2 = x_1^2 \equiv 9^2 \equiv 81$. This is $3^4 \pmod{101}$.

$x_3 = x_2^2 \equiv 81^2 \equiv 97$. This is $3^8 \pmod{101}$.

$x_4 = x_3^2 \equiv 97^2 \equiv 16$. This is $3^{16} \pmod{101}$.

$x_5 = x_4^2 \equiv 16^2 \equiv 54$. This is $3^{32} \pmod{101}$.

$x_6 = x_5^2 \equiv 54^2 \equiv 88$. This is $3^{64} \pmod{101}$.

So in 6 steps we got the answer!

Discuss How many steps are used to compute $a^n \pmod{p}$?

$\sim \lg n$.

But the above algorithm only seems to work if n is a power of 2.

Discuss What if n is **not a power of 2?**

A Review of Base 2

Say we want to do $a^n \pmod{p}$.

A Review of Base 2

Say we want to do $a^n \pmod{p}$.
Express n in binary.

A Review of Base 2

Say we want to do $a^n \pmod{p}$.
Express n in binary.
$7 = (111)_2 = 1 \times 2^{\mathbf{2}} + 1 \times 2^1 + 1 \times 2^0$. Note $\mathbf{2} = \lfloor \lg 7 \rfloor$

A Review of Base 2

Say we want to do $a^n \pmod{p}$.
Express n in binary.
$7 = (111)_2 = 1 \times 2^{\mathbf{2}} + 1 \times 2^1 + 1 \times 2^0$. Note $\mathbf{2} = \lfloor \lg 7 \rfloor$
$8 = (1000)_2 = 1 \times 2^{\mathbf{3}} + 0 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$. Note $\mathbf{3} = \lfloor \lg 8 \rfloor$

A Review of Base 2

Say we want to do $a^n \pmod{p}$.
Express n in binary.
$7 = (111)_2 = 1 \times 2^{\mathbf{2}} + 1 \times 2^1 + 1 \times 2^0$. Note $\mathbf{2} = \lfloor \lg 7 \rfloor$
$8 = (1000)_2 = 1 \times 2^{\mathbf{3}} + 0 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$. Note $\mathbf{3} = \lfloor \lg 8 \rfloor$
$9 = (1001)_2 = 1 \times 2^{\mathbf{3}} + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$. Note $\mathbf{3} = \lfloor \lg 9 \rfloor$

A Review of Base 2

Say we want to do $a^n \pmod{p}$.
Express n in binary.
$7 = (111)_2 = 1 \times 2^{\mathbf{2}} + 1 \times 2^1 + 1 \times 2^0$. Note $\mathbf{2} = \lfloor \lg 7 \rfloor$
$8 = (1000)_2 = 1 \times 2^{\mathbf{3}} + 0 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$. Note $\mathbf{3} = \lfloor \lg 8 \rfloor$
$9 = (1001)_2 = 1 \times 2^{\mathbf{3}} + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$. Note $\mathbf{3} = \lfloor \lg 9 \rfloor$
Upshot If write n as a sum of powers of 2 with 0,1 coefficients then n is of the form

A Review of Base 2

Say we want to do $a^n \pmod{p}$.
Express n in binary.
$7 = (111)_2 = 1 \times 2^{\mathbf{2}} + 1 \times 2^1 + 1 \times 2^0$. Note $\mathbf{2} = \lfloor \lg 7 \rfloor$
$8 = (1000)_2 = 1 \times 2^{\mathbf{3}} + 0 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$. Note $\mathbf{3} = \lfloor \lg 8 \rfloor$
$9 = (1001)_2 = 1 \times 2^{\mathbf{3}} + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$. Note $\mathbf{3} = \lfloor \lg 9 \rfloor$
Upshot If write n as a sum of powers of 2 with 0,1 coefficients then n is of the form

$$n = n_L 2^L + \cdots + n_1 2^1 + n_0 2^0 = \sum_{i=0}^{L} n_i 2^i$$

Where $L = \lfloor \lg(n) \rfloor$ and $n_i \in \{0, 1\}$.

A Review of Base 2

Say we want to do $a^n \pmod{p}$.
Express n in binary.
$7 = (111)_2 = 1 \times 2^{\mathbf{2}} + 1 \times 2^1 + 1 \times 2^0$. Note $\mathbf{2} = \lfloor \lg 7 \rfloor$
$8 = (1000)_2 = 1 \times 2^{\mathbf{3}} + 0 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$. Note $\mathbf{3} = \lfloor \lg 8 \rfloor$
$9 = (1001)_2 = 1 \times 2^{\mathbf{3}} + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$. Note $\mathbf{3} = \lfloor \lg 9 \rfloor$
Upshot If write n as a sum of powers of 2 with 0,1 coefficients then n is of the form

$$n = n_L 2^L + \cdots + n_1 2^1 + n_0 2^0 = \sum_{i=0}^{L} n_i 2^i$$

Where $L = \lfloor \lg(n) \rfloor$ and $n_i \in \{0, 1\}$.
Note that L is one less than the number of bits needed for n.

Repeated Squaring Algorithm

All math is mod p.

Repeated Squaring Algorithm

All math is mod p.

1. Input (a, n, p).

Repeated Squaring Algorithm

All math is mod p.

1. Input (a, n, p).
2. Convert n to base 2: $n = \sum_{i=0}^{L} n_i 2^i$. ($L$ is $\lfloor \lg(n) \rfloor$)

Repeated Squaring Algorithm

All math is mod p.

1. Input (a, n, p).
2. Convert n to base 2: $n = \sum_{i=0}^{L} n_i 2^i$. ($L$ is $\lfloor \lg(n) \rfloor$)
3. $x_0 = a$.

Repeated Squaring Algorithm

All math is mod p.

1. Input (a, n, p).
2. Convert n to base 2: $n = \sum_{i=0}^{L} n_i 2^i$. ($L$ is $\lfloor \lg(n) \rfloor$)
3. $x_0 = a$.
4. For $i = 1$ to L, $x_i = x_{i-1}^2$

Repeated Squaring Algorithm

All math is mod p.

1. Input (a, n, p).
2. Convert n to base 2: $n = \sum_{i=0}^{L} n_i 2^i$. ($L$ is $\lfloor \lg(n) \rfloor$)
3. $x_0 = a$.
4. For $i = 1$ to L, $x_i = x_{i-1}^2$
5. (Now have $a^{n_0 2^0}, \ldots, a^{n_L 2^L}$) Answer is $a^{n_0 2^0} \times \cdots \times a^{n_L 2^L}$

Repeated Squaring Algorithm

All math is mod p.

1. Input (a, n, p).
2. Convert n to base 2: $n = \sum_{i=0}^{L} n_i 2^i$. ($L$ is $\lfloor \lg(n) \rfloor$)
3. $x_0 = a$.
4. For $i = 1$ to L, $x_i = x_{i-1}^2$
5. (Now have $a^{n_0 2^0}, \ldots, a^{n_L 2^L}$) Answer is $a^{n_0 2^0} \times \cdots \times a^{n_L 2^L}$

Number of operations:

Repeated Squaring Algorithm

All math is mod p.

1. Input (a, n, p).
2. Convert n to base 2: $n = \sum_{i=0}^{L} n_i 2^i$. ($L$ is $\lfloor \lg(n) \rfloor$)
3. $x_0 = a$.
4. For $i = 1$ to L, $x_i = x_{i-1}^2$
5. (Now have $a^{n_0 2^0}, \ldots, a^{n_L 2^L}$) Answer is $a^{n_0 2^0} \times \cdots \times a^{n_L 2^L}$

Number of operations:

Number of **MULTS** in step 4: $\leq \lfloor \lg(n) \rfloor \leq \lg(n)$

Repeated Squaring Algorithm

All math is mod p.

1. Input (a, n, p).
2. Convert n to base 2: $n = \sum_{i=0}^{L} n_i 2^i$. ($L$ is $\lfloor \lg(n) \rfloor$)
3. $x_0 = a$.
4. For $i = 1$ to L, $x_i = x_{i-1}^2$
5. (Now have $a^{n_0 2^0}, \ldots, a^{n_L 2^L}$) Answer is $a^{n_0 2^0} \times \cdots \times a^{n_L 2^L}$

Number of operations:

Number of **MULTS** in step 4: $\leq \lfloor \lg(n) \rfloor \leq \lg(n)$

Number of **MULTS** in step 5: $\leq L = \lfloor \lg(n) \rfloor \leq \lg(n)$

Repeated Squaring Algorithm

All math is mod p.

1. Input (a, n, p).
2. Convert n to base 2: $n = \sum_{i=0}^{L} n_i 2^i$. ($L$ is $\lfloor \lg(n) \rfloor$)
3. $x_0 = a$.
4. For $i = 1$ to L, $x_i = x_{i-1}^2$
5. (Now have $a^{n_0 2^0}, \ldots, a^{n_L 2^L}$) Answer is $a^{n_0 2^0} \times \cdots \times a^{n_L 2^L}$

Number of operations:
Number of **MULTS** in step 4: $\leq \lfloor \lg(n) \rfloor \leq \lg(n)$
Number of **MULTS** in step 5: $\leq L = \lfloor \lg(n) \rfloor \leq \lg(n)$
Total number of **MULTS** $\leq 2\lg(n)$.

Repeated Squaring Algorithm

All math is mod p.

1. Input (a, n, p).
2. Convert n to base 2: $n = \sum_{i=0}^{L} n_i 2^i$. ($L$ is $\lfloor \lg(n) \rfloor$)
3. $x_0 = a$.
4. For $i = 1$ to L, $x_i = x_{i-1}^2$
5. (Now have $a^{n_0 2^0}, \ldots, a^{n_L 2^L}$) Answer is $a^{n_0 2^0} \times \cdots \times a^{n_L 2^L}$

Number of operations:

Number of **MULTS** in step 4: $\leq \lfloor \lg(n) \rfloor \leq \lg(n)$

Number of **MULTS** in step 5: $\leq L = \lfloor \lg(n) \rfloor \leq \lg(n)$

Total number of **MULTS** $\leq 2\lg(n)$.

More refined: $\lg(n) +$ (number of 1's in binary rep of n) -1

Repeated Squaring Algorithm

All math is mod p.

1. Input (a, n, p).
2. Convert n to base 2: $n = \sum_{i=0}^{L} n_i 2^i$. ($L$ is $\lfloor \lg(n) \rfloor$)
3. $x_0 = a$.
4. For $i = 1$ to L, $x_i = x_{i-1}^2$
5. (Now have $a^{n_0 2^0}, \ldots, a^{n_L 2^L}$) Answer is $a^{n_0 2^0} \times \cdots \times a^{n_L 2^L}$

Number of operations:
Number of **MULTS** in step 4: $\leq \lfloor \lg(n) \rfloor \leq \lg(n)$
Number of **MULTS** in step 5: $\leq L = \lfloor \lg(n) \rfloor \leq \lg(n)$
Total number of **MULTS** $\leq 2\lg(n)$.
More refined: $\lg(n) +$ (number of 1's in binary rep of n) -1
Example on next page

Example of Exponentiation: $17^{265} \pmod{101}$

$$265 = 2^8 + 2^3 + 2^0 = (100001001)_2$$

Example of Exponentiation: $17^{265} \pmod{101}$

$$265 = 2^8 + 2^3 + 2^0 = (100001001)_2$$

$17^{2^0} \equiv 17$ (0 steps)

Example of Exponentiation: $17^{265} \pmod{101}$

$$265 = 2^8 + 2^3 + 2^0 = (100001001)_2$$

$17^{2^0} \equiv 17$ (0 steps)
$17^{2^1} \equiv 17^2 \equiv 87$ (1 step)

Example of Exponentiation: $17^{265} \pmod{101}$

$$265 = 2^8 + 2^3 + 2^0 = (100001001)_2$$

$$17^{2^0} \equiv 17 \text{ (0 steps)}$$
$$17^{2^1} \equiv 17^2 \equiv 87 \text{ (1 step)}$$
$$17^{2^2} \equiv 87^2 \equiv 95 \text{ (1 step)}$$

Example of Exponentiation: $17^{265} \pmod{101}$

$$265 = 2^8 + 2^3 + 2^0 = (100001001)_2$$

$$
\begin{aligned}
17^{2^0} &\equiv 17 \text{ (0 steps)}\\
17^{2^1} &\equiv 17^2 \equiv 87 \text{ (1 step)}\\
17^{2^2} &\equiv 87^2 \equiv 95 \text{ (1 step)}\\
17^{2^3} &\equiv 95^2 \equiv 36 \text{ (1 step)}
\end{aligned}
$$

Example of Exponentiation: $17^{265} \pmod{101}$

$$265 = 2^8 + 2^3 + 2^0 = (100001001)_2$$

$$\begin{aligned}
17^{2^0} &\equiv 17 \text{ (0 steps)} \\
17^{2^1} &\equiv 17^2 \equiv 87 \text{ (1 step)} \\
17^{2^2} &\equiv 87^2 \equiv 95 \text{ (1 step)} \\
17^{2^3} &\equiv 95^2 \equiv 36 \text{ (1 step)} \\
17^{2^4} &\equiv 36^2 \equiv 84 \text{ (1 step)}
\end{aligned}$$

Example of Exponentiation: $17^{265} \pmod{101}$

$$265 = 2^8 + 2^3 + 2^0 = (100001001)_2$$

$17^{2^0} \equiv 17$ (0 steps)
$17^{2^1} \equiv 17^2 \equiv 87$ (1 step)
$17^{2^2} \equiv 87^2 \equiv 95$ (1 step)
$17^{2^3} \equiv 95^2 \equiv 36$ (1 step)
$17^{2^4} \equiv 36^2 \equiv 84$ (1 step)
$17^{2^5} \equiv 84^2 \equiv 87$ (1 step)

Example of Exponentiation: $17^{265} \pmod{101}$

$$265 = 2^8 + 2^3 + 2^0 = (100001001)_2$$

$$\begin{aligned}
17^{2^0} &\equiv 17 \text{ (0 steps)} \\
17^{2^1} &\equiv 17^2 \equiv 87 \text{ (1 step)} \\
17^{2^2} &\equiv 87^2 \equiv 95 \text{ (1 step)} \\
17^{2^3} &\equiv 95^2 \equiv 36 \text{ (1 step)} \\
17^{2^4} &\equiv 36^2 \equiv 84 \text{ (1 step)} \\
17^{2^5} &\equiv 84^2 \equiv 87 \text{ (1 step)} \\
17^{2^6} &\equiv 87^2 \equiv 95 \text{ (1 step)}
\end{aligned}$$

Example of Exponentiation: $17^{265} \pmod{101}$

$$265 = 2^8 + 2^3 + 2^0 = (100001001)_2$$

$$\begin{aligned}
17^{2^0} &\equiv 17 \text{ (0 steps)}\\
17^{2^1} &\equiv 17^2 \equiv 87 \text{ (1 step)}\\
17^{2^2} &\equiv 87^2 \equiv 95 \text{ (1 step)}\\
17^{2^3} &\equiv 95^2 \equiv 36 \text{ (1 step)}\\
17^{2^4} &\equiv 36^2 \equiv 84 \text{ (1 step)}\\
17^{2^5} &\equiv 84^2 \equiv 87 \text{ (1 step)}\\
17^{2^6} &\equiv 87^2 \equiv 95 \text{ (1 step)}\\
17^{2^7} &\equiv 95^2 \equiv 36 \text{ (1 step)}
\end{aligned}$$

Example of Exponentiation: $17^{265} \pmod{101}$

$$265 = 2^8 + 2^3 + 2^0 = (100001001)_2$$

$17^{2^0} \equiv 17$ (0 steps)
$17^{2^1} \equiv 17^2 \equiv 87$ (1 step)
$17^{2^2} \equiv 87^2 \equiv 95$ (1 step)
$17^{2^3} \equiv 95^2 \equiv 36$ (1 step)
$17^{2^4} \equiv 36^2 \equiv 84$ (1 step)
$17^{2^5} \equiv 84^2 \equiv 87$ (1 step)
$17^{2^6} \equiv 87^2 \equiv 95$ (1 step)
$17^{2^7} \equiv 95^2 \equiv 36$ (1 step)
$17^{2^8} \equiv 36^2 \equiv 84$ (1 step)

Example of Exponentiation: $17^{265} \pmod{101}$

$$265 = 2^8 + 2^3 + 2^0 = (100001001)_2$$

$17^{2^0} \equiv 17$ (0 steps)
$17^{2^1} \equiv 17^2 \equiv 87$ (1 step)
$17^{2^2} \equiv 87^2 \equiv 95$ (1 step)
$17^{2^3} \equiv 95^2 \equiv 36$ (1 step)
$17^{2^4} \equiv 36^2 \equiv 84$ (1 step)
$17^{2^5} \equiv 84^2 \equiv 87$ (1 step)
$17^{2^6} \equiv 87^2 \equiv 95$ (1 step)
$17^{2^7} \equiv 95^2 \equiv 36$ (1 step)
$17^{2^8} \equiv 36^2 \equiv 84$ (1 step)
This took $8 \sim \lg(265)$ multiplications so far.

Example of Exponentiation: $17^{265} \pmod{101}$

$$265 = 2^8 + 2^3 + 2^0 = (100001001)_2$$

$17^{2^0} \equiv 17$ (0 steps)
$17^{2^1} \equiv 17^2 \equiv 87$ (1 step)
$17^{2^2} \equiv 87^2 \equiv 95$ (1 step)
$17^{2^3} \equiv 95^2 \equiv 36$ (1 step)
$17^{2^4} \equiv 36^2 \equiv 84$ (1 step)
$17^{2^5} \equiv 84^2 \equiv 87$ (1 step)
$17^{2^6} \equiv 87^2 \equiv 95$ (1 step)
$17^{2^7} \equiv 95^2 \equiv 36$ (1 step)
$17^{2^8} \equiv 36^2 \equiv 84$ (1 step)
This took $8 \sim \lg(265)$ multiplications so far.
The next step takes only two multiplications:

$$17^{265} \equiv 17^{2^8} \times 17^{2^3} \times 17^{2^0} \equiv 84 \times 36 \times 17 \equiv 100$$

Example of Exponentiation: $17^{265} \pmod{101}$

$$265 = 2^8 + 2^3 + 2^0 = (100001001)_2$$

$17^{2^0} \equiv 17$ (0 steps)
$17^{2^1} \equiv 17^2 \equiv 87$ (1 step)
$17^{2^2} \equiv 87^2 \equiv 95$ (1 step)
$17^{2^3} \equiv 95^2 \equiv 36$ (1 step)
$17^{2^4} \equiv 36^2 \equiv 84$ (1 step)
$17^{2^5} \equiv 84^2 \equiv 87$ (1 step)
$17^{2^6} \equiv 87^2 \equiv 95$ (1 step)
$17^{2^7} \equiv 95^2 \equiv 36$ (1 step)
$17^{2^8} \equiv 36^2 \equiv 84$ (1 step)
This took $8 \sim \lg(265)$ multiplications so far.
The next step takes only two multiplications:

$$17^{265} \equiv 17^{2^8} \times 17^{2^3} \times 17^{2^0} \equiv 84 \times 36 \times 17 \equiv 100$$

Point: Step 2 took $< \lg(265)$ steps since base-2 rep had few 1's.

Generators and Discrete Logarithms

Generators $\pmod{p}$

Let's take powers of 3 mod 7. All math is mod 7.

Generators $\pmod{p}$

Let's take powers of 3 mod 7. All math is mod 7.
$3^1 \equiv 3$

Generators $\pmod{p}$

Let's take powers of 3 mod 7. All math is mod 7.
$3^1 \equiv 3$
$3^2 \equiv 3 \times 3^1 \equiv 9 \equiv 2$

Generators $\pmod{p}$

Let's take powers of 3 mod 7. All math is mod 7.
$3^1 \equiv 3$
$3^2 \equiv 3 \times 3^1 \equiv 9 \equiv 2$
$3^3 \equiv 3 \times 3^2 \equiv 3 \times 2 \equiv 6$

Generators $\pmod{p}$

Let's take powers of 3 mod 7. All math is mod 7.

$3^1 \equiv 3$

$3^2 \equiv 3 \times 3^1 \equiv 9 \equiv 2$

$3^3 \equiv 3 \times 3^2 \equiv 3 \times 2 \equiv 6$

$3^4 \equiv 3 \times 3^3 \equiv 3 \times 6 \equiv 18 \equiv 4$

Generators $\pmod{p}$

Let's take powers of 3 mod 7. All math is mod 7.
$3^1 \equiv 3$
$3^2 \equiv 3 \times 3^1 \equiv 9 \equiv 2$
$3^3 \equiv 3 \times 3^2 \equiv 3 \times 2 \equiv 6$
$3^4 \equiv 3 \times 3^3 \equiv 3 \times 6 \equiv 18 \equiv 4$
$3^5 \equiv 3 \times 3^4 \equiv 3 \times 4 \equiv 12 \equiv 5$

Generators $\pmod{p}$

Let's take powers of 3 mod 7. All math is mod 7.

$3^1 \equiv 3$

$3^2 \equiv 3 \times 3^1 \equiv 9 \equiv 2$

$3^3 \equiv 3 \times 3^2 \equiv 3 \times 2 \equiv 6$

$3^4 \equiv 3 \times 3^3 \equiv 3 \times 6 \equiv 18 \equiv 4$

$3^5 \equiv 3 \times 3^4 \equiv 3 \times 4 \equiv 12 \equiv 5$

$3^6 \equiv 3 \times 3^5 \equiv 3 \times 5 \equiv 15 \equiv 1$

Generators $\pmod{p}$

Let's take powers of 3 mod 7. All math is mod 7.
$3^1 \equiv 3$
$3^2 \equiv 3 \times 3^1 \equiv 9 \equiv 2$
$3^3 \equiv 3 \times 3^2 \equiv 3 \times 2 \equiv 6$
$3^4 \equiv 3 \times 3^3 \equiv 3 \times 6 \equiv 18 \equiv 4$
$3^5 \equiv 3 \times 3^4 \equiv 3 \times 4 \equiv 12 \equiv 5$
$3^6 \equiv 3 \times 3^5 \equiv 3 \times 5 \equiv 15 \equiv 1$

$$\{3^1, 3^2, 3^3, 3^4, 3^5, 3^6\} = \{1, 2, 3, 4, 5, 6\} \text{ Not in order.}$$

Generators $\pmod{p}$

Let's take powers of 3 mod 7. All math is mod 7.

$3^1 \equiv 3$
$3^2 \equiv 3 \times 3^1 \equiv 9 \equiv 2$
$3^3 \equiv 3 \times 3^2 \equiv 3 \times 2 \equiv 6$
$3^4 \equiv 3 \times 3^3 \equiv 3 \times 6 \equiv 18 \equiv 4$
$3^5 \equiv 3 \times 3^4 \equiv 3 \times 4 \equiv 12 \equiv 5$
$3^6 \equiv 3 \times 3^5 \equiv 3 \times 5 \equiv 15 \equiv 1$

$$\{3^1, 3^2, 3^3, 3^4, 3^5, 3^6\} = \{1, 2, 3, 4, 5, 6\} \text{ Not in order.}$$

3 is a **generator** for $\mathbb{Z}_7^*$.

Generators $\pmod{p}$

Let's take powers of 3 mod 7. All math is mod 7.
$3^1 \equiv 3$
$3^2 \equiv 3 \times 3^1 \equiv 9 \equiv 2$
$3^3 \equiv 3 \times 3^2 \equiv 3 \times 2 \equiv 6$
$3^4 \equiv 3 \times 3^3 \equiv 3 \times 6 \equiv 18 \equiv 4$
$3^5 \equiv 3 \times 3^4 \equiv 3 \times 4 \equiv 12 \equiv 5$
$3^6 \equiv 3 \times 3^5 \equiv 3 \times 5 \equiv 15 \equiv 1$

$$\{3^1, 3^2, 3^3, 3^4, 3^5, 3^6\} = \{1, 2, 3, 4, 5, 6\} \text{ Not in order.}$$

3 is a **generator** for $\mathbb{Z}_7^*$.
Definition: If p is a prime and $\{g^1, \ldots, g^{p-1}\} = \{1, \ldots, p-1\}$ then g is a **generator** for $\mathbb{Z}_p^*$.

Discrete Log-Example

Fact: 3 is a generator mod 101. All math is mod 101.
Discuss the following with your neighbor:

1. Find x such that $3^x \equiv 81$.

Discrete Log-Example

Fact: 3 is a generator mod 101. All math is mod 101.
Discuss the following with your neighbor:

1. Find x such that $3^x \equiv 81$. $x = 4$ obv works.

Discrete Log-Example

Fact: 3 is a generator mod 101. All math is mod 101.
Discuss the following with your neighbor:

1. Find x such that $3^x \equiv 81$. $x = 4$ obv works.
2. Find x such that $3^x \equiv 92$.

Discrete Log-Example

Fact: 3 is a generator mod 101. All math is mod 101.
Discuss the following with your neighbor:

1. Find x such that $3^x \equiv 81$. $x = 4$ obv works.
2. Find x such that $3^x \equiv 92$.
   Try computing $3^1, 3^2, \ldots,$ until you get 92.

Discrete Log-Example

Fact: 3 is a generator mod 101. All math is mod 101.
Discuss the following with your neighbor:

1. Find x such that $3^x \equiv 81$. $x = 4$ obv works.
2. Find x such that $3^x \equiv 92$.
   Try computing $3^1, 3^2, \ldots,$ until you get 92.
   Might take ~ 100 steps.

Discrete Log-Example

Fact: 3 is a generator mod 101. All math is mod 101.
Discuss the following with your neighbor:

1. Find x such that $3^x \equiv 81$. $x = 4$ obv works.
2. Find x such that $3^x \equiv 92$.
   Try computing $3^1, 3^2, \ldots,$ until you get 92.
   Might take ~ 100 steps. Shortcut?

Discrete Log-Example

Fact: 3 is a generator mod 101. All math is mod 101.
Discuss the following with your neighbor:

1. Find x such that $3^x \equiv 81$. $x = 4$ obv works.
2. Find x such that $3^x \equiv 92$.
   Try computing $3^1, 3^2, \ldots,$ until you get 92.
   Might take ~ 100 steps. Shortcut?
3. Find x such that $3^x \equiv 93$.

Discrete Log-Example

Fact: 3 is a generator mod 101. All math is mod 101.
Discuss the following with your neighbor:

1. Find x such that $3^x \equiv 81$. $x = 4$ obv works.
2. Find x such that $3^x \equiv 92$.
   Try computing $3^1, 3^2, \ldots,$ until you get 92.
   Might take ~ 100 steps. Shortcut?
3. Find x such that $3^x \equiv 93$.
   Try computing $3^1, 3^2, \ldots,$ until you get 93.

Discrete Log-Example

Fact: 3 is a generator mod 101. All math is mod 101.
Discuss the following with your neighbor:

1. Find x such that $3^x \equiv 81$. $x = 4$ obv works.
2. Find x such that $3^x \equiv 92$.
   Try computing $3^1, 3^2, \ldots,$ until you get 92.
   Might take ~ 100 steps. Shortcut?
3. Find x such that $3^x \equiv 93$.
   Try computing $3^1, 3^2, \ldots,$ until you get 93.
   Might take ~ 100 steps.

Discrete Log-Example

Fact: 3 is a generator mod 101. All math is mod 101.
Discuss the following with your neighbor:

1. Find x such that $3^x \equiv 81$. $x = 4$ obv works.
2. Find x such that $3^x \equiv 92$.
   Try computing $3^1, 3^2, \ldots,$ until you get 92.
   Might take ~ 100 steps. Shortcut?
3. Find x such that $3^x \equiv 93$.
   Try computing $3^1, 3^2, \ldots,$ until you get 93.
   Might take ~ 100 steps. Shortcut?

Discrete Log-Example

Fact: 3 is a generator mod 101. All math is mod 101.
Discuss the following with your neighbor:

1. Find x such that $3^x \equiv 81$. $x = 4$ obv works.
2. Find x such that $3^x \equiv 92$.
   Try computing $3^1, 3^2, \ldots,$ until you get 92.
   Might take ~ 100 steps. Shortcut?
3. Find x such that $3^x \equiv 93$.
   Try computing $3^1, 3^2, \ldots,$ until you get 93.
   Might take ~ 100 steps. Shortcut?

2nd and 3th look hard. Are they?

Discrete Log-Example

Fact: 3 is a generator mod 101. All math is mod 101.
Discuss the following with your neighbor:

1. Find x such that $3^x \equiv 81$. $x = 4$ obv works.
2. Find x such that $3^x \equiv 92$.
   Try computing $3^1, 3^2, \ldots,$ until you get 92.
   Might take ~ 100 steps. Shortcut?
3. Find x such that $3^x \equiv 93$.
   Try computing $3^1, 3^2, \ldots,$ until you get 93.
   Might take ~ 100 steps. Shortcut?

2nd and 3th look hard. Are they?
VOTE Both hard, both easy, one of each, unknown to science.

Discrete Log-Example

Fact: 3 is a generator mod 101. All math is mod 101.
Discuss the following with your neighbor:

1. Find x such that $3^x \equiv 81$. $x = 4$ obv works.
2. Find x such that $3^x \equiv 92$.
   Try computing $3^1, 3^2, \ldots,$ until you get 92.
   Might take ~ 100 steps. Shortcut?
3. Find x such that $3^x \equiv 93$.
   Try computing $3^1, 3^2, \ldots,$ until you get 93.
   Might take ~ 100 steps. Shortcut?

2nd and 3th look hard. Are they?
VOTE Both hard, both easy, one of each, unknown to science.

$3^x \equiv 92$ easy. $3^x \equiv 93$ Not known how hard.

Discrete Log-Example: $3^x \equiv 92 \pmod{101}$

Fact: 3 is a generator mod 101. All math is mod 101.

Discrete Log-Example: $3^x \equiv 92 \pmod{101}$

Fact: 3 is a generator mod 101. All math is mod 101.
Find x such that $3^x \equiv 92$. Easy!

Discrete Log-Example: $3^x \equiv 92 \pmod{101}$

Fact: 3 is a generator mod 101. All math is mod 101.
Find x such that $3^x \equiv 92$. Easy!

1. $92 \equiv 101 - 9 \equiv (-1)(9) \equiv (-1)3^2$.

Discrete Log-Example: $3^x \equiv 92 \pmod{101}$

Fact: 3 is a generator mod 101. All math is mod 101.
Find x such that $3^x \equiv 92$. Easy!

1. $92 \equiv 101 - 9 \equiv (-1)(9) \equiv (-1)3^2$.
2. $3^{50} \equiv -1$ (WHAT! Really?)

Discrete Log-Example: $3^x \equiv 92 \pmod{101}$

Fact: 3 is a generator mod 101. All math is mod 101.
Find x such that $3^x \equiv 92$. Easy!

1. $92 \equiv 101 - 9 \equiv (-1)(9) \equiv (-1)3^2$.
2. $3^{50} \equiv -1$ (WHAT! Really?)
3. $92 \equiv 3^{50} \times 3^2 \equiv 3^{52}$. So $x = 52$ works.

Discrete Log-Example: $3^x \equiv 92 \pmod{101}$

Fact: 3 is a generator mod 101. All math is mod 101.
Find x such that $3^x \equiv 92$. Easy!

1. $92 \equiv 101 - 9 \equiv (-1)(9) \equiv (-1)3^2$.
2. $3^{50} \equiv -1$ (WHAT! Really?)
3. $92 \equiv 3^{50} \times 3^2 \equiv 3^{52}$. So $x = 52$ works.

Generalize:

Discrete Log-Example: $3^x \equiv 92 \pmod{101}$

Fact: 3 is a generator mod 101. All math is mod 101.
Find x such that $3^x \equiv 92$. Easy!

1. $92 \equiv 101 - 9 \equiv (-1)(9) \equiv (-1)3^2$.
2. $3^{50} \equiv -1$ (WHAT! Really?)
3. $92 \equiv 3^{50} \times 3^2 \equiv 3^{52}$. So $x = 52$ works.

Generalize:

1. If g is a generator of $\mathbb{Z}_p^*$ then $g^{(p-1)/2} \equiv p - 1 \equiv -1$.

Discrete Log-Example: $3^x \equiv 92 \pmod{101}$

Fact: 3 is a generator mod 101. All math is mod 101.
Find x such that $3^x \equiv 92$. Easy!

1. $92 \equiv 101 - 9 \equiv (-1)(9) \equiv (-1)3^2$.
2. $3^{50} \equiv -1$ (WHAT! Really?)
3. $92 \equiv 3^{50} \times 3^2 \equiv 3^{52}$. So $x = 52$ works.

Generalize:

1. If g is a generator of $\mathbb{Z}_p^*$ then $g^{(p-1)/2} \equiv p - 1 \equiv -1$.
2. So finding x such that $g^x \equiv p - g^a \equiv -g^a$ is as easy as g^a.

$$x = \frac{p-1}{2} + a : \qquad g^{\frac{p-1}{2}+a} = g^{\frac{p-1}{2}} g^a \equiv -g^a$$

Discrete Log-Example: $3^x \equiv 93 \pmod{101}$

Fact: 3 is a generator mod 101. All math is mod 101.
Is there a trick for $g^x \equiv 93 \pmod{101}$? Not that I know of.

Formally Discrete Log is...

Def The **Discrete Log (DL)** problem is a follows:

Formally Discrete Log is...

Def The **Discrete Log (DL)** problem is a follows:

1. Input g, a, p. With $1 \le g, a \le p-1$. g is a gen for $\mathbb{Z}_p^*$.

Formally Discrete Log is...

Def The **Discrete Log (DL)** problem is a follows:

1. Input g, a, p. With $1 \le g, a \le p-1$. g is a gen for $\mathbb{Z}_p^*$.
2. Output x such that $g^x \equiv a \pmod{p}$.

Formally Discrete Log is. . .

Def The **Discrete Log (DL)** problem is a follows:

1. Input g, a, p. With $1 \le g, a \le p-1$. g is a gen for $\mathbb{Z}_p^*$.
2. Output x such that $g^x \equiv a \pmod{p}$.

Recall

Formally Discrete Log is...

Def The **Discrete Log (DL)** problem is a follows:

1. Input g, a, p. With $1 \le g, a \le p-1$. g is a gen for $\mathbb{Z}_p^*$.
2. Output x such that $g^x \equiv a \pmod{p}$.

Recall

- A **good** alg would be time $(\log p)^{O(1)}$.

Formally Discrete Log is...

Def The **Discrete Log (DL)** problem is a follows:

1. Input g, a, p. With $1 \le g, a \le p-1$. g is a gen for $\mathbb{Z}_p^*$.
2. Output x such that $g^x \equiv a \pmod{p}$.

Recall

- A **good** alg would be time $(\log p)^{O(1)}$.
- A **bad** alg would be time $p^{O(1)}$.

Formally Discrete Log is...

Def The **Discrete Log (DL)** problem is a follows:

1. Input g, a, p. With $1 \le g, a \le p-1$. g is a gen for $\mathbb{Z}_p^*$.
2. Output x such that $g^x \equiv a \pmod{p}$.

Recall

- A **good** alg would be time $(\log p)^{O(1)}$.
- A **bad** alg would be time $p^{O(1)}$.
- If an algorithm is in time (say) $p^{1/10}$ still not efficient but will force Alice and Bob to up their game.

The Complexity of Discrete Log?

Input is (g, a, p).

The Complexity of Discrete Log?

Input is (g, a, p).

1. Naive algorithm is $O(p)$ time.

The Complexity of Discrete Log?

Input is (g, a, p).

1. Naive algorithm is $O(p)$ time.
2. Exists an $O(\sqrt{p})$ time, $O(\sqrt{p})$ space alg. Time and Space makes it NOT practical.

The Complexity of Discrete Log?

Input is (g, a, p).

1. Naive algorithm is $O(p)$ time.
2. Exists an $O(\sqrt{p})$ time, $O(\sqrt{p})$ space alg. Time and Space makes it NOT practical.
3. Exists an $O(\sqrt{p})$ time, $(\log p)^{O(1)}$ space alg. Space fine, but time still a problem.

The Complexity of Discrete Log?

Input is (g, a, p).

1. Naive algorithm is $O(p)$ time.
2. Exists an $O(\sqrt{p})$ time, $O(\sqrt{p})$ space alg. Time and Space makes it NOT practical.
3. Exists an $O(\sqrt{p})$ time, $(\log p)^{O(1)}$ space alg. Space fine, but time still a problem.
4. Not much progress on theory front since 1985.

The Complexity of Discrete Log?

Input is (g, a, p).

1. Naive algorithm is $O(p)$ time.
2. Exists an $O(\sqrt{p})$ time, $O(\sqrt{p})$ space alg. Time and Space makes it NOT practical.
3. Exists an $O(\sqrt{p})$ time, $(\log p)^{O(1)}$ space alg. Space fine, but time still a problem.
4. Not much progress on theory front since 1985.
5. Discrete Log is in QuantumP.

The Complexity of Discrete Log?

Input is (g, a, p).

1. Naive algorithm is $O(p)$ time.
2. Exists an $O(\sqrt{p})$ time, $O(\sqrt{p})$ space alg. Time and Space makes it NOT practical.
3. Exists an $O(\sqrt{p})$ time, $(\log p)^{O(1)}$ space alg. Space fine, but time still a problem.
4. Not much progress on theory front since 1985.
5. Discrete Log is in QuantumP.

Good Candidate for a hard problem for Eve.

Bill's Opinion on DL. Also Applies to Factoring

Bill's Opinion on DL. Also Applies to Factoring

1. **Fact:** DL in in QuantumP.

Bill's Opinion on DL. Also Applies to Factoring

1. **Fact:** DL in in QuantumP.
2. **BILL Opinion:** Quantum computers that can do DL **fast** won't happen

Bill's Opinion on DL. Also Applies to Factoring

1. **Fact:** DL in in QuantumP.
2. **BILL Opinion:** Quantum computers that can do DL **fast** won't happen in my lifetime.

Bill's Opinion on DL. Also Applies to Factoring

1. **Fact:** DL in in QuantumP.
2. **BILL Opinion:** Quantum computers that can do DL **fast** won't happen in my lifetime. In your lifetime.

Bill's Opinion on DL. Also Applies to Factoring

1. **Fact:** DL in in QuantumP.
2. **BILL Opinion:** Quantum computers that can do DL **fast** won't happen in my lifetime. In your lifetime. Ever.

Bill's Opinion on DL. Also Applies to Factoring

1. **Fact:** DL in in QuantumP.
2. **BILL Opinion:** Quantum computers that can do DL **fast** won't happen in my lifetime. In your lifetime. Ever.
3. **Fact:** Good classical algorithms using hard number theory exist and have been implemented. Still exponential but low constants.

Bill's Opinion on DL. Also Applies to Factoring

1. **Fact:** DL in in QuantumP.
2. **BILL Opinion:** Quantum computers that can do DL **fast** won't happen in my lifetime. In your lifetime. Ever.
3. **Fact:** Good classical algorithms using hard number theory exist and have been implemented. Still exponential but low constants. Some are amenable to parallelism.

Bill's Opinion on DL. Also Applies to Factoring

1. **Fact:** DL in in QuantumP.
2. **BILL Opinion:** Quantum computers that can do DL **fast** won't happen in my lifetime. In your lifetime. Ever.
3. **Fact:** Good classical algorithms using hard number theory exist and have been implemented. Still exponential but low constants. Some are amenable to parallelism.
4. **BILL Opinion:** The biggest threat to crypto is from hard math combined with special purpose parallel hardware.

Bill's Opinion on DL. Also Applies to Factoring

1. **Fact:** DL in in QuantumP.
2. **BILL Opinion:** Quantum computers that can do DL **fast** won't happen in my lifetime. In your lifetime. Ever.
3. **Fact:** Good classical algorithms using hard number theory exist and have been implemented. Still exponential but low constants. Some are amenable to parallelism.
4. **BILL Opinion:** The biggest threat to crypto is from hard math combined with special purpose parallel hardware.
5. **Fact:** If computers do DL much better (e.g., $O(p^{1/10})$) then Alice and Bob can increase size of p and be fine. Still, Eve has made them work harder.

Bill's Opinion on DL. Also Applies to Factoring

1. **Fact:** DL in in QuantumP.
2. **BILL Opinion:** Quantum computers that can do DL **fast** won't happen in my lifetime. In your lifetime. Ever.
3. **Fact:** Good classical algorithms using hard number theory exist and have been implemented. Still exponential but low constants. Some are amenable to parallelism.
4. **BILL Opinion:** The biggest threat to crypto is from hard math combined with special purpose parallel hardware.
5. **Fact:** If computers do DL much better (e.g., $O(p^{1/10})$) then Alice and Bob can increase size of p and be fine. Still, Eve has made them work harder.
6. **BILL Opinion:** When people really really need to up their parameters

Bill's Opinion on DL. Also Applies to Factoring

1. **Fact:** DL in in QuantumP.
2. **BILL Opinion:** Quantum computers that can do DL **fast** won't happen in my lifetime. In your lifetime. Ever.
3. **Fact:** Good classical algorithms using hard number theory exist and have been implemented. Still exponential but low constants. Some are amenable to parallelism.
4. **BILL Opinion:** The biggest threat to crypto is from hard math combined with special purpose parallel hardware.
5. **Fact:** If computers do DL much better (e.g., $O(p^{1/10})$) then Alice and Bob can increase size of p and be fine. Still, Eve has made them work harder.
6. **BILL Opinion:** When people really really need to up their parameters they don't.

Bill's Opinion on DL. Also Applies to Factoring

1. **Fact:** DL in in QuantumP.
2. **BILL Opinion:** Quantum computers that can do DL **fast** won't happen in my lifetime. In your lifetime. Ever.
3. **Fact:** Good classical algorithms using hard number theory exist and have been implemented. Still exponential but low constants. Some are amenable to parallelism.
4. **BILL Opinion:** The biggest threat to crypto is from hard math combined with special purpose parallel hardware.
5. **Fact:** If computers do DL much better (e.g., $O(p^{1/10})$) then Alice and Bob can increase size of p and be fine. Still, Eve has made them work harder.
6. **BILL Opinion:** When people really really need to up their parameters they don't.They say

Bill's Opinion on DL. Also Applies to Factoring

1. **Fact:** DL in in QuantumP.
2. **BILL Opinion:** Quantum computers that can do DL **fast** won't happen in my lifetime. In your lifetime. Ever.
3. **Fact:** Good classical algorithms using hard number theory exist and have been implemented. Still exponential but low constants. Some are amenable to parallelism.
4. **BILL Opinion:** The biggest threat to crypto is from hard math combined with special purpose parallel hardware.
5. **Fact:** If computers do DL much better (e.g., $O(p^{1/10})$) then Alice and Bob can increase size of p and be fine. Still, Eve has made them work harder.
6. **BILL Opinion:** When people really really need to up their parameters they don't.They say

   It won't happen to me

Bill's Opinion on DL. Also Applies to Factoring

1. **Fact:** DL in in QuantumP.
2. **BILL Opinion:** Quantum computers that can do DL **fast** won't happen in my lifetime. In your lifetime. Ever.
3. **Fact:** Good classical algorithms using hard number theory exist and have been implemented. Still exponential but low constants. Some are amenable to parallelism.
4. **BILL Opinion:** The biggest threat to crypto is from hard math combined with special purpose parallel hardware.
5. **Fact:** If computers do DL much better (e.g., $O(p^{1/10})$) then Alice and Bob can increase size of p and be fine. Still, Eve has made them work harder.
6. **BILL Opinion:** When people really really need to up their parameters they don't.They say

   It won't happen to me Until it does.

Kunal's Opinion on DL. Also Applies to Factoring

Kunal's Opinion on DL. Also Applies to Factoring

1. Quantum computing has the power to break modern crypto. But

Kunal's Opinion on DL. Also Applies to Factoring

1. Quantum computing has the power to break modern crypto. But
2. Quantum computing is like graphene: quantum can do everything except leave the lab. See `https://en.wikipedia.org/wiki/Graphene` for information on graphene which seems to always be 5 years away from applications.

Kunal's Opinion on DL. Also Applies to Factoring

1. Quantum computing has the power to break modern crypto. But
2. Quantum computing is like graphene: quantum can do everything except leave the lab. See `https://en.wikipedia.org/wiki/Graphene` for information on graphene which seems to always be 5 years away from applications.

So Kunal and Bill agree.

Josh's Opinion on DL. Also Applies to Factoring

Josh's Opinion on DL. Also Applies to Factoring

1. **PRO** A lot of corporations and governments are putting a lot of money into Quantum, so (unlike other alt-computing ideas) this one really has a shot.

Josh's Opinion on DL. Also Applies to Factoring

1. **PRO** A lot of corporations and governments are putting a lot of money into Quantum, so (unlike other alt-computing ideas) this one really has a shot.
2. **CON** The error-correction problem still seems hard.

Josh's Opinion on DL. Also Applies to Factoring

1. **PRO** A lot of corporations and governments are putting a lot of money into Quantum, so (unlike other alt-computing ideas) this one really has a shot.
2. **CON** The error-correction problem still seems hard.
3. **CONCLUSION** The question **When will quantum computers be able to really do DL fast** should be asked to physicists, not to CMSC/ENEE/MATH TAs.

Josh's Opinion on DL. Also Applies to Factoring

1. **PRO** A lot of corporations and governments are putting a lot of money into Quantum, so (unlike other alt-computing ideas) this one really has a shot.
2. **CON** The error-correction problem still seems hard.
3. **CONCLUSION** The question **When will quantum computers be able to really do DL fast** should be asked to physicists, not to CMSC/ENEE/MATH TAs.

Bill Since lots of money is being put into it, if it does not work they won't have the excuse that other technologies have of not having been tried.

Sajjad Opinion on DL. Also Applies to Factoring

Sajjad is working in Quantum Computing so

Sajjad Opinion on DL. Also Applies to Factoring

Sajjad is working in Quantum Computing so

1. He actually knows stuff.

Sajjad Opinion on DL. Also Applies to Factoring

Sajjad is working in Quantum Computing so

1. He actually knows stuff.
2. His funding depends on the answer being YES Quantum is practical!

Sajjad Opinion on DL. Also Applies to Factoring

Sajjad is working in Quantum Computing so

1. He actually knows stuff.
2. His funding depends on the answer being YES Quantum is practical!

His opinion is on the next slide.

Sajjad Opinion on DL. Also Applies to Factoring

Sajjad Opinion on DL. Also Applies to Factoring

1. There has been **huge** progress in the last 10 years in error-correction.

Sajjad Opinion on DL. Also Applies to Factoring

1. There has been **huge** progress in the last 10 years in error-correction.
2. There are now real problems (sampling) that can be done by a Quantum Computer much faster than a Classical Computer. Yeah!

Sajjad Opinion on DL. Also Applies to Factoring

1. There has been **huge** progress in the last 10 years in error-correction.
2. There are now real problems (sampling) that can be done by a Quantum Computer much faster than a Classical Computer. Yeah!
3. The sampling problem is not related to problems in number theory like Discrete Log. Boo!

Sajjad Opinion on DL. Also Applies to Factoring

1. There has been **huge** progress in the last 10 years in error-correction.
2. There are now real problems (sampling) that can be done by a Quantum Computer much faster than a Classical Computer. Yeah!
3. The sampling problem is not related to problems in number theory like Discrete Log. Boo!
4. **Prediction** in ~ 25 years we will have real quantum computers that can do DL and factoring quickly. This is based on what experimentalists say (see next slide).

Sajjad Opinion on DL. Also Applies to Factoring

1. There has been **huge** progress in the last 10 years in error-correction.
2. There are now real problems (sampling) that can be done by a Quantum Computer much faster than a Classical Computer. Yeah!
3. The sampling problem is not related to problems in number theory like Discrete Log. Boo!
4. **Prediction** in ~ 25 years we will have real quantum computers that can do DL and factoring quickly. This is based on what experimentalists say (see next slide).

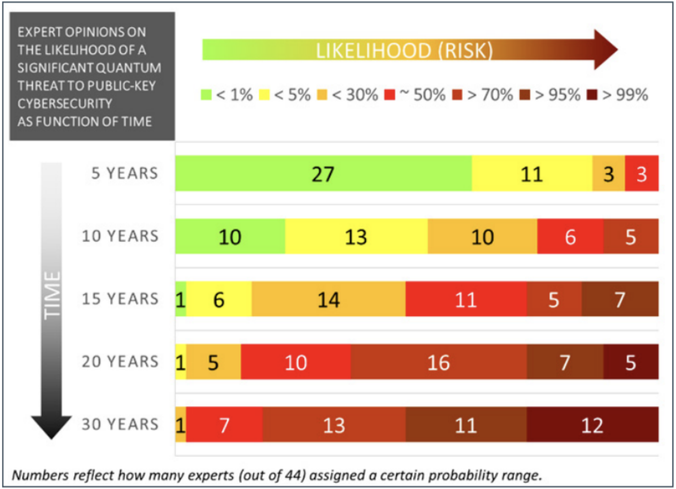
Bill Its very hard to predict things

Sajjad Opinion on DL. Also Applies to Factoring

1. There has been **huge** progress in the last 10 years in error-correction.
2. There are now real problems (sampling) that can be done by a Quantum Computer much faster than a Classical Computer. Yeah!
3. The sampling problem is not related to problems in number theory like Discrete Log. Boo!
4. **Prediction** in ~ 25 years we will have real quantum computers that can do DL and factoring quickly. This is based on what experimentalists say (see next slide).

Bill Its very hard to predict things especially about the future.

Expert Opinion As a Chart



Expert Opinions on the Technical Realization of Quantum Computers

Expert Opinion As a Paper

See also this paper: `qtime.pdf`

Discrete Log-General

Definition Let p be a prime and g be a generator mod p.
The **Discrete Log Problem:**
Given $a \in \{1, \ldots, p\}$, find x such that $g^x \equiv a \pmod{p}$. We call this $DL_{p,g}(a)$.

Discrete Log-General

Definition Let p be a prime and g be a generator mod p.
The **Discrete Log Problem:**
Given $a \in \{1, \ldots, p\}$, find x such that $g^x \equiv a \pmod{p}$. We call this $DL_{p,g}(a)$.

1. If g is small then $DL(g^a)$ might be easy: $DL_{1009,7}(49) = 2$ since $7^2 \equiv 49 \pmod{1009}$.

Discrete Log-General

Definition Let p be a prime and g be a generator mod p.
The **Discrete Log Problem:**
Given $a \in \{1, \ldots, p\}$, find x such that $g^x \equiv a \pmod{p}$. We call this $DL_{p,g}(a)$.

1. If g is small then $DL(g^a)$ might be easy: $DL_{1009,7}(49) = 2$ since $7^2 \equiv 49 \pmod{1009}$.
2. If g is small then $DL(p - g^a)$ might be easy: $DL_{1009,7}(1009 - 49) = 506$ since $7^{504}7^2 \equiv -7^2 \equiv 1009 - 49 \pmod{1009}$.

Discrete Log-General

Definition Let p be a prime and g be a generator mod p.
The **Discrete Log Problem:**
Given $a \in \{1, \ldots, p\}$, find x such that $g^x \equiv a \pmod{p}$. We call this $DL_{p,g}(a)$.

1. If g is small then $DL(g^a)$ might be easy: $DL_{1009,7}(49) = 2$ since $7^2 \equiv 49 \pmod{1009}$.
2. If g is small then $DL(p - g^a)$ might be easy: $DL_{1009,7}(1009 - 49) = 506$ since $7^{504}7^2 \equiv -7^2 \equiv 1009 - 49 \pmod{1009}$.
3. If $g, a \in \{\frac{p}{3}, \ldots, \frac{2p}{3}\}$ then problem suspected hard.

Discrete Log-General

Definition Let p be a prime and g be a generator mod p.
The **Discrete Log Problem:**
Given $a \in \{1, \ldots, p\}$, find x such that $g^x \equiv a \pmod{p}$. We call this $DL_{p,g}(a)$.

1. If g is small then $DL(g^a)$ might be easy: $DL_{1009,7}(49) = 2$ since $7^2 \equiv 49 \pmod{1009}$.
2. If g is small then $DL(p - g^a)$ might be easy: $DL_{1009,7}(1009 - 49) = 506$ since $7^{504}7^2 \equiv -7^2 \equiv 1009 - 49 \pmod{1009}$.
3. If $g, a \in \{\frac{p}{3}, \ldots, \frac{2p}{3}\}$ then problem suspected hard.
4. **Tradeoff:** By restricting a we are cutting down search space for Eve. Even so, in this case we need to since she REALLY can recognize when DL is easy.

Consider What We Already Have Here

Consider What We Already Have Here

- **Exponentiation mod p** is Easy.

Consider What We Already Have Here

- **Exponentiation mod p** is Easy.
- **Discrete Log** is thought to be Hard.

Consider What We Already Have Here

- **Exponentiation mod p** is Easy.
- **Discrete Log** is thought to be Hard.

We want a crypto system where:

Consider What We Already Have Here

- **Exponentiation mod p** is Easy.
- **Discrete Log** is thought to be Hard.

We want a crypto system where:

- Alice and Bob do **Exponentiation mod p** to encrypt and decrypt.

Consider What We Already Have Here

- **Exponentiation mod p** is Easy.
- **Discrete Log** is thought to be Hard.

We want a crypto system where:

- Alice and Bob do **Exponentiation mod p** to encrypt and decrypt.
- Eve has to do **Discrete Log** to crack it.

Consider What We Already Have Here

- **Exponentiation mod p** is Easy.
- **Discrete Log** is thought to be Hard.

We want a crypto system where:

- Alice and Bob do **Exponentiation mod p** to encrypt and decrypt.
- Eve has to do **Discrete Log** to crack it.

Do we have this?

Consider What We Already Have Here

- **Exponentiation mod p** is Easy.
- **Discrete Log** is thought to be Hard.

We want a crypto system where:

- Alice and Bob do **Exponentiation mod p** to encrypt and decrypt.
- Eve has to do **Discrete Log** to crack it.

Do we have this?

No. But we'll come close.

Convention

For the rest of the slides on **Diffie-Hellman Key Exchange** there will always be a prime p that we are considering.

ALL math done from that point on is mod p.

ALL numbers are in $\{1, \ldots, p-1\}$.

Finding Generators

Finding Gens; How Many Gens Are There?

Problem Given p, find g such that

- g generates $\mathbb{Z}_p^*$.
- $g \in \{\frac{p}{3}, \ldots, \frac{2p}{3}\}$. (We ignore floors and ceilings for notational convenience.)

Finding Gens; How Many Gens Are There?

Problem Given p, find g such that

- g generates $\mathbb{Z}_p^*$.
- $g \in \{\frac{p}{3}, \ldots, \frac{2p}{3}\}$. (We ignore floors and ceilings for notational convenience.)

We could test $\frac{p}{3}$, then $\frac{p}{3} + 1$, etc. Will we hit a generator soon?

Finding Gens; How Many Gens Are There?

Problem Given p, find g such that

- g generates $\mathbb{Z}_p^*$.
- $g \in \{\frac{p}{3}, \ldots, \frac{2p}{3}\}$. (We ignore floors and ceilings for notational convenience.)

We could test $\frac{p}{3}$, then $\frac{p}{3} + 1$, etc. Will we hit a generator soon?

How many elts of $\{1, \ldots, p-1\}$ are gens?

Finding Gens; How Many Gens Are There?

Problem Given p, find g such that

- g generates $\mathbb{Z}_p^*$.
- $g \in \{\frac{p}{3}, \ldots, \frac{2p}{3}\}$. (We ignore floors and ceilings for notational convenience.)

We could test $\frac{p}{3}$, then $\frac{p}{3}+1$, etc. Will we hit a generator soon?

How many elts of $\{1, \ldots, p-1\}$ are gens? $\Theta(\frac{p}{\log \log p})$

Finding Gens; How Many Gens Are There?

Problem Given p, find g such that

- g generates $\mathbb{Z}_p^*$.
- $g \in \{\frac{p}{3}, \ldots, \frac{2p}{3}\}$. (We ignore floors and ceilings for notational convenience.)

We could test $\frac{p}{3}$, then $\frac{p}{3} + 1$, etc. Will we hit a generator soon?

How many elts of $\{1, \ldots, p-1\}$ are gens? $\Theta(\frac{p}{\log \log p})$

Hence if you just look for a gen you will find one soon.

Finding Gens: First Attempt

Given prime p, find a gen for $\mathbb{Z}_p^*$

Finding Gens: First Attempt

Given prime p, find a gen for $\mathbb{Z}_p^*$

1. Input p.

Finding Gens: First Attempt

Given prime p, find a gen for $\mathbb{Z}_p^*$

1. Input p.
2. For $g = \frac{p}{3}$ to $\frac{2p}{3}$:
   Compute $g^1, g^2, \ldots, g^{p-1}$ until either hit a repeat or finish. If repeats then g is NOT a generator, so goto the next g. If finishes then output g and stop.

PRO You will find a gen fairly soon.

Finding Gens: First Attempt

Given prime p, find a gen for $\mathbb{Z}_p^*$

1. Input p.
2. For $g = \frac{p}{3}$ to $\frac{2p}{3}$:
   Compute $g^1, g^2, \ldots, g^{p-1}$ until either hit a repeat or finish. If repeats then g is NOT a generator, so goto the next g. If finishes then output g and stop.

PRO You will find a gen fairly soon.
CON Computing $g^1, \ldots, g^{p-1}$ is $O(p \log p)$ operations.

Finding Gens: First Attempt

Given prime p, find a gen for $\mathbb{Z}_p^*$

1. Input p.
2. For $g = \frac{p}{3}$ to $\frac{2p}{3}$:
   Compute $g^1, g^2, \ldots, g^{p-1}$ until either hit a repeat or finish. If repeats then g is NOT a generator, so goto the next g. If finishes then output g and stop.

PRO You will find a gen fairly soon.
CON Computing $g^1, \ldots, g^{p-1}$ is $O(p \log p)$ operations.
Bad! Recall $(\log p)^{O(1)}$ is fast, $O(p)$ is slow.

Finding Gens: Second Attempt

Theorem: If g is **not** a generator then there exists x that (1) x divides $p-1$, (2) $x \neq p-1$, and (3) $g^x \equiv 1$.

Given prime p, find a gen for $\mathbb{Z}_p^*$

Finding Gens: Second Attempt

Theorem: If g is **not** a generator then there exists x that (1) x divides $p-1$, (2) $x \neq p-1$, and (3) $g^x \equiv 1$.

Given prime p, find a gen for $\mathbb{Z}_p^*$

1. Input p.

Finding Gens: Second Attempt

Theorem: If g is **not** a generator then there exists x that (1) x divides $p-1$, (2) $x \neq p-1$, and (3) $g^x \equiv 1$.

Given prime p, find a gen for $\mathbb{Z}_p^*$

1. Input p.
2. Factor $p-1$. Let F be the set of its factors except $p-1$.

Finding Gens: Second Attempt

Theorem: If g is **not** a generator then there exists x that (1) x divides $p-1$, (2) $x \neq p-1$, and (3) $g^x \equiv 1$.

Given prime p, find a gen for $\mathbb{Z}_p^*$

1. Input p.
2. Factor $p-1$. Let F be the set of its factors except $p-1$.
3. For $g = \frac{p}{3}$ to $\frac{2p}{3}$:
   Compute g^x for all $x \in F$. If any $= 1$ then g **not** *generator.*
   If none are 1 then output g and stop.

Finding Gens: Second Attempt

Theorem: If g is **not** a generator then there exists x that (1) x divides $p-1$, (2) $x \neq p-1$, and (3) $g^x \equiv 1$.

Given prime p, find a gen for $\mathbb{Z}_p^*$

1. Input p.
2. Factor $p-1$. Let F be the set of its factors except $p-1$.
3. For $g = \frac{p}{3}$ to $\frac{2p}{3}$:
   Compute g^x for all $x \in F$. If any $= 1$ then g **not** *generator.*
   If none are 1 then output g and stop.

Is this a good algorithm?

Finding Gens: Second Attempt

Theorem: If g is **not** a generator then there exists x that (1) x divides $p-1$, (2) $x \neq p-1$, and (3) $g^x \equiv 1$.

Given prime p, find a gen for $\mathbb{Z}_p^*$

1. Input p.
2. Factor $p-1$. Let F be the set of its factors except $p-1$.
3. For $g = \frac{p}{3}$ to $\frac{2p}{3}$:
   Compute g^x for all $x \in F$. If any $= 1$ then g **not** *generator.*
   If none are 1 then output g and stop.

Is this a good algorithm?
Time Every iteration takes $O(|F|(\log p))$ ops. $|F|$ might be huge!
So no good. But wait for next slide...

Finding Gens: Second Attempt

Theorem: If g is **not** a generator then there exists x that (1) x divides $p-1$, (2) $x \neq p-1$, and (3) $g^x \equiv 1$.

Given prime p, find a gen for $\mathbb{Z}_p^*$

1. Input p.
2. Factor $p-1$. Let F be the set of its factors except $p-1$.
3. For $g = \frac{p}{3}$ to $\frac{2p}{3}$:
   Compute g^x for all $x \in F$. If any $= 1$ then g **not** *generator.*
   If none are 1 then output g and stop.

Is this a good algorithm?
Time Every iteration takes $O(|F|(\log p))$ ops. $|F|$ might be huge!
So no good. But wait for next slide. . ..
BIG CON: Factoring $p-1$?

Finding Gens: Second Attempt

Theorem: If g is **not** a generator then there exists x that (1) x divides $p-1$, (2) $x \neq p-1$, and (3) $g^x \equiv 1$.

Given prime p, find a gen for $\mathbb{Z}_p^*$

1. Input p.
2. Factor $p-1$. Let F be the set of its factors except $p-1$.
3. For $g = \frac{p}{3}$ to $\frac{2p}{3}$:
   Compute g^x for all $x \in F$. If any $= 1$ then g **not** *generator.*
   If none are 1 then output g and stop.

Is this a good algorithm?
Time Every iteration takes $O(|F|(\log p))$ ops. $|F|$ might be huge!
So no good. But wait for next slide. . ..
BIG CON: Factoring $p-1$? **Really?**

Finding Gens: Second Attempt

Theorem: If g is **not** a generator then there exists x that (1) x divides $p-1$, (2) $x \neq p-1$, and (3) $g^x \equiv 1$.

Given prime p, find a gen for $\mathbb{Z}_p^*$

1. Input p.
2. Factor $p-1$. Let F be the set of its factors except $p-1$.
3. For $g = \frac{p}{3}$ to $\frac{2p}{3}$:
   Compute g^x for all $x \in F$. If any $= 1$ then g **not** *generator.*
   If none are 1 then output g and stop.

Is this a good algorithm?
Time Every iteration takes $O(|F|(\log p))$ ops. $|F|$ might be huge!
So no good. But wait for next slide. . ..
BIG CON: Factoring $p-1$? **Really?**
Borrow Sajjad's Quantum Computer?

Factoring is Hard. Or is it?

Second Attempt had two problems:

Factoring is Hard. Or is it?

Second Attempt had two problems:

1. Factoring is hard.

Factoring is Hard. Or is it?

Second Attempt had two problems:

1. Factoring is hard.
2. $p - 1$ may have many factors.

Factoring is Hard. Or is it?

Second Attempt had two problems:

1. Factoring is hard.
2. $p-1$ may have many factors.

We want $p-1$ to be easy to factor and have few factors.

Factoring is Hard. Or is it?

Second Attempt had two problems:

1. Factoring is hard.
2. $p - 1$ may have many factors.

We want $p - 1$ to be easy to factor and have few factors.

There are three kinds of people in the world:

Factoring is Hard. Or is it?

Second Attempt had two problems:

1. Factoring is hard.
2. $p - 1$ may have many factors.

We want $p - 1$ to be easy to factor and have few factors.

There are three kinds of people in the world:

1. Those who **make** things happen.

Factoring is Hard. Or is it?

Second Attempt had two problems:

1. Factoring is hard.
2. $p - 1$ may have many factors.

We want $p - 1$ to be easy to factor and have few factors.

There are three kinds of people in the world:

1. Those who **make** things happen.
2. Those who **watch** things happen.

Factoring is Hard. Or is it?

Second Attempt had two problems:

1. Factoring is hard.
2. $p - 1$ may have many factors.

We want $p - 1$ to be easy to factor and have few factors.

There are three kinds of people in the world:

1. Those who **make** things happen.
2. Those who **watch** things happen.
3. Those who **wonder** what happened.

Factoring is Hard. Or is it?

Second Attempt had two problems:

1. Factoring is hard.
2. $p - 1$ may have many factors.

We want $p - 1$ to be easy to factor and have few factors.

There are three kinds of people in the world:

1. Those who **make** things happen.
2. Those who **watch** things happen.
3. Those who **wonder** what happened.

We will **make things happen.**

Factoring is Hard. Or is it?

Second Attempt had two problems:

1. Factoring is hard.
2. $p - 1$ may have many factors.

We want $p - 1$ to be easy to factor and have few factors.

There are three kinds of people in the world:

1. Those who **make** things happen.
2. Those who **watch** things happen.
3. Those who **wonder** what happened.

We will **make things happen.**
We will **make $p - 1$ easy to factor.**

Factoring is Hard. Or is it?

Second Attempt had two problems:

1. Factoring is hard.
2. $p - 1$ may have many factors.

We want $p - 1$ to be easy to factor and have few factors.

There are three kinds of people in the world:

1. Those who **make** things happen.
2. Those who **watch** things happen.
3. Those who **wonder** what happened.

We will **make things happen.**
We will **make $p - 1$ easy to factor.**
We will **make $p - 1$ have few factors.**

Finding Gens: Third Attempt

Idea: Pick p such that $p - 1 = 2q$ where q is prime.

Finding Gens: Third Attempt

Idea: Pick p such that $p - 1 = 2q$ where q is prime.
Given prime p, find a gen for $\mathbb{Z}_p^*$

Finding Gens: Third Attempt

Idea: Pick p such that $p-1=2q$ where q is prime.

Given prime p, find a gen for $\mathbb{Z}_p^*$

1. Input p a prime such that $p-1=2q$ where q is prime. (We later explore how we can find such a prime.)

Finding Gens: Third Attempt

Idea: Pick p such that $p-1=2q$ where q is prime.
Given prime p, find a gen for $\mathbb{Z}_p^*$

1. Input p a prime such that $p-1=2q$ where q is prime. (We later explore how we can find such a prime.)
2. Factor $p-1$. Let F be the set of its factors except $p-1$. That's EASY: $F=\{2,q\}$.

Finding Gens: Third Attempt

Idea: Pick p such that $p-1=2q$ where q is prime.

Given prime p, find a gen for $\mathbb{Z}_p^*$

1. Input p a prime such that $p-1=2q$ where q is prime. (We later explore how we can find such a prime.)
2. Factor $p-1$. Let F be the set of its factors except $p-1$. That's EASY: $F=\{2,q\}$.
3. For $g=\frac{p}{3}$ to $\frac{2p}{3}$:
   Compute g^x for all $x \in F$. If any $=1$ then g NOT generator. If none are 1 then output g and stop.

Finding Gens: Third Attempt

Idea: Pick p such that $p-1=2q$ where q is prime.
Given prime p, find a gen for $\mathbb{Z}_p^*$

1. Input p a prime such that $p-1=2q$ where q is prime. (We later explore how we can find such a prime.)
2. Factor $p-1$. Let F be the set of its factors except $p-1$. That's EASY: $F=\{2,q\}$.
3. For $g=\frac{p}{3}$ to $\frac{2p}{3}$:
   Compute g^x for all $x \in F$. If any $=1$ then g NOT generator. If none are 1 then output g and stop.

Is this a good algorithm?
PRO Every iteration does $O(\log p)$ operations.

Finding Gens: Third Attempt

Idea: Pick p such that $p-1=2q$ where q is prime.
Given prime p, find a gen for $\mathbb{Z}_p^*$

1. Input p a prime such that $p-1=2q$ where q is prime. (We later explore how we can find such a prime.)
2. Factor $p-1$. Let F be the set of its factors except $p-1$. That's EASY: $F=\{2,q\}$.
3. For $g=\frac{p}{3}$ to $\frac{2p}{3}$:
   Compute g^x for all $x \in F$. If any $=1$ then g NOT generator. If none are 1 then output g and stop.

Is this a good algorithm?
PRO Every iteration does $O(\log p)$ operations.
CON: Need both p and $\frac{p-1}{2}$ are primes.

Finding Gens: Third Attempt

Idea: Pick p such that $p-1=2q$ where q is prime.
Given prime p, find a gen for $\mathbb{Z}_p^*$

1. Input p a prime such that $p-1=2q$ where q is prime. (We later explore how we can find such a prime.)
2. Factor $p-1$. Let F be the set of its factors except $p-1$. That's EASY: $F=\{2,q\}$.
3. For $g=\frac{p}{3}$ to $\frac{2p}{3}$:
   Compute g^x for all $x \in F$. If any $=1$ then g NOT generator. If none are 1 then output g and stop.

Is this a good algorithm?
PRO Every iteration does $O(\log p)$ operations.
CON: Need both p and $\frac{p-1}{2}$ are primes.
CAVEAT We need to pick certain kinds of primes. **Can** do that!

BILL, STOP RECORDING LECTURE!!!!

BILL STOP RECORDING LECTURE!!!