CMSC/Math 456: Cryptography (Fall 2023)

Lecture 12

Daniel Gottesman

Administrative

Problem set #5 should have been turned in. Problem set #6 is now available on the course web page, and solution set #4 is on ELMS.

Midterm: Thursday, Oct. 19 (2 weeks from today)

- In class
- Open book (including textbook), no electronic devices
- Will cover classical cryptographic, private key encryption, modular arithmetic, and public key exchange (probably not public key encryption).
- Those with accommodations remember to book with ADS.

Two Questions

Question: How is a cryptographer like a magician?

Two Questions

Question: How is a cryptographer like a magician?

Answer: A cryptographer never reveals their secrets.

Two Questions

Question: How is a cryptographer like a magician?

Answer: A cryptographer never reveals their secrets.

Question: How is a cryptographer *not* like a magician?

Two Questions

Question: How is a cryptographer like a magician?

Answer: A cryptographer never reveals their secrets.

Question: How is a cryptographer *not* like a magician?

Answer: A cryptographer will tell you how they did it.

Two Questions

Question: How is a cryptographer like a magician?

Answer: A cryptographer never reveals their secrets.

Question: How is a cryptographer *not* like a magician?

Answer: A cryptographer will tell you how they did it.

Let us now perform a cryptographic magic trick:

Two Questions

Question: How is a cryptographer like a magician?

Answer: A cryptographer never reveals their secrets.

Question: How is a cryptographer *not* like a magician?

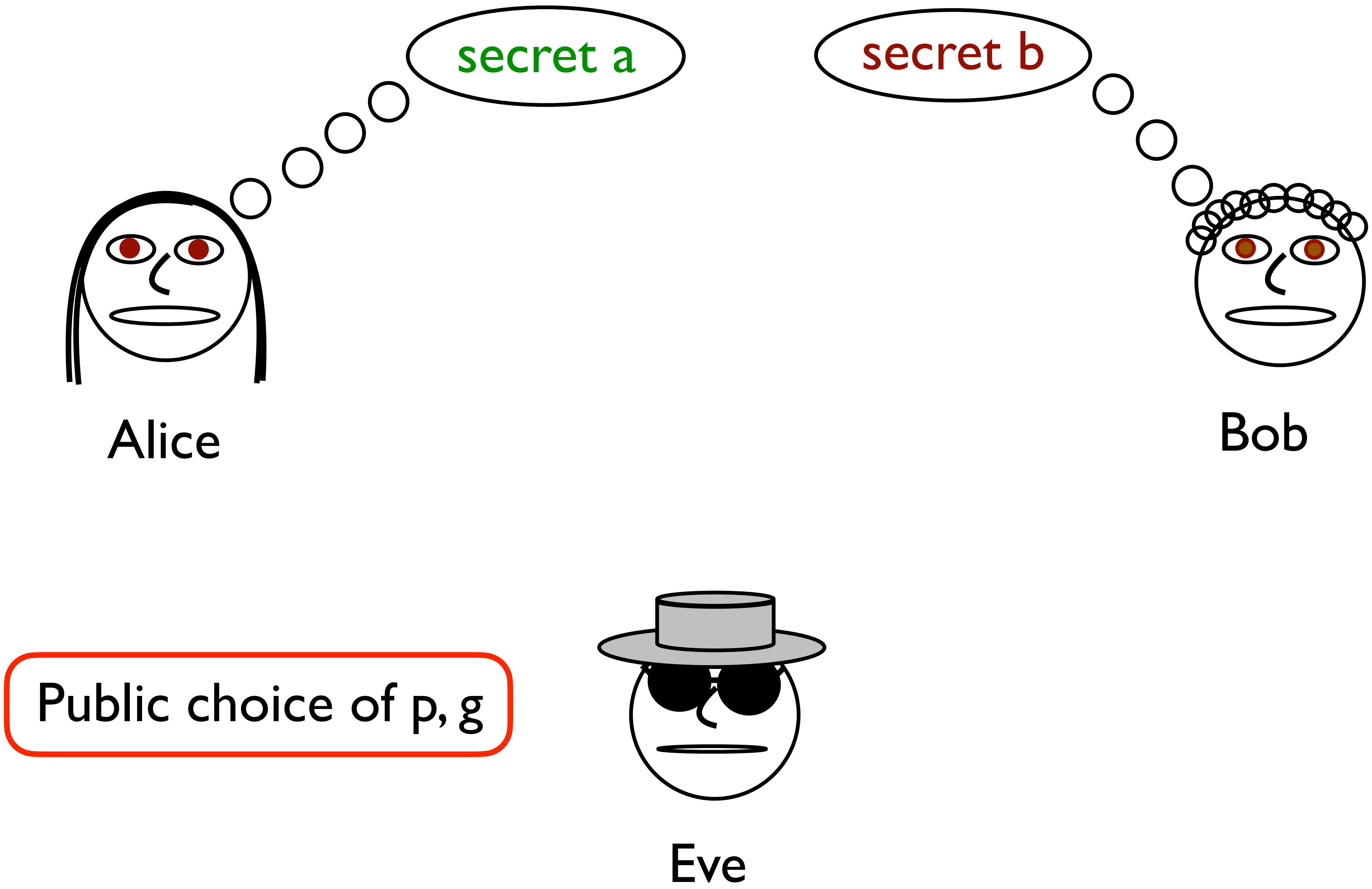
Answer: A cryptographer will tell you how they did it.

Let us now perform a cryptographic magic trick:

Alice and Bob will generate a shared secret key using only public communication!

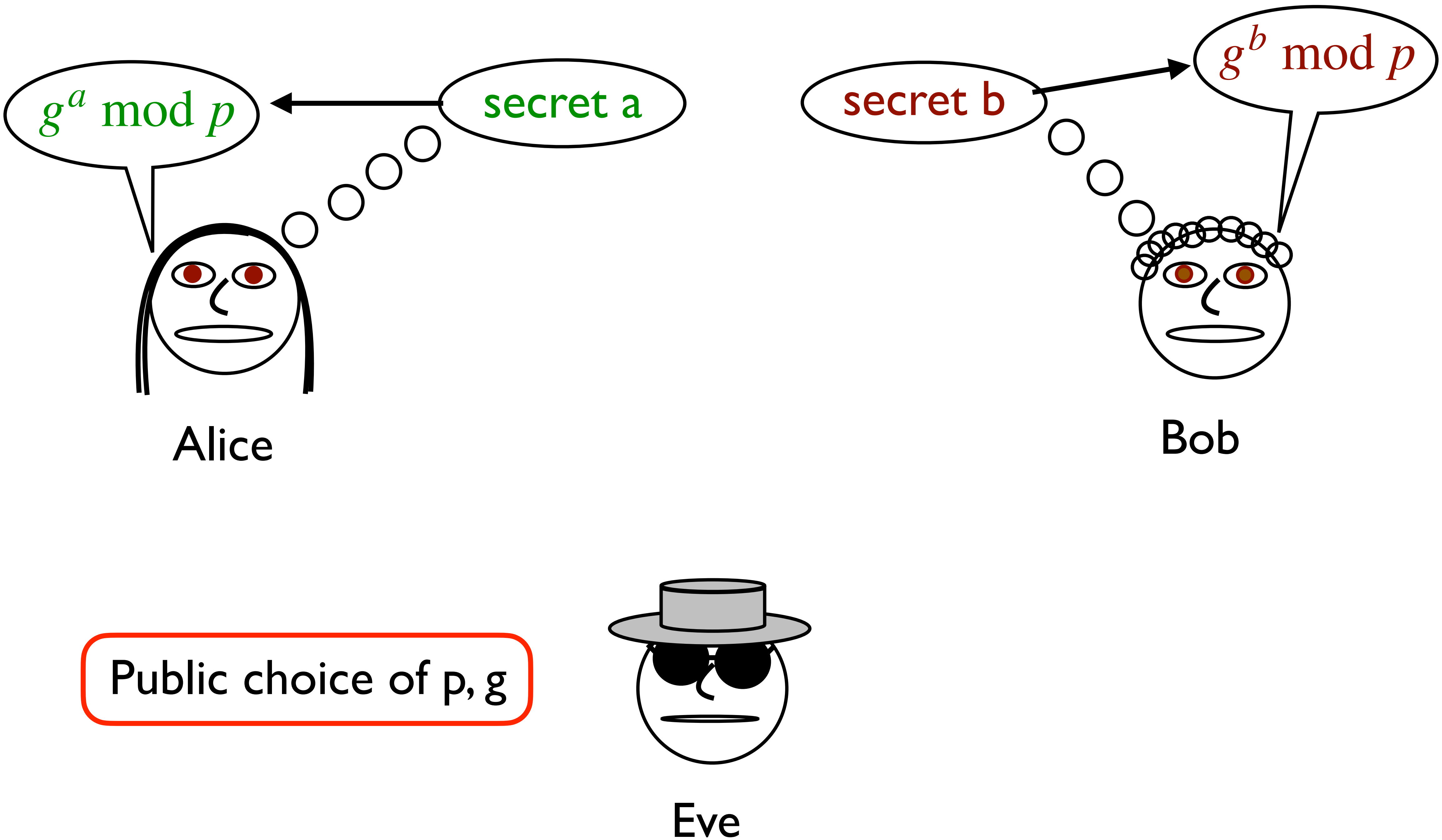
This class is being recorded

Diffie-Hellman Key Exchange



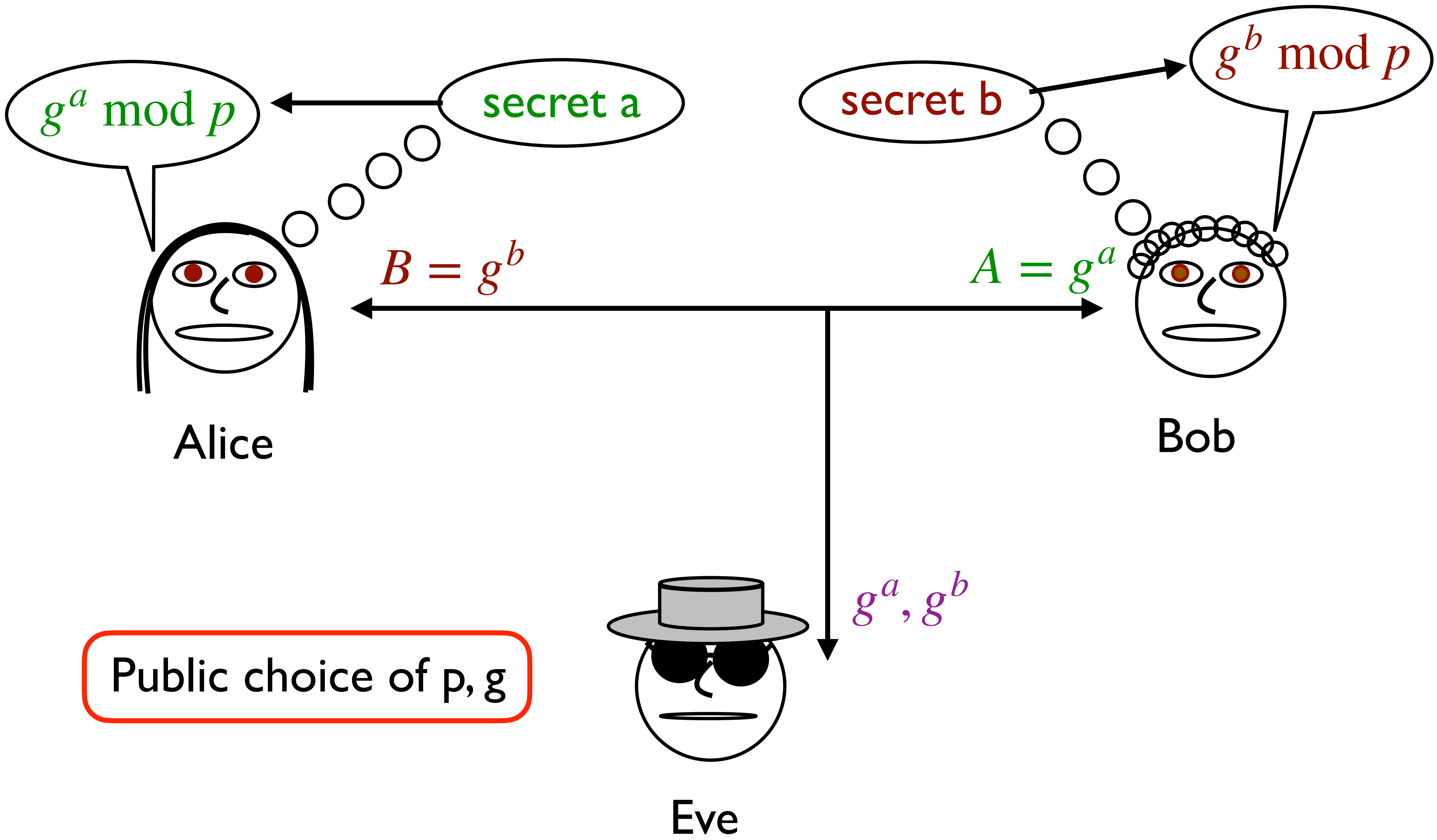
This class is being recorded

Diffie-Hellman Key Exchange



This class is being recorded

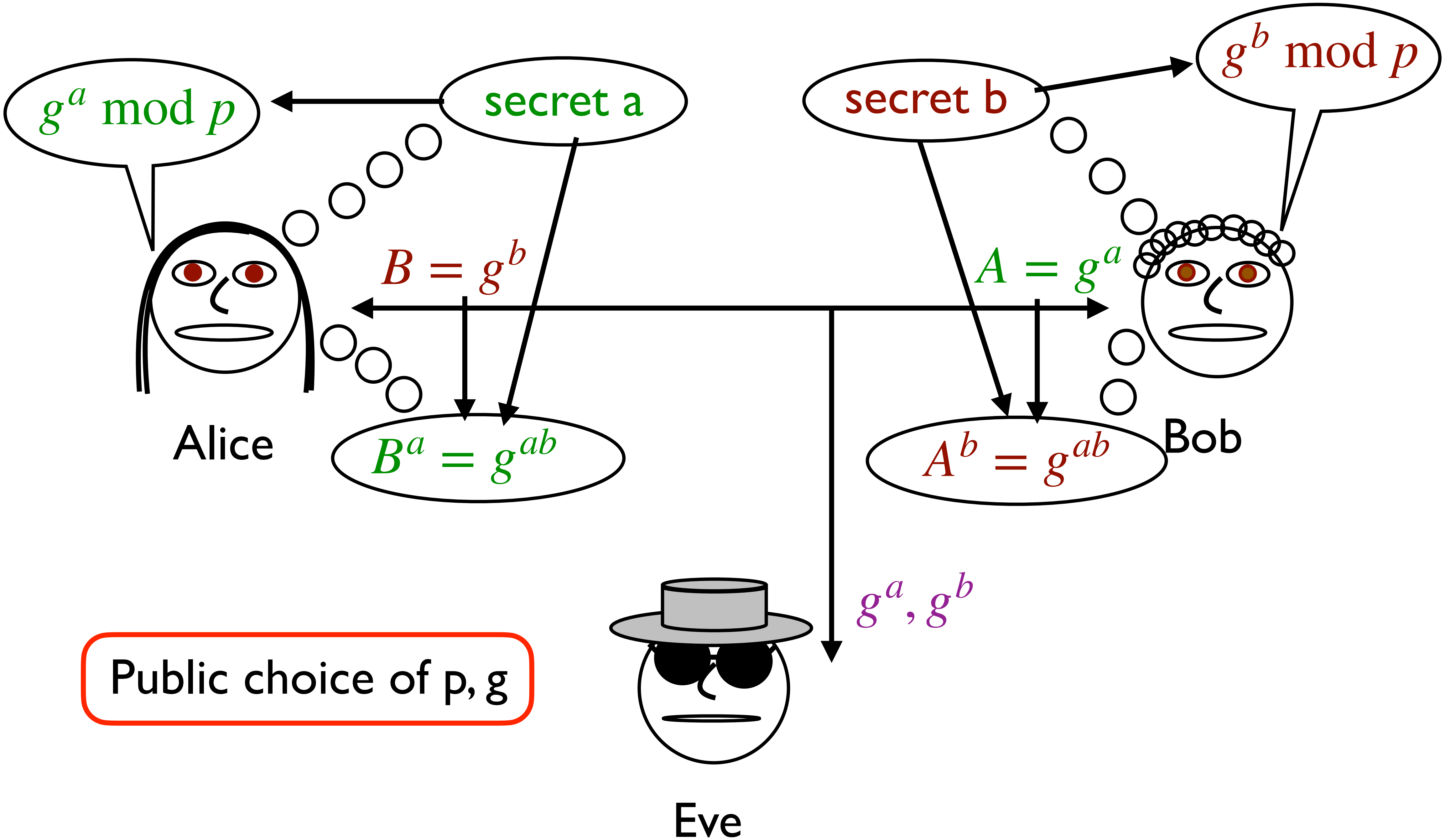
Diffie-Hellman Key Exchange



Public choice of p, g

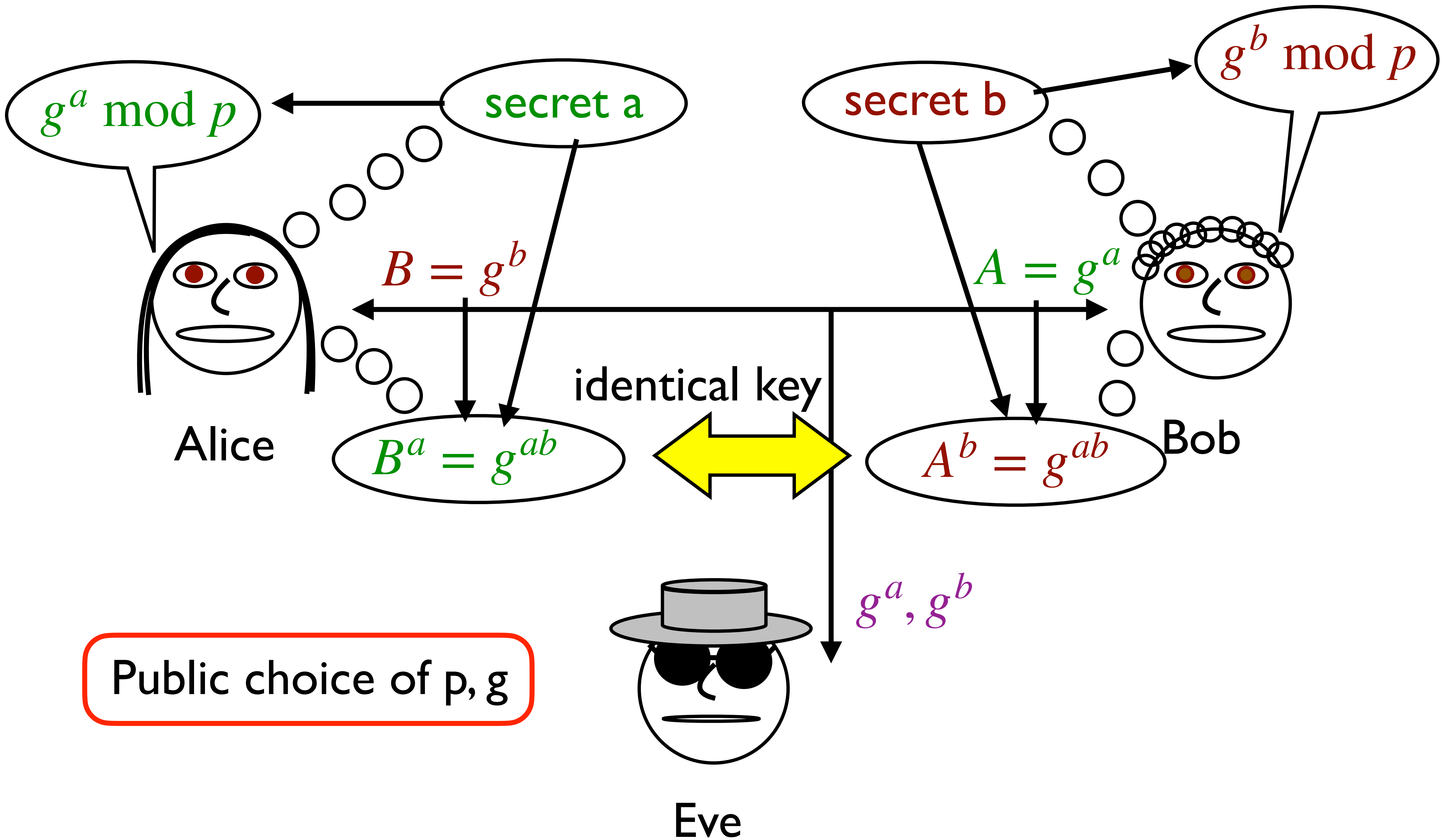
This class is being recorded

Diffie-Hellman Key Exchange



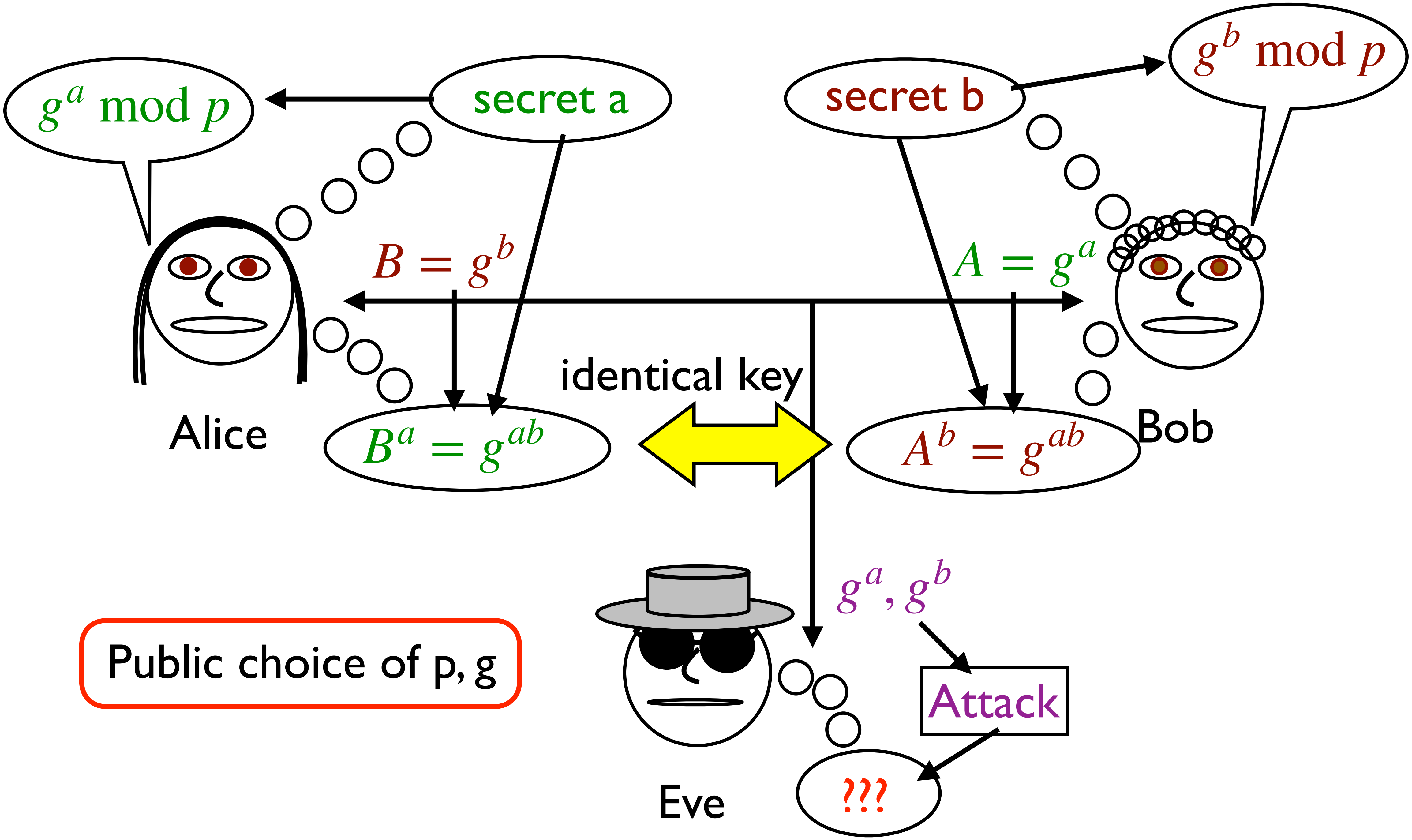
This class is being recorded

Diffie-Hellman Key Exchange



This class is being recorded

Diffie-Hellman Key Exchange



This class is being recorded

Diffie-Hellman Security Idea

In Diffie-Hellman, Alice and Bob must perform modular exponentiation: Alice announces $A = g^a \bmod p$ and Bob announces $B = g^b \bmod p$ for secret a and b chosen by Alice and Bob respectively and not shared with each other or Eve. Then they do another pair of modular exponentiations B^a and A^b to calculate the key.

- Alice and Bob can compute modular exponentials efficiently.

Eve can break Diffie-Hellman if she can calculate the discrete log for (g,p): That is, if given A, she can find a such that $g^a = A \bmod p$.

- But we believe that computing the discrete log is hard. Thus, Eve cannot learn a or b to help her find $g^{ab} \bmod p$.

Diffie-Hellman Magic Demonstration

We will use $p = 71$ and $g = 65$. Note that p is prime and g has order 70.

Diffie-Hellman Magic Demonstration

We will use p = 71 and g = 65. Note that p is prime and g has order 70.

Alice: Choose & record a secret number a from 2 to 70. Compute

$$A = 65^a \bmod 71$$

Bob: Choose &record a secret number b from 2 to 70. Compute

$$B = 65^b \bmod 71$$

Diffie-Hellman Magic Demonstration

We will use p = 71 and g = 65. Note that p is prime and g has order 70.

Alice: Choose & record a secret number a from 2 to 70. Compute

$$A = 65^a \bmod 71$$

Bob: Choose &record a secret number b from 2 to 70. Compute

$$B = 65^b \bmod 71$$

Alice and Bob: announce A and B to the class.

Diffie-Hellman Magic Demonstration

We will use p = 71 and g = 65. Note that p is prime and g has order 70.

Alice: Choose & record a secret number a from 2 to 70. Compute

$$A = 65^a \bmod 71$$

Bob: Choose &record a secret number b from 2 to 70. Compute

$$B = 65^b \bmod 71$$

Alice and Bob: announce A and B to the class.

Alice: Compute $B^a \bmod 71$ and write it down secretly.

Bob: Compute $A^b \bmod 71$ and write it down secretly.

Do not reveal them until I say to.

Bad Primes for Discrete Log

How did that attack work? Ideas?

Bad Primes for Discrete Log

How did that attack work? Ideas?

Maybe p was not big enough.

Bad Primes for Discrete Log

How did that attack work? Ideas?

Maybe p was not big enough.

Indeed, 71 is not very big. I could have pre-computed all powers $65^a \bmod 71$.

But that is not what I did.

Bad Primes for Discrete Log

How did that attack work? Ideas?

Maybe p was not big enough.

Indeed, 71 is not very big. I could have pre-computed all powers $65^a \bmod 71$.

But that is not what I did.

I did precompute a few powers, and I used the fact that $p - 1 = 70 = 2 \cdot 5 \cdot 7$, and 2, 5, and 7 are all small primes.

Bad Primes for Discrete Log

How did that attack work? Ideas?

Maybe p was not big enough.

Indeed, 71 is not very big. I could have pre-computed all powers $65^a \bmod 71$.

But that is not what I did.

I did precompute a few powers, and I used the fact that $p - 1 = 70 = 2 \cdot 5 \cdot 7$, and 2, 5, and 7 are all small primes.

Hint: Our goal is to find a mod 70, since the powers repeat after the order of g.

Bad Primes for Discrete Log

How did that attack work? Ideas?

Maybe p was not big enough.

Indeed, 71 is not very big. I could have pre-computed all powers $65^a \bmod 71$.

But that is not what I did.

I did precompute a few powers, and I used the fact that $p - 1 = 70 = 2 \cdot 5 \cdot 7$, and 2, 5, and 7 are all small primes.

Hint: Our goal is to find a mod 70, since the powers repeat after the order of g.

Hint: When I asked Alice to compute some additional modular exponentials, that was telling me the values of a mod 2, mod 5, and mod 7. How? And why does that help?

Bad Primes for Discrete Log

How did that attack work? Ideas?

Maybe p was not big enough.

Indeed, 71 is not very big. I could have pre-computed all powers $65^a \bmod 71$.

But that is not what I did.

I did precompute a few powers, and I used the fact that $p-1 = 70 = 2 \cdot 5 \cdot 7$, and 2, 5, and 7 are all small primes.

Hint: Our goal is to find a mod 70, since the powers repeat after the order of g.

Hint: When I asked Alice to compute some additional modular exponentials, that was telling me the values of a mod 2, mod 5, and mod 7. How? And why does that help?

Hint: What is the order of A^{10}? How is it related to g^{10}?

Pohlig-Hellman Algorithm

When p-1 is itself a product of small primes, there is a fast algorithm for discrete log (Pohlig-Hellman).

> We are given p, g, with p-1 a product of small primes, g a generator, and we are also given A. We wish to find a such that $g^a = A \bmod p$.

Pohlig-Hellman Algorithm

When p-1 is itself a product of small primes, there is a fast algorithm for discrete log (Pohlig-Hellman).

> We are given p, g, with p-1 a product of small primes, g a generator, and we are also given A. We wish to find a such that $g^a = A \bmod p$.

Suppose $p - 1 = \prod_i p_i$.

Pohlig-Hellman Algorithm

When p-1 is itself a product of small primes, there is a fast algorithm for discrete log (Pohlig-Hellman).

> We are given p, g, with p-1 a product of small primes, g a generator, and we are also given A. We wish to find a such that $g^a = A \bmod p$.

Suppose $p - 1 = \prod_i p_i$. $\quad \frac{p-1}{p_i} = p_1 p_2 \cdots p_{i-1} p_{i+1} \cdots p_m$

Note: What is the order of $g_i = g^{(p-1)/p_i} \bmod p$?

Pohlig-Hellman Algorithm

When p-1 is itself a product of small primes, there is a fast algorithm for discrete log (Pohlig-Hellman).

We are given p, g, with p-1 a product of small primes, g a generator, and we are also given A. We wish to find a such that $g^a = A \bmod p$.

Suppose $p - 1 = \prod_i p_i$.

$$\frac{p-1}{p_i} = p_1 p_2 \cdots p_{i-1} p_{i+1} \cdots p_m$$

Note: What is the order of $g_i = g^{(p-1)/p_i} \bmod p$?

$(p-1)/p_i$ is a factor of $\operatorname{ord}(g) = p - 1$, so $\gcd(p-1, (p-1)/p_i) = (p-1)/p_i$.

Pohlig-Hellman Algorithm

When p-1 is itself a product of small primes, there is a fast algorithm for discrete log (Pohlig-Hellman).

> We are given p, g, with p-1 a product of small primes, g a generator, and we are also given A. We wish to find a such that $g^a = A \bmod p$.

Suppose $p - 1 = \prod_i p_i$.

$$\frac{p-1}{p_i} = p_1 p_2 \cdots p_{i-1} p_{i+1} \cdots p_m$$

Note: What is the order of $g_i = g^{(p-1)/p_i} \bmod p$?

> $(p-1)/p_i$ is a factor of $\mathrm{ord}(g) = p - 1$, so $\gcd(p-1, (p-1)/p_i) = (p-1)/p_i$.

Thus, $\mathrm{ord}(g_i) = \dfrac{p-1}{(p-1)/p_i} = p_i$.

Pohlig-Hellman Algorithm

Given p, g, A. We wish to find a such that $g^a = A \bmod p$.

We have $p - 1 = \prod_i p_i$ and $g_i = g^{(p-1)/p_i} \bmod p$ has order p_i.

This class is being recorded

Pohlig-Hellman Algorithm

Given p, g, A. We wish to find a such that $g^a = A \bmod p$.

We have $p - 1 = \prod_i p_i$ and $g_i = g^{(p-1)/p_i} \bmod p$ has order p_i.

Note: $A^{(p-1)/p_i} = (g^a)^{(p-1)/p_i} = (g^{(p-1)/p_i})^a = g_i^a \bmod p$

This class is being recorded

Pohlig-Hellman Algorithm

Given p, g, A. We wish to find a such that $g^a = A \bmod p$.

We have $p - 1 = \prod_i p_i$ and $g_i = g^{(p-1)/p_i} \bmod p$ has order p_i.

Note: $A^{(p-1)/p_i} = (g^a)^{(p-1)/p_i} = (g^{(p-1)/p_i})^a = g_i^a \bmod p$

Since g_i has order p_i, $g_i^a = g_i^{a_i} \bmod p$, where

$$a_i = a \bmod p_i$$

This class is being recorded

Pohlig-Hellman Algorithm

Given p, g, A. We wish to find a such that $g^a = A \bmod p$.

We have $p - 1 = \prod_i p_i$ and $g_i = g^{(p-1)/p_i} \bmod p$ has order p_i.

Note: $A^{(p-1)/p_i} = (g^a)^{(p-1)/p_i} = (g^{(p-1)/p_i})^a = g_i^a \bmod p$

Since g_i has order p_i, $g_i^a = g_i^{a_i} \bmod p$, where

$$a_i = a \bmod p_i$$

But p_i is small. We can easily precompute all powers

$g_i^0, g_i^1, g_i^2, \ldots g_i^{p_i - 1}$ (all mod p)

and compare them with $A^{(p-1)/p_i}$. This tells us a_i.

Pohlig-Hellman Algorithm

Given p, g, A. We wish to find a such that $g^a = A \bmod p$.

We have $p - 1 = \prod_i p_i$ and $g_i = g^{(p-1)/p_i} \bmod p$ has order p_i.

Note: $A^{(p-1)/p_i} = (g^a)^{(p-1)/p_i} = (g^{(p-1)/p_i})^a = g_i^a \bmod p$

Since g_i has order p_i, $g_i^a = g_i^{a_i} \bmod p$, where

$$a_i = a \bmod p_i$$

But p_i is small. We can easily precompute all powers

$g_i^0, g_i^1, g_i^2, \ldots g_i^{p_i - 1}$ (all mod p)

and compare them with $A^{(p-1)/p_i}$. This tells us a_i.

With all a_i, we can find a using the Chinese remainder theorem.

Pohlig-Hellman Summary

Given p, g, A. We wish to find a such that $g^a = A \bmod p$.

precompute

1. Write $p - 1 = \prod_i p_i$.
2. Compute $g_i = g^{(p-1)/p_i} \bmod p$.
3. Compute $g_i^j \bmod p$ for all $j = 0, \ldots p_i - 1$.
4. Receive A.
5. For each i, compute $A^{(p-1)/p_i}$ and find a_i such that $A^{(p-1)/p_i} = g_i^{a_i} \bmod p$.
6. Use the Chinese remainder theorem to find a such that $a_i = a \bmod p_i$ for all i.

Discrete Log Example

Example: $g = 65$, $p = 71$.

Precompute phase:

Discrete Log Example

Example: g = 65, p = 71.

Precompute phase:

1. $p - 1 = 2 \cdot 5 \cdot 7, p_1 = 7, p_2 = 5, p_3 = 2$.

Discrete Log Example

Example: g = 65, p = 71.

Precompute phase:

1. $p - 1 = 2 \cdot 5 \cdot 7, p_1 = 7, p_2 = 5, p_3 = 2$.

2. Compute $g_i = g^{(p-1)/p_i} \bmod p$:

$$65^{10} = 20 \bmod 71$$

$$65^{14} = 5 \bmod 71$$

$$65^{35} = 70 \bmod 71$$

Discrete Log Example

Example: g = 65, p = 71.

Precompute phase:

1. $p - 1 = 2 \cdot 5 \cdot 7, p_1 = 7, p_2 = 5, p_3 = 2.$

2. Compute $g_i = g^{(p-1)/p_i} \bmod p$:

$$65^{10} = 20 \bmod 71$$

$$65^{14} = 5 \bmod 71$$

$$65^{35} = 70 \bmod 71$$

3. Compute powers of 20, 5, and 70 mod 71. E.g.:

$$20^3 = 48 \bmod 71$$

$$5^2 = 25 \bmod 71$$

$$\vdots$$

Discrete Log Example

4. Receive A = 54. Want to find a such that $65^a = 54 \bmod 71$

Discrete Log Example

4. Receive A = 54. Want to find a such that $65^a = 54 \bmod 71$

5. For each i, compute $A^{(p-1)/p_i}$ and find a_i such that $A^{(p-1)/p_i} = g_i^{a_i} \bmod p$:

Discrete Log Example

4. Receive A = 54. Want to find a such that $65^a = 54 \bmod 71$

5. For each i, compute $A^{(p-1)/p_i}$ and find a_i such that $A^{(p-1)/p_i} = g_i^{a_i} \bmod p$:

   Calculate $54^{10} = 1 \bmod 71$ and compare to $20^{a_1} \bmod 71$.

   $a_1 = 0 \bmod 7$

Discrete Log Example

4. Receive A = 54. Want to find a such that $65^a = 54 \bmod 71$

5. For each i, compute $A^{(p-1)/p_i}$ and find a_i such that $A^{(p-1)/p_i} = g_i^{a_i} \bmod p$:

Calculate $54^{10} = 1 \bmod 71$ and compare to $20^{a_1} \bmod 71$.

$a_1 = 0 \bmod 7$

Calculate $54^{14} = 25 \bmod 71$ and compare to $5^{a_2} \bmod 71$.

$a_2 = 2 \bmod 5$

Discrete Log Example

4. Receive A = 54. Want to find a such that $65^a = 54 \bmod 71$

5. For each i, compute $A^{(p-1)/p_i}$ and find a_i such that $A^{(p-1)/p_i} = g_i^{a_i} \bmod p$:

   Calculate $54^{10} = 1 \bmod 71$ and compare to $20^{a_1} \bmod 71$.

   $a_1 = 0 \bmod 7$

   Calculate $54^{14} = 25 \bmod 71$ and compare to $5^{a_2} \bmod 71$.

   $a_2 = 2 \bmod 5$

   Calculate $54^{35} = 1 \bmod 71$ and compare to $70^{a_3} \bmod 71$.

   $a_3 = 0 \bmod 2$

Discrete Log Example

4. Receive A = 54. Want to find a such that $65^a = 54 \bmod 71$

5. For each i, compute $A^{(p-1)/p_i}$ and find a_i such that $A^{(p-1)/p_i} = g_i^{a_i} \bmod p$:

   Calculate $54^{10} = 1 \bmod 71$ and compare to $20^{a_1} \bmod 71$.

   $a_1 = 0 \bmod 7$

   Calculate $54^{14} = 25 \bmod 71$ and compare to $5^{a_2} \bmod 71$.

   $a_2 = 2 \bmod 5$

   Calculate $54^{35} = 1 \bmod 71$ and compare to $70^{a_3} \bmod 71$.

   $a_3 = 0 \bmod 2$

6. $a = 0 \bmod 14$, $a = 2 \bmod 5$

Discrete Log Example

4. Receive A = 54. Want to find a such that $65^a = 54 \bmod 71$

5. For each i, compute $A^{(p-1)/p_i}$ and find a_i such that $A^{(p-1)/p_i} = g_i^{a_i} \bmod p$:

Calculate $54^{10} = 1 \bmod 71$ and compare to $20^{a_1} \bmod 71$.

$a_1 = 0 \bmod 7$

Calculate $54^{14} = 25 \bmod 71$ and compare to $5^{a_2} \bmod 71$.

$a_2 = 2 \bmod 5$

Calculate $54^{35} = 1 \bmod 71$ and compare to $70^{a_3} \bmod 71$.

$a_3 = 0 \bmod 2$

6. $a = 0 \bmod 14$, $a = 2 \bmod 5$

Formula for Chinese remainder theorem

$a = 15 \cdot 0 - 14 \cdot 2 = -28 = 42 \bmod 70$

Discrete Log Example

4. Receive A = 54. Want to find a such that $65^a = 54 \bmod 71$

5. For each i, compute $A^{(p-1)/p_i}$ and find a_i such that $A^{(p-1)/p_i} = g_i^{a_i} \bmod p$:

Calculate $54^{10} = 1 \bmod 71$ and compare to $20^{a_1} \bmod 71$.

$a_1 = 0 \bmod 7$

Calculate $54^{14} = 25 \bmod 71$ and compare to $5^{a_2} \bmod 71$.

$a_2 = 2 \bmod 5$

Calculate $54^{35} = 1 \bmod 71$ and compare to $70^{a_3} \bmod 71$.

$a_3 = 0 \bmod 2$

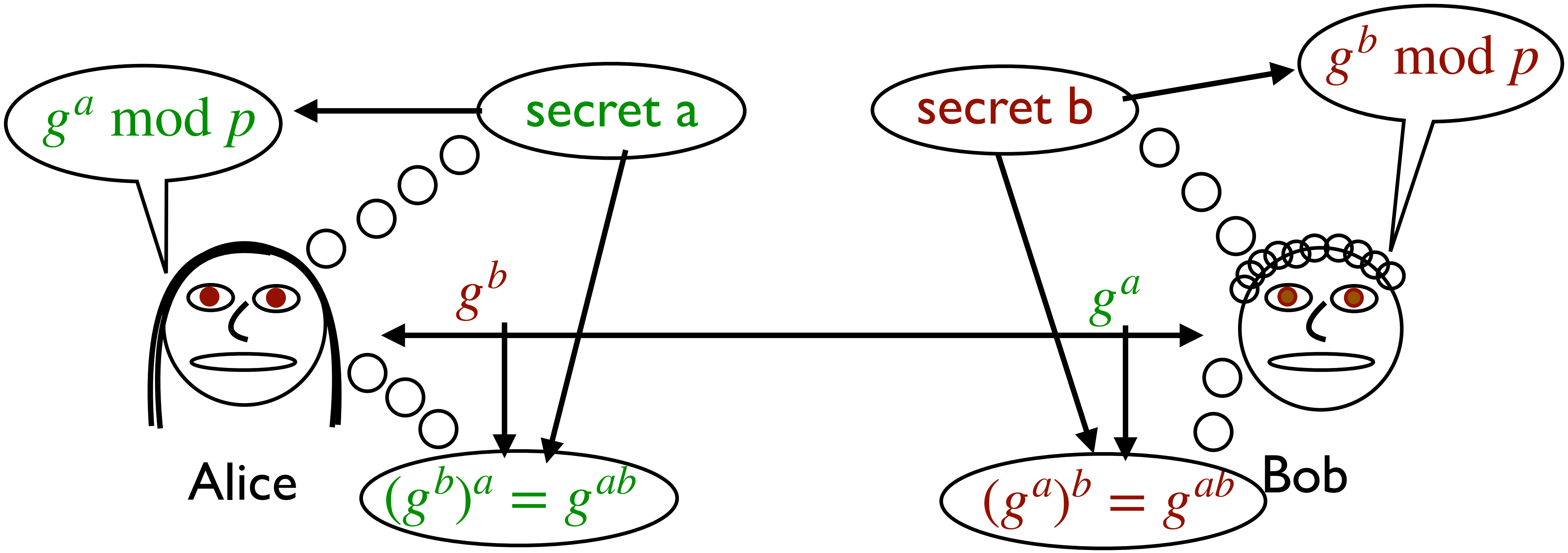
6. $a = 0 \bmod 14$, $a = 2 \bmod 5$

Formula for Chinese remainder theorem

$a = 15 \cdot 0 - 14 \cdot 2 = -28 = 42 \bmod 70$

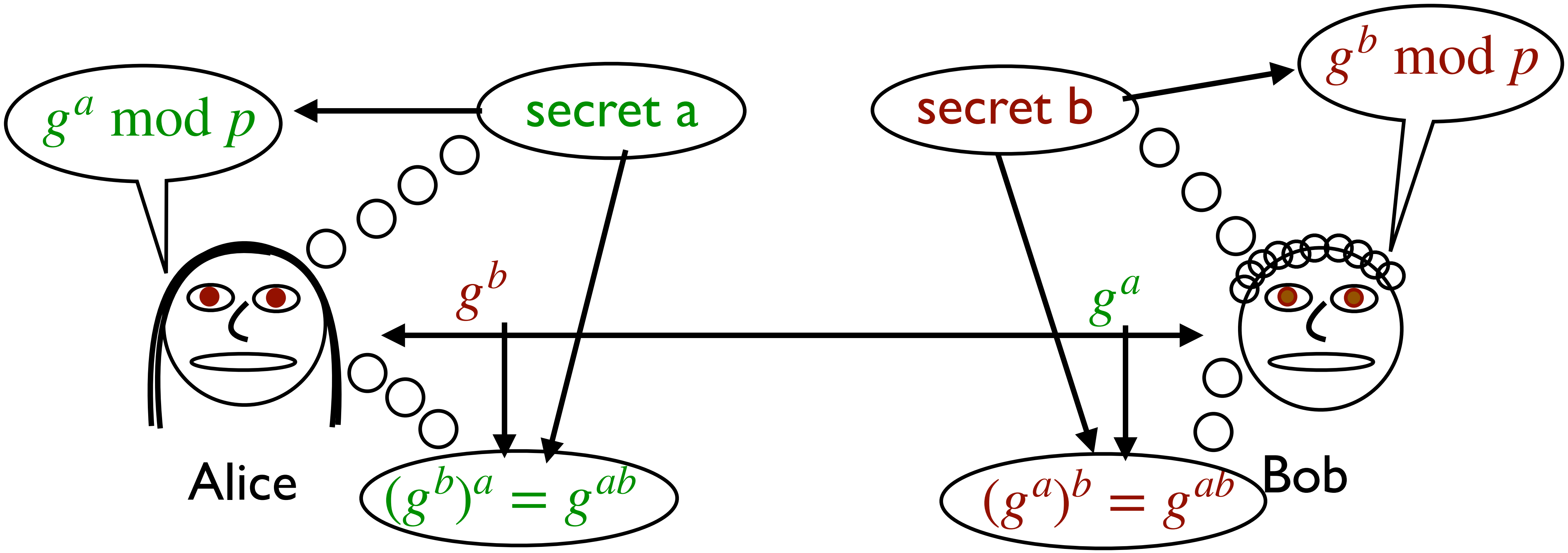
$65^{42} = 54 \bmod 71$

Diffie-Hellman Requirements



In order to have some hope that Diffie-Hellman is secure, we want:

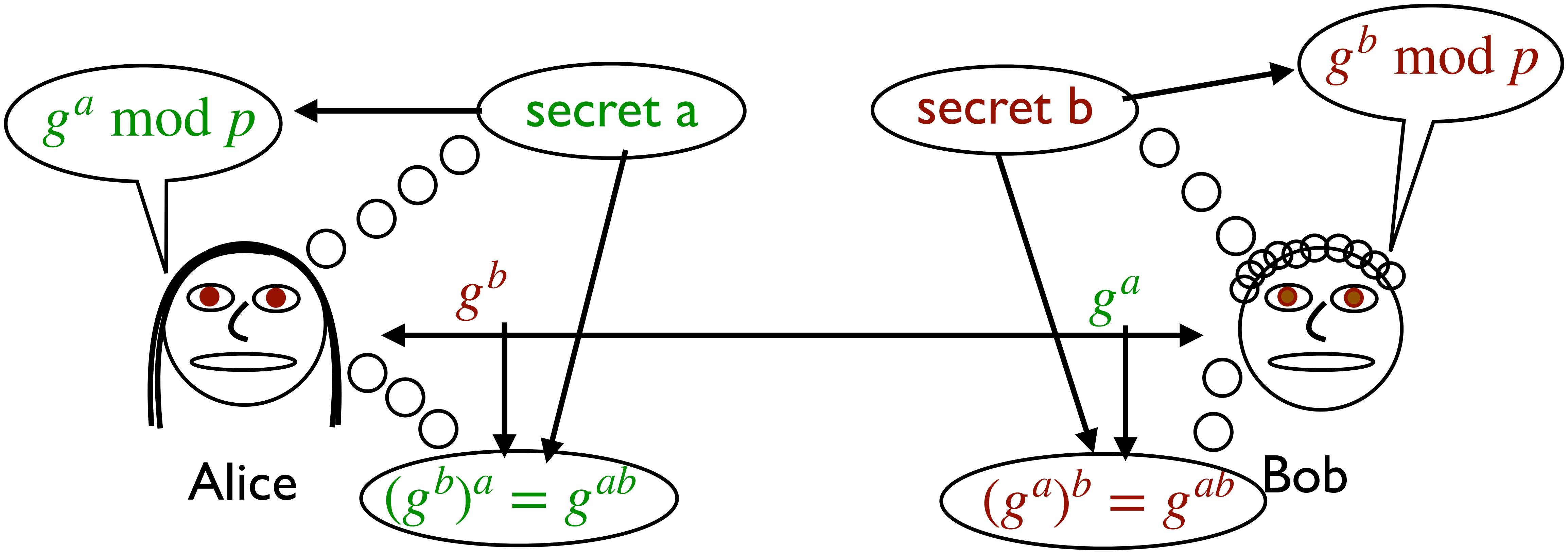
Diffie-Hellman Requirements



In order to have some hope that Diffie-Hellman is secure, we want:

- To pick a large prime p

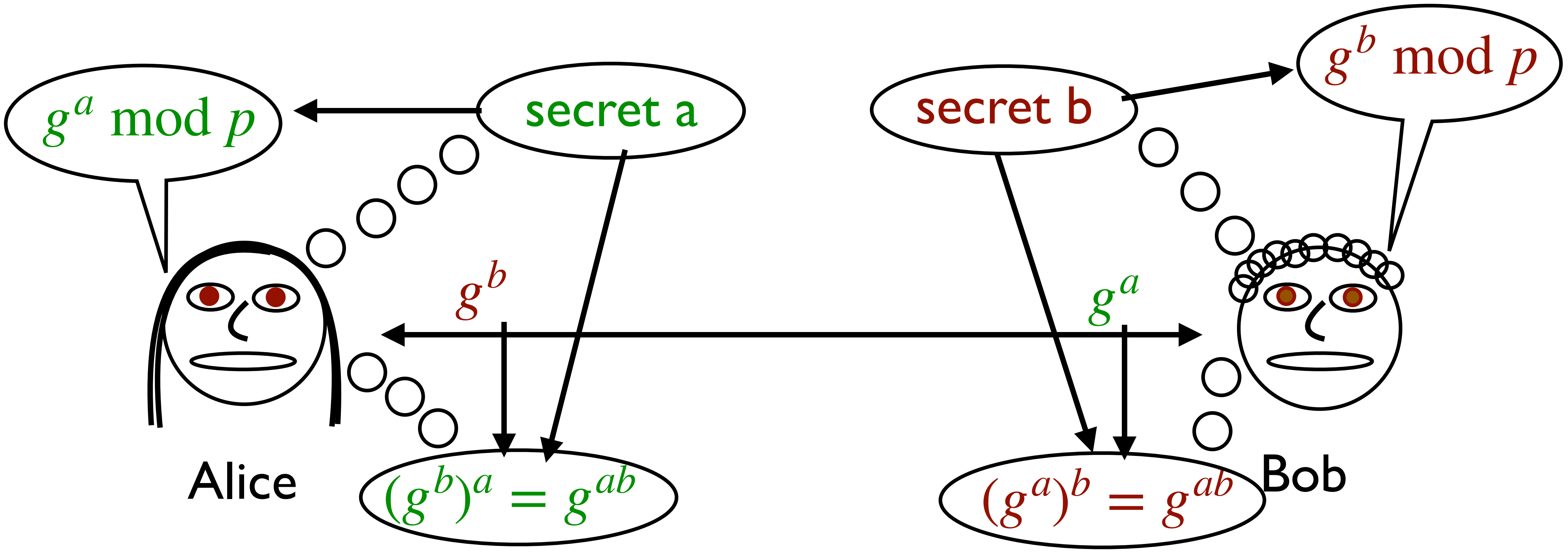
Diffie-Hellman Requirements



In order to have some hope that Diffie-Hellman is secure, we want:

- To pick a large prime p
- To ensure that p-1 is not a product of small primes

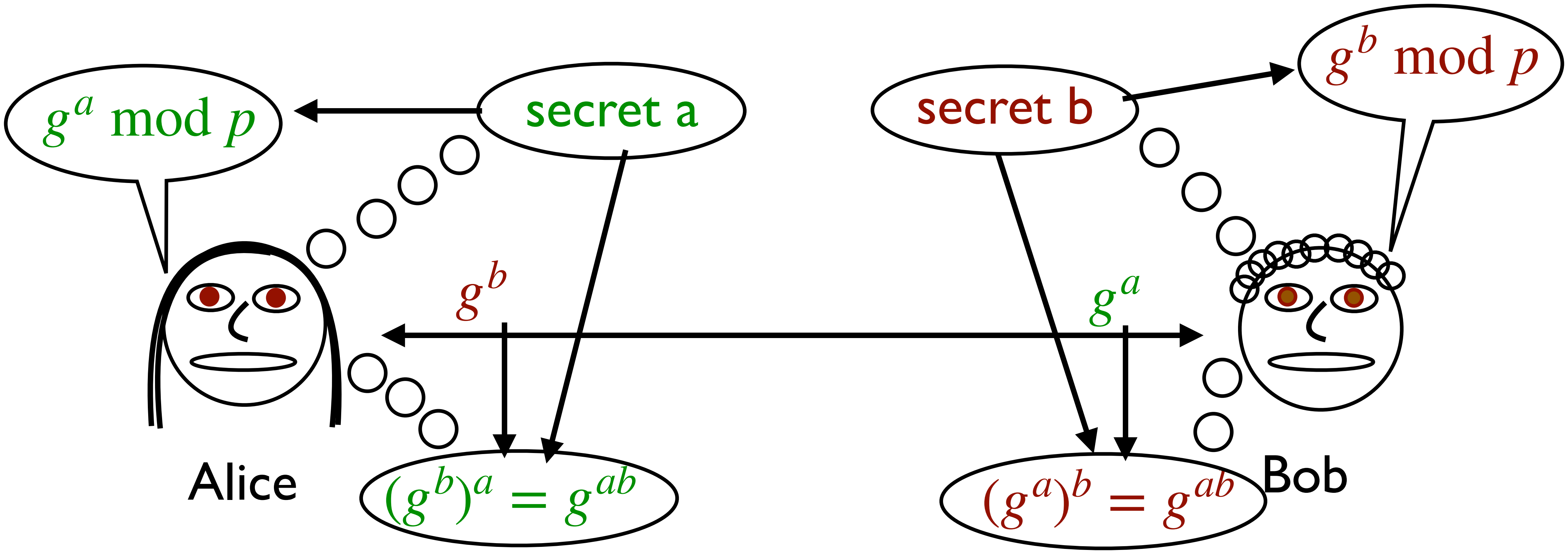
Diffie-Hellman Requirements



In order to have some hope that Diffie-Hellman is secure, we want:

- To pick a large prime p
- To ensure that p-1 is not a product of small primes
- To have $\varphi(p-1)$ large so it is not too hard to find elements with high order

Diffie-Hellman Requirements



In order to have some hope that Diffie-Hellman is secure, we want:

- To pick a large prime p
- To ensure that p-1 is not a product of small primes
- To have $\varphi(p-1)$ large so it is not too hard to find elements with high order
- To actually pick a g with high order

Safe Primes

For Diffie-Hellman to be secure, we need to find a prime base p such that p-1 has at least one large prime factor, and we also need to be able to find g with large order mod p.

(If the order is not large, Eve can just try all powers of g.)

Safe Primes

For Diffie-Hellman to be secure, we need to find a prime base p such that p-1 has at least one large prime factor, and we also need to be able to find g with large order mod p.

(If the order is not large, Eve can just try all powers of g.)

We will look for a prime of the form $p = rq + 1$, where r is small (e.g., r=2) and q is also prime. This guarantees a large prime factor for p-1.

Let us specialize to prime r, although this is not essential.

Safe Primes

For Diffie-Hellman to be secure, we need to find a prime base p such that p-1 has at least one large prime factor, and we also need to be able to find g with large order mod p.

(If the order is not large, Eve can just try all powers of g.)

We will look for a prime of the form $p = rq + 1$, where r is small (e.g., r=2) and q is also prime. This guarantees a large prime factor for p-1.

Let us specialize to prime r, although this is not essential.

Since p is prime, $\mathbb{Z}_p^*$ is cyclic, so there exist elements of order rq. How many?

Safe Primes

For Diffie-Hellman to be secure, we need to find a prime base p such that p-1 has at least one large prime factor, and we also need to be able to find g with large order mod p.

(If the order is not large, Eve can just try all powers of g.)

We will look for a prime of the form $p = rq + 1$, where r is small (e.g., r=2) and q is also prime. This guarantees a large prime factor for p-1.

Let us specialize to prime r, although this is not essential.

Since p is prime, $\mathbb{Z}_p^*$ is cyclic, so there exist elements of order rq. How many?

$\varphi(p-1) = (r-1)(q-1)$: the relatively prime powers of the generator. This is large.

There are also q-1 elements of order q and r-1 of order r.

This class is being recorded

Finding Primes

How can we find a prime, let alone a prime of a specific form?

1. Choose a random number p of the desired length.
2. Check that p is prime.
3. Check that $(p-1)/r$ is prime.
4. Repeat until both p and $(p-1)/r$ are prime.

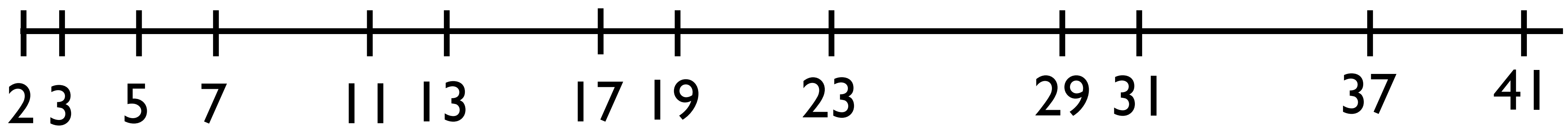
Remarkably, this works. But there are two pieces needed to make it work:

- We need to be sure that primes are sufficiently common that we can find a prime in a reasonable time.
- We need an efficient algorithm to check that a number is prime.

For secure Diffie-Hellman, we will need p that is at least thousands of bits long, so efficiency is important.

How Common Are Primes?

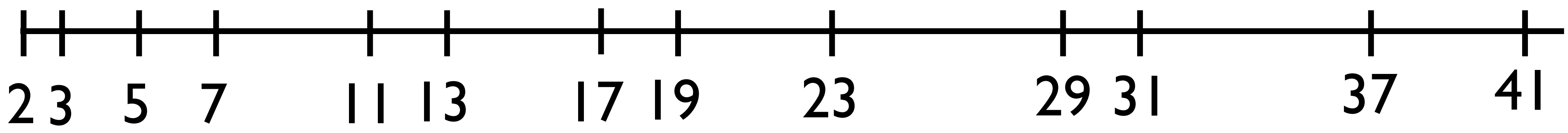
What do we expect?



Primes get rarer as the numbers get larger, but only slowly.

How Common Are Primes?

What do we expect?



Primes get rarer as the numbers get larger, but only slowly.

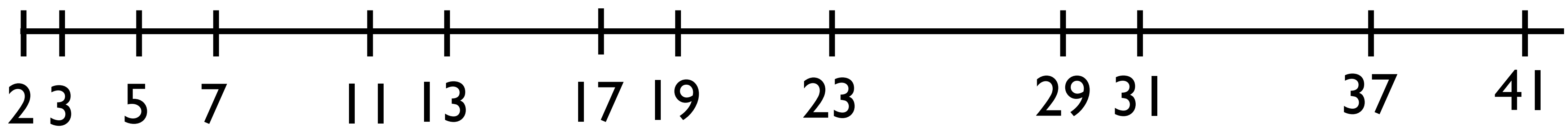
Prime Number Theorem: Let $\pi(n)$ be the number of primes less than or equal to n. Then

$$\pi(n) \approx n/\ln n$$

The probability that a random s-bit number is prime is about 1/s.

How Common Are Primes?

What do we expect?



Primes get rarer as the numbers get larger, but only slowly.

Prime Number Theorem: Let $\pi(n)$ be the number of primes less than or equal to n. Then

$$\pi(n) \approx n/\ln n$$

The probability that a random s-bit number is prime is about 1/s.

Therefore, if we choose s-bit numbers at random, we find a prime after O(s) tries, which is efficient.

Testing Primes

We also need a method to test for primes: Given N, is N prime?

Fermat's Little Theorem states that, if N is prime, then

$$x^N = x \bmod N$$

for all x.

This suggests the following algorithm:

1. Choose random x
2. Calculate $y = x^N \bmod N$
3. If $y \neq x$, end the loop and return: Composite
4. Repeat steps 1-3 a number of times
5. If we are still going, return: Prime

Vote: Does this algorithm work? (Yes/No)

Testing Primes

We also need a method to test for primes: Given N, is N prime?

Fermat's Little Theorem states that, if N is prime, then

$$x^N = x \bmod N$$

for all x.

This suggests the following algorithm:

1. Choose random x
2. Calculate $y = x^N \bmod N$
3. If $y \neq x$, end the loop and return: Composite
4. Repeat steps 1-3 a number of times
5. If we are still going, return: Prime

Vote: Does this algorithm work? (Yes/No) Answer: No

Pseudoprimes

Unfortunately, there are some composite numbers N such that

$$x^N = x \bmod N$$

for all x. These are called pseudoprimes or Carmichael numbers.

The smallest one is 561. They seem to be rarer than prime numbers, but it is not clear if they are sufficiently rare that we can neglect the probability of choosing one if we choose N at random.

(Carmichael numbers fail the test $x^{N-1} = 1 \bmod N$ when x is not relatively prime to N. Unfortunately, it is possible that only a small fraction of possible x's share a common factor with N.)

Miller-Rabin Primality Test

Some modifications are needed to make the previous algorithm work.

The Miller-Rabin primality test is a probabilistic test but one that works (except with negligible probability) for all N, including pseudoprimes.

> It takes advantage of the fact that if N is composite, then exists some a such that $a \neq \pm 1 \bmod N$ but $a^2 = 1 \bmod N$.

E.g., for N = 561, $188^2 = 1 \bmod 561$.

Miller-Rabin Primality Test

Some modifications are needed to make the previous algorithm work.

The Miller-Rabin primality test is a probabilistic test but one that works (except with negligible probability) for all N, including pseudoprimes.

> It takes advantage of the fact that if N is composite, then exists some a such that $a \neq \pm 1 \bmod N$ but $a^2 = 1 \bmod N$.

E.g., for N = 561, $188^2 = 1 \bmod 561$.

This follows from the Chinese remainder theorem:

If $N = uv$, there is a solution to

$$a = -1 \bmod u$$
$$a = 1 \bmod v$$

which must satisfy the desired two conditions.

Choosing a Base

Once we have a modulus $p = rq + 1$, with p and q both prime and r small (e.g., r=2), the next step is to find a base g.

We want to pick g to have large order. Let us specialize to r=2. Then the factors of p-1 are 1, 2, q, and 2q. These are the possible orders for g. Obviously we shouldn't choose g with order 2, since then g^a would either be g or 1, which can be easily solved.

> Vote: Do we prefer order q or order 2q? Or does it not matter?

Base of Order 2q

Suppose we choose g with order 2q, so $g^{2q} = 1 \bmod p$, but $g^q \neq 1 \bmod p$.

In this case, g generates the whole group of $\mathbb{Z}_p^*$, which has an order q subgroup consisting of elements g^{2i} for integer i.

Notice: Eve can deduce something about a: Given $A = g^a \bmod p$, Eve can tell if A is in the order q subgroup or not.

How?

Base of Order 2q

Suppose we choose g with order 2q, so $g^{2q} = 1 \bmod p$, but $g^q \neq 1 \bmod p$.

In this case, g generates the whole group of $\mathbb{Z}_p^*$, which has an order q subgroup consisting of elements g^{2i} for integer i.

Notice: Eve can deduce something about a: Given $A = g^a \bmod p$, Eve can tell if A is in the order q subgroup or not.

How? Calculate $A^q \bmod p$ and see if it is 1.

Base of Order 2q

Suppose we choose g with order 2q, so $g^{2q} = 1 \bmod p$, but $g^q \neq 1 \bmod p$.

In this case, g generates the whole group of $\mathbb{Z}_p^*$, which has an order q subgroup consisting of elements g^{2i} for integer i.

Notice: Eve can deduce something about a: Given $A = g^a \bmod p$, Eve can tell if A is in the order q subgroup or not.

How? Calculate $A^q \bmod p$ and see if it is 1.

A has order q iff a is even. Similarly, B has order q iff b is even.

Base of Order 2q

Suppose we choose g with order 2q, so $g^{2q} = 1 \bmod p$, but $g^q \neq 1 \bmod p$.

In this case, g generates the whole group of $\mathbb{Z}_p^*$, which has an order q subgroup consisting of elements g^{2i} for integer i.

Notice: Eve can deduce something about a: Given $A = g^a \bmod p$, Eve can tell if A is in the order q subgroup or not.

How? Calculate $A^q \bmod p$ and see if it is 1.

A has order q iff a is even. Similarly, B has order q iff b is even.

The final key $k = g^{ab} \bmod p$ has order q iff either a or b is even, which happens iff A or B is order q.

Base of Order 2q

Suppose we choose g with order 2q, so $g^{2q} = 1 \bmod p$, but $g^q \neq 1 \bmod p$.

In this case, g generates the whole group of $\mathbb{Z}_p^*$, which has an order q subgroup consisting of elements g^{2i} for integer i.

Notice: Eve can deduce something about a: Given $A = g^a \bmod p$, Eve can tell if A is in the order q subgroup or not.

How? Calculate $A^q \bmod p$ and see if it is 1.

A has order q iff a is even. Similarly, B has order q iff b is even.

The final key $k = g^{ab} \bmod p$ has order q iff either a or b is even, which happens iff A or B is order q.

Eve can deduce one bit of information about the key k. She can use this to distinguish k from random k'.

Picking an Element of Order q

This means it is better to use a base which has order q.

How can we choose an element of order q?

Picking an Element of Order q

This means it is better to use a base which has order q.

How can we choose an element of order q?

1. Choose random $x \in \mathbb{Z}_p^*$.
2. Let $g = x^r \bmod p$.
3. Repeat until $g \neq 1$.

This generates a random element of order q in $\mathbb{Z}_p^*$.

Picking an Element of Order q

This means it is better to use a base which has order q.

How can we choose an element of order q?

1. Choose random $x \in \mathbb{Z}_p^*$.
2. Let $g = x^r \bmod p$.
3. Repeat until $g \neq 1$.

This generates a random element of order q in $\mathbb{Z}_p^*$.

Steps 1 and 2 generate a random element of the order q cyclic subgroup of $\mathbb{Z}_p^*$. Since q is prime, all elements of that subgroup have order q except for 1.

Picking an Element of Order q

This means it is better to use a base which has order q.

How can we choose an element of order q?

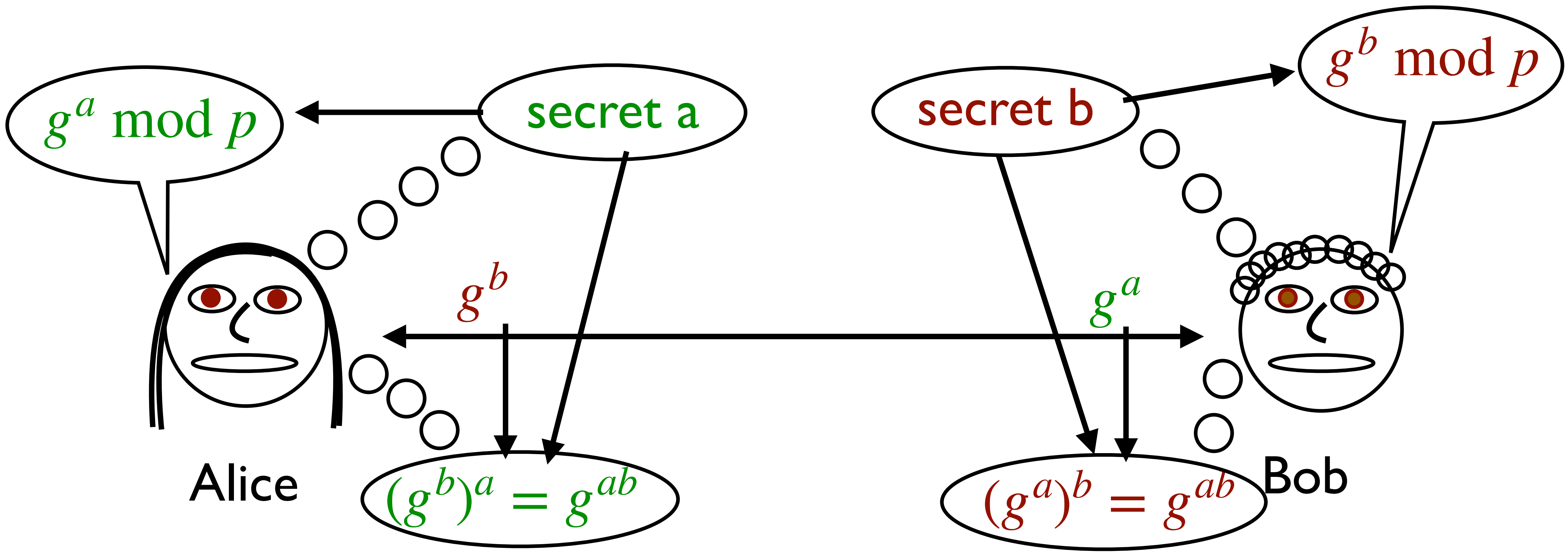
1. Choose random $x \in \mathbb{Z}_p^*$.
2. Let $g = x^r \bmod p$.
3. Repeat until $g \neq 1$.

This generates a random element of order q in $\mathbb{Z}_p^*$.

Steps 1 and 2 generate a random element of the order q cyclic subgroup of $\mathbb{Z}_p^*$. Since q is prime, all elements of that subgroup have order q except for 1.

We want to pick an element of prime order to avoid leaking any information about the key. This is why we need to pick a prime p of this specific form to make Diffie-Hellman secure.

Choosing g and p for Diffie-Hellman



- Choose random p until we find one such that p is prime and p-1 = rq, for small r and prime q.
- Choose $g \in \mathbb{Z}_p^*$ with order q.
- Or use standard values for g and p.

Is this secure?

This class is being recorded