CMSC/Math 456: Cryptography (Fall 2023)

Lecture 13

Daniel Gottesman

Administrative

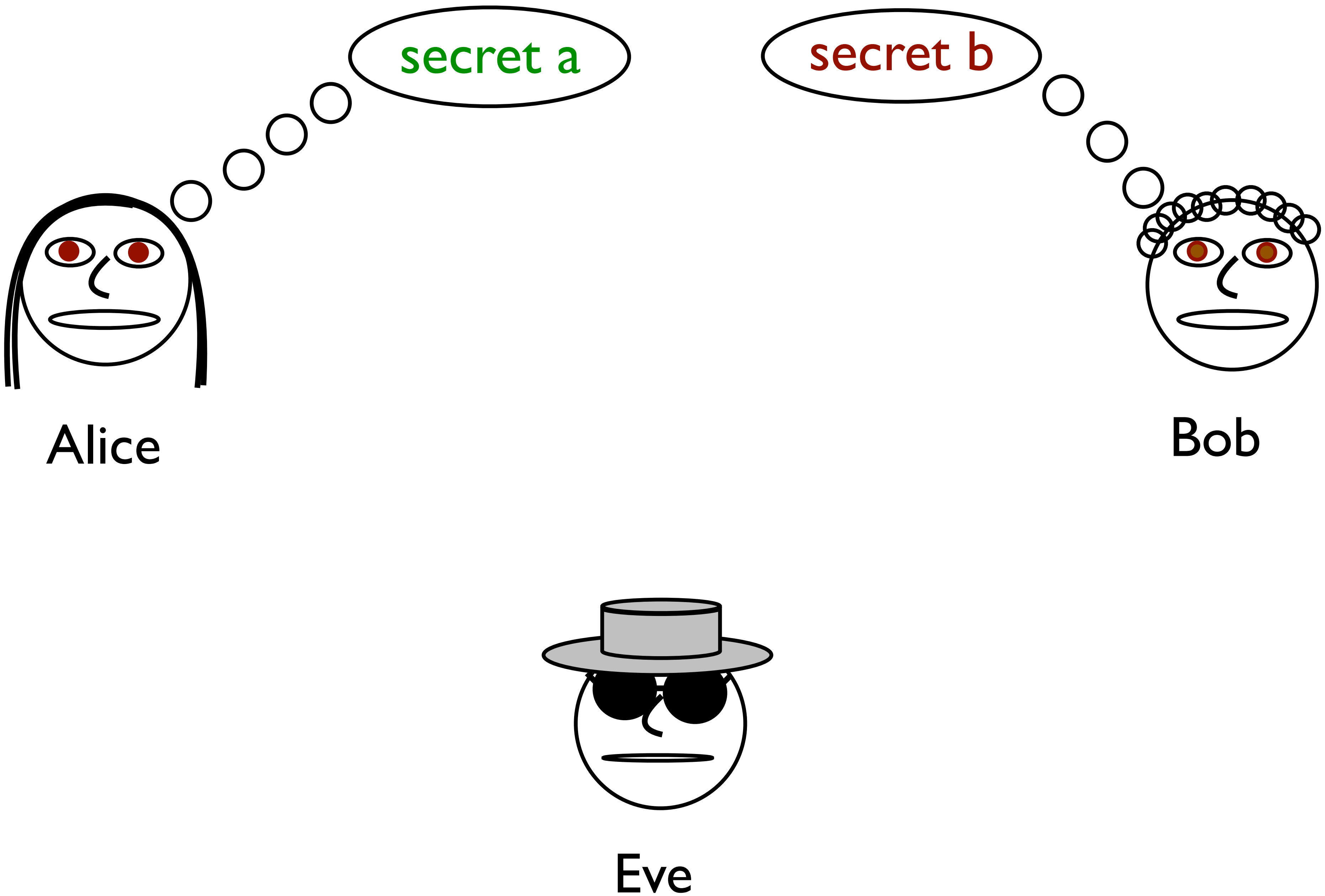
Problem set #6 is due Thursday at noon.

Midterm: Thursday, Oct. 19 (1.5 weeks from today)

- In class
- Open book (including textbook), no electronic devices
- Will cover material through key exchange, but not public key encryption.
- Those with accommodations remember to book with ADS.

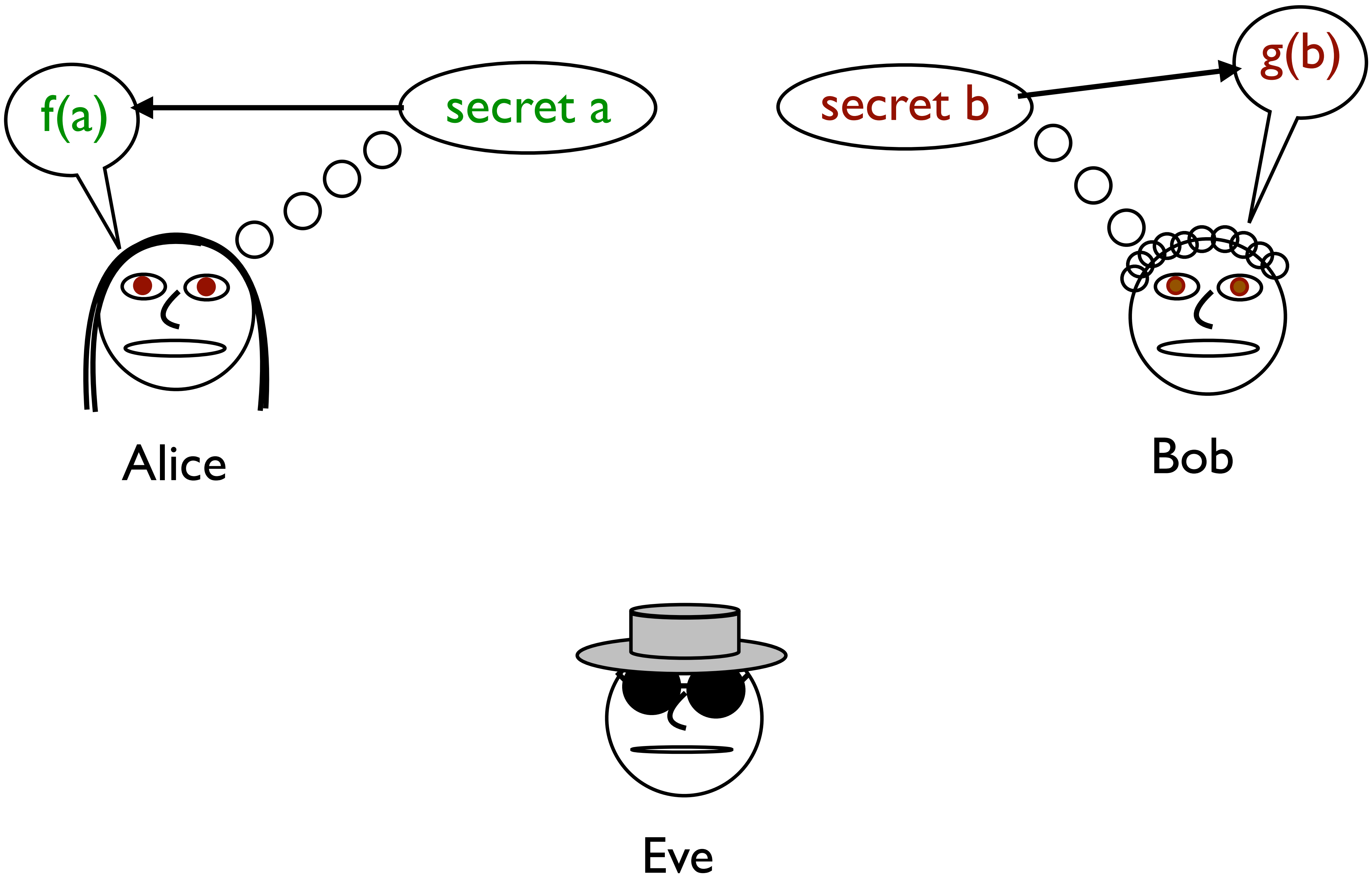
Tuesday, Oct. 17 I will answer questions and review a few (probably 1-3) selected topics from the first half of the class. I will create a poll on Piazza as to which topics people would like to see reviewed.

Key Exchange



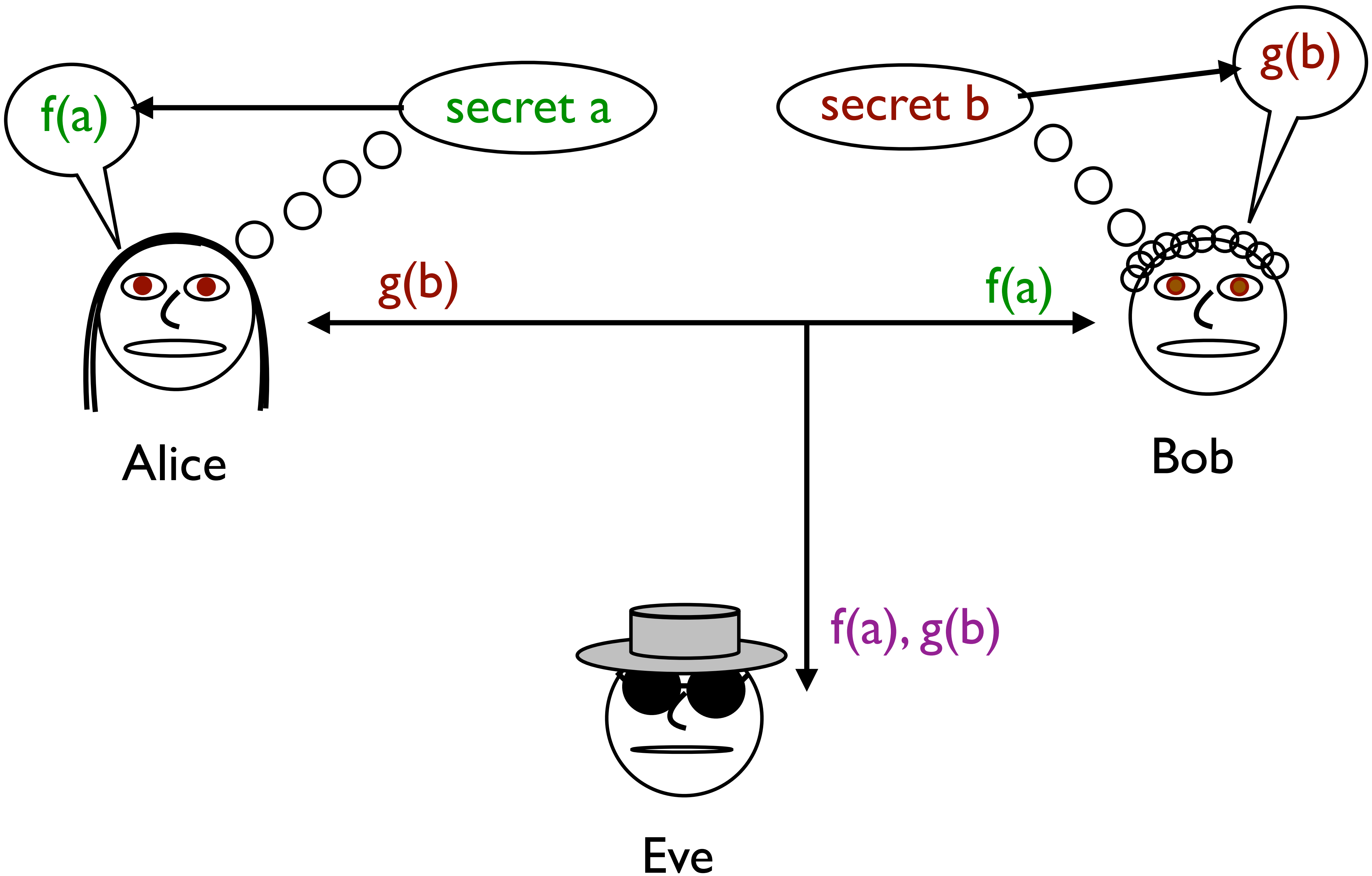
This class is being recorded

Key Exchange



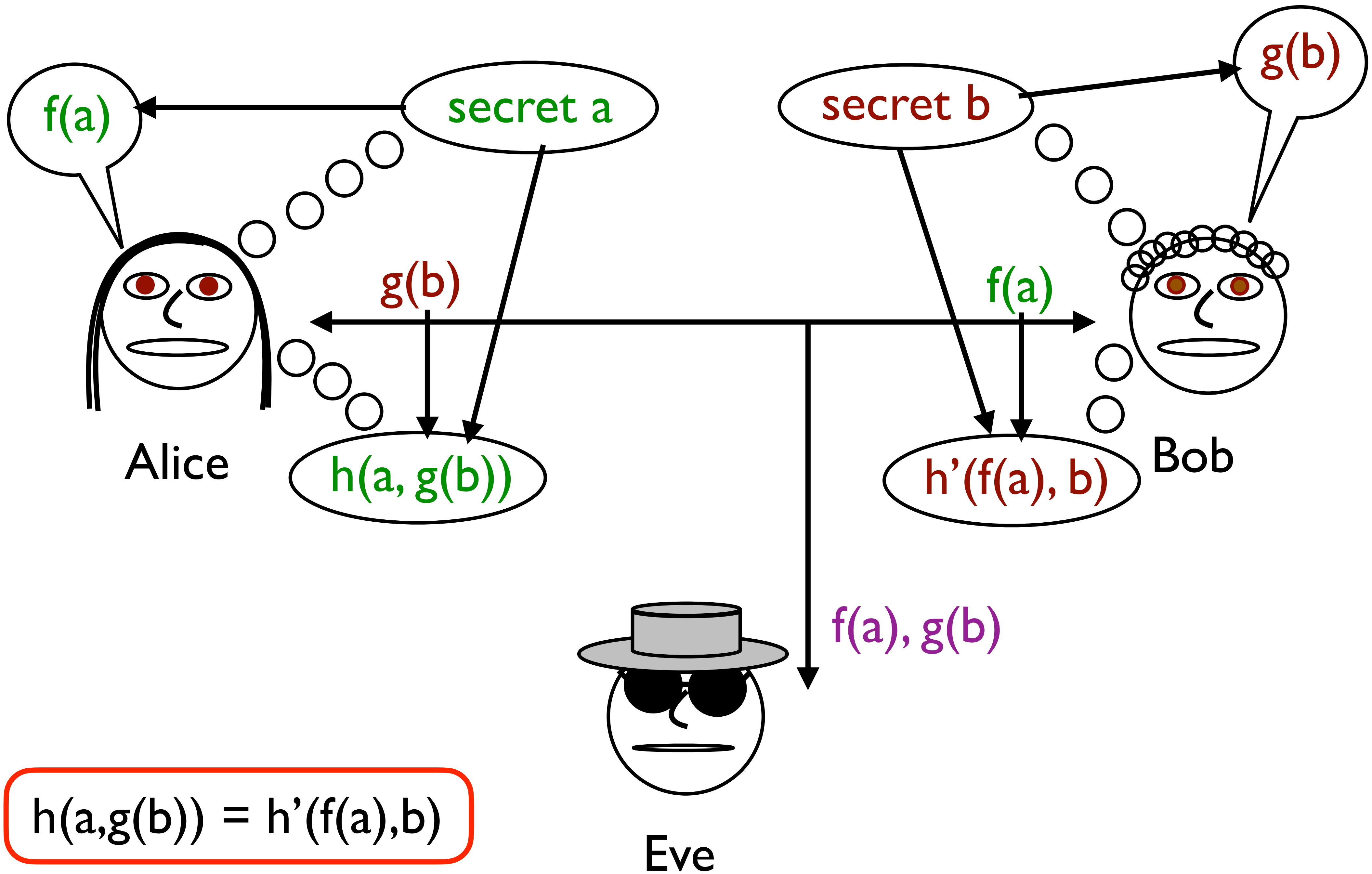
This class is being recorded

Key Exchange



This class is being recorded

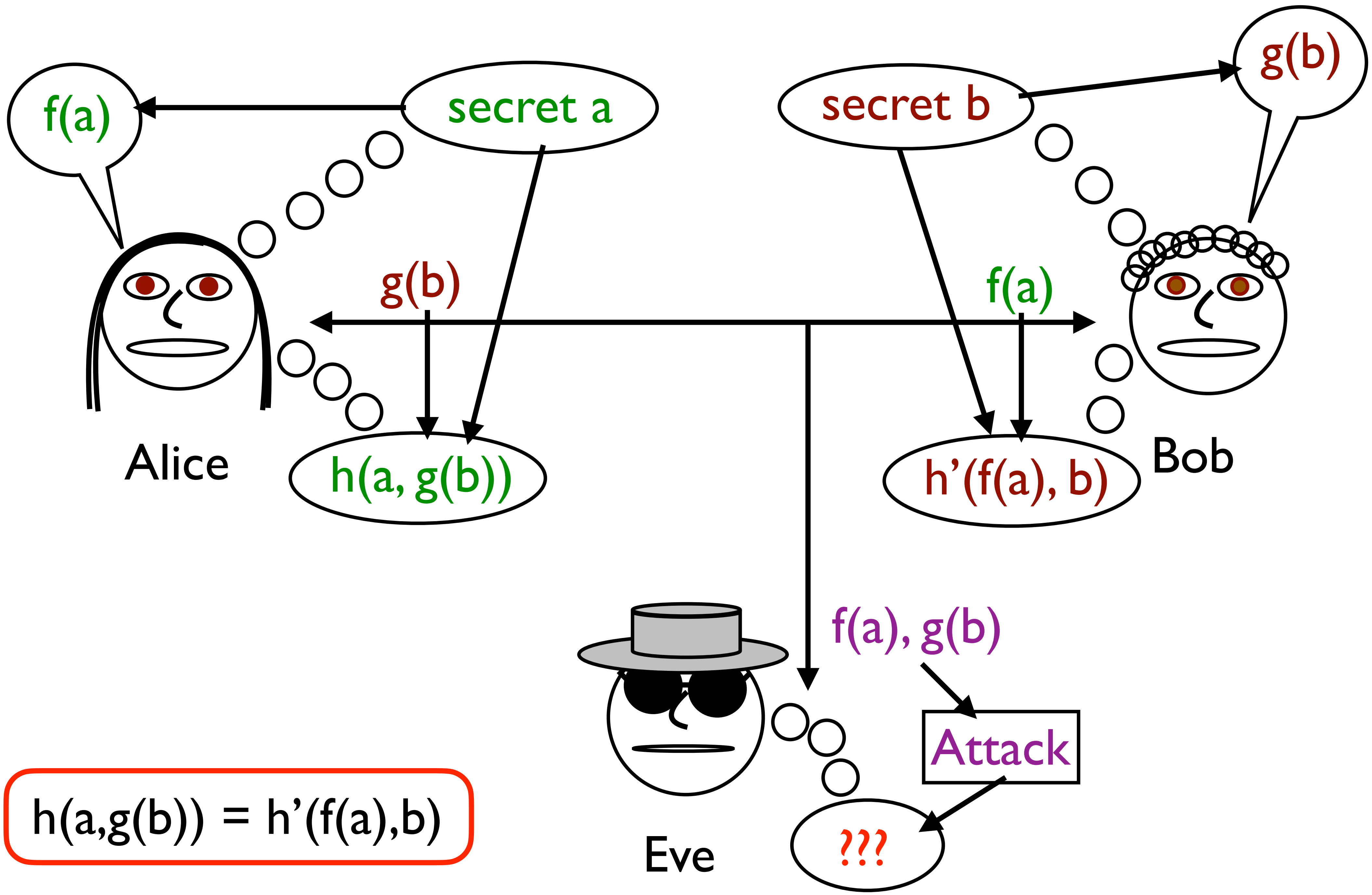
Key Exchange



$h(a,g(b)) = h'(f(a),b)$

This class is being recorded

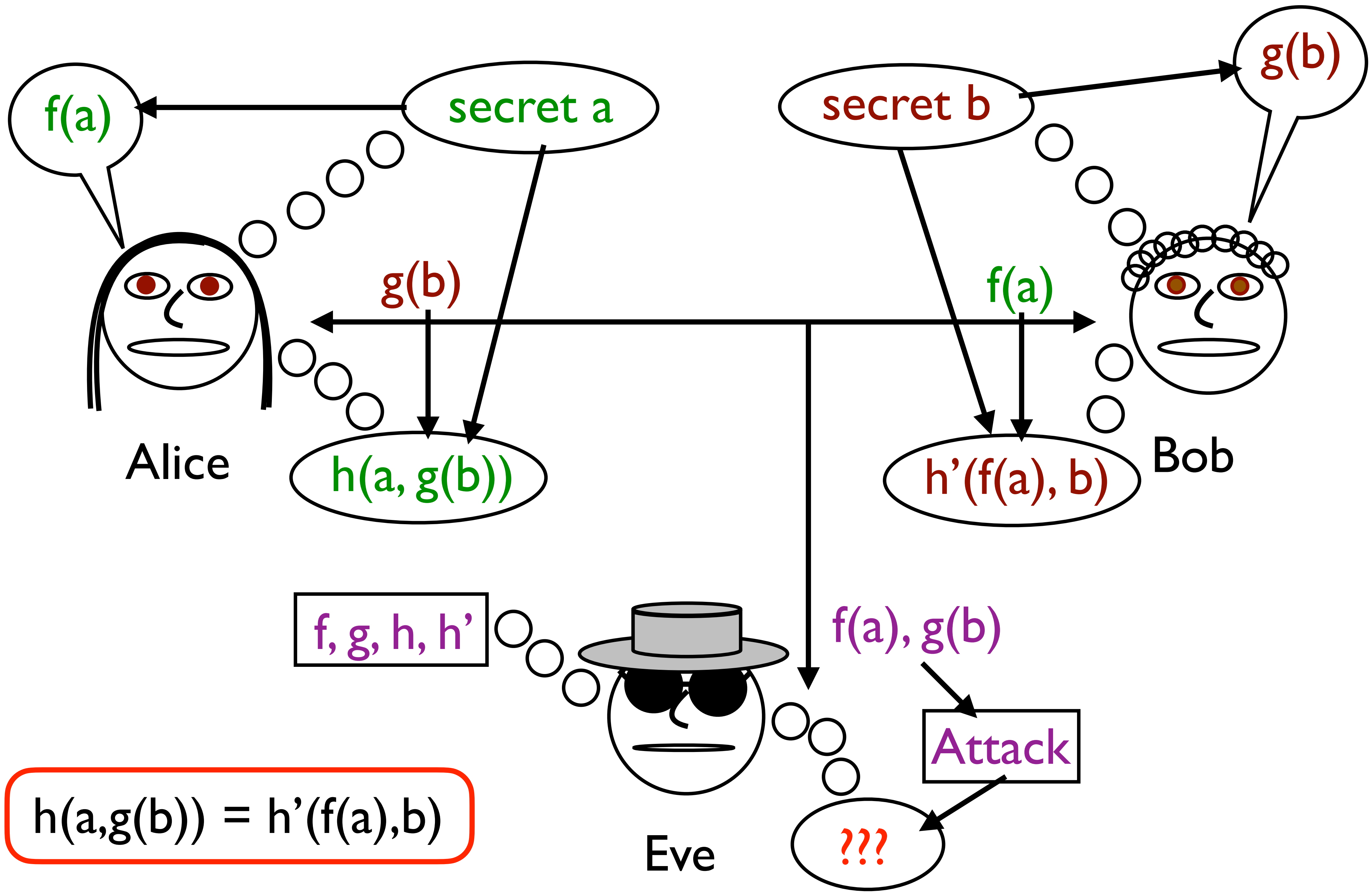
Key Exchange



$h(a,g(b)) = h'(f(a),b)$

This class is being recorded

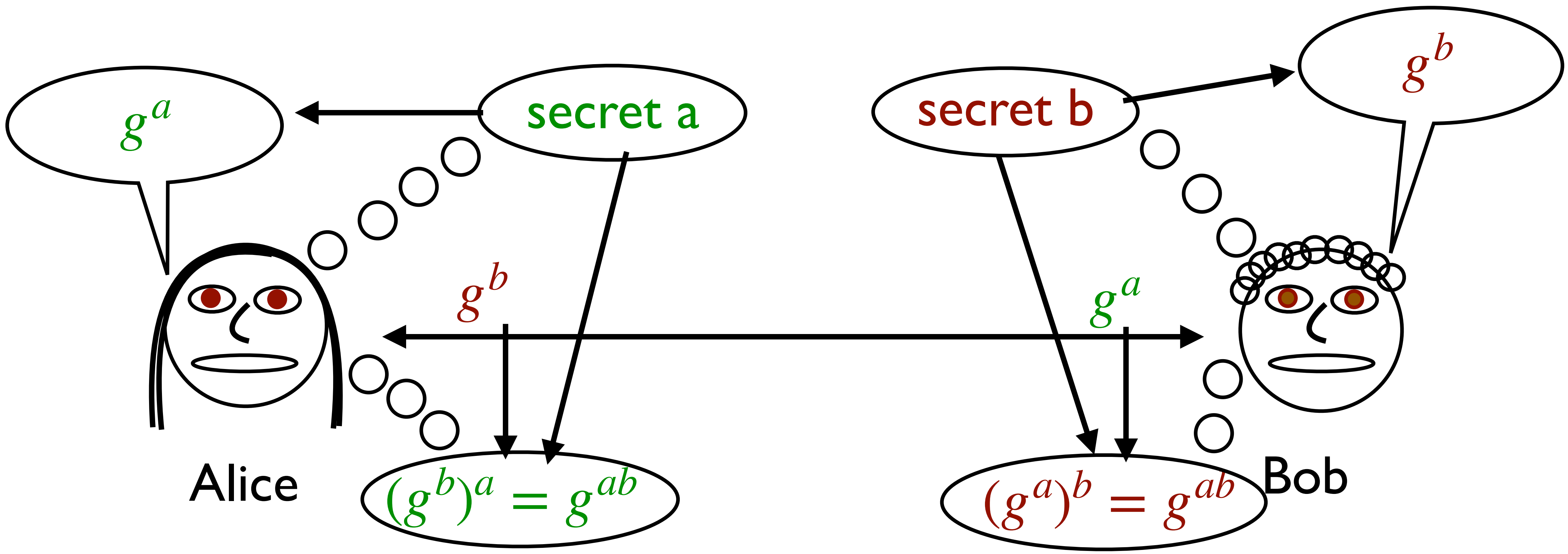
Key Exchange



This class is being recorded

Diffie-Hellman with Groups

Diffie-Hellman also works when g is drawn from a group G.

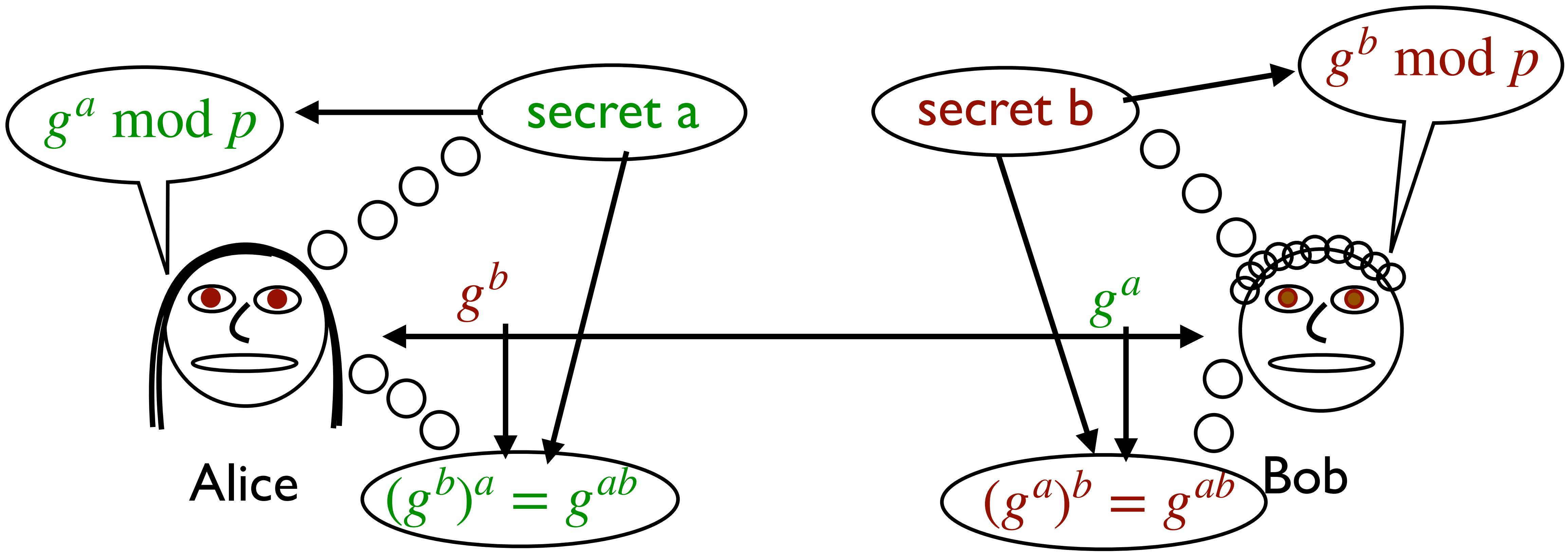


Alice and Bob must first agree on the group G and the element g. G is cyclic and $G = \langle g \rangle$.

Again, they can use standardized values for g and G.

Elliptic curves are common; they allow smaller groups than modular arithmetic.

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?

Hardness of Discrete Log

Definition: Given a security parameter s, let N_s be an s-bit long number, and let $x_s \in \mathbb{Z}_{N_s}^*$ be an element of $\mathbb{Z}_{N_s}^*$. We say that discrete log for (N_s, x_s) is worst-case hard if there is *no* polynomial time algorithm $\mathscr{A}$ such that for all $y \in \langle x_s \rangle$, $\mathscr{A}(y) = a$ with $y = x_s^a \bmod N_s$.

To get an asymptotic complexity definition, we take a sequence of pairs (modulus, base) that get longer.

Vote: If we have a family (p_s, g_s) such that discrete log for (p_s, g_s) is worst-case hard, does this suffice to prove the security of Diffie-Hellman? (Yes/No/No one knows)

Hardness of Discrete Log

Definition: Given a security parameter s, let N_s be an s-bit long number, and let $x_s \in \mathbb{Z}_{N_s}^*$ be an element of $\mathbb{Z}_{N_s}^*$. We say that discrete log for (N_s, x_s) is worst-case hard if there is *no* polynomial time algorithm $\mathscr{A}$ such that for all $y \in \langle x_s \rangle$, $\mathscr{A}(y) = a$ with $y = x_s^a \bmod N_s$.

To get an asymptotic complexity definition, we take a sequence of pairs (modulus, base) that get longer.

Vote: If we have a family (p_s, g_s) such that discrete log for (p_s, g_s) is worst-case hard, does this suffice to prove the security of Diffie-Hellman? (Yes/No/No one knows) Unknown

Hardness of Discrete Log

Definition: Given a security parameter s, let N_s be an s-bit long number, and let $x_s \in \mathbb{Z}_{N_s}^*$ be an element of $\mathbb{Z}_{N_s}^*$. We say that discrete log for (N_s, x_s) is worst-case hard if there is *no* polynomial time algorithm $\mathscr{A}$ such that for all $y \in \langle x_s \rangle$, $\mathscr{A}(y) = a$ with $y = x_s^a \bmod N_s$.

To get an asymptotic complexity definition, we take a sequence of pairs (modulus, base) that get longer.

Vote: If we have a family (p_s, g_s) such that discrete log for (p_s, g_s) is worst-case hard, does this suffice to prove the security of Diffie-Hellman? (Yes/No/No one knows) Unknown

One possible problem is that Alice and Bob are choosing random a and b, which might not be the hardest examples.

Discrete Log Average Case

Try again:

Definition: Given a security parameter s, let $p_s = rq_s + 1$ be an s-bit long prime with q_s also prime, and let $g_s \in \mathbb{Z}_{p_s}^*$ be an element of order q_s. We say that discrete log for (p_s, g_s) is average-case hard if for *any* polynomial time algorithm $\mathscr{A}$, for random $y \in \langle g_s \rangle$,

$$\Pr(\mathscr{A}(y) \text{ succeeds}) \leq \epsilon(s)$$

for $\epsilon(s)$ a negligible function, where we say $\mathscr{A}(y)$ succeeds if $\mathscr{A}(y) = x$ with $y = g_s^x \bmod p_s$.

Discrete Log Average Case

Try again:

Definition: Given a security parameter s, let $p_s = rq_s + 1$ be an s-bit long prime with q_s also prime, and let $g_s \in \mathbb{Z}_{p_s}^*$ be an element of order q_s. We say that discrete log for (p_s, g_s) is average-case hard if for *any* polynomial time algorithm $\mathscr{A}$, for random $y \in \langle g_s \rangle$,

$$\Pr(\mathscr{A}(y) \text{ succeeds}) \leq \epsilon(s)$$

for $\epsilon(s)$ a negligible function, where we say $\mathscr{A}(y)$ succeeds if $\mathscr{A}(y) = x$ with $y = g_s^x \bmod p_s$.

However, this isn't actually the problem. It turns out that if discrete log is worst-case hard, it is average-case hard. (This is known as random self-reducibility.)

Discrete Log Average Case

Try again:

Definition: Given a security parameter s, let $p_s = rq_s + 1$ be an s-bit long prime with q_s also prime, and let $g_s \in \mathbb{Z}_{p_s}^*$ be an element of order q_s. We say that discrete log for (p_s, g_s) is average-case hard if for *any* polynomial time algorithm $\mathscr{A}$, for random $y \in \langle g_s \rangle$,

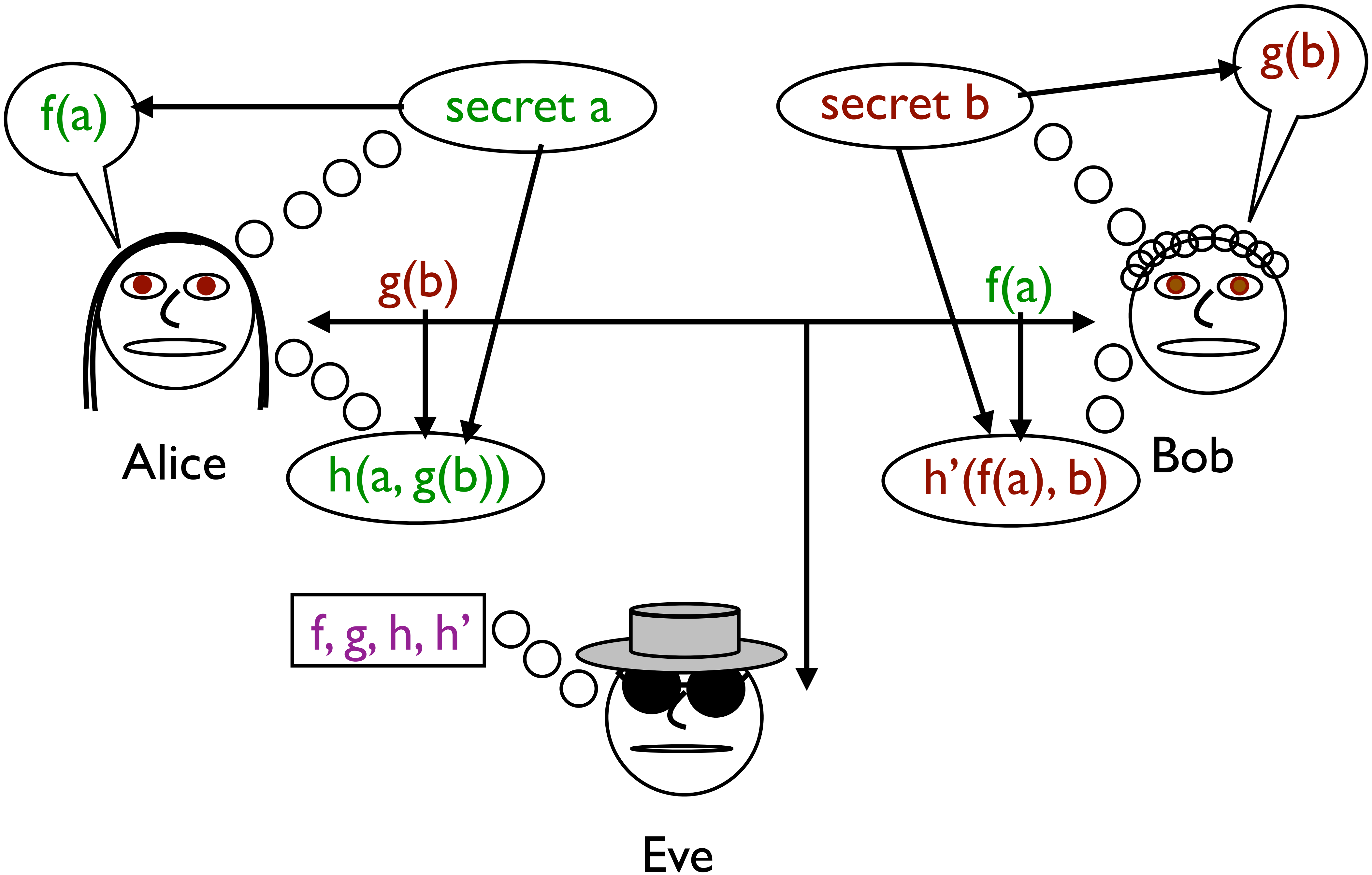
$$\Pr(\mathscr{A}(y) \text{ succeeds}) \leq \epsilon(s)$$

for $\epsilon(s)$ a negligible function, where we say $\mathscr{A}(y)$ succeeds if $\mathscr{A}(y) = x$ with $y = g_s^x \bmod p_s$.

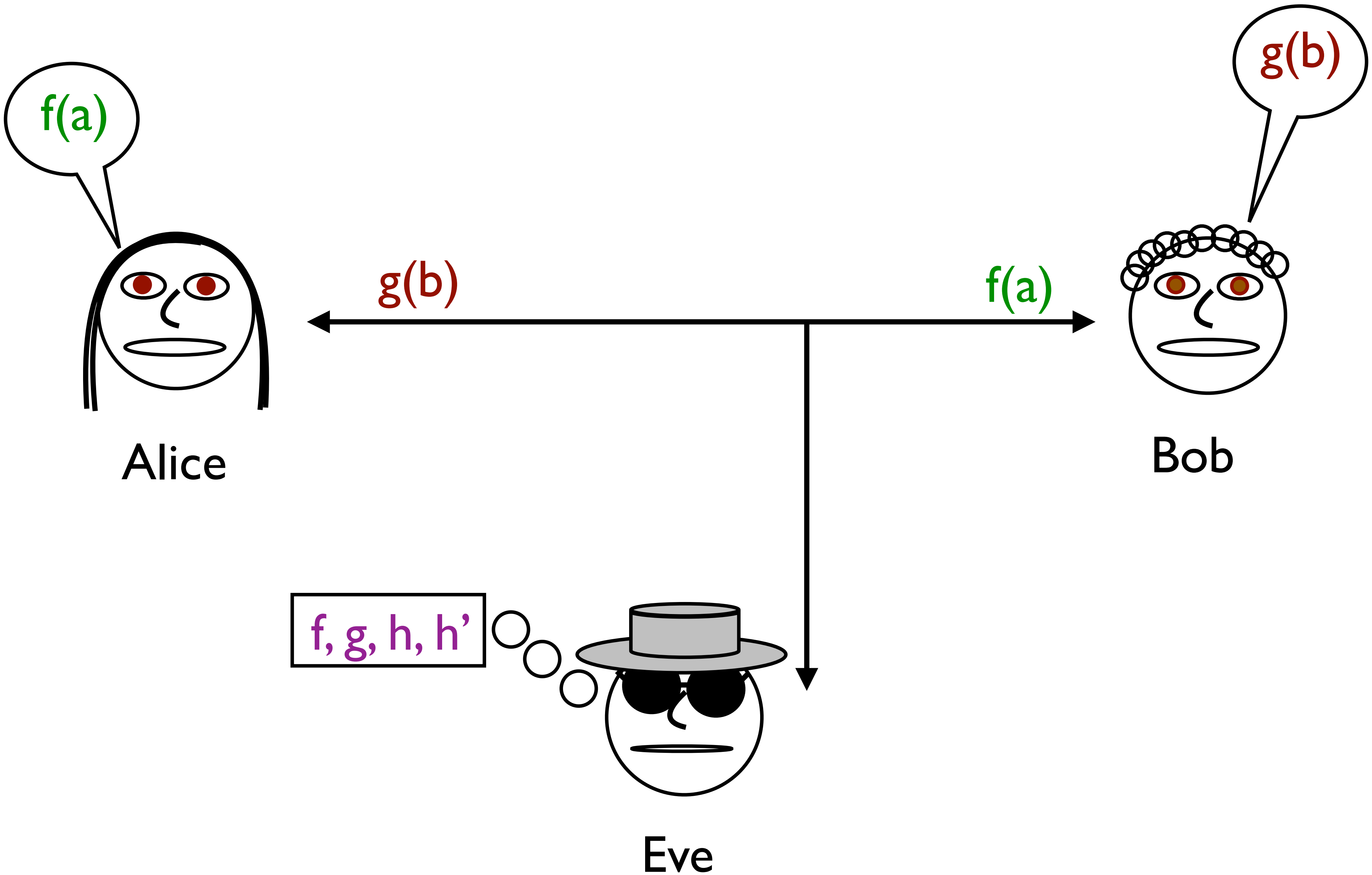
However, this isn't actually the problem. It turns out that if discrete log is worst-case hard, it is average-case hard. (This is known as random self-reducibility.)

But what does it mean for Diffie-Hellman to be secure?

Transcript

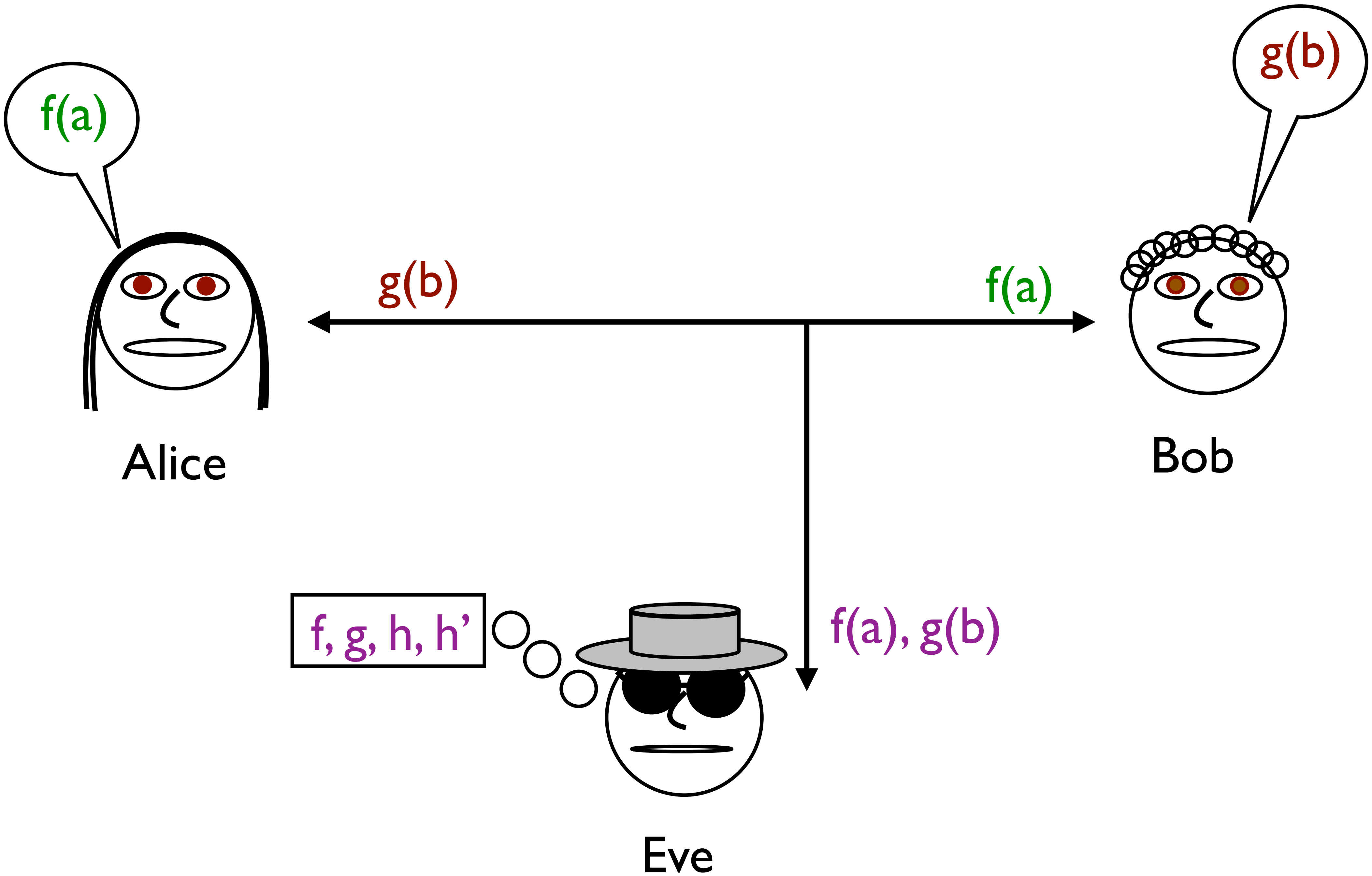


Transcript



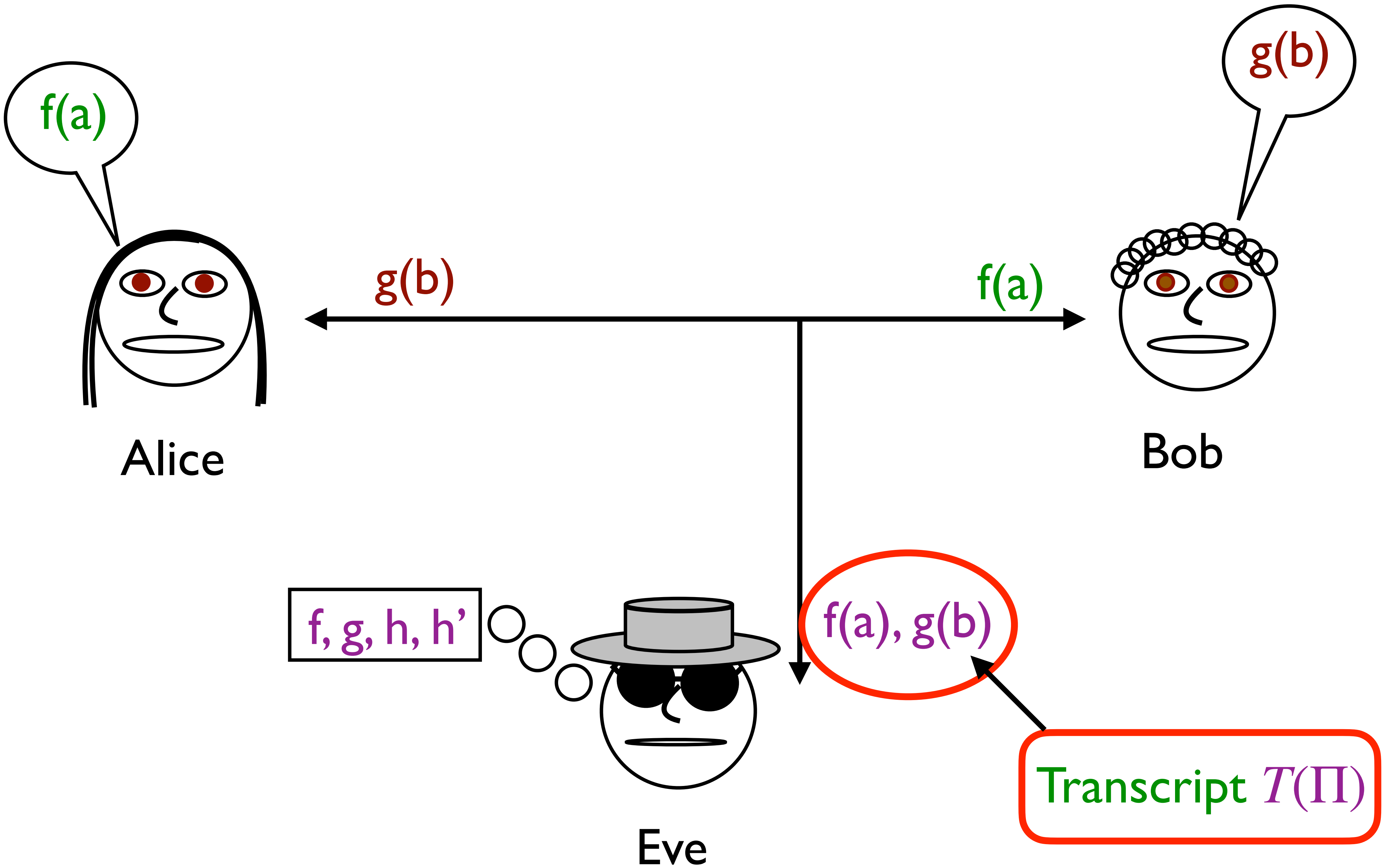
This class is being recorded

Transcript



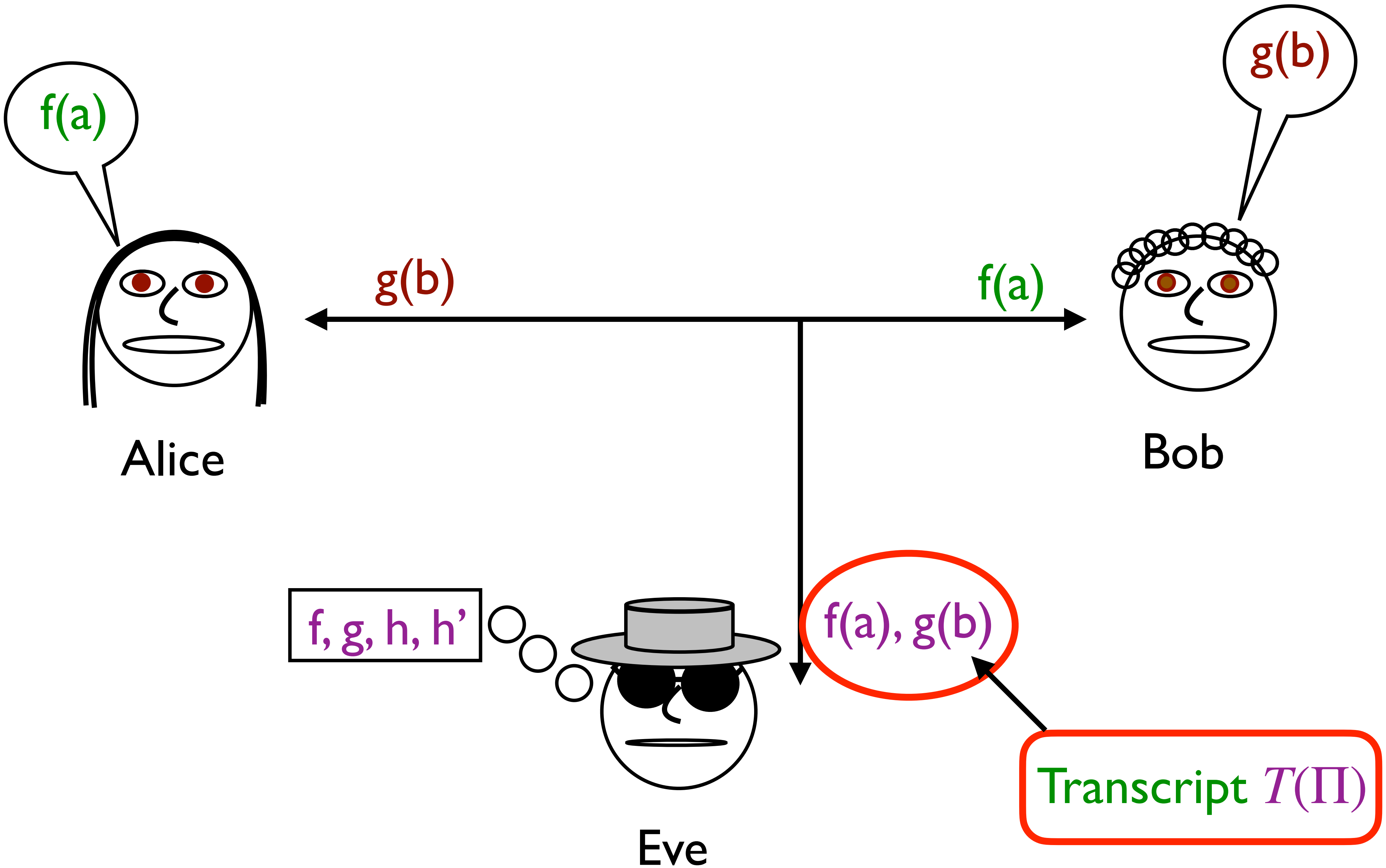
This class is being recorded

Transcript



This class is being recorded

Transcript

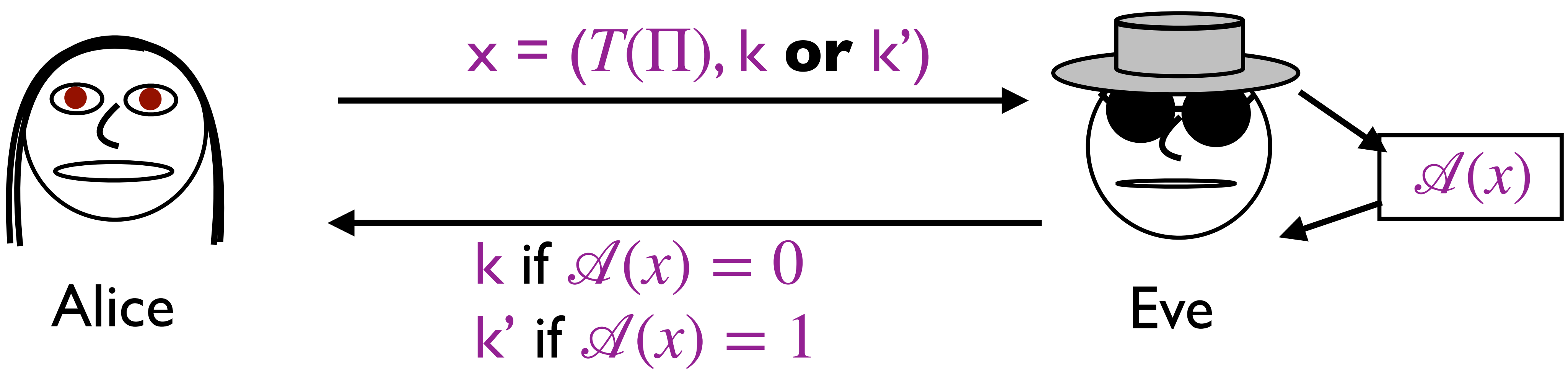


The transcript $T(\Pi)$ is all the information publicly announced during a run of the key exchange protocol Π.

This class is being recorded

Key Exchange Security Game

In the key exchange security game, Eve receives a transcript $T(\Pi)$ generated through a run of the key exchange protocol Π. Suppose that the actual key generated in the run that produced $T(\Pi)$ is k. Can Eve determine if the key is k or a random k'?



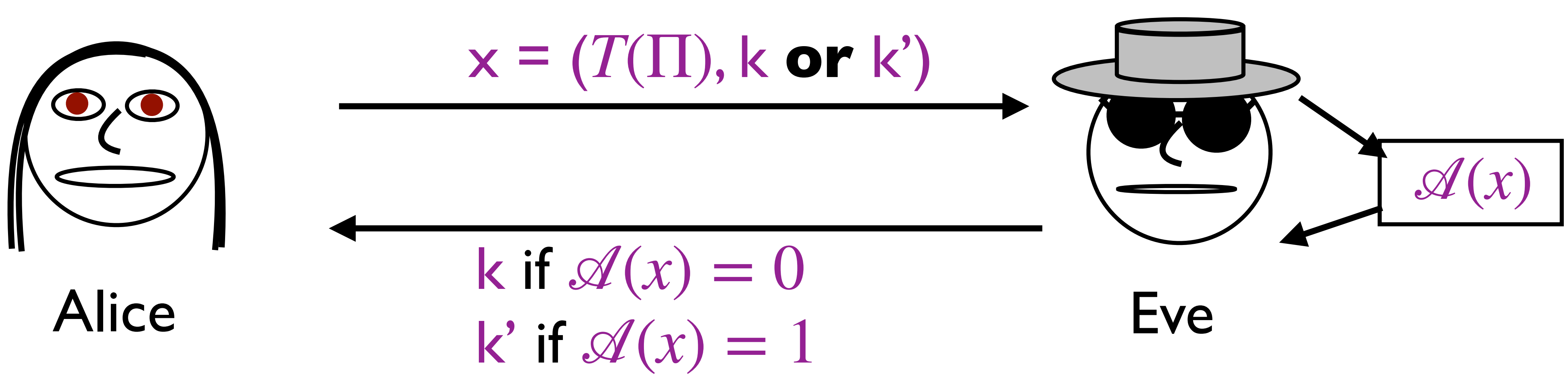
Alice sends Eve $T(\Pi)$ corresponding to key k, and then she either sends Eve k **or** she generates a random key k' and sends it to Eve instead. Eve must determine if the key she received is the one generated with the transcript she received.

Security Definition for Key Exchange

Definition: Consider a key exchange protocol Π. The **transcript** $T(\Pi)$ for the protocol is a full record of all public information announced during a run of the protocol. Suppose the protocol is run generating the key k and let k' be a uniformly randomly generated key. Then Π is **secure in the presence of an eavesdropper** if for all attacks $\mathscr{A}$ with a 1-bit output and taking as inputs a transcript $T(\Pi)$ and a key k or k',

$$|\Pr(\mathscr{A}(T(\Pi), k) = 1) - \Pr(\mathscr{A}(T(\Pi), k') = 1)| \leq \epsilon(s)$$

with $\epsilon(s)$ negligible and the probabilities averaged over k' and over the randomness of $\mathscr{A}$ and Π.



Why This Definition?

The definition says that the key generated by Alice and Bob looks the same to Eve as a random key, even when Eve has access to Alice and Bob's transcript.

- It is similar to the definition of security for a pseudorandom generator and for EAV-secure encryption. This means the key generated can be used the same way, e.g., in a pseudo one-time pad.
- In particular, we can prove a similar reduction to that for pseudorandom generators: Define a pseudo one-time pad protocol Π, which uses a key k, generated with the key exchange protocol. If Eve has an attack against Π, then Eve has an attack against the key exchange protocol.

Diffie-Hellman and Discrete Log

Proposition: If Diffie-Hellman is a secure key exchange protocol using modulus and base (p_s, g_s), then the discrete log problem is average-case hard for (p_s, g_s).

Diffie-Hellman and Discrete Log

Proposition: If Diffie-Hellman is a secure key exchange protocol using modulus and base (p_s, g_s), then the discrete log problem is average-case hard for (p_s, g_s).

Proof: By a reduction from the Diffie-Hellman decision problem to discrete log.

Diffie-Hellman and Discrete Log

Proposition: If Diffie-Hellman is a secure key exchange protocol using modulus and base (p_s, g_s), then the discrete log problem is average-case hard for (p_s, g_s).

Proof: By a reduction from the Diffie-Hellman decision problem to discrete log.

If we have an algorithm $\mathscr{A}(y)$ which succeeds in solving discrete log for (p_s, g_s) with non-negligible probability (for any y), we can use it to create an algorithm to find k in Diffie-Hellman using (p_s, g_s), also with non-negligible probability.

HW#6, problem 2b asks you to do this, essentially.

Diffie-Hellman and Discrete Log

Proposition: If Diffie-Hellman is a secure key exchange protocol using modulus and base (p_s, g_s), then the discrete log problem is average-case hard for (p_s, g_s).

Proof: By a reduction from the Diffie-Hellman decision problem to discrete log.

If we have an algorithm $\mathscr{A}(y)$ which succeeds in solving discrete log for (p_s, g_s) with non-negligible probability (for any y), we can use it to create an algorithm to find k in Diffie-Hellman using (p_s, g_s), also with non-negligible probability.

HW#6, problem 2b asks you to do this, essentially.

If you know the value of k implied by the transcript of Diffie-Hellman, you can easily distinguish k from random k'.

Diffie-Hellman and Discrete Log

Proposition: If Diffie-Hellman is a secure key exchange protocol using modulus and base (p_s, g_s), then the discrete log problem is average-case hard for (p_s, g_s).

Proof: By a reduction from the Diffie-Hellman decision problem to discrete log.

If we have an algorithm $\mathscr{A}(y)$ which succeeds in solving discrete log for (p_s, g_s) with non-negligible probability (for any y), we can use it to create an algorithm to find k in Diffie-Hellman using (p_s, g_s), also with non-negligible probability.

HW#6, problem 2b asks you to do this, essentially.

If you know the value of k implied by the transcript of Diffie-Hellman, you can easily distinguish k from random k'.

Therefore, if discrete log is easy, Diffie-Hellman is insecure. Or conversely, if Diffie-Hellman is secure, discrete log is hard.

Hardness of Diffie-Hellman

The reduction shows that discrete log is at least as hard as Diffie-Hellman. But can we show that Diffie-Hellman is exactly as hard as discrete log?

Hardness of Diffie-Hellman

The reduction shows that discrete log is at least as hard as Diffie-Hellman. But can we show that Diffie-Hellman is exactly as hard as discrete log?

No one knows how to do this.

Hardness of Diffie-Hellman

The reduction shows that discrete log is at least as hard as Diffie-Hellman. But can we show that Diffie-Hellman is exactly as hard as discrete log?

No one knows how to do this.

We would want to reduce discrete log to Diffie-Hellman. That is, given an attack $\mathscr{A}$ against Diffie-Hellman, use it to break discrete log.

Hardness of Diffie-Hellman

The reduction shows that discrete log is at least as hard as Diffie-Hellman. But can we show that Diffie-Hellman is exactly as hard as discrete log?

No one knows how to do this.

We would want to reduce discrete log to Diffie-Hellman. That is, given an attack $\mathscr{A}$ against Diffie-Hellman, use it to break discrete log.

The issue is that given $A = g^a \bmod N$ and $B = g^b \bmod N$, there might be a way to find $k = g^{ab} \bmod N$, or just to distinguish k from random k' without learning much about a or b. Maybe. We don't know.

Hardness of Diffie-Hellman

The reduction shows that discrete log is at least as hard as Diffie-Hellman. But can we show that Diffie-Hellman is exactly as hard as discrete log?

No one knows how to do this.

We would want to reduce discrete log to Diffie-Hellman. That is, given an attack $\mathscr{A}$ against Diffie-Hellman, use it to break discrete log.

The issue is that given $A = g^a \bmod N$ and $B = g^b \bmod N$, there might be a way to find $k = g^{ab} \bmod N$, or just to distinguish k from random k' without learning much about a or b. Maybe. We don't know.

Why do we care?

- Because discrete log is harder, it is more likely that is genuinely hard, so it is better to base security on that (a weaker assumption).
- Discrete log is a cleaner problem, easier to reuse in other cryptosystems.

Algorithms for Discrete Log

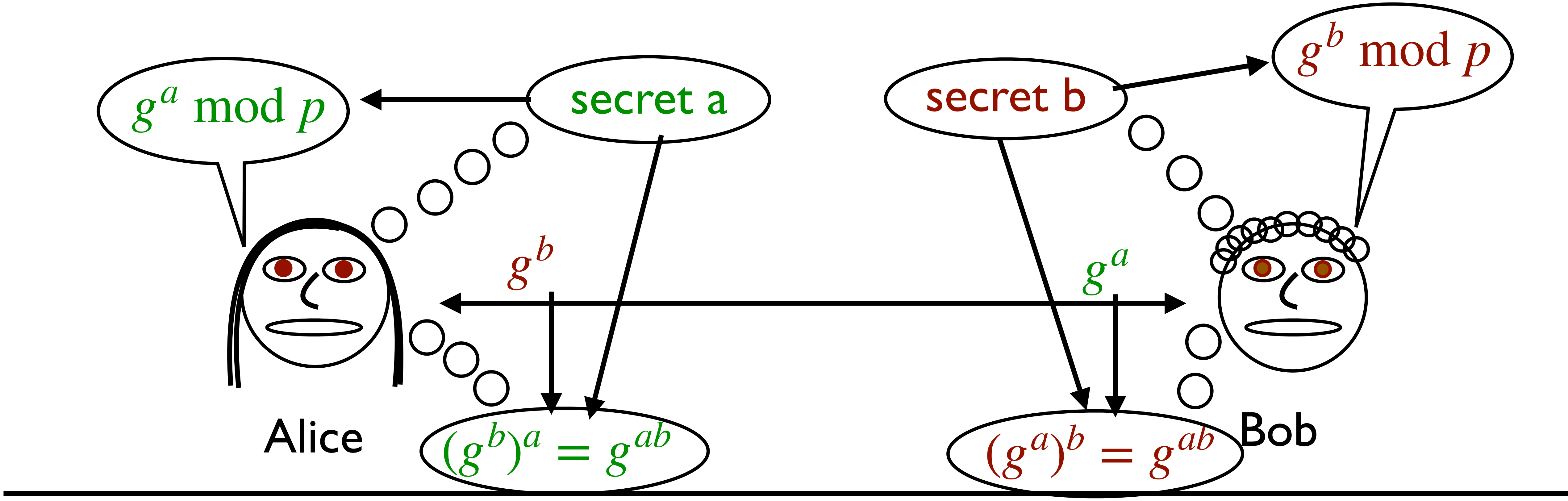
In practice, the best known algorithms for breaking Diffie-Hellman work by breaking discrete log.

- For poor choices of p and/or g, there are good algorithms (such as the Pohlig-Hellman algorithm we saw when p-1 is a product of small primes).
- For general $\mathbb{Z}_p^*$, the number field sieve runs in time $2^{O((\log p)^{1/3}(\log \log p)^{2/3})}$ (apparently). This is sub-exponential.
- No sub-exponential algorithm for Diffie-Hellman over elliptic curves is known.
- Except: A quantum computer can efficiently break discrete log over any abelian group, including elliptic curves.

Recommended key lengths:

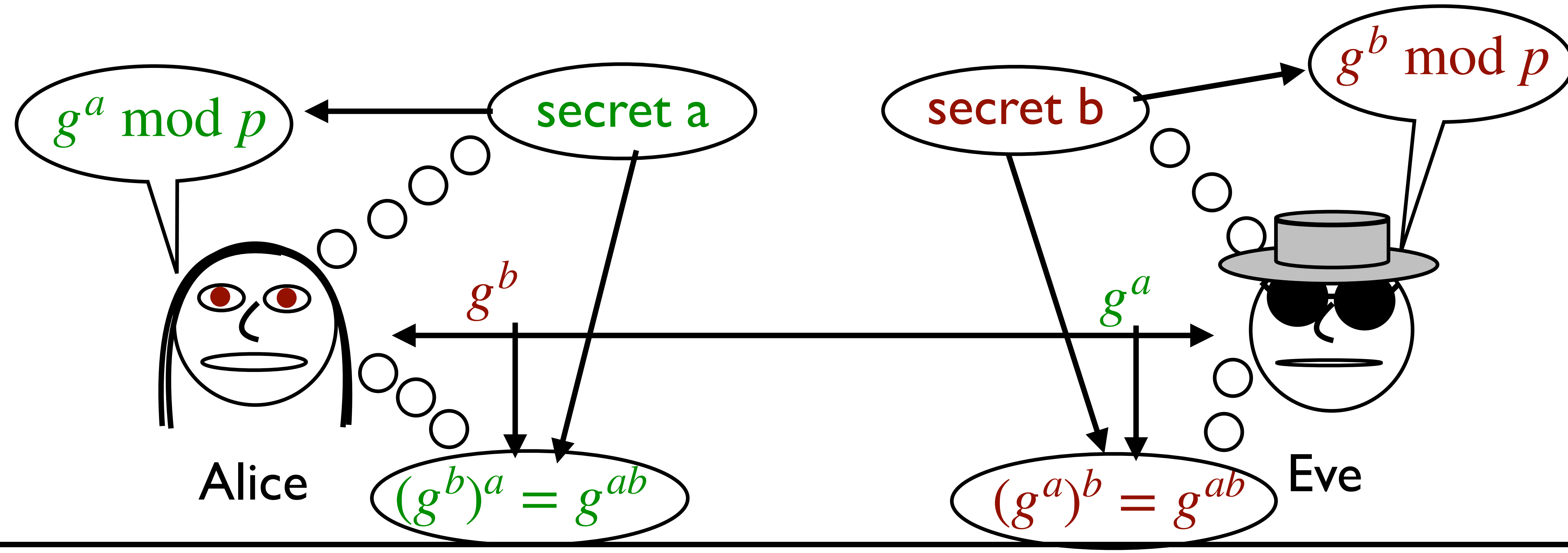
- Over $\mathbb{Z}_p^*$: use p of length 2048 bits or longer.
- Elliptic curves key length: 224 bits or higher.
- But don't use either if concerned about quantum attacks.

Another Attack on Diffie-Hellman



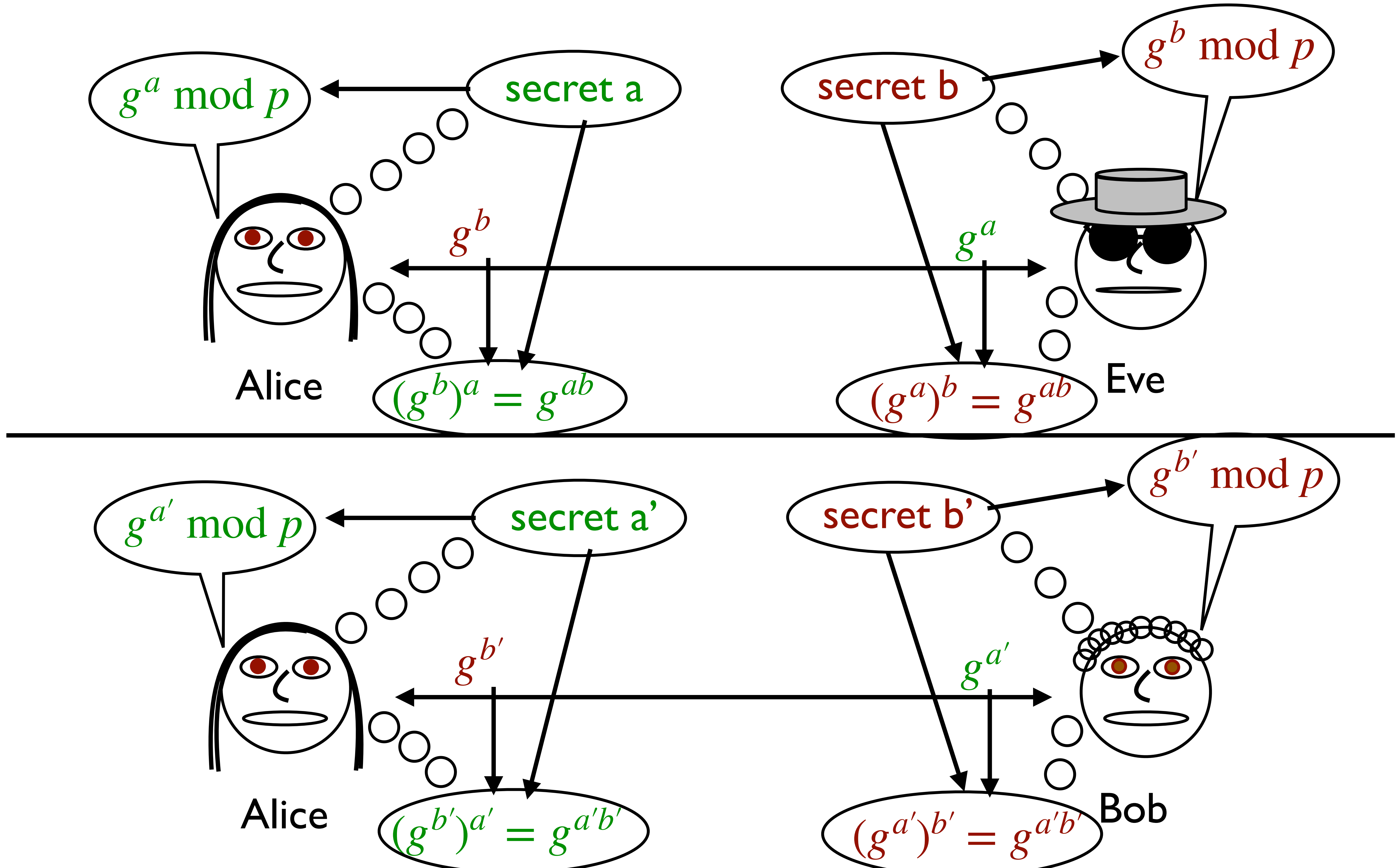
This class is being recorded

Another Attack on Diffie-Hellman



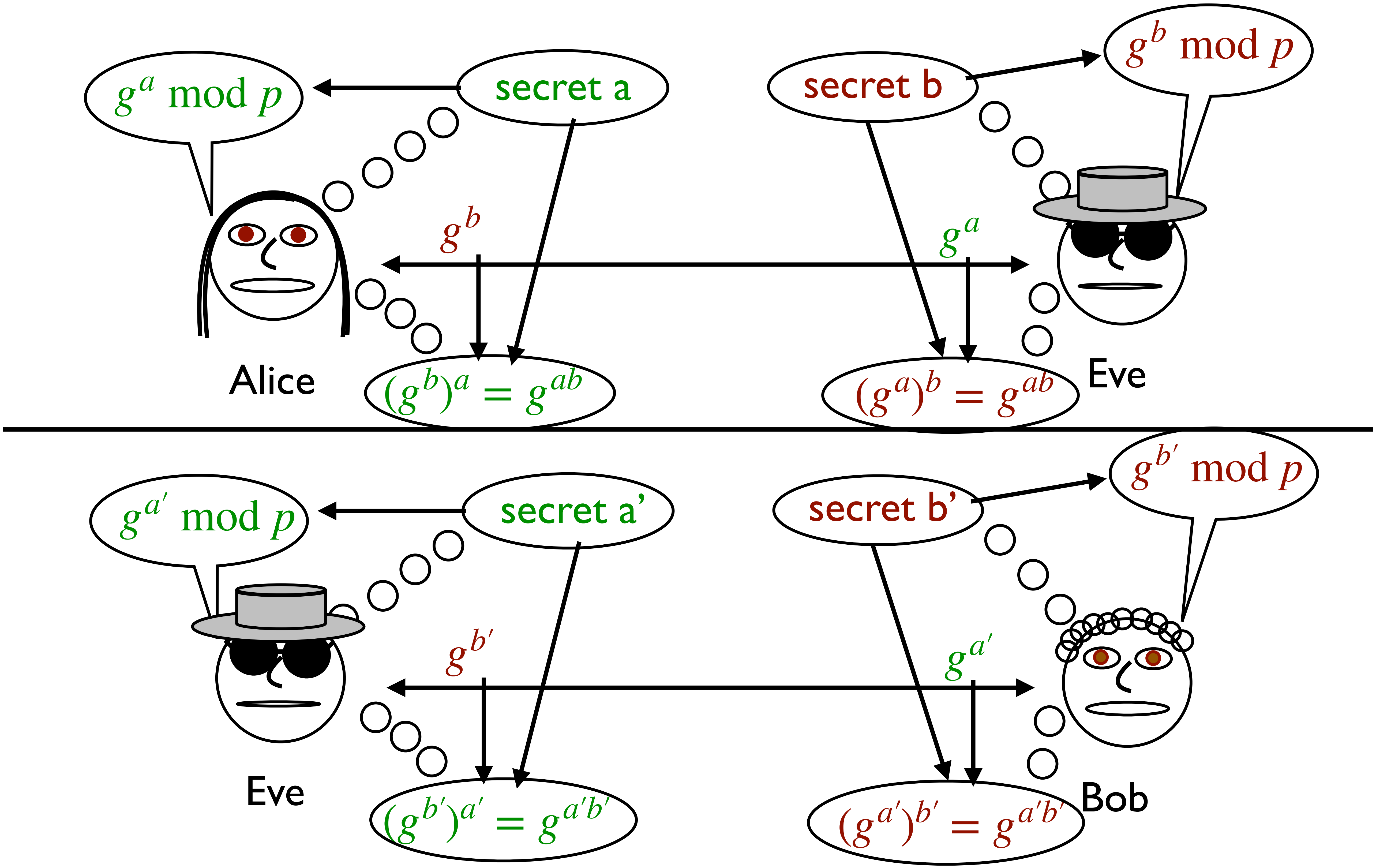
This class is being recorded

Another Attack on Diffie-Hellman



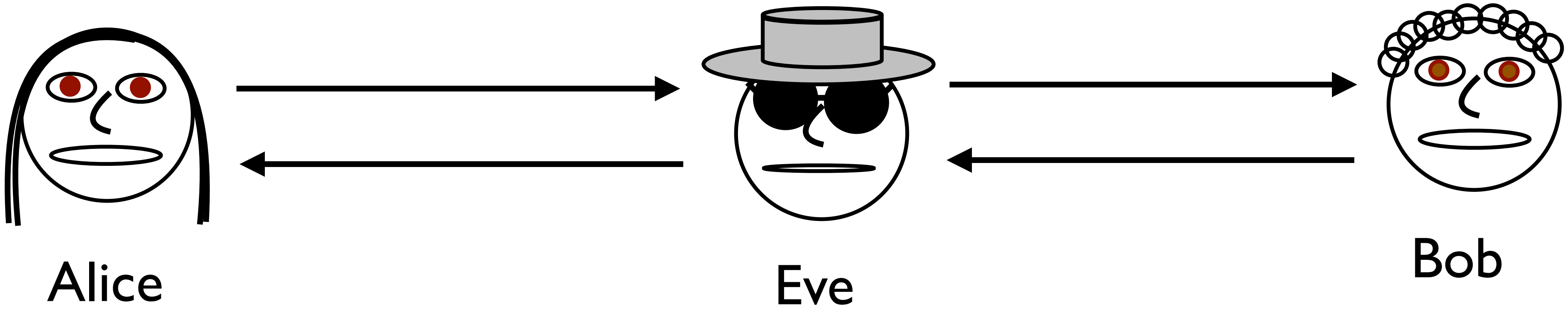
This class is being recorded

Another Attack on Diffie-Hellman



This class is being recorded

Man-in-the-Middle Attack



In a man-in-the-middle attack, Eve intercepts all communications between Alice and Bob and replaces them with messages of her choice. In Diffie-Hellman as we've discussed it, Alice and Bob have no way to fight this attack and Eve can read all their messages.

Alice and Bob need to authenticate their messages.

Diffie-Hellman and Encryption

Diffie-Hellman generates a key, a random element of a prime-order group G (a subgroup of $\mathbb{Z}_p^*$ or an elliptic curve).

How do we use it to encrypt?

We can use it as the key for a pseudo one-time pad.

But … it is not a bit string.

Diffie-Hellman and Encryption

Diffie-Hellman generates a key, a random element of a prime-order group G (a subgroup of $\mathbb{Z}_p^*$ or an elliptic curve).

How do we use it to encrypt?

We can use it as the key for a pseudo one-time pad.

But … it is not a bit string.

> Use some key derivation function H(k) to convert it into a bit string. H(k) needs to be carefully chosen to make sure the key still appears uniform.

Diffie-Hellman and Encryption

Diffie-Hellman generates a key, a random element of a prime-order group G (a subgroup of $\mathbb{Z}_p^*$ or an elliptic curve).

How do we use it to encrypt?

We can use it as the key for a pseudo one-time pad.

But … it is not a bit string.

> Use some key derivation function H(k) to convert it into a bit string. H(k) needs to be carefully chosen to make sure the key still appears uniform.
>
> Example: Key k is random number in $\{0,\ldots,96\}$. Write k in binary (7 bits) and let H(k) be the least significant 6 bits of k.

Diffie-Hellman and Encryption

Diffie-Hellman generates a key, a random element of a prime-order group G (a subgroup of $\mathbb{Z}_p^*$ or an elliptic curve).

How do we use it to encrypt?

We can use it as the key for a pseudo one-time pad.

But … it is not a bit string.

Use some key derivation function H(k) to convert it into a bit string. H(k) needs to be carefully chosen to make sure the key still appears uniform.

Example: Key k is random number in $\{0,\ldots,96\}$. Write k in binary (7 bits) and let H(k) be the least significant 6 bits of k.

Problem: About 1/3 of the time, $64 \leq k < 96$, which means the most significant bit of H(k) is 0. When $k < 64$, the first bit of H(k) is random. Overall, first bit is more likely to be 0!

Encryption With Diffie-Hellman

key generation

1. Alice and Bob choose their secrets a and b.
2. Alice and Bob compute and transmit $A = g^a \bmod p$ and $B = g^b \bmod p$, respectively. (Eve gets both.)
3. Alice and Bob compute $k = B^a = A^b \bmod p$.
4. Alice and Bob apply the key derivation function to get $\hat{k} = H(k)$.

message encryption and decryption

5. Alice wishes to send a message m. She encrypts to ciphertext $c = m \oplus \hat{k}$ and transmits c.
6. Bob (and Eve) receive c and Bob decrypts to $c \oplus \hat{k} = m$.

Encryption With Diffie-Hellman

key generation

1. Alice and Bob choose their secrets a and b.
2. Alice and Bob compute and transmit $A = g^a \bmod p$ and $B = g^b \bmod p$, respectively. (Eve gets both.)
3. Alice and Bob compute $k = B^a = A^b \bmod p$.
4. Alice and Bob apply the key derivation function to get $\hat{k} = H(k)$.

message encryption and decryption

5. Alice wishes to send a message m. She encrypts to ciphertext $c = m \oplus \hat{k}$ and transmits c.
6. Bob (and Eve) receive c and Bob decrypts to $c \oplus \hat{k} = m$.

Question: Is this EAV-secure or CPA-secure?

Encryption With Diffie-Hellman

key generation

1. Alice and Bob choose their secrets a and b.
2. Alice and Bob compute and transmit $A = g^a \bmod p$ and $B = g^b \bmod p$, respectively. (Eve gets both.)
3. Alice and Bob compute $k = B^a = A^b \bmod p$.
4. Alice and Bob apply the key derivation function to get $\hat{k} = H(k)$.

message encryption and decryption

5. Alice wishes to send a message m. She encrypts to ciphertext $c = m \oplus \hat{k}$ and transmits c.
6. Bob (and Eve) receive c and Bob decrypts to $c \oplus \hat{k} = m$.

Question: Is this EAV-secure or CPA-secure?

Answer: It is EAV-secure since the key can only be used once. But Alice and Bob can easily run the key generation phase again to send a new message.

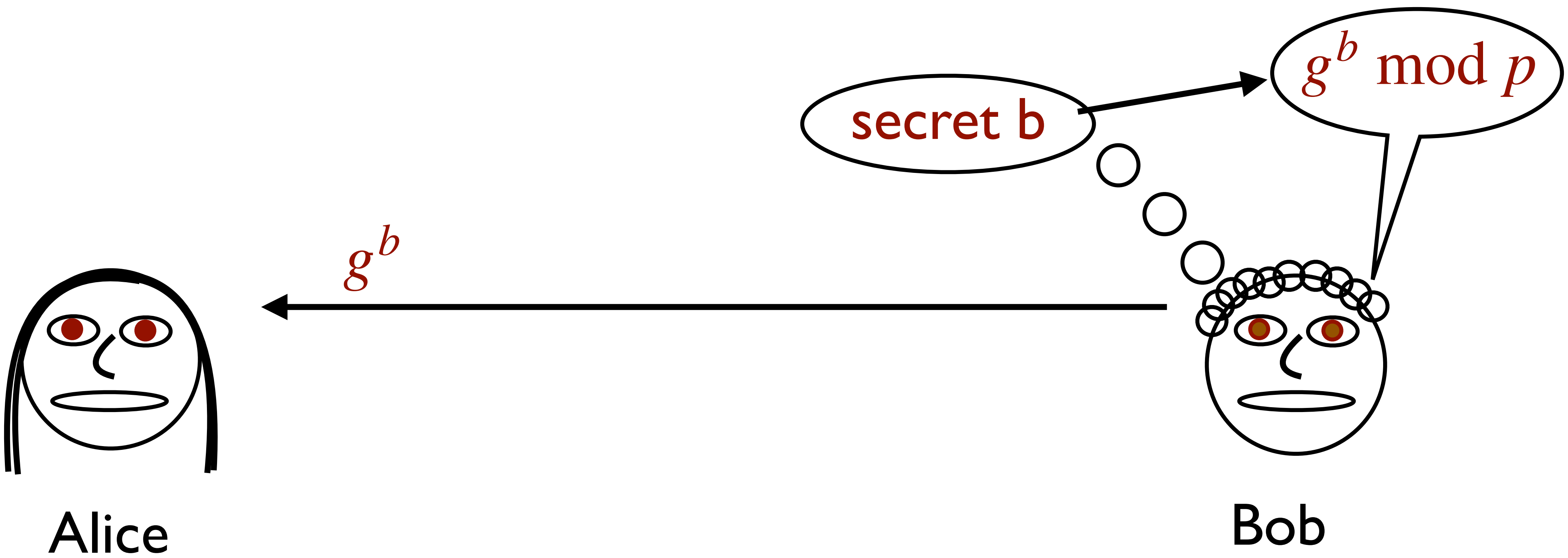
Another Approach to Encryption

Another approach to encryption is to integrate encryption into the key exchange process.

To see how to do this, the first step is notice that Alice's and Bob's announcements in Diffie-Hellman don't have to be simultaneous.

Another Approach to Encryption

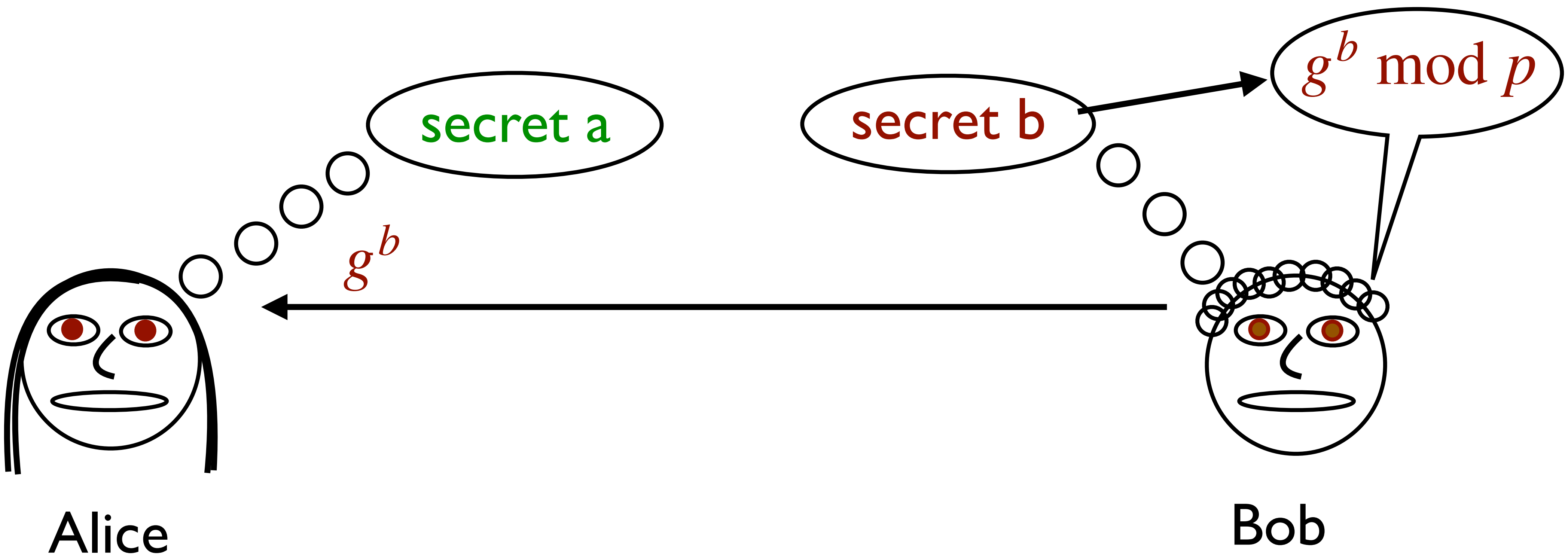
Another approach to encryption is to integrate encryption into the key exchange process.



To see how to do this, the first step is notice that Alice's and Bob's announcements in Diffie-Hellman don't have to be simultaneous.

Another Approach to Encryption

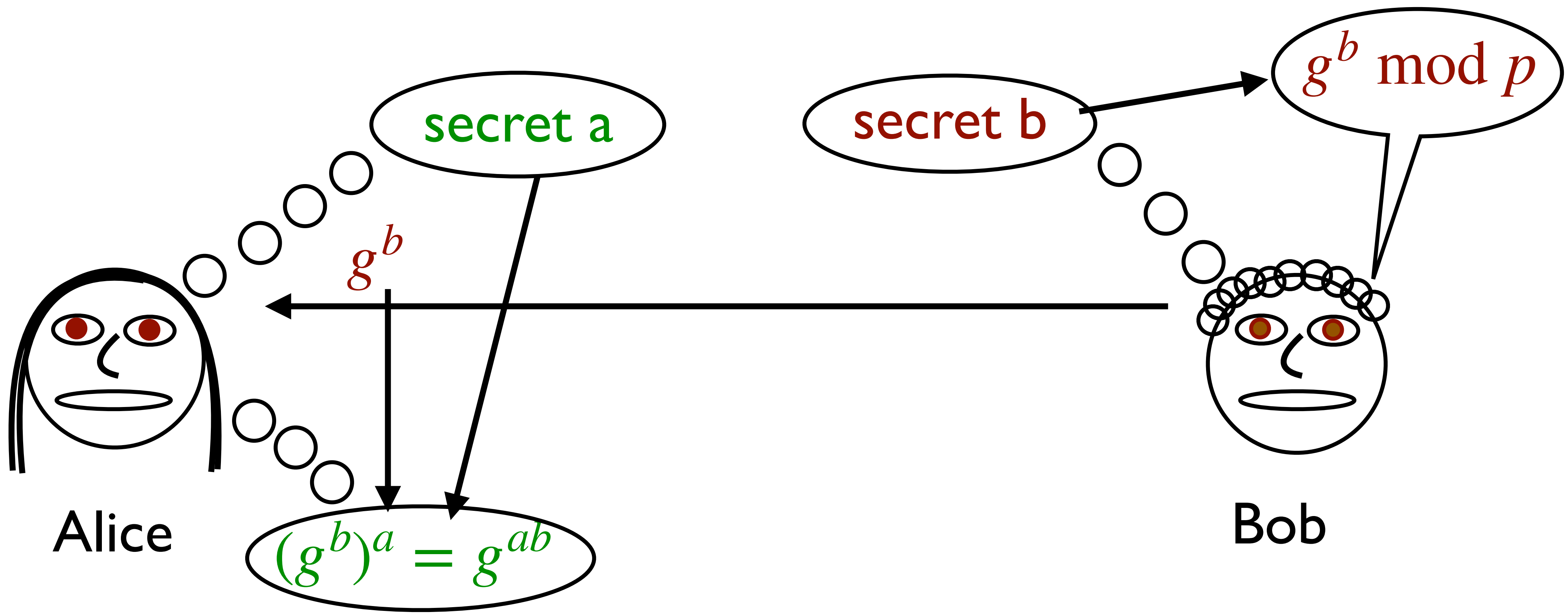
Another approach to encryption is to integrate encryption into the key exchange process.



To see how to do this, the first step is notice that Alice's and Bob's announcements in Diffie-Hellman don't have to be simultaneous.

Another Approach to Encryption

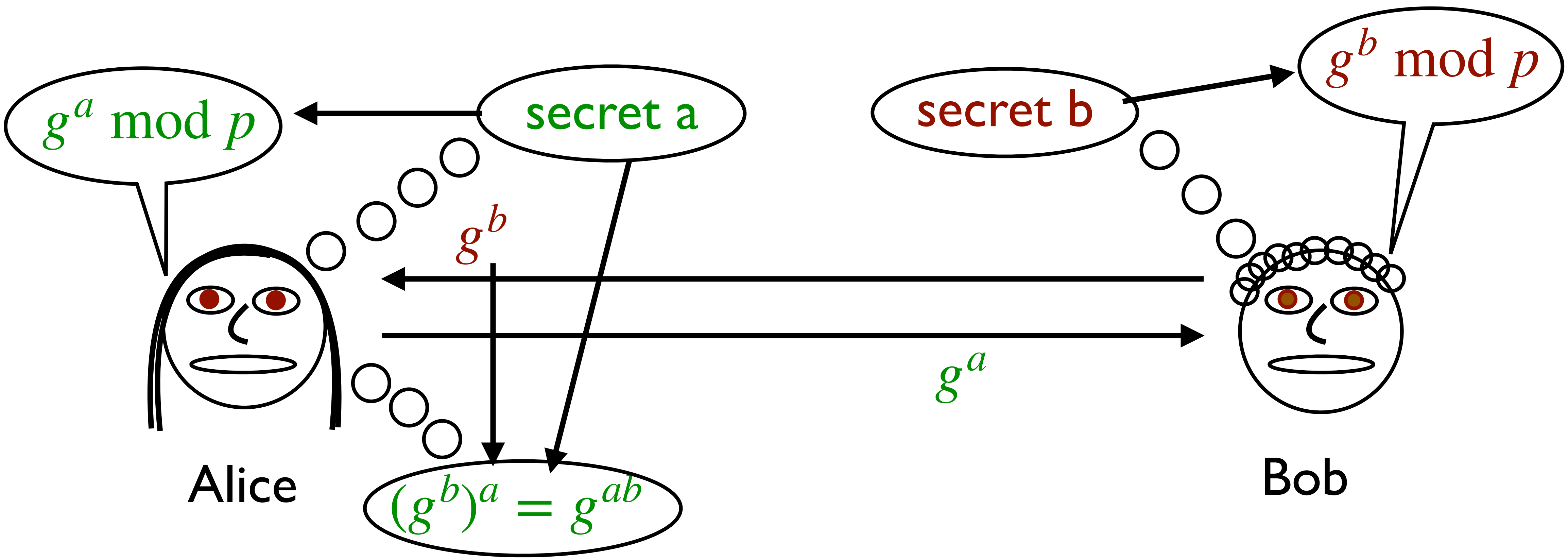
Another approach to encryption is to integrate encryption into the key exchange process.



To see how to do this, the first step is notice that Alice's and Bob's announcements in Diffie-Hellman don't have to be simultaneous.

Another Approach to Encryption

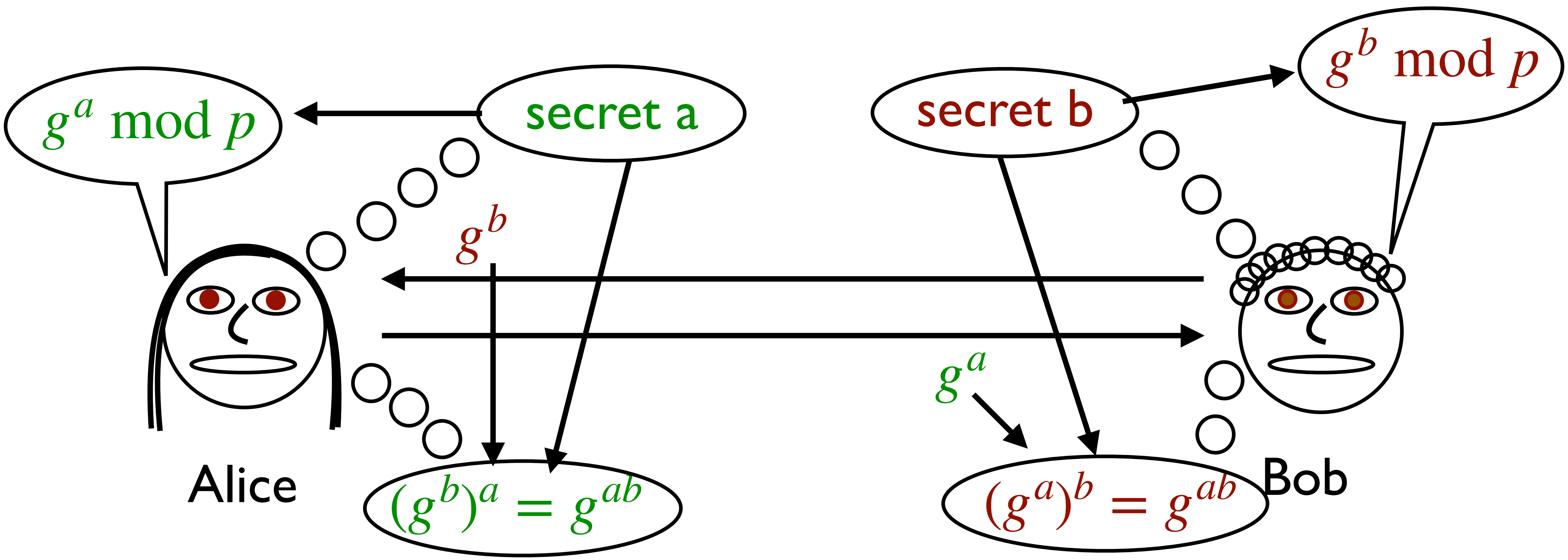
Another approach to encryption is to integrate encryption into the key exchange process.



To see how to do this, the first step is notice that Alice's and Bob's announcements in Diffie-Hellman don't have to be simultaneous.

Another Approach to Encryption

Another approach to encryption is to integrate encryption into the key exchange process.



To see how to do this, the first step is notice that Alice's and Bob's announcements in Diffie-Hellman don't have to be simultaneous.

El Gamal Encryption

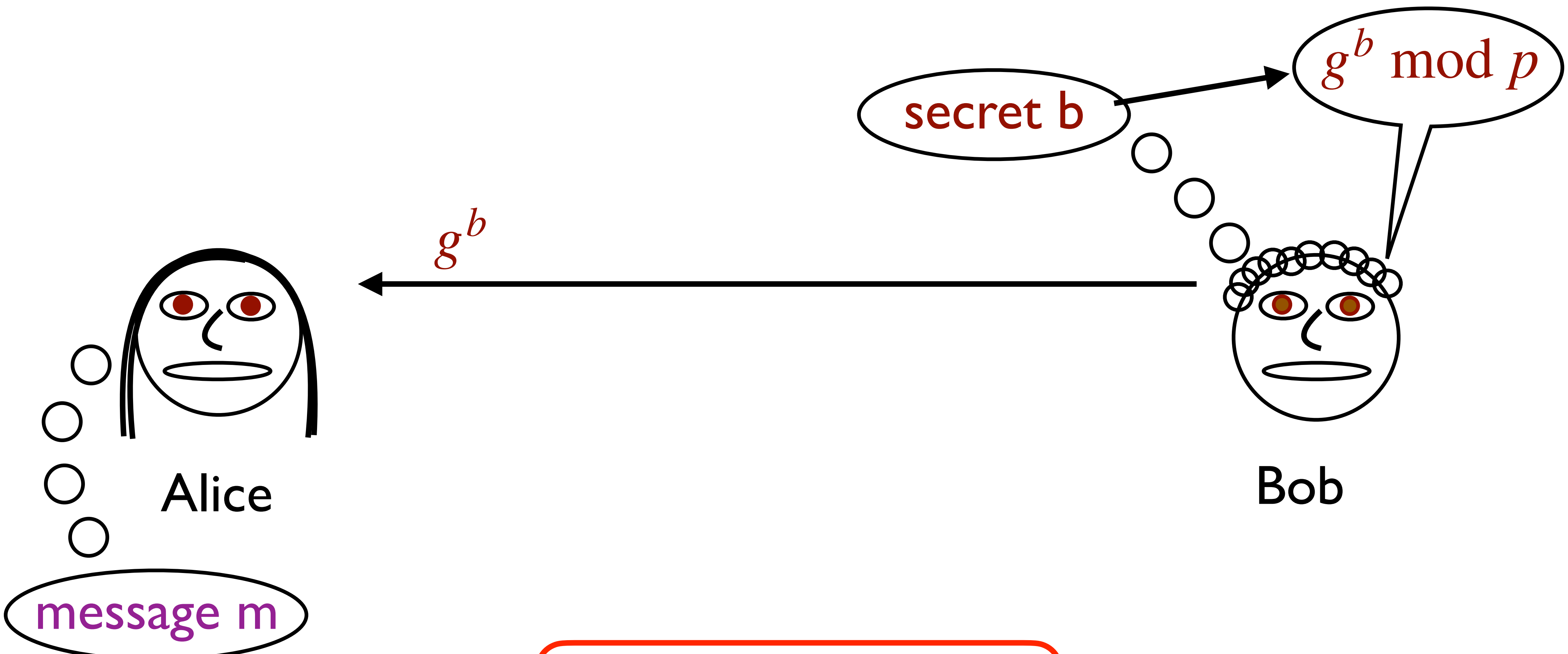
When Alice sends g^a, she can also send an encrypted message m, for instance with ciphertext $c = m \cdot g^{ab}$.

Bob's public key is g^b.

Advantage over Diffie-Hellman: each phase is non-interactive

El Gamal Encryption

When Alice sends g^a, she can also send an encrypted message m, for instance with ciphertext $c = m \cdot g^{ab}$.

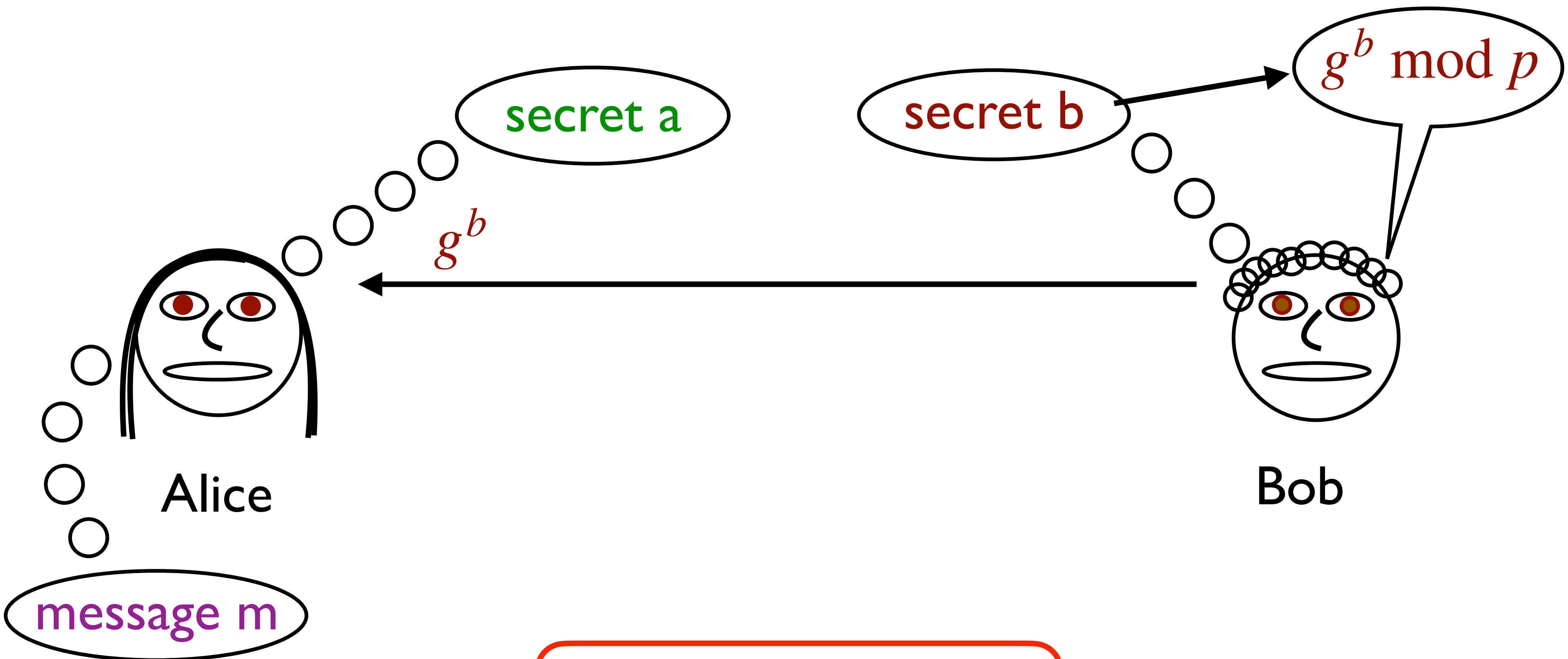


Bob's public key is g^b.

Advantage over Diffie-Hellman: each phase is non-interactive

El Gamal Encryption

When Alice sends g^a, she can also send an encrypted message m, for instance with ciphertext $c = m \cdot g^{ab}$.



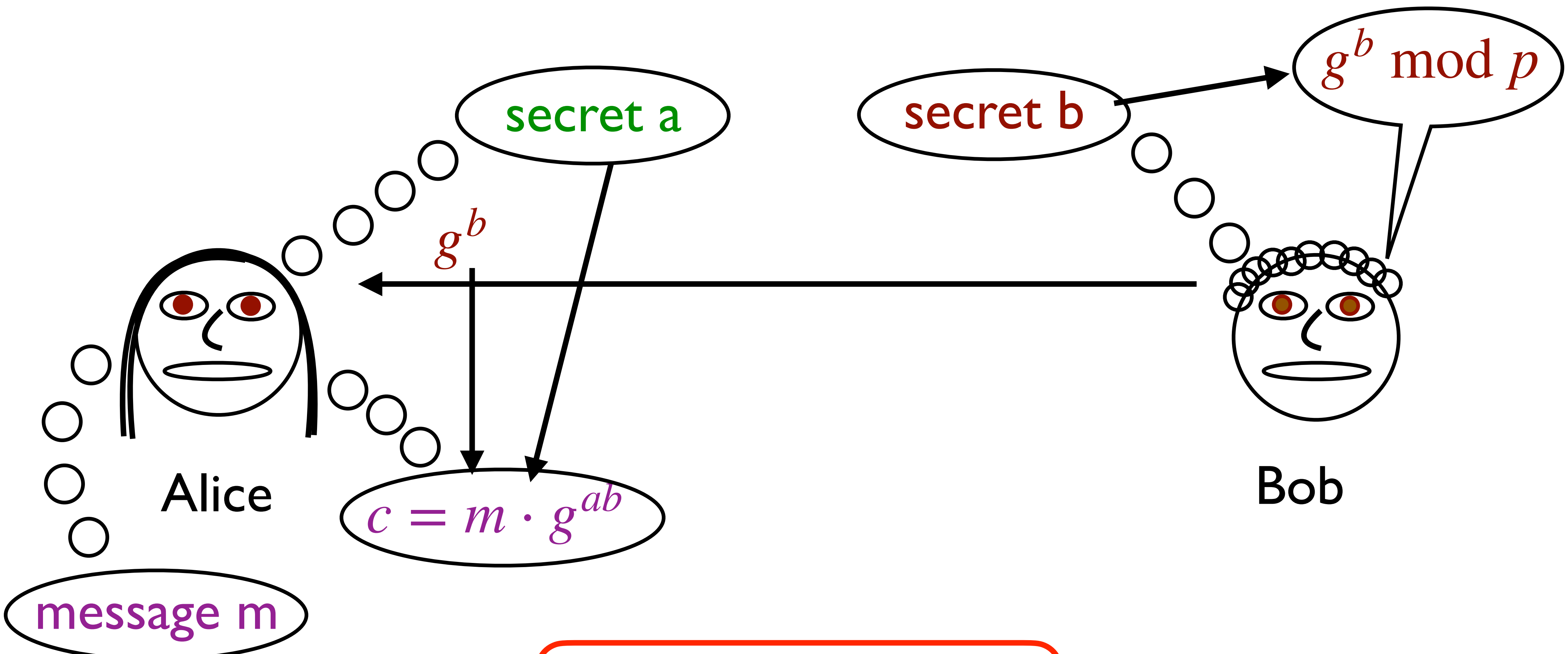
Bob's public key is g^b.

Advantage over Diffie-Hellman: each phase is non-interactive

This class is being recorded

El Gamal Encryption

When Alice sends g^a, she can also send an encrypted message m, for instance with ciphertext $c = m \cdot g^{ab}$.

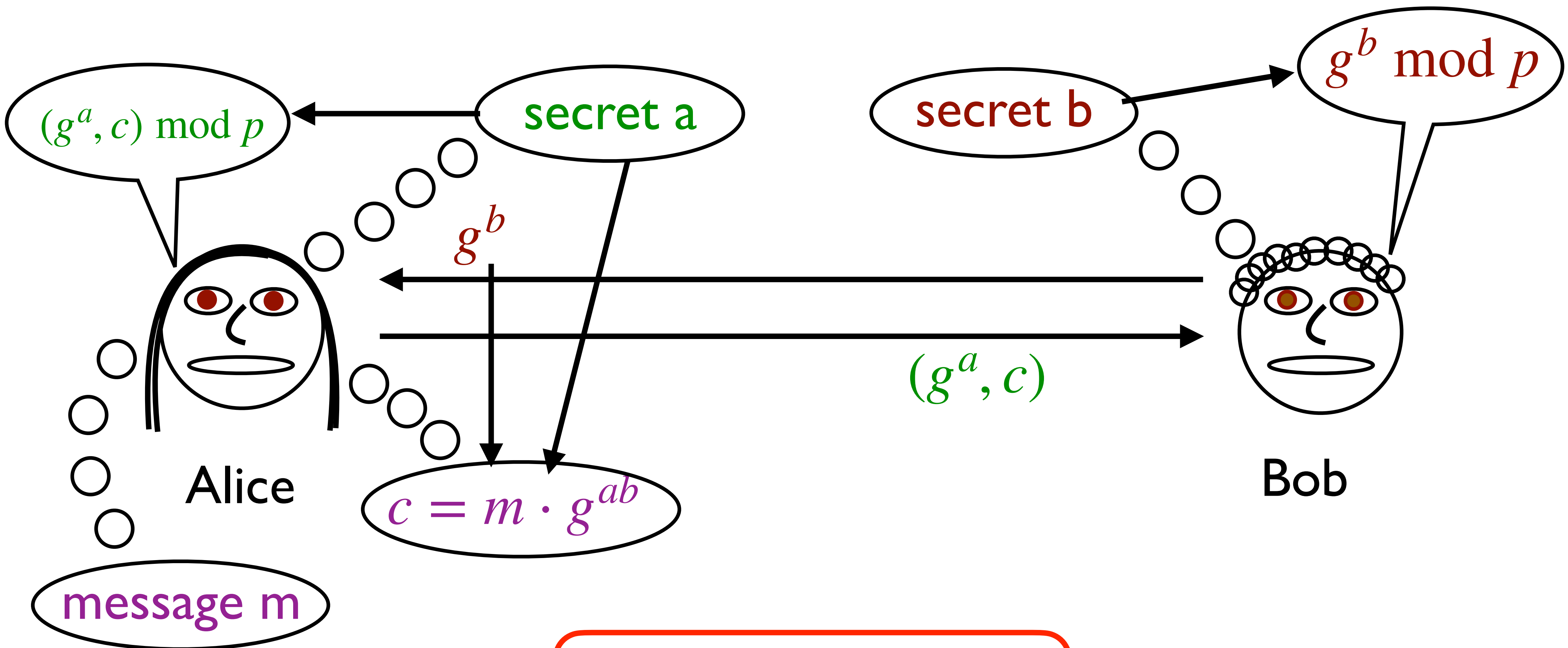


Bob's public key is g^b.

Advantage over Diffie-Hellman: each phase is non-interactive

El Gamal Encryption

When Alice sends g^a, she can also send an encrypted message m, for instance with ciphertext $c = m \cdot g^{ab}$.



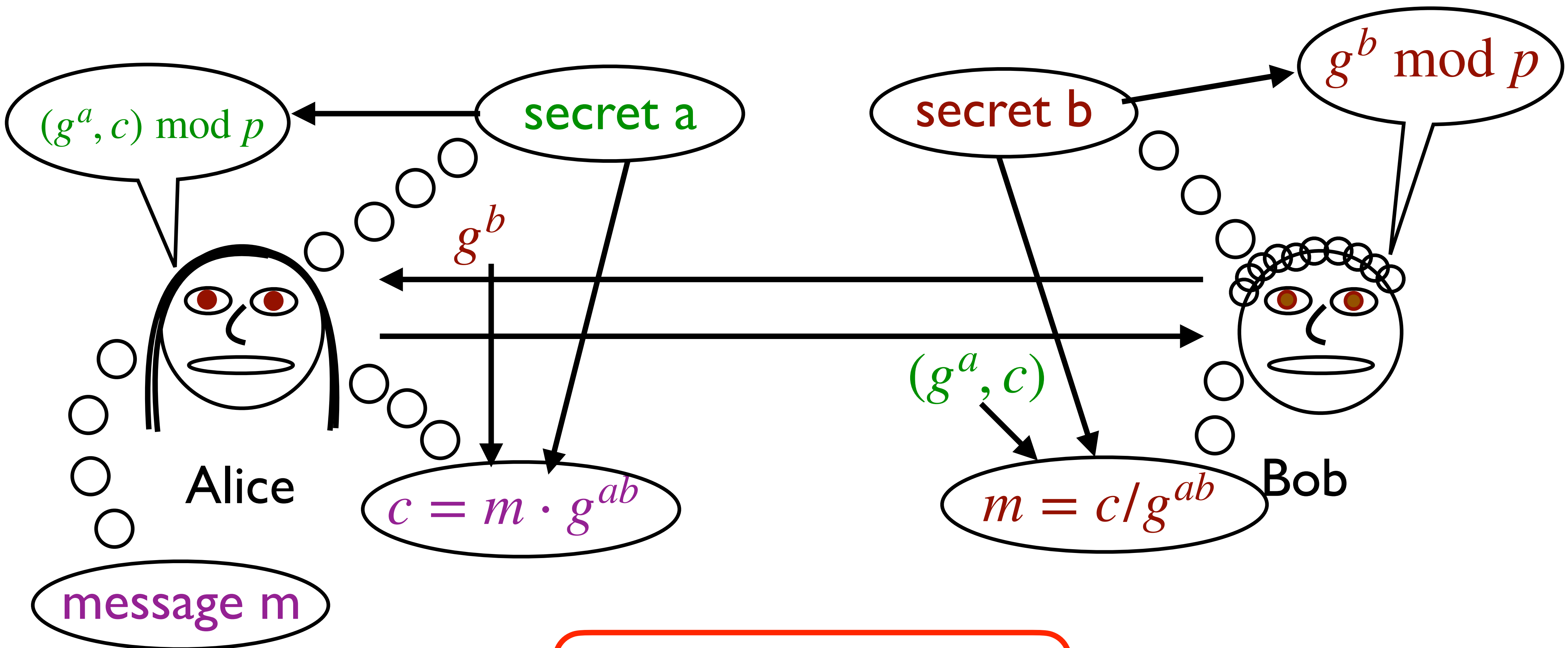
Bob's public key is g^b.

Advantage over Diffie-Hellman: each phase is non-interactive

This class is being recorded

El Gamal Encryption

When Alice sends g^a, she can also send an encrypted message m, for instance with ciphertext $c = m \cdot g^{ab}$.



Bob's public key is g^b.

Advantage over Diffie-Hellman: each phase is non-interactive

This class is being recorded

El Gamal Encryption

key generation

1. A prime p and base g are chosen as part of the protocol, as with Diffie-Hellman.
2. Bob picks a random value b as his private key.
3. Bob computes $B = g^b \bmod p$. This is his public key. Bob announces it, and Alice receives it (but so does Eve).

message encryption and decryption

4. Alice wishes to send a message m. She picks a secret random value a. Alice computes $A = g^a \bmod p$ and $c = m \cdot B^a \bmod p$.
5. Alice transmits (A, c) (but *not* a). Bob receives it, as does Eve.
6. Bob computes $c/A^b \bmod p$. This is his decryption of the message.

El Gamal Encryption

key generation

1. A prime p and base g are chosen as part of the protocol, as with Diffie-Hellman.
2. Bob picks a random value b as his private key.
3. Bob computes $B = g^b \bmod p$. This is his public key. Bob announces it, and Alice receives it (but so does Eve).

message encryption and decryption

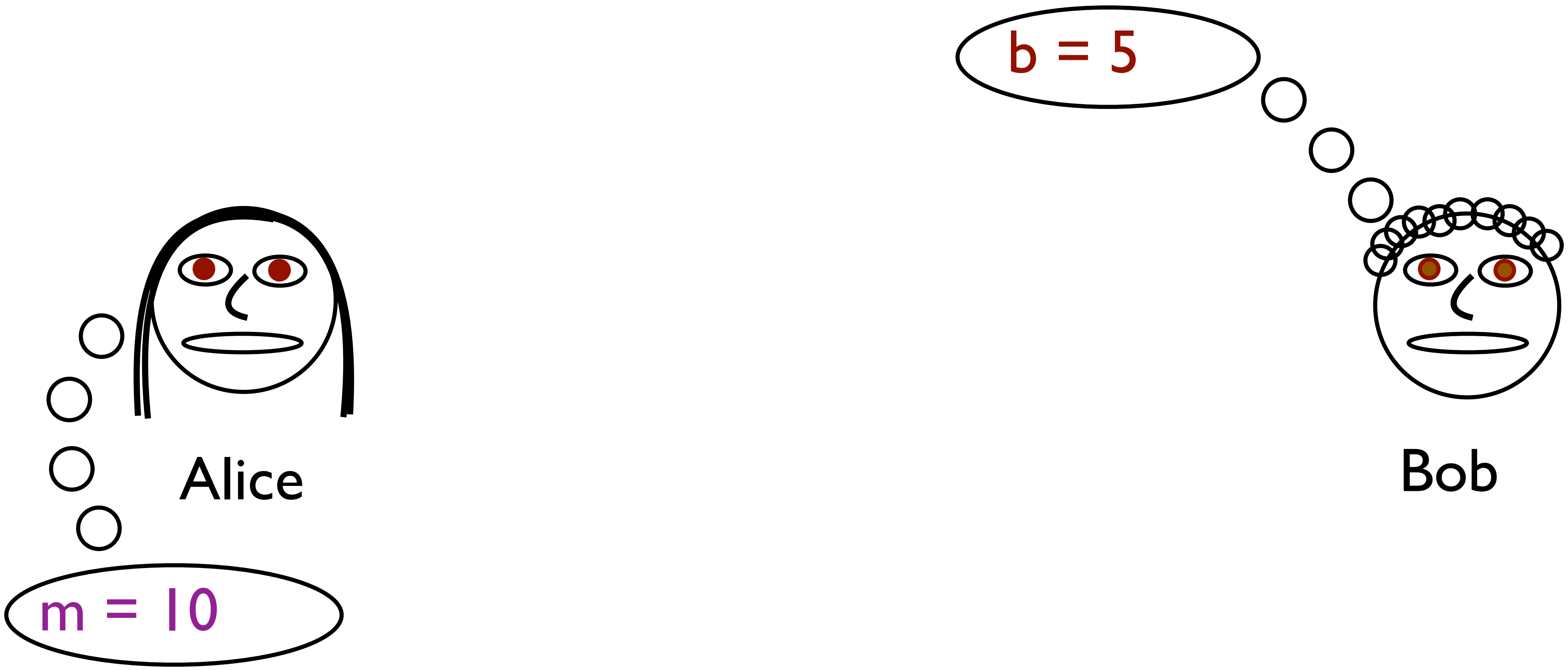
4. Alice wishes to send a message m. She picks a secret random value a. Alice computes $A = g^a \bmod p$ and $c = m \cdot B^a \bmod p$.
5. Alice transmits (A, c) (but *not* a). Bob receives it, as does Eve.
6. Bob computes $c/A^b \bmod p$. This is his decryption of the message.

Since $c/A^b = mB^a/A^b = mg^{ab}/g^{ab} = m \bmod p$

this protocol is correct.

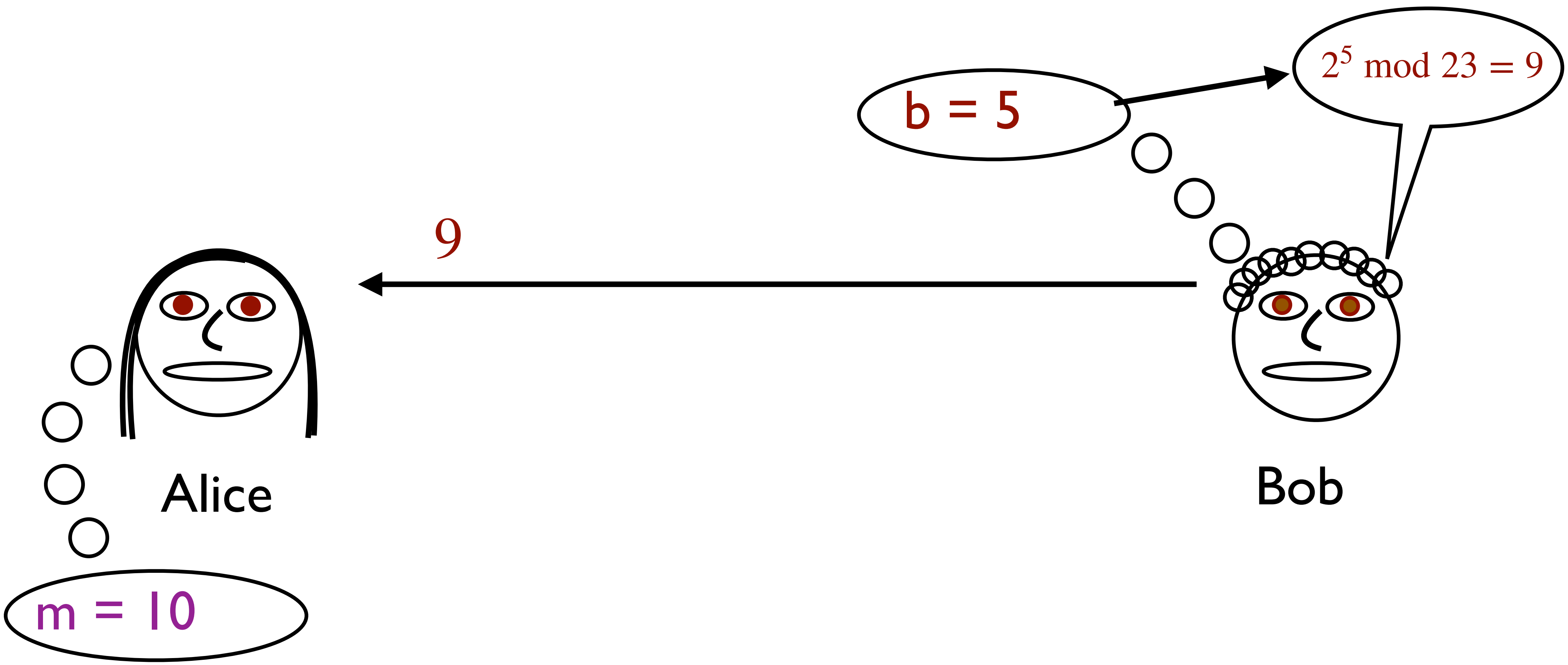
El Gamal Example

Example using p = 23, g = 2. (g has order 11 in $\mathbb{Z}_{23}^*$.)



El Gamal Example

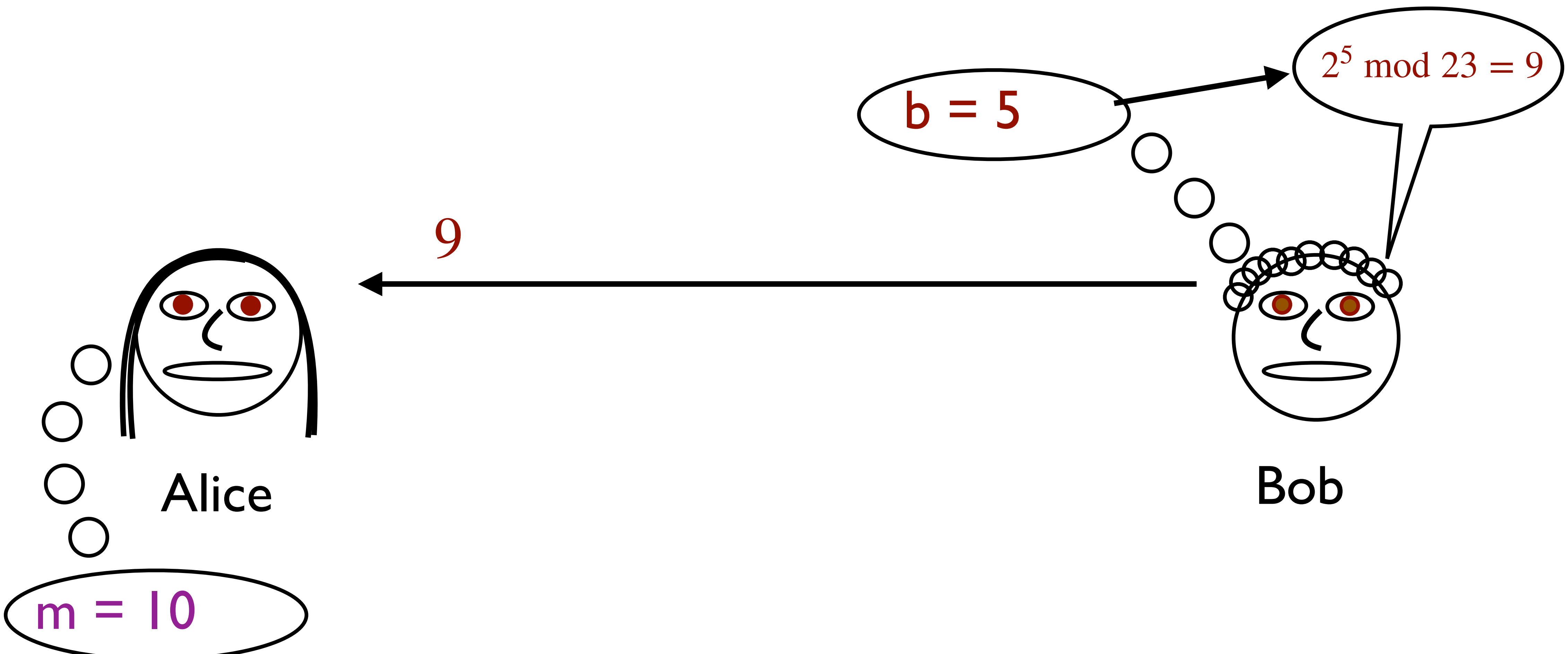
Example using $p = 23$, $g = 2$. (g has order 11 in $\mathbb{Z}_{23}^*$.)



This class is being recorded

El Gamal Example

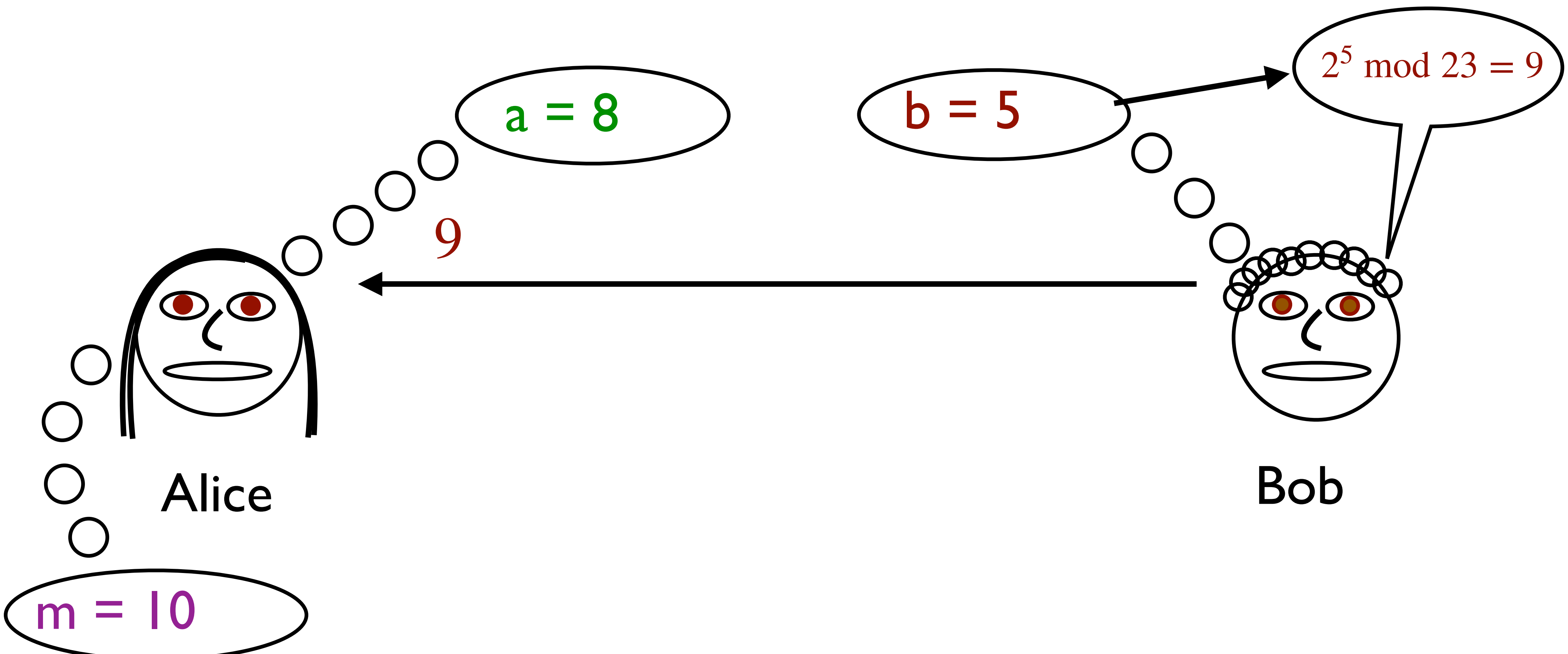
Example using $p = 23$, $g = 2$. (g has order 11 in $\mathbb{Z}_{23}^*$.)



In this example, Bob's public key is 9. Alice uses it to encrypt.

El Gamal Example

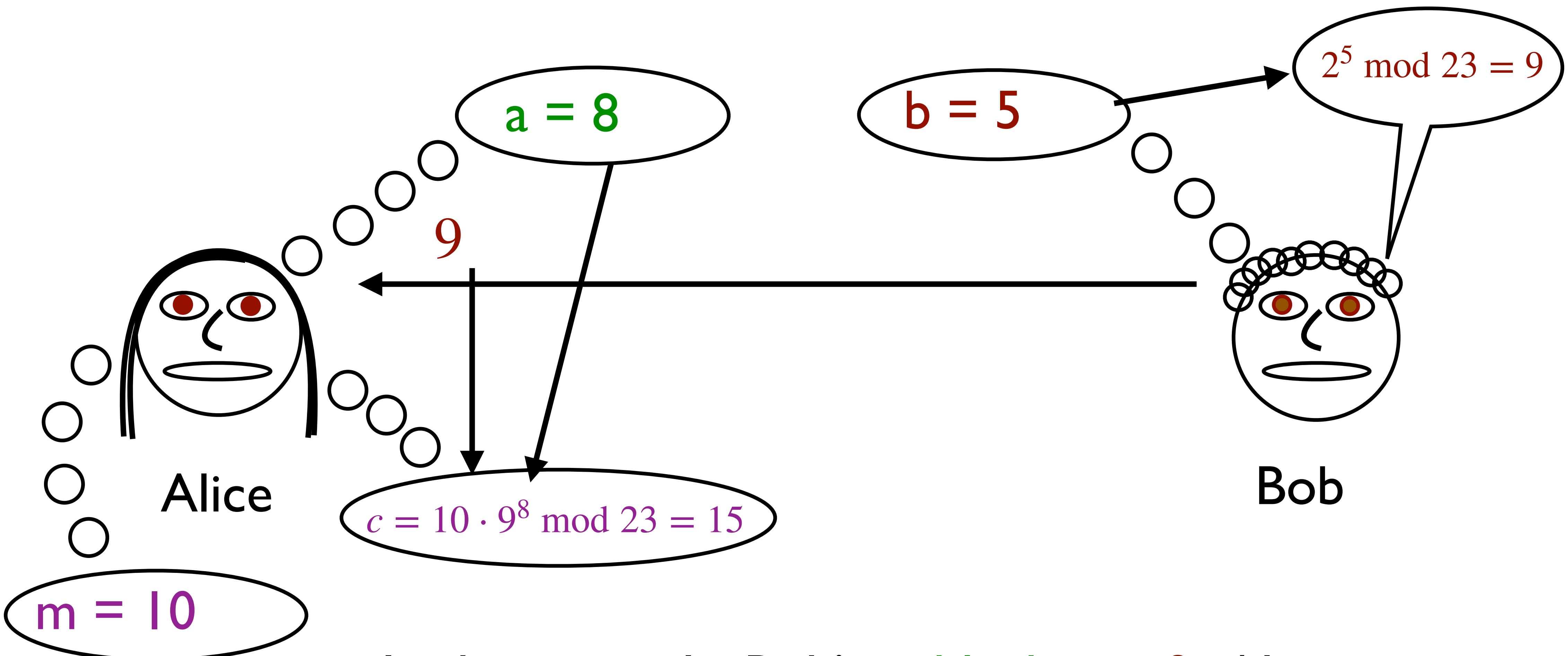
Example using $p = 23, g = 2$. (g has order 11 in $\mathbb{Z}_{23}^*$.)



In this example, Bob's public key is 9. Alice uses it to encrypt.

El Gamal Example

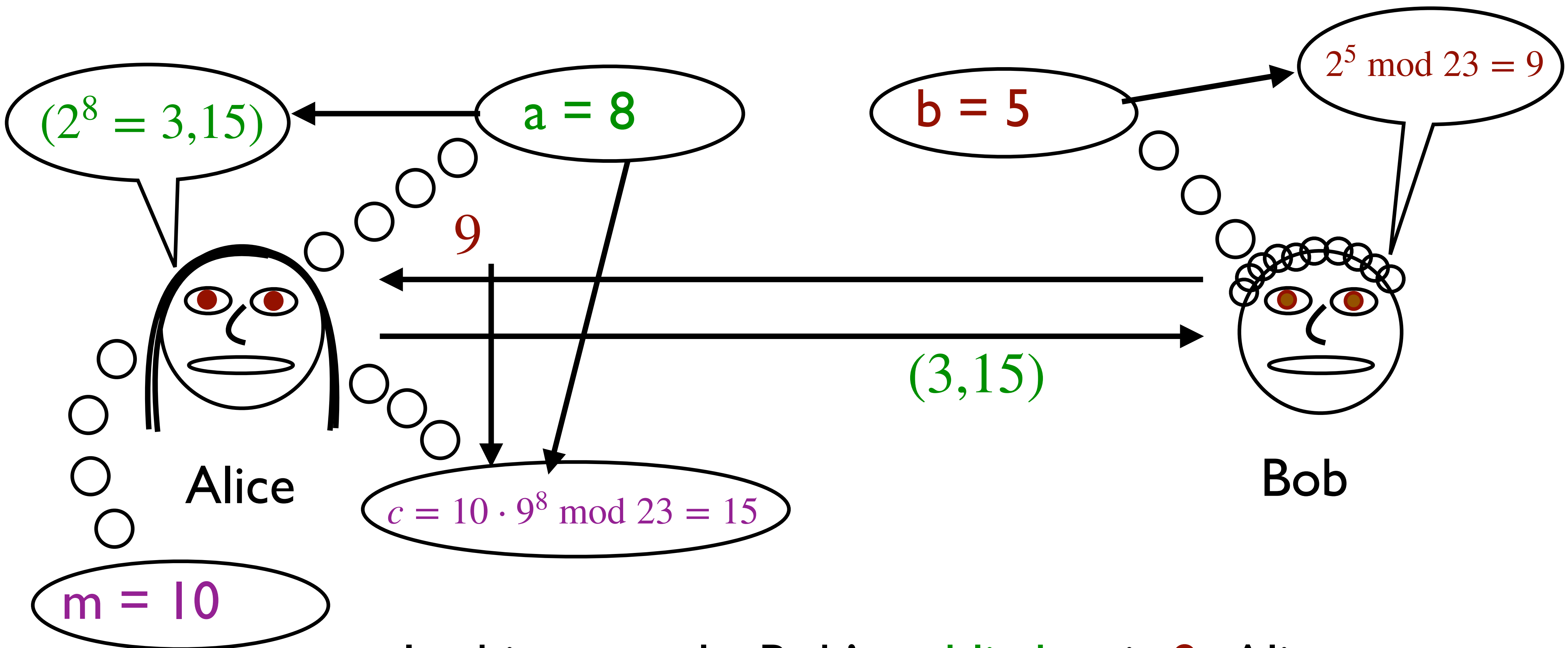
Example using p = 23, g = 2. (g has order 11 in $\mathbb{Z}_{23}^*$.)



In this example, Bob's public key is 9. Alice uses it to encrypt.

El Gamal Example

Example using p = 23, g = 2. (g has order 11 in $\mathbb{Z}^*_{23}$.)

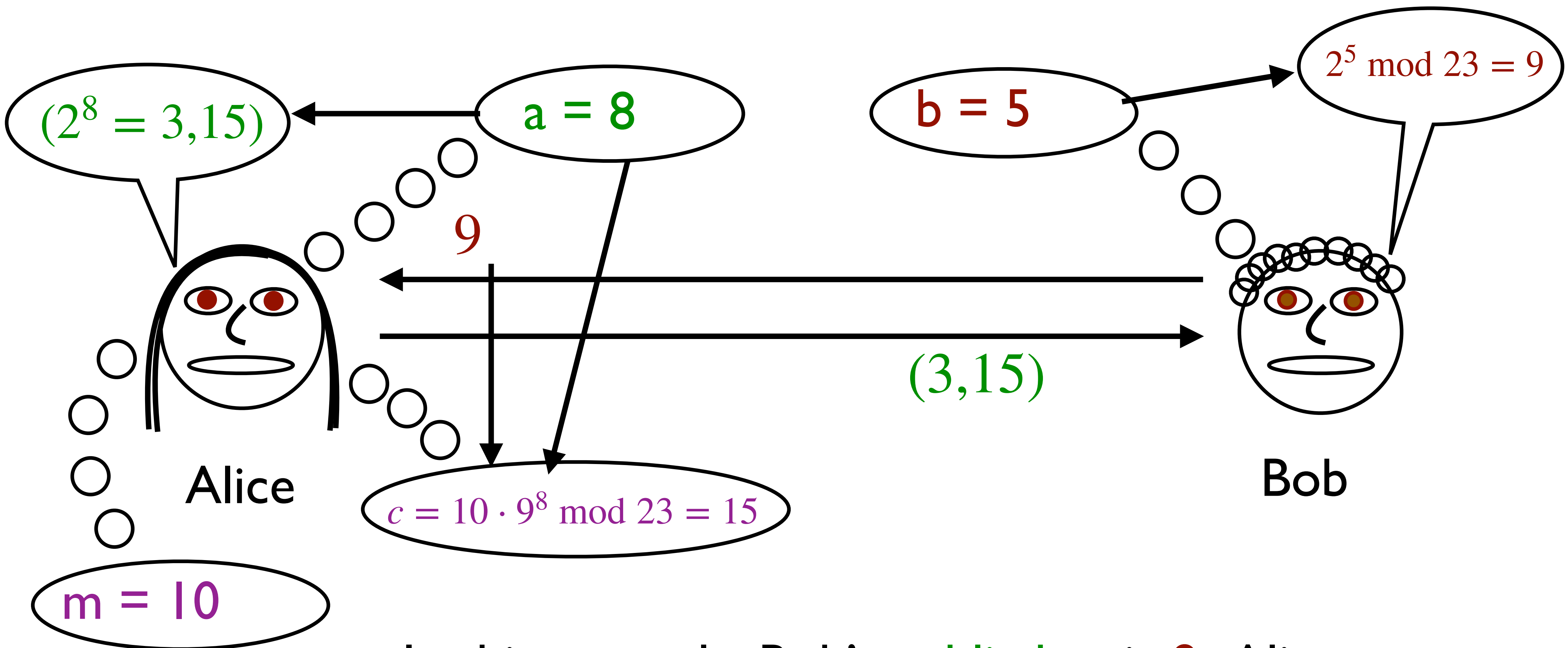


In this example, Bob's public key is 9. Alice uses it to encrypt.

This class is being recorded

El Gamal Example

Example using p = 23, g = 2. (g has order 11 in $\mathbb{Z}_{23}^*$.)

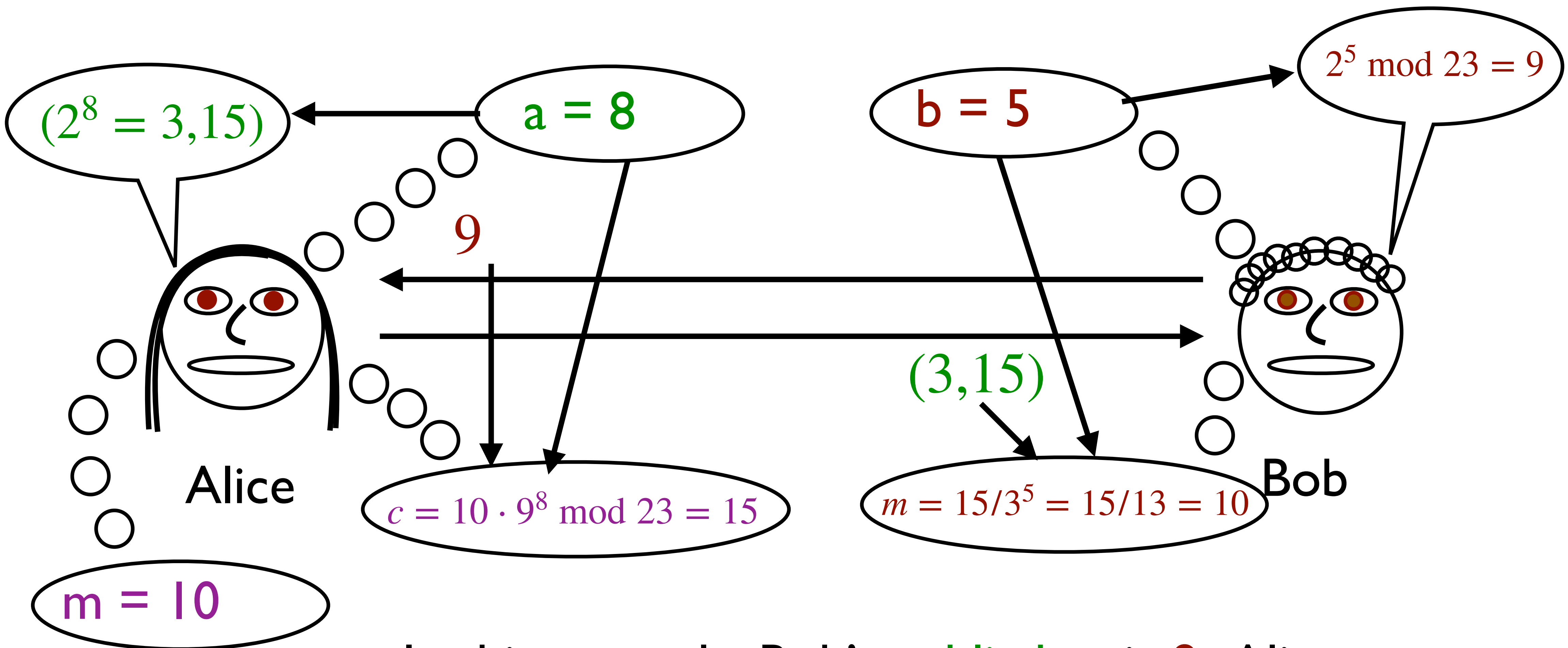


In this example, Bob's public key is 9. Alice uses it to encrypt.

Bob's private key is 5. He uses it to decrypt.

El Gamal Example

Example using p = 23, g = 2. (g has order 11 in $\mathbb{Z}_{23}^*$.)



In this example, Bob's public key is 9. Alice uses it to encrypt.

Bob's private key is 5. He uses it to decrypt.