# CMSC/Math 456: Cryptography (Fall 2022)

Lecture 13

Daniel Gottesman

# Administrative

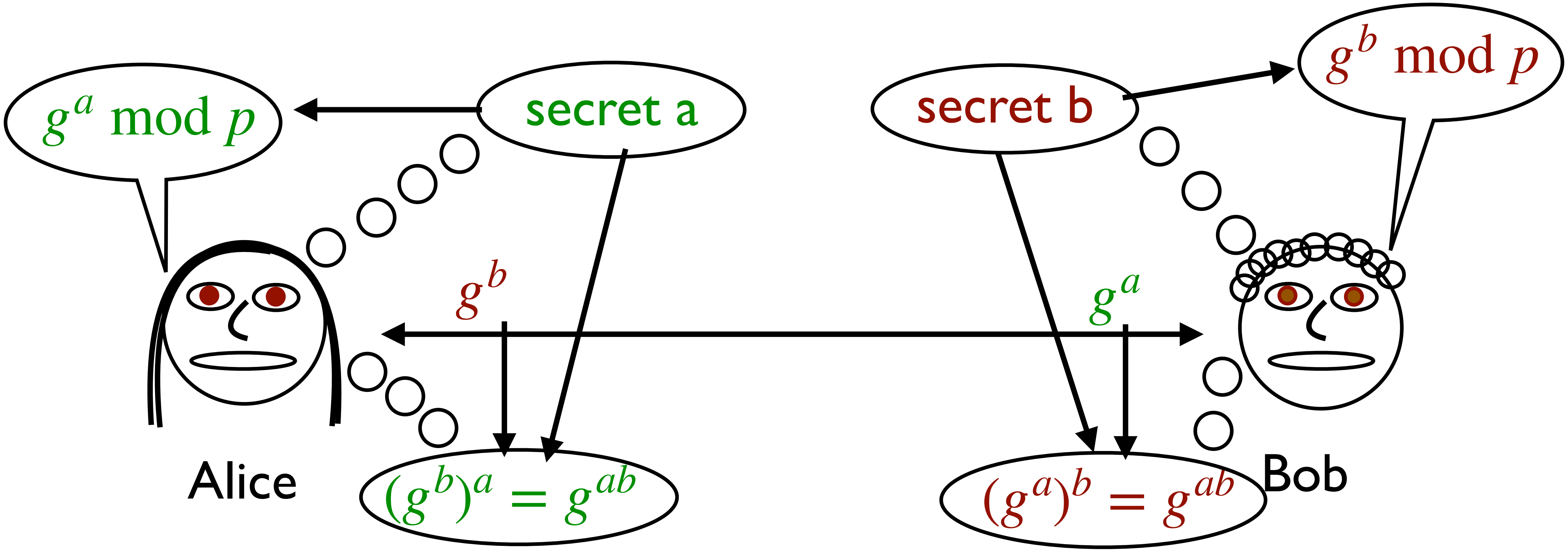
There have been multiple replacements to fix errors in problem set #5. Make sure you have the latest version. Also, please do not use a brute-force approach to problem 1.

Midterm: Thursday, Oct. 20 (1.5 weeks from today)

- In class
- Open book (including textbook), no electronic devices
- Will cover material through Diffie-Hellman and El Gamal, but not RSA.
- Those with accommodations remember to book with ADS.

Tuesday, Oct. 18 I will 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.

# 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 high order.
- Or use standard values for g and p.

Is this secure?

This class is being recorded

# Hardness of Discrete Log

In order to talk about computational hardness of discrete log, we need to consider a family of instances of increasing size. (Remember we are defining hardness asymptotically.)

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 worst-case hard if there is *no* polynomial time algorithm $\mathscr{A}$ such that for all $y \in \langle g_s \rangle$, $\mathscr{A}(y) = x$ with $y = g_s^x \bmod p_s$.

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

In order to talk about computational hardness of discrete log, we need to consider a family of instances of increasing size. (Remember we are defining hardness asymptotically.)

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 worst-case hard if there is *no* polynomial time algorithm $\mathscr{A}$ such that for all $y \in \langle g_s \rangle$, $\mathscr{A}(y) = x$ with $y = g_s^x \bmod p_s$.

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

In order to talk about computational hardness of discrete log, we need to consider a family of instances of increasing size. (Remember we are defining hardness asymptotically.)

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 worst-case hard if there is *no* polynomial time algorithm $\mathscr{A}$ such that for all $y \in \langle g_s \rangle$, $\mathscr{A}(y) = x$ with $y = g_s^x \bmod p_s$.

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.

This class is being recorded

# 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 average-case hard, it is worst-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 $y \in \langle g_s \rangle$,

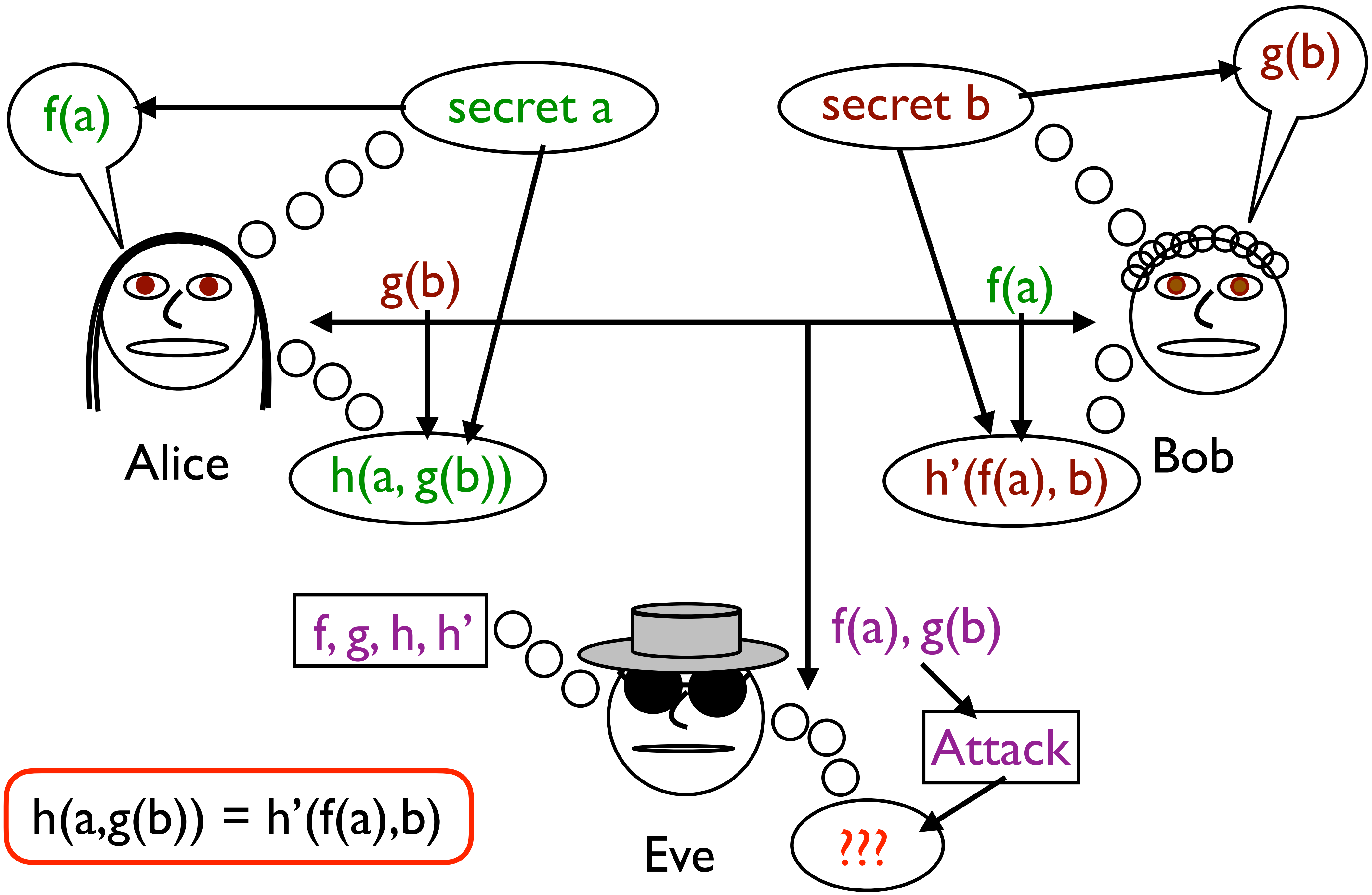
$$\Pr(\mathscr{A}(y) \text{ succeeds}) \leq \epsilon(s)$$

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 average-case hard, it is worst-case hard. (This is known as random self-reducibility.)

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

# Reminder: General Key Exchange




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

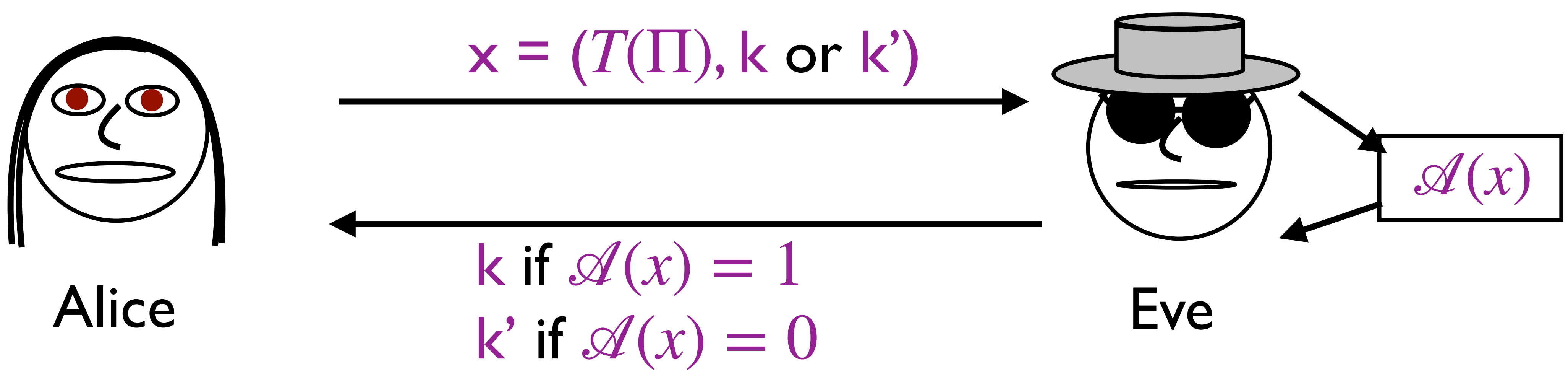
This class is being recorded

# Security Definition for Key Exchange

**Definition:** Consider a key exchange protocol $\Pi$. 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 $\Pi$ 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 $\Pi$.




This class is being recorded

# 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 $\Pi$, which uses a key k, generated with the key exchange protocol. If Eve has an attack against $\Pi$, then Eve has an attack against the key exchange protocol.

# 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?

- Pick a random element $h \in \mathbb{Z}_p^*$
- Let $g = h^2$ (or more generally, $g = h^r$)
- If $g = 1$ try again

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

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.

# 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#5, 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 Diffie-Hellman is hard, discrete log is also 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?

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 and B, there might be a way to find k, 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

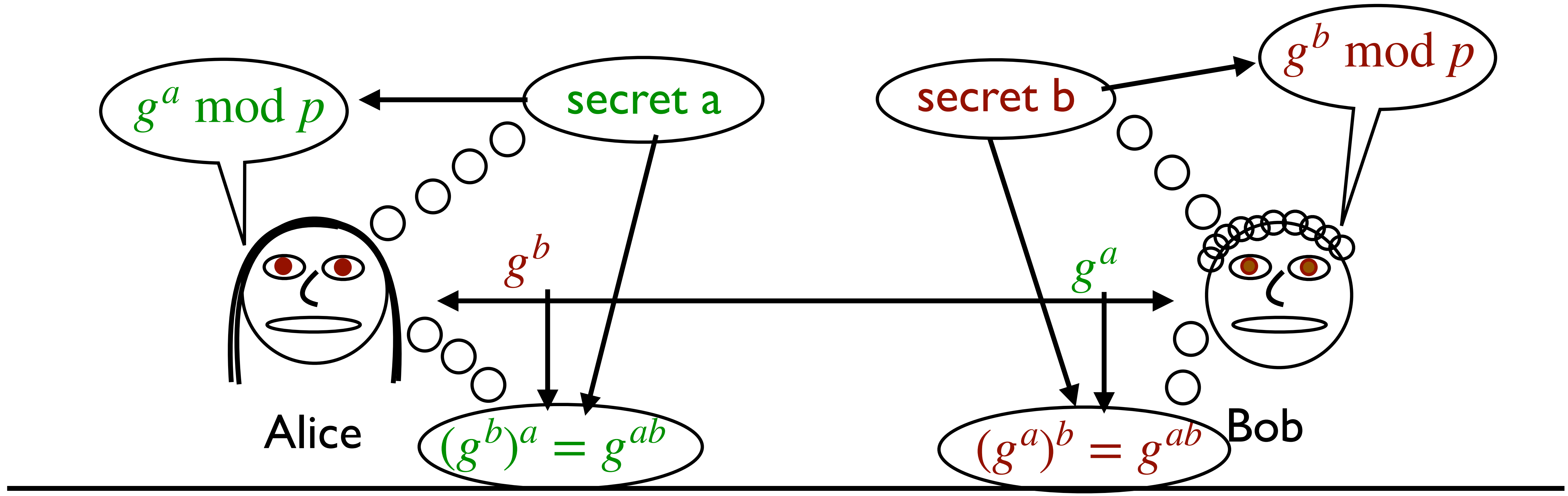
In practice, the best known algorithms for breaking Diffie-Hellman work by breaking discrete log.

- 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:

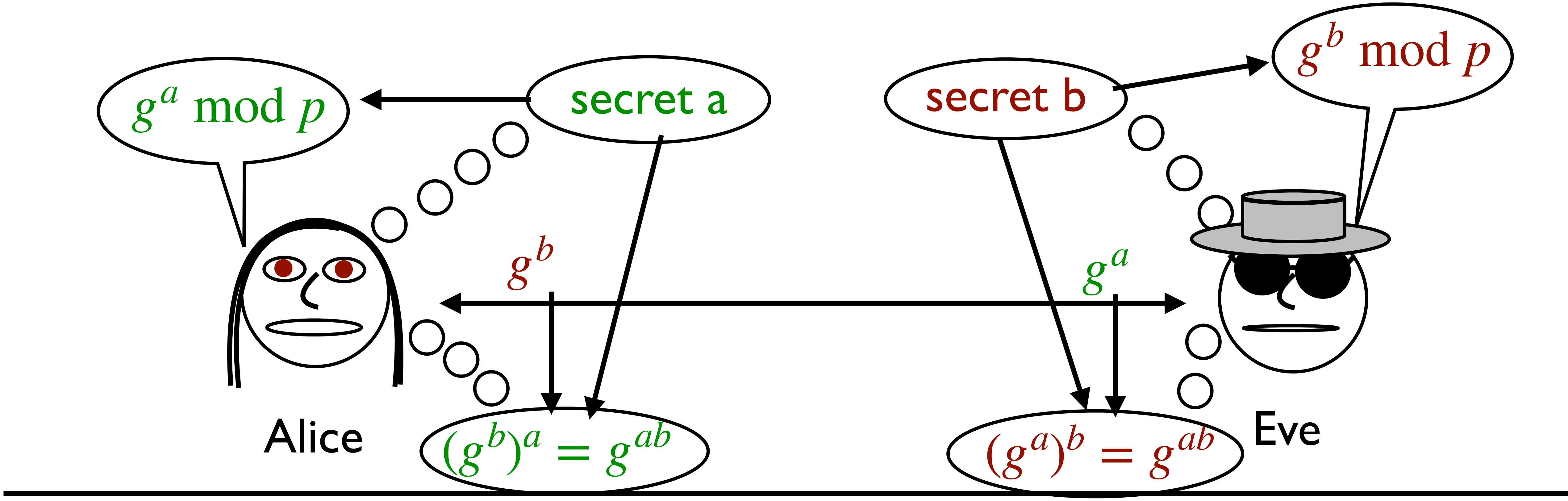
- Over $\mathbb{Z}_p^*$: use p of length 2048 bits or longer.
- Elliptic curves key length: 224 bits or higher.
- But don't use either if concerned about quantum attacks.

# Another Attack on Diffie-Hellman



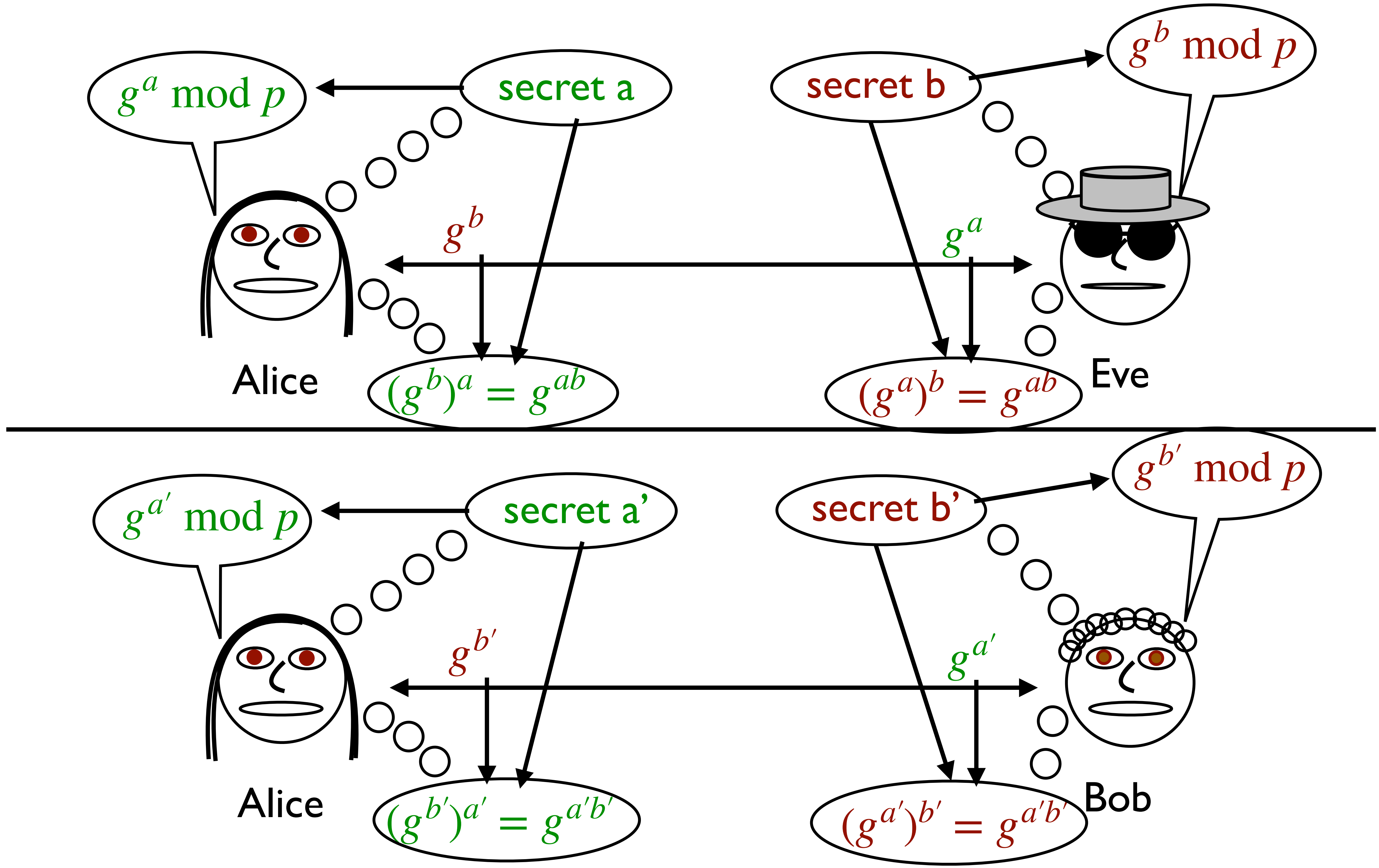

This class is being recorded

# Another Attack on Diffie-Hellman



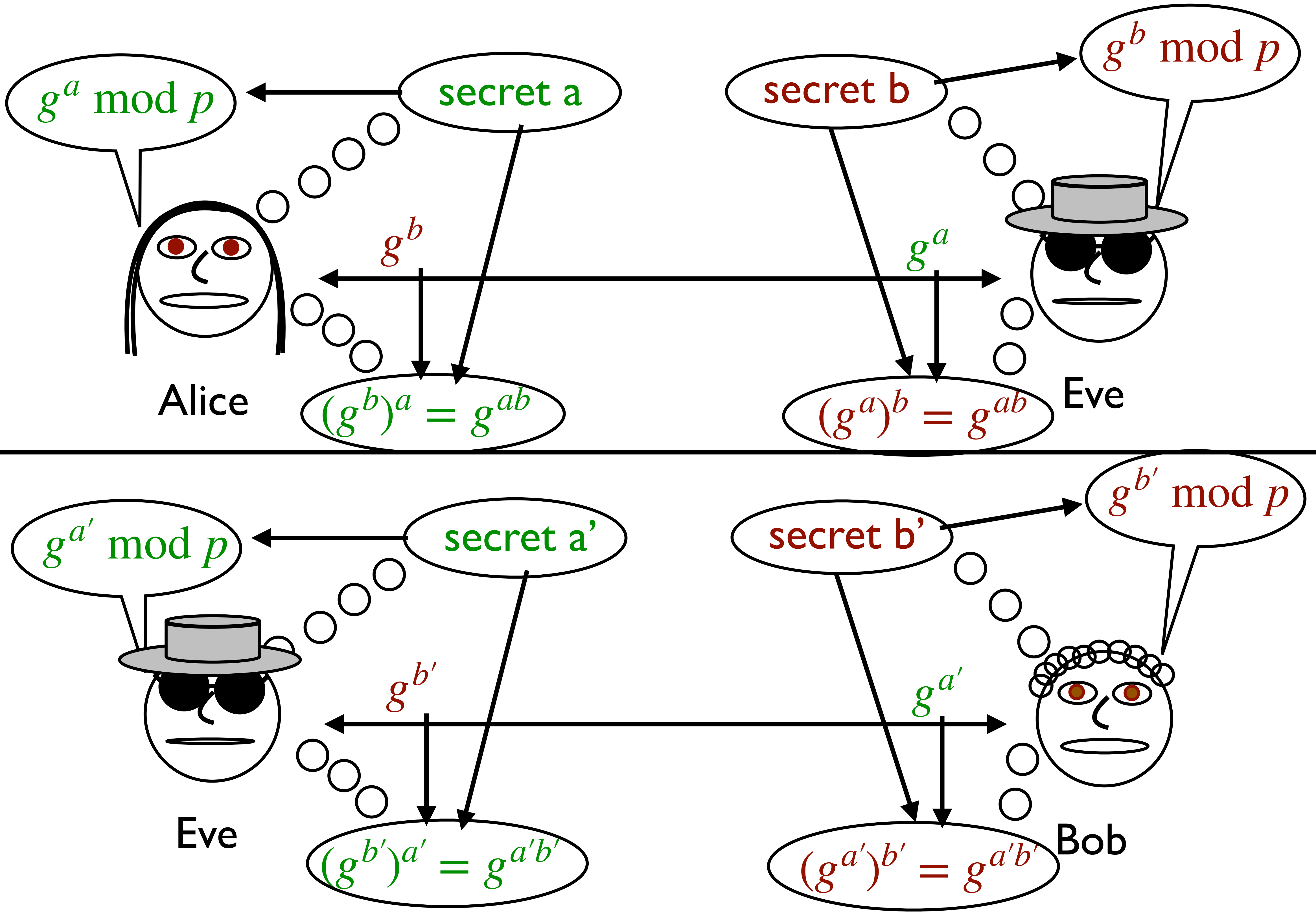

This class is being recorded

# Another Attack on Diffie-Hellman



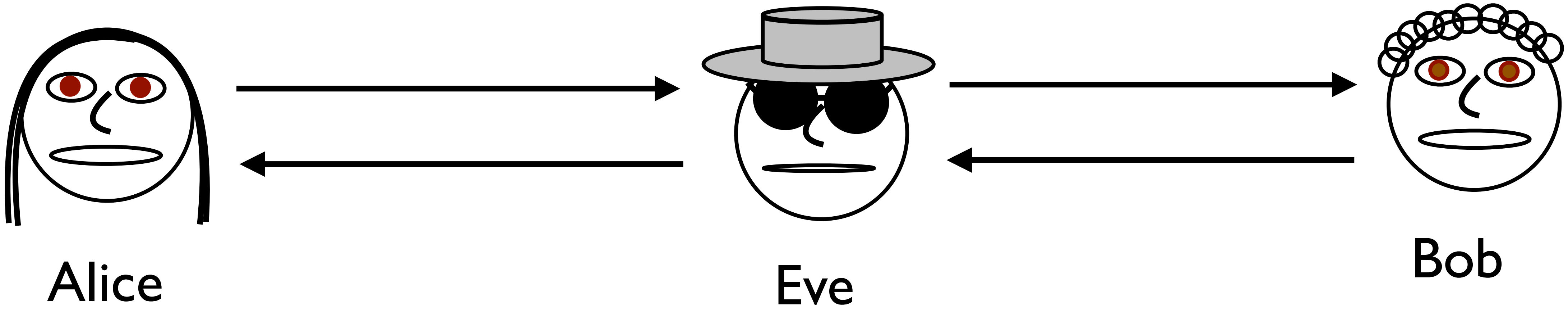

This class is being recorded

# 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.