CMSC/Math 456: Cryptography (Fall 2023)

Lecture 25

Daniel Gottesman

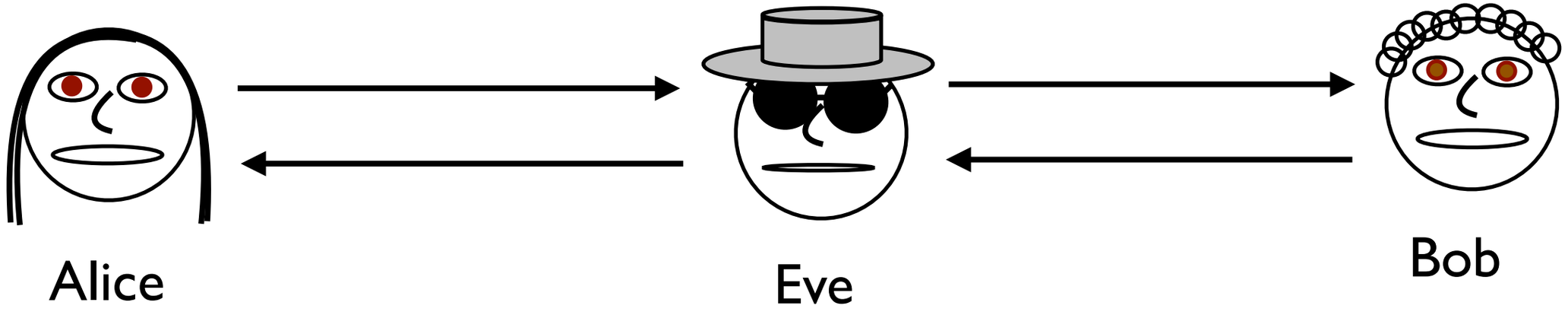
Administrative

Problem set #10 is out and is due next Thursday.

Course evaluations are now available to fill out.

This class is being recorded

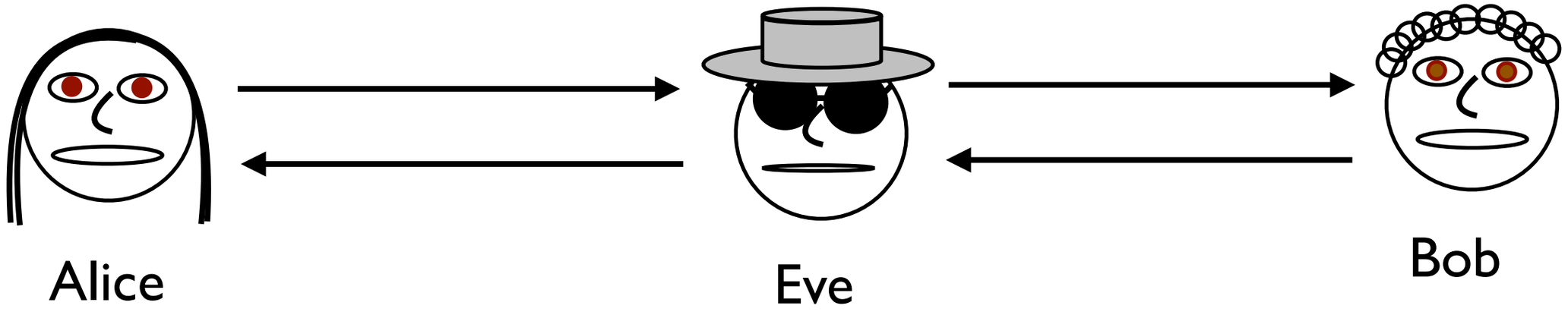
Man-in-the-Middle Attack



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 without any authentication, Alice and Bob have no way to fight this attack and Eve can read all their messages.

This class is being recorded

Man-in-the-Middle Attack

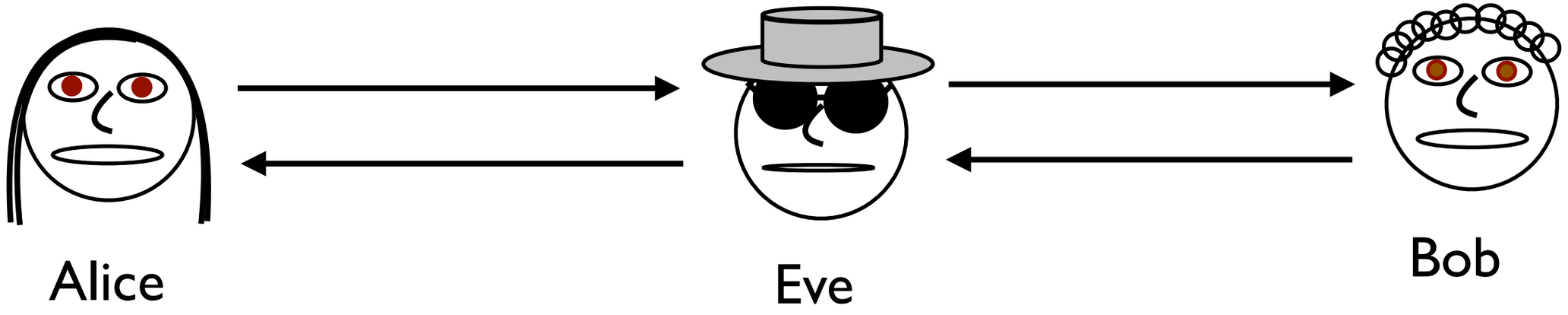


In a man-in-the-middle attack, Eve intercepts all communications between Alice and Bob and replaces them with messages of her choice. In Diffie-Hellman without any authentication, Alice and Bob have no way to fight this attack and Eve can read all their messages.

If Alice and Bob digitally sign their messages, then Eve can't replace them.

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 without any authentication, Alice and Bob have no way to fight this attack and Eve can read all their messages.

If Alice and Bob digitally sign their messages, then Eve can't replace them.

But this requires that each has the other's public key, and how did they get those in the first place?

This class is being recorded

In-Person Public Key Distribution

The first way to get someone's public key is to meet them in person. Either they need to be someone you know or you should verify their identity (e.g., by checking their ID).

Once this is done, you can get their public key. You should also give them yours.

This class is being recorded

In-Person Public Key Distribution

The first way to get someone's public key is to meet them in person. Either they need to be someone you know or you should verify their identity (e.g., by checking their ID).

Once this is done, you can get their public key. You should also give them yours.

The important thing is to exchange *signature* keys. You can then use those signature keys to sign encryption public keys to each other at some later time.

This class is being recorded

In-Person Public Key Distribution

The first way to get someone's public key is to meet them in person. Either they need to be someone you know or you should verify their identity (e.g., by checking their ID).

Once this is done, you can get their public key. You should also give them yours.

The important thing is to exchange *signature* keys. You can then use those signature keys to sign encryption public keys to each other at some later time.

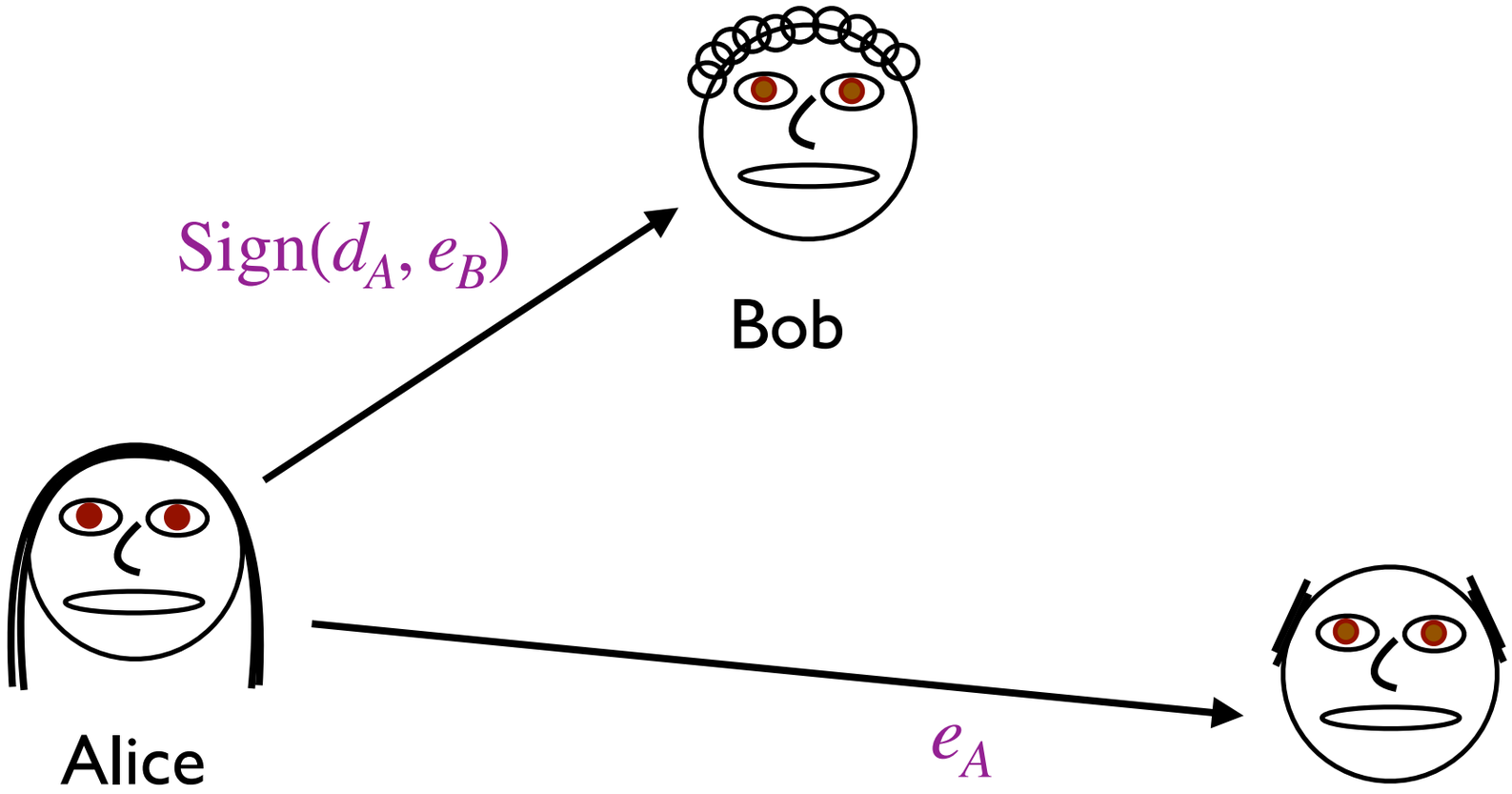
You should also consider signing the other person's public signature key and giving them a copy of the signature.

That is, if (d_A, e_A) is Alice's key and (d_B, e_B) is Bob's key, produce $\mathrm{Sign}(d_A, e_B)$ and give it to Bob. Also produce $\mathrm{Sign}(d_B, e_A)$ and give it to Alice.

This class is being recorded

Why Sign Someone Else's Key?

Transferability means that they can then pass on that key to someone else.

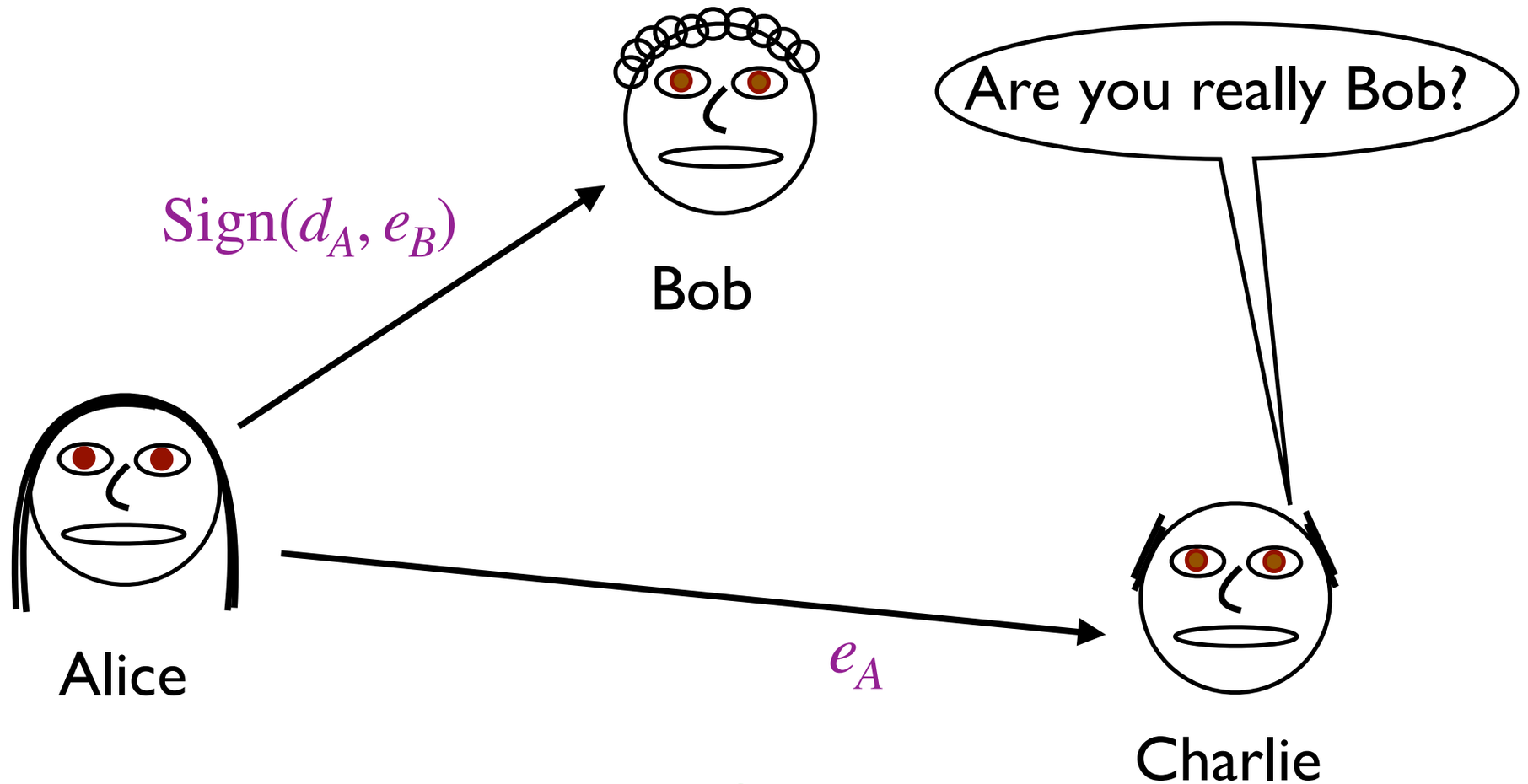


Signing a key means that you certify the key belongs to the right person.

This class is being recorded

Why Sign Someone Else's Key?

Transferability means that they can then pass on that key to someone else.

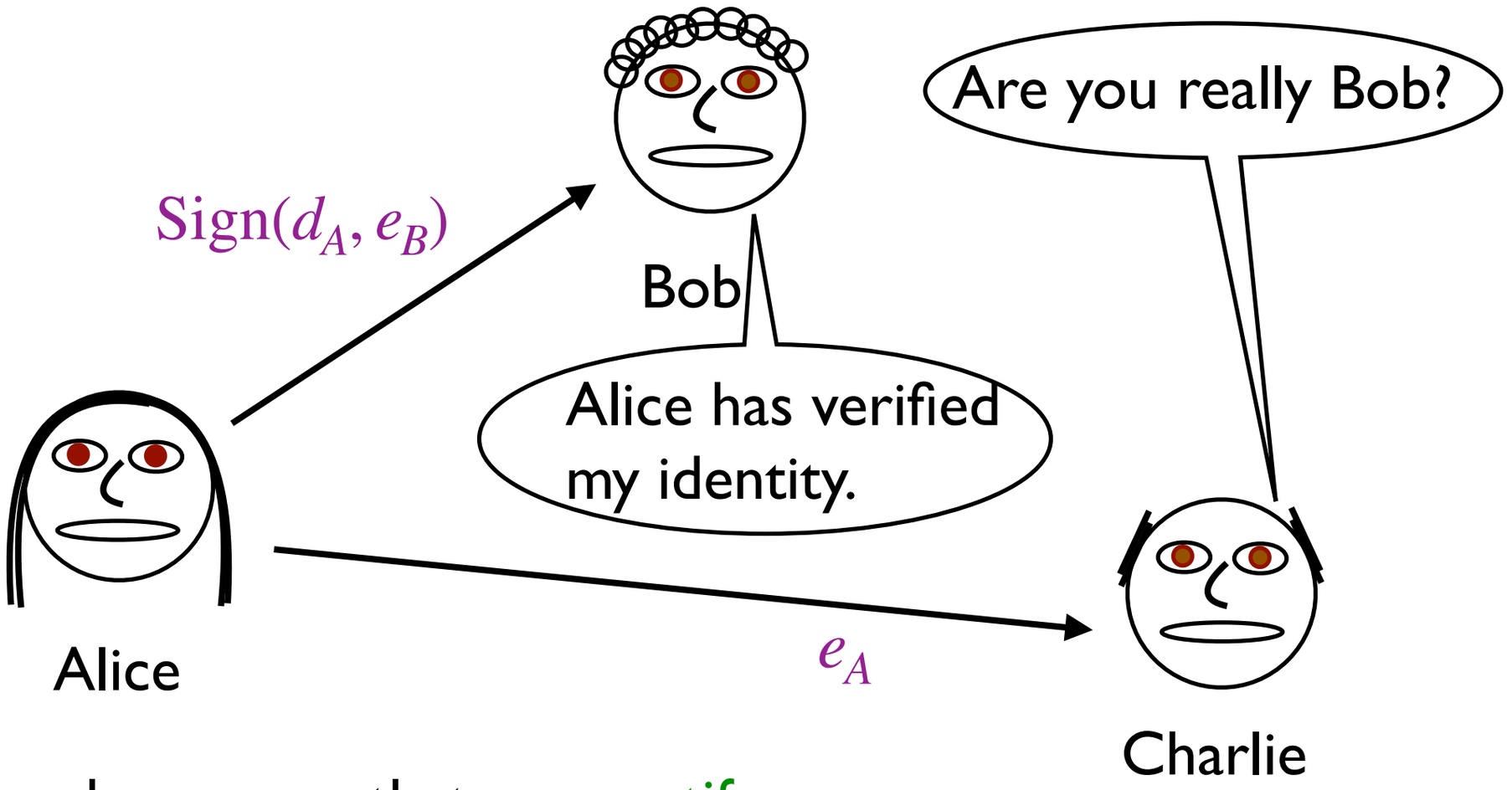


Signing a key means that you certify the key belongs to the right person.

This class is being recorded

Why Sign Someone Else's Key?

Transferability means that they can then pass on that key to someone else.

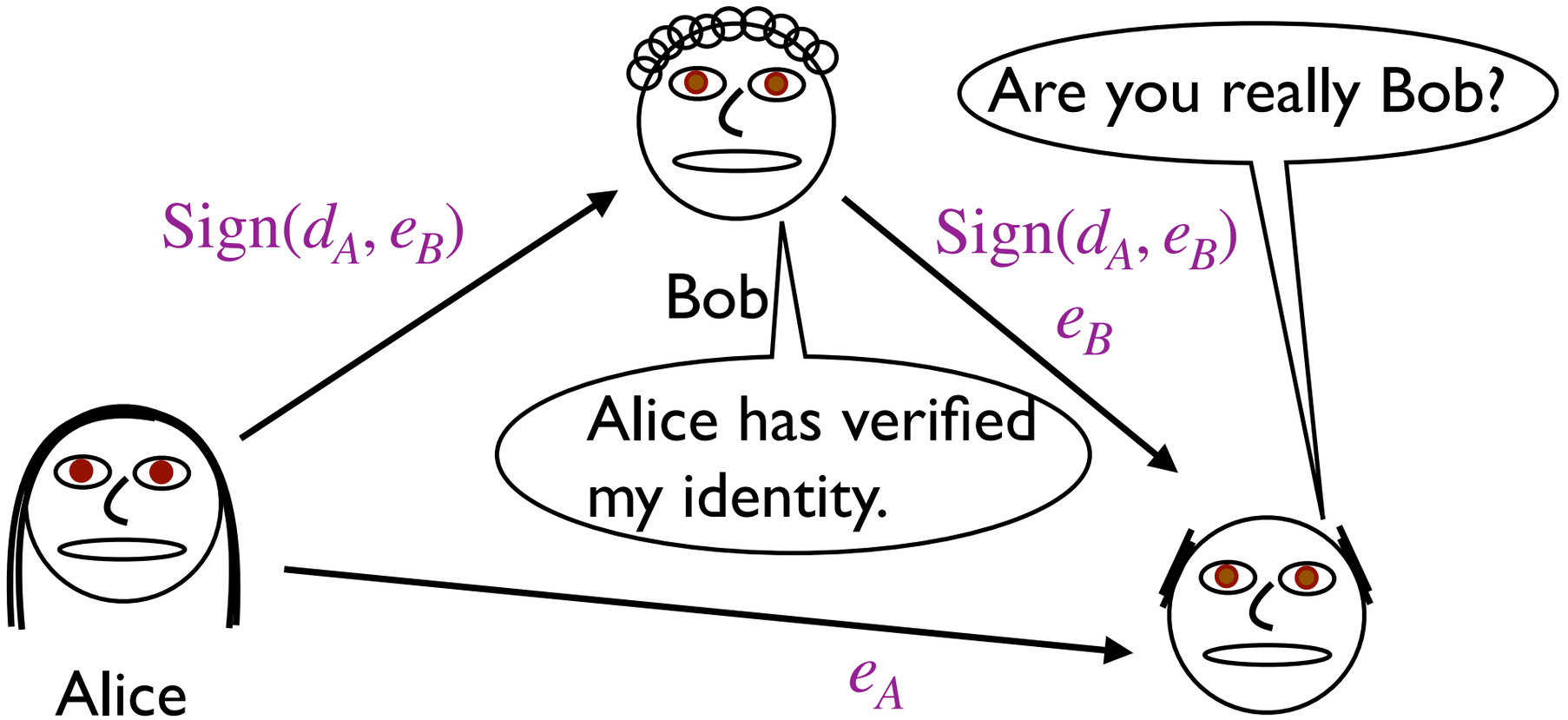


Signing a key means that you certify the key belongs to the right person.

This class is being recorded

Why Sign Someone Else's Key?

Transferability means that they can then pass on that key to someone else.

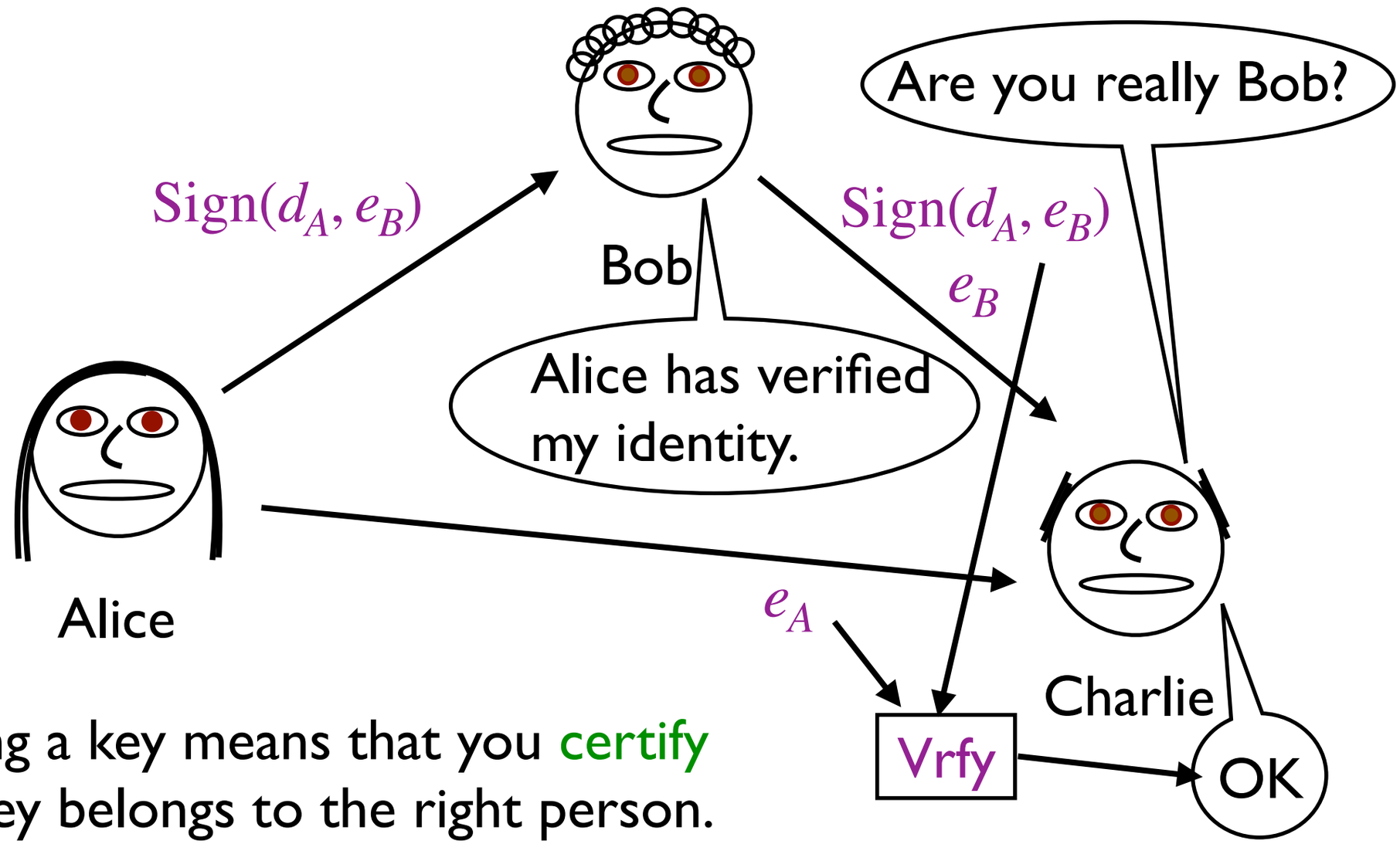


Signing a key means that you certify the key belongs to the right person.

This class is being recorded

Why Sign Someone Else's Key?

Transferability means that they can then pass on that key to someone else.

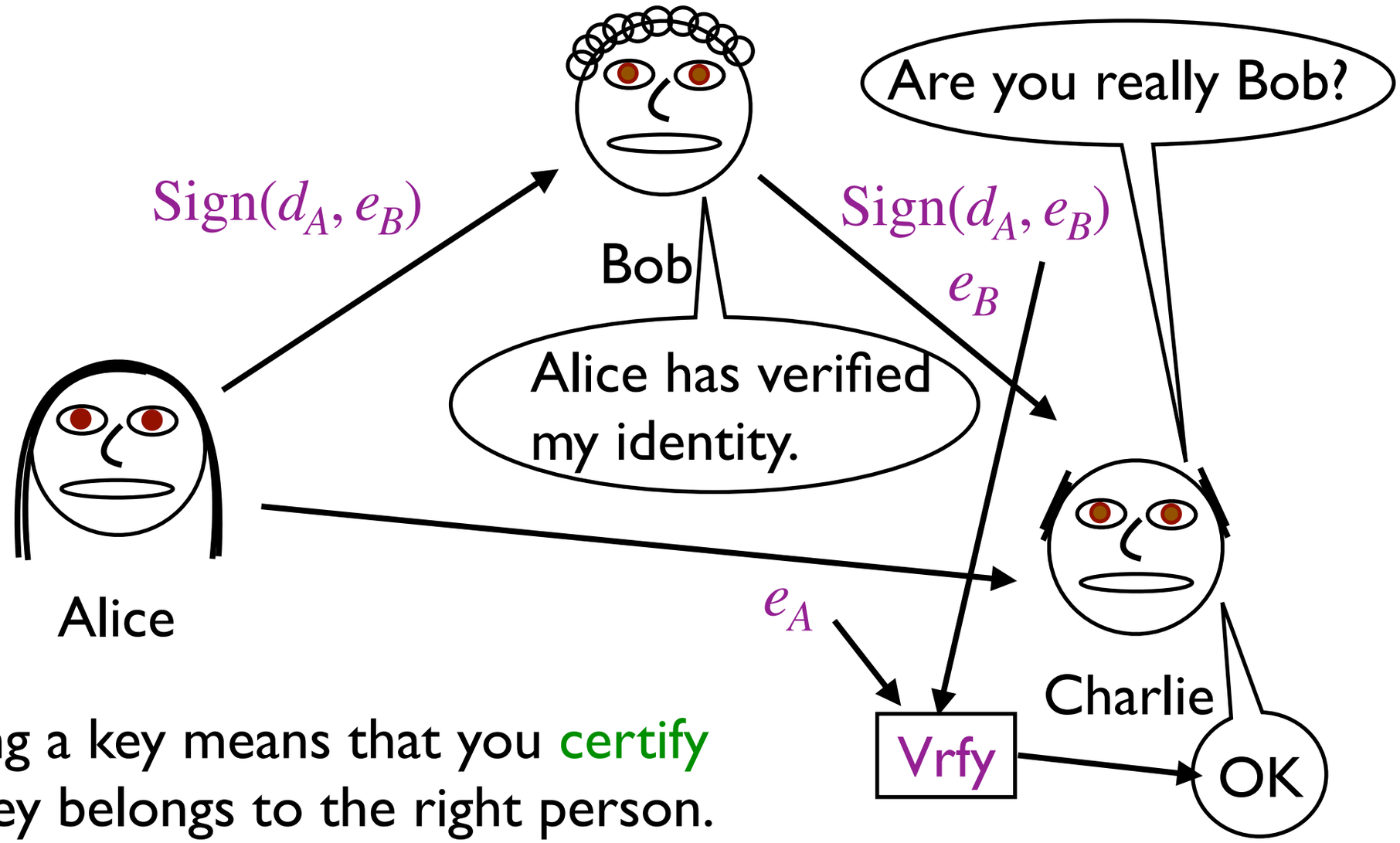


Signing a key means that you certify the key belongs to the right person.

This class is being recorded

Why Sign Someone Else's Key?

Transferability means that they can then pass on that key to someone else.



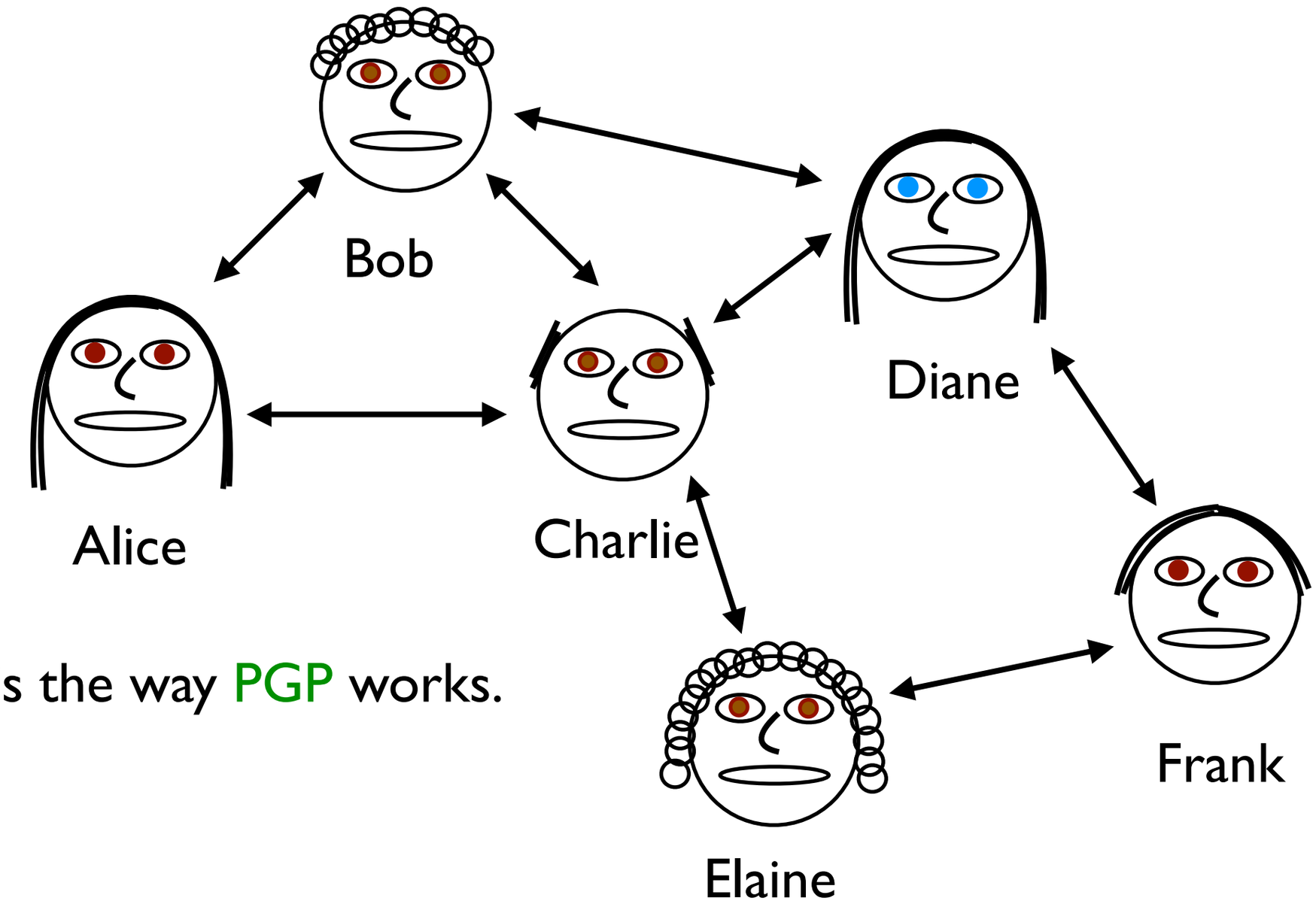
Signing a key means that you certify the key belongs to the right person.

In this example, Charlie could then also sign Bob's public key.

This class is being recorded

Web of Trust

You can create a web of trust: Frank doesn't know Alice, but he trusts Diane, and she trusts Charlie, who does know Alice.

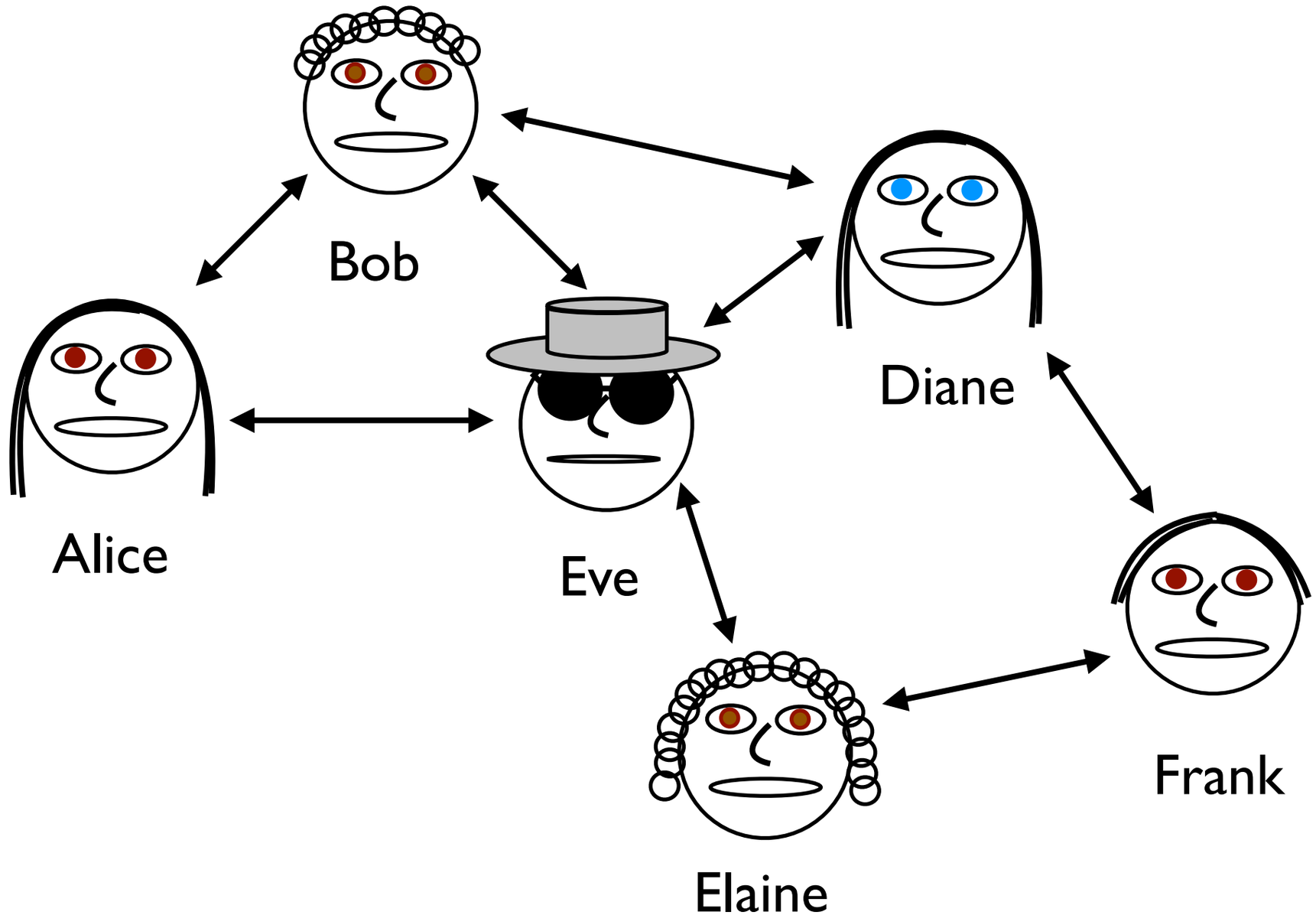


This is the way PGP works.

This class is being recorded

Limitation of Web of Trust

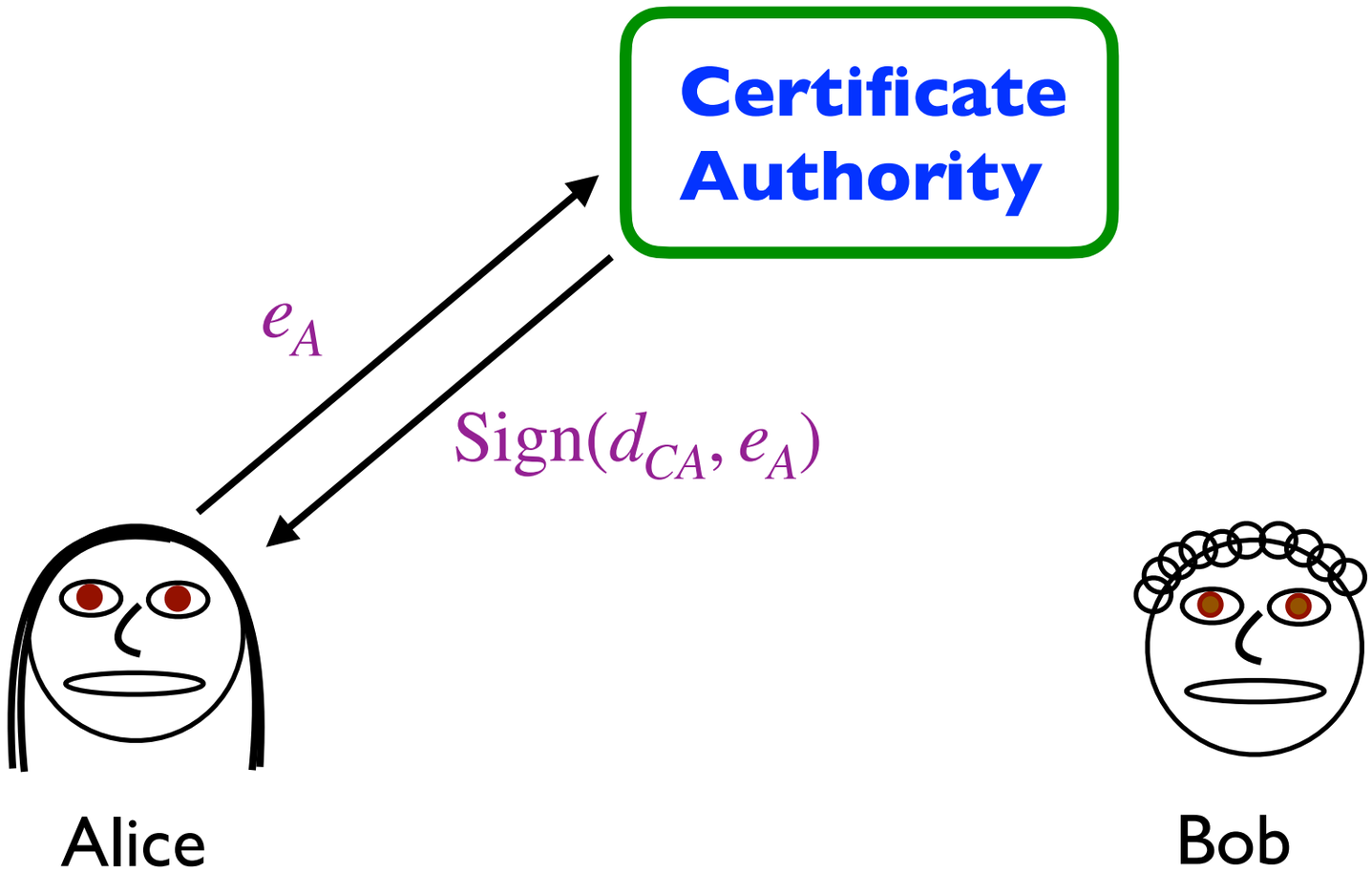
The longer the chain, the more likely someone in it has made an error in judgement and trusted the wrong person.



This class is being recorded

Certificate Authorities

Another solution is to have organizations, certificate authorities (CAs), that verify everyone's identities and sign their keys.

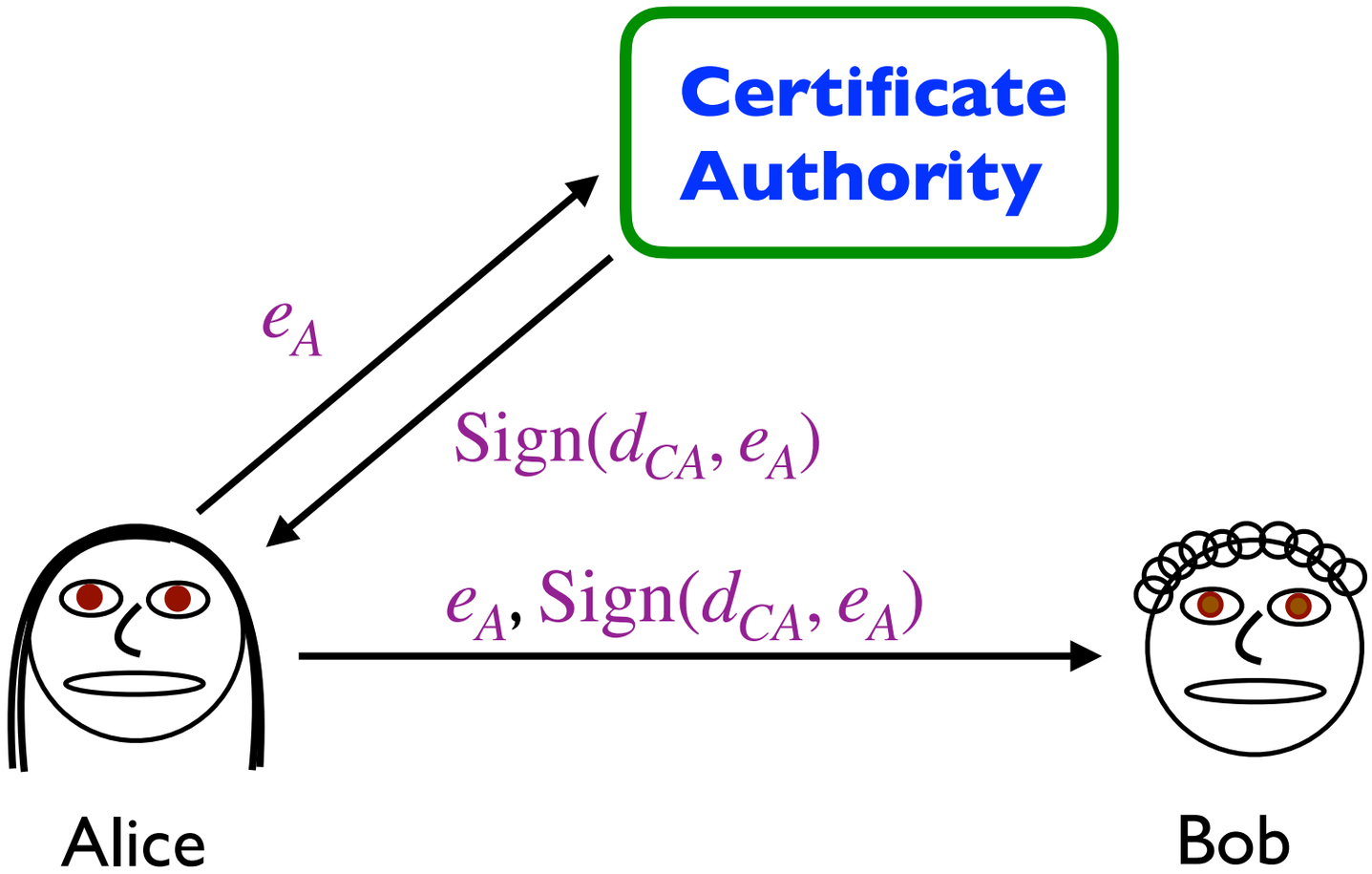


The CA issues a certificate signing a public key indicating that the owner's identity has been checked.

This class is being recorded

Certificate Authorities

Another solution is to have organizations, certificate authorities (CAs), that verify everyone's identities and sign their keys.

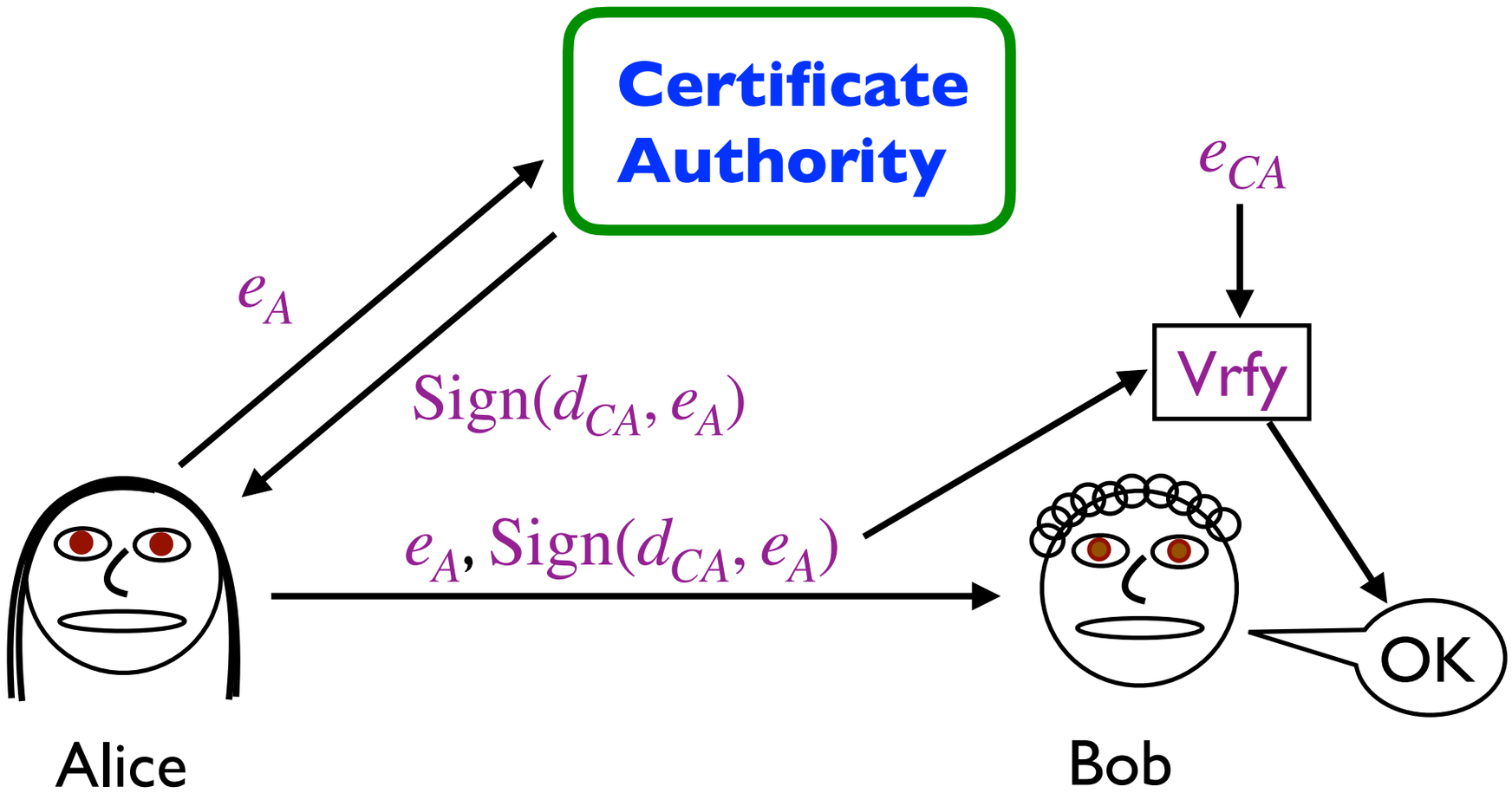


The CA issues a certificate signing a public key indicating that the owner's identity has been checked.

This class is being recorded

Certificate Authorities

Another solution is to have organizations, certificate authorities (CAs), that verify everyone's identities and sign their keys.



The CA issues a certificate signing a public key indicating that the owner's identity has been checked.

This class is being recorded

CAs and the Web

Authentication on the web using https uses certificate authorities.

> Browsers have built-in public keys for a number of major certificate authorities. These allow the browser to verify certificates issued by any of these CAs.

Only the websites get the certificates usually. When you visit a website, you check its certificate to verify that it is the correct website.

> If the certificate doesn't check out, your browser will give you a warning message.

You may also prove your identity to the website, e.g., through entering a password. That part is not handled by the CA, but only by any prior information the website might have on you.

This class is being recorded

Expiring and Revoking Certificates

CAs are not perfect, of course. They can be

This class is being recorded

Expiring and Revoking Certificates

CAs are not perfect, of course. They can be

- Fooled, and issue a certificate to an imposter.

This class is being recorded

Expiring and Revoking Certificates

CAs are not perfect, of course. They can be

- Fooled, and issue a certificate to an imposter.
- Hacked. A hacker that steals a CA's private key can create "valid" certificates for anyone.

This class is being recorded

Expiring and Revoking Certificates

CAs are not perfect, of course. They can be

- Fooled, and issue a certificate to an imposter.
- Hacked. A hacker that steals a CA's private key can create "valid" certificates for anyone.
- Co-opted, for instance by the government of the CA's home country.

This class is being recorded

Expiring and Revoking Certificates

CAs are not perfect, of course. They can be

- Fooled, and issue a certificate to an imposter.
- Hacked. A hacker that steals a CA's private key can create "valid" certificates for anyone.
- Co-opted, for instance by the government of the CA's home country.

This is a real problem. The first two have definitely happened, and quite possibly the third as well.

This class is being recorded

Expiring and Revoking Certificates

CAs are not perfect, of course. They can be

- Fooled, and issue a certificate to an imposter.
- Hacked. A hacker that steals a CA's private key can create "valid" certificates for anyone.
- Co-opted, for instance by the government of the CA's home country.

This is a real problem. The first two have definitely happened, and quite possibly the third as well.

As a partial solution, certificates are designed to

- Expire after a certain amount of time.
- Be revoked if discovered to be fraudulent.

TLS

Certificate authorities (if not corrupted) make connecting on the web secure against a man-in-the-middle attack, but making a truly secure connection requires many different steps.

> The current protocol for https is called TLS (transport layer security).

TLS has two phases:

- A handshake protocol, which establishes the shared keys needed to communicate securely.
- A record-layer protocol, in which the actual communication takes place.

This class is being recorded

TLS Handshake Overview

The current version (1.3) of the TLS handshake consists of three messages:

- ClientHello: The client (i.e., browser) indicates desire to make a connection to the website and begins to set up the parameters and keys needed to do so securely.
- ServerHello: The server (i.e. website) replies with its certificate, and the remaining key and protocol parameter information.
- After verifying the certificate and deriving the necessary key information, the client confirms that the handshake was successful.

It needs to do the following:

- Verify the server's certificate
- Establish a shared session key
- Determine the cryptographic protocols to be used

This class is being recorded

Encryption Strategy

TLS begins with a Diffie-Hellman key exchange. The group can be either a prime subgroup of $\mathbb{Z}_p^*$ or an elliptic curve group.

The key generated via Diffie-Hellman is then used as the key for a private key protocol for authenticated encryption.

Precisely which private key protocol (cipher suite) is used is decided during the handshake.

TLS 1.3 uses authenticated encryption on the later parts of the handshake protocol and then uses a separate key for the record-layer protocol.

The key generated is generally used for only one session. (It is a session key.) A new session must go through the handshake again and generate a new session key.

(TLS also does support pre-shared keys to resume a previous session.)

This class is being recorded

TLS Cipher Suites

TLS 1.3 supports the following cipher suites for the authenticated encryption (with associated data, AEAD):

- GCM: AES w/128 or 256 bit keys, GMAC for the MAC, and SHA256 or SHA384 for the hash function.
- CCM: AES w/ 128 bit keys, CBC-MAC for the MAC (with 16 or 8-byte tags), and SHA256 for the hash function.
- ChaCha20-Poly1305: These are a stream cipher and a MAC we didn't discuss; this uses SHA256 for the hash function.

For key exchange, TLS 1.3 supports only Diffie-Hellman with a limited set of modular groups or elliptic curves.

For digital signatures, TLS 1.3 accepts:

- RSA (secure variants).
- ECDSA: discrete-log signatures with elliptic curves.
- EdDSA: another discrete-log-based signature protocol.

This class is being recorded

ClientHello

The first message is from the client. It includes the following:

- Information about which cipher suites (and TLS version) the client supports.
- (G, q, g): The group, group order, and generator that the client wants to use for Diffie-Hellman, chosen from a list of standard values.
- g^x: This is the first step in Diffie-Hellman, half of what is needed to establish the key.
- A random bit string, a "nonce". A nonce is a value used only one time.

At this point, the client has only one piece of secret information:

- x: The exponent used by the client in DH.

This class is being recorded

ServerHello

The server responds with the following:

- Confirmation of the cipher suite.
- g^y: The second message in Diffie-Hellman, establishing the shared key.
- A new nonce
- An authenticated & encrypted message containing:
  - e_{server}: The server's public signature key
  - $\text{Sign}(d_{CA}, e_{\text{server}})$: The certificate for the server's public key
  - $\text{Sign}(d_{\text{server}}, \text{handshake})$: A signature (created using the server's private signature key) for the previous messages in the handshake

The server now has the following secret information:

- $H(g^{xy})$: The key created via Diffie-Hellman, split into four keys for authenticated encryption. The first is used here.

This class is being recorded

Client Verification

At this point, the client has to perform the following checks and computations:

- $H(g^{xy})$: The Diffie-Hellman key, again split into four private keys.
- Decrypt and verify e_{server} and $\text{Sign}(d_{CA}, e_{\text{server}})$ using the first private key.
- Decrypt and verify $\text{Sign}(d_{\text{server}}, \text{handshake})$ using the first private key.

The client then uses the second private key and a MAC to send:

- Authentication of all handshake messages.

The client and server now share two remaining private keys, to be used for the record-layer protocol.

This class is being recorded

Handshake Analysis

The point of signing and authenticating the handshake transcript in both directions is to make sure the handshake is with the correct entity. That is, it proves the server whose certificate is presented matches the one in the handshake; and it proves that the client sending the final message of the handshake is the same as the one that sent the first one.

The nonces are needed to prevent Eve from simply repeating messages from a previous session to masquerade as one or the other of the participants.

The security of the Diffie-Hellman protocol means they generate secure private keys to be used in the record-layer protocol.

This class is being recorded

Record-Layer Protocol

In the record-layer protocol, the client and server send back and forth messages protected via the chosen authenticated encryption scheme. The client uses one of the two keys derived during the handshake and the server uses the other.

Using separate keys prevents reflection attacks, where Eve repeats back to one of the two sides their own messages.

These keys are used as master keys from which subkeys are derived for additional security.

Each message should also include a nonce and counter to prevent replay or reordering attacks, where Eve repeats previous messages or changes the order in which they received.

This class is being recorded

Forward Secrecy

TLS 1.3 only allows Diffie-Hellman for key exchange.

TLS 1.2 and earlier also allowed RSA. This option was removed to ensure forward secrecy:

> If the session key is encrypted using an RSA public key, but the corresponding private key is later compromised, Eve learns the session key and can read any encrypted messages she happens to have recorded.

With Diffie-Hellman, the derived private key $H(g^{xy})$ is only used for one session, so it is erased after that session is done. That key can no longer be leaked, so the messages of the session remain secure.

> Note that this does not protect against a computational breakthrough which allows Eve to defeat Diffie-Hellman, only against a leak of the key.

This class is being recorded

Signcryption

Signcryption is the public key analog of authenticated encryption. Now, to encrypt we need the receiver's public key and to sign we need the sender's private key.

> Therefore, all senders and receivers need their own public key-private key pairs. (And they should have two such pairs if sometimes they are sending and sometimes receiving.)

We could use $(c = \mathrm{Enc}(e_{\mathrm{receiver}}, m), \mathrm{Sign}(d_{\mathrm{sender}}, c))$.

> Vote: Does encrypt-then-authenticate work in this context (given secure public key encryption and digital signatures)? (Yes/No/Unknown)

This class is being recorded

Signcryption

Signcryption is the public key analog of authenticated encryption. Now, to encrypt we need the receiver's public key and to sign we need the sender's private key.

> Therefore, all senders and receivers need their own public key-private key pairs. (And they should have two such pairs if sometimes they are sending and sometimes receiving.)

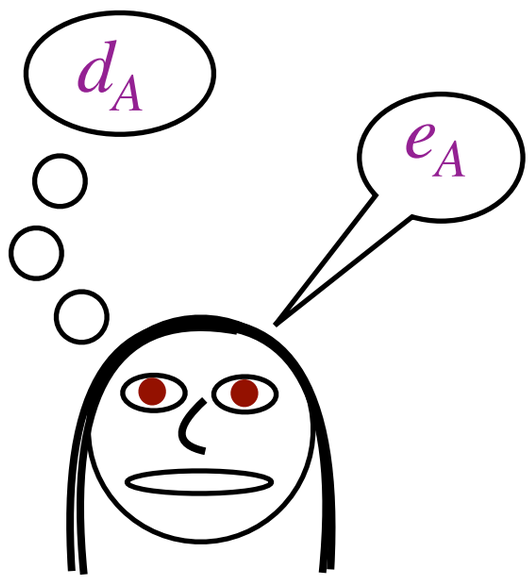
We could use $(c = \mathrm{Enc}(e_{\mathrm{receiver}}, m), \mathrm{Sign}(d_{\mathrm{sender}}, c))$.

Vote: Does encrypt-then-authenticate work in this context (given secure public key encryption and digital signatures)? (Yes/No/Unknown)

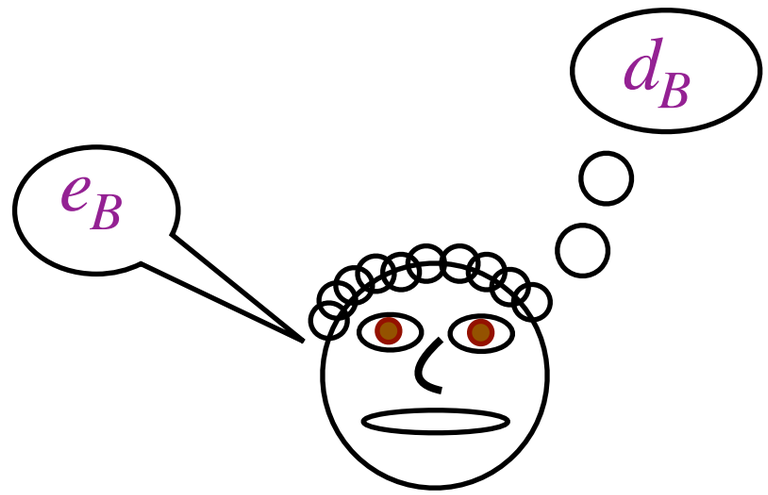
Answer: No, there is a reattribution attack.

This class is being recorded

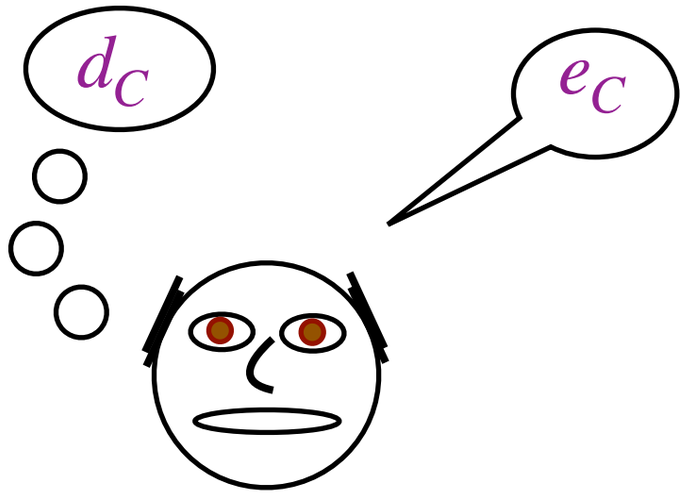
Reattribution



Alice



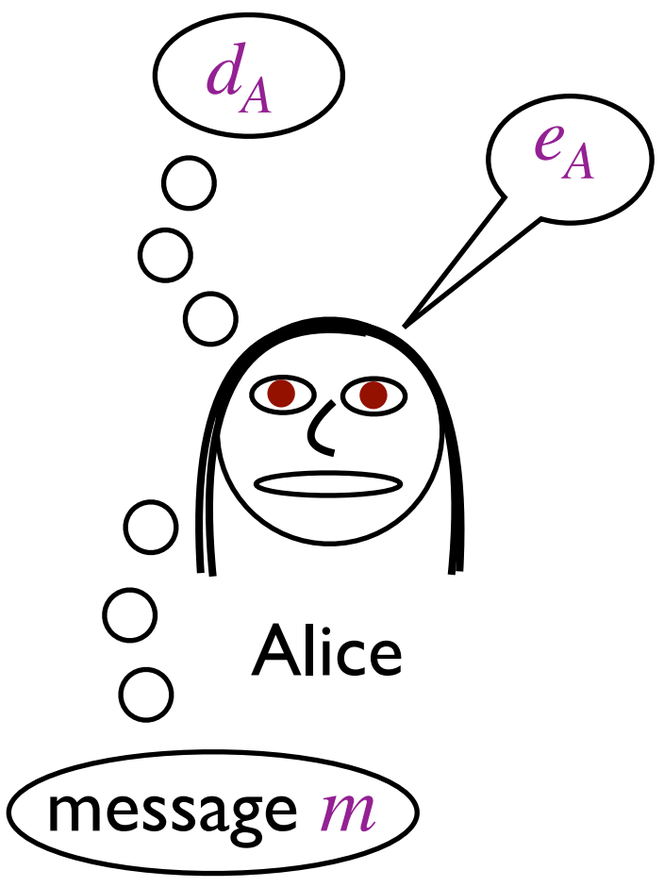
Bob

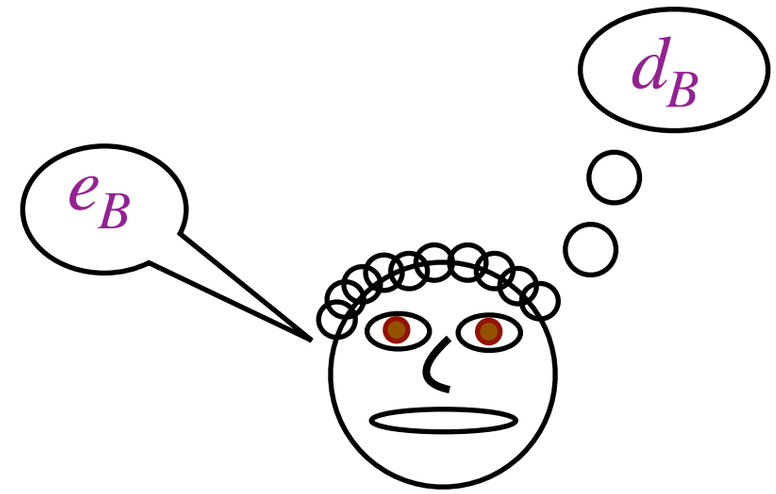


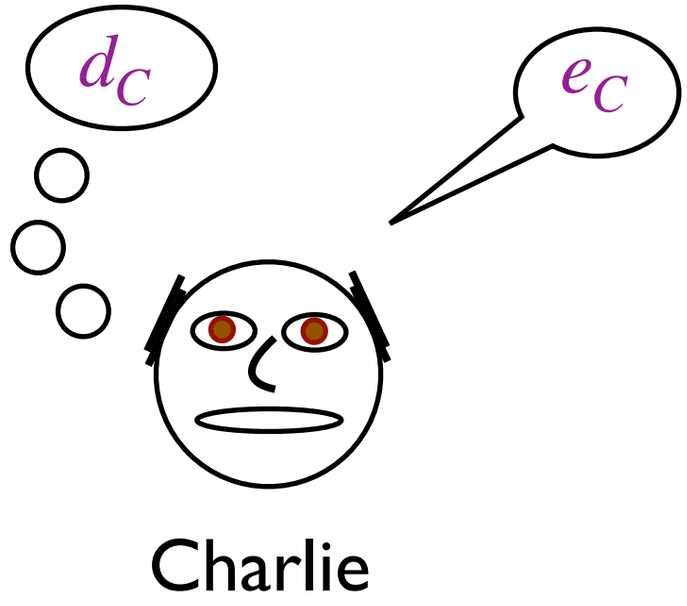
Charlie

This class is being recorded

Reattribution

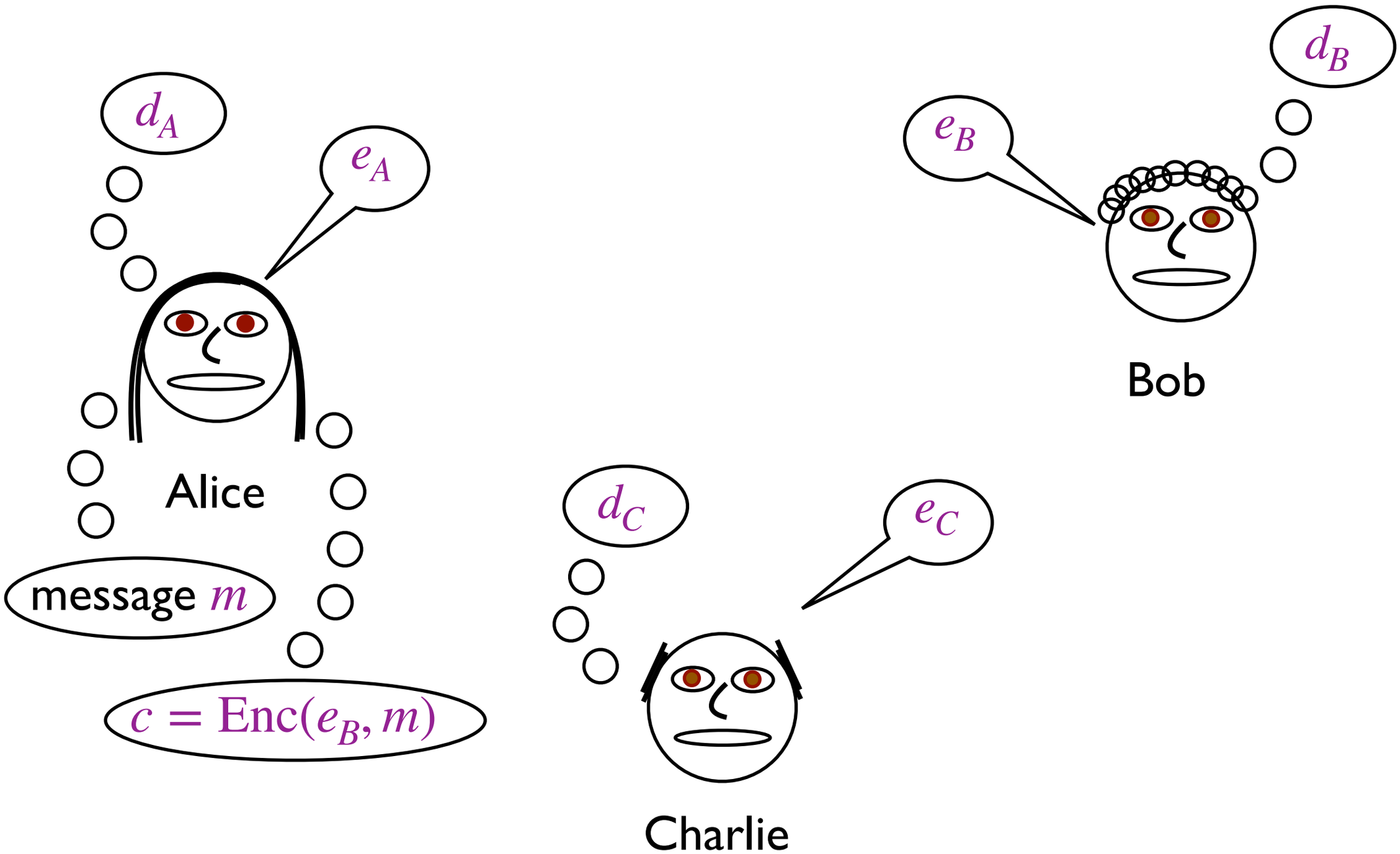






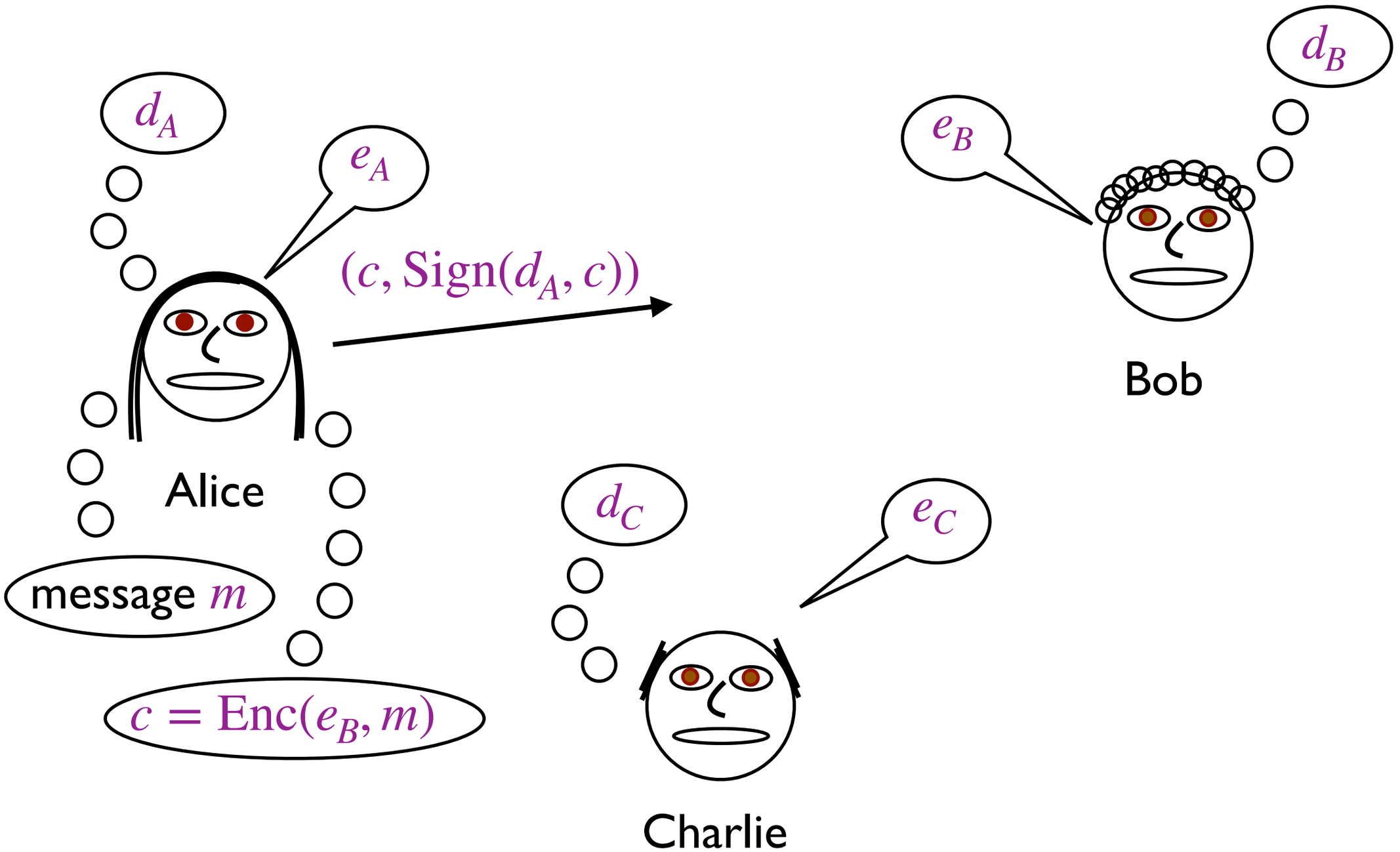
This class is being recorded

Reattribution



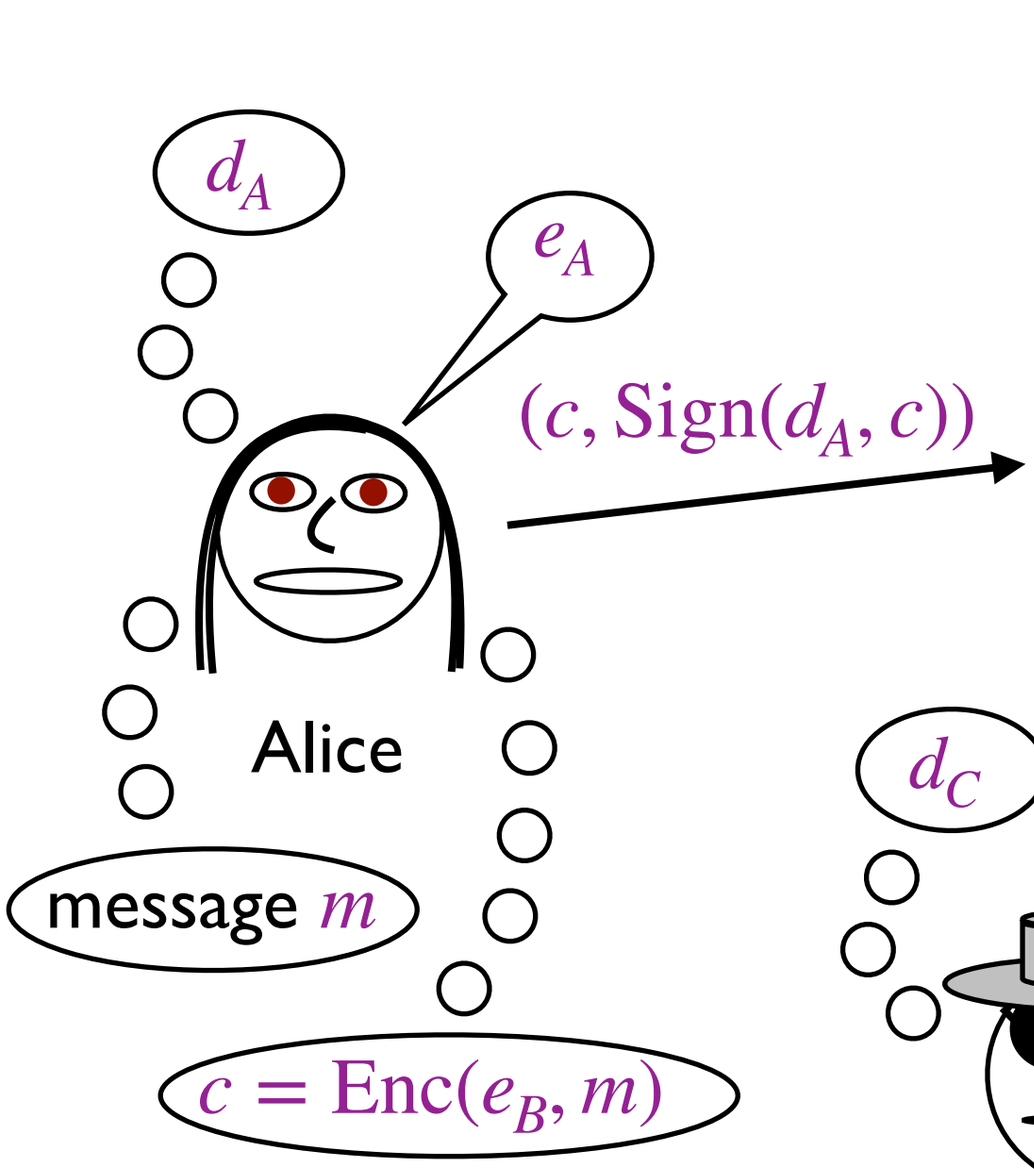
This class is being recorded

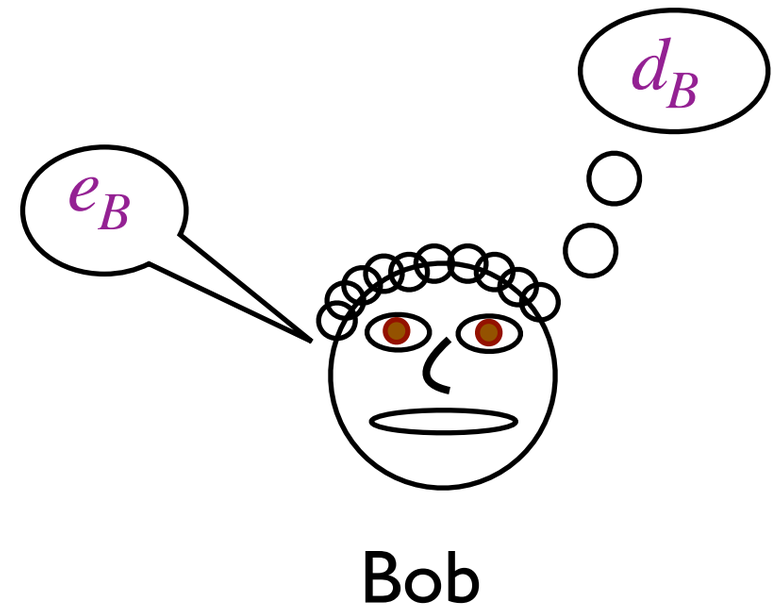
Reattribution

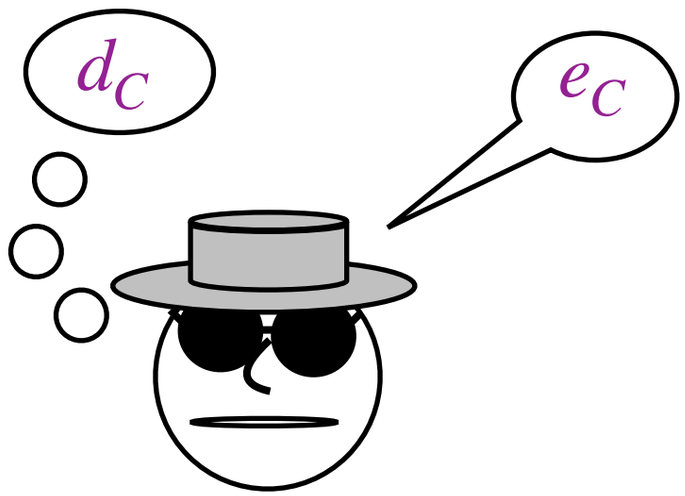


This class is being recorded

Reattribution

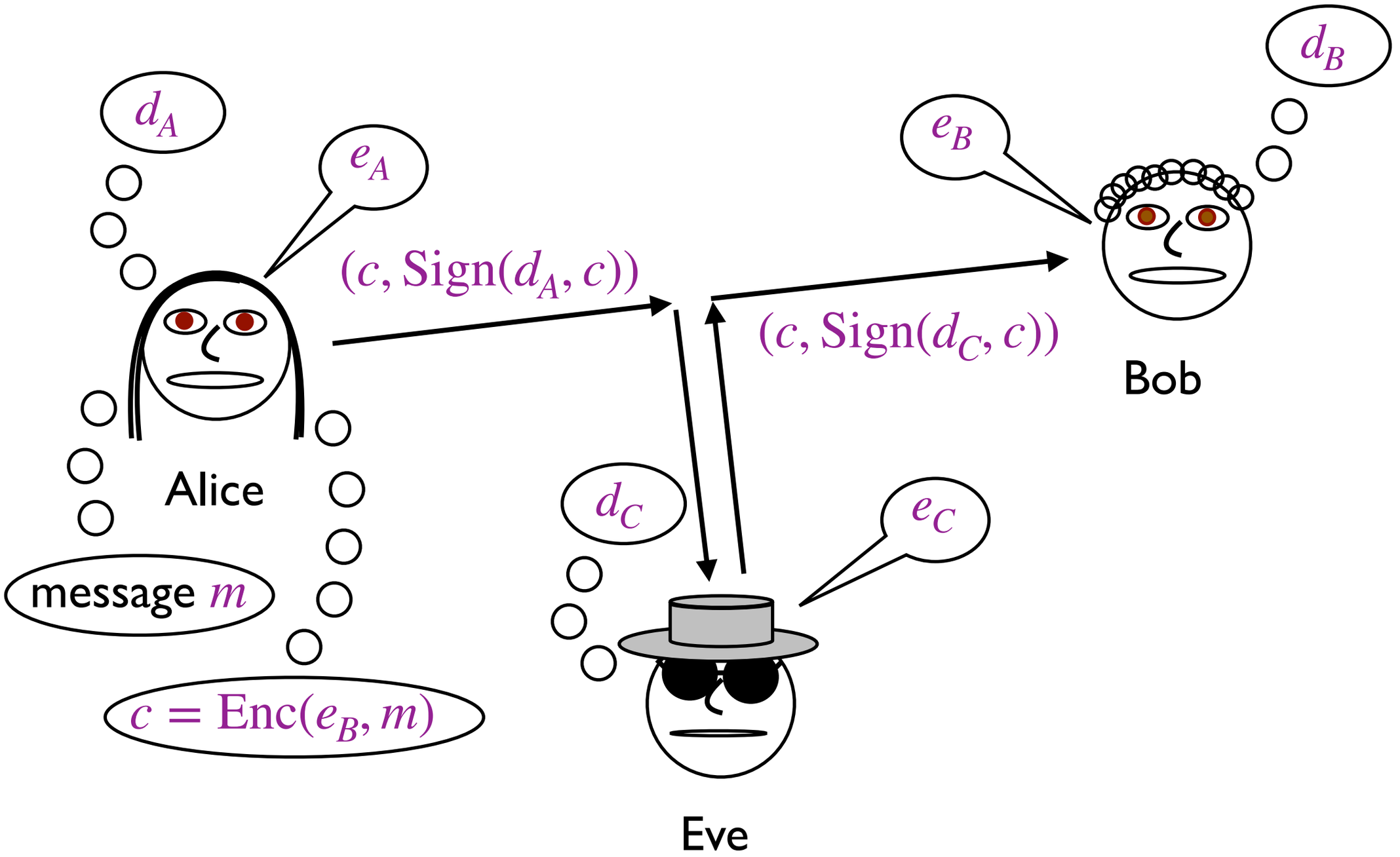






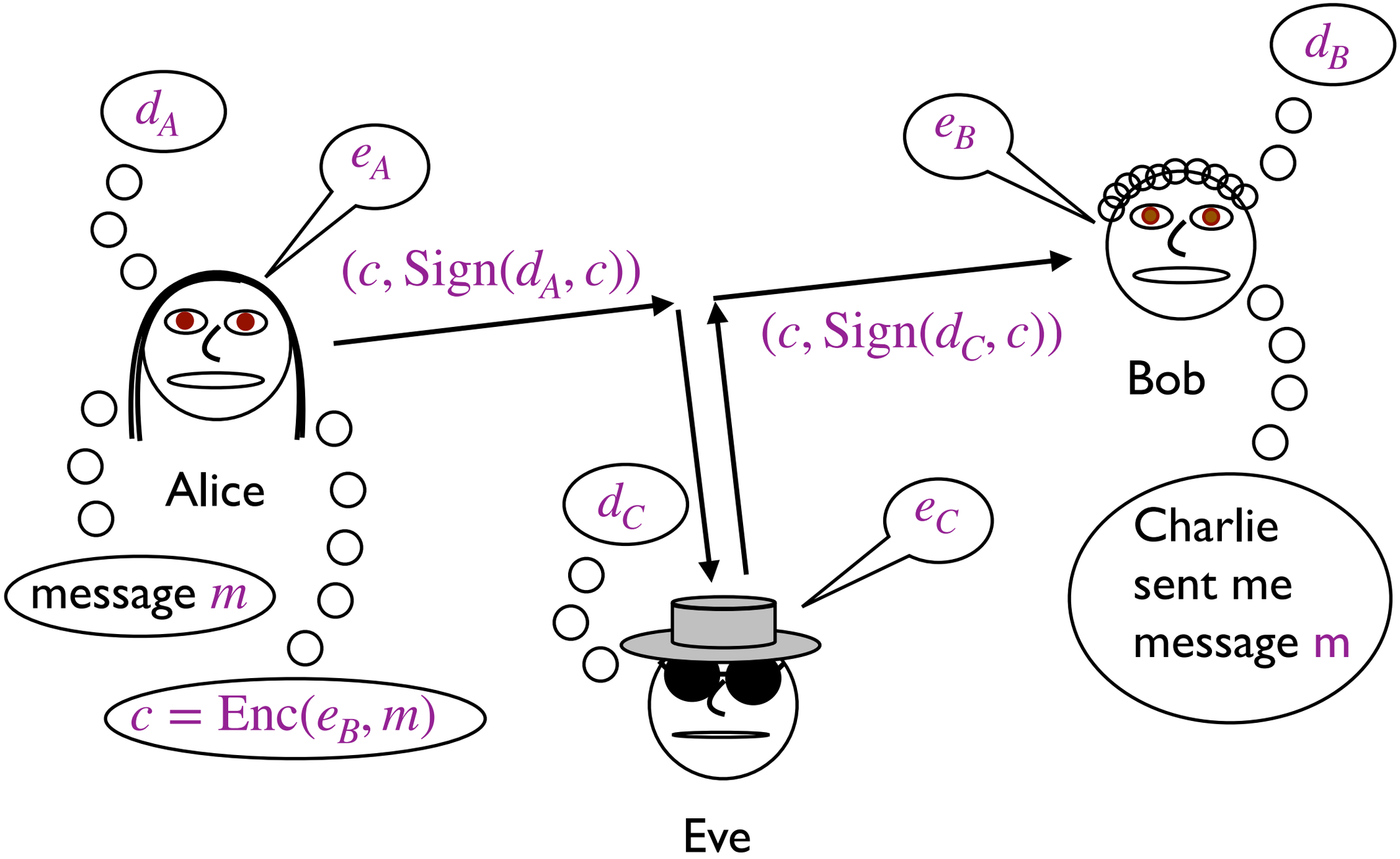
This class is being recorded

Reattribution



This class is being recorded

Reattribution



This class is being recorded

Reattribution Solution

Solution: Make sure to include From: and To: lines in your message.

"From:" should be encrypted using a CCA-secure (and therefore non-malleable) encryption scheme.

"To:" should be signed using a strongly secure signature scheme.

You can use either encrypt-then-authenticate or authenticate-then-encrypt (with the usual cautions about making different kinds of errors indistinguishable).

Note that doing this does not conceal the sender or receiver: That can be achieved via other anonymizing cryptographic protocols.

This class is being recorded