CMSC/Math 456: Cryptography (Fall 2023)

Lecture 16

Daniel Gottesman

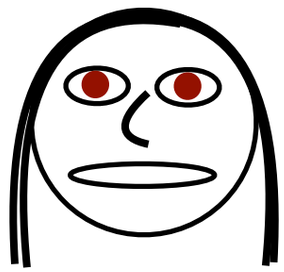
Administrative

I am not sure if there will be a problem set this week.

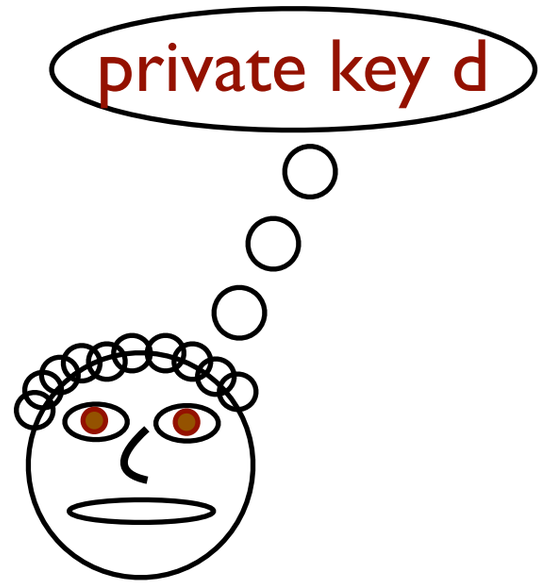
Next week I will be away, and there will be guest lectures by Gorjan Alagic, topic TBA.

This class is being recorded

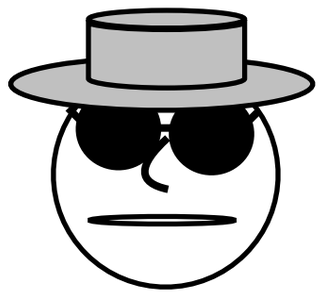
Public Key Encryption



Alice



Bob

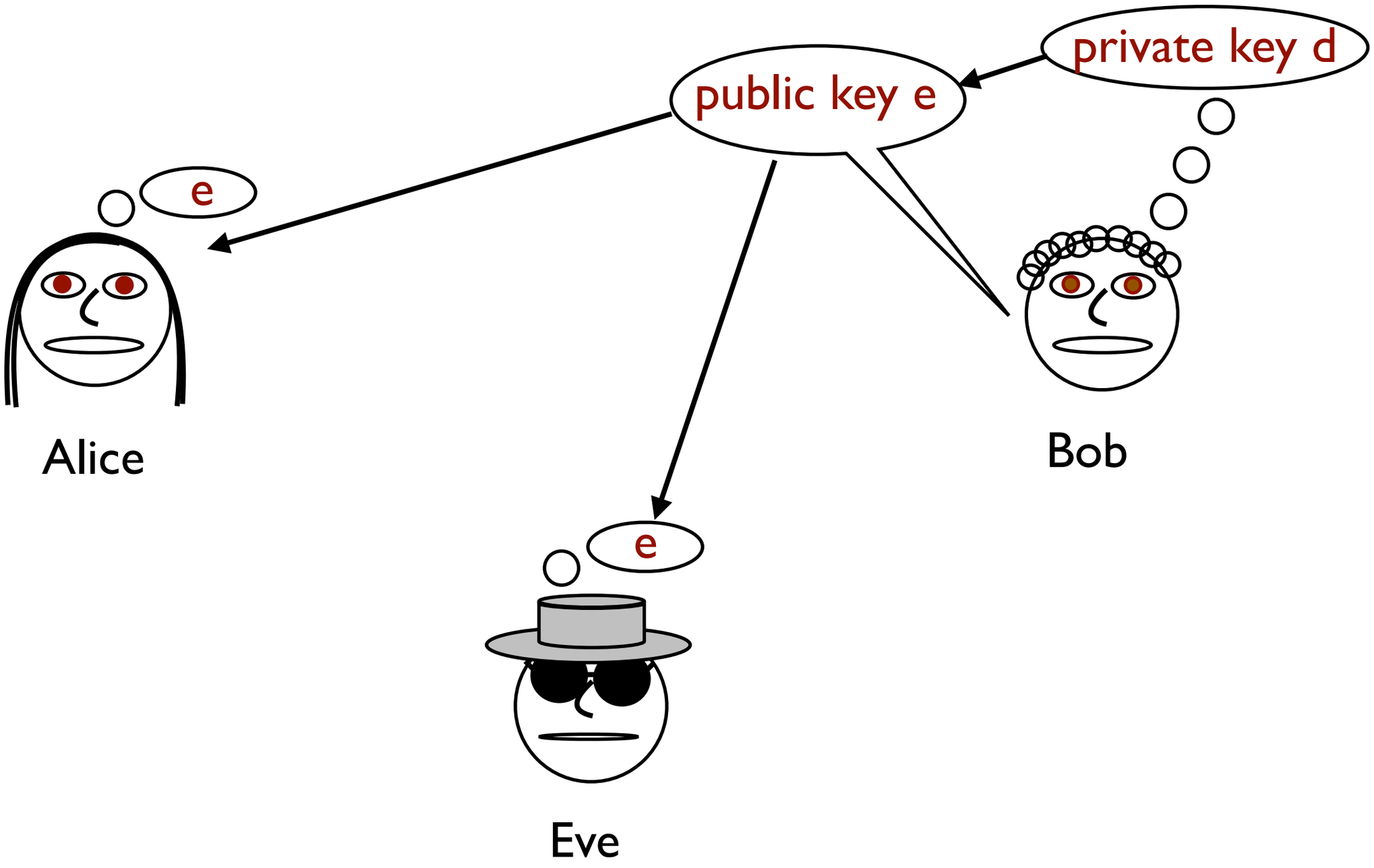


Eve

Public-key encryption is an asymmetric protocol.

This class is being recorded

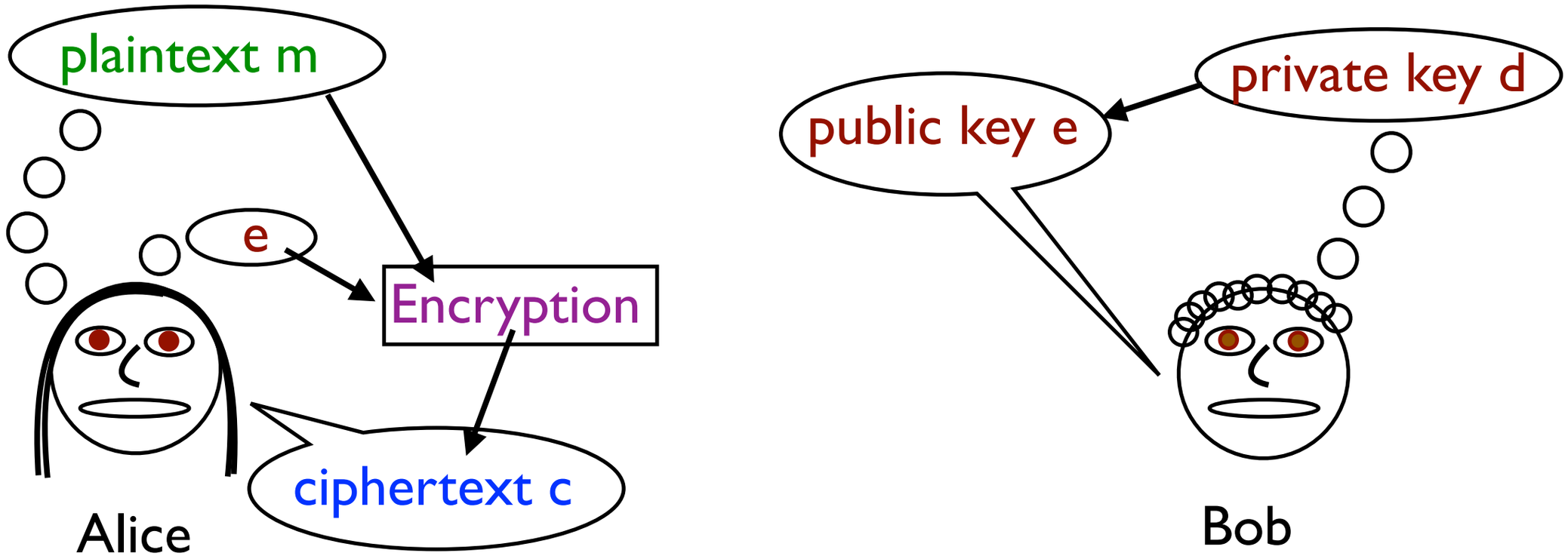
Public Key Encryption

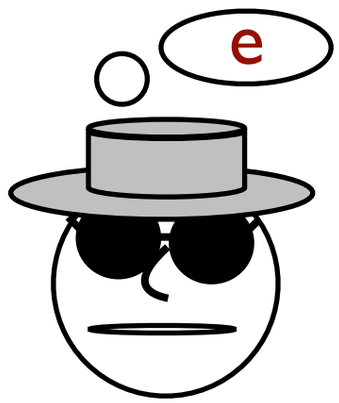


Public-key encryption is an asymmetric protocol.

This class is being recorded

Public Key Encryption



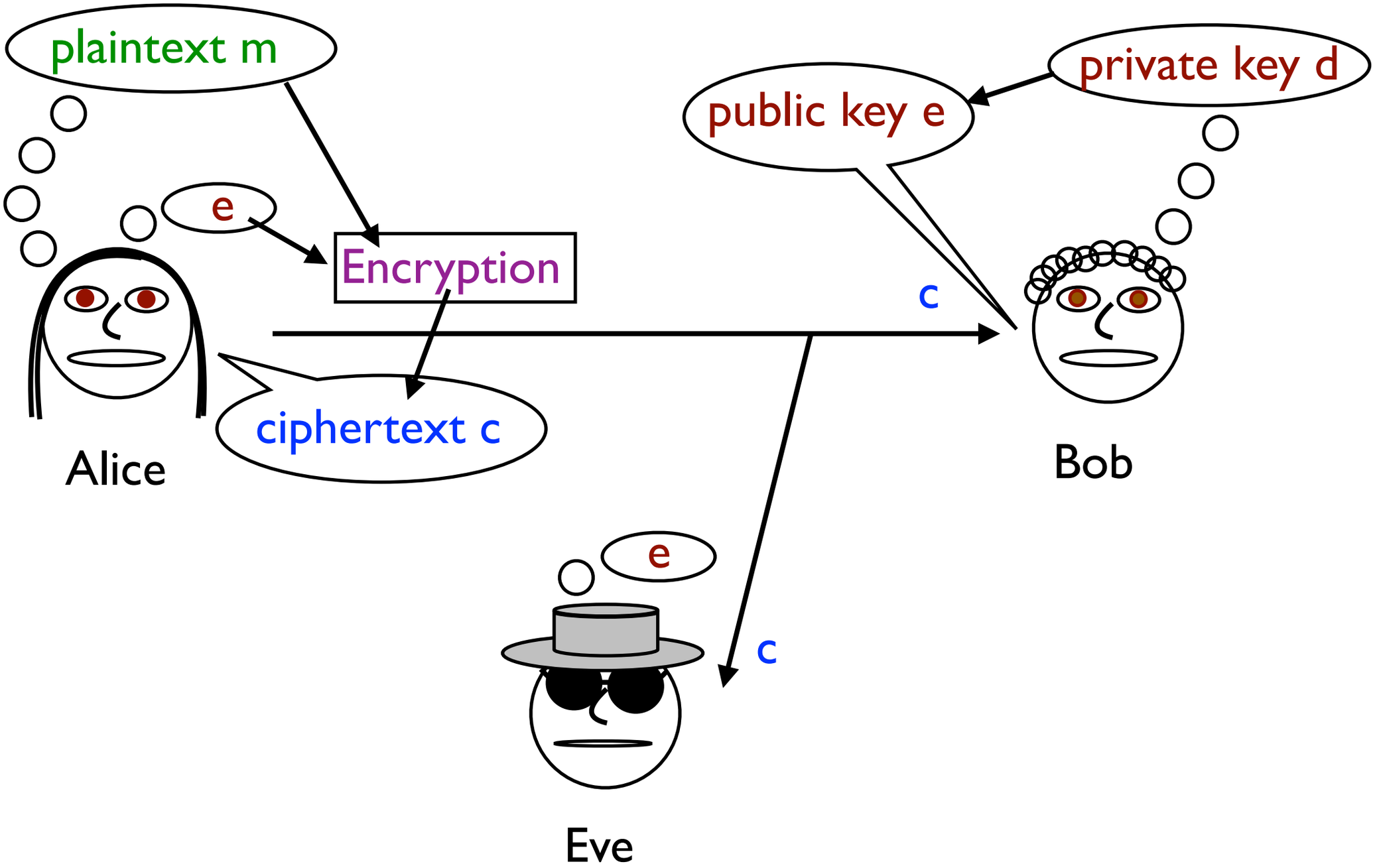


Eve

Public-key encryption is an asymmetric protocol.

This class is being recorded

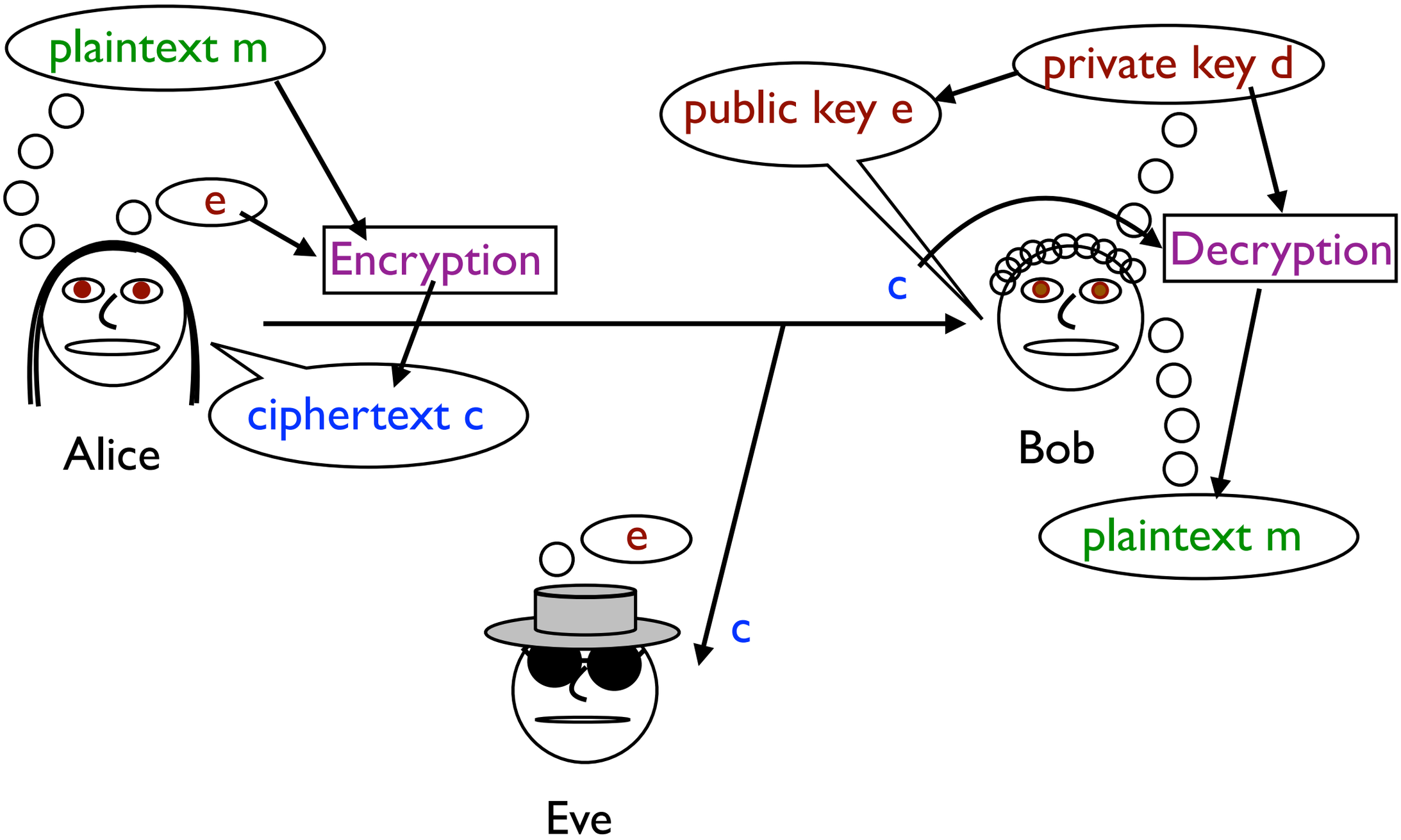
Public Key Encryption



Public-key encryption is an asymmetric protocol.

This class is being recorded

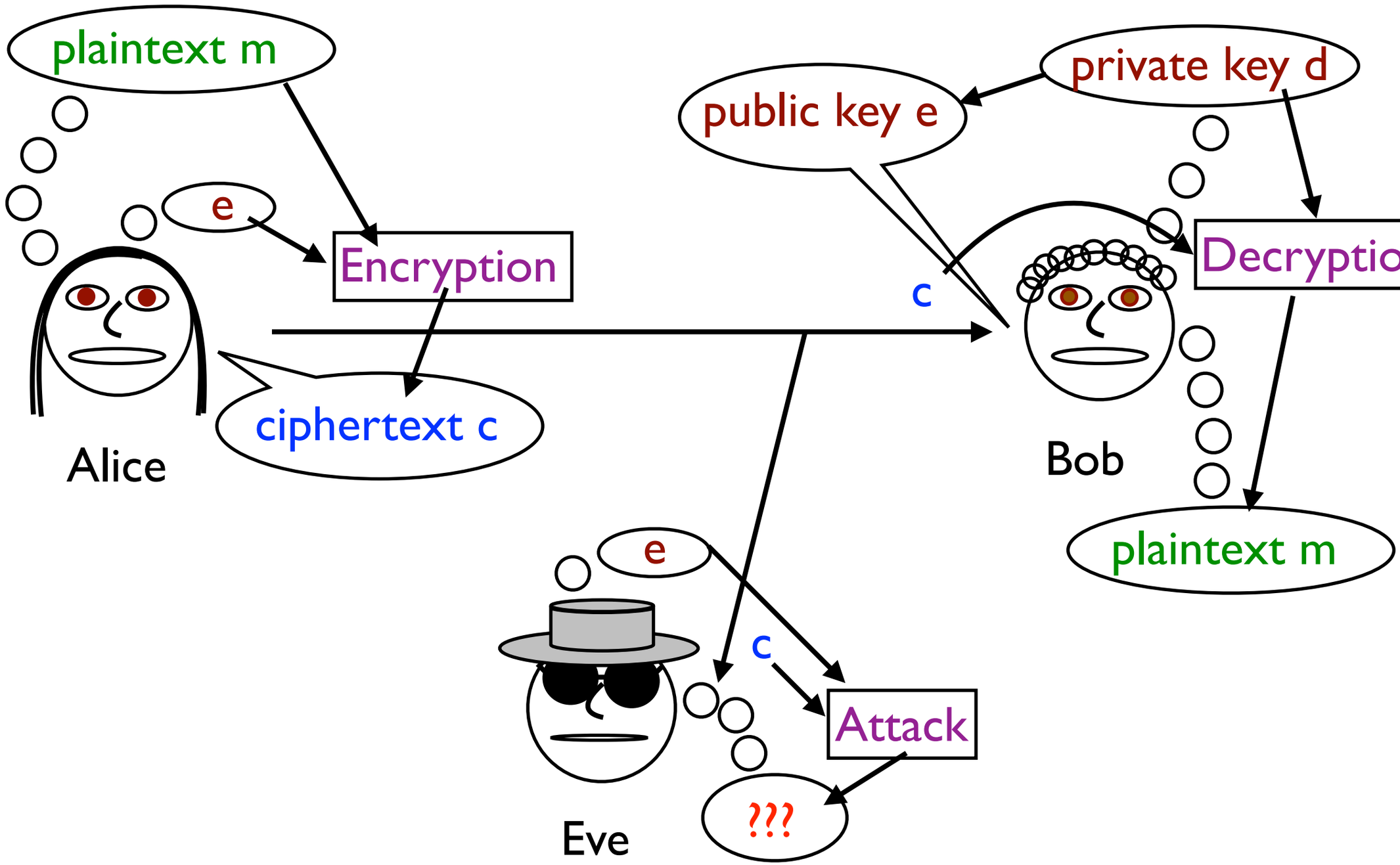
Public Key Encryption



Public-key encryption is an asymmetric protocol.

This class is being recorded

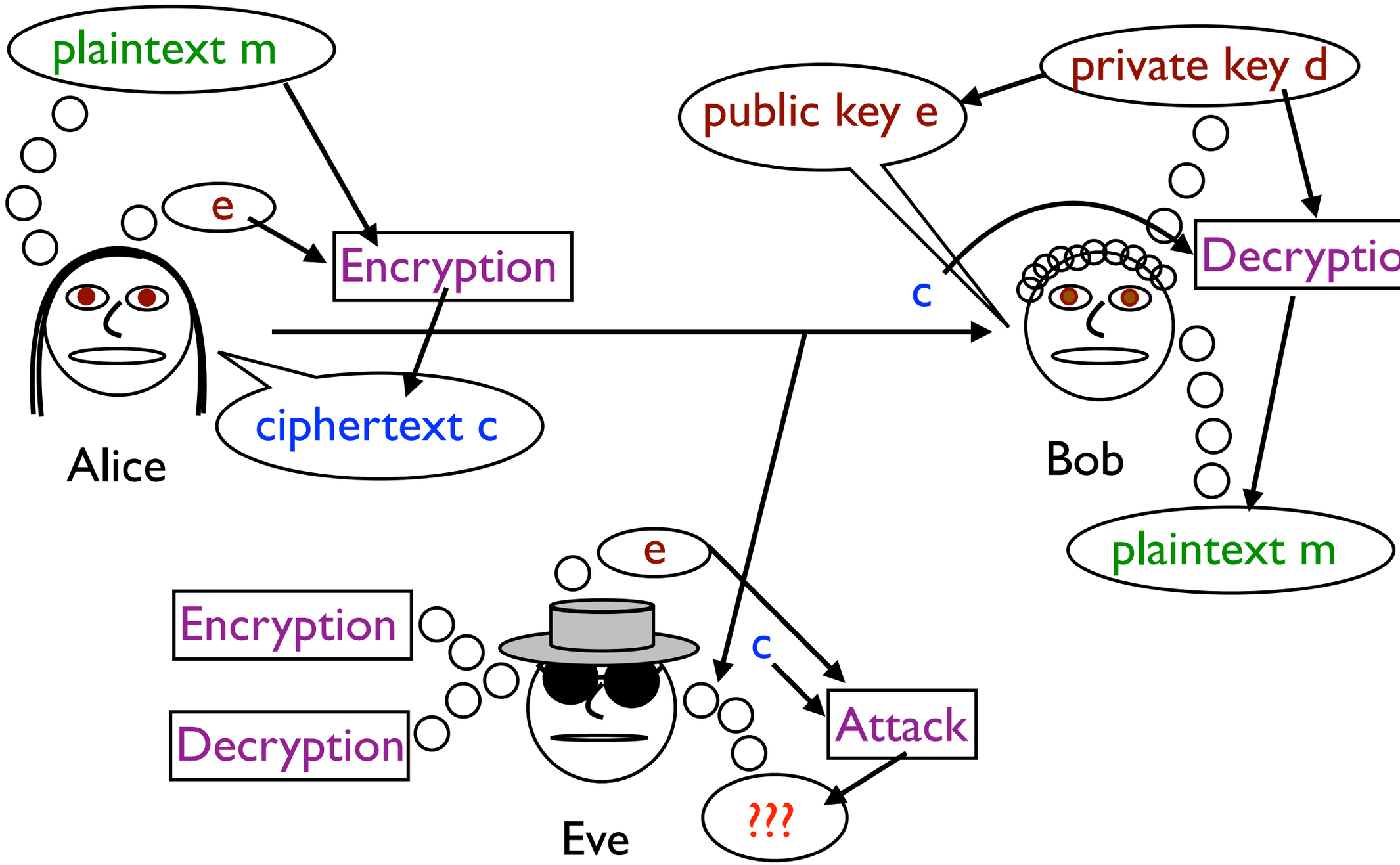
Public Key Encryption



Public-key encryption is an asymmetric protocol.

This class is being recorded

Public Key Encryption



Public-key encryption is an asymmetric protocol.

This class is being recorded

Definition of Public Key Encryption

Definition: A public-key encryption protocol is a set of three probabilistic polynomial-time algorithms (Gen, Enc, Dec).

Gen is the key generation algorithm. It takes as input s, the security parameter, and outputs a public key, private key pair $(e, d) \in \{0,1\}^* \times \{0,1\}^*$.

Enc is the encryption algorithm. It takes as input e and a plaintext or message $m \in \{0,1\}^*$ and outputs a ciphertext $c \in \{0,1\}^*$.

Dec is the decryption algorithm. It takes as input d and c and outputs some $m' \in \{0,1\}^*$.

The encryption protocol is correct if

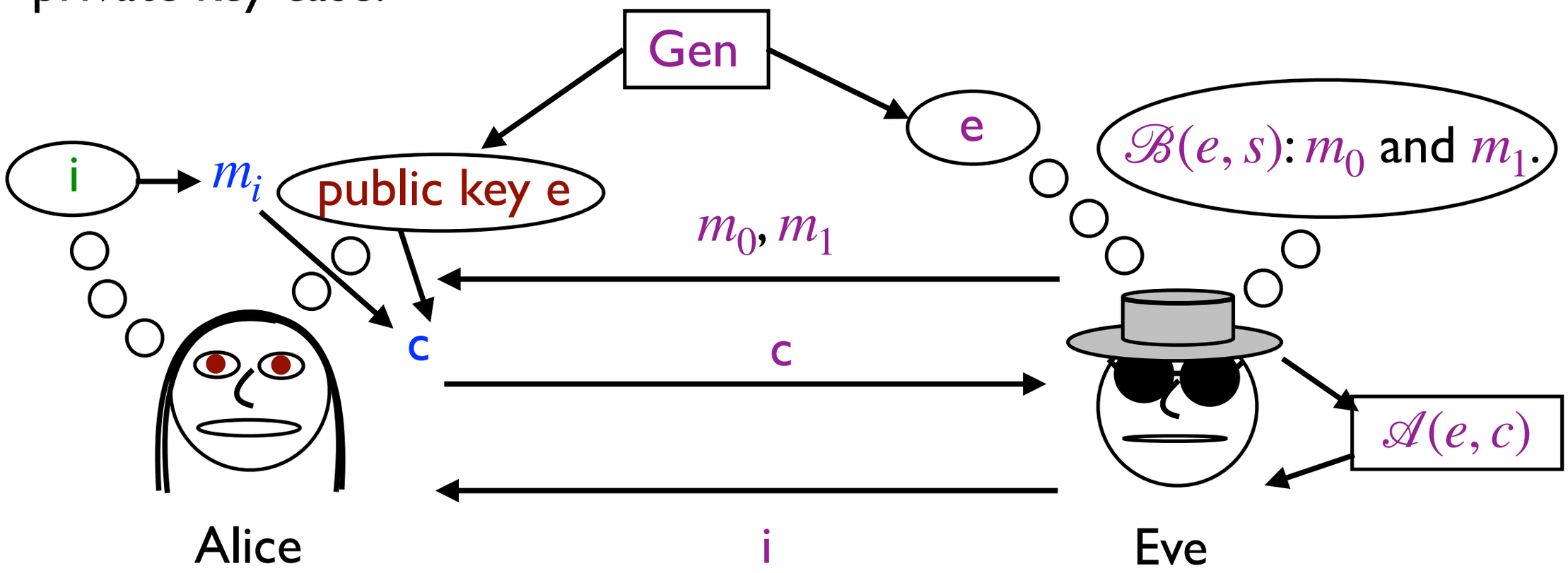
$$Dec(d, Enc(e, m)) = m$$

Note: Gen here is much more complex than for private-key encryption and doesn't just generate random bit strings.

This class is being recorded

Security of Public Key Encryption

Recall that EAV security for public key encryption is the same as CPA security and that both are defined by a game similar to the private key case:



The main difference from private keys is that now Eve is given the public key e. She can use e to choose the messages she wants to use as challenges as well as in her attack to decipher the message.

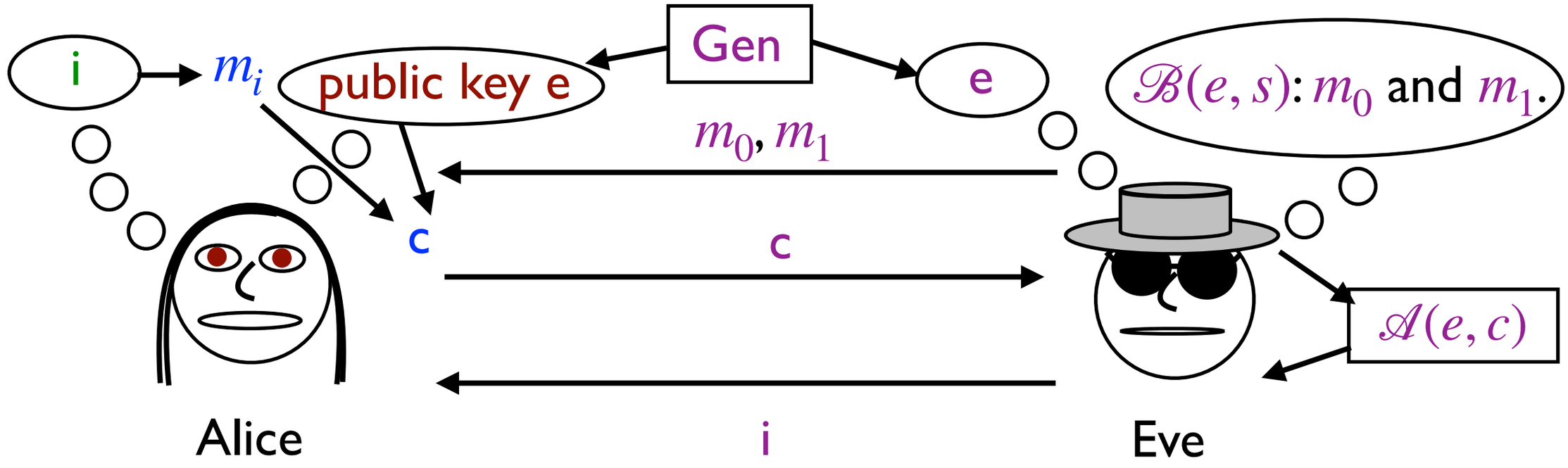
This class is being recorded

Public Key Security

Definition: A public-key encryption protocol (Gen, Enc, Dec) with security parameter s has is EAV-secure and CPA-secure if, for any pair of messages m_0 and m_1 chosen by the adversary (using efficient algorithm $\mathcal{B}(e, s)$) and for any efficient attack $\mathcal{A}(e, c)$,

$$|\Pr_{(e,d)}(\mathcal{A}(e, Enc(e, m_0)) = 1) - \Pr_{(e,d)}(\mathcal{A}(e, Enc(e, m_1)) = 1)| \leq \epsilon(s)$$

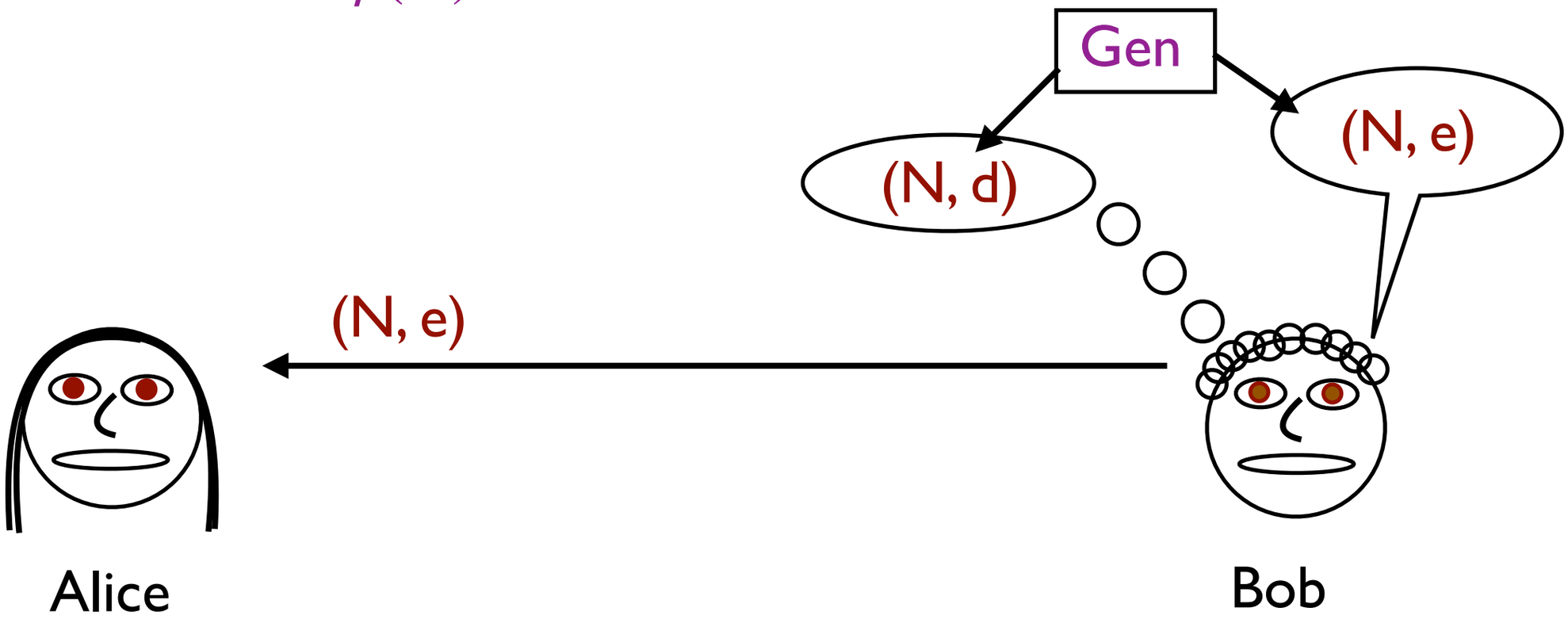
for negligible $\epsilon(s)$ and probability taken over valid public key, private key pairs (e,d) and randomness of Enc, $\mathcal{A}$, and $\mathcal{B}$.



This class is being recorded

RSA Encryption (Picture)

In RSA, Gen creates a public key (N, e) and a private key (N,d) with $ed = 1 \bmod \varphi(N)$.

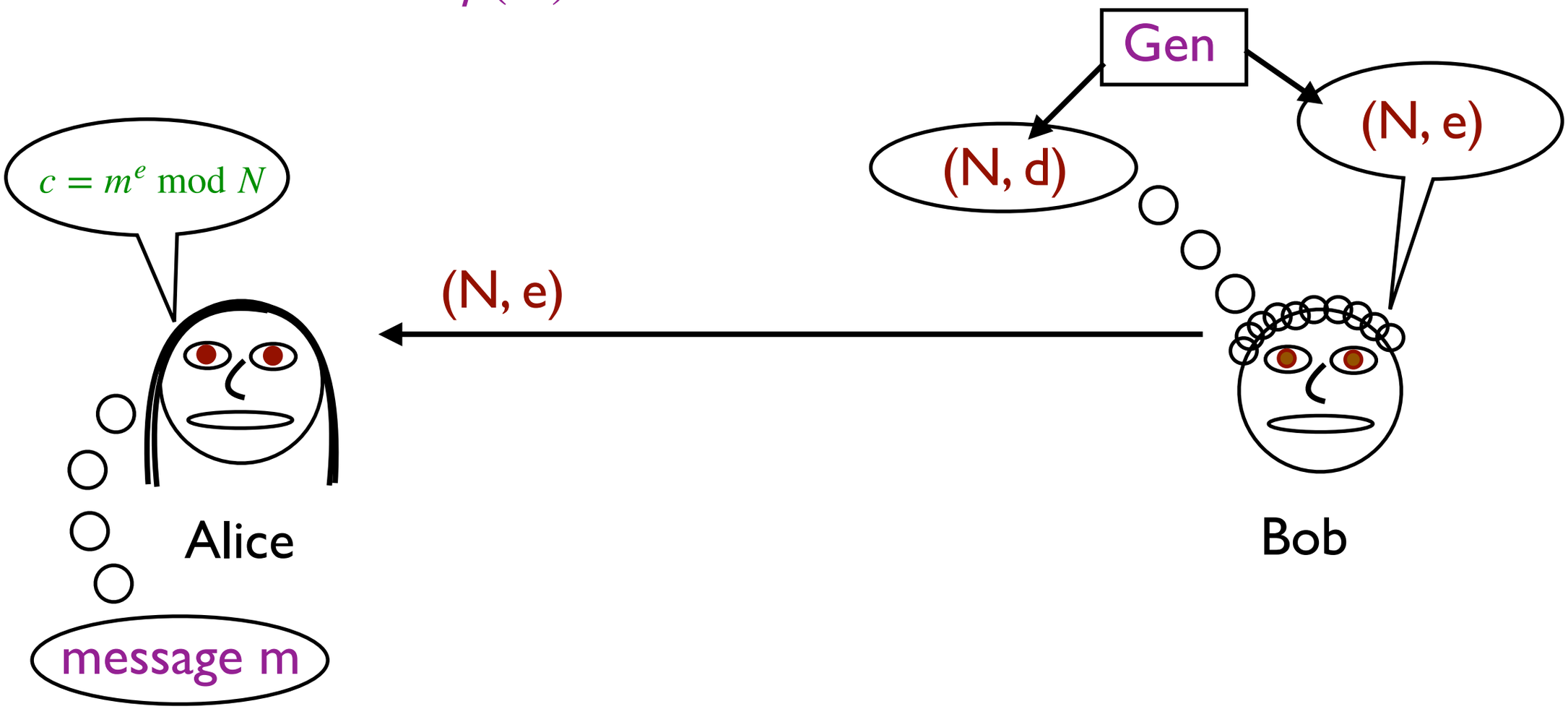


"RSA" stands for the inventors: Rivest, Shamir, and Adleman

This class is being recorded

RSA Encryption (Picture)

In RSA, Gen creates a public key (N, e) and a private key (N,d) with $ed = 1 \bmod \varphi(N)$.

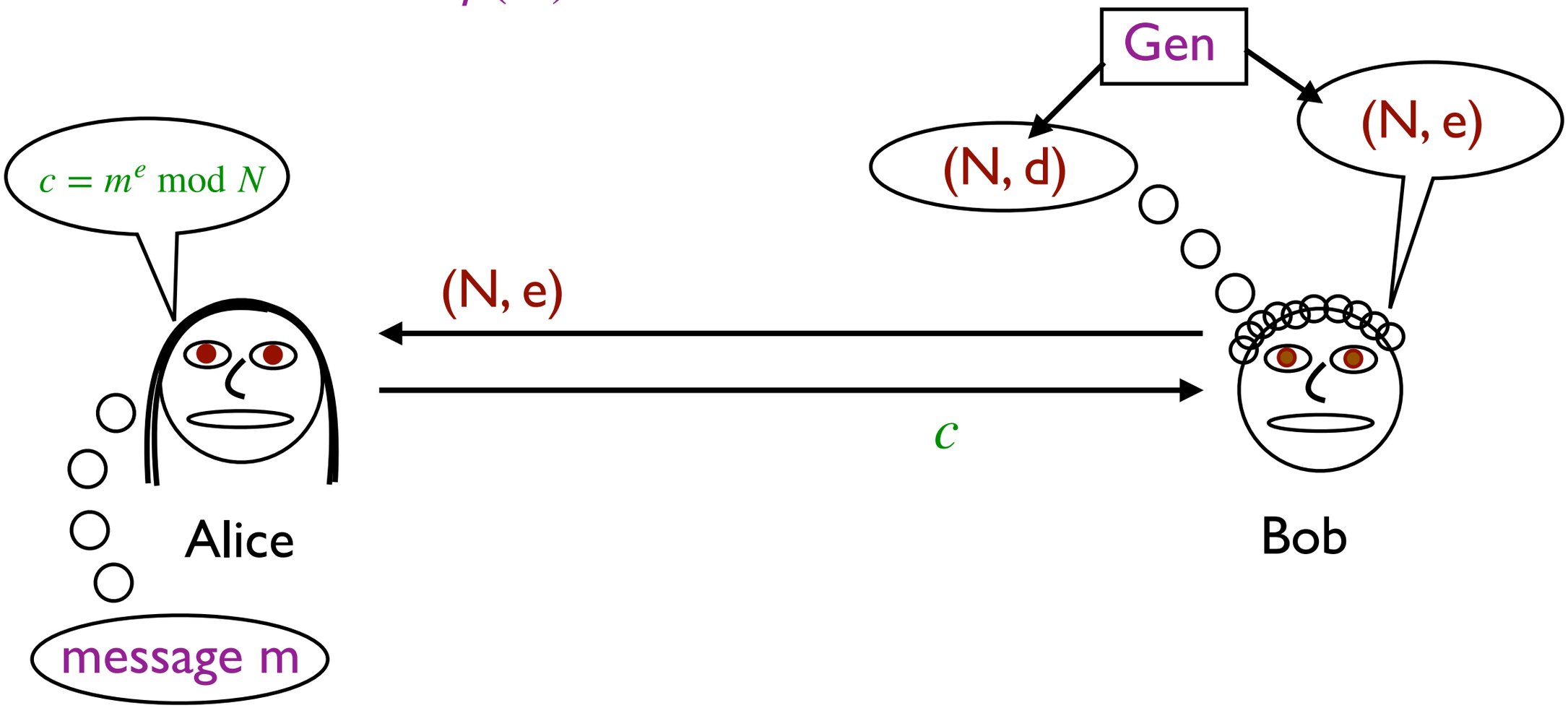


"RSA" stands for the inventors: Rivest, Shamir, and Adleman

This class is being recorded

RSA Encryption (Picture)

In RSA, Gen creates a public key (N, e) and a private key (N,d) with $ed = 1 \bmod \varphi(N)$.

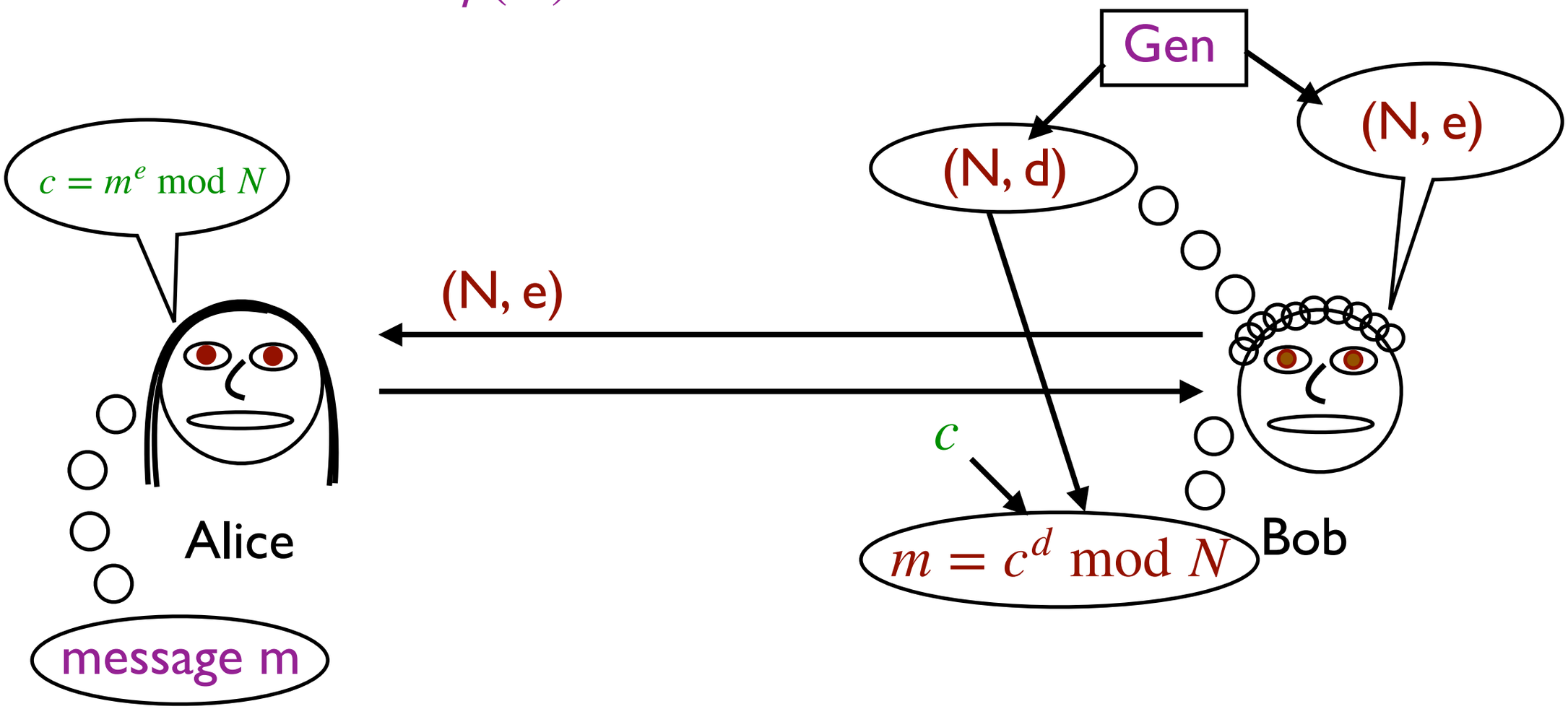


"RSA" stands for the inventors: Rivest, Shamir, and Adleman

This class is being recorded

RSA Encryption (Picture)

In RSA, Gen creates a public key (N, e) and a private key (N,d) with $ed = 1 \bmod \varphi(N)$.



"RSA" stands for the inventors: Rivest, Shamir, and Adleman

This class is being recorded

RSA Encryption (Gen, Enc, Dec)

Gen: Generate two random primes p and q which are s bits long. Let N = pq. Choose $e, d \in \mathbb{Z}^*_{\varphi(N)}$ such that $ed = 1 \bmod \varphi(N)$.

The public key is (N, e) and the private key is (N, d).

Enc: Given message m and public key (N, e). The ciphertext is $c = m^e \bmod N$.

Dec: Given ciphertext c and private key (N, d). The decrypted message is $m' = c^d \bmod N$.

This class is being recorded

RSA Encryption (Gen, Enc, Dec)

Gen: Generate two random primes p and q which are s bits long. Let N = pq. Choose $e, d \in \mathbb{Z}^*_{\varphi(N)}$ such that $ed = 1 \bmod \varphi(N)$.

The public key is (N, e) and the private key is (N, d).

Enc: Given message m and public key (N, e). The ciphertext is $c = m^e \bmod N$.

Dec: Given ciphertext c and private key (N, d). The decrypted message is $m' = c^d \bmod N$.

Example:

Gen: Let p = 11, q = 17. Then N = 187 and $\varphi(N) = (p-1)(q-1) = 10 \cdot 16 = 160$. Let e = 3; then d = 107 (so $3 \cdot 107 = 1 \bmod 160$).

Enc: Let m = 113. Then $c = 113^3 \bmod 187 = 5$.

Dec: $m' = 5^{107} \bmod 187 = 113$.

This class is being recorded

Why is RSA Correct?

Why does $\mathrm{Dec}(d, \mathrm{Enc}(e, m)) = m$?

Recall the Euler-Fermat theorem: $x^{\varphi(N)} = 1 \bmod N$.

Then

$$\mathrm{Dec}(d, \mathrm{Enc}(e, m)) = m^{ed} \bmod N$$

But

$$ed = 1 + k\varphi(N)$$

so

$$\begin{aligned}\mathrm{Dec}(d, \mathrm{Enc}(e, m)) &= m^{1+k\varphi(N)} \bmod N \\ &= m \cdot (m^{\varphi(N)})^k \bmod N \\ &= m \cdot 1^k \bmod N\end{aligned}$$

This class is being recorded

Picking e and d

Gen: Generate two random primes p and q which are s bits long.
Let N = pq. Choose $e, d \in \mathbb{Z}^*_{\varphi(N)}$ such that $ed = 1 \bmod \varphi(N)$.

The public key is (N, e) and the private key is (N, d).

How do we pick e and d?

This class is being recorded

Picking e and d

Gen: Generate two random primes p and q which are s bits long.
Let N = pq. Choose $e, d \in \mathbb{Z}^*_{\varphi(N)}$ such that $ed = 1 \bmod \varphi(N)$.

The public key is (N, e) and the private key is (N, d).

How do we pick e and d?

1. From p and q, we can find $\varphi(N) = (p-1)(q-1)$.

This class is being recorded

Picking e and d

Gen: Generate two random primes p and q which are s bits long.
Let N = pq. Choose $e, d \in \mathbb{Z}^*_{\varphi(N)}$ such that $ed = 1 \bmod \varphi(N)$.

The public key is (N, e) and the private key is (N, d).

How do we pick e and d?

1. From p and q, we can find $\varphi(N) = (p-1)(q-1)$.
2. Choose a convenient value for $e \in \mathbb{Z}^*_{\varphi(N)}$. It doesn't need to be random. A common choice is e=3.

This class is being recorded

Picking e and d

Gen: Generate two random primes p and q which are s bits long. Let N = pq. Choose $e, d \in \mathbb{Z}^*_{\varphi(N)}$ such that $ed = 1 \bmod \varphi(N)$.

The public key is (N, e) and the private key is (N, d).

How do we pick e and d?

1. From p and q, we can find $\varphi(N) = (p-1)(q-1)$.
2. Choose a convenient value for $e \in \mathbb{Z}^*_{\varphi(N)}$. It doesn't need to be random. A common choice is e=3.
3. Then use Euclid's algorithm. Since $e \in \mathbb{Z}^*_{\varphi(N)}$, e and $\varphi(N)$ are relatively prime, so we find X,Y such that:

$$Xe + Y\varphi(N) = 1$$

This class is being recorded

Picking e and d

Gen: Generate two random primes p and q which are s bits long. Let N = pq. Choose $e, d \in \mathbb{Z}^*_{\varphi(N)}$ such that $ed = 1 \bmod \varphi(N)$.

The public key is (N, e) and the private key is (N, d).

How do we pick e and d?

1. From p and q, we can find $\varphi(N) = (p-1)(q-1)$.
2. Choose a convenient value for $e \in \mathbb{Z}^*_{\varphi(N)}$. It doesn't need to be random. A common choice is e=3.
3. Then use Euclid's algorithm. Since $e \in \mathbb{Z}^*_{\varphi(N)}$, e and $\varphi(N)$ are relatively prime, so we find X, Y such that:

$$Xe + Y\varphi(N) = 1$$

4. Then we can let d=X, because

$$Xe = 1 - Y\varphi(N) = 1 \bmod \varphi(N)$$

This class is being recorded

RSA Q&A

- Why pick p and q instead of picking N directly?

This class is being recorded

RSA Q&A

- Why pick p and q instead of picking N directly?

  To pick e and d, we need to know $\varphi(N)$, and to find $\varphi(N)$, we need to know the prime factorization of N. Factoring seems hard, so we pick the prime factors directly.

This class is being recorded

RSA Q&A

- Why pick p and q instead of picking N directly?

  To pick e and d, we need to know $\varphi(N)$, and to find $\varphi(N)$, we need to know the prime factorization of N. Factoring seems hard, so we pick the prime factors directly.

- Eve knows e. Can't she also apply Euclid's algorithm to find e?

This class is being recorded

RSA Q&A

- Why pick p and q instead of picking N directly?

  To pick e and d, we need to know $\varphi(N)$, and to find $\varphi(N)$, we need to know the prime factorization of N. Factoring seems hard, so we pick the prime factors directly.

- Eve knows e. Can't she also apply Euclid's algorithm to find e?

  Only if she knows $\varphi(N)$. But to compute $\varphi(N)$, it seems she needs to know the prime factorization of N. And factoring seems hard.

This class is being recorded

Hardness of Factoring

Factoring a product of two large primes seems to be hard.

Definition: Given a security parameter s, let P(N) be a probability distribution over (2s)-bit numbers. We say that factoring is average-case hard for P(N) if for *any* polynomial time algorithm $\mathscr{A}$, for random N chosen according to P(N),

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

for $\epsilon(s)$ a negligible function, where we say $\mathscr{A}(N)$ succeeds if $\mathscr{A}(N) = p$ with p a non-trivial factor of N: $1 < p < N$ and $p \,|\, N$.

When N is chosen to be the product of two random s-bit primes, it appears that factoring is hard for the resulting distribution.

This class is being recorded

Factoring and RSA

If Eve can factor N, she can break RSA:

Eve factors N = pq and computes $\varphi(N) = (p-1)(q-1)$.
Using Euclid's algorithm, she computes d given e and $\varphi(N)$.

This class is being recorded

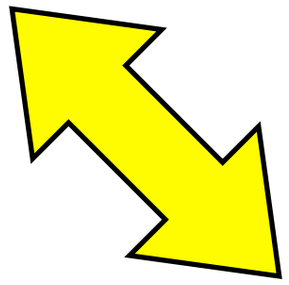
Factoring and RSA

If Eve can factor N, she can break RSA:

Eve factors N = pq and computes $\varphi(N) = (p-1)(q-1)$.
Using Euclid's algorithm, she computes d given e and $\varphi(N)$.

Theorem: Given N, if Eve knows d and e, she has an efficient probabilistic algorithm to factor N.

Factoring N



Learning the private key for RSA

This class is being recorded

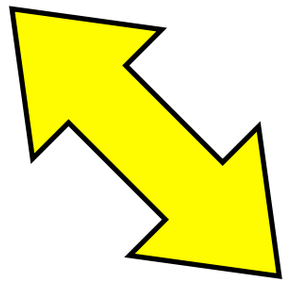
Factoring and RSA

If Eve can factor N, she can break RSA:

Eve factors N = pq and computes $\varphi(N) = (p-1)(q-1)$.
Using Euclid's algorithm, she computes d given e and $\varphi(N)$.

Theorem: Given N, if Eve knows d and e, she has an efficient probabilistic algorithm to factor N.

Factoring N



Learning the private key for RSA

But perhaps it is possible to break RSA without learning the private key ….

This class is being recorded

Is RSA Secure?

Vote: Is RSA as I've presented it CPA-secure? (Yes/No/Unknown)

This class is being recorded

Is RSA Secure?

Vote: Is RSA as I've presented it CPA-secure? (Yes/No/Unknown)

Answer: No.

This version of RSA (Plain RSA) is deterministic: It cannot be CPA-secure.

If Eve is trying to determine which of two messages was sent, she can just try encrypting each of them using the public key. In Plain RSA, the message m_i always corresponds to ciphertext $c_i = m_i^e \bmod N$, so she can easily find i by matching $c = c_i$.

This class is being recorded

Is RSA Secure?

Vote: Is RSA as I've presented it CPA-secure? (Yes/No/Unknown)

Answer: No.

This version of RSA (Plain RSA) is deterministic: It cannot be CPA-secure.

If Eve is trying to determine which of two messages was sent, she can just try encrypting each of them using the public key. In Plain RSA, the message m_i always corresponds to ciphertext $c_i = m_i^e \bmod N$, so she can easily find i by matching $c = c_i$.

We need to add a random element to Enc.

This class is being recorded

Padded RSA

One option to add randomness to RSA is to pad the message with random bits:

Gen remains the same.

Enc: Given message m, generate ℓ-bit random r and let $\tilde{m} = r||m$ (the concatenation r followed by m). Then the ciphertext $c = \tilde{m}^e \bmod N$.

Dec: Given c, compute $\tilde{m}' = c^d \bmod N$. Discard the first ℓ bits of $\tilde{m}'$ to get m'.

This class is being recorded

Padded RSA

One option to add randomness to RSA is to pad the message with random bits:

Gen remains the same.

Enc: Given message m, generate ℓ-bit random r and let $\tilde{m} = r||m$ (the concatenation r followed by m). Then the ciphertext $c = \tilde{m}^e \bmod N$.

Dec: Given c, compute $\tilde{m}' = c^d \bmod N$. Discard the first ℓ bits of $\tilde{m}'$ to get m'.

Example:

Gen: N = 187, e = 3, and d = 107 as before.

Enc: Let $m = 15 = 1111_2$. Pad using $r = 110_2$. Then $\tilde{m} = 1101111_2 = 111$ and $c = 111^3 \bmod 187 = 100$.

Dec: $\tilde{m}' = 100^{107} \bmod 187 = 111$ and m' = 15.

This class is being recorded

Security of Padded RSA

If $\ell = \lfloor \log_2 N \rfloor - 1$, Padded RSA is CPA-secure assuming security of an RSA assumption: It is hard to find x such that $x^e = y \bmod N$.

Note: This is a stronger assumption than factoring being hard, as far as we know.

It also implies just 1-bit messages.

Shorter padding is not known to be secure (at least without a stronger assumption), but is plausibly secure as long as ℓ is not too short.

RSA PKCS #1 v1.5 is a variant of padded RSA where some of the padded bits are random and some have fixed values. With a good choice of parameters, it is probably CPA-secure in the same sense as Padded RSA.

This class is being recorded

Another Attack on RSA

Suppose e and m are both small: $m^e < N$.

This class is being recorded

Another Attack on RSA

Suppose e and m are both small: $m^e < N$.

Then the modular arithmetic is irrelevant, since $m^e \in \mathbb{Z}_N^*$ is the same as $m^e \in \mathbb{Z}$ (the integers).

This class is being recorded

Another Attack on RSA

Suppose e and m are both small: $m^e < N$.

Then the modular arithmetic is irrelevant, since $m^e \in \mathbb{Z}_N^*$ is the same as $m^e \in \mathbb{Z}$ (the integers).

Here's an attack that takes advantage of this:

This class is being recorded

Another Attack on RSA

Suppose e and m are both small: $m^e < N$.

Then the modular arithmetic is irrelevant, since $m^e \in \mathbb{Z}_N^*$ is the same as $m^e \in \mathbb{Z}$ (the integers).

Here's an attack that takes advantage of this:

1. Guess a value m_0 for m. Compute m_0^e.

This class is being recorded

Another Attack on RSA

Suppose e and m are both small: $m^e < N$.

Then the modular arithmetic is irrelevant, since $m^e \in \mathbb{Z}_N^*$ is the same as $m^e \in \mathbb{Z}$ (the integers).

Here's an attack that takes advantage of this:

1. Guess a value m_0 for m. Compute m_0^e.
2. If $m_0^e < m^e$, guess a new m_1 that is larger.

This class is being recorded

Another Attack on RSA

Suppose e and m are both small: $m^e < N$.

Then the modular arithmetic is irrelevant, since $m^e \in \mathbb{Z}_N^*$ is the same as $m^e \in \mathbb{Z}$ (the integers).

Here's an attack that takes advantage of this:

1. Guess a value m_0 for m. Compute m_0^e.
2. If $m_0^e < m^e$, guess a new m_1 that is larger.
3. Otherwise, guess a new smaller m_1.

This class is being recorded

Another Attack on RSA

Suppose e and m are both small: $m^e < N$.

Then the modular arithmetic is irrelevant, since $m^e \in \mathbb{Z}_N^*$ is the same as $m^e \in \mathbb{Z}$ (the integers).

Here's an attack that takes advantage of this:

1. Guess a value m_0 for m. Compute m_0^e.
2. If $m_0^e < m^e$, guess a new m_1 that is larger.
3. Otherwise, guess a new smaller m_1.
4. Repeat steps 1-3, narrowing the search space exponentially (for instance, by choosing an m_i halfway between the current upper and lower bounds) until $m_i^e = m^e$.

This class is being recorded

Another Attack on RSA

Suppose e and m are both small: $m^e < N$.

Then the modular arithmetic is irrelevant, since $m^e \in \mathbb{Z}_N^*$ is the same as $m^e \in \mathbb{Z}$ (the integers).

Here's an attack that takes advantage of this:

1. Guess a value m_0 for m. Compute m_0^e.
2. If $m_0^e < m^e$, guess a new m_1 that is larger.
3. Otherwise, guess a new smaller m_1.
4. Repeat steps 1-3, narrowing the search space exponentially (for instance, by choosing an m_i halfway between the current upper and lower bounds) until $m_i^e = m^e$.

Why doesn't this work if $m^e > N$?

This class is being recorded

Another Attack on RSA

Suppose e and m are both small: $m^e < N$.

Then the modular arithmetic is irrelevant, since $m^e \in \mathbb{Z}_N^*$ is the same as $m^e \in \mathbb{Z}$ (the integers).

Here's an attack that takes advantage of this:

1. Guess a value m_0 for m. Compute m_0^e.
2. If $m_0^e < m^e$, guess a new m_1 that is larger.
3. Otherwise, guess a new smaller m_1.
4. Repeat steps 1-3, narrowing the search space exponentially (for instance, by choosing an m_i halfway between the current upper and lower bounds) until $m_i^e = m^e$.

Why doesn't this work if $m^e > N$?

Because < and > are not well-defined in modular arithmetic!

This class is being recorded

Another Attack on RSA

Suppose e and m are both small: $m^e < N$.

Then the modular arithmetic is irrelevant, since $m^e \in \mathbb{Z}_N^*$ is the same as $m^e \in \mathbb{Z}$ (the integers).

Here's an attack that takes advantage of this:

1. Guess a value m_0 for m. Compute m_0^e.
2. If $m_0^e < m^e$, guess a new m_1 that is larger.
3. Otherwise, guess a new smaller m_1.
4. Repeat steps 1-3, narrowing the search space exponentially (for instance, by choosing an m_i halfway between the current upper and lower bounds) until $m_i^e = m^e$.

Why doesn't this work if $m^e > N$?

Because < and > are not well-defined in modular arithmetic!

Taking e^{th} roots is easy in $\mathbb{Z}$ and hard in $\mathbb{Z}_N^*$!

This class is being recorded

Picking e

How do we pick e?

- e must be relatively prime to $\varphi(N)$ so that it has an inverse $\bmod \varphi(N)$.
- If e and m are both small ($m^e < N$), decrypting is easy.
- But when we pad the high bits of m, this is very unlikely, so any e works.
- Might as well pick an e so that it is easy to calculate $m^e \bmod N$.
  - Using repeated squaring, the # of terms to multiply together is the # of 1's in the binary representation of e.
  - So pick e with weight 2, e.g., e=3.
  - Or maybe a larger e of the form $2^a + 1$ to avoid the possible small e and small m attack.

This class is being recorded

Algorithms for Factoring

A factoring algorithm breaks RSA … so how hard is factoring?

Similar to discrete log:

- When N=pq and p-1 is a product of small primes, Pollard's p-1 algorithm factors efficiently.
- For general N, the number field sieve runs in time $2^{O((\log N)^{1/3}(\log\log N)^{2/3})}$ (apparently). This is sub-exponential.
- Except: A quantum computer can efficiently factor arbitrary N.

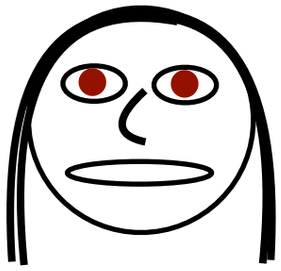
So recommended key lengths for secure variants of RSA are 2048 bits and higher.

This class is being recorded

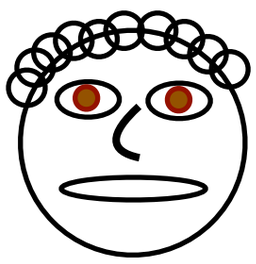
KEM/DEM

Recall that a KEM creates and sends a random key string to someone whose public key you have.

> That key then becomes the key for a private-key encryption system, which is here called a data-encapsulation mechanism (DEM)



Alice



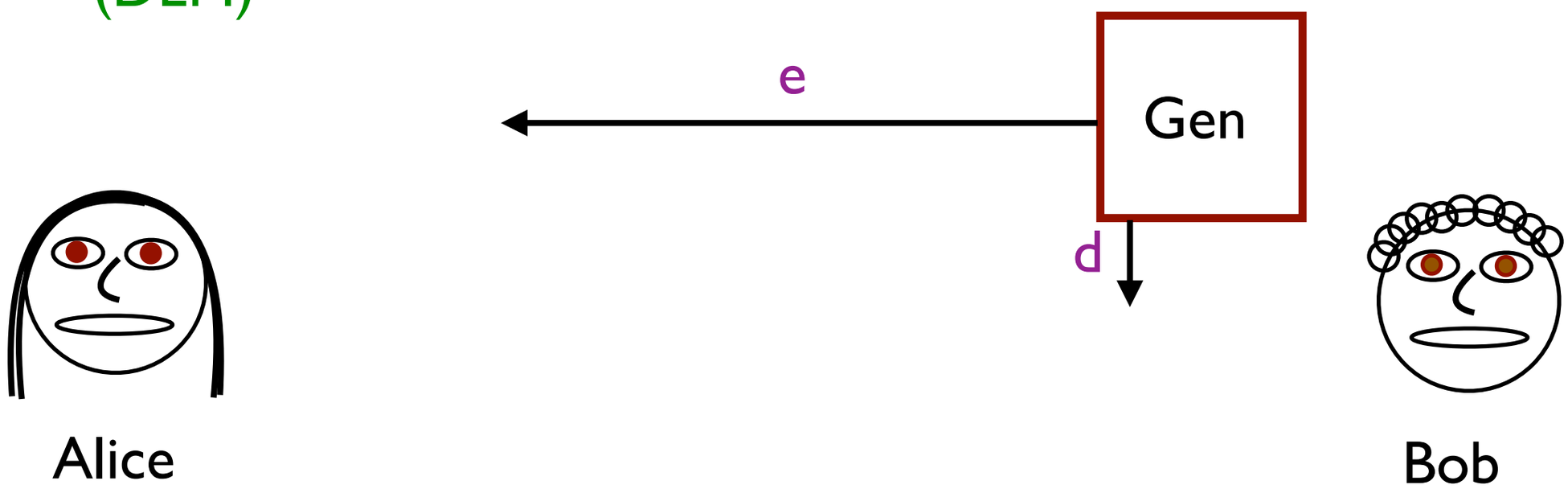
Bob

This class is being recorded

KEM/DEM

Recall that a KEM creates and sends a random key string to someone whose public key you have.

That key then becomes the key for a private-key encryption system, which is here called a data-encapsulation mechanism (DEM)

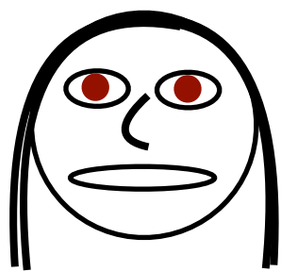


This class is being recorded

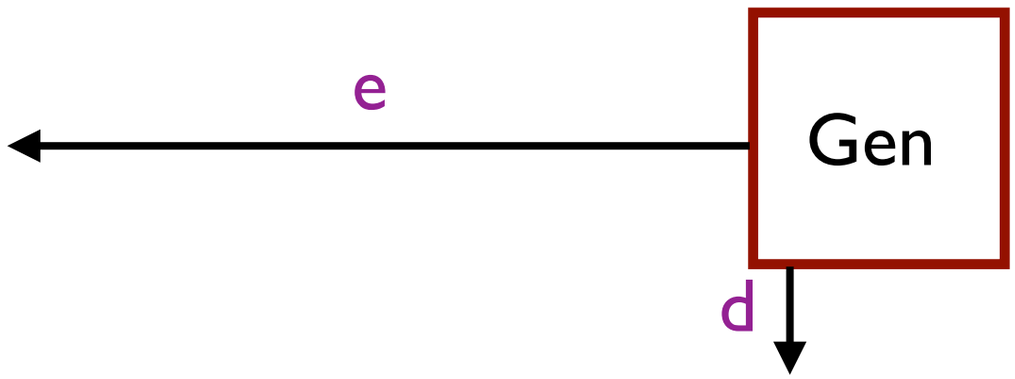
KEM/DEM

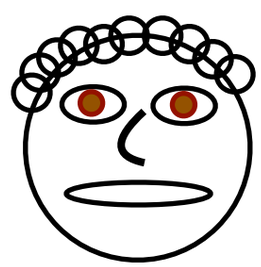
Recall that a KEM creates and sends a random key string to someone whose public key you have.

That key then becomes the key for a private-key encryption system, which is here called a data-encapsulation mechanism (DEM)



Alice





Bob

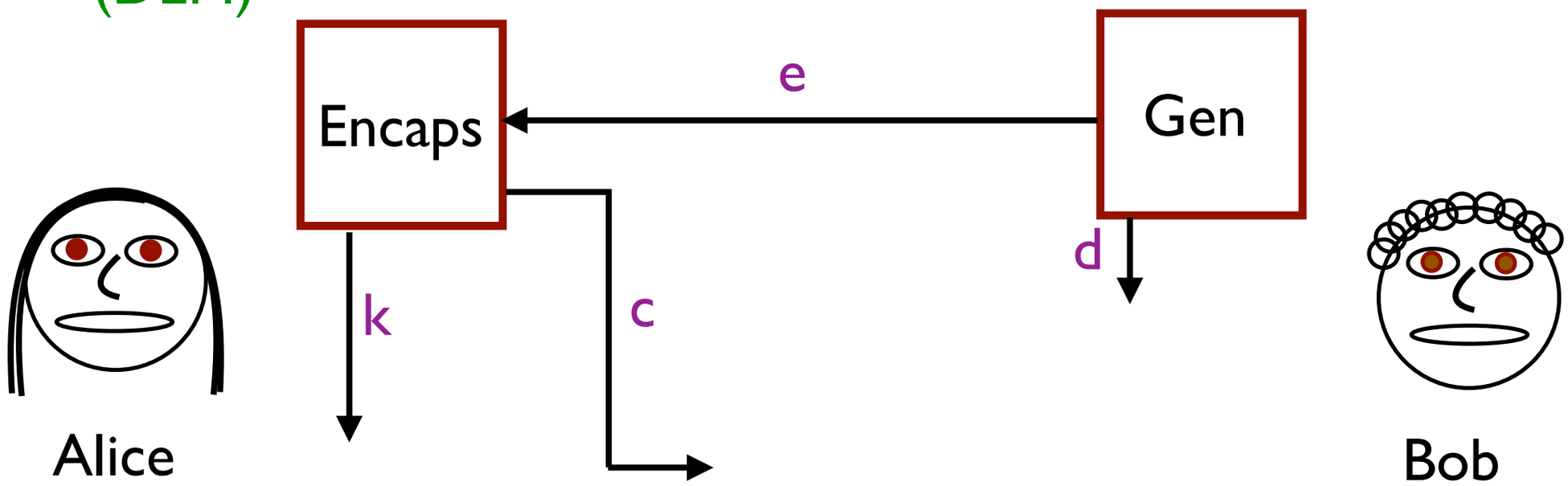
message m

This class is being recorded

KEM/DEM

Recall that a KEM creates and sends a random key string to someone whose public key you have.

That key then becomes the key for a private-key encryption system, which is here called a data-encapsulation mechanism (DEM)



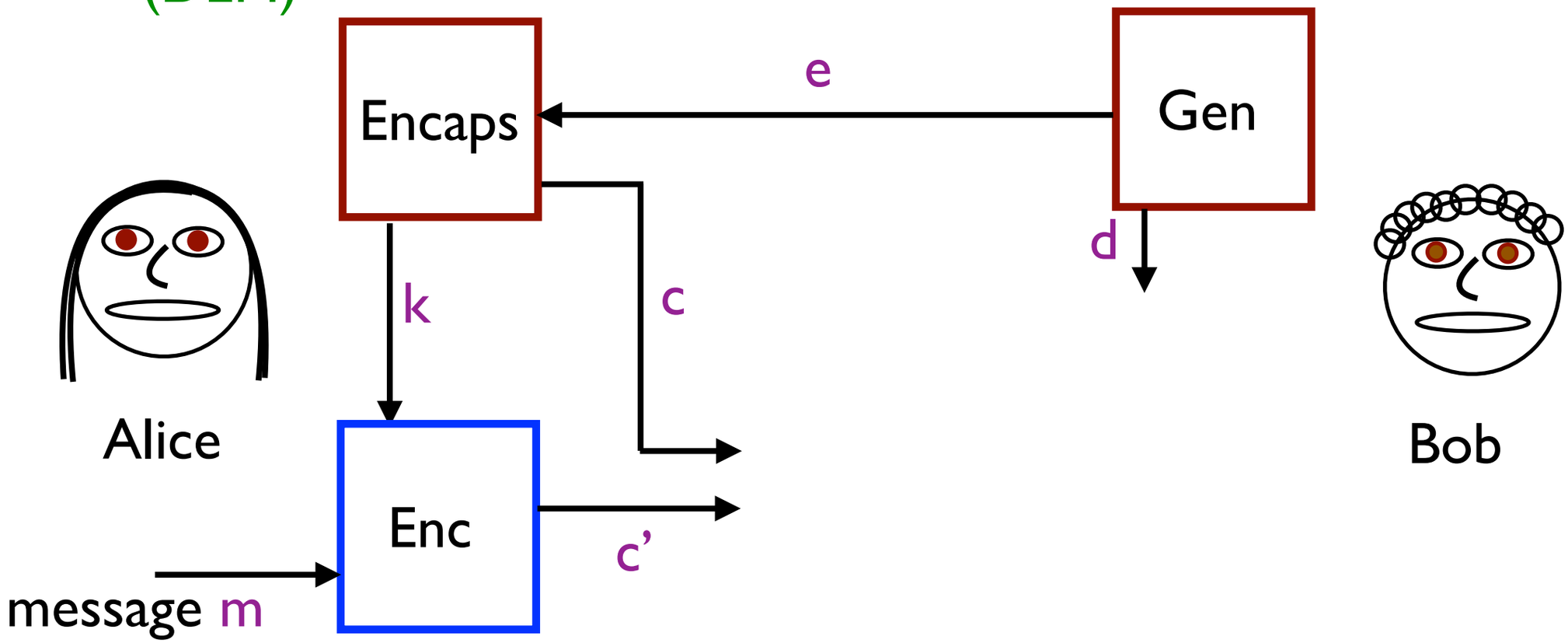
message m

This class is being recorded

KEM/DEM

Recall that a KEM creates and sends a random key string to someone whose public key you have.

That key then becomes the key for a private-key encryption system, which is here called a data-encapsulation mechanism (DEM)

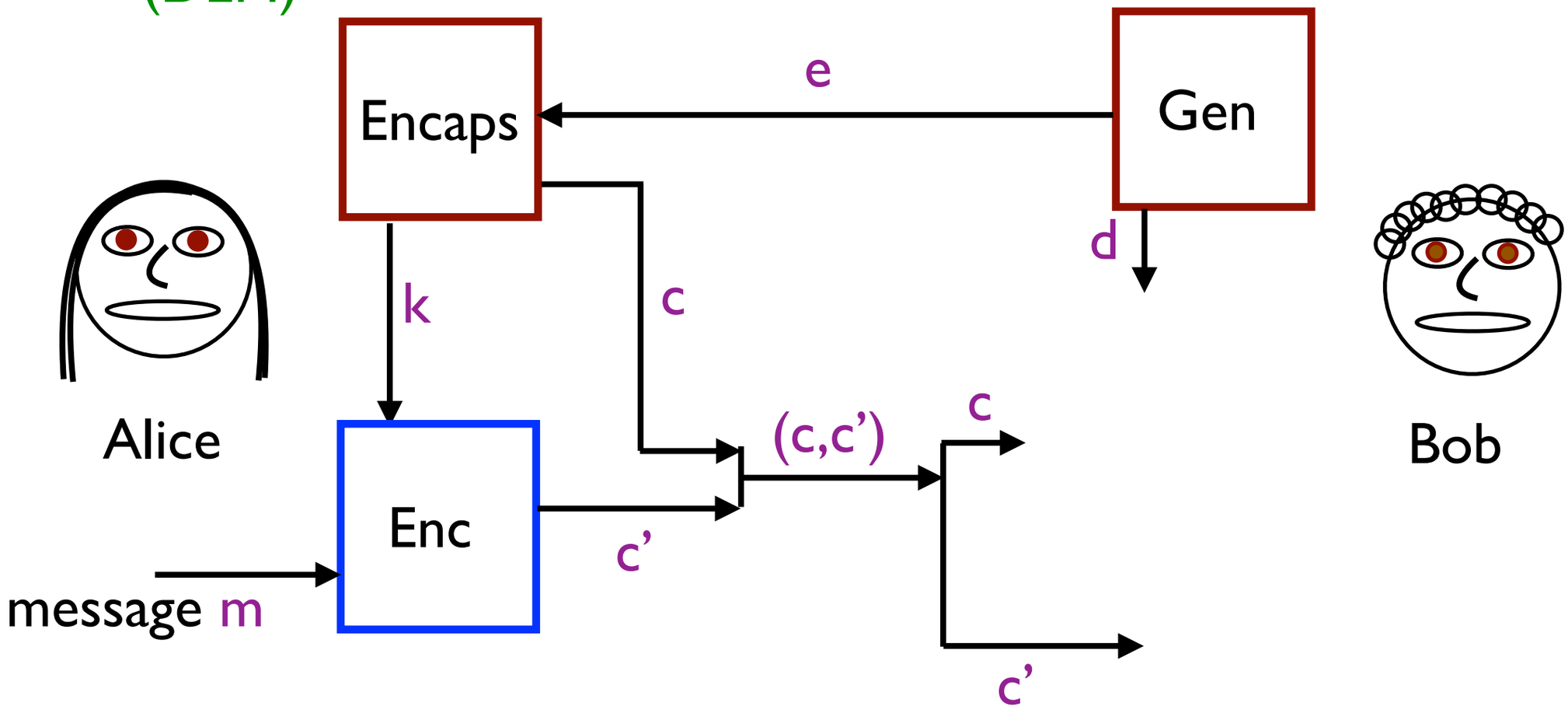


This class is being recorded

KEM/DEM

Recall that a KEM creates and sends a random key string to someone whose public key you have.

That key then becomes the key for a private-key encryption system, which is here called a data-encapsulation mechanism (DEM)

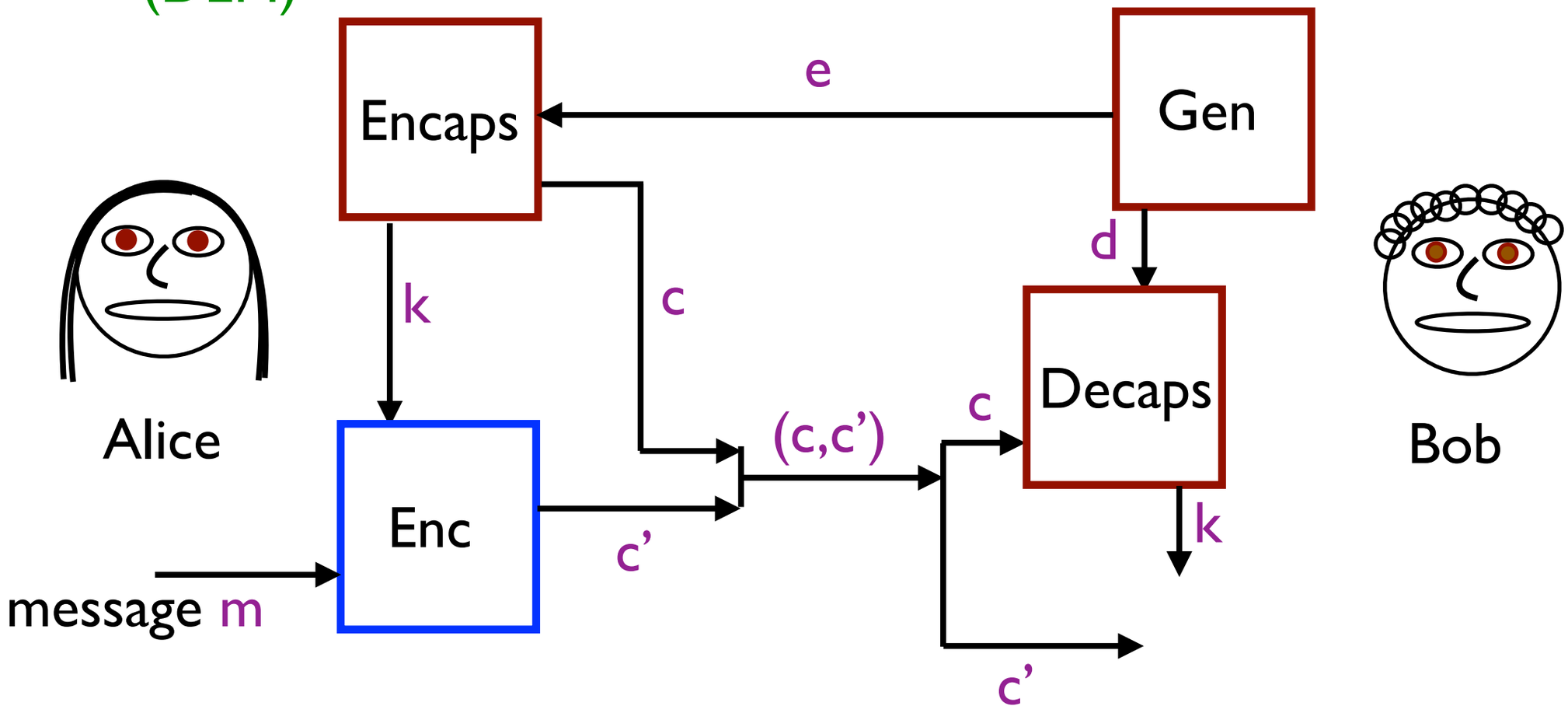


This class is being recorded

KEM/DEM

Recall that a KEM creates and sends a random key string to someone whose public key you have.

That key then becomes the key for a private-key encryption system, which is here called a data-encapsulation mechanism (DEM)

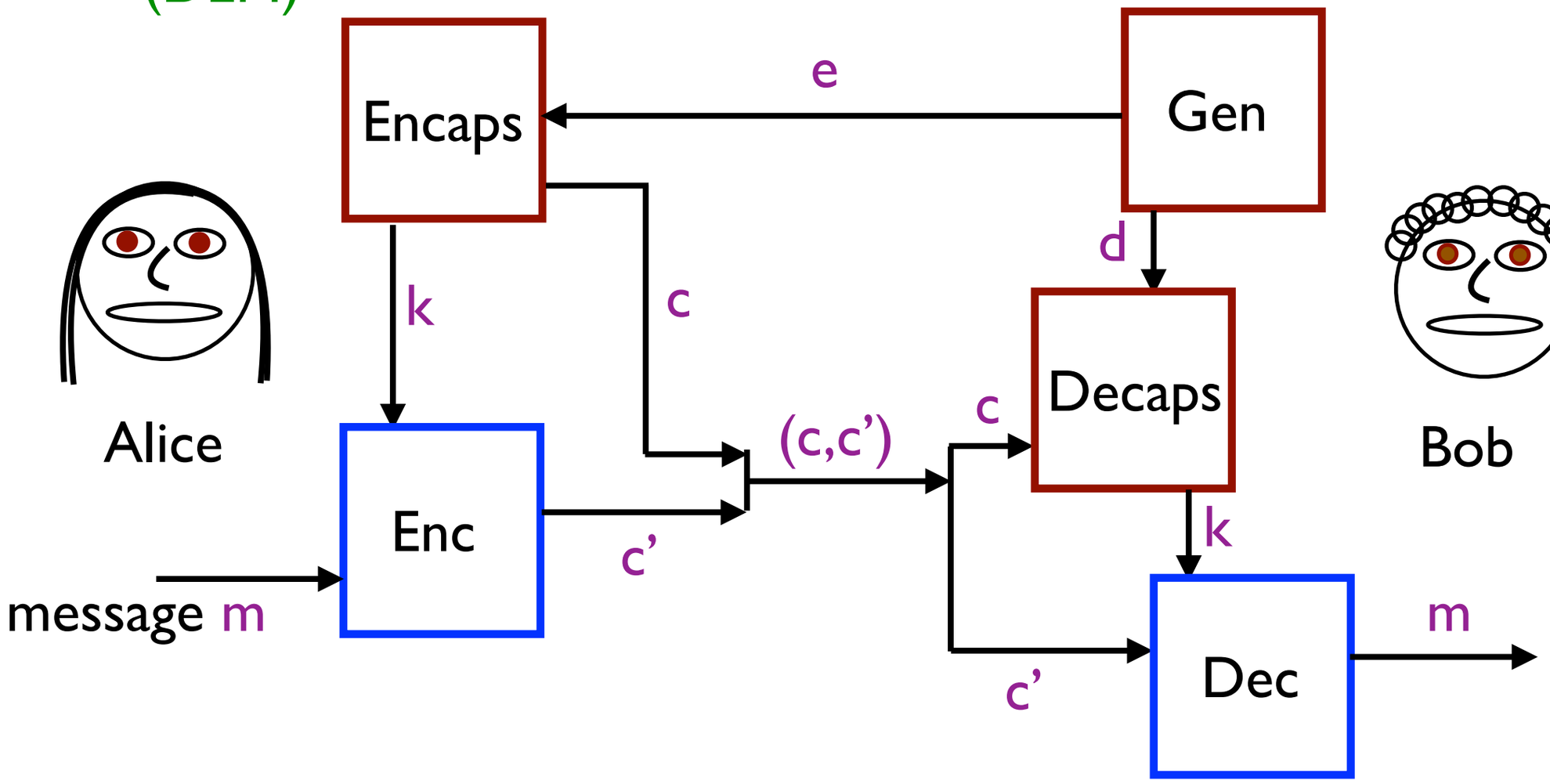


This class is being recorded

KEM/DEM

Recall that a KEM creates and sends a random key string to someone whose public key you have.

That key then becomes the key for a private-key encryption system, which is here called a data-encapsulation mechanism (DEM)



This class is being recorded

RSA-based KEM

Gen: Pick a (public) key derivation function H(x), then as usual for RSA, i.e., generate two random primes p and q which are s bits long. Let N = pq. Choose $e, d \in \mathbb{Z}^*_{\varphi(N)}$ s.t. $ed = 1 \bmod \varphi(N)$.

The public key is (N, e) and the private key is (N, d).

Encaps: Choose random x. The ciphertext is $c = x^e \bmod N$ and the key is H(x).

Decaps: Given c and d, compute $x' = c^d \bmod N$. Then the key is H(x').

As with Diffie-Hellman-based KEM/DEM, this allows us to combine the strengths of public key and private key cryptography.

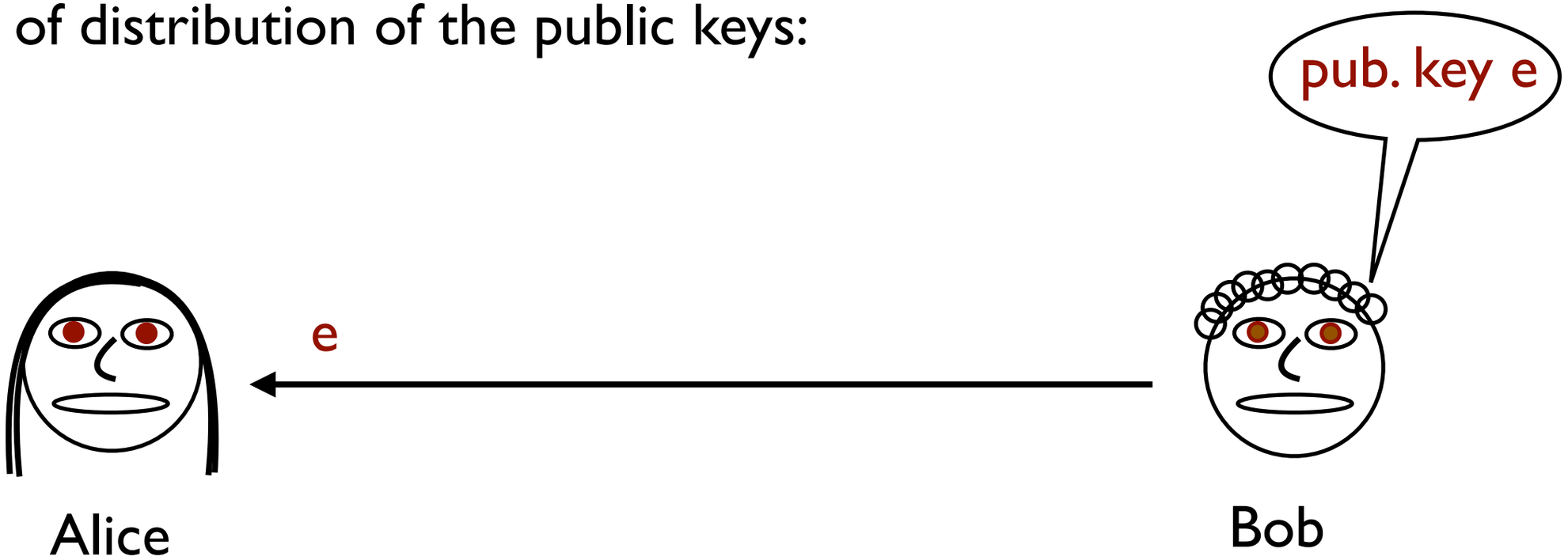
This KEM is secure given RSA assumption and an assumption on the key derivation function H(x).

This class is being recorded

How Do We Distribute Public Keys?

Recall that Diffie-Hellman had a man-in-the-middle attack where Eve pretended to be Bob when talking to Alice and vice-versa.

Public key systems have the same sort of problem at the time of distribution of the public keys:

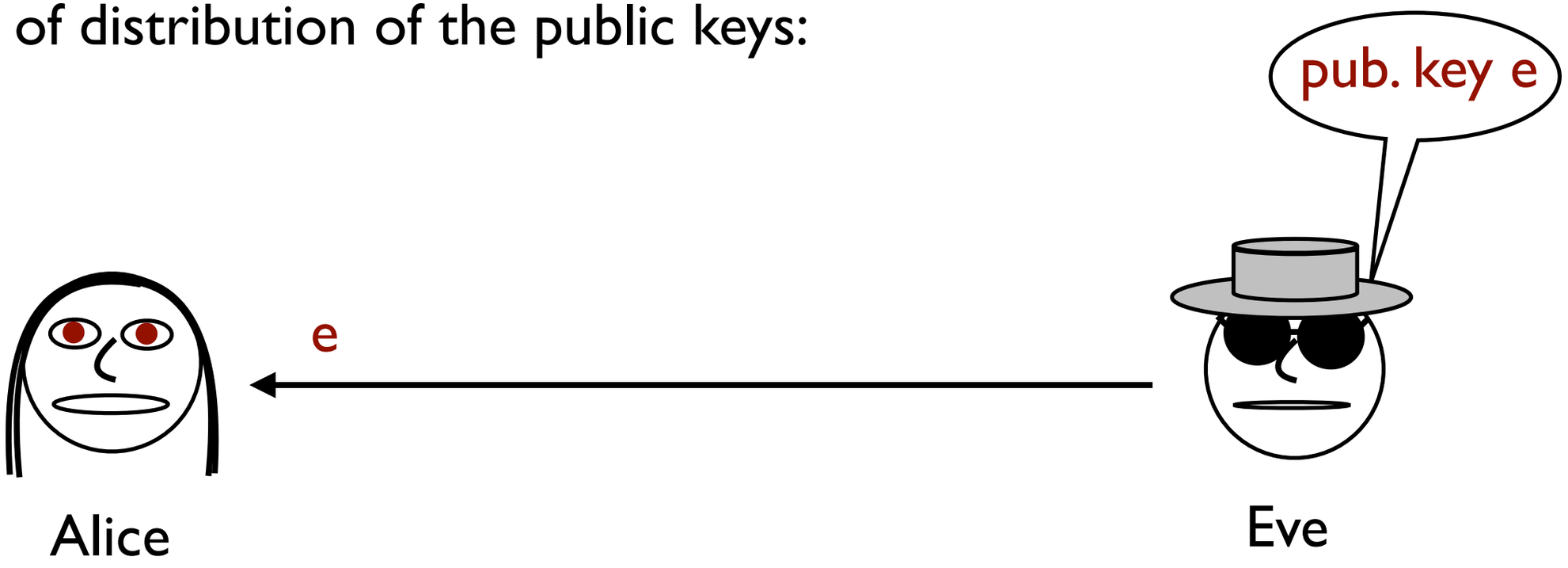


This class is being recorded

How Do We Distribute Public Keys?

Recall that Diffie-Hellman had a man-in-the-middle attack where Eve pretended to be Bob when talking to Alice and vice-versa.

Public key systems have the same sort of problem at the time of distribution of the public keys:



How does Alice know Bob's public key really comes from Bob and not from Eve? Bob needs some way to authenticate it.

This class is being recorded

Chosen Ciphertext Attack on RSA

Plain RSA and Padded RSA are vulnerable to another kind of attack with a stronger threat model, a *chosen ciphertext attack*:

Suppose Alice sends Bob the ciphertext $c = \tilde{m}^e \bmod N$.

Eve can easily create $c' = 2^e c = (2\tilde{m})^e \bmod N$.

Why is that a problem? What if Eve sends c' to Bob, who decrypts it and then acts on the message assuming it to be from Alice. By observing Bob, Eve may be able to deduce the plaintext corresponding to c', namely $2\tilde{m}$. This tells her $\tilde{m}$.

Authenticating messages will help us deal with this class of attacks as well, by removing Eve's ability to change the message.

This class is being recorded

To Do List

- Message authentication and digital signatures to ensure validity of public keys.
- Hash functions to serve as key derivation functions.
- Chosen-ciphertext attacks (CCA) and CCA-security.

Message authentication, digital signatures, and hash functions all have other applications beyond their usefulness for public key cryptography.

This class is being recorded