CMSC/Math 456: Cryptography (Fall 2022)

Lecture 27

Daniel Gottesman

Administrative

Problem set #9 due Thursday, Dec. 8 at midnight.

Final exam: Monday, Dec. 19, 1:30-3:30 PM

- Will be open book again (textbook, lecture notes)
- Students taking the final at ADS: Remember to book with them soon.
- Today and Thursday (last lecture): Review for final
- Topics covered: Everything up to (and including) post-quantum cryptography

Course evaluations are now available to fill out.

The last 15 minutes of class on Thursday will be reserved for course evaluations.

A list of topics covered in the course is available on the course website.

Review Plan

- Principles and basic tools (Kerckhoff's principle, computational complexity, proof by reduction)
- Cryptographic primitives (Pseudorandom generators, pseudorandom functions, hash functions)
- Cryptographic protocols (Private and public-key encryption, key agreement, KEM, MAC, authenticated encryption, digital signature, identification protocol)
- Modular arithmetic — Thursday

Principles and Basic Tools

- Kerckhoff’s principle
- Computational complexity
- Proof by reduction

This class is being recorded

Kerckhoff's Principle

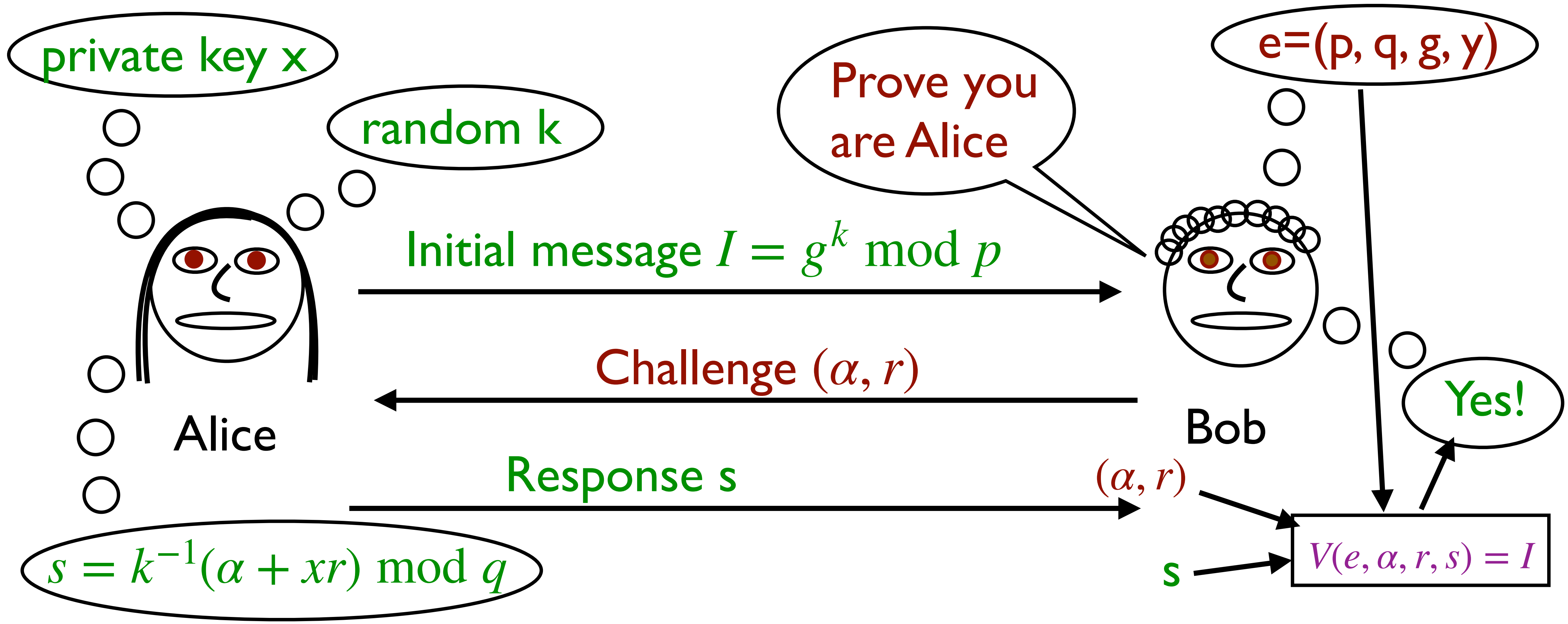
Assume the protocol is known by the adversary. Only the key is secret.

This means that anything that is not explicitly listed as part of the private key (or otherwise is secret) is known to Eve:

- Any parameters of the protocol (e.g., prime q or base g) are known to Eve.
- Any functions involved (e.g., hash function H(x)) are known to Eve.
- Public keys are certainly known to Eve.
- Private keys are not known by Eve.
- Random values picked by a participant and not explicitly announced are not known by Eve.

Example: Identification Protocol

The protocol involves the following values: p, q, g, x, y, k, l, α, r, s.
Which are known to Eve and which are not?



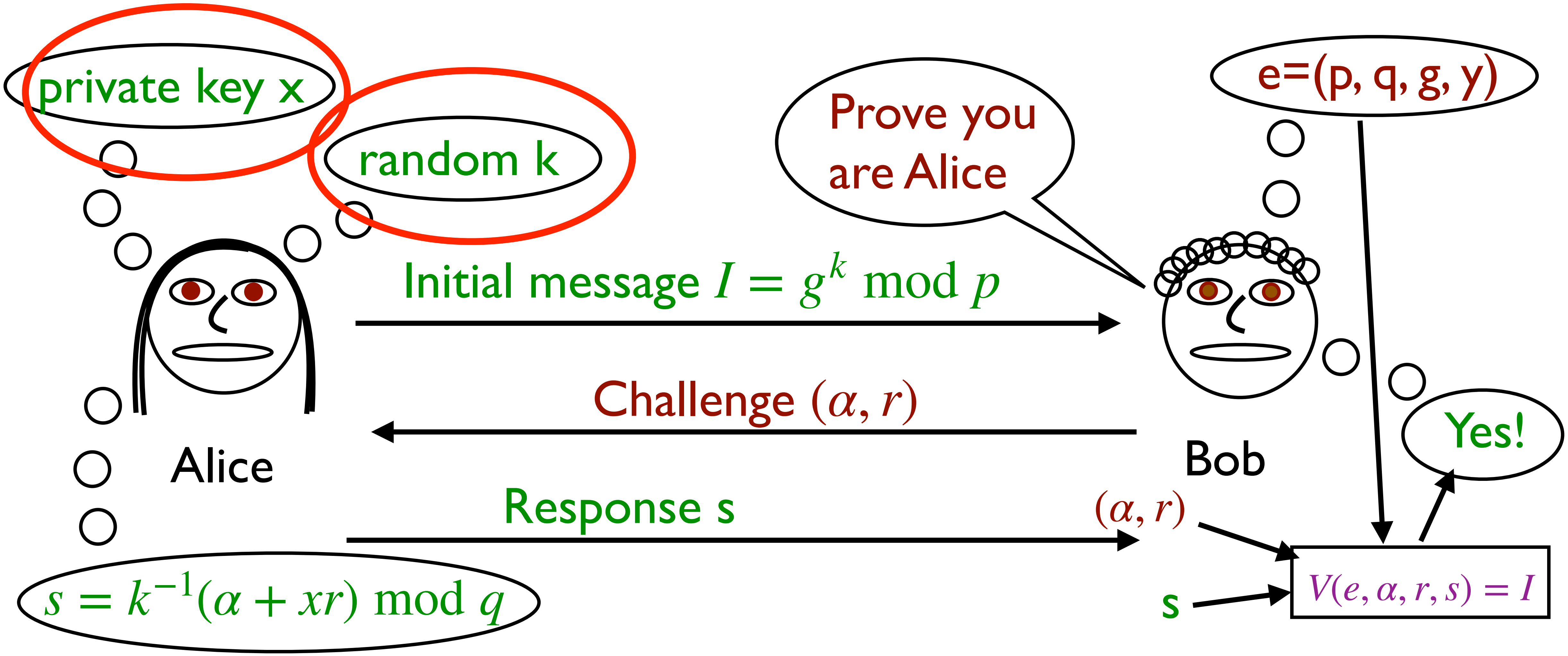
$$V(e, \alpha, r, s) = g^{\alpha s^{-1}} y^{rs^{-1}}$$

This class is being recorded

Example: Identification Protocol

The protocol involves the following values: p, q, g, x, y, k, l, α, r, s.
Which are known to Eve and which are not?

Secret: x, k



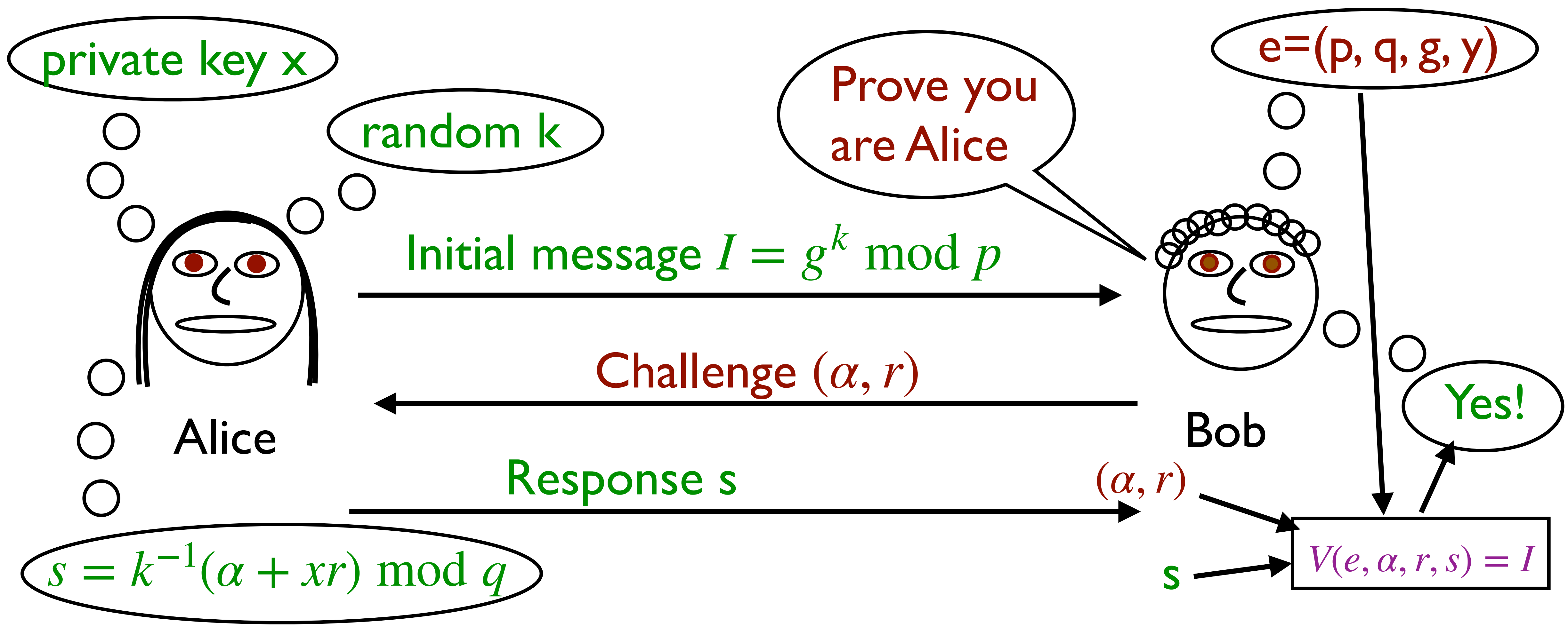
$$V(e, \alpha, r, s) = g^{\alpha s^{-1}} y^{r s^{-1}}$$

This class is being recorded

Example: Identification Protocol

The protocol involves the following values: p, q, g, x, y, k, l, α, r, s.
Which are known to Eve and which are not?

Secret: x, k



$$V(e, \alpha, r, s) = g^{\alpha s^{-1}} y^{r s^{-1}}$$

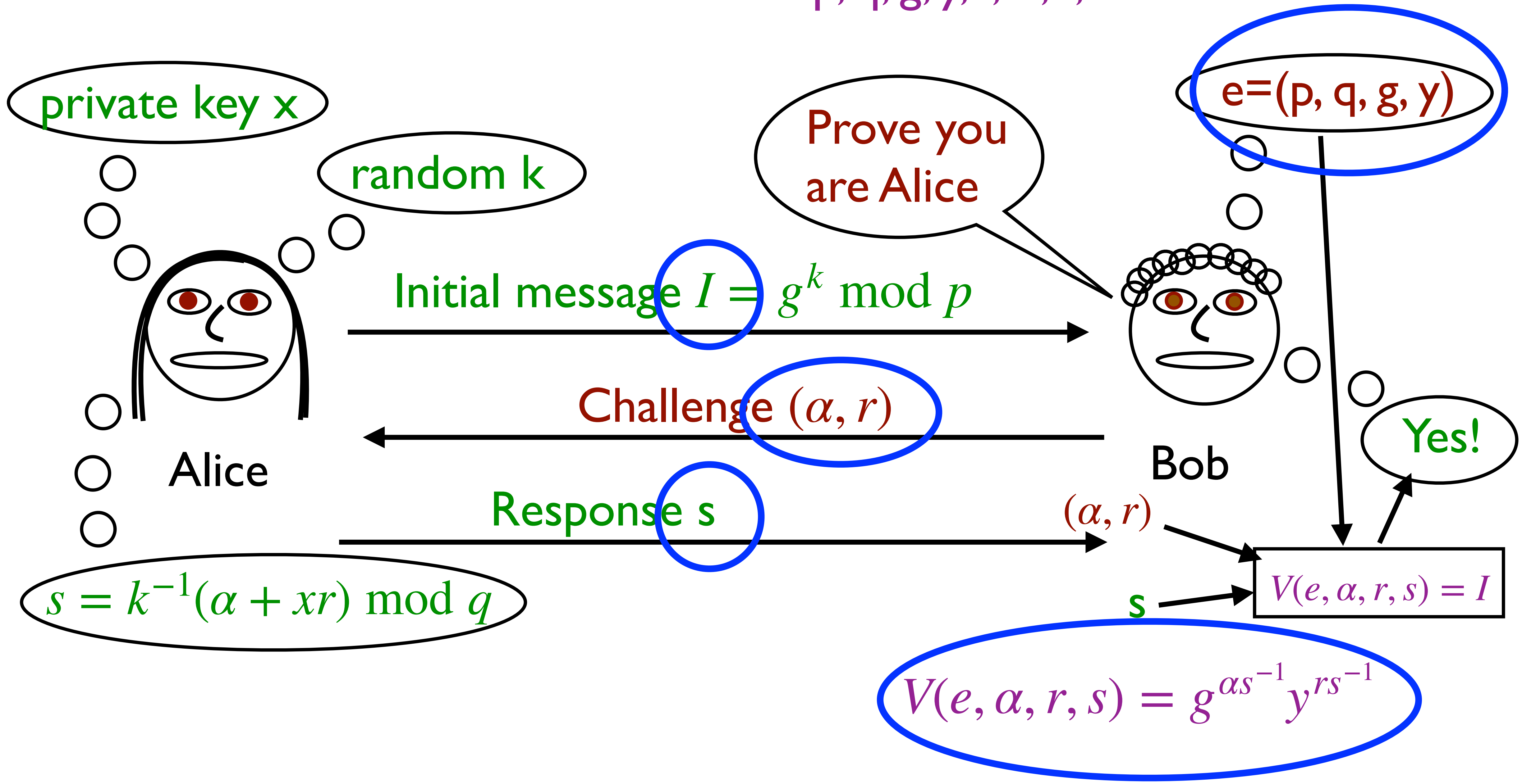
This class is being recorded

Example: Identification Protocol

The protocol involves the following values: p, q, g, x, y, k, l, α, r, s.
Which are known to Eve and which are not?

Secret: x, k

Public: p, q, g, y, l, α, r, s. Also V.



This class is being recorded

Efficient and Negligible

Almost always, we are interested in efficient algorithms, namely ones which run in a time polynomial as a function of the size of the input to the function.

E.g.: Can G(s) be a pseudorandom generator if $|G(s)| = f(s)$?

Efficient and Negligible

Almost always, we are interested in efficient algorithms, namely ones which run in a time polynomial as a function of the size of the input to the function.

E.g.: Can G(s) be a pseudorandom generator if $|G(s)| = f(s)$?

Answer: It can be if f(s) is polynomial and it cannot if f(s) is exponential c^s. Why? The brute force attack $\mathscr{A}$ takes as input a string of size $|G(s)|$ and tries all values of s. This runs in time 2^s, which is $(c^s)^{\log c}$. This is a polynomial as a function of the size of the input to $\mathscr{A}$.

Efficient and Negligible

Almost always, we are interested in efficient algorithms, namely ones which run in a time polynomial as a function of the size of the input to the function.

E.g.: Can G(s) be a pseudorandom generator if $|G(s)| = f(s)$?

Answer: It can be if f(s) is polynomial and it cannot if f(s) is exponential c^s. Why? The brute force attack $\mathscr{A}$ takes as input a string of size $|G(s)|$ and tries all values of s. This runs in time 2^s, which is $(c^s)^{\log c}$. This is a polynomial as a function of the size of the input to $\mathscr{A}$.

A function is negligible if it goes to 0 faster than *any* polynomial. Specifically, $\lim_{x \to \infty} f(x)p(x) = 0$ for all polynomials p(x).

E.g.: $f(x) = 1/(100x^3)$ not negligible

$f(x) = \exp(-\sqrt{x})$ negligible

Reductions

Imagine you are a robot in a world full of similar robots. Your job is to play one of the cryptographic games we have discussed for defining security of protocols. Say you play the game G.

Unfortunately, you are very bad at your job. You can't figure out how to win consistently, or even much better than random chance.

> But one day you have a bright idea! Your friend has the job to play the game H, and you've realized that G and H are related. While your friend is sleeping, you download a copy of their AI and construct an elaborate simulation.

In the simulation, your copy of the friend thinks they are going to work and need to play H. But you have arranged that the simulation uses specific values to H so that you can use their answers to help you play G.

> Note: The simulation must be identical to the friend's real job or the copy will realize it is a copy.

Reduction Example

For example, suppose your job is to break the RSA assumption: Given N, e, and random y, find x such that $x^e = y \bmod N$.

Your friend plays the factoring game: Given N find a factor of N.

When you are given (N, e, y), you start your simulation and your friend's copy thinks they are going to work. You arrange for them to be given the problem N and they answer p. You take this value out of the simulation and calculate q = N/p, then $\varphi(N)$. Then you use Euclid's algorithm to find $e^{-1} \bmod \varphi(N)$ and compute $y^{\varphi(N)} \bmod N$ and use that for your answer x.

You have reduced breaking RSA to factoring.

Hardness via Reductions

Unfortunately for you, your friend is also bad at their job. The answers they give are not real factors, or maybe they don't answer at all. Either way, your algorithm will fail.

> Disappointed, you conclude that there is no way to do your job.

Is this a valid conclusion?

Hardness via Reductions

Unfortunately for you, your friend is also bad at their job. The answers they give are not real factors, or maybe they don't answer at all. Either way, your algorithm will fail.

> Disappointed, you conclude that there is no way to do your job.

Is this a valid conclusion?

No. There could be a different robot that is better at factoring, or maybe there is a way to beat RSA that doesn't involve virtually kidnapping anyone.

> But: If your friend finds out about your plan, he might be able to conclude that his job — factoring — is hopeless. If RSA is unbreakable, then factoring must be hard as well.

Cryptographic Primitives

- Pseudorandom generators
- Pseudorandom functions
- Hash functions

Pseudorandomness

Pseudorandom generator G(s)

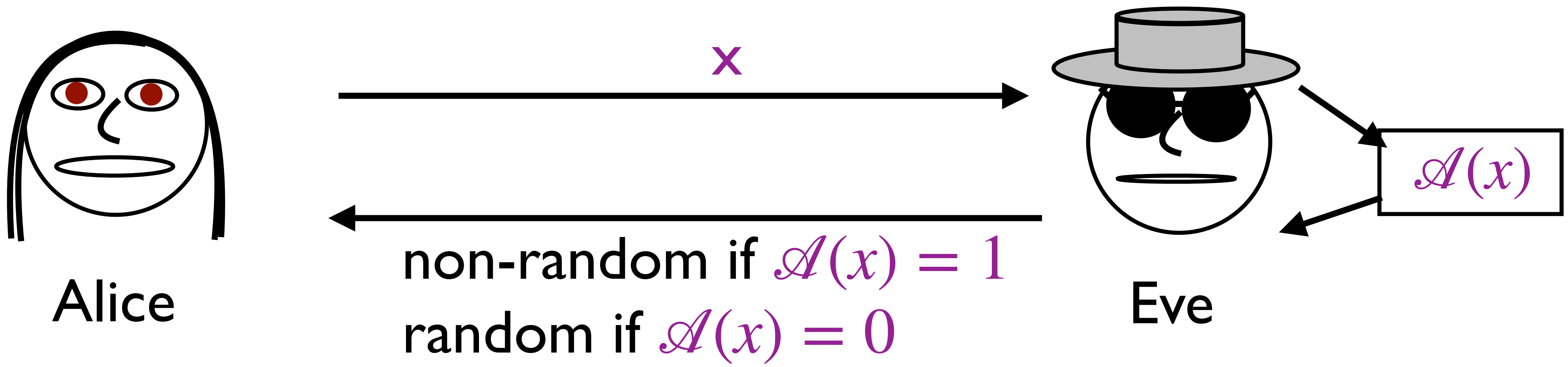
- One input s, "seed"
- Output looks like a random string when s unknown
- Output should be longer than the seed
- Stream cipher is a more flexible version

Pseudorandom function $F_k(r)$

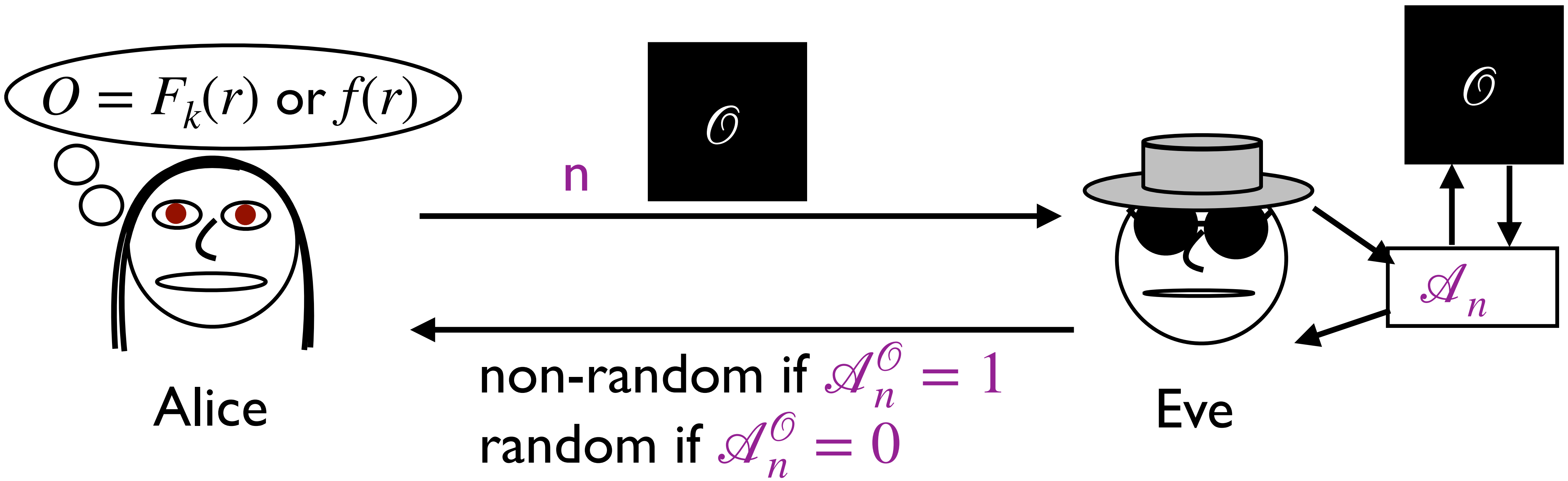
- Two inputs: k (key) and r
- For fixed but unknown k, looks like a random function of r
- Output can be the same size or smaller than the input
- Block cipher is a fixed-size version, but must be a permutation (with computable inverse for known k)

Pseudorandomness Games

Pseudorandom generator:



Pseudorandom function:



This class is being recorded

Hash Functions

Hash function H(x)

- Unlike pseudorandom functions and generators, function and input are both known
- Output must be shorter than the input
- Main cryptographic property is collision resistance, meaning it is hard to find two inputs x_1, x_2 with the same output $H(x_1) = H(x_2)$
- Does not need to look like a random function
- But is often modeled as a random oracle anyway
- Note: but (unlike a pseudorandom function) it is always easy to distinguish from a truly random function since we can just test it on specific inputs
- Often take arbitrary-length inputs

Cryptographic Protocols

- Private-key encryption
- Public-key encryption
- Key agreement
- Key encapsulation mechanism (KEM)
- MAC
- Authenticated encryption
- Digital signature
- Identification protocol

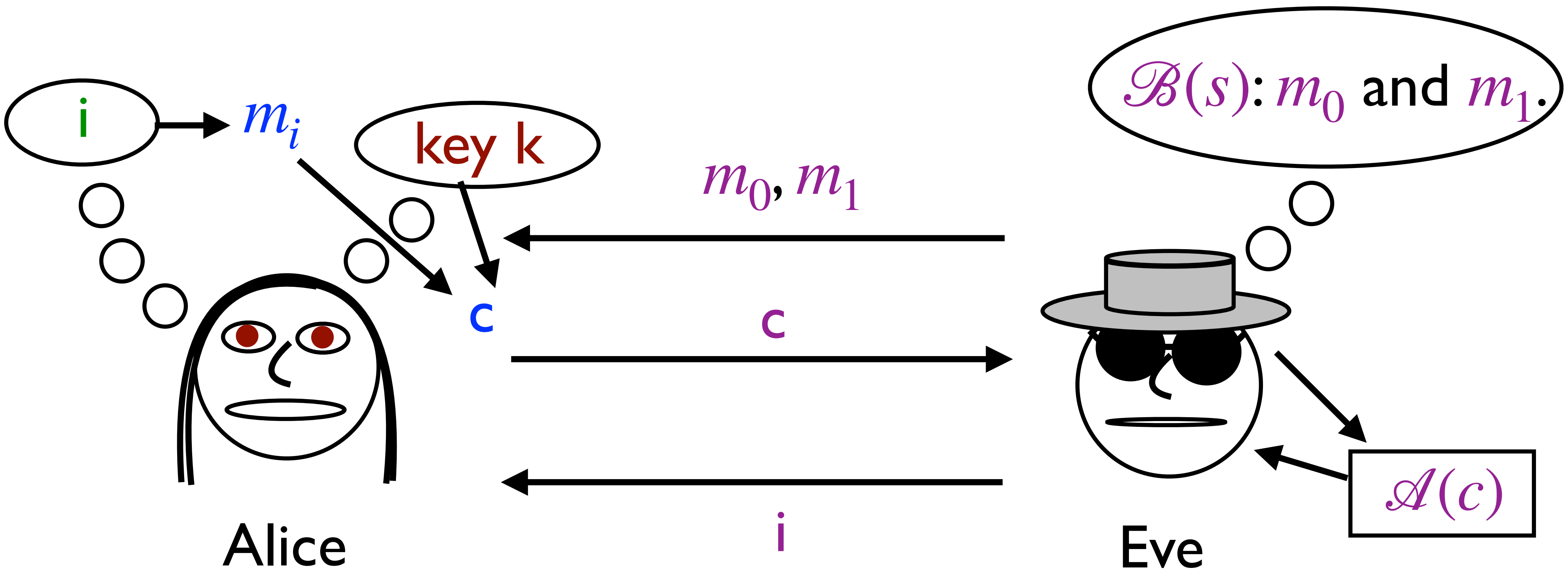
Cryptographic Protocols Comparison

Protocol	Purpose	Pub./Priv.	Interactive?
Private-key encryption	Encryption	Private	No
Public-key encryption	Encryption	Public	No
Key agreement	Gen. key	None	Yes
KEM	Gen. key	Public	No
MAC	Authenticate	Private	No
Authenticated encrypt.	Enc. + Auth.	Private	No
Digital signature	Authenticate	Public	No
Identification protocol	Authenticate	Public	Yes

Encryption Security Definitions

Eve chooses two messages m_0 and m_1 and must identify an encryption of one of them.

EAV security:

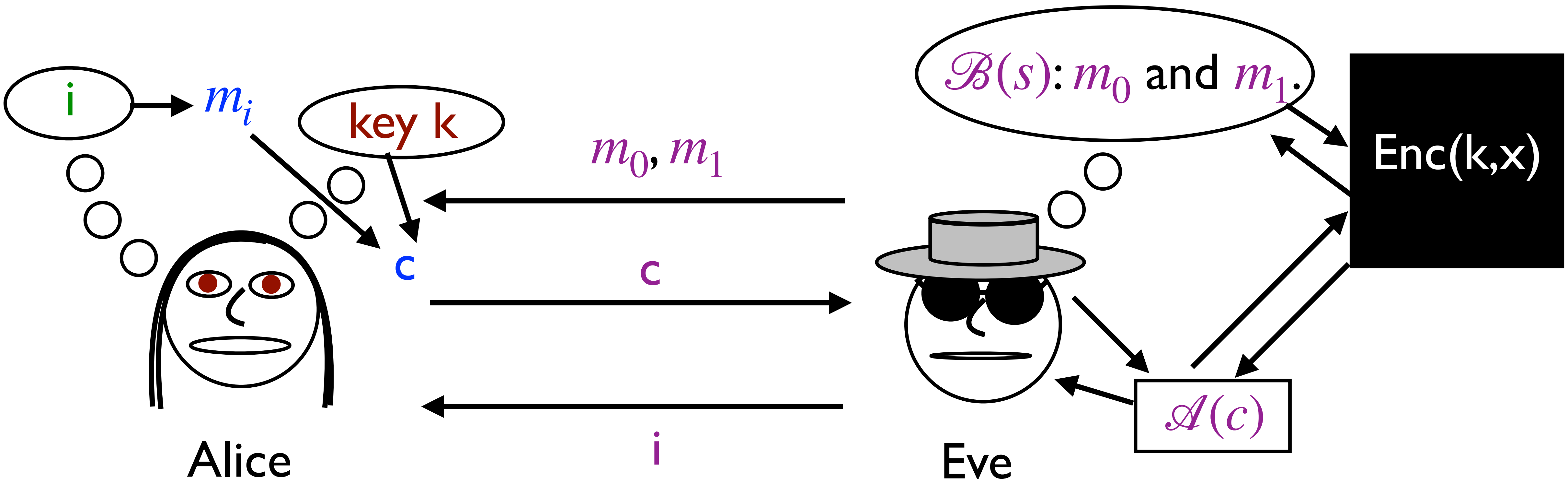


This class is being recorded

Encryption Security Definitions

Eve chooses two messages m_0 and m_1 and must identify an encryption of one of them.

CPA security (private key):

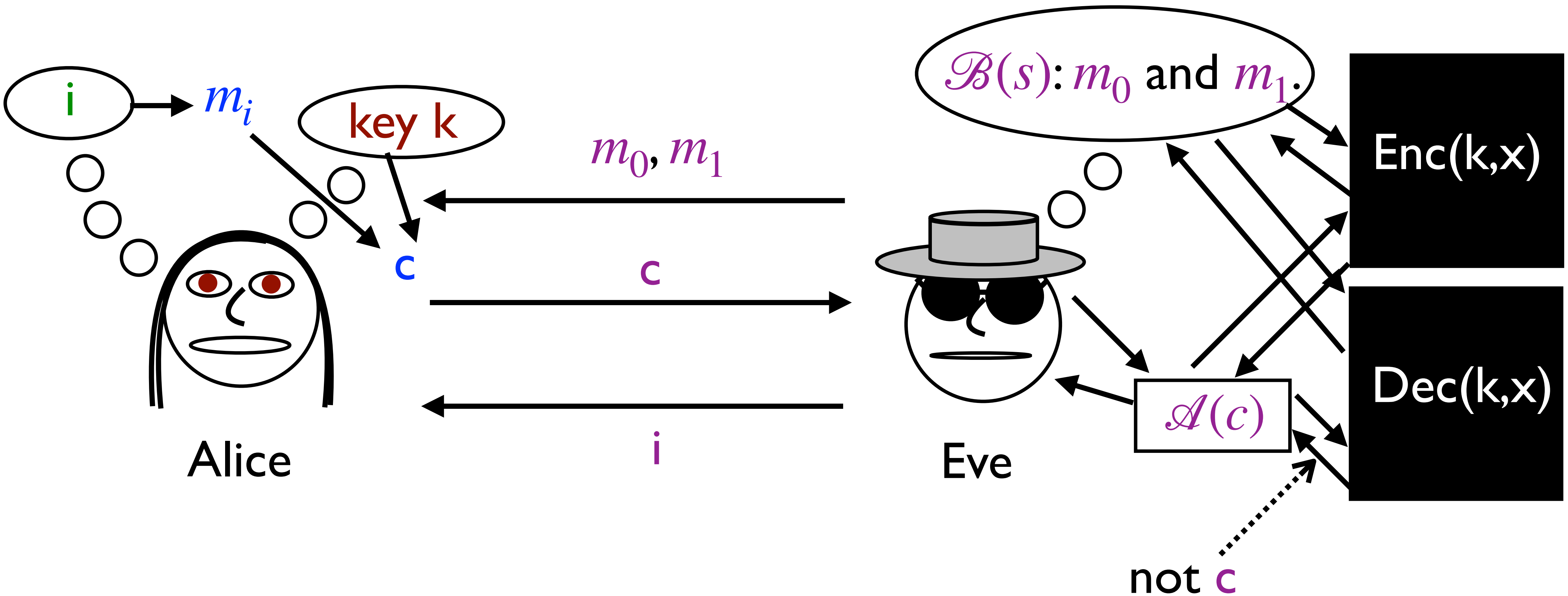


This class is being recorded

Encryption Security Definitions

Eve chooses two messages m_0 and m_1 and must identify an encryption of one of them.

CCA security (private key):

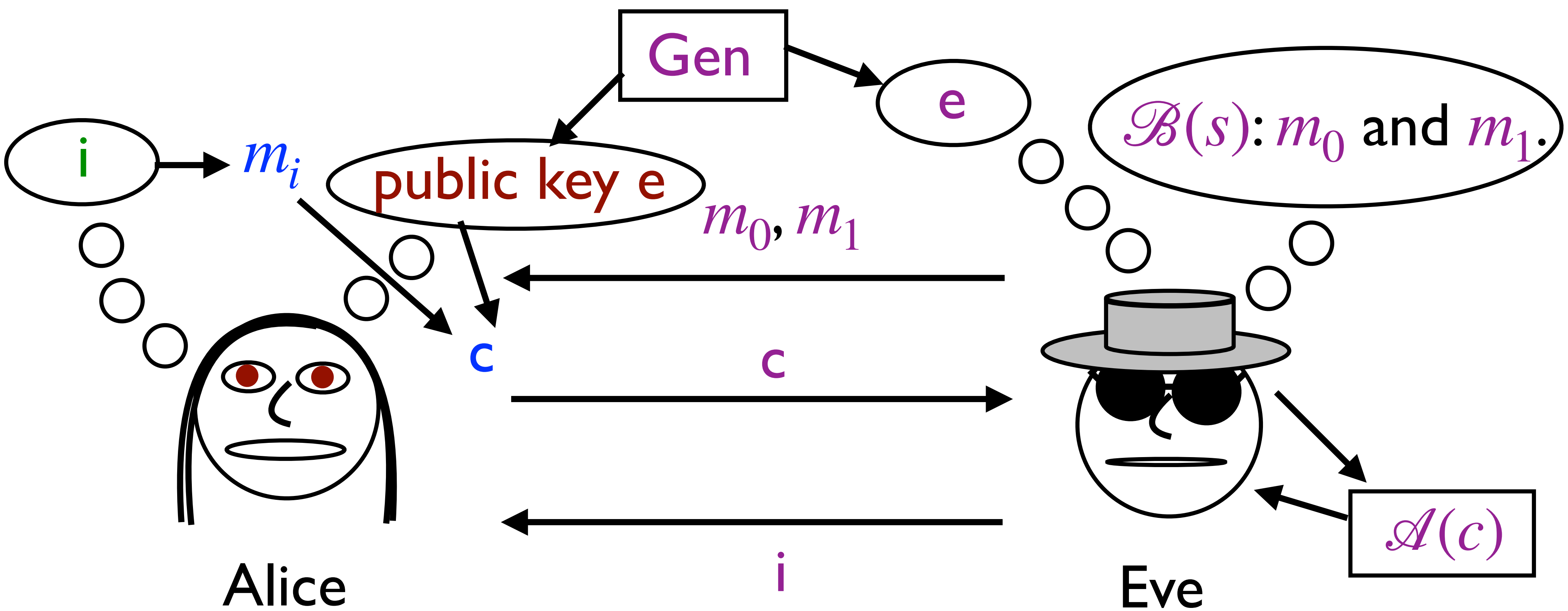


This class is being recorded

Encryption Security Definitions

Eve chooses two messages m_0 and m_1 and must identify an encryption of one of them.

CPA security (public key):

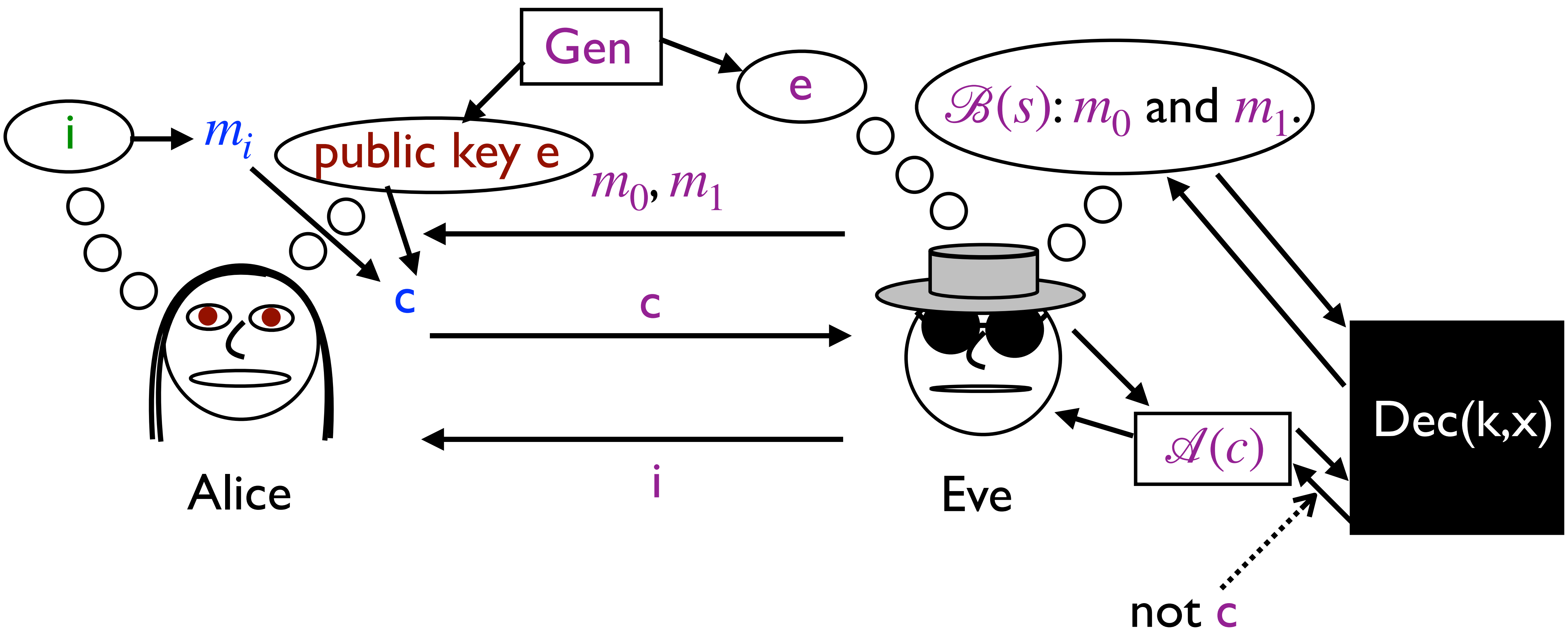


This class is being recorded

Encryption Security Definitions

Eve chooses two messages m_0 and m_1 and must identify an encryption of one of them.

CCA security (public key):

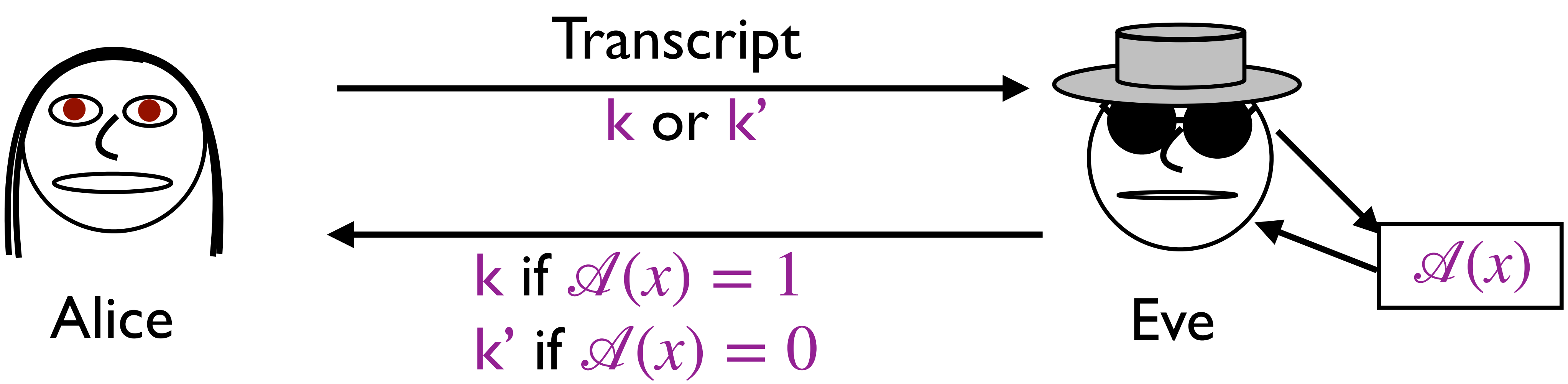


This class is being recorded

Security Definitions for Key Gen.

Eve must distinguish between k generated by the protocol and a uniformly random k'.

Key agreement security:

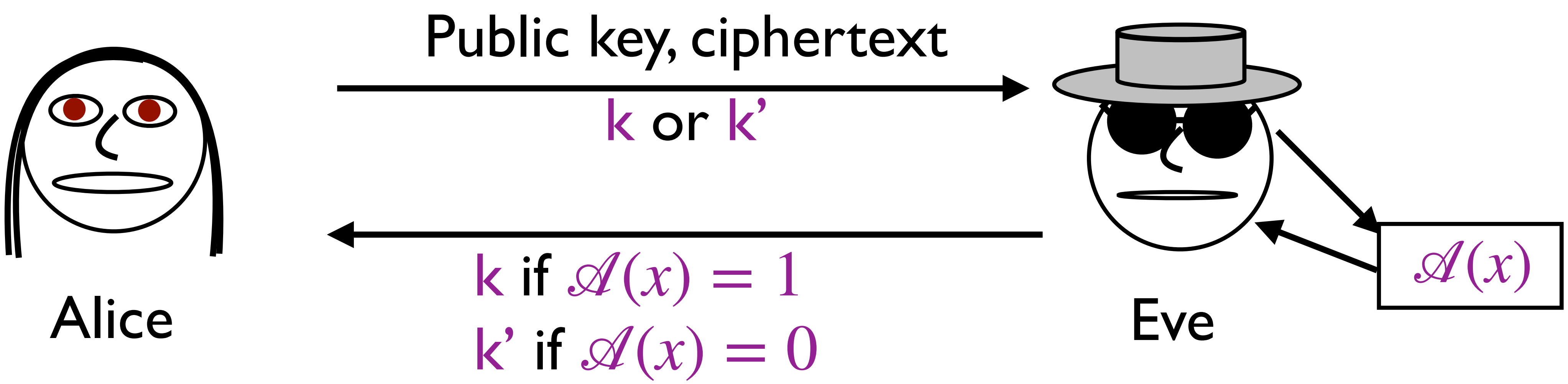


This class is being recorded

Security Definitions for Key Gen.

Eve must distinguish between k generated by the protocol and a uniformly random k'.

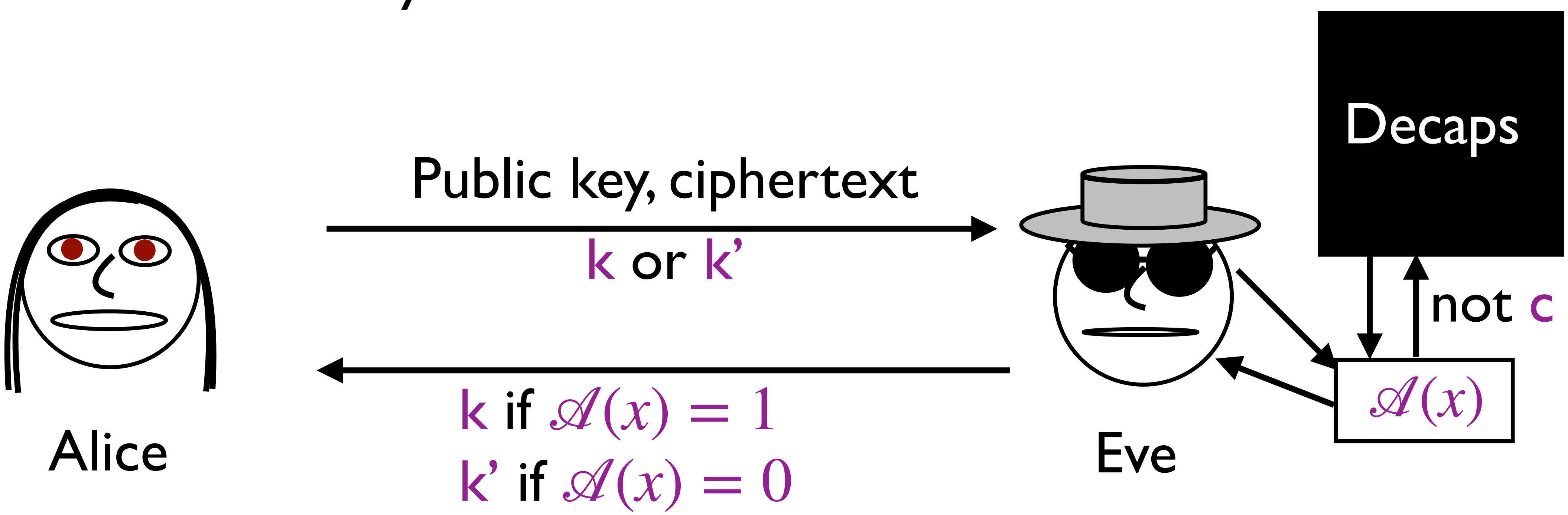
KEM CPA security:



Security Definitions for Key Gen.

Eve must distinguish between k generated by the protocol and a uniformly random k'.

KEM CCA security:

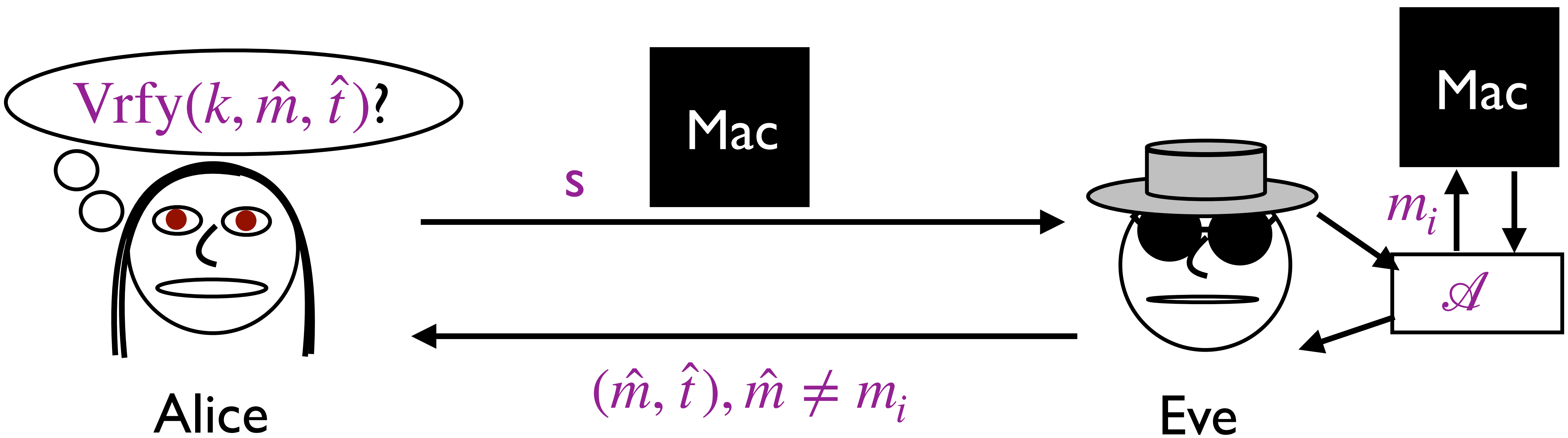


This class is being recorded

Authentication Security Definitions

Eve must forge a message which she hasn't queried to her oracle.

MAC security:

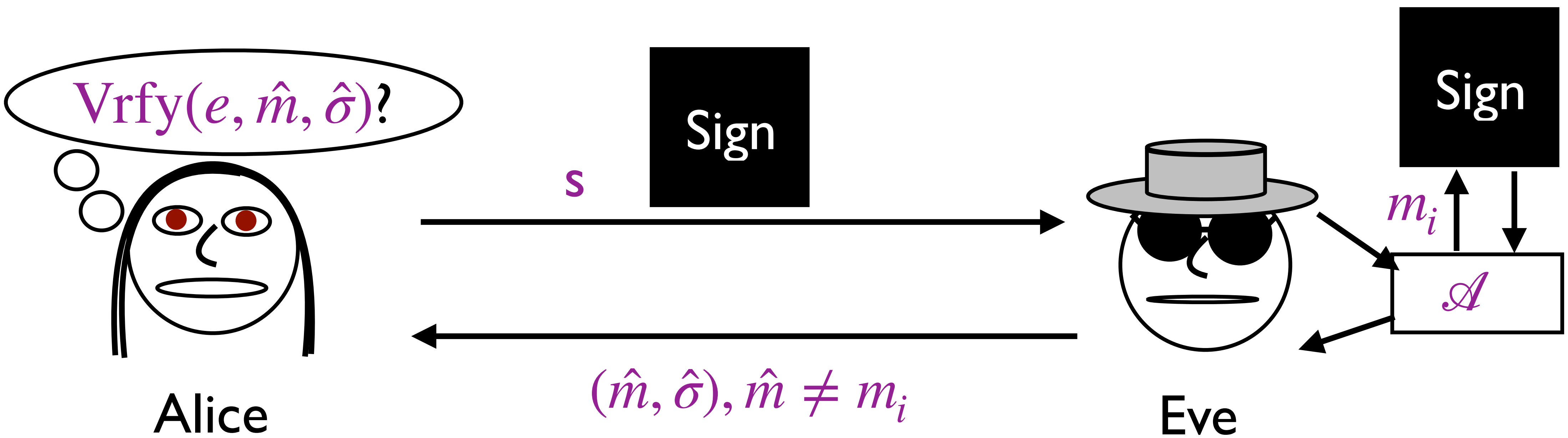


This class is being recorded

Authentication Security Definitions

Eve must forge a message which she hasn't queried to her oracle.

Digital signature security:

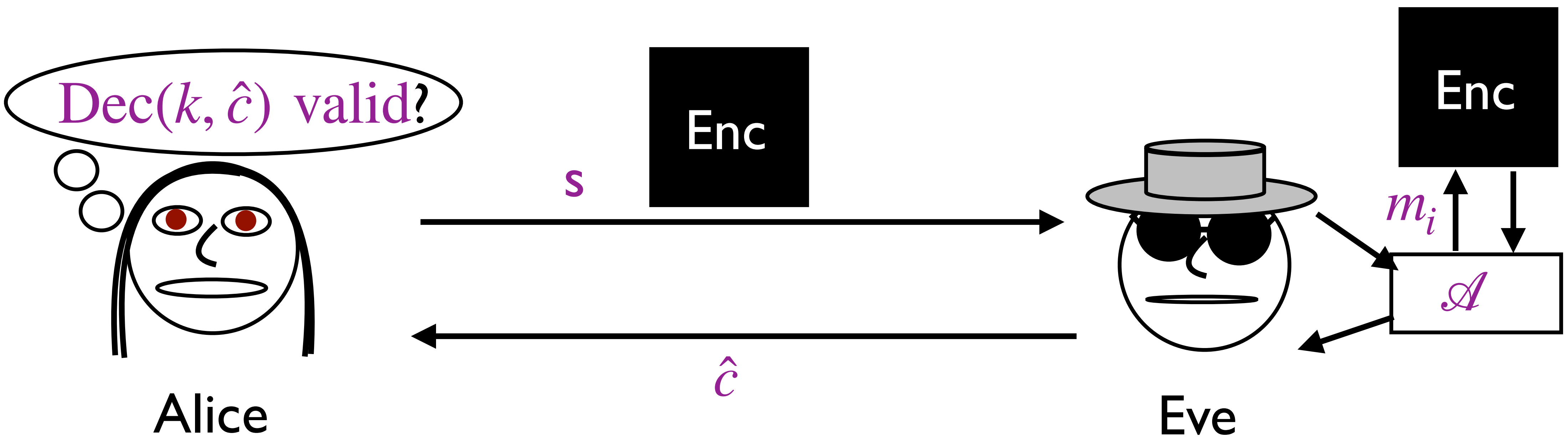


This class is being recorded

Authentication Security Definitions

Eve must forge a message which she hasn't queried to her oracle.

Unforgeability (for encryption):



Recall that authenticated encryption is CCA security plus unforgeability.

General Encryption Constructions

EAV security:

Pseudorandom generator G(s)

Pseudo one-time pad
$c = G(k) \oplus m$

CPA security:

Pseudorandom function $F_k(r)$

$(r, F_k(r) \oplus m)$
Need random IV r to avoid repeating ciphertext

CBC mode or CTR mode for longer messages

CCA security/AE:

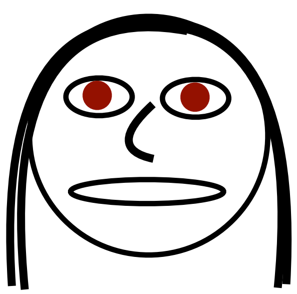
MAC

Encrypt then authenticate

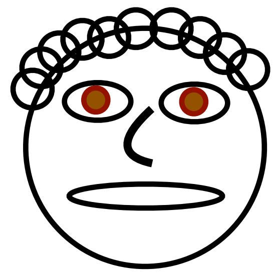
This class is being recorded

KEM/DEM

Using a KEM, we can effectively upgrade these private-key encryption protocols into public-key encryption protocols:



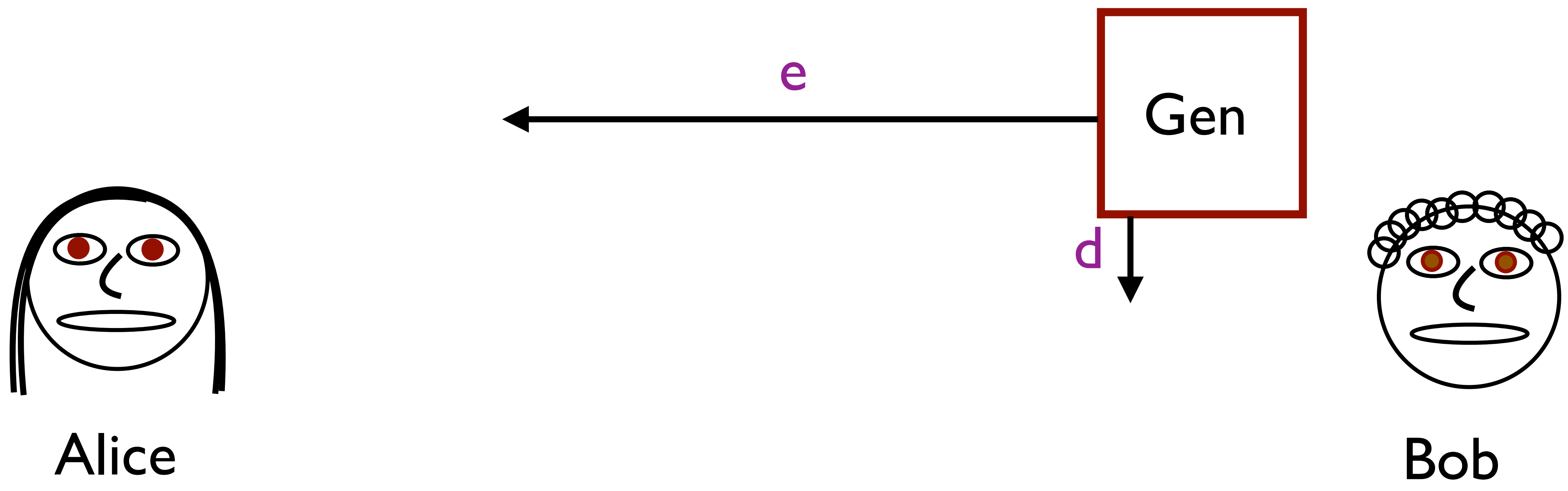
Alice



Bob

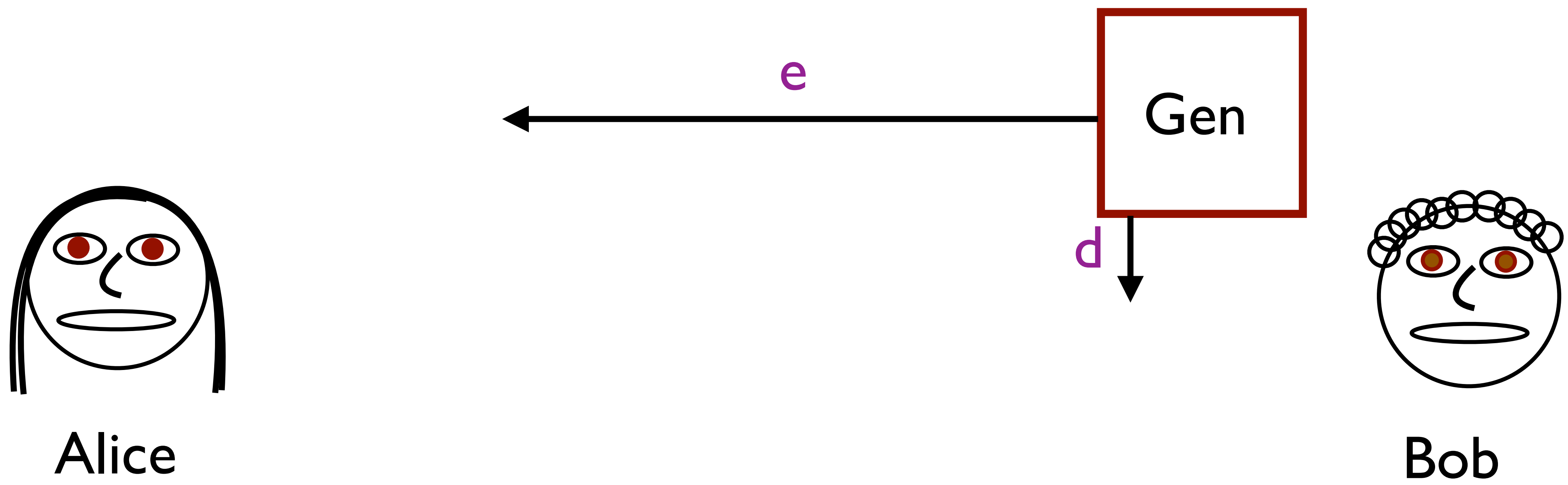
KEM/DEM

Using a KEM, we can effectively upgrade these private-key encryption protocols into public-key encryption protocols:



KEM/DEM

Using a KEM, we can effectively upgrade these private-key encryption protocols into public-key encryption protocols:

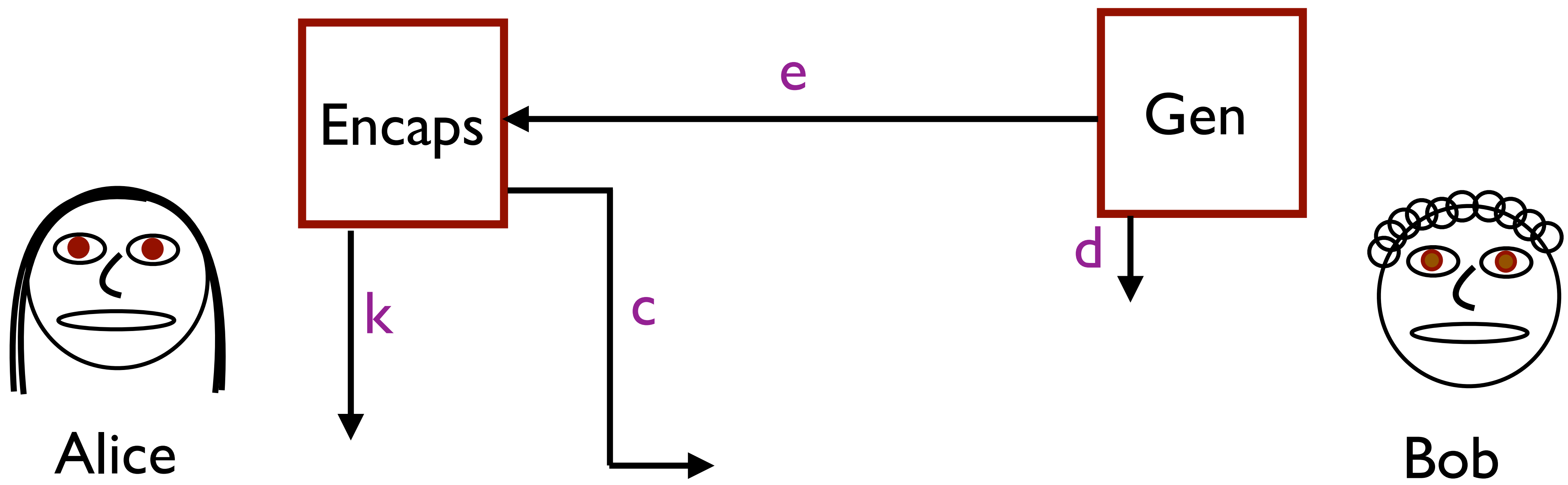


message m

This class is being recorded

KEM/DEM

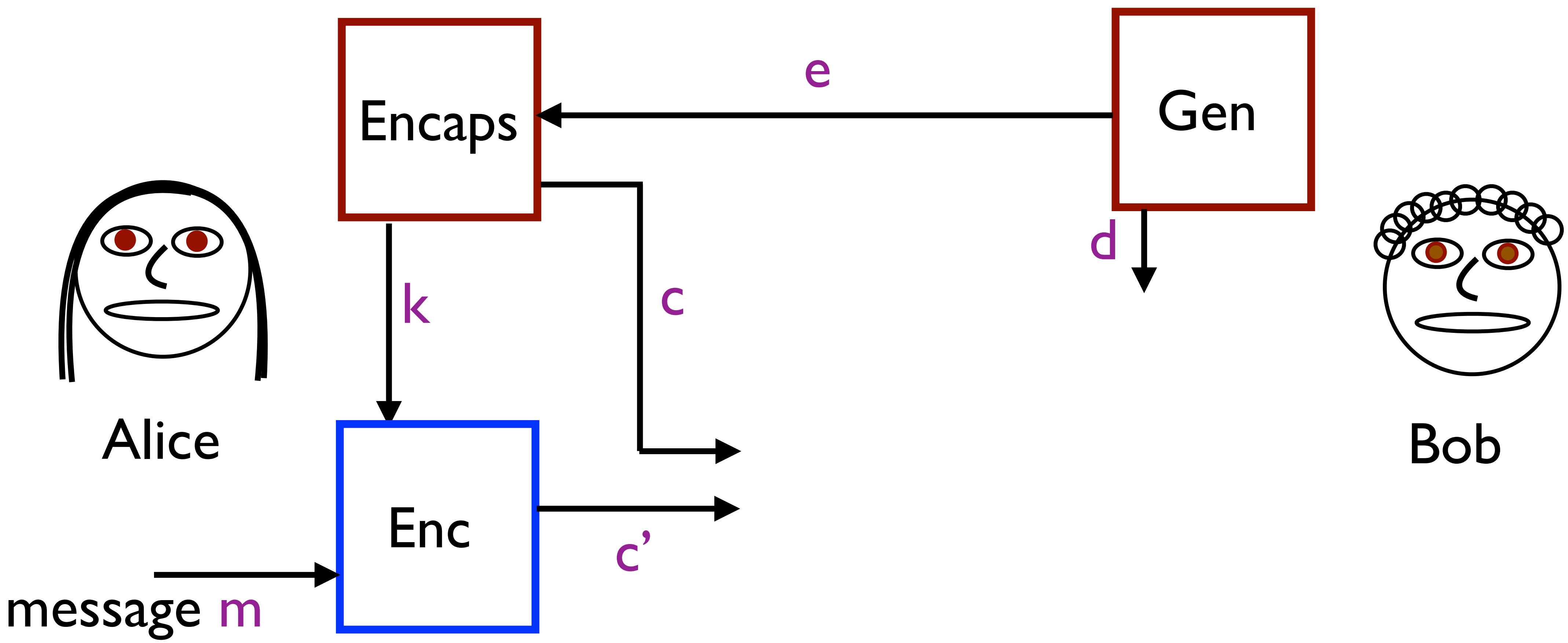
Using a KEM, we can effectively upgrade these private-key encryption protocols into public-key encryption protocols:



message m

KEM/DEM

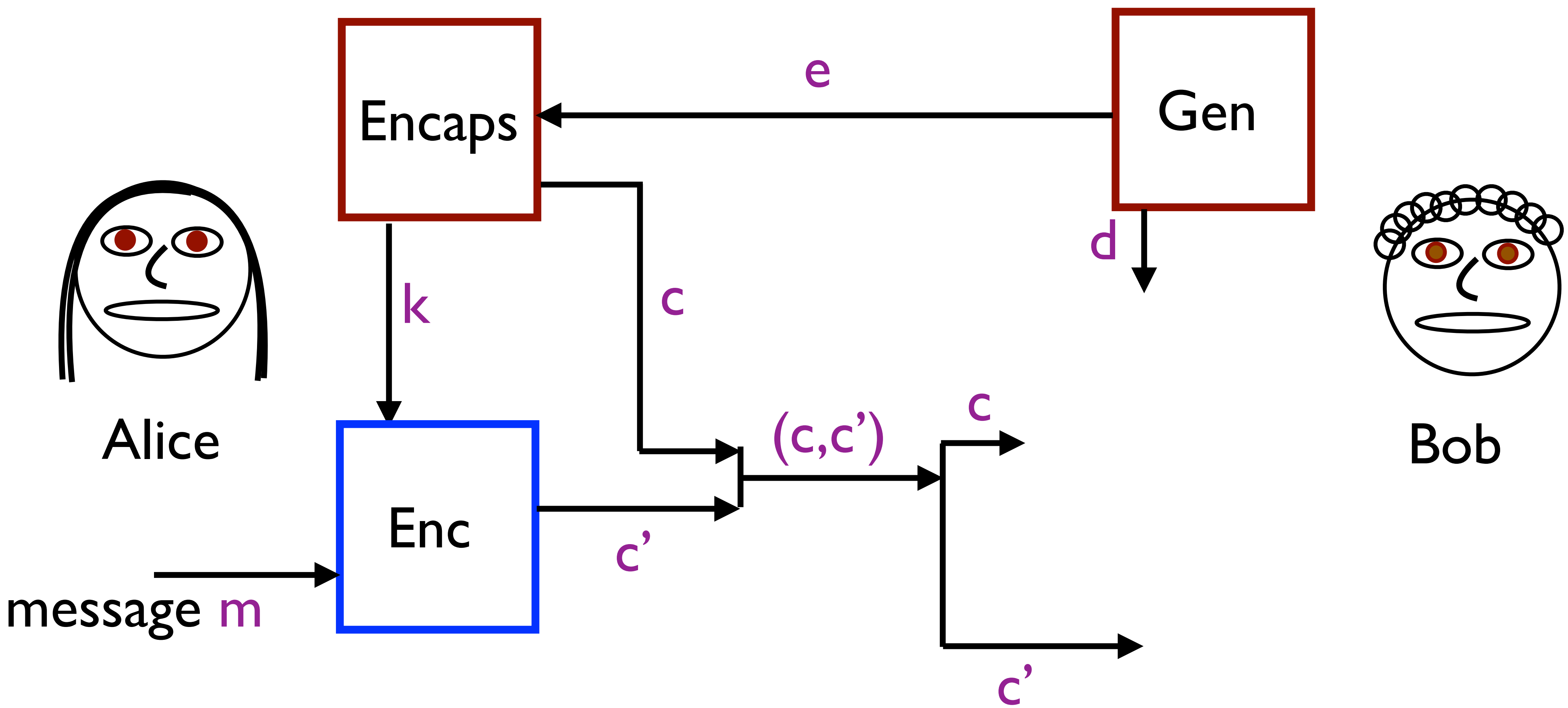
Using a KEM, we can effectively upgrade these private-key encryption protocols into public-key encryption protocols:



This class is being recorded

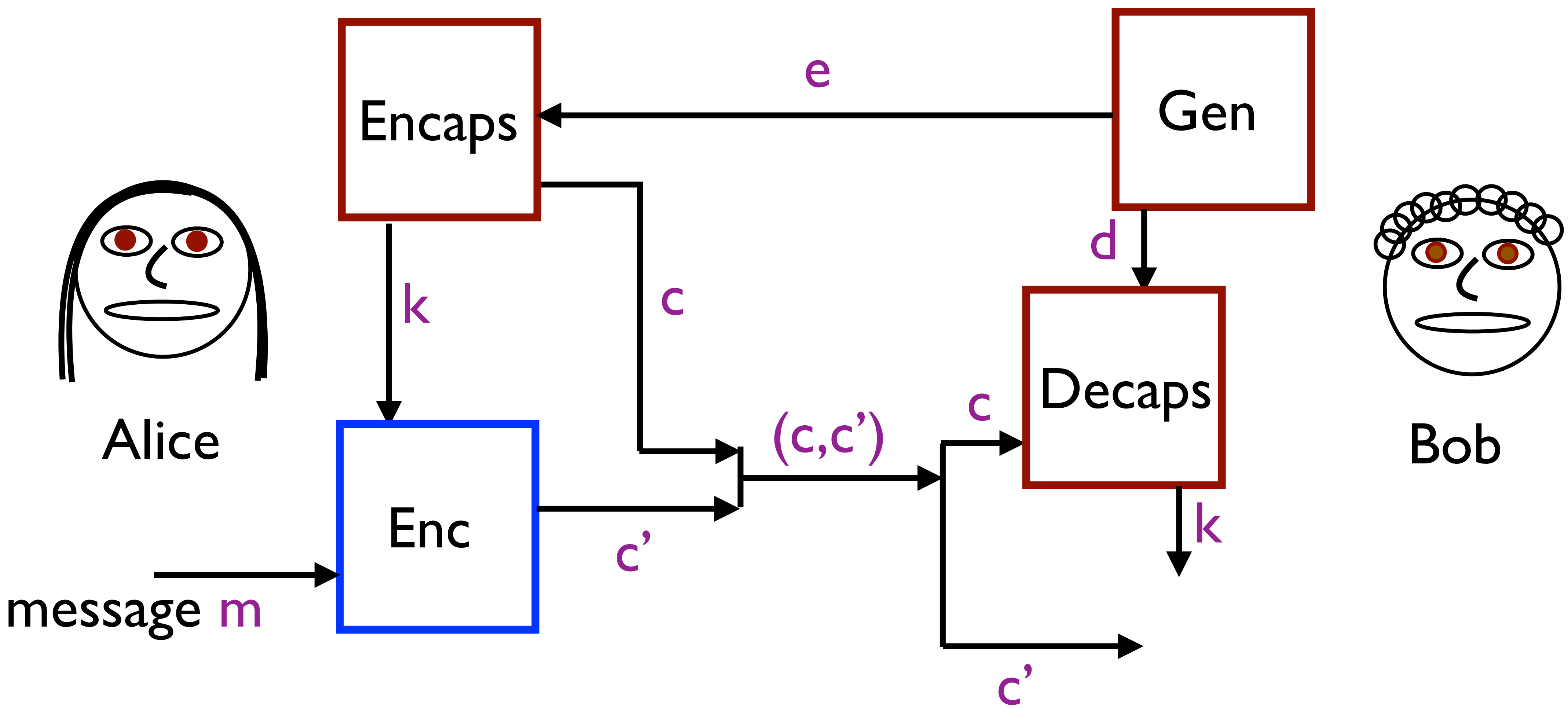
KEM/DEM

Using a KEM, we can effectively upgrade these private-key encryption protocols into public-key encryption protocols:



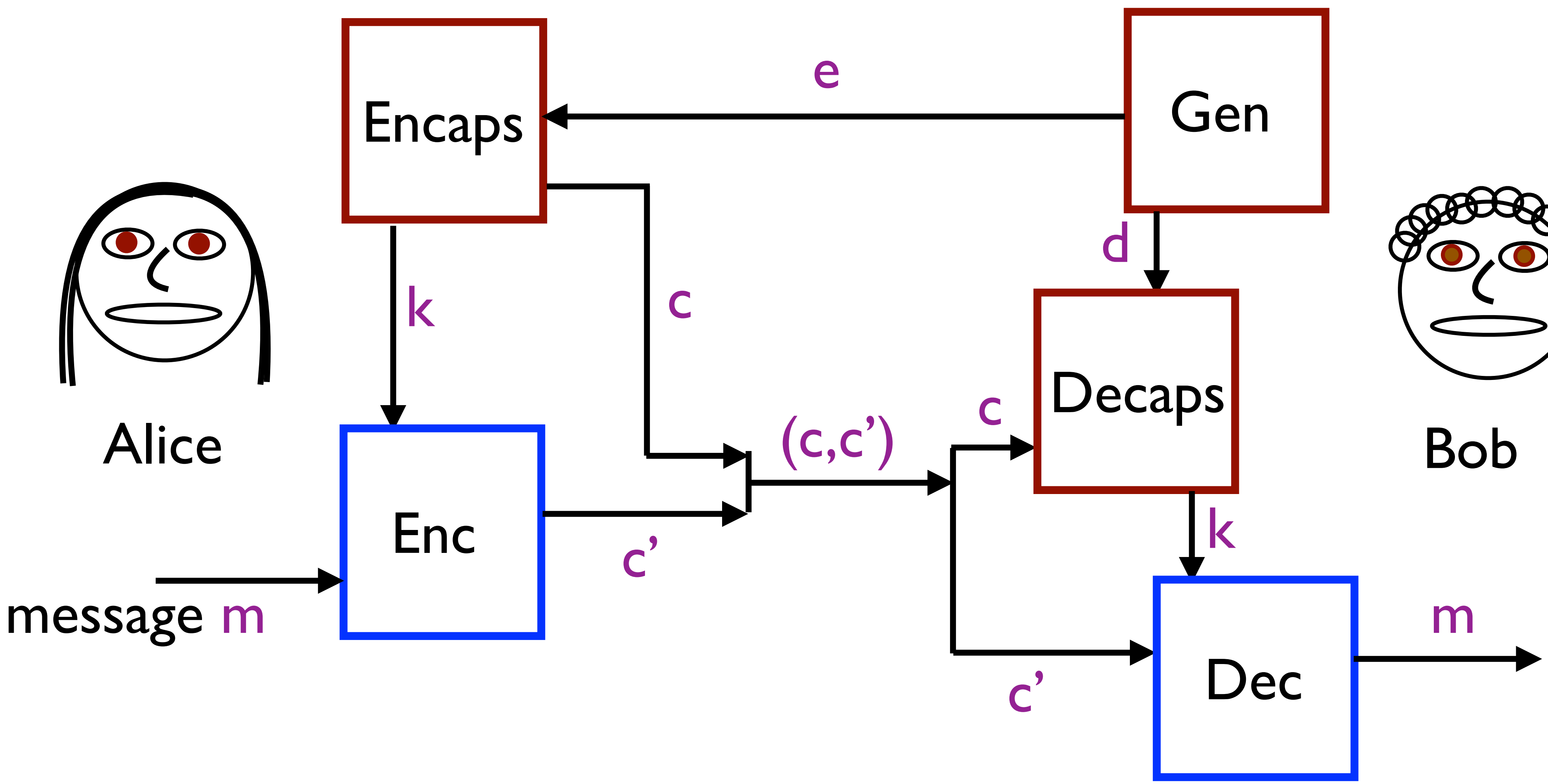
KEM/DEM

Using a KEM, we can effectively upgrade these private-key encryption protocols into public-key encryption protocols:



KEM/DEM

Using a KEM, we can effectively upgrade these private-key encryption protocols into public-key encryption protocols:



This class is being recorded

Generic MAC Constructions

Pseudorandom function $F_k(r)$ → $\mathrm{Mac}(k, m) = F_k(m)$

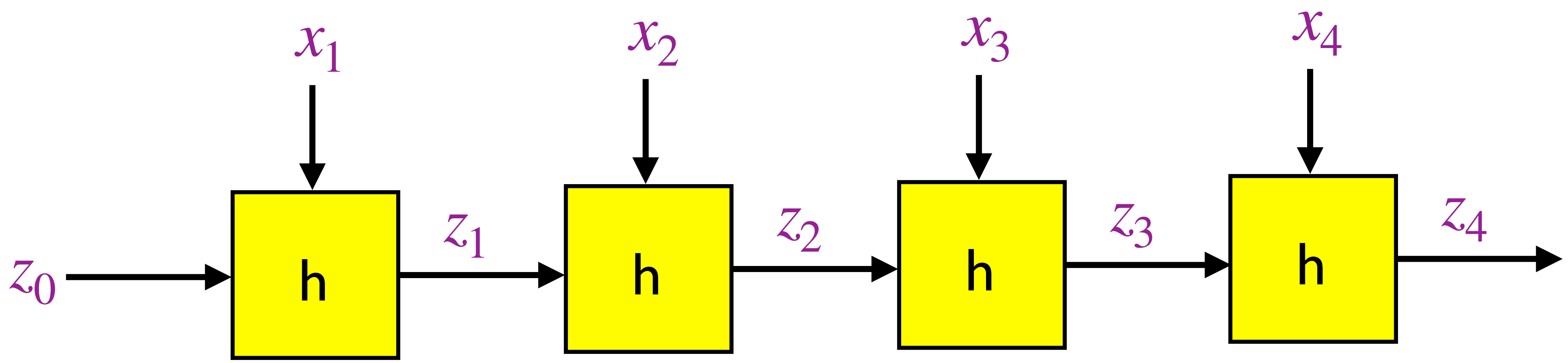
CBC-Mac → For longer messages; no IV, include length as first input, tag is only output of last block

Hash-and-Mac → $\mathrm{Mac}(k, H(m))$
(Note: previously I also had H(m) as part of the tag, but this is not needed.)

This class is being recorded

Hash function construction

Merkle-Damgard construction makes hash functions for arbitrary input out of a compression function of fixed size.



Need to pad the input appropriately.

RSA and Diffie-Hellman Encryption

Diffie-Hellman: Alice sends $A = g^a \bmod p$, Bob sends $B = g^b \bmod p$, key is $A^b = B^a = g^{ab} \bmod p$.

El Gamal: Public key is $g^b \bmod p$, private key is b, encryption is $mg^{ab} \bmod p$ (for secret random a).

DH KEM: Public key is $g^b \bmod p$, private key is b, ciphertext is $A = g^a \bmod p$, key is $H(A^b) \bmod p = H(g^{ab})$.

RSA: Public key is (N, e), private key is d such that $de = 1 \bmod \varphi(N)$. Encryption is $\tilde{m}^e \bmod N$ (with m appropriately padded to $\tilde{m}$).

RSA KEM: Public key is (N, e), private key is d such that $de = 1 \bmod \varphi(N)$. Encryption is $x^e \bmod N$, key is $H(x)$.

RSA and DSA signatures

RSA: Public key is (N, e), private key is d such that $de = 1 \bmod \varphi(N)$. Signature is $\sigma = H(m)^d \bmod N$.

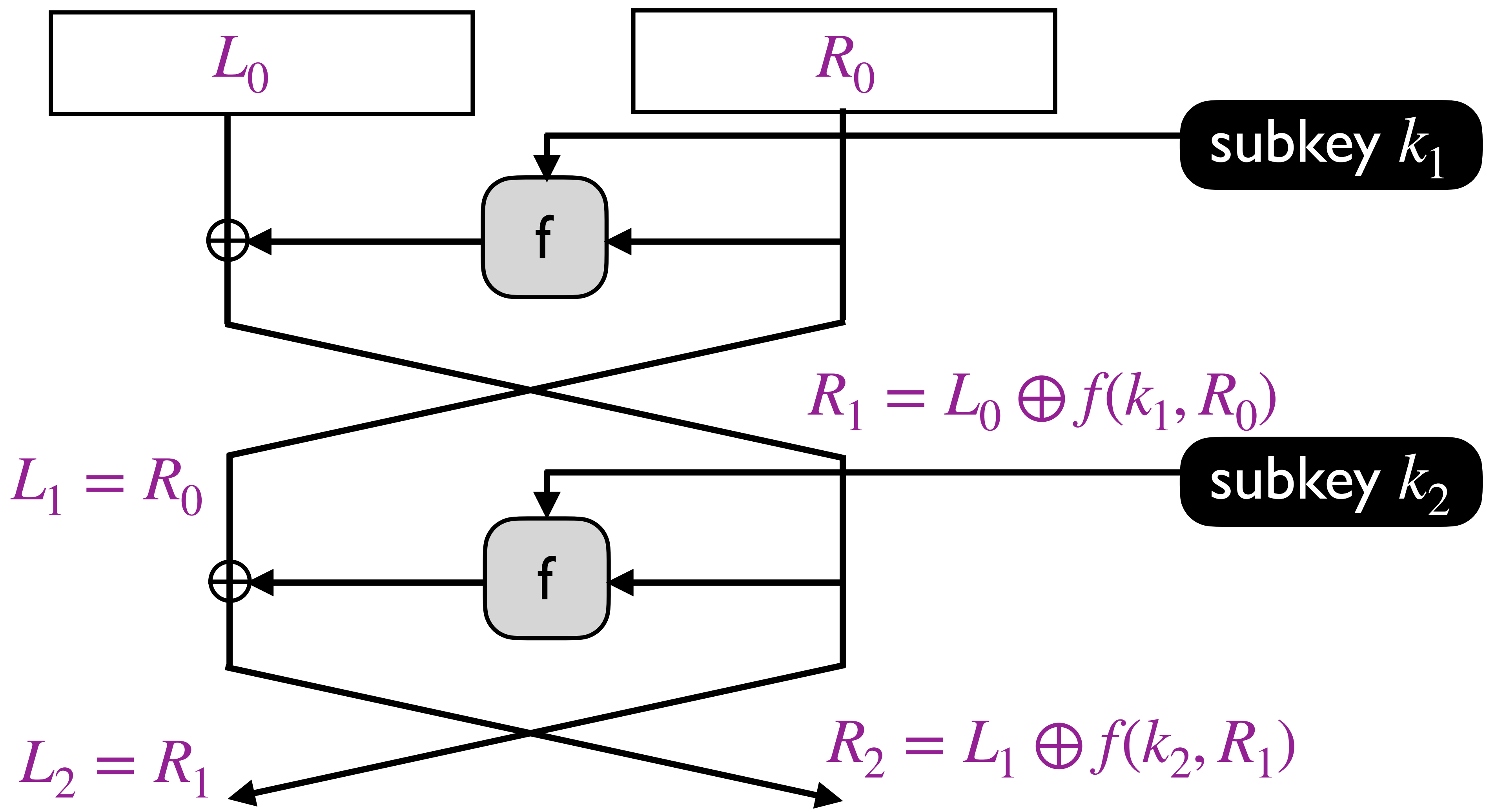
DSA: Public key is $y = g^x \bmod p$, private key is x. Signature is $s = k^{-1}(H(m) + xr) \bmod q$ for random k, $r = g^k \bmod p$.

(Verify by checking that $r = g^{H(m)s^{-1}} y^{rs^{-1}} \bmod p$)

This class is being recorded

Feistel Network

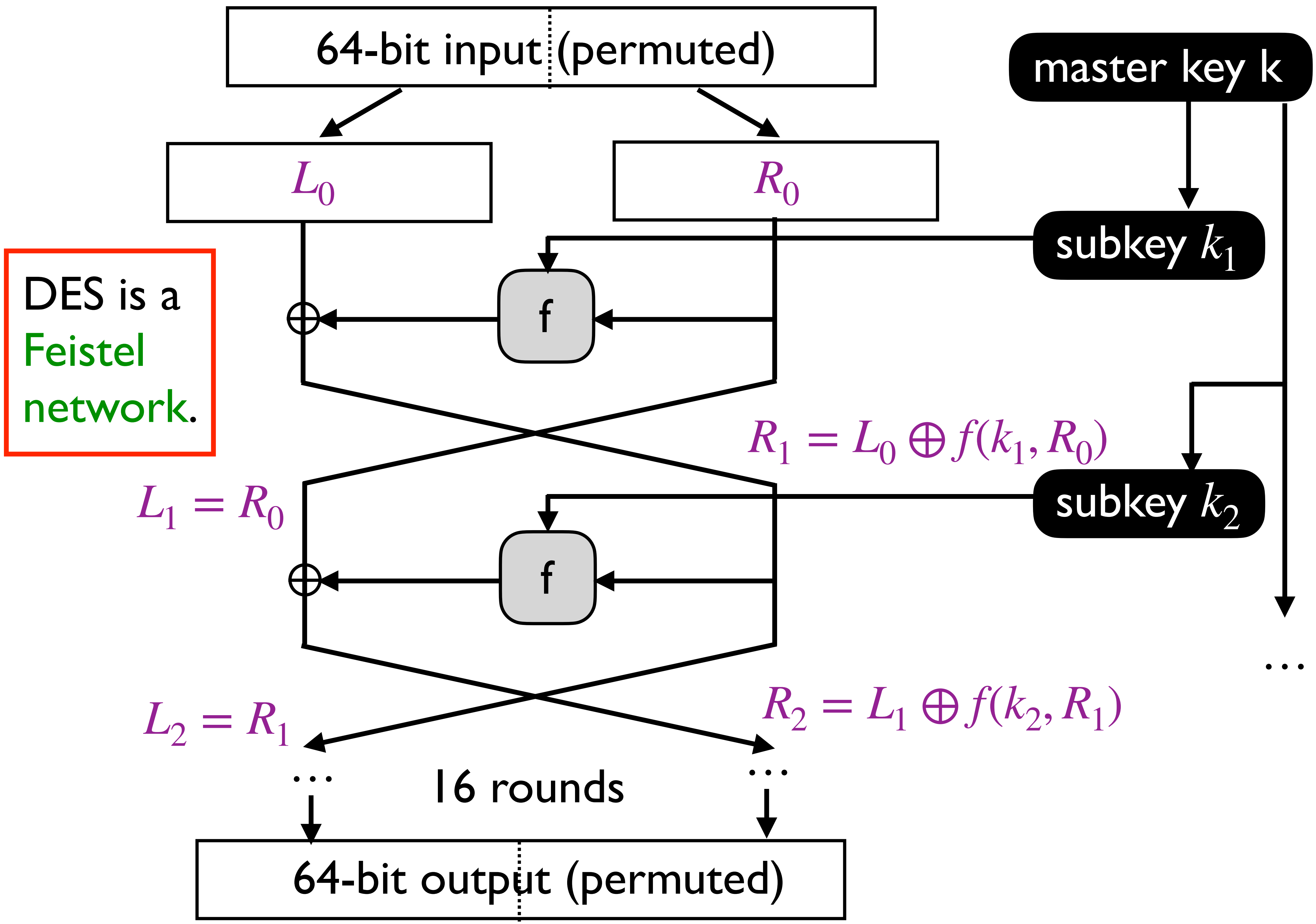
A Feistel network consists of a sequence of rounds sequentially acting on the message, which is split into a left and right half.



In each round, the current right half is fed into a round function f with a key for the round and then XORed with the left half. The modified left half and old right half are then switched.

This class is being recorded

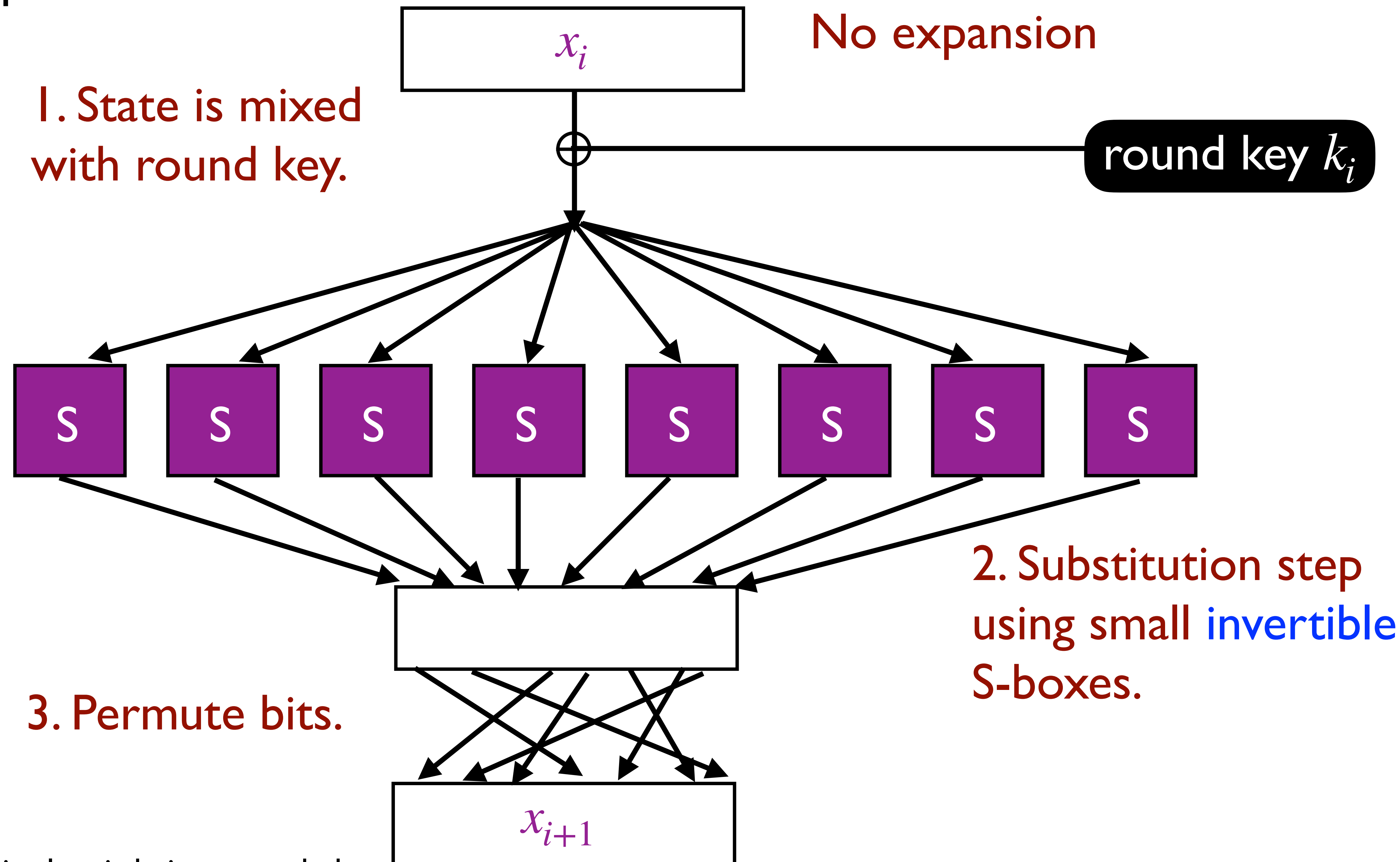
DES Overview



This class is being recorded

Substitution-Permutation Networks

The DES mangler function is a variant of a substitution-permutation network, a design paradigm for pseudorandom permutations.

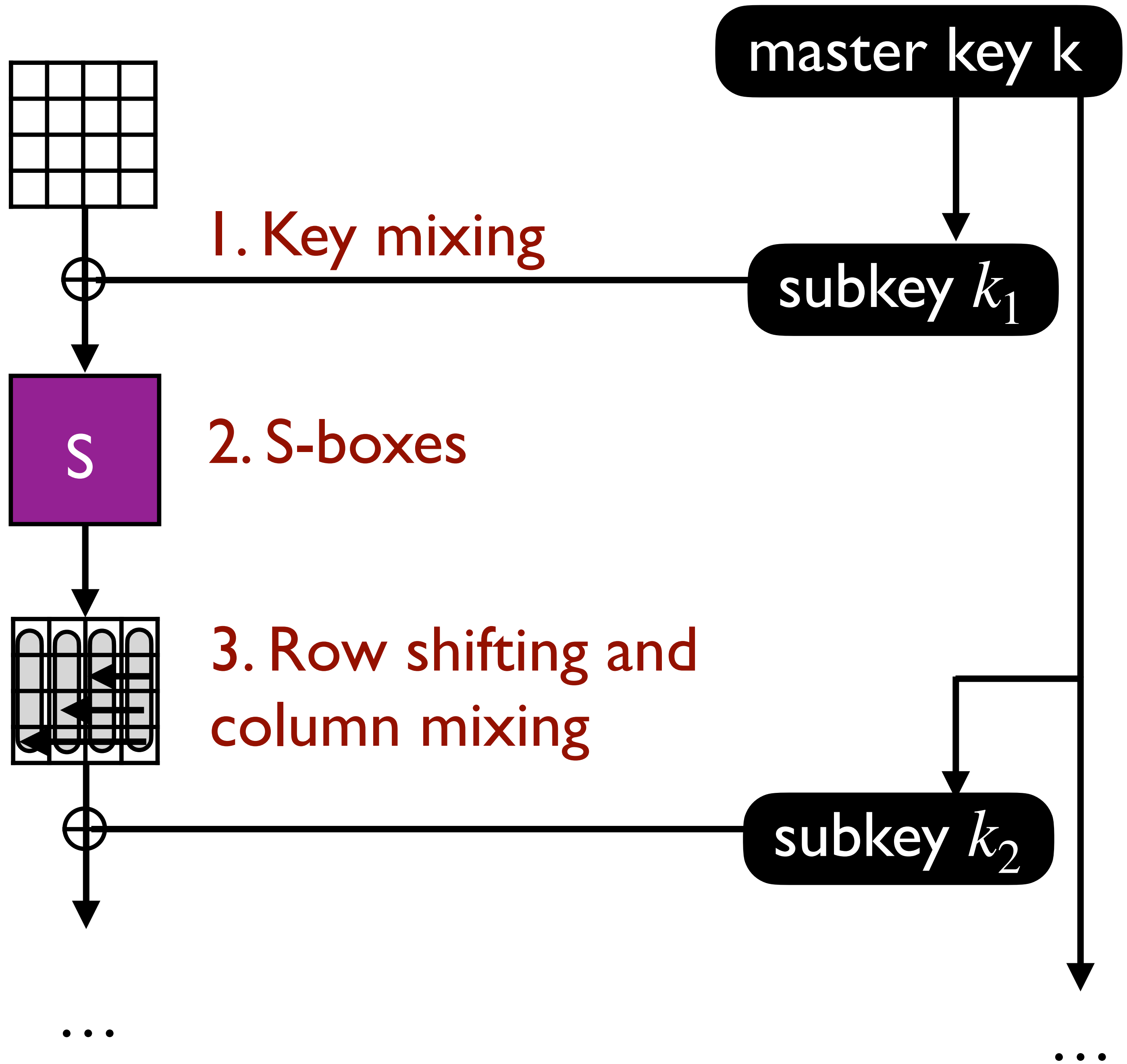


This class is being recorded

AES Overview

The AES permutation takes a 128-bit input represented as 4 x 4 matrix of bytes:

AES is basically a substitution-permutation network.



This class is being recorded