CMSC/Math 456: Cryptography (Fall 2022)

Lecture 20

Daniel Gottesman

Administrative

Problem set #6 (a programming assignment) is due Thursday at *midnight*.

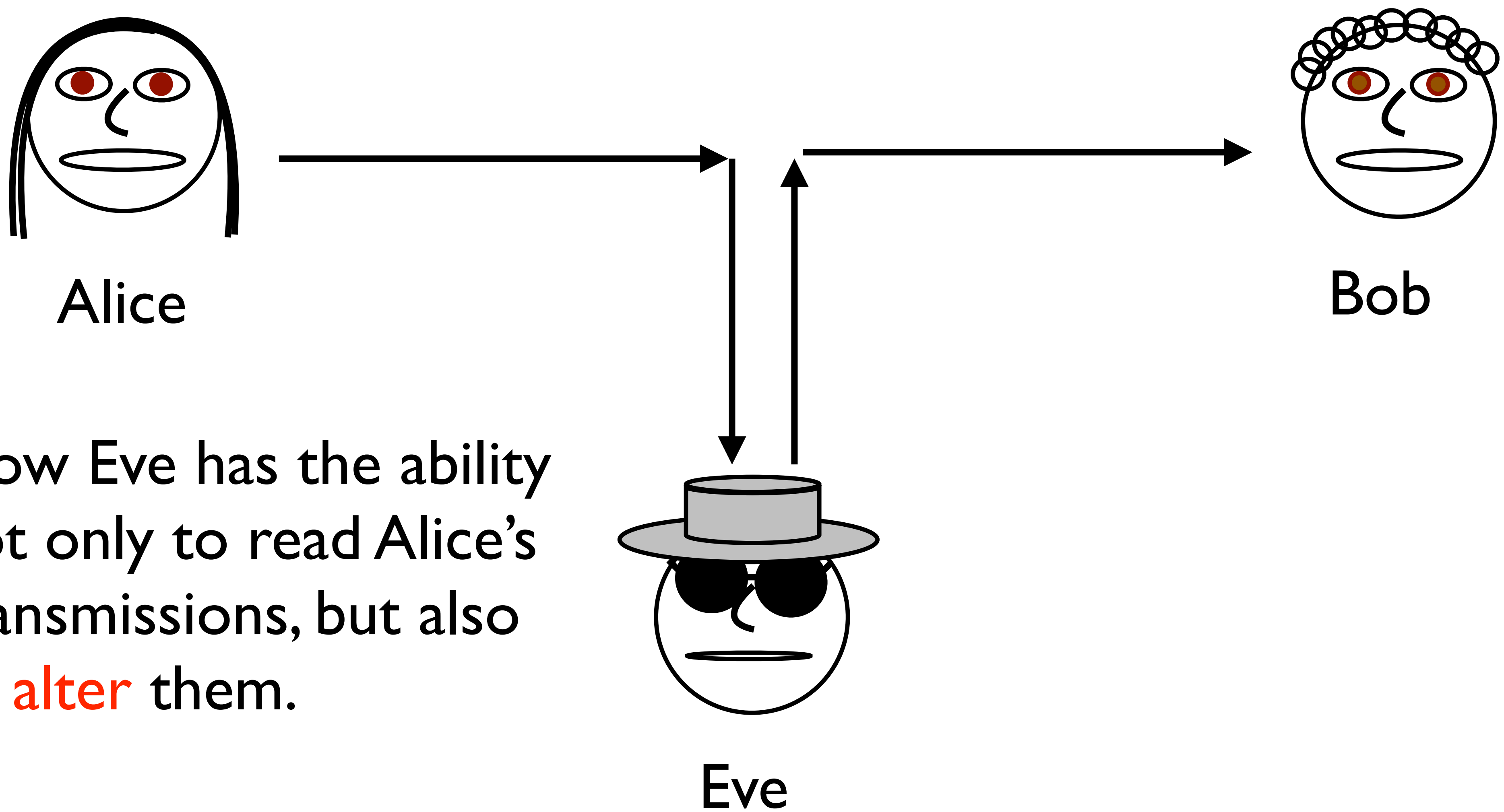
Important reminder: You can collaborate with other students, but you must

- Write your own code! You can share an algorithm but you shouldn't be looking at someone else's solution when you write your code.
- Put a comment in the code saying who you collaborated with.

This class is being recorded

Active Threat Model

Recall that for message authentication, Eve is able to change messages sent between Alice and Bob. How does this threat model affect encryption?



Now Eve has the ability not only to read Alice's transmissions, but also to alter them.

Malleability

One thing we have to worry about: Malleability. Eve can still change the message in predictable ways even if she can't read it.

One-time pad:

Key: 00101000101110101010

Message: 10111110010011001100

Ciphertext: 10010110111101100110

↓

Alteration: 10010110111100100110

Decryption: 10111110010010001100

↑

Changing the ciphertext produces a predictable change in the decrypted plaintext.

Malleability of RSA

Plain RSA and Padded RSA are also malleable:

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

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

Bob then decrypts the message $2\tilde{m}$ instead of $\tilde{m}$.

Eve can multiply the message by any constant factor in this way.

Malleability is a Threat

Why do we care about malleability if Eve doesn't learn the secret?

For the same reasons we care about authenticity: Bob might take incorrect and damaging actions if he acts on an altered message.

- Changing the URL in an email message might direct you to website with malware.
- Changing the votes in an encrypted ballot.
- Changing the dollar amount or account number in a message to a bank might result in too much money being sent or the money being sent to Eve.
- Changing orders to a military unit might result in the unit being out of position.

Padding Oracle Attack

If Bob reacts differently to different messages, malleability can also result in loss of secrecy!

Example:

Alice sends a message to Bob using AES with CBC-mode. Since the message might not be an exact multiple of the block size, it must be padded to reach the right size.

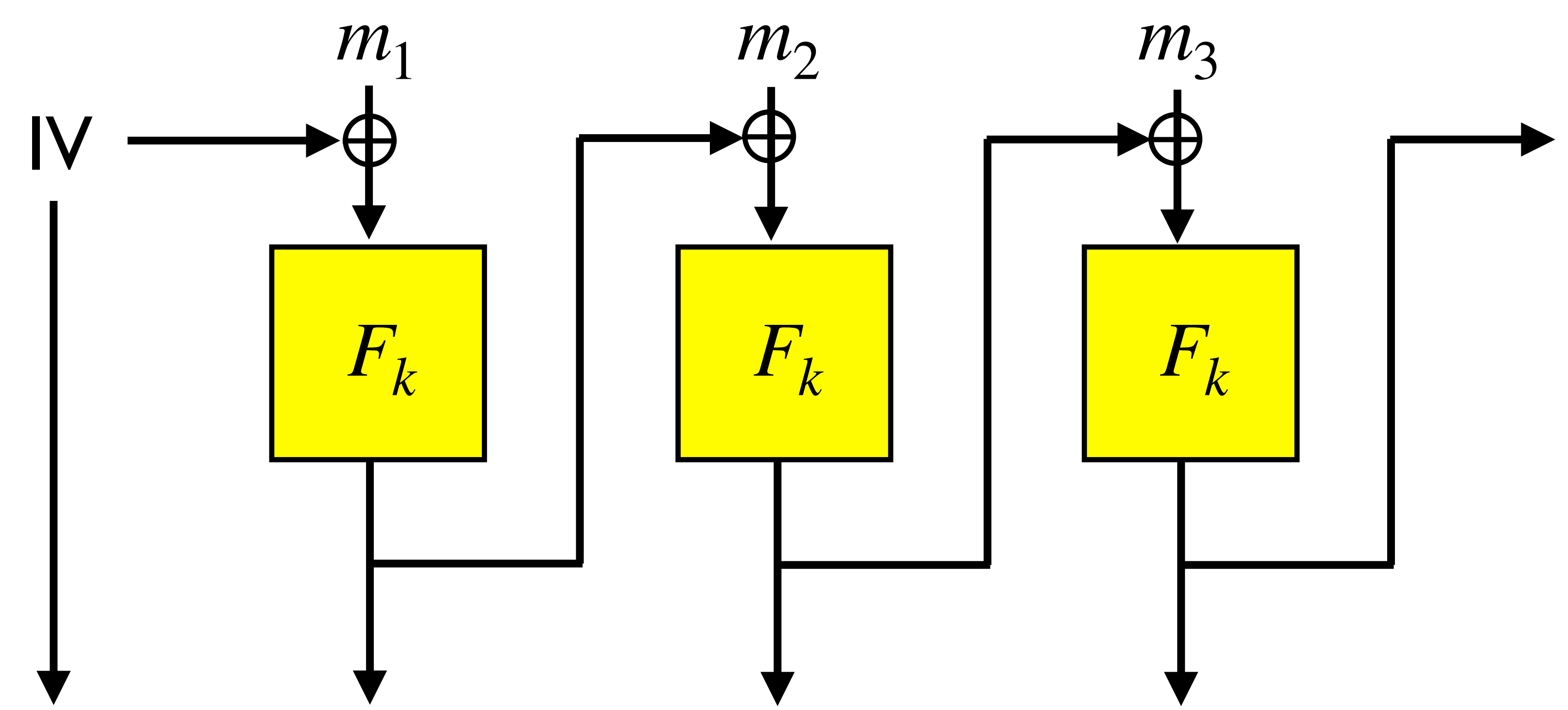
Suppose we pad in this way (which is standard, PKCS #7):

If b bytes are needed to reach the block size, fill those b blocks all with the number b. (This is then easy for the receiver to strip off.)

If Bob receives a message which is not correctly padded, he returns an error message: "Please resend."

CBC Mode

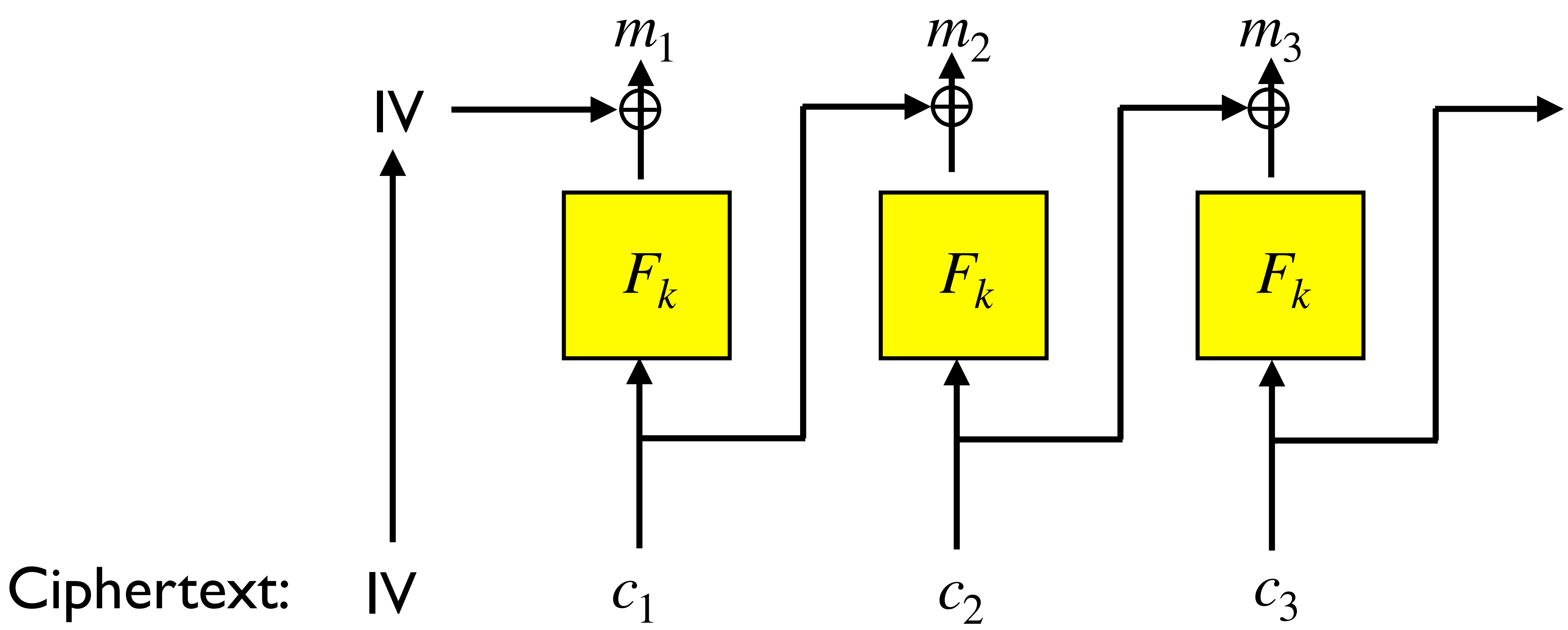
Recall CBC mode. Decryption runs this backwards.



Ciphertext: IV c_1 c_2 c_3

CBC Mode

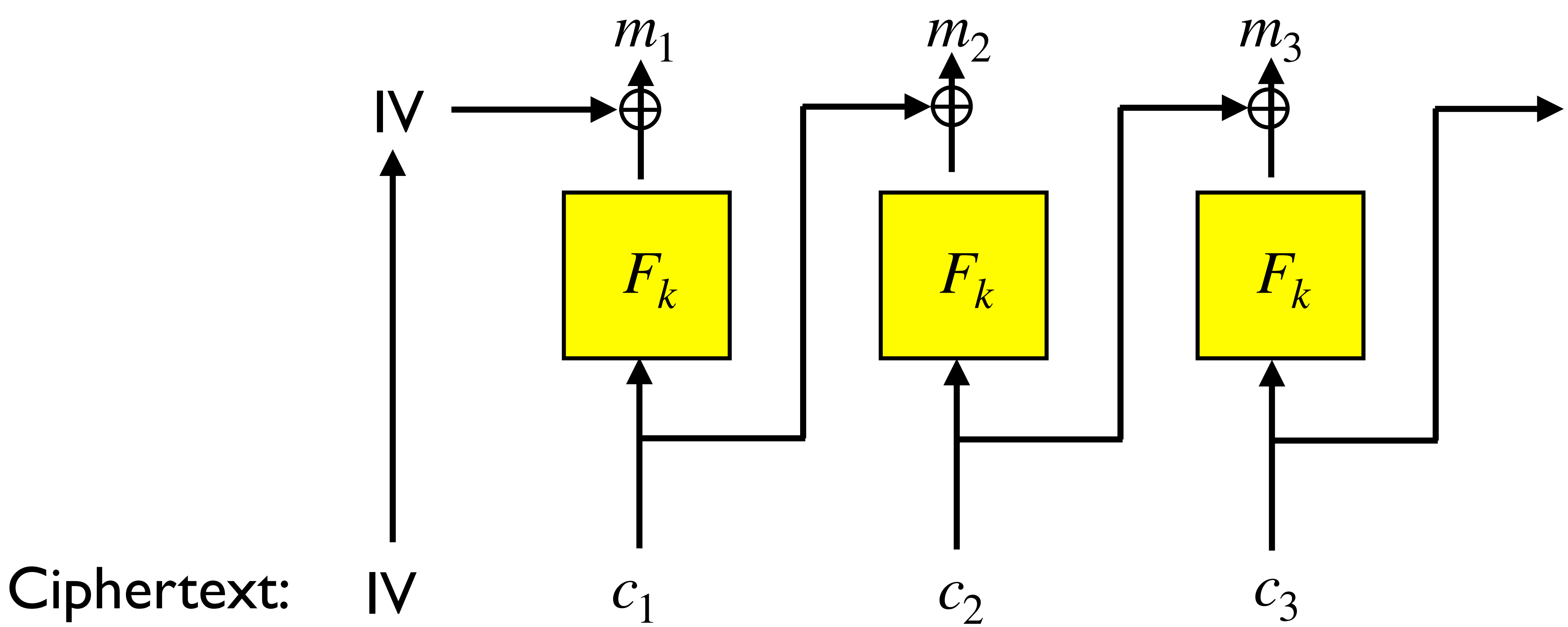
Recall CBC mode. Decryption runs this backwards.

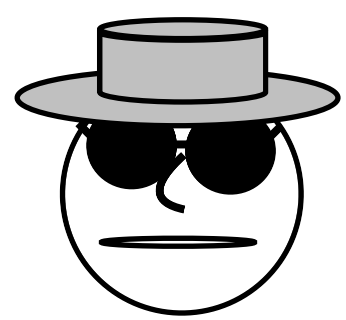


This class is being recorded

CBC Mode

Recall CBC mode. Decryption runs this backwards.

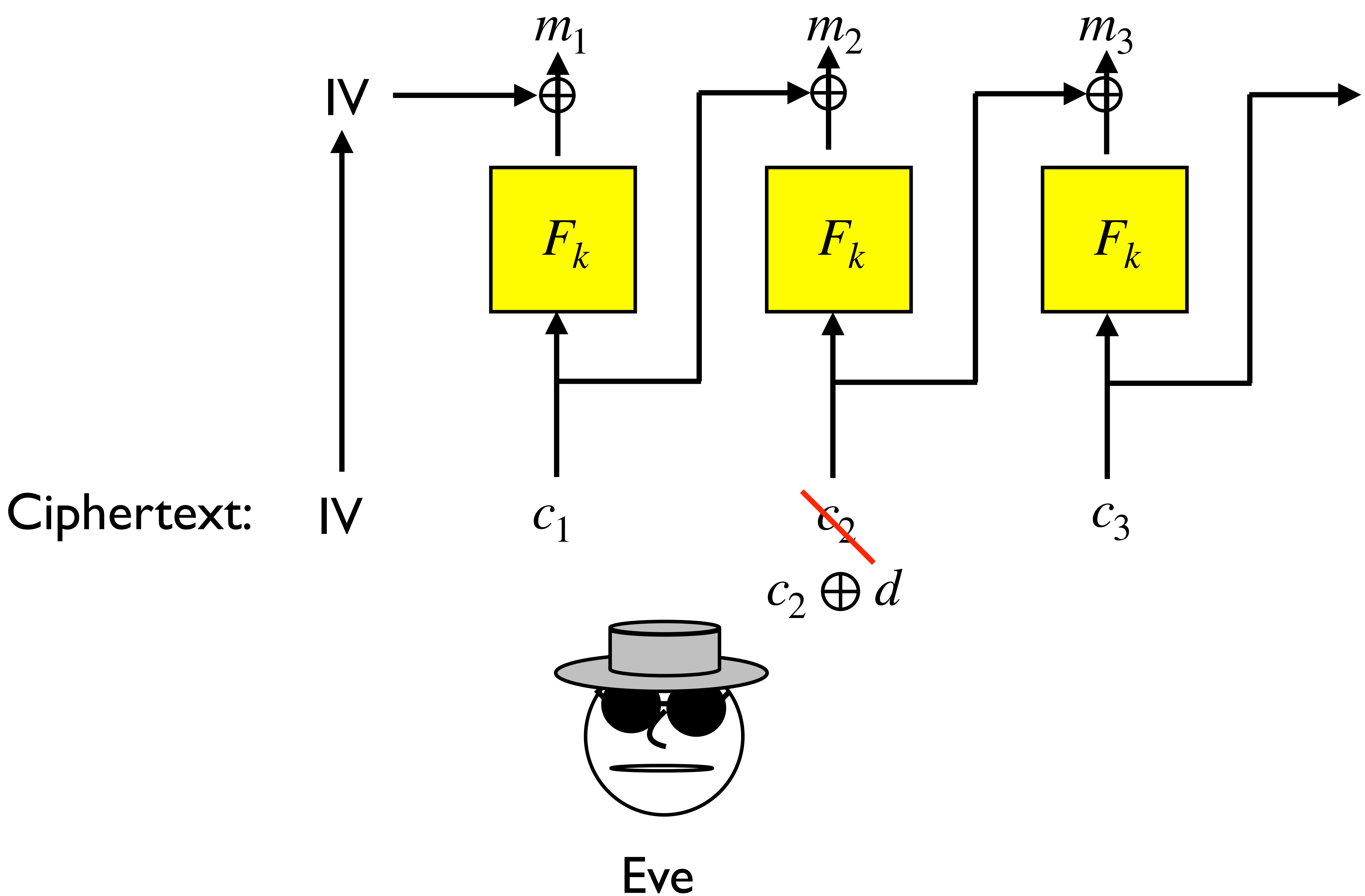




Eve

CBC Mode

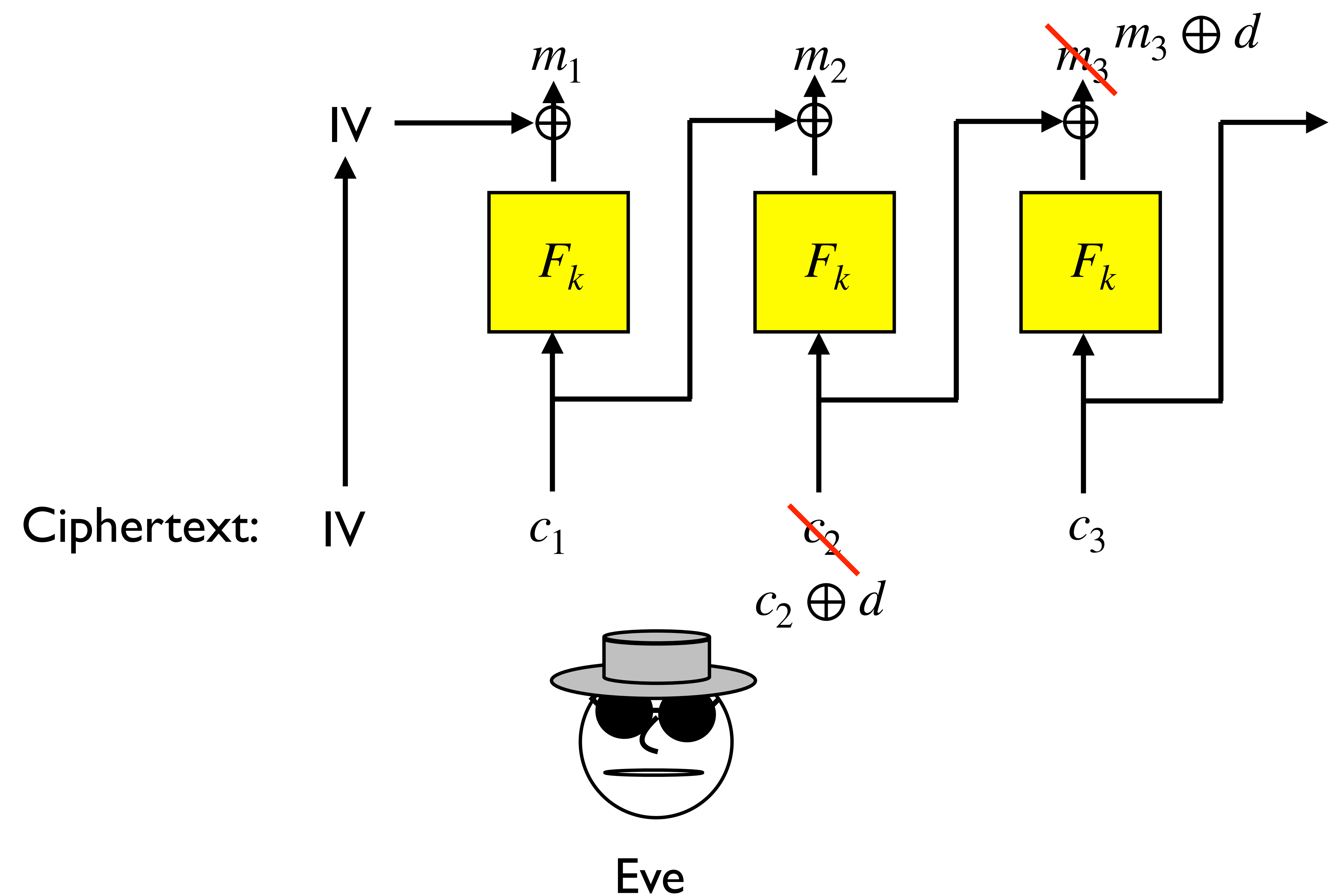
Recall CBC mode. Decryption runs this backwards.



This class is being recorded

CBC Mode

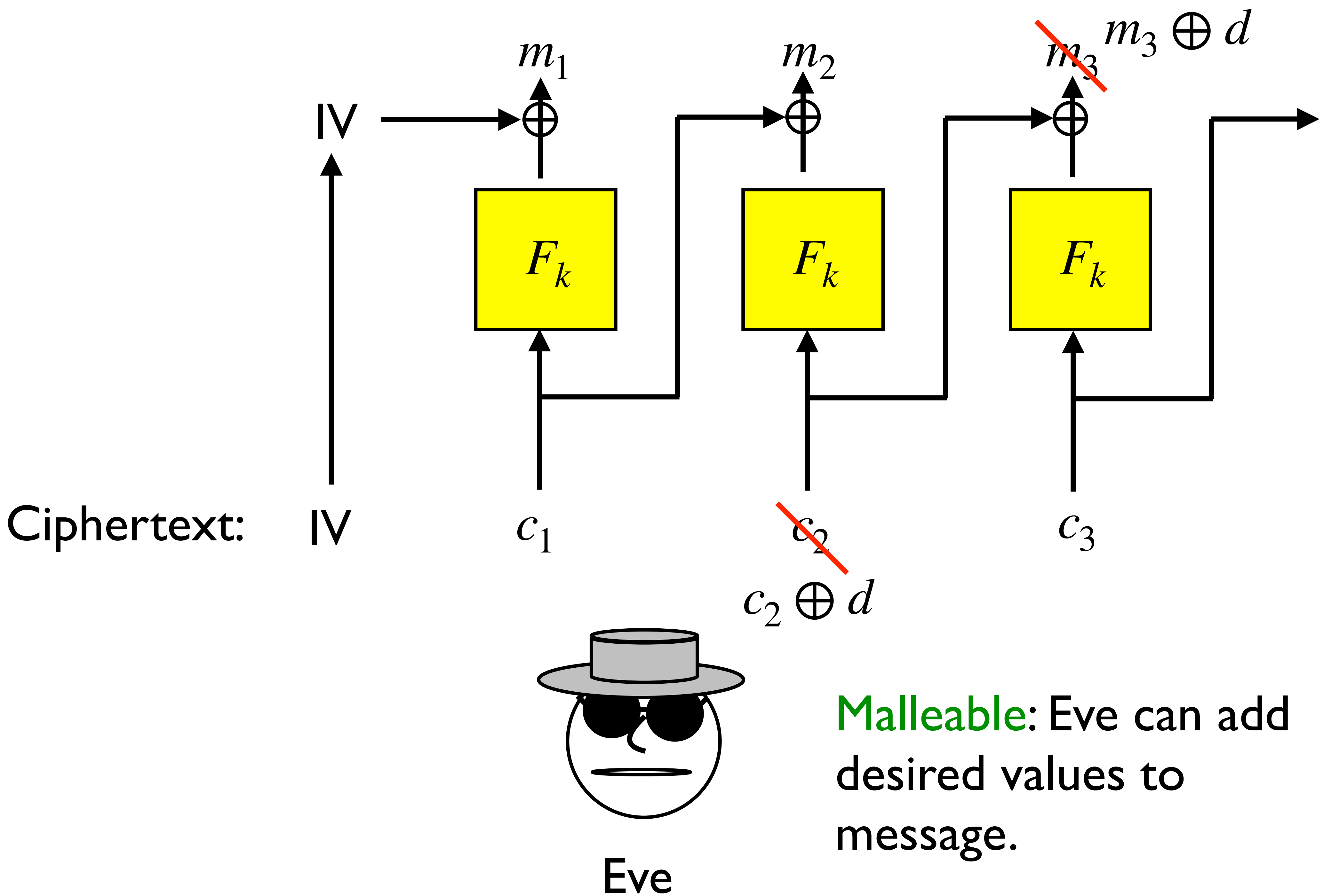
Recall CBC mode. Decryption runs this backwards.



This class is being recorded

CBC Mode

Recall CBC mode. Decryption runs this backwards.

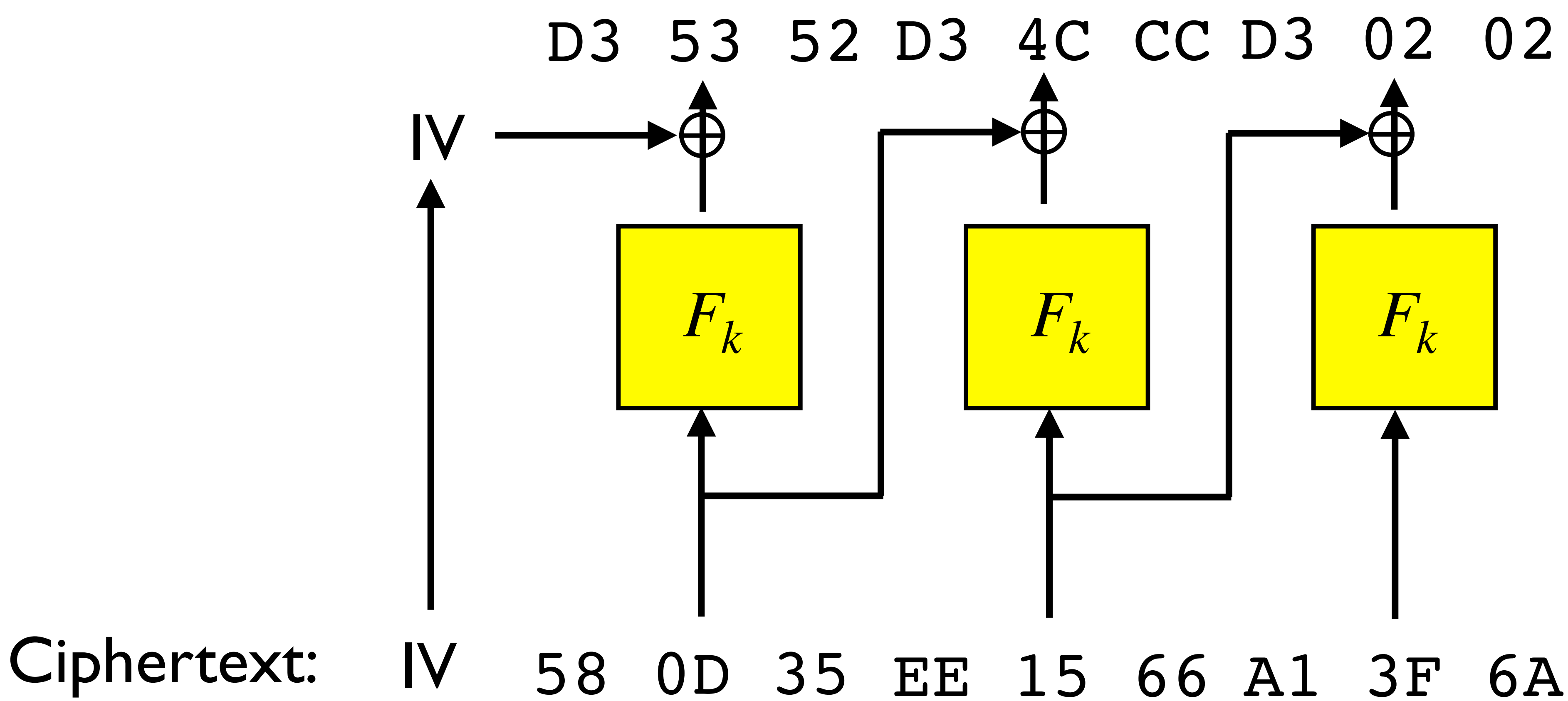


Malleable: Eve can add desired values to message.

This class is being recorded

Learning # of Padded Bits

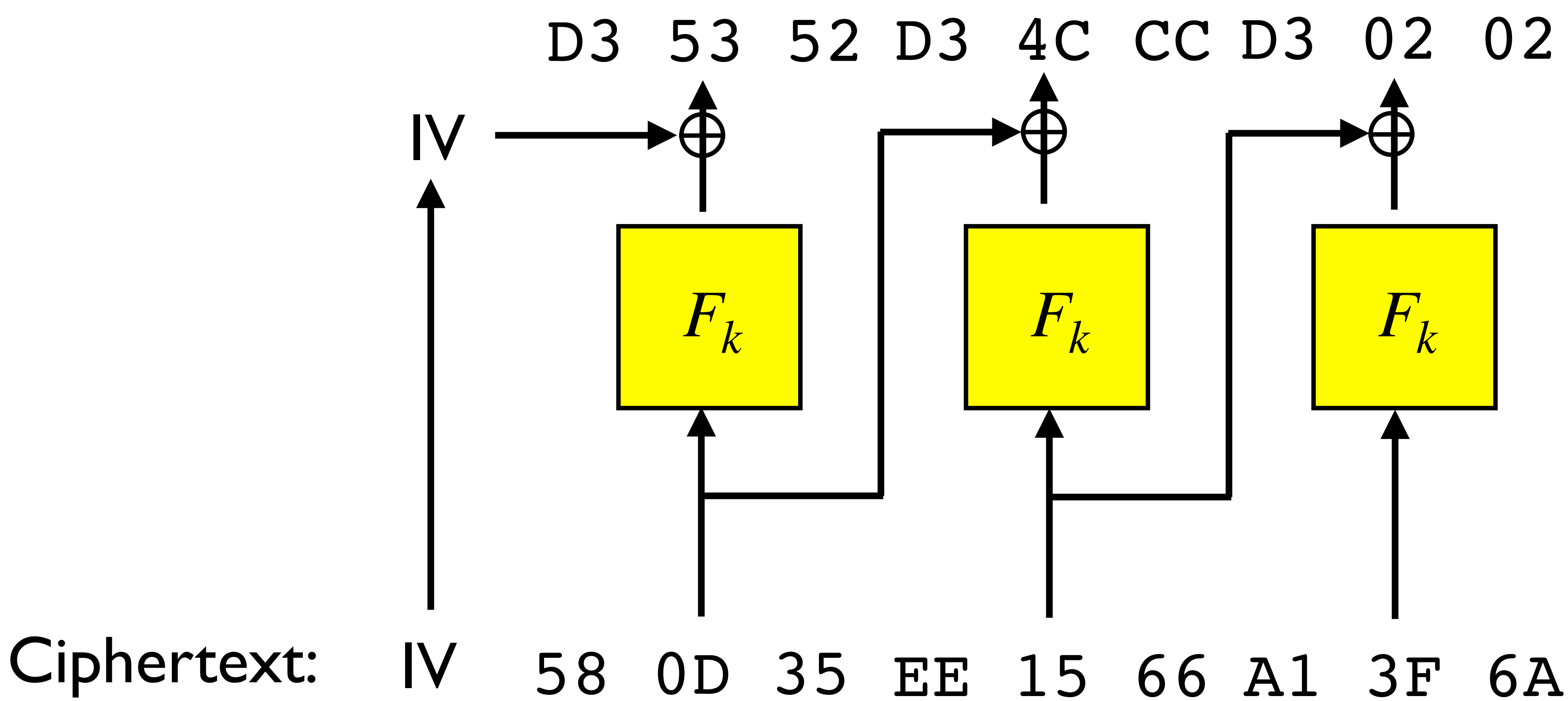
If Eve can alter sent messages, she can learn the padding length.

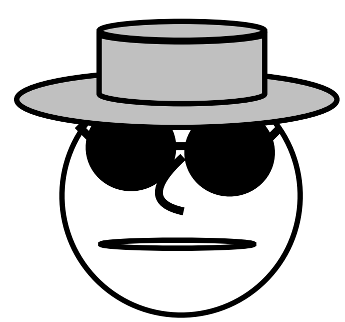


This class is being recorded

Learning # of Padded Bits

If Eve can alter sent messages, she can learn the padding length.



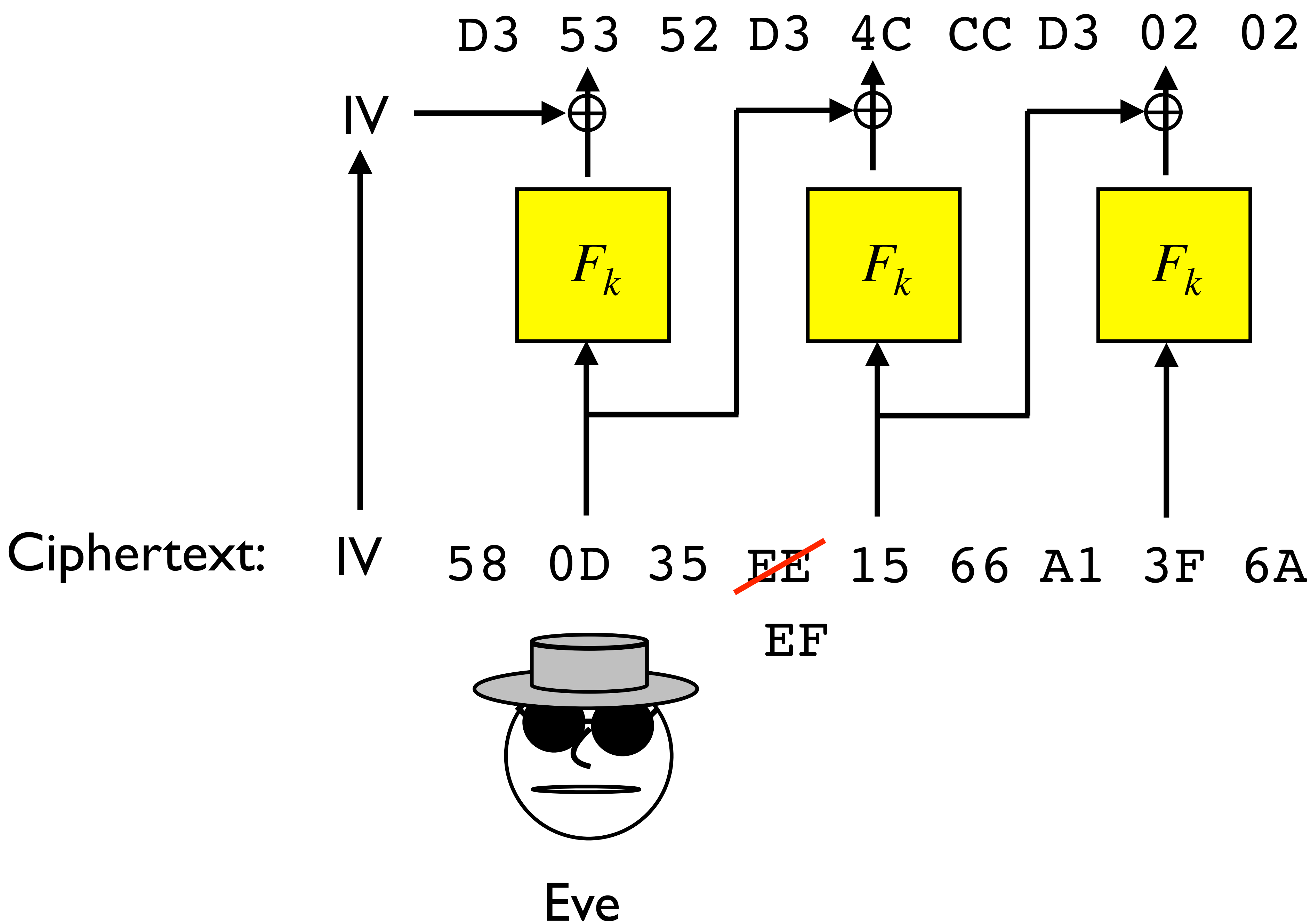


Eve

This class is being recorded

Learning # of Padded Bits

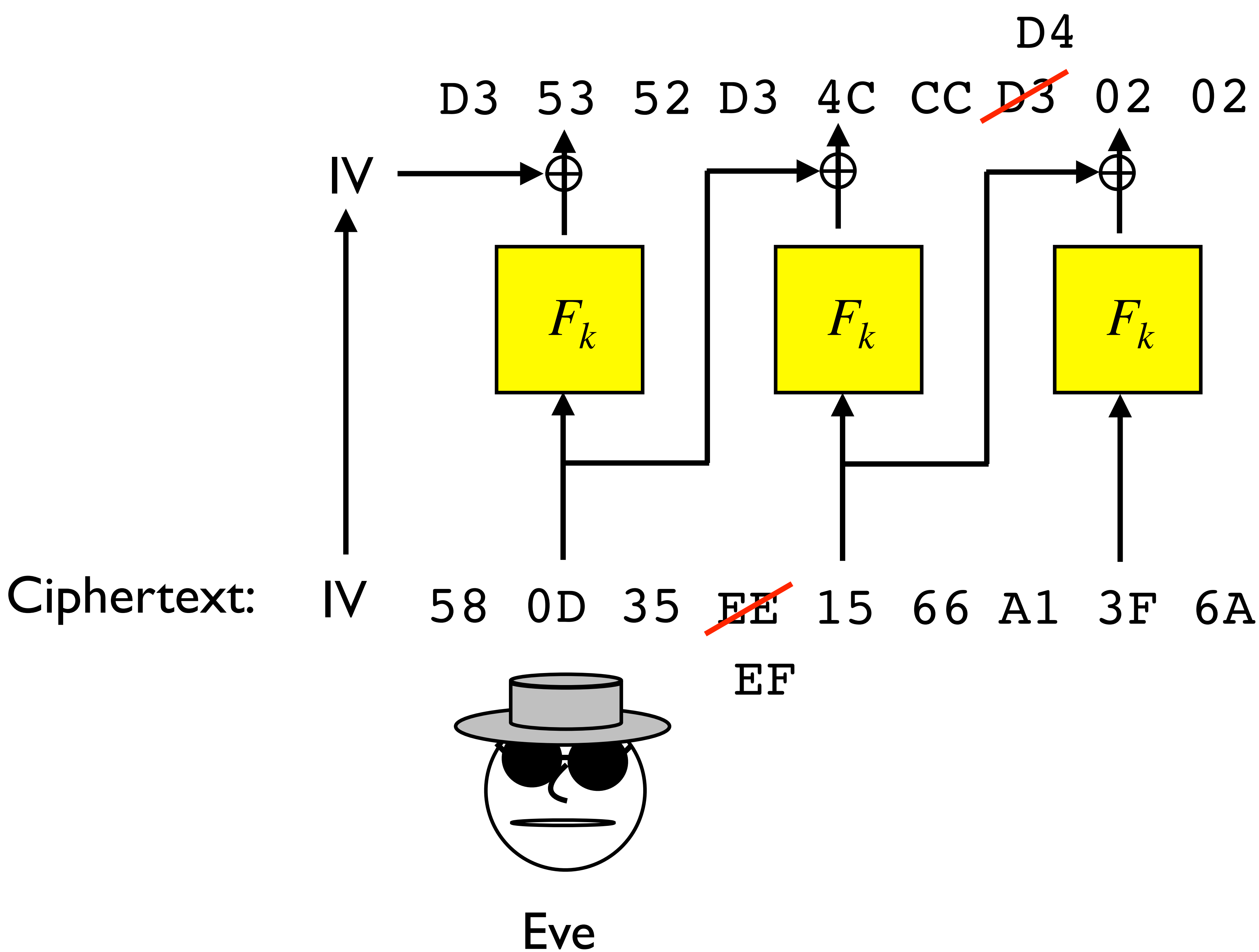
If Eve can alter sent messages, she can learn the padding length.



This class is being recorded

Learning # of Padded Bits

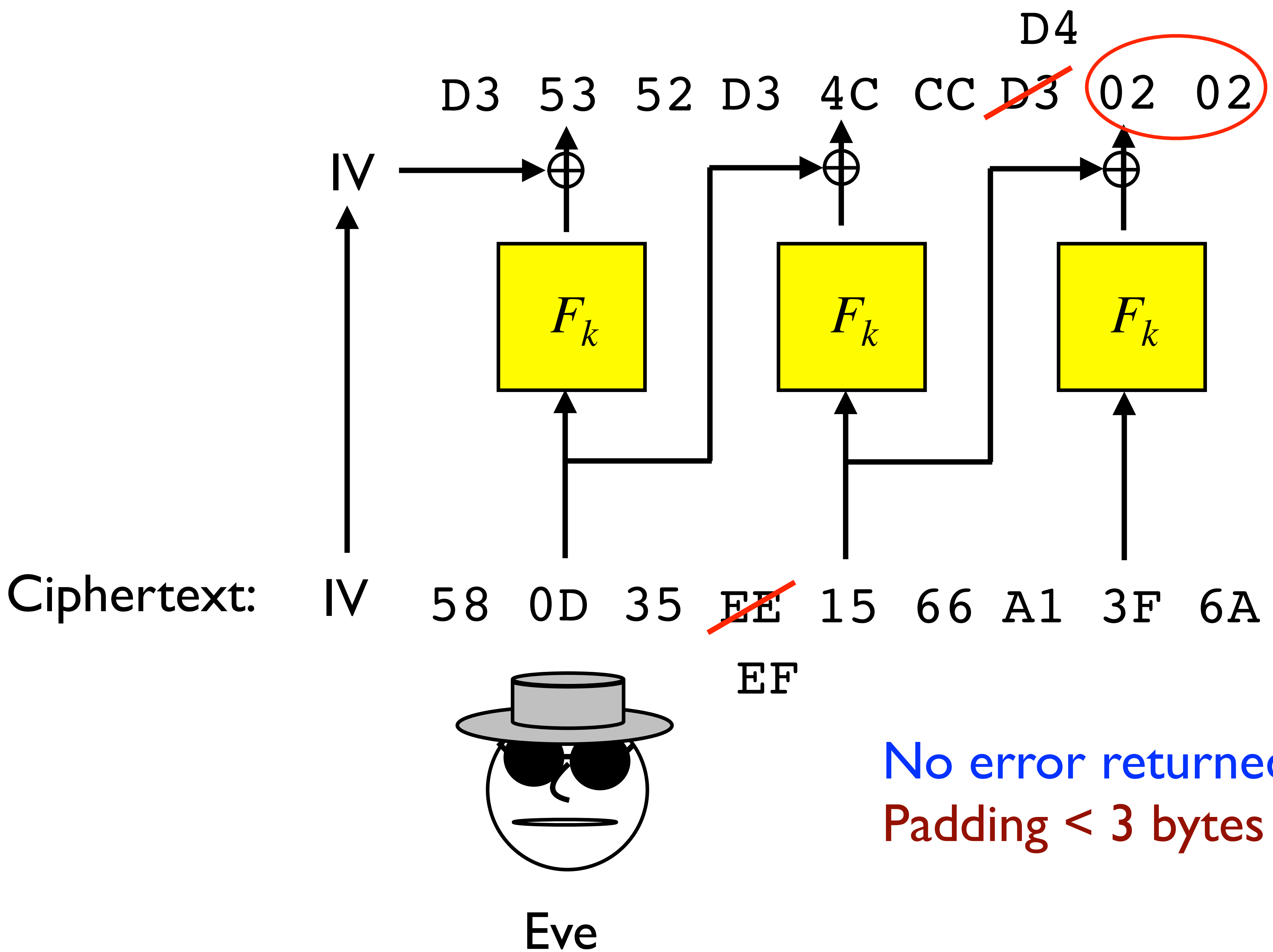
If Eve can alter sent messages, she can learn the padding length.



This class is being recorded

Learning # of Padded Bits

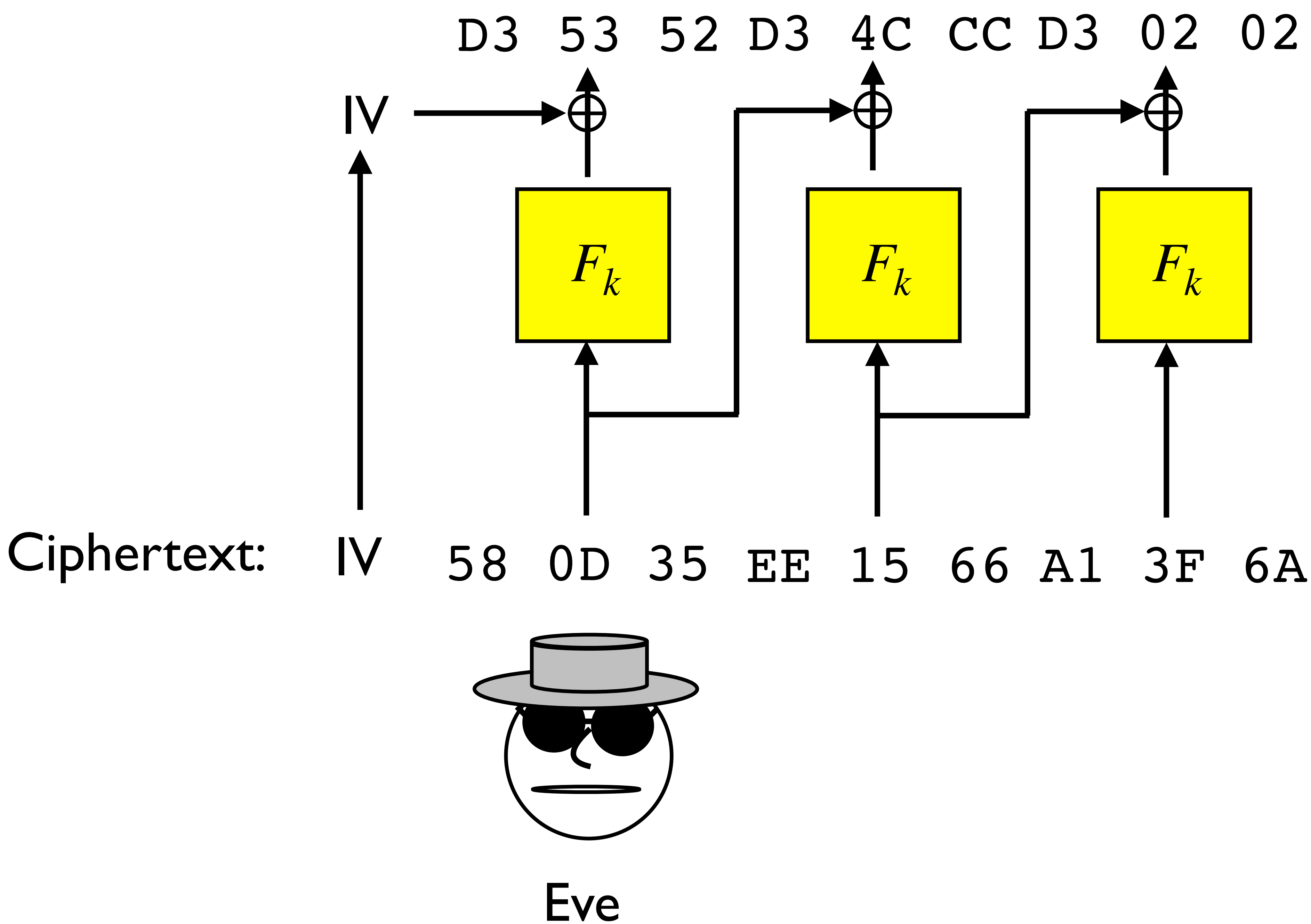
If Eve can alter sent messages, she can learn the padding length.



No error returned:
Padding < 3 bytes

Learning # of Padded Bits

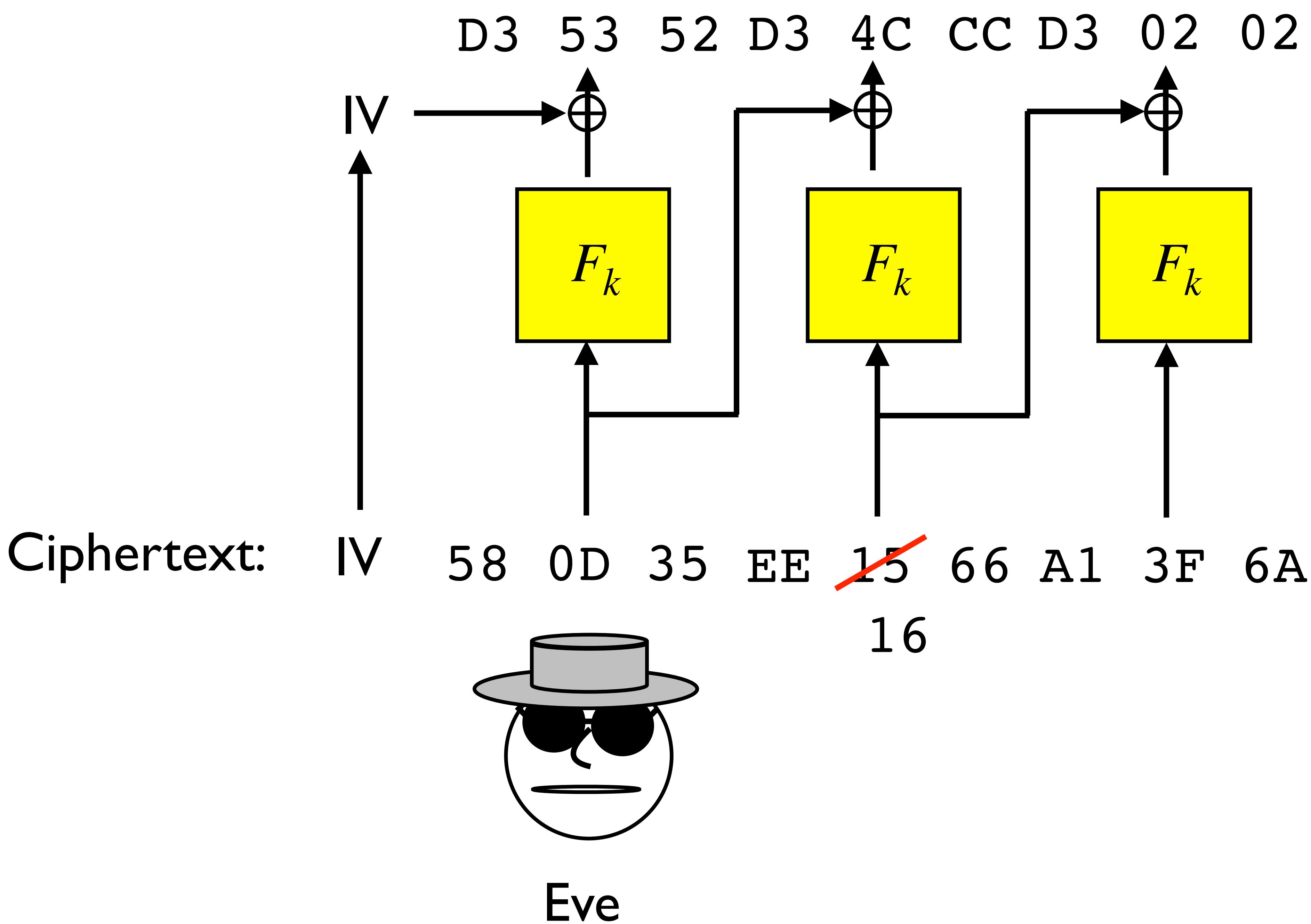
If Eve can alter sent messages, she can learn the padding length.



This class is being recorded

Learning # of Padded Bits

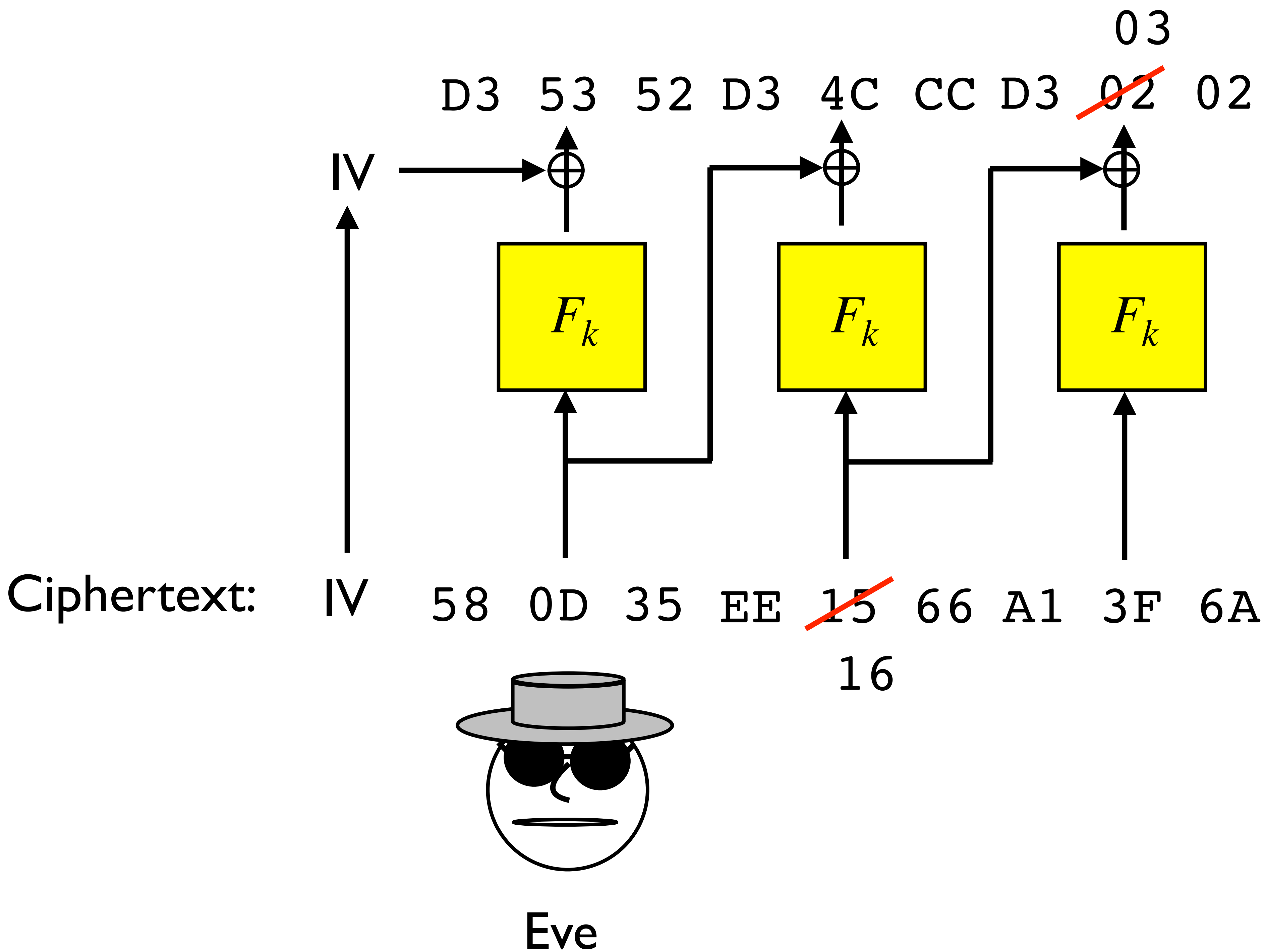
If Eve can alter sent messages, she can learn the padding length.



This class is being recorded

Learning # of Padded Bits

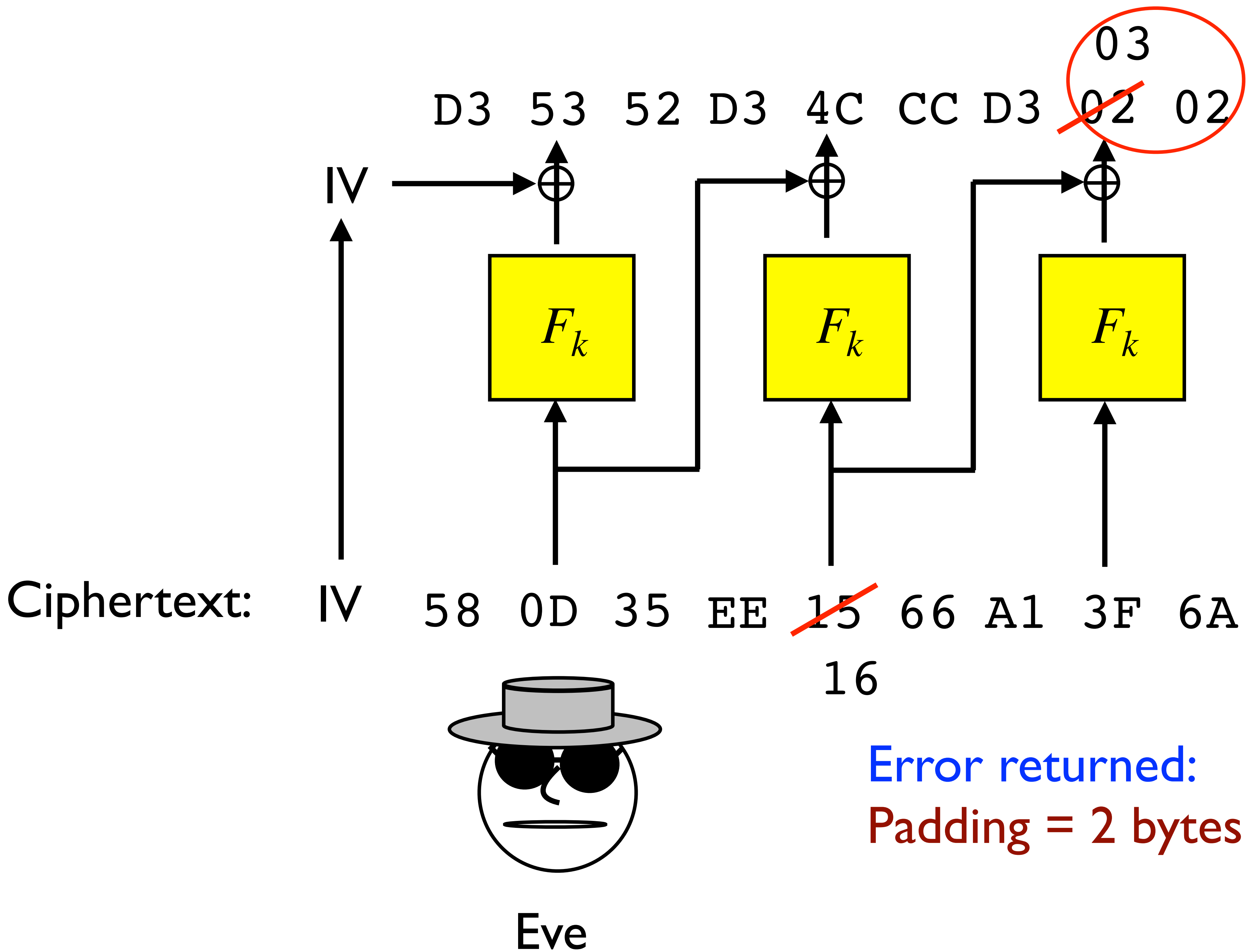
If Eve can alter sent messages, she can learn the padding length.



This class is being recorded

Learning # of Padded Bits

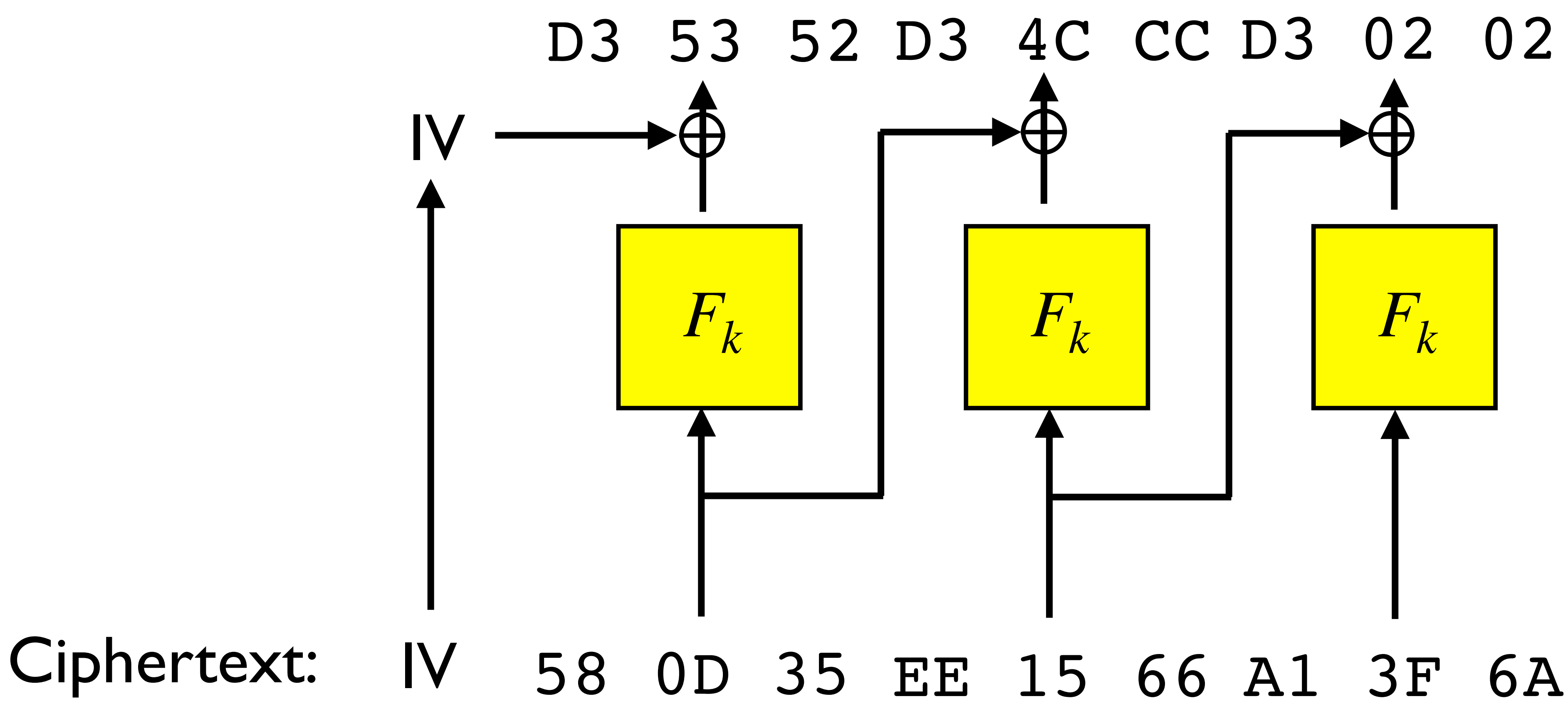
If Eve can alter sent messages, she can learn the padding length.

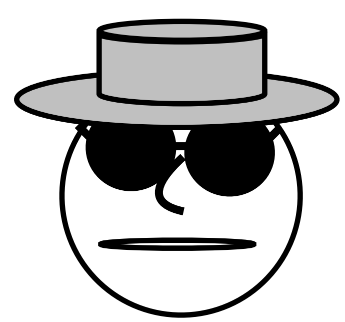


Error returned:
Padding = 2 bytes

Learning Message Bits

Moreover, Eve can break the encryption.



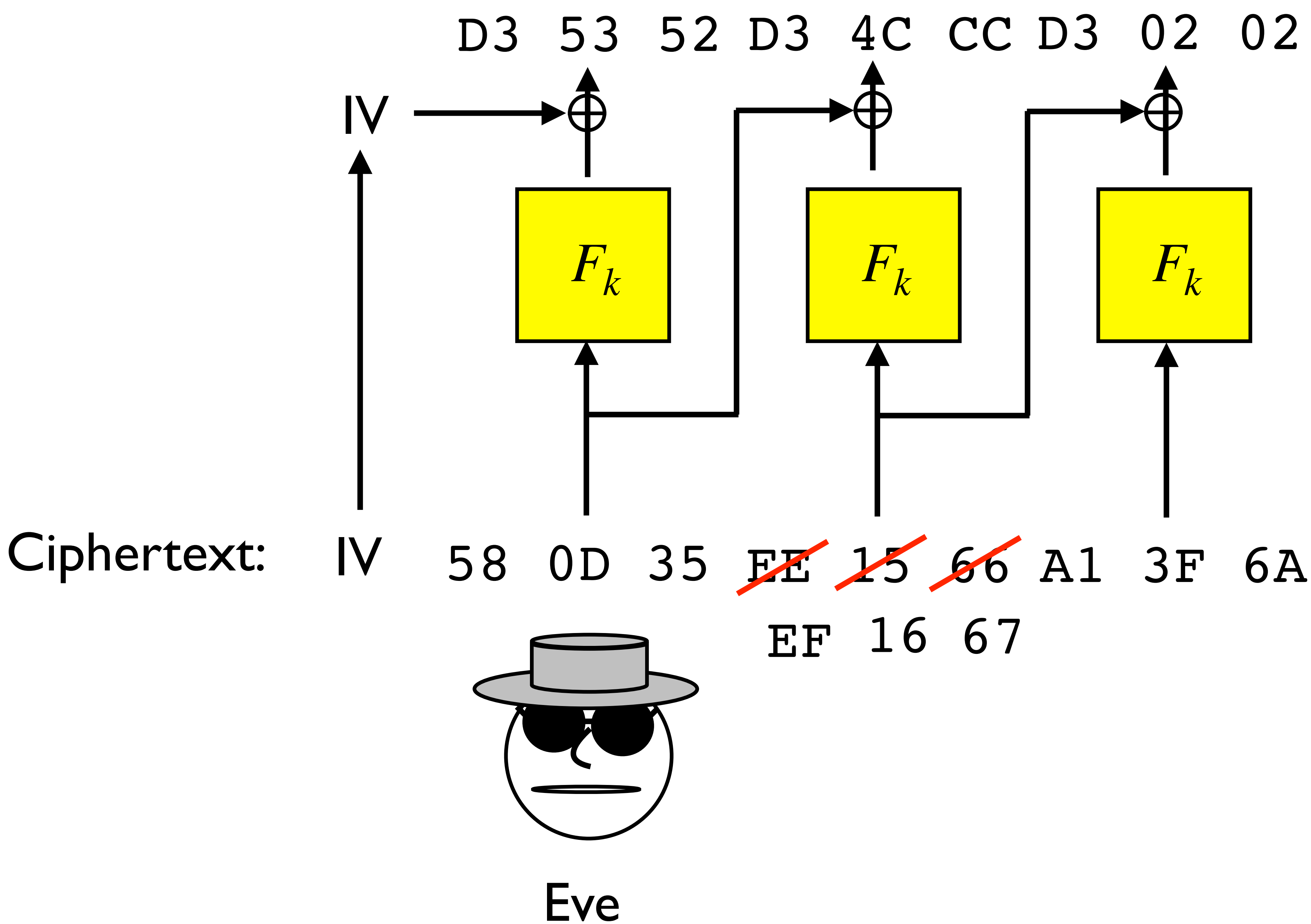


Eve

This class is being recorded

Learning Message Bits

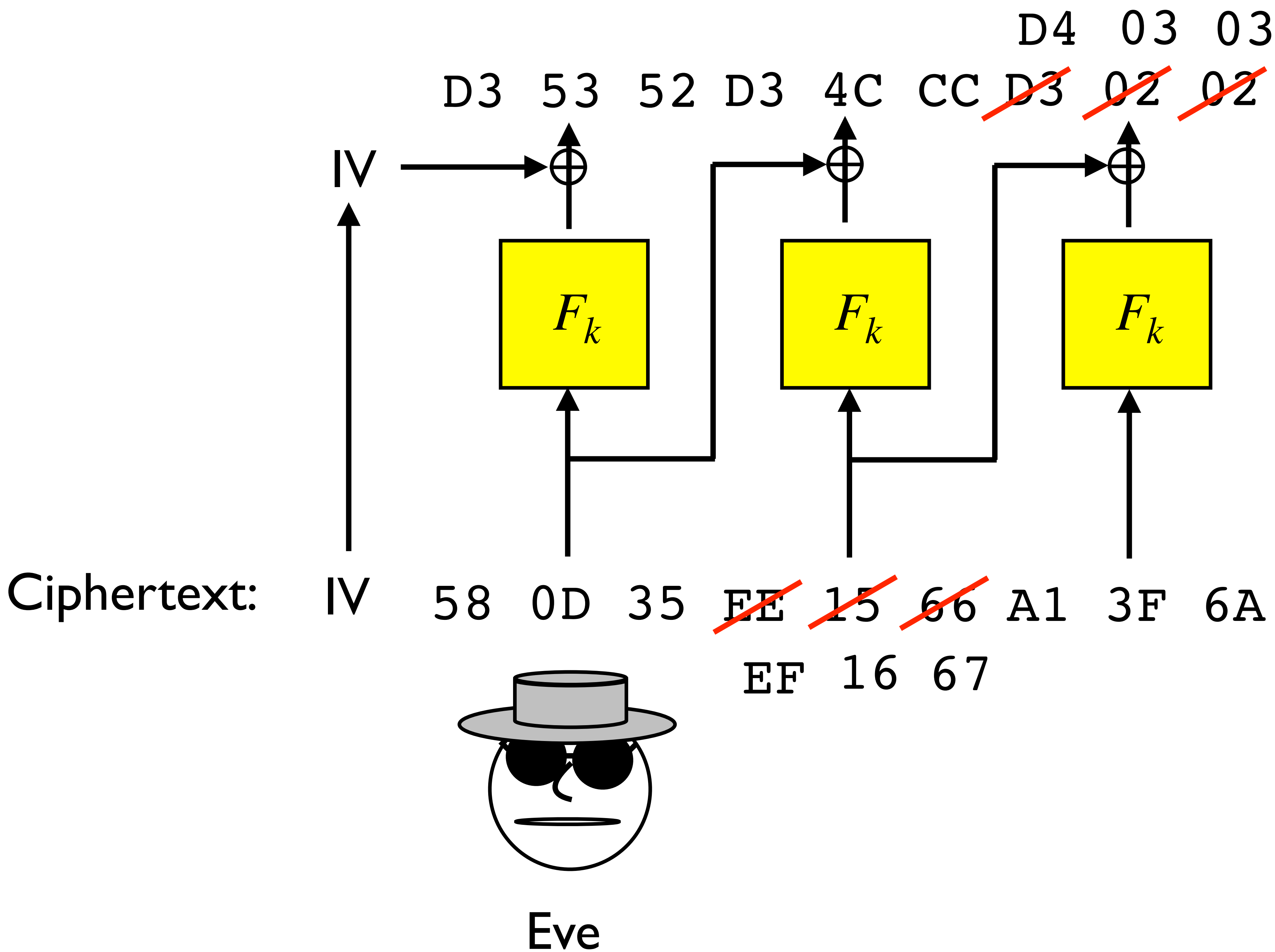
Moreover, Eve can break the encryption.



This class is being recorded

Learning Message Bits

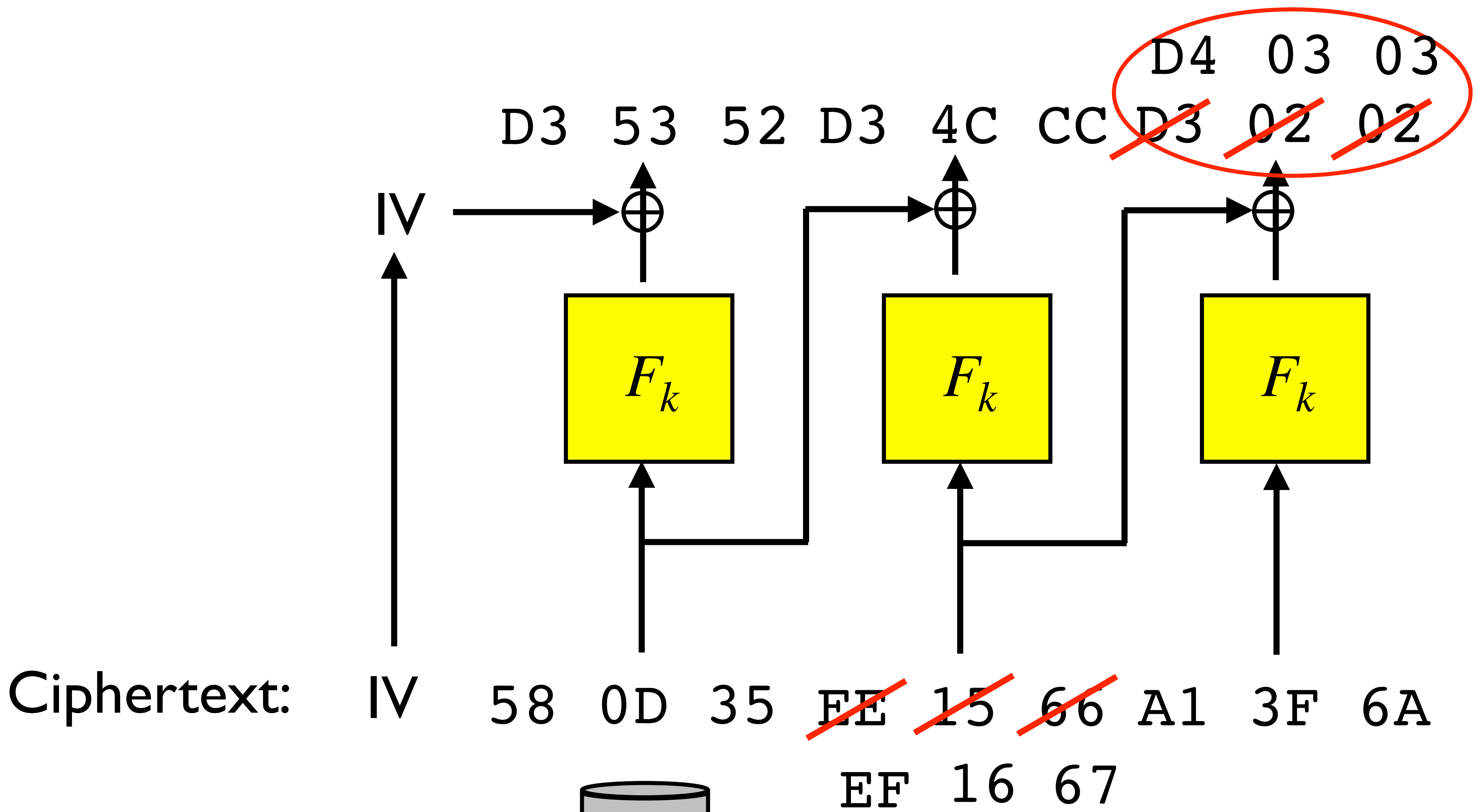
Moreover, Eve can break the encryption.

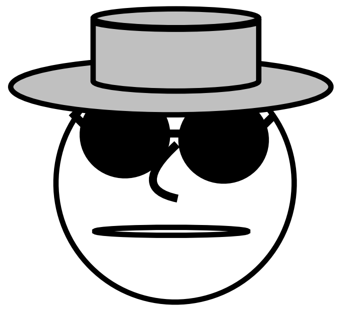


This class is being recorded

Learning Message Bits

Moreover, Eve can break the encryption.

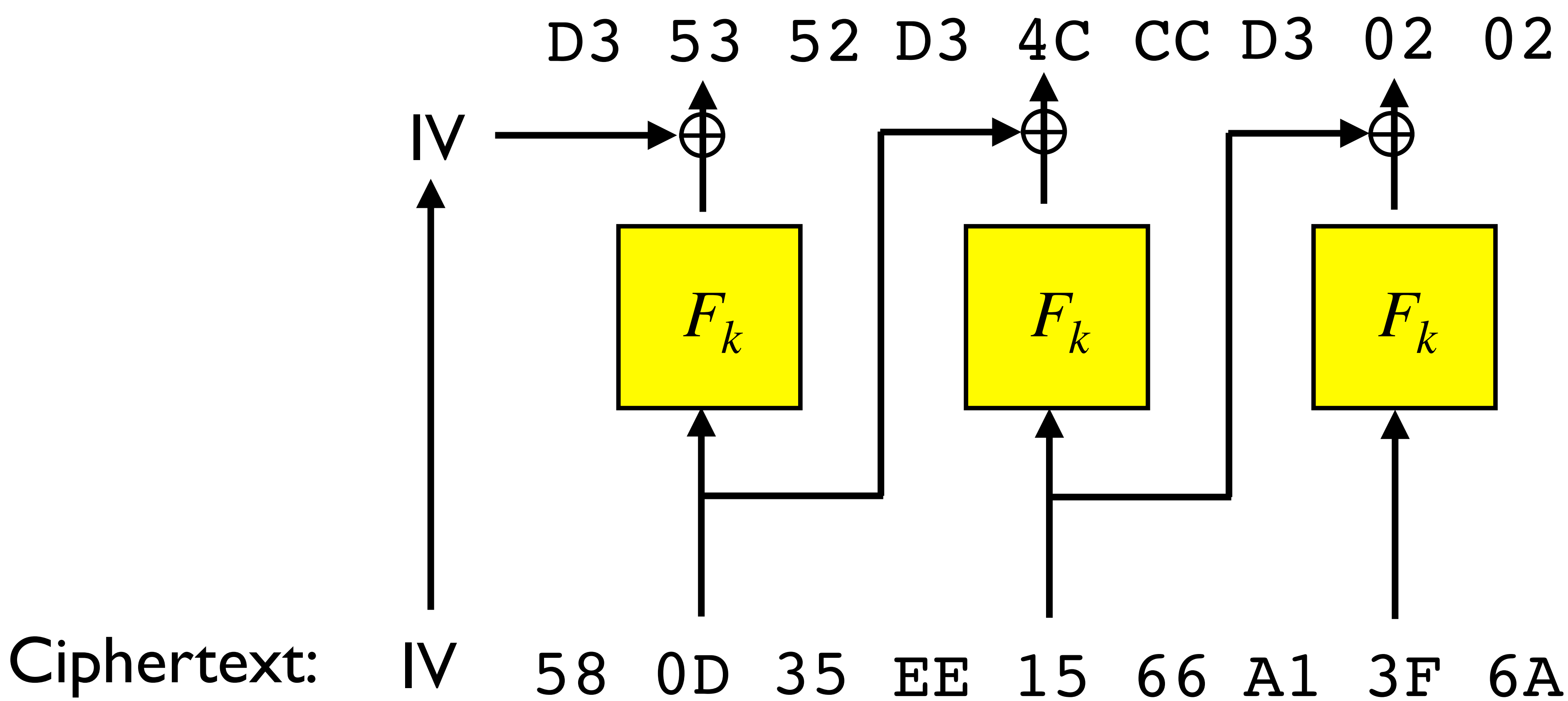


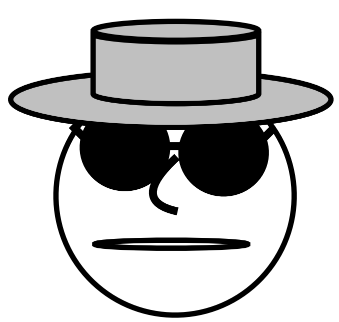


Error returned:
Last byte is not 02

Learning Message Bits

Moreover, Eve can break the encryption.



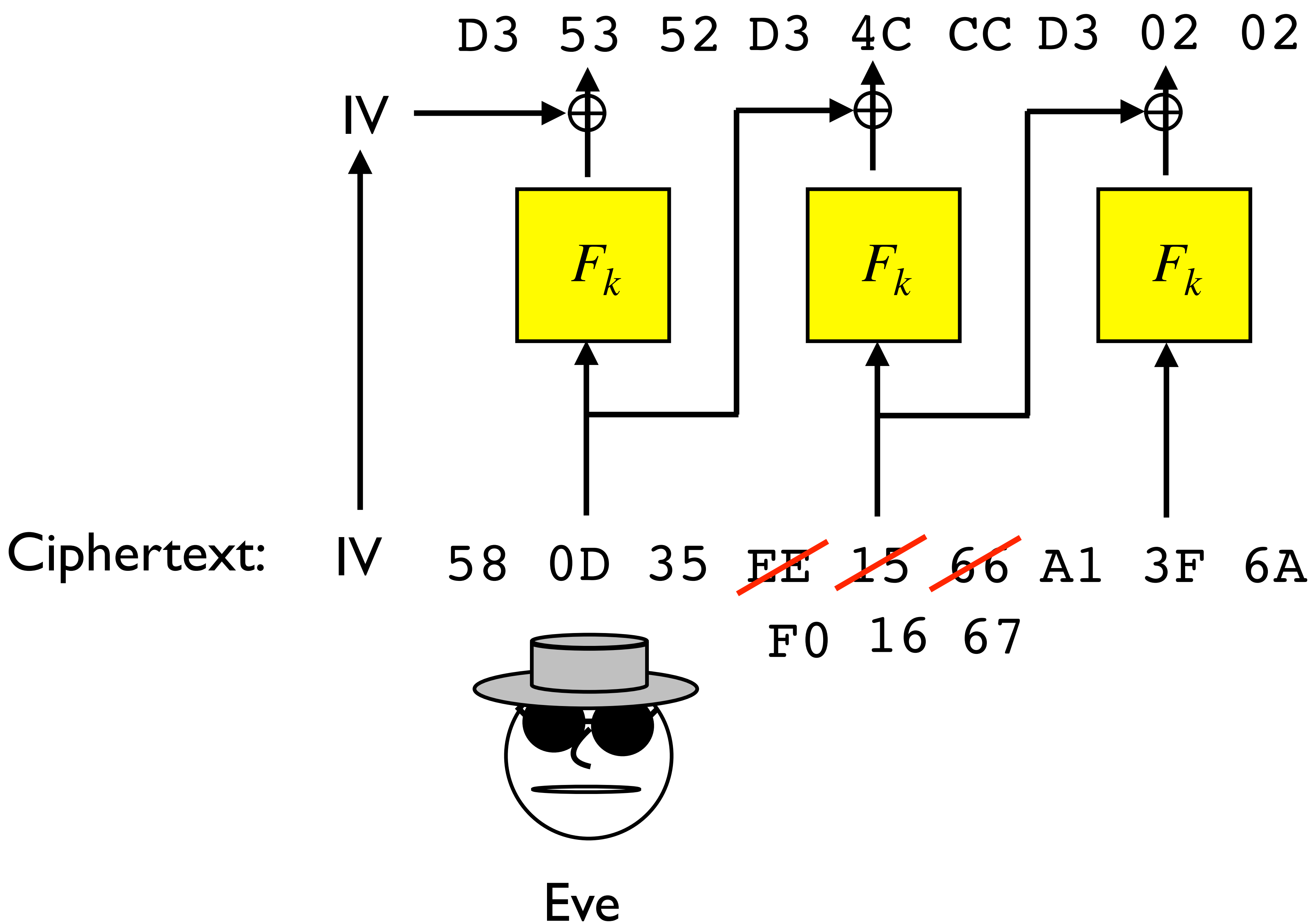


Eve

This class is being recorded

Learning Message Bits

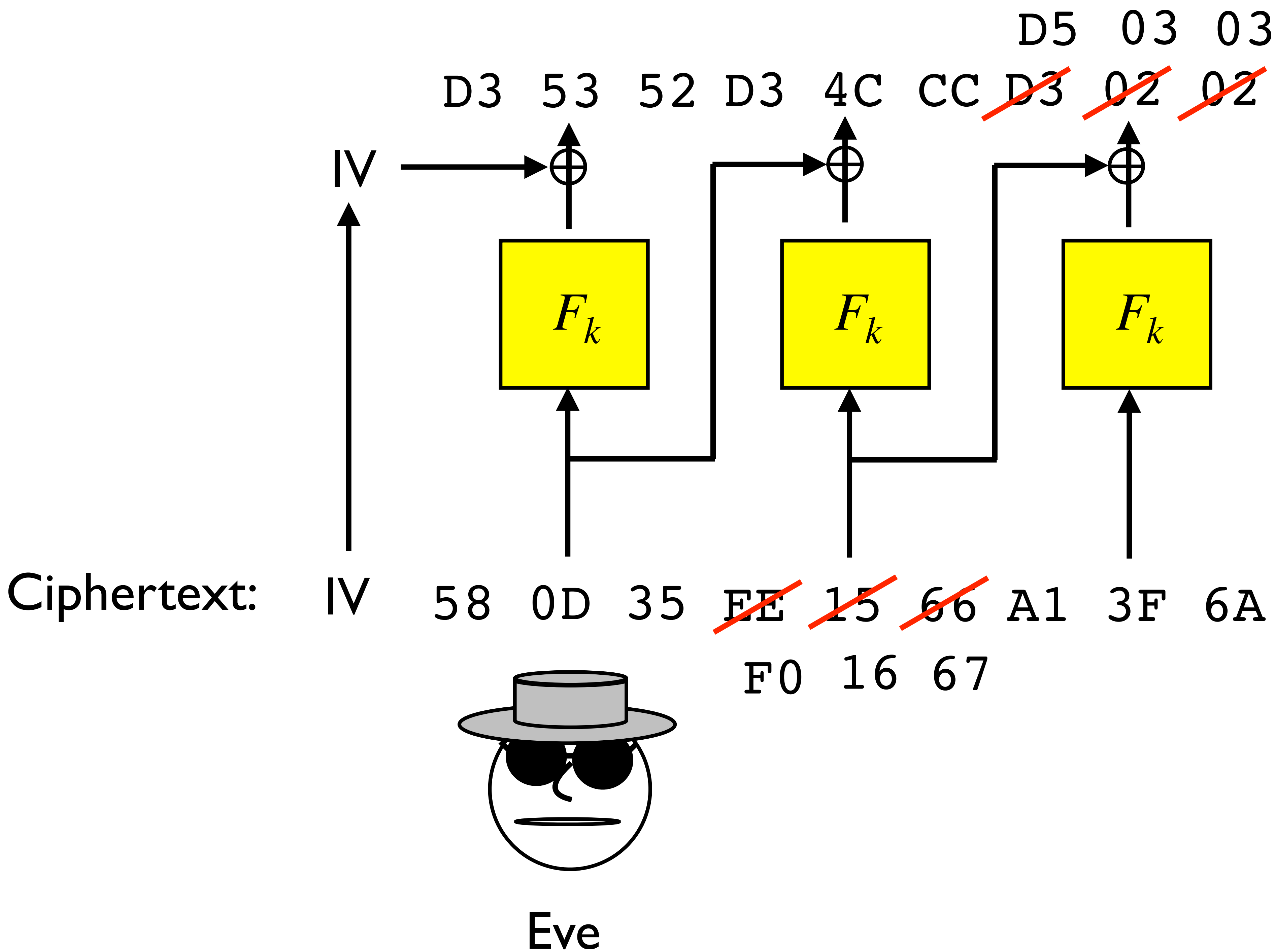
Moreover, Eve can break the encryption.



This class is being recorded

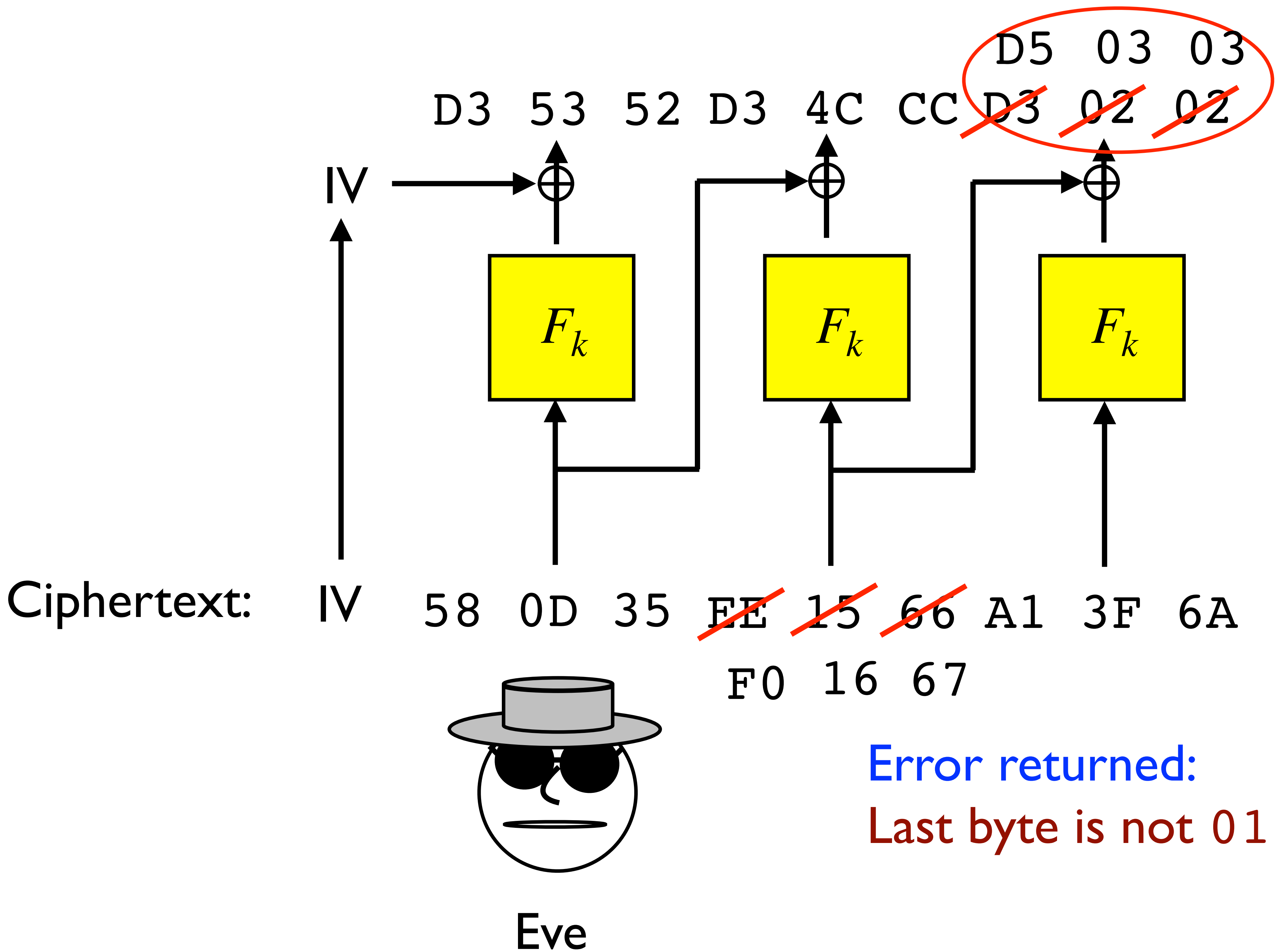
Learning Message Bits

Moreover, Eve can break the encryption.



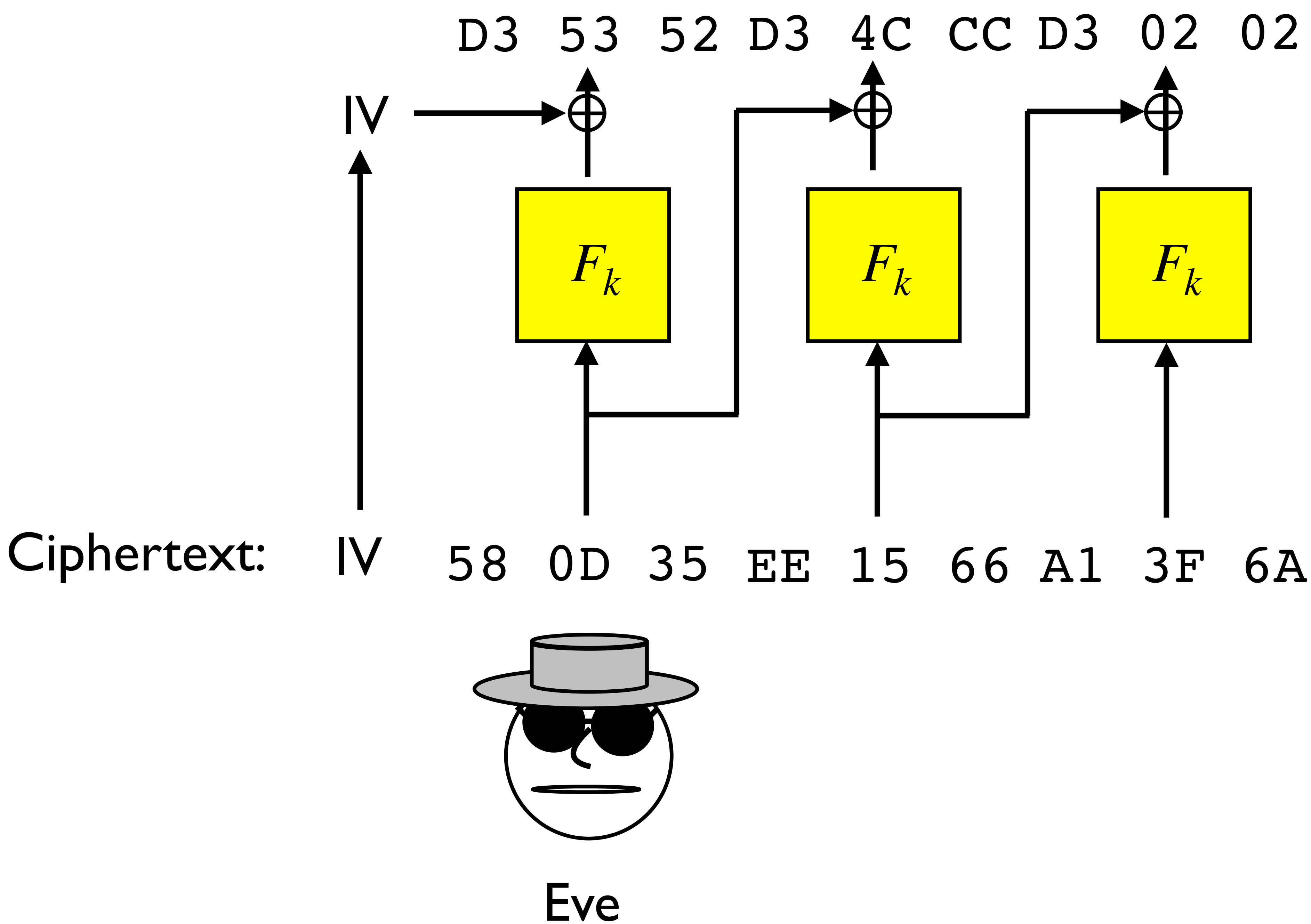
Learning Message Bits

Moreover, Eve can break the encryption.



Learning Message Bits

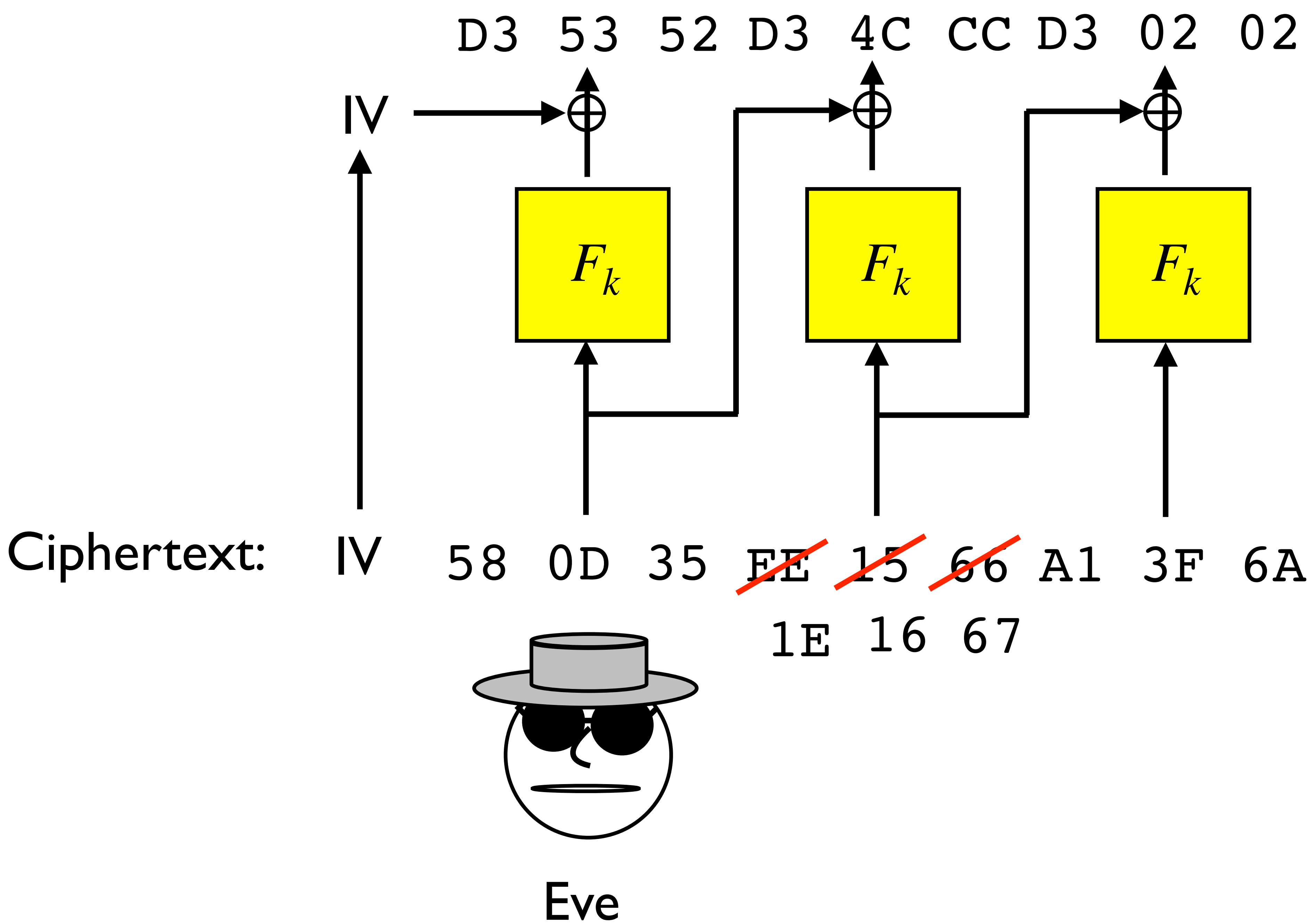
Moreover, Eve can break the encryption.



This class is being recorded

Learning Message Bits

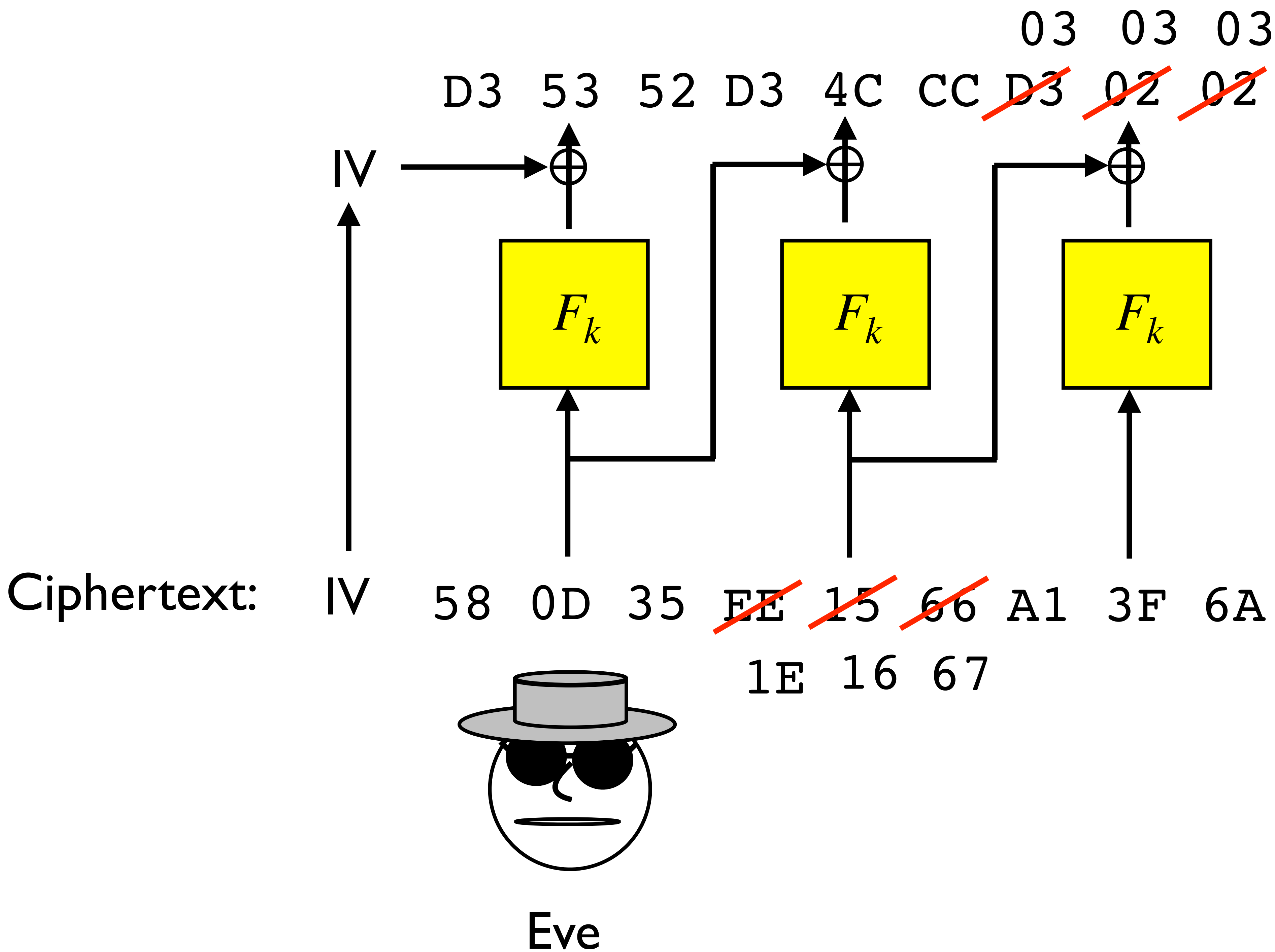
Moreover, Eve can break the encryption.



This class is being recorded

Learning Message Bits

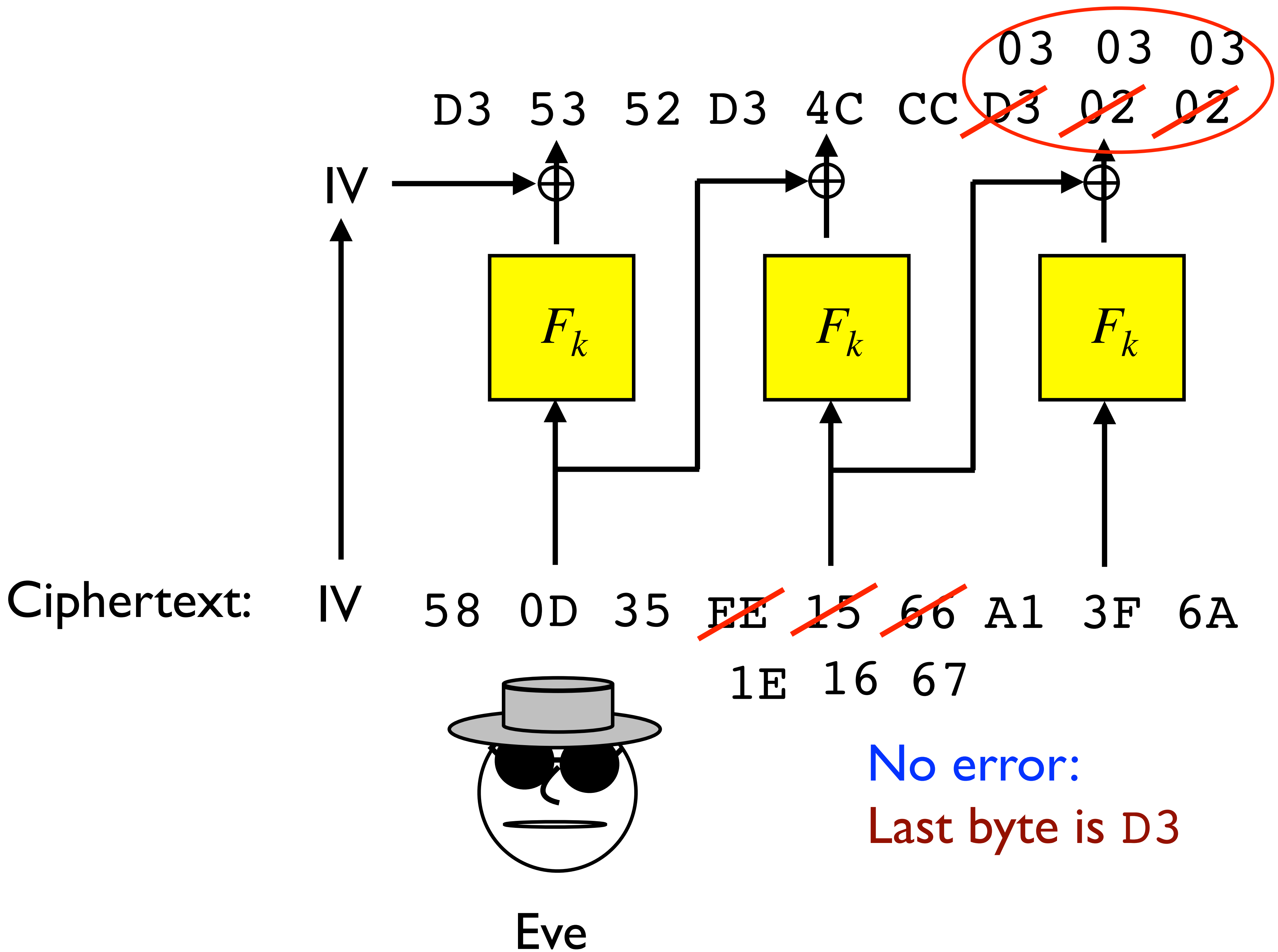
Moreover, Eve can break the encryption.



This class is being recorded

Learning Message Bits

Moreover, Eve can break the encryption.



This class is being recorded

Learning the Full Message

If the block contains N bytes, Eve needs at most N attempts to learn the amount of padding.

After that, it takes at most 256 attempts for Eve to learn one byte of the message.

> If the message consists of L bytes, Eve learns the whole message after sending at most N + 256 L trial messages.

Compare this to trying every possible key for a 128-bit key.

This kind of attack has been demonstrated in real systems.

> Not returning an error message can help, but you have to be sure that the error information is not available through a side-channel attack (e.g., timing attack).

What Threat Model to Use?

Bob's behavior may well reveal partial information about the messages he decrypts. How can we define the types of information that Eve might learn?

Recall: When defining threat models, we want to be conservative and give Eve as much power as we can. Otherwise, something we have overlooked might form the basis for an attack.

Therefore: We will give Eve access to decryptions of any ciphertext except the one we are trying to hide.

In particular, Eve will get access to a Dec oracle, but she is not allowed to query it on the particular ciphertext she is trying to decode.

This is a chosen ciphertext attack (CCA).

An Example of the Threat Model

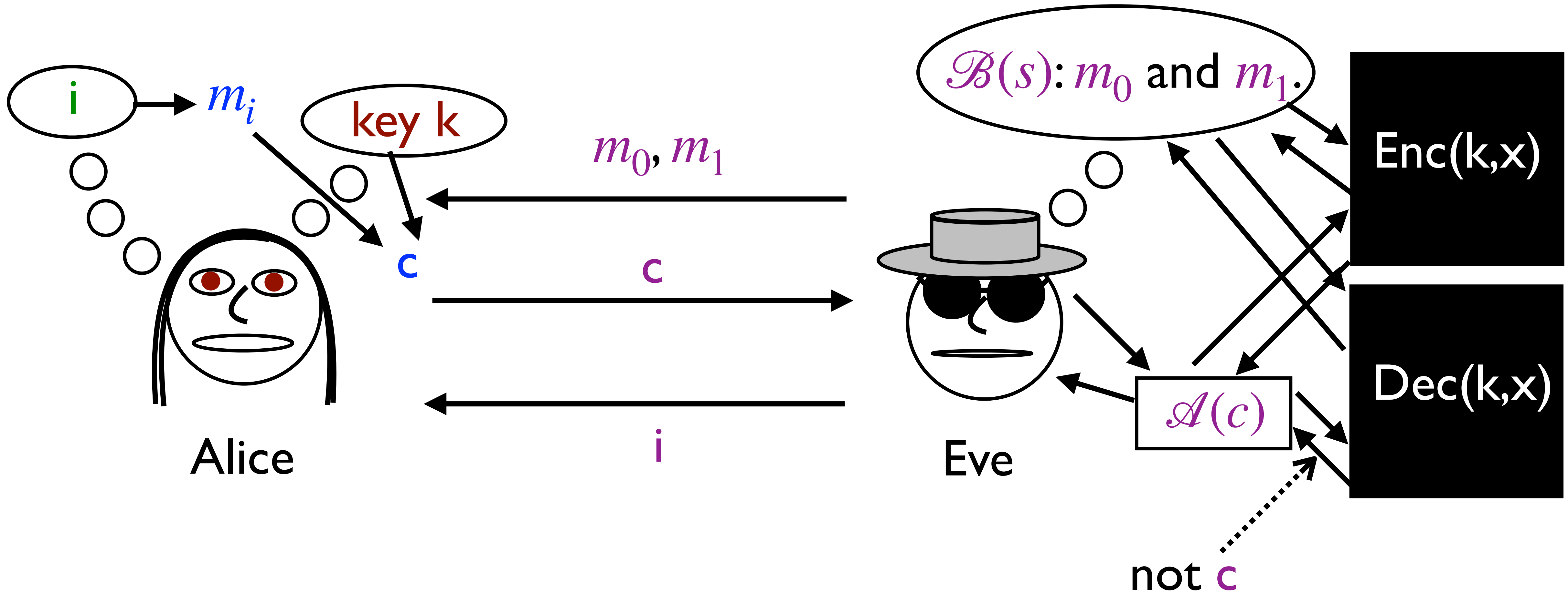
We are giving Eve access to a Dec oracle to be conservative, but occasionally it might be close to realistic.

Example: Suppose we have a public key protocol. Eve intercepts an encrypted email with ciphertext c sent from Alice. The From: line is in a known location and format; Eve knows the message is from Alice. She may therefore be able to alter that part of the message to give the ciphertext c' corresponding to Alice's message but sent from Eve.

Bob decrypts c' and reads the message. Bob has no way of knowing that Eve wasn't the original sender. Then suppose Bob replies to Eve and his reply email automatically includes the original message. Bob has unwittingly decrypted the ciphertext c' for Eve.

CCA Security Game

The game to define security is very similar to CPA security except now we allow Eve to have access to the Dec oracle.



This class is being recorded

CCA Security Definition

Definition: (Enc, Dec) with security parameter s is CCA-secure if, for any pair of messages m_0 and m_1 chosen by the adversary (using $\mathscr{B}(s)$ and oracle access to Enc(k,x) and Dec(k,x)) and for any efficient attack $\mathscr{A}(c)$ (also with oracle access to Enc(k,x) and Dec(k,x) except that $\mathscr{A}(c)$ may not query Dec(k,c) on input c)

$$|\Pr_k(\mathscr{A}(Enc(k,m_0)) = 1) - \Pr_k(\mathscr{A}(Enc(k,m_1)) = 1)| \leq \epsilon(s)$$

for negligible $\epsilon(s)$ and probability taken over k and randomness of the attack and encryptions.

This is the private key definition. For a public key protocol, we give Eve access to the public key so she does not need an Enc(k,x) oracle. She still has access to the Dec(k,x) oracle.

CCA Security vs. Malleability

Non-malleability and CCA security are distinct but related notions.

> CCA security implies non-malleability: With a CCA attack, given ciphertext c, Eve can alter it to c' and get the decryption m'. If the protocol were malleable, then Eve would know m' = f(m) and could deduce partial or full information about m.

How to Achieve CCA Security

One way we might hope to achieve CCA security is to make sure we can detect any alteration to the ciphertext.

We might expect to achieve this using MACs.

If we achieve this, then Dec can simply reply "invalid" to any ciphertext Eve submits to it and she gets no information from Dec.

This is actually a separate notion from CCA security called unforgeability.

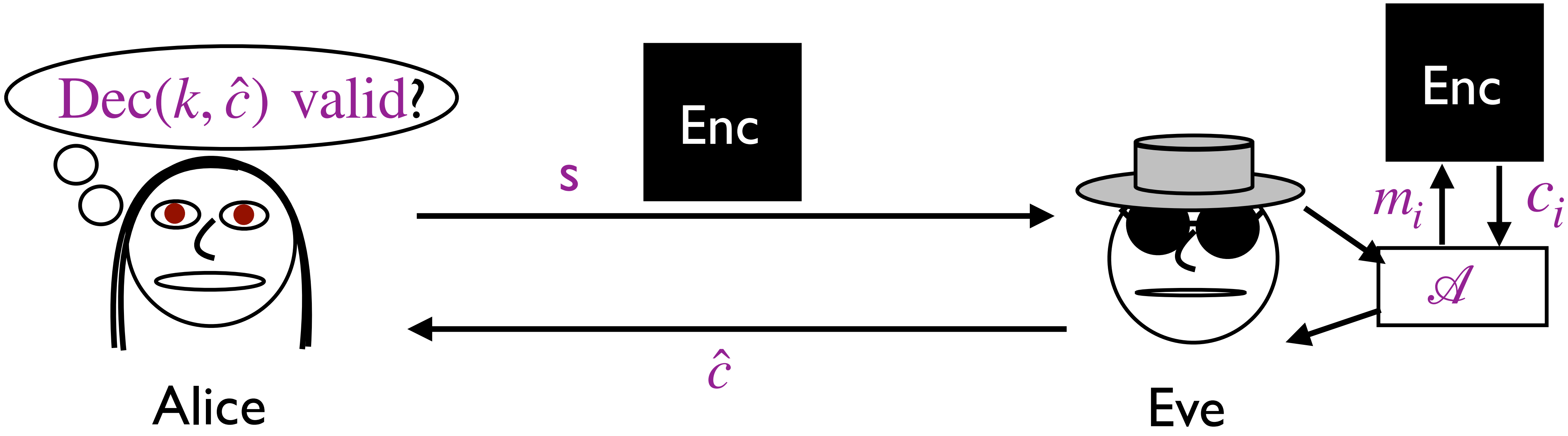
Then unforgeability plus CCA security gives authenticated encryption, which is stronger than either in principle. However, in practice, the most straightforward way to achieve CCA security is via authenticated encryption.

Unforgeability

Definition: A encryption protocol (Gen, Enc, Dec) with security parameter s is unforgeable if, for any polynomial-time attack $\mathscr{A}$ with oracle access to $\text{Enc}(k, m)$, where $\mathscr{A}$ outputs $\hat{c}$ with $\hat{m} = \text{Dec}(\hat{c})$ such that $\mathscr{A}$ never queried the oracle for $m = \hat{m}$,

$$\Pr(\hat{c} \text{ is valid}) \leq \epsilon(s)$$

where $\epsilon(s)$ is a negligible function and the probability is averaged over k generated by Gen and the randomness used in any of the functions.



This class is being recorded

Authenticated Encryption

Definition: A private-key encryption scheme provides authenticated encryption if it is CCA-secure and unforgeable.

To achieve this, we will want to combine an encryption protocol with a MAC protocol.

The MAC will provide the unforgeability. Since Eve cannot successfully find ciphertexts that correspond to messages other than ones she got from the Enc oracle or the challenge, the Dec oracle does her little good and if the original encryption protocol is CPA secure, we would expect to get CCA security from the combination encryption + MAC.

We will still need to be careful.

Encrypt and Authenticate

Let (Enc, Dec) be a CPA-secure encryption scheme and (Mac, Vrfy) be a secure MAC. Consider the following encryption protocol:

> Enc': Given message m and keys k and k', the ciphertext is (Enc(k,m), Mac(k',m)).

> Dec': Given ciphertext (c,t) and keys (k,k'), decrypt to m = Dec(k,c) but output Invalid if $\mathrm{Vrfy}(k', m, t)$ is invalid. Otherwise output m.

Vote: Does this always work? (Yes/No/Unknown)

Encrypt and Authenticate

Let (Enc, Dec) be a CPA-secure encryption scheme and (Mac, Vrfy) be a secure MAC. Consider the following encryption protocol:

> Enc': Given message m and keys k and k', the ciphertext is (Enc(k,m), Mac(k',m)).
>
> Dec': Given ciphertext (c,t) and keys (k,k'), decrypt to m = Dec(k,c) but output Invalid if $\mathrm{Vrfy}(k', m, t)$ is invalid. Otherwise output m.

Vote: Does this always work? (Yes/No/Unknown)

Answer: No.

> The tag Mac(k',m) could contain information about m. If Mac(k',m) is deterministic, then it is easy to tell if the same message is repeated twice.

This class is being recorded

Authenticate Then Encrypt

Let (Enc, Dec) be a CPA-secure encryption scheme and (Mac, Vrfy) be a secure MAC. Consider the following encryption protocol:

> Enc': Given message m and keys k and k', the ciphertext is Enc(k,(m, Mac(k',m)).
>
> Dec': Given ciphertext c and keys (k,k'), decrypt to (m,t) = Dec(k,c) and output Invalid if $\mathrm{Vrfy}(k', m, t)$ is invalid. Otherwise output m.

Vote: Does this always work? (Yes/No/Unknown)

Authenticate Then Encrypt

Let (Enc, Dec) be a CPA-secure encryption scheme and (Mac, Vrfy) be a secure MAC. Consider the following encryption protocol:

> Enc': Given message m and keys k and k', the ciphertext is Enc(k,(m, Mac(k',m)).
>
> Dec': Given ciphertext c and keys (k,k'), decrypt to (m,t) = Dec(k,c) and output Invalid if $\mathrm{Vrfy}(k', m, t)$ is invalid. Otherwise output m.

Vote: Does this always work? (Yes/No/Unknown)

Answer: No, not reliably.

> For instance, the padding is analyzed first and if the system returns an error for bad padding (deliberately or through a side channel), the padding oracle attack works still.

Encrypt Then Authenticate

Let (Enc, Dec) be a CPA-secure encryption scheme and (Mac, Vrfy) be a secure MAC. Consider the following encryption protocol:

> Enc': Given message m and keys k and k', the ciphertext is (Enc(k,m), Mac(k',Enc(k,m))).

> Dec': Given ciphertext (c,t) and keys (k,k'), output Invalid if $\mathrm{Vrfy}(k', c, t)$ is invalid. Otherwise decrypt c to m = Dec(k,c) and output m.

Vote: Does this always work? (Yes/No/Unknown)

Encrypt Then Authenticate

Let (Enc, Dec) be a CPA-secure encryption scheme and (Mac, Vrfy) be a secure MAC. Consider the following encryption protocol:

> Enc': Given message m and keys k and k', the ciphertext is (Enc(k,m), Mac(k',Enc(k,m))).

> Dec': Given ciphertext (c,t) and keys (k,k'), output Invalid if $\mathrm{Vrfy}(k', c, t)$ is invalid. Otherwise decrypt c to m = Dec(k,c) and output m.

Vote: Does this always work? (Yes/No/Unknown)

Answer: Yes.

> This combination finally achieves what we wanted: That Eve cannot successfully submit a new ciphertext to the oracle.

Authenticated Encryption Achieved

Theorem: If (Enc, Dec) is a CPA-secure encryption scheme and (Mac, Vrfy) is a secure MAC, then the following encryption scheme is an authenticated encryption protocol:

> Enc': Given message m and keys k and k', the ciphertext is (Enc(k,m), Mac(k',Enc(k,m))).
>
> Dec': Given ciphertext (c,t) and keys (k,k'), output Invalid if $\mathrm{Vrfy}(k', c, t)$ is invalid. Otherwise decrypt c to m = Dec(k,c) and output m.

There are standardized authenticated encryption protocols that don't follow exactly this template but use specific properties of the construction and component protocols to achieve security.