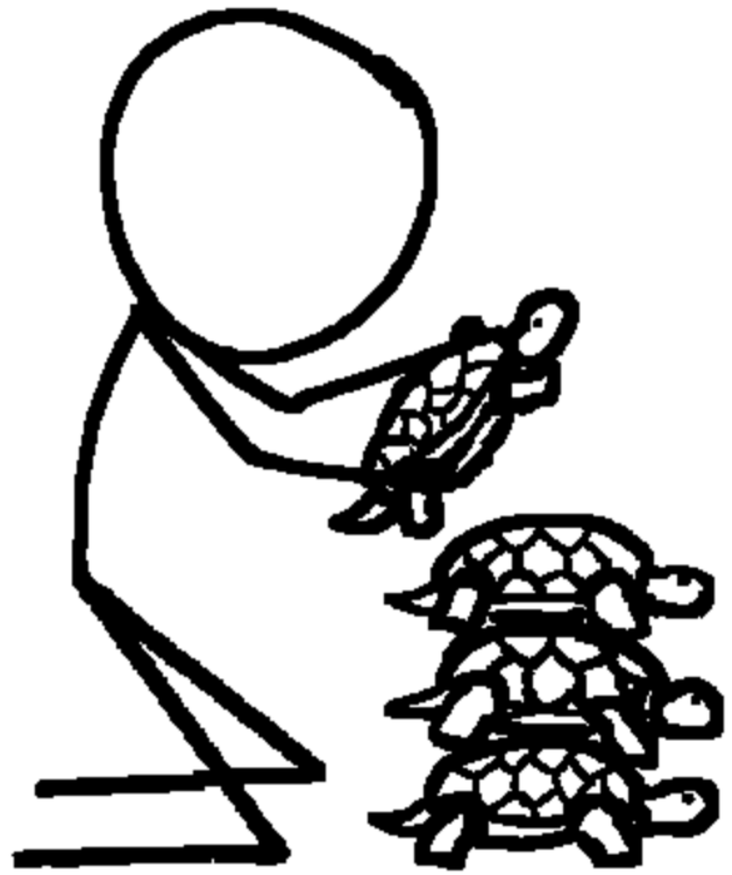
Secure channels

Building real world crypto systems

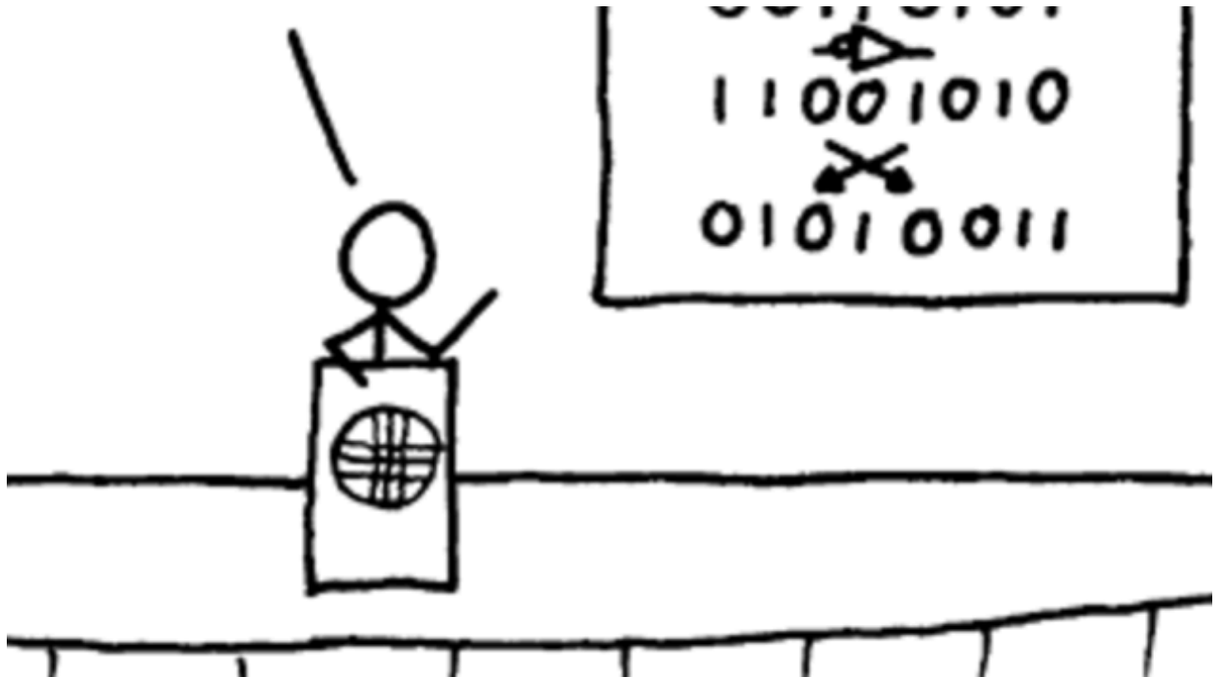
Sander Demeester

Saying ‘Hi’

- Some fun crypto theory
  - What are secure channels
  - Security notions
- Constructing secure channels
- Protocol design
- Real world protocols
- Conclude



Some fun theory!



What are secure channels?

Conceptually we want to guarantee the confidentiality and integrity of data traveling over untrusted networks

This conceptual goal is accomplished by something called “a secure channel”

[Canetti and Krawczyk] define a secure channel as a **channel protocol** that is both a secure **authentication protocol** and a **secure encryption protocol**

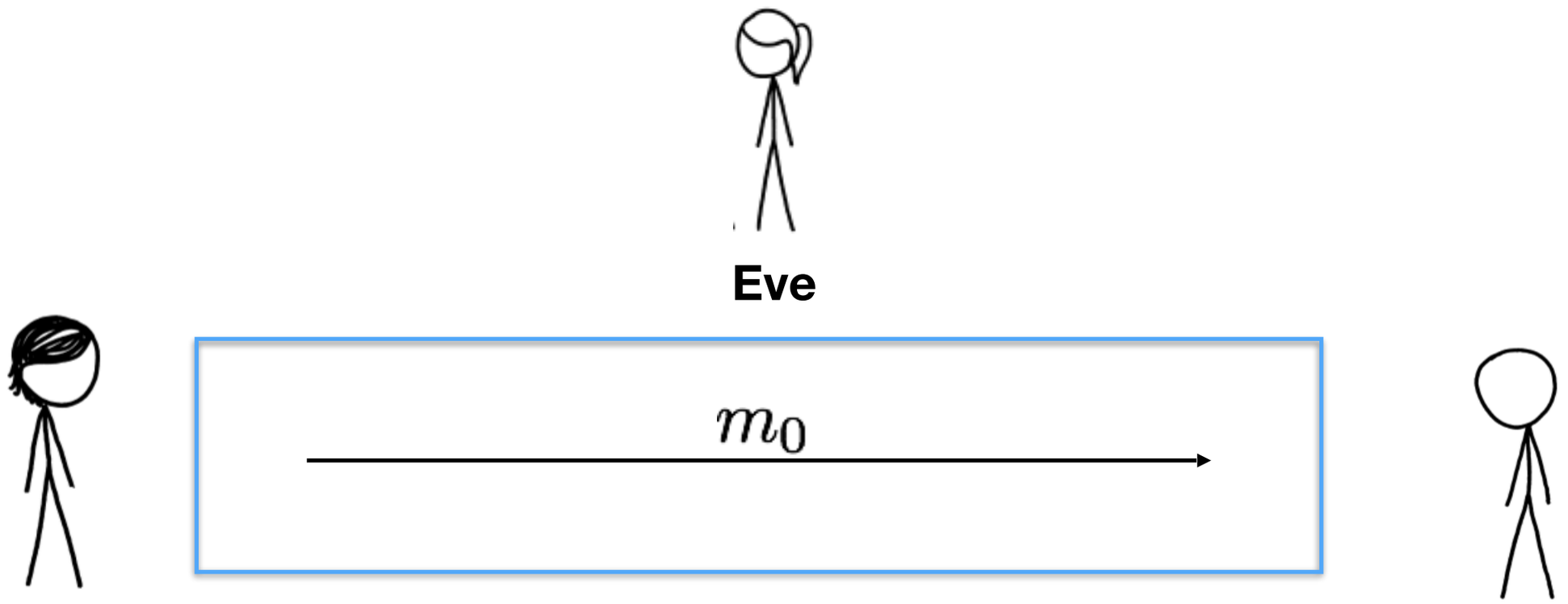
Often this channel protocol is **session-based message driven.**

Secure channel objectives

The concept is achieved by building a secure channel, we want to guarantee certain properties of the messages being exchanged over this channel

- **Confidentiality / data secrecy**
- **Integrity: Making sure data has not changed while transferring**
- **Authenticity: Making sure the origin is bound to a know entity**

- **Application protocol multiplexing**
  - Allowing multiple higher level protocols to be multiplexed over the same record layer
- **Being able to communicate alerts**

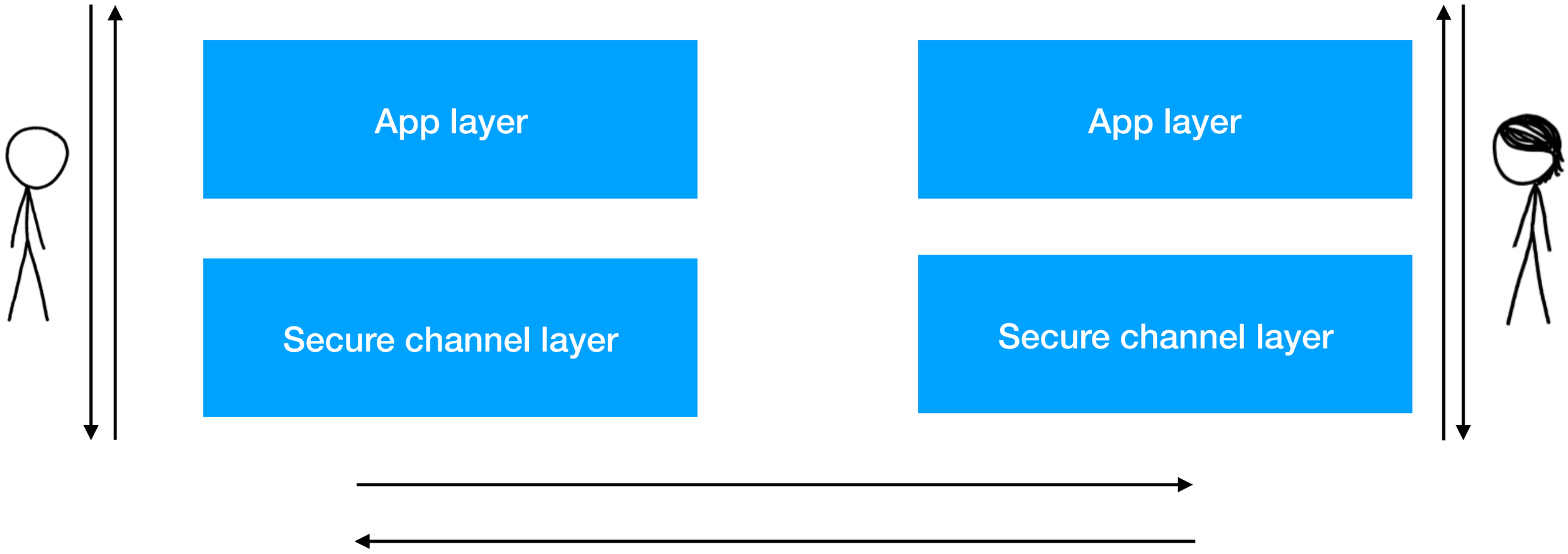


- **Confidentiality: data secrecy (m_0 is only read by Bob)**
- **Integrity: Making sure data has not changed while transferring (m_0 is not changed)**
- **Authenticity: Making sure the origin is bound to a know entity (m_0 is sent by Alice)**

Application protocol multiplexing

Being able to communicate alerts

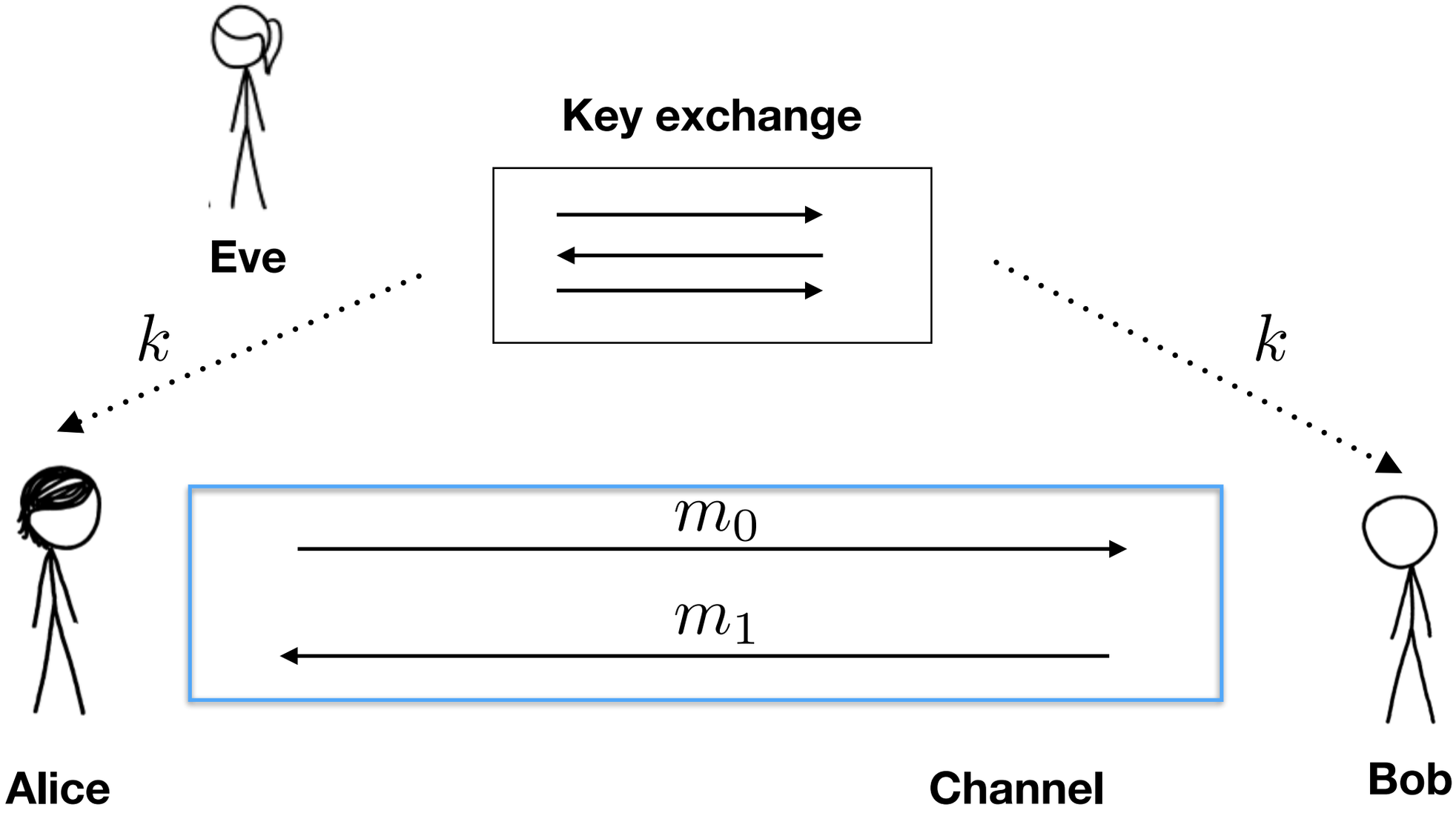
Layers of abstraction



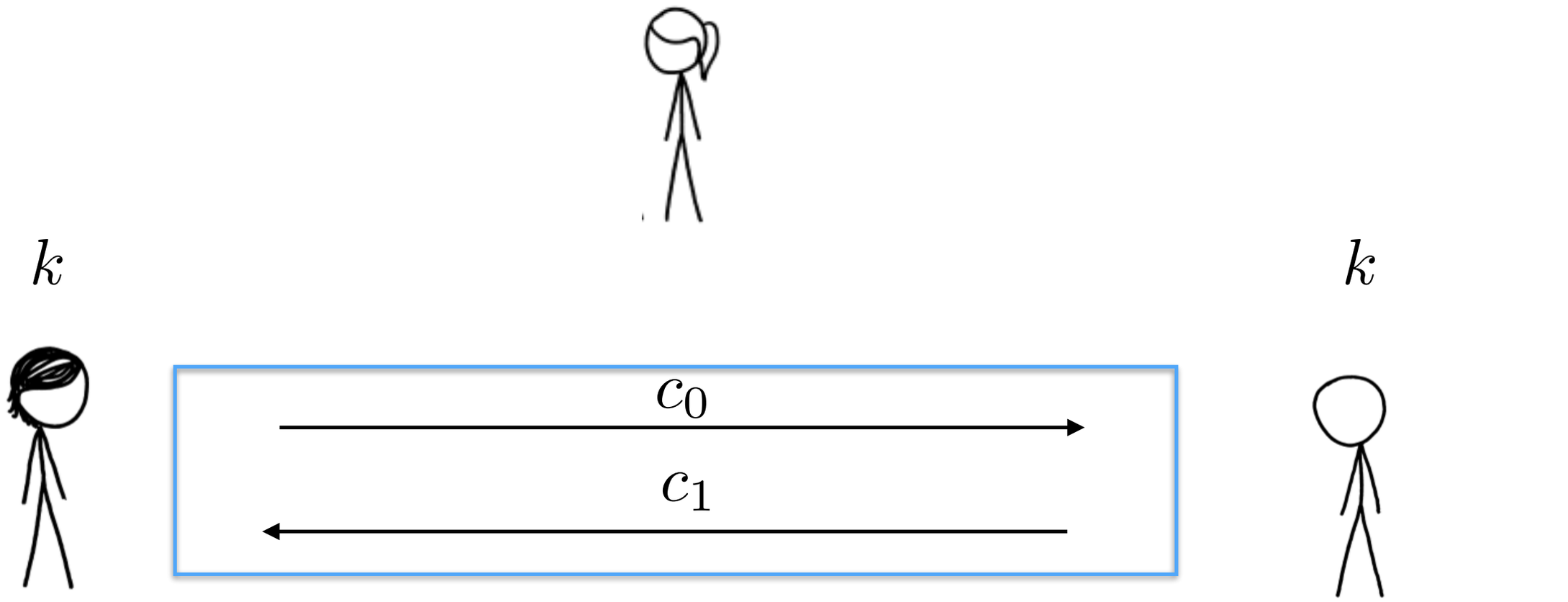
- Higher level protocols can be multiplexed over the same channel

Security notions

What do we mean when we say “secure”?


Key exchange
Eve
k
k
m_0
m_1
Alice
Channel
Bob

Setup

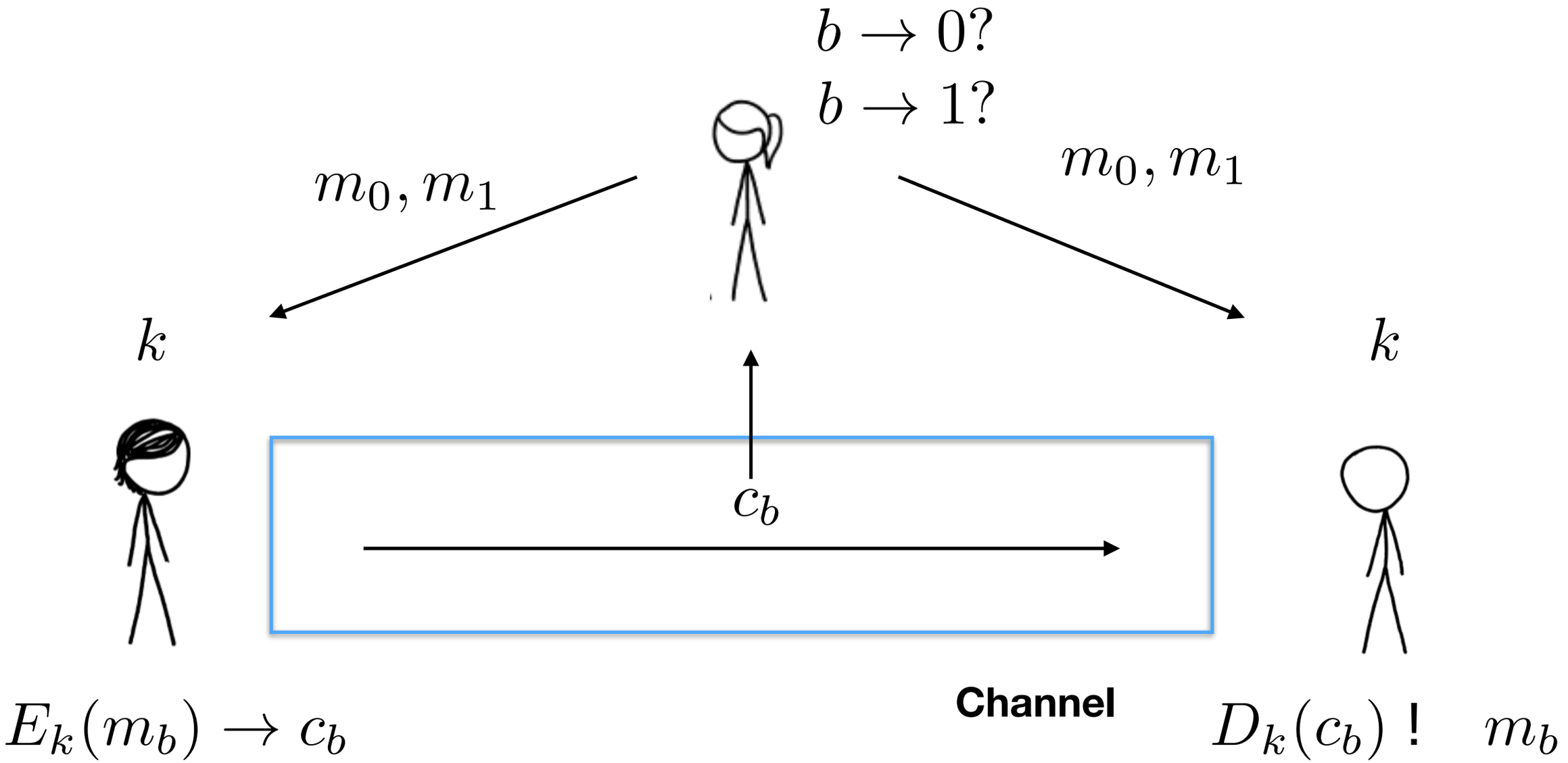


$E_k(m_0) \rightarrow c_0$
$D_k(c_1) \rightarrow m_1$

Channel

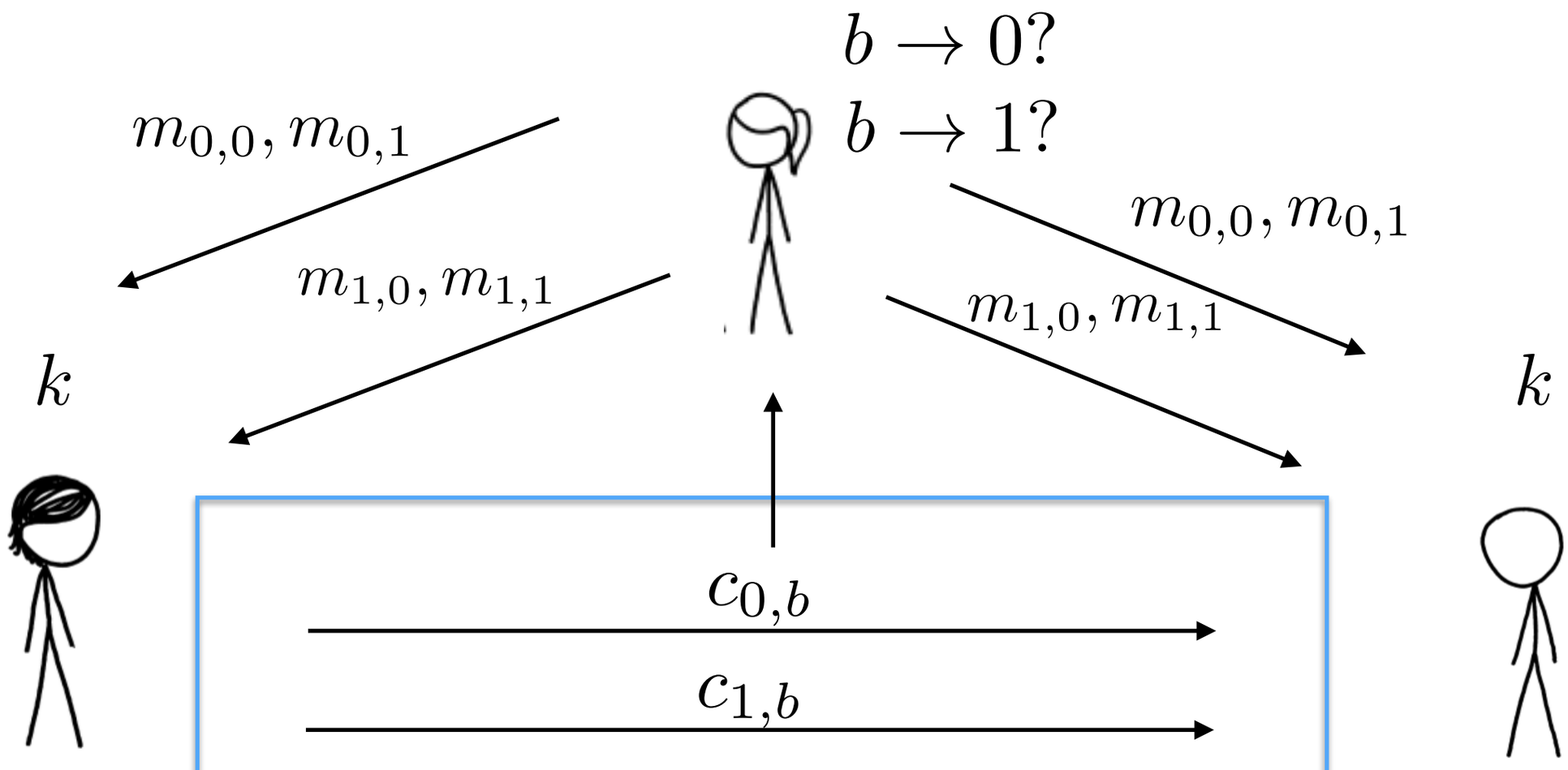
$D_k(c_0) \rightarrow m_0$
$E_k(m_1) \rightarrow c_1$

What do we mean with “confidentiality”



(Single round)

What do we mean with “confidentiality” when reusing a key



$E_k(m_{0,b}) \to c_{0,b}$

$E_k(m_{1,b})$! $c_{1,b}$

$b \xrightarrow{r} \{1, 0\}$

$D_k(c_{0,b}) \to m_{0,b}$

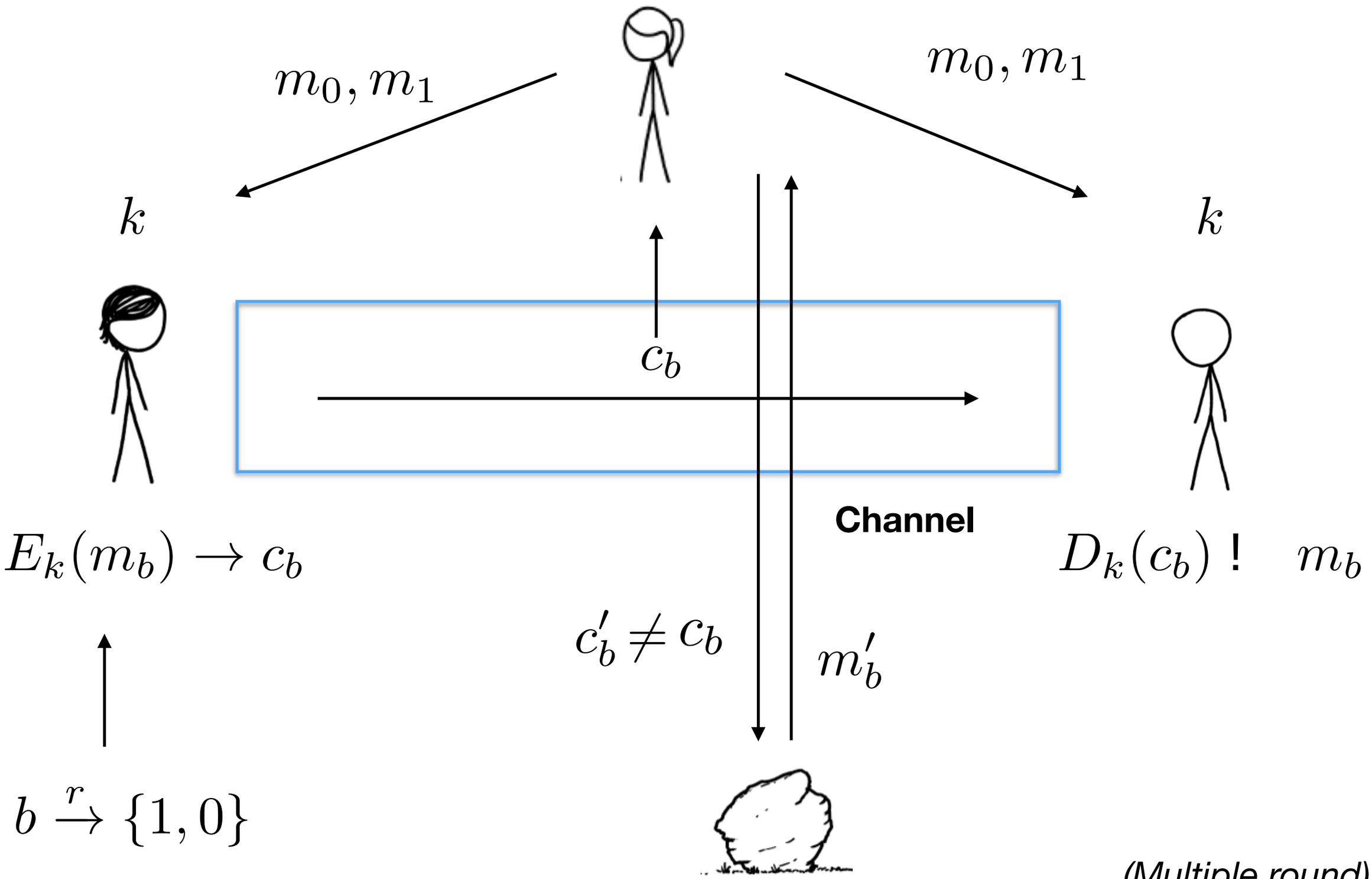
$D_k(c_{1,b}) \to m_{1,b}$

(Multiple round)

What do we mean with “authenticated message”?

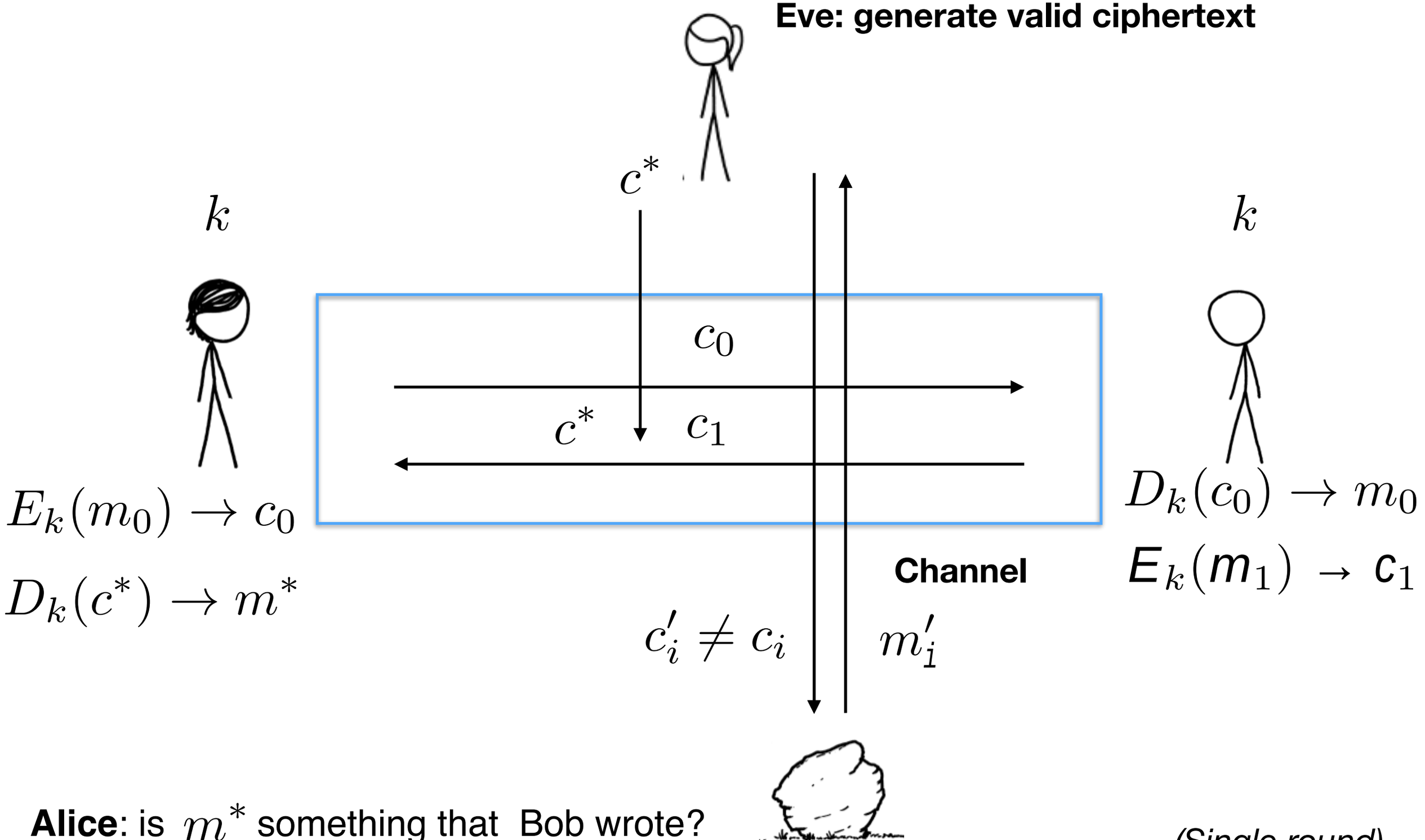
$b \to 0?$

$b \to 1?$



(Multiple round)

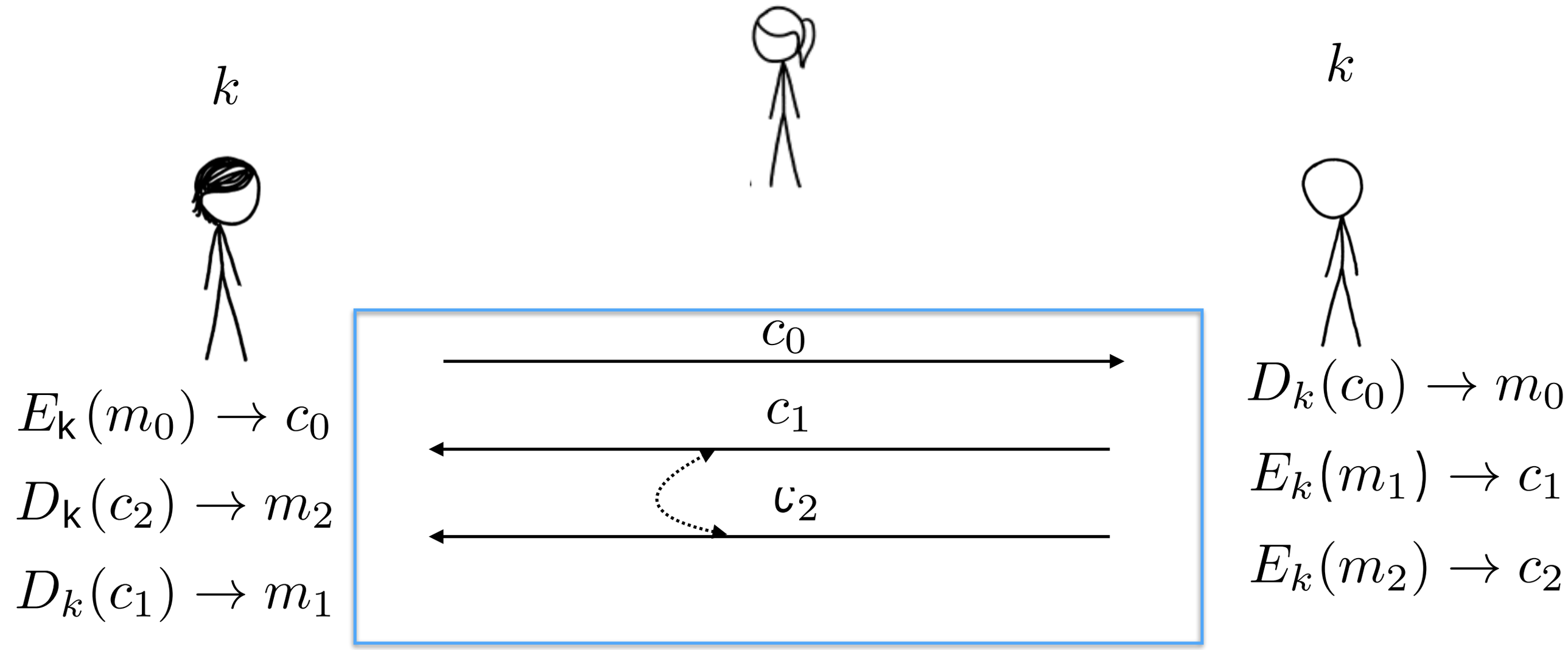
What do we mean with “integrity”?



Alice: is m^* something that Bob wrote?

(Single round)

What we mean with “stateful secure channel”?



m_0, m_1 **or** m_1, m_0 **?**

(Single round)

Constructing a secure channel

Constructing a secure channel

- An authenticated **key establishment protocol.**
  - One, both or more parties are authenticated (depending on the channel setting)
  - Typically using asymmetric or symmetric crypto
- A **key derivation** phase
  - **Channels keys** to be used for symmetric encryption and MAC
- Using the derived keys to **further protect communication**
  - Encryption gives **confidentiality**
  - Message authentication provides **data authenticity** and **data origin authentication**

Secure channel: Nice to have?

- A clean and **well defined API**
    - Developers do not care about tweaking the channel when it works
    - Developers want to “open”, “close”, “send” and “receive”
    - Does the channel provide **stream based** functionality or **message oriented** functionality
    - How are channel errors communicated to the application layer?
    - Does the protocol channel handle retransmission? The application protocol or the transport protocol?
    - Does the channel provide compression?

These options all impact security of the channel

Secure channel protocol design

Phases of a “secure channel protocol” - run

- **Channel establishment**
- **Key establishment**
- **Secure data transfer**
- **Finish the protocol**

- Part 1: Authenticated key exchange
  - Key creation (Diffie–Hellman)
  - Key transport (using RSA)
- Part 2: Secure data transmission (messages must be authenticated and encrypted)
  - Using symmetric constructions to encrypt and authenticate the data

Authenticated key exchange protocol

- (A)KE is a protocol that specifies how two parties can establish a shared session key to be used during a session

    - During the execution of KE it is desired that parties are also able to get some assurance about the identity of the communicating parties.

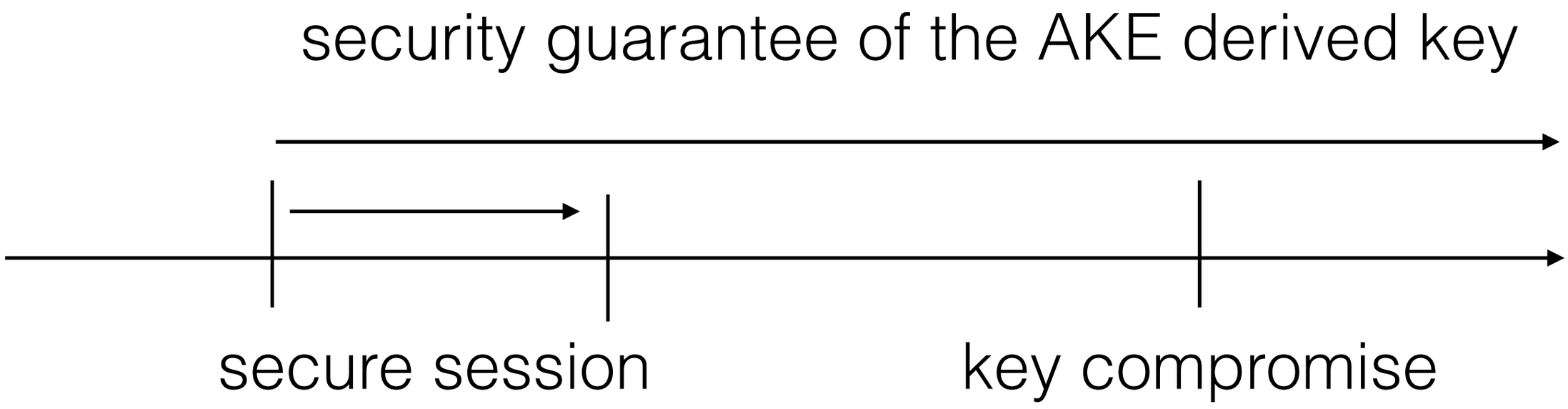
        - Key exchanges should typically be linked to authentication

- Preferably AKE should have the **perfect forward secrecy property**

Perfect forward secrecy

- The AKE provides keys that are “perfectly” secure in the future

- Introduce the notion of a “**long term key**” (Ltk) and “**short term key**” (Stk)



Perfect forward secrecy

More formally we say that an AKE provides perfect forward secrecy if the disclosure of the long term key does not compromise the derived session key

Key derivation phases

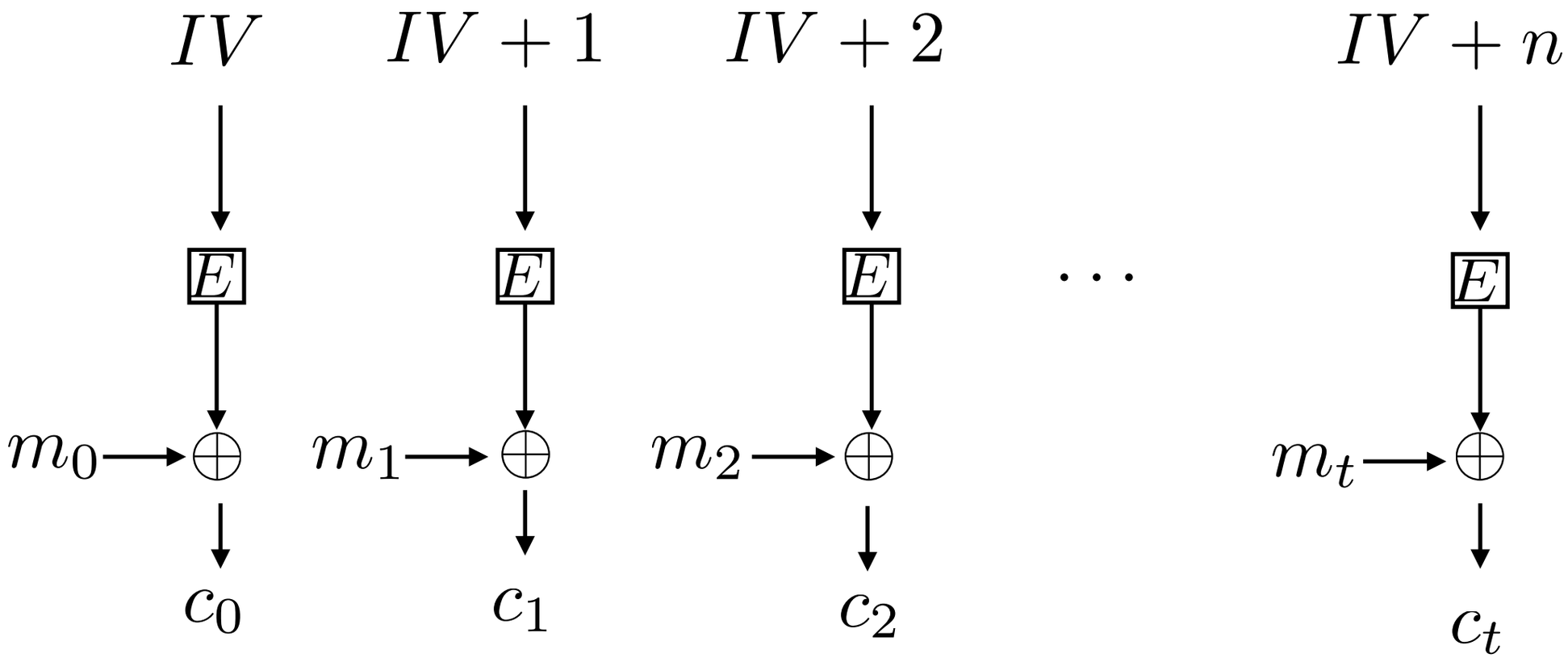
- Our AKE will result in a shared secret, but we need more then a single key for our secure channel
- Often multiple keys are required (refer to this as **channel keys**)
  - Read an write encryption keys, MAC keys
- TLS 1.3 uses HKDF (HMAC based key derivation function, RFC5869)


```
Inputs:
   PRK      a pseudorandom key of at least HashLen octets
            (usually, the output from the extract step)
   info     optional context and application specific information
            (can be a zero-length string)
   L        length of output keying material in octets
```

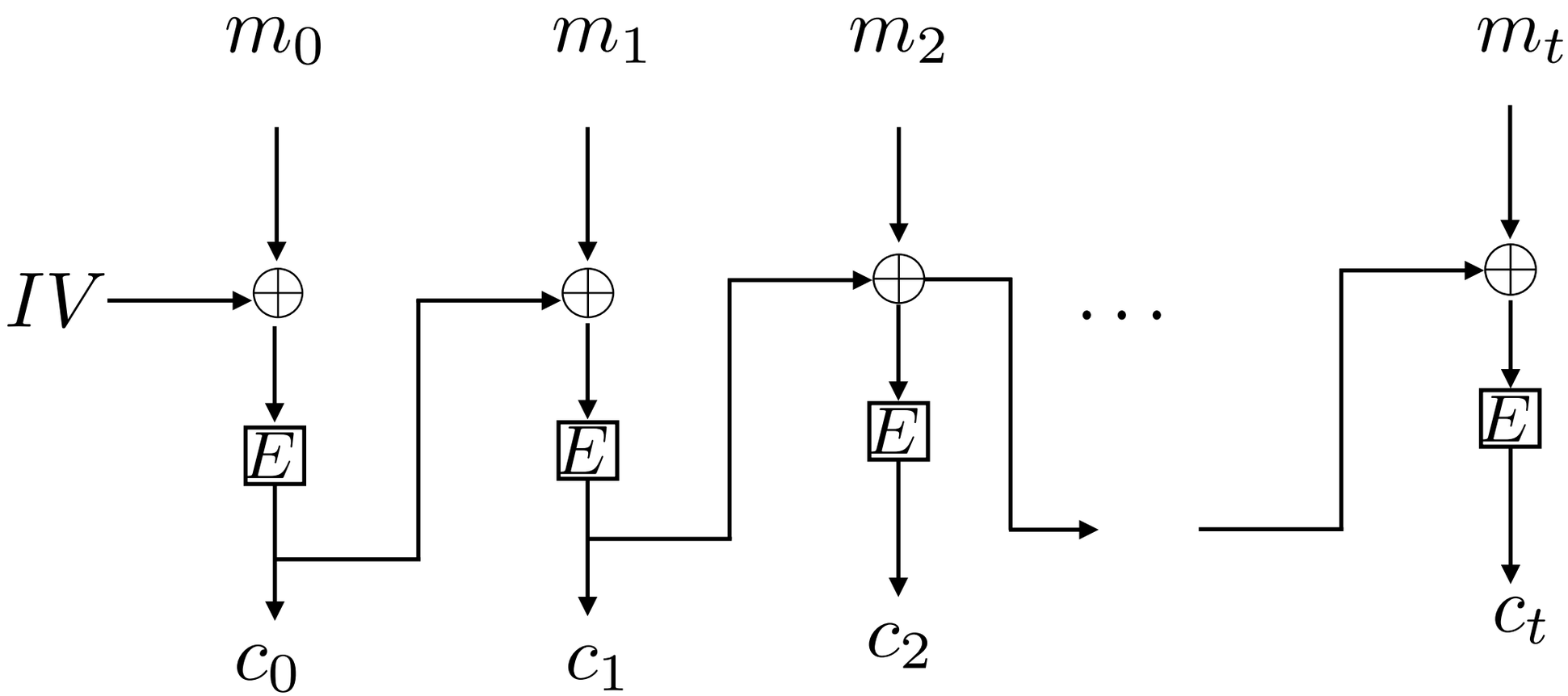
- SSHv2 uses a hash function (RFC4253)

```
o  Initial IV server to client: HASH(K || H || "B" || session_id)

o  Encryption key client to server: HASH(K || H || "C" || session_id)

o  Encryption key server to client: HASH(K || H || "D" || session_id)

o  Integrity key client to server: HASH(K || H || "E" || session_id)

o  Integrity key server to client: HASH(K || H || "F" || session_id)
```

Protecting messages



- **CTR** uses a block cipher to build a stream cipher
- Random initial value chosen for IV
    - Encrypt blocks to create a stream of ciphertext blocks
- Same process to decrypt

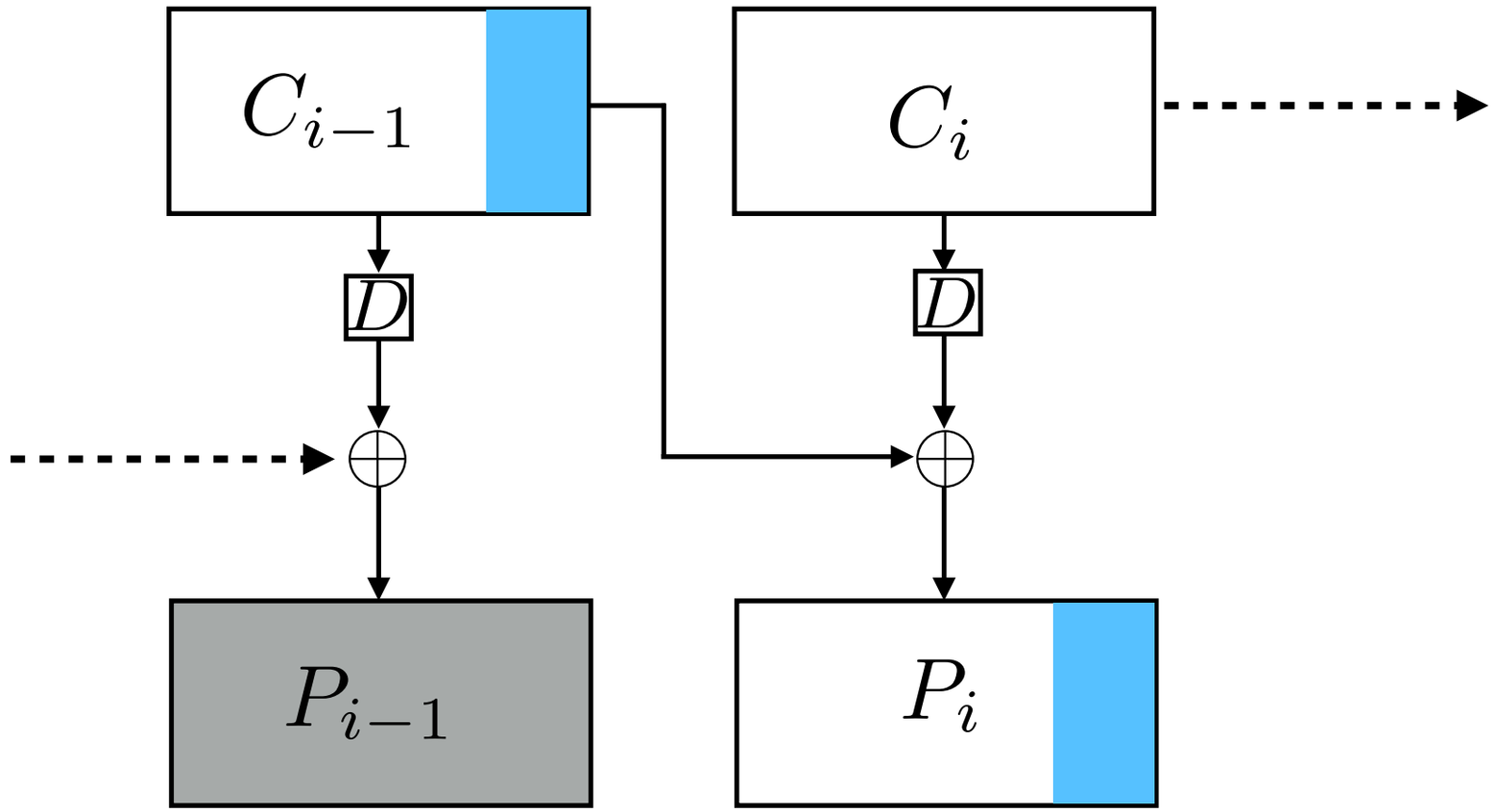


- With **CBC** every encrypted block depends on all previous encrypted blocks
  - Identical plaintext blocks will encrypt to different values
- Randomised IV will result in different ciphertext messages when identical plaintext messages are encrypted

- CBC and CTR
  - Offers no message authentication
  - Adversaries can still modify the ciphertext leading to predictable changes in the plaintext

$$f(c_i) \; ! \quad c_i', D_k(c_i') \; ! \quad p_i'$$

Example: CBC



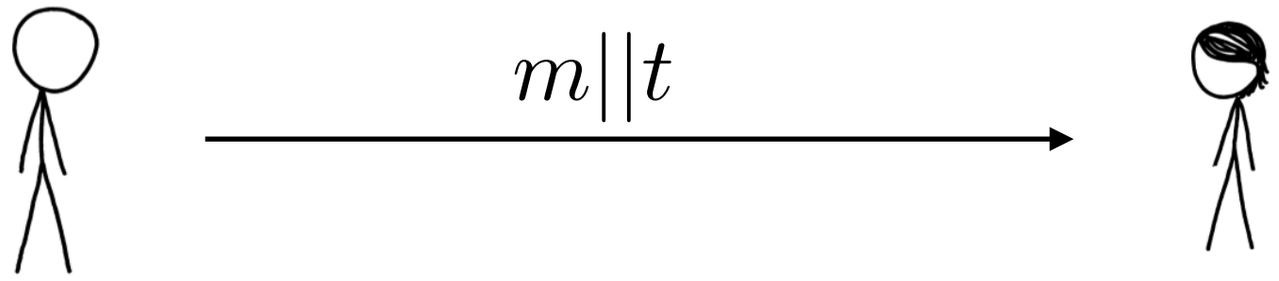
- Flipping bits in C_{i-1} leads to controlled changes in P_i
- But block P_{i-1} is randomised

Message integrity (MAC)

$I = (S, V)$

$S : K \times M \to T$

$V : K \times M \times T \to \{1, \perp\}$



- Message authentication code (MAC's) provide authentication and authenticity protection for messages
- HMAC is general method for building a MAC scheme from a hash function.
- The key is derived from the key derivation phase of the channel protocol
- This MAC construction guarantees unforgeability

The “Cryptographic doom principle”

As stated by Moxie Marlinspike:

“if you have to perform ***any*** cryptographic operation before verifying the MAC tag on a message you’ve received, it will *somehow* inevitably lead to doom.”

And it has…

Authenticated encryption

- AE is a conceptual goal were we try to accomplish both **message authentication** and **data confidentiality** in a single construction

$$D_k(C_i) \rightarrow P_i \vee \perp$$

- GCM, OCB, CCM constructions

Authenticated encryption with additional data

- Same as AE but now we can include data that is **not encrypted** but is **authenticated**.

AD : **Additional data (not encrypted, only authenticated)**

$$E_k(AD, M_i) \rightarrow C_i$$

$$D_k(AD, C_i) \rightarrow M_i \vee \perp$$

Real world protocols

What a **developer** wants:

A **secure** drop in replacement from TCP

TLS

- **TLS handshake protocol (AKE)**
    - Will provide us with channel keys
- **TLS record protocol (AEAD)**
    - Will encrypt and authenticated application provided messages

TLS 1.3: Record protocol

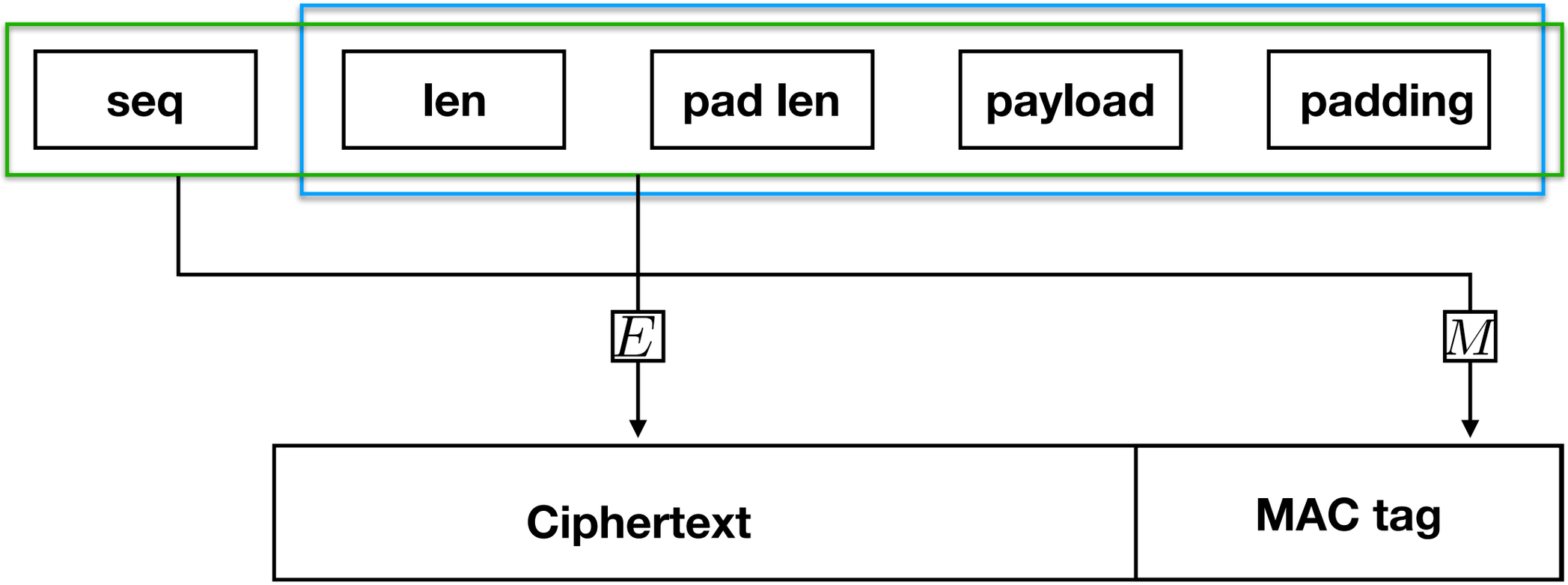
- Record protocol:
    - All messages are protected using AEAD schemes, so the result is encrypted and authenticated. No longer a dedicated MAC
        - The additional data will always be zero
    - Records are typed, allowing for multiplexing of higher level protocols:
        - Handshake
        - Application
        - Alert
- Keys for the algorithms are supplied by the HKDF function

SSHv2

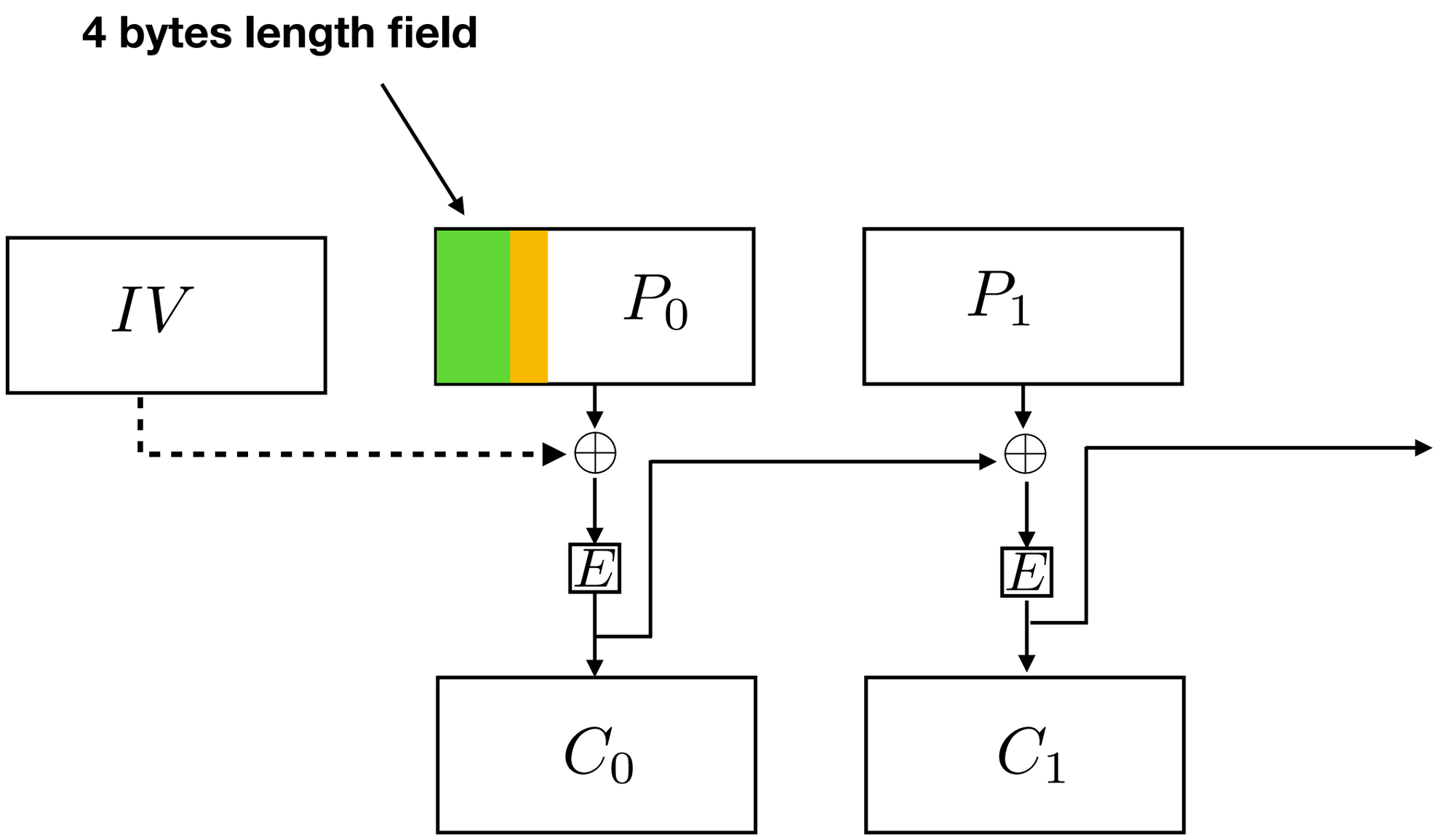
- **SSH uses a multi layered architecture**

- SSH transport layer
  - Initial connection (handshake)
    - Creates a secure channel using the **binary packet protocol**
- SSH authentication protocol (client authentication)
  - Runs on the transport protocol
- SSH connection protocol (concurrent connections over a single transport layer)

SSH binary packet protocol

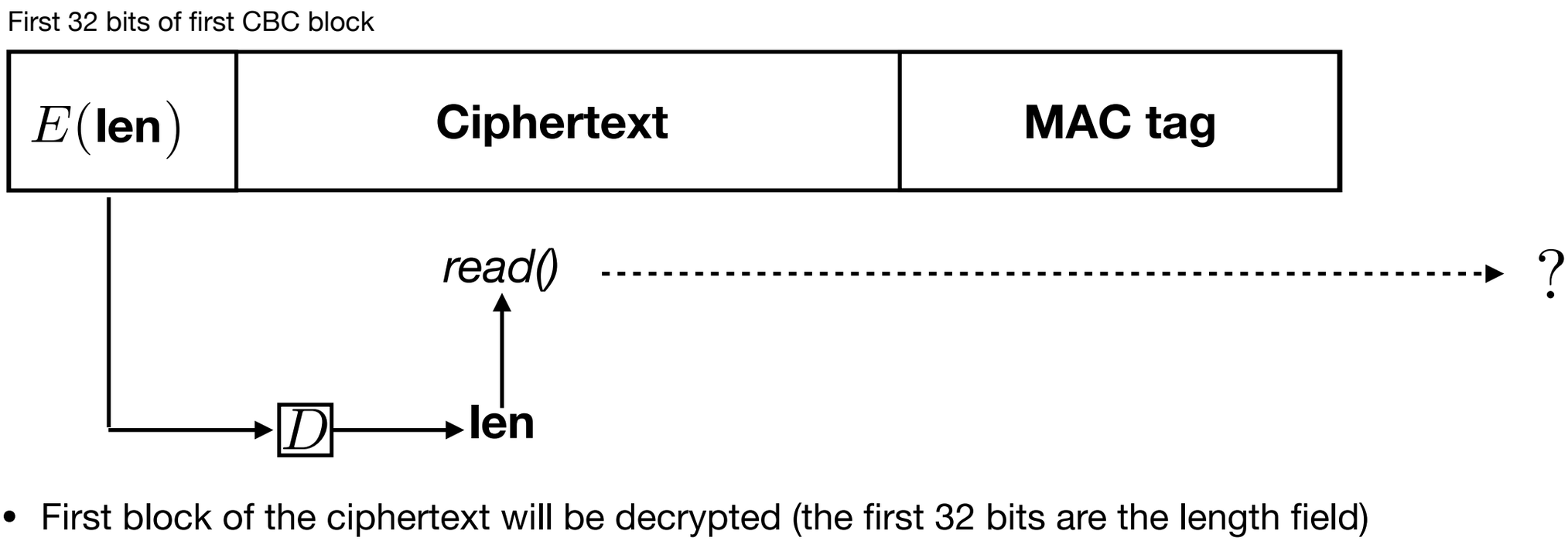


- Single ciphertext message can be fragmented over multiple packets
- Stream of ciphertext bytes
- CBC mode uses **chained IV** (last ciphertext block is IV of new message)
- CTR mode with initial counter value from handshake

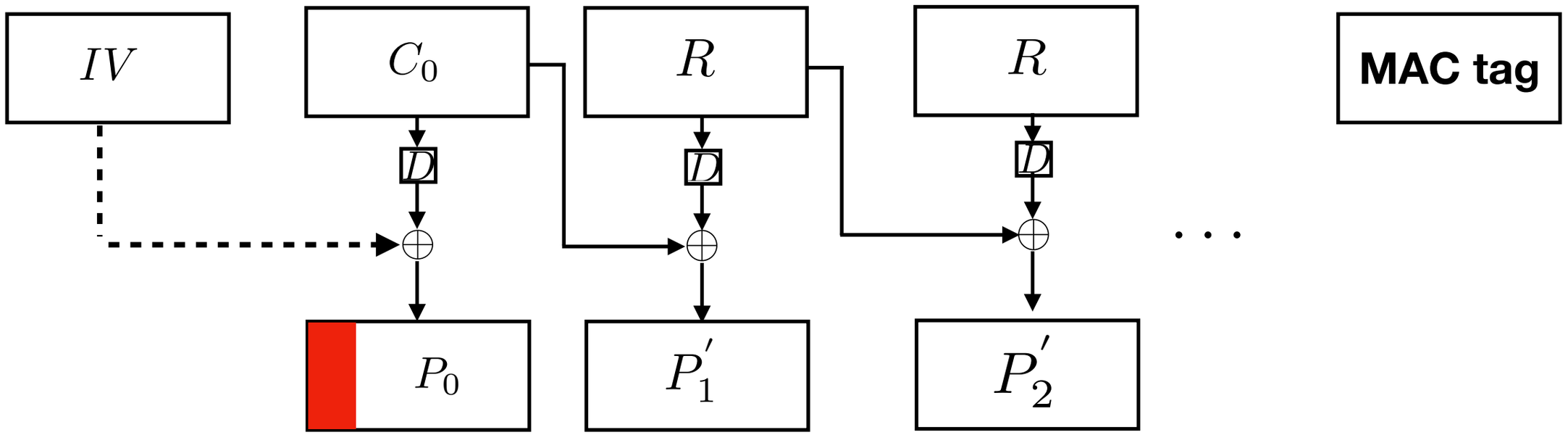


SSH binary packet protocol with CBC

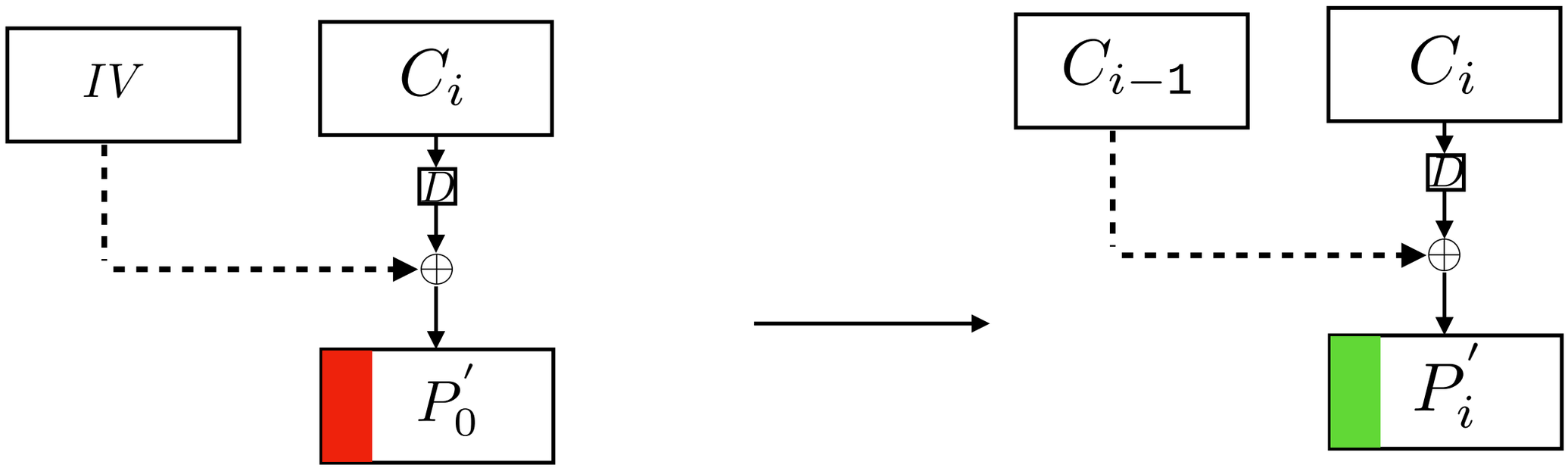
SSH binary packet protocol



- First block of the ciphertext will be decrypted (the first 32 bits are the length field)
- This will tell the SSH server how many bytes to read before attempting decryption
- Server will decrypt as many data as the length field specifies, after decryption **it will validated the MAC**
- **The attacker will send one block at a time until we receive a MAC error**
    - The attacker knows when the first 32 bits of the first plaintext block (length)!
    - **Can be used to mount a plaintext recovery attack**



- We submit the IV and first block
- We keep submitting random blocks (16 bytes) until we receive a MAC error
- Once we receive a MAC error we know the value of



$$P_i^{'} = C_{i-1} \oplus D(C_i)$$

$$= C_{i-1} \oplus IV \oplus P_0^{'}$$

Paper: http://www.isg.rhul.ac.uk/~kp/SandPfinal.pdf

To conclude

To conclude

- We looked at the definition and properties of a secure channel
- We discussed the notions of security for those properties
- We looked at the different phases of a secure channel
- We explored TLS record protocol and SSH binary packet protocol

Thank you