Bar-Ilan Winter School
Lecture 4
Symmetric crypto for secure channels

Kenny Paterson @kennyog

ROYAL HOLLOWAY UNIVERSITY OF LONDON

Overview

- Secure channels and their properties
- A glance at the literature
- AEAD
- AEAD ≠ secure channel
- Building better models
- Closing remarks


Secure channels and their properties

Security properties

We assume that symmetric keys are already in place (see days 1-4!).

We then seek:

- **Confidentiality** – privacy for data
- **Integrity** – detection of data modification
- **Authenticity** – assurance concerning the source of data

Some less obvious security properties

- **Anti-replay**
  - Detection that messages have been repeated.
- **Detection of deletion (and truncation)**
  - Detection that messages or parts of messages have been deleted by the adversary or dropped by the network.
- **Detection of re-ordering**
  - Ensuring that the relative order of messages in *each* direction on the secure channel is preserved.
  - Possibly performing buffering of messages received out of order and re-ordering, in the event of violation.
  - Possibly maintaining "correct interleaving" for messages in both directions.
- **Prevention of traffic-analysis.**
  - Using traffic padding and length-hiding techniques.
  - Switch from CBC-mode to AES-GCM makes traffic analysis trivial in TLS!

Possible functionality requirements

- **Fast and low-memory requirements.**
  - Performance may be heavily hardware-dependent.
  - May have different algorithms for different platforms, e.g. AES on Intel CPUs, ChaCha20 on mobile CPUs.
- **On-line/parallelisable crypto-operations**
- **IPR-friendly**
  - This issue has slowed down adoption of many otherwise good algorithms, e.g. OCB.
- **Easy to implement**
  - Without introducing any side-channels.

Additional requirements

- We need a clean and well-defined API.
- Because the reality is that our secure channel protocol will probably be used blindly by a security-naïve developer.
- Developers want to "open" and "close" secure channels, and issue "send" and "recv" commands.
- They'd like to simply replace TCP with a "secure TCP" having the same API.
- Or to just have a simple API for wrapping atomic messages securely.

Additional API-driven requirements

- Does the channel provide a stream-based functionality or a message-oriented functionality? (TCP-like or UDP-like)
- Does the channel accept messages of arbitrary length and perform its own fragmentation and reassembly, or is there a maximum message length?
- Does the channel offer data compression?

- How is error handling performed? Is a single error fatal, leading to tear-down of channel, or is the channel tolerant of errors?
- How are these errors signalled to the calling application? How should the programmer handle them?

Additional API-driven requirements

- Does the secure channel itself handle retransmissions if they are needed? (QUIC)
- Or is this left up to the application using the secure channel if it desires to have it? (DTLS, IPsec, WEP/WPA/WPA2)
- Or is it assumed to be handled by the underlying network transport? (SSH, TLS)

- **These are design choices that all impact on security**
- **They are not well-reflected in the basic security definitions for symmetric encryption**

What does the literature tell us?

- Shoup (http://shoup.net/papers/skey.pdf, 1999):
  - 2 pages on secure sessions in a 50 page+ paper on key exchange.
  - Simulation-based rather than game-based indistinguishability notions.
  - "It should be simple to fill in the details…"
- Canetti (eprint 2000/067):
  - The Universal Composability framework.
  - Heavy use of *ideal* secure channels.
  - *Impractical* construction of secure channels via one-time use of public keys and ideal authenticated channels.
  - Needs non-committing encryption to achieve UC against adaptive corruptions.
- Canetti-Krawczyk (eprint 2001/040):
  - Basic definition for secure channels using game-based, indistinguishability notion.
  - Construction via "EtM".

What does the literature tell us?

- Canetti-Krawczyk (eprint 2002/059):
  - UC notion for secure channels, realization using EtM.
- Bellare-Kohno-Namprempre (CCS'02):
  - Game-based stateful security notions for Authenticated Encryption (AE).
  - Capturing reordering and dropping attacks in addition to the usual CIA attacks.
- Kohno-Palacio-Black (eprint 2003/177):
  - Explicit consideration of reordering, replay, packet drop issues in game-based setting.
  - Different models allowing/denying different combinations of features.

What does the literature tell us?

- Maurer-Tackmann (CCS'10)
  - Secure channels in the "constructive cryptography" framework.
- Paterson-Ristenpart-Shrimpton (Asiacrypt'11)
  - LH-AEAD notion.
  - Incorporating basic length-hiding into AEAD notions.
- Jager-Kohlar-Shäge-Schwenk (Crypto'12)
  - ACCE: secure key establishment and channel definition built on LH-AEAD + key exchange.
  - Monolithic and hard to work with, but justified for analysing TLS.
  - Used in Krawczyk-Paterson-Wee (Crypto'13) to analyse several TLS modes.

What does the literature tell us?

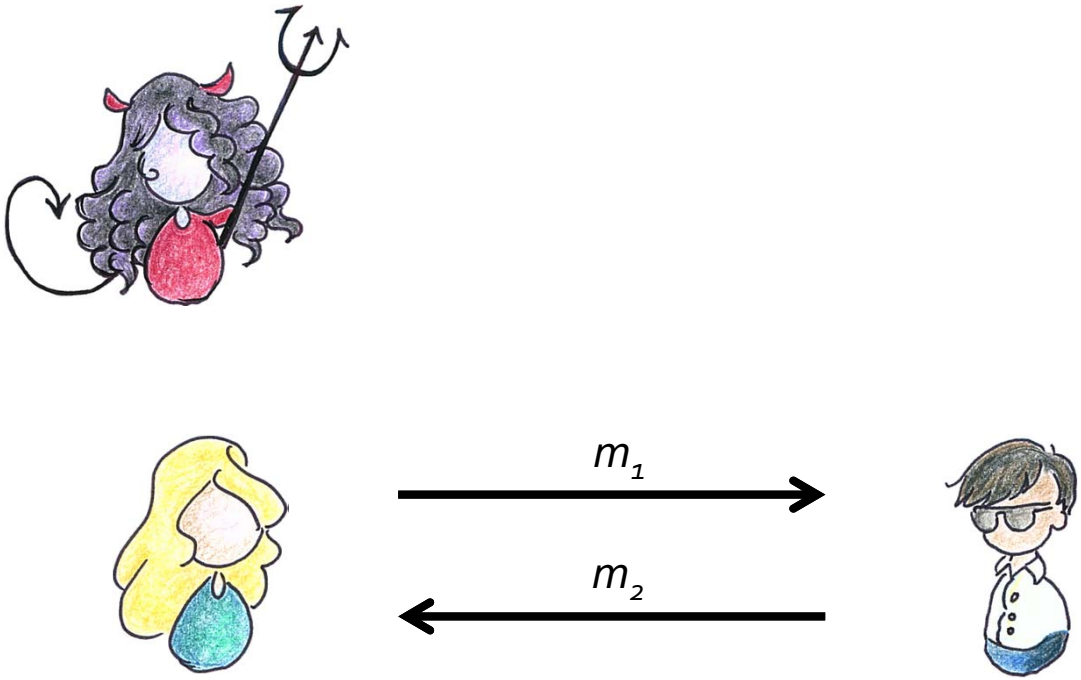
- Boldyeva-Degabriele-Paterson-Stam (EC'12); Albrecht-Degabriele-Hansen-Paterson (CCS'16):
  - Development of "symmetric encryption supporting fragmented decryption" framework, capturing SSH's specific security goals.
  - Analysis of SSH's constructions.
- Fischlin-Günther-Marson-Paterson (C'15):
  - Development of streaming secure channels framework, capturing TLS security goals, from the API perspective.
- Delignat-Lavaud *et al.* (IEEE S&P'17):
  - Analysis of TLS 1.3 Record Protocol (as was) from a streaming perspective.

Summary of the literature

- Lots of literature on AE/AEAD.
- Much less on the more complex secure channel primitive.
- Current models are do not yet capture all of subtleties of secure channels as they are used in practice.
- Work to be done!

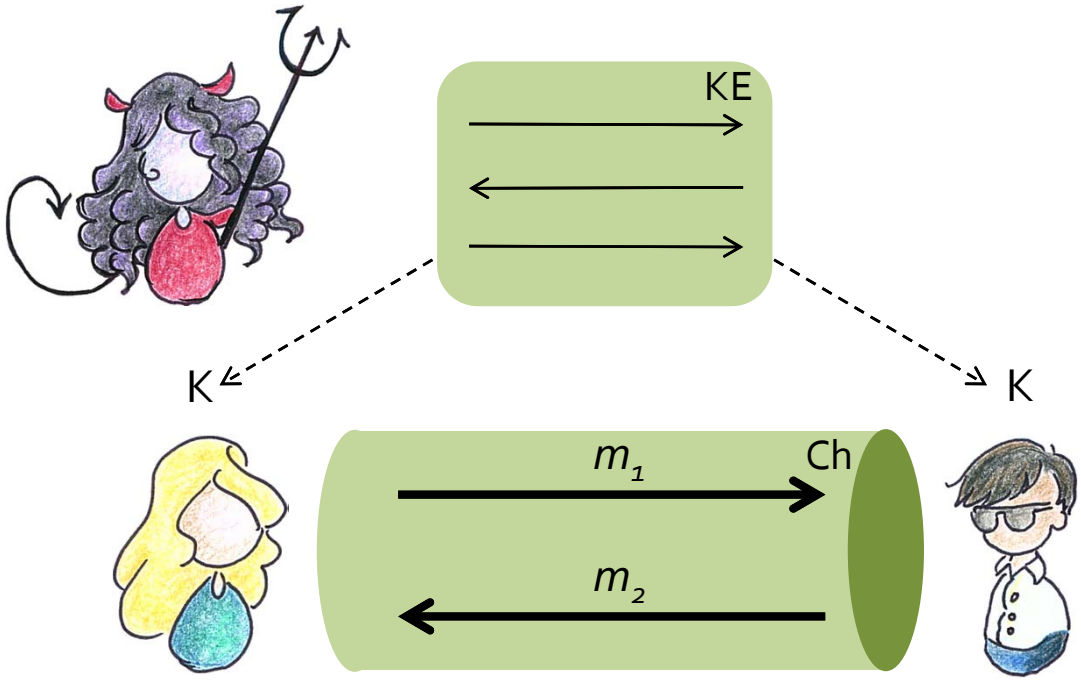

AEAD

Security for Symmetric Encryption

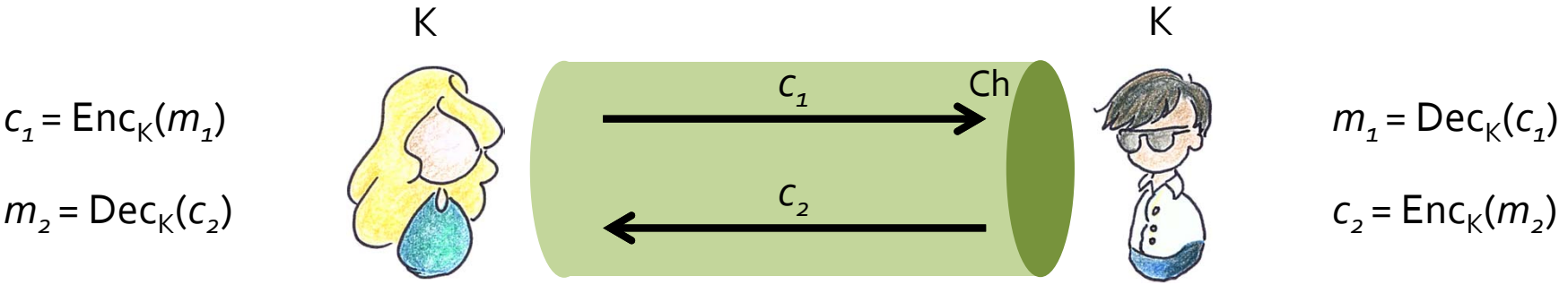


Pictures by Giorgia Azzurra Marson

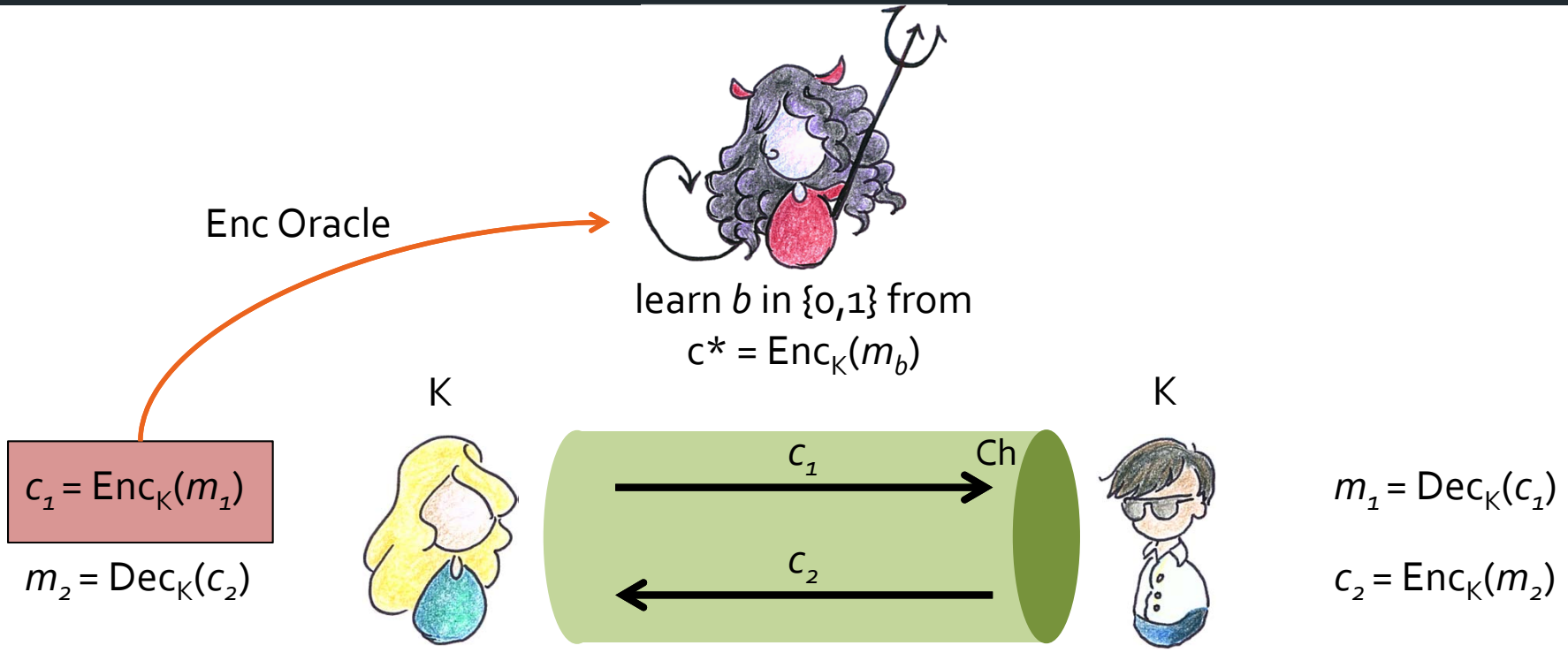
Security for Symmetric Encryption



Security for Symmetric Encryption

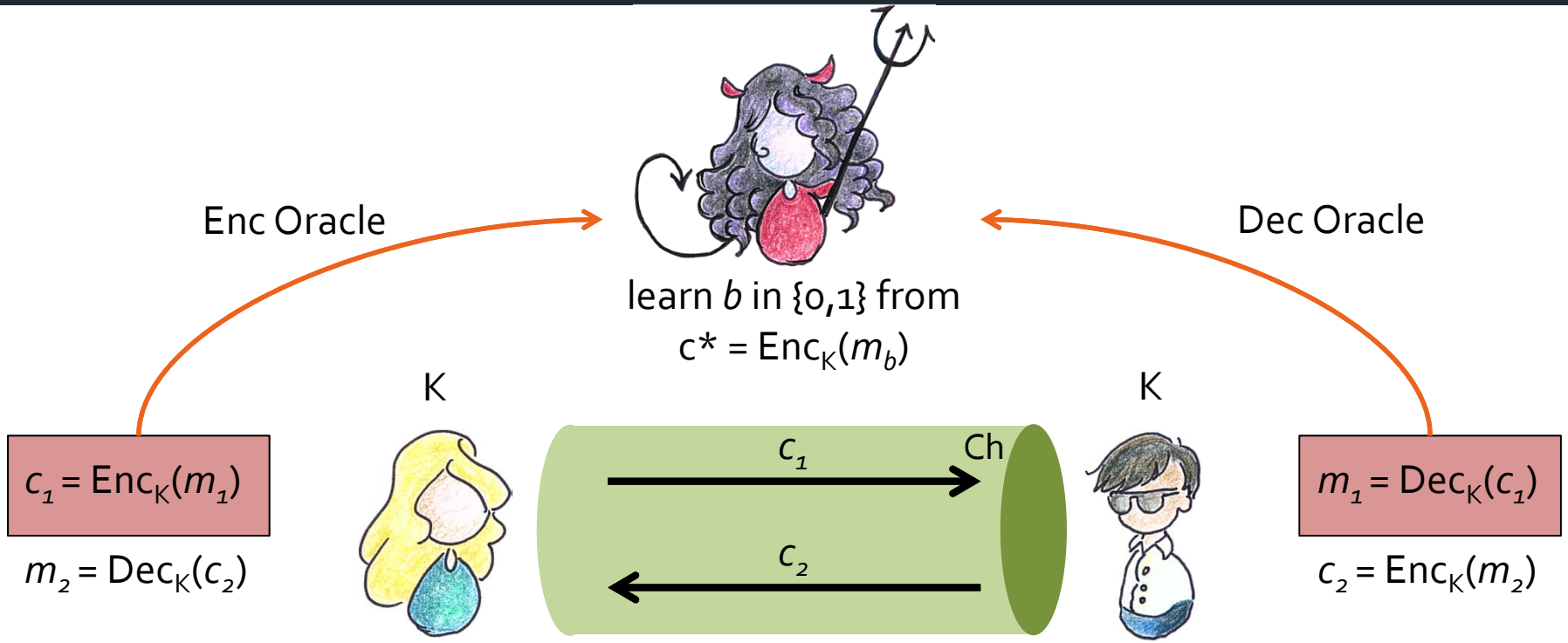


Security for Symmetric Encryption – Confidentiality



IND-CPA
(Goldwasser-Micali, 1984;
Bellare-Desai-Jokipii-Rogaway, 1997).

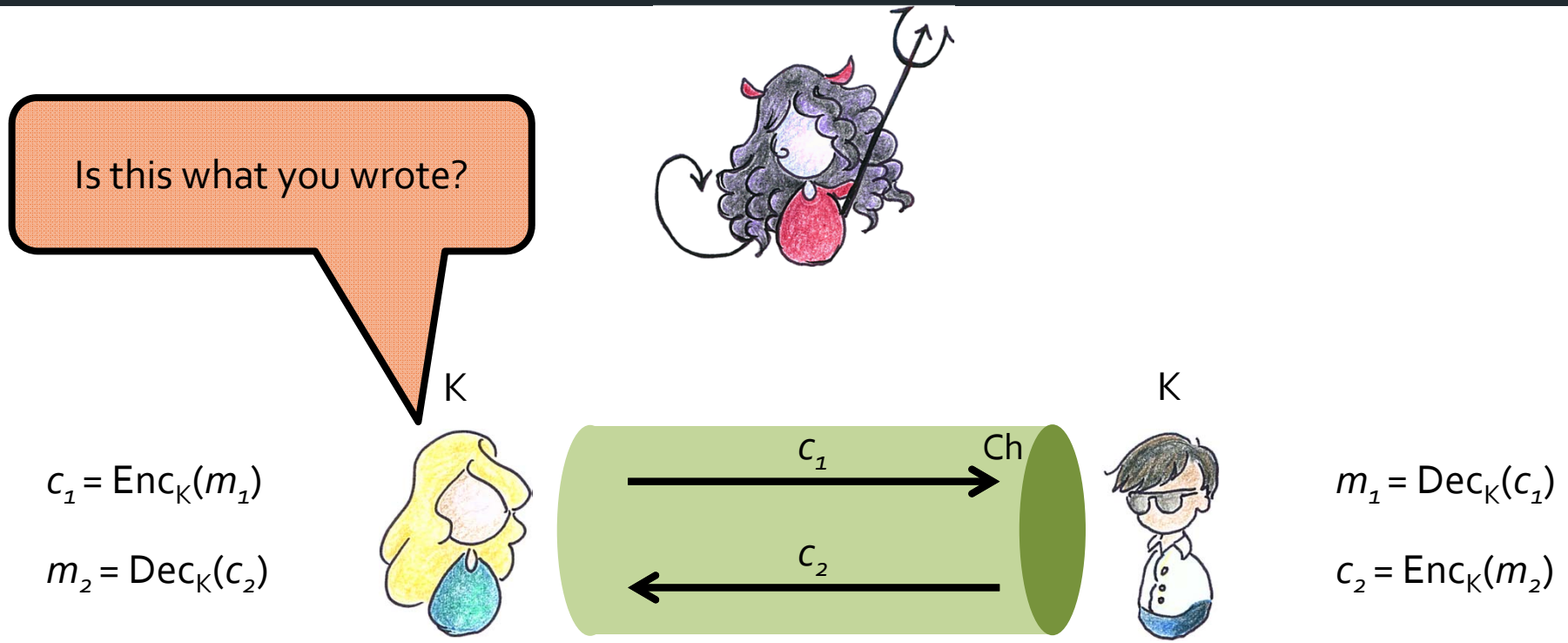
Security for Symmetric Encryption – Confidentiality



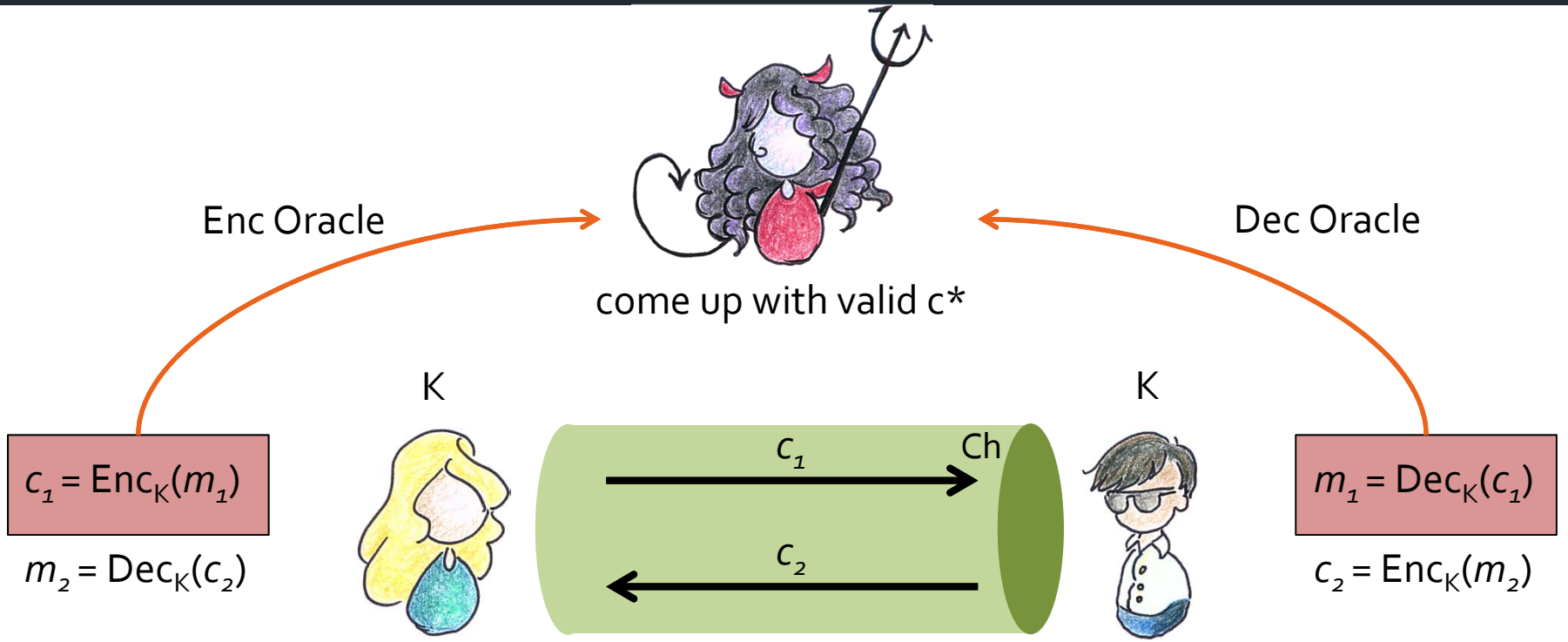
IND-CPA
(Goldwasser-Micali, 1984;
Bellare-Desai-Jokipii-Rogaway, 1997).

IND-CCA
(Naor-Yung, 1990;
Rackoff-Simon, 1997).

Security for Symmetric Encryption – Integrity

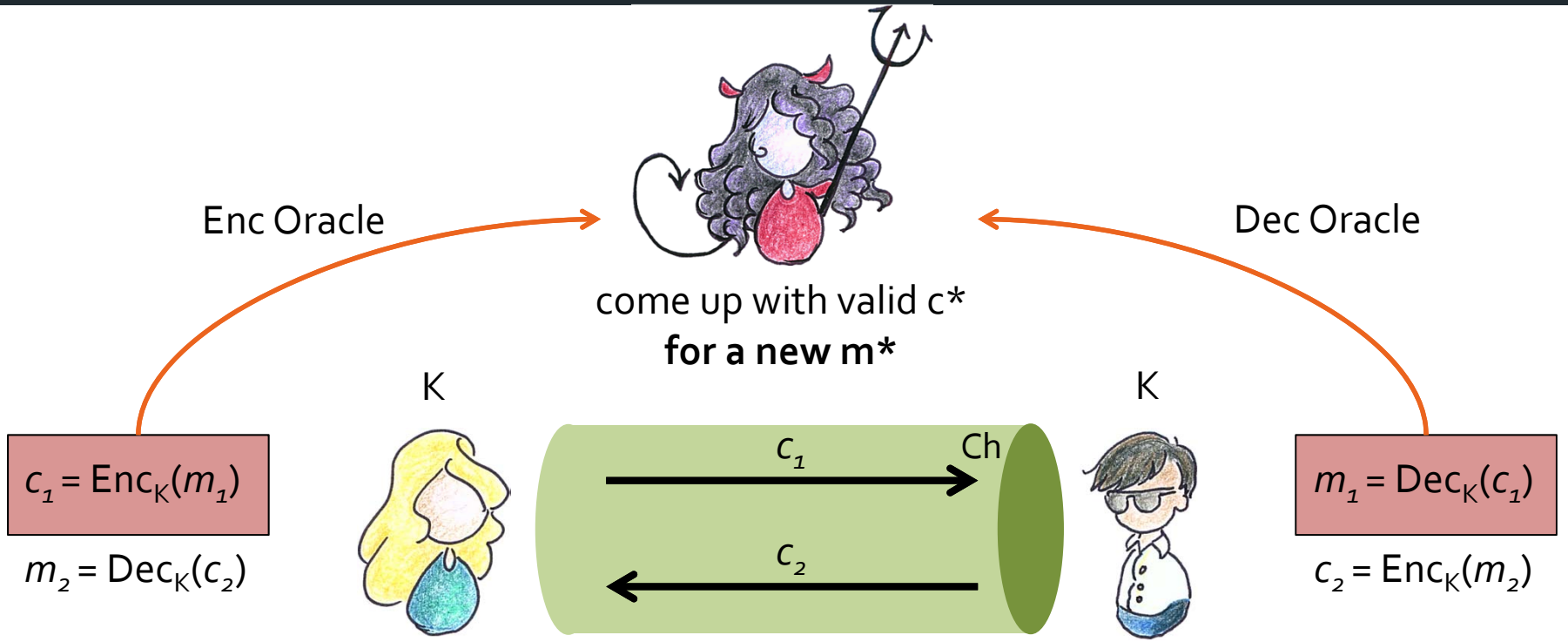


Security for Symmetric Encryption – Integrity



INT-CTXT
(Bellare, Rogaway, 2000)

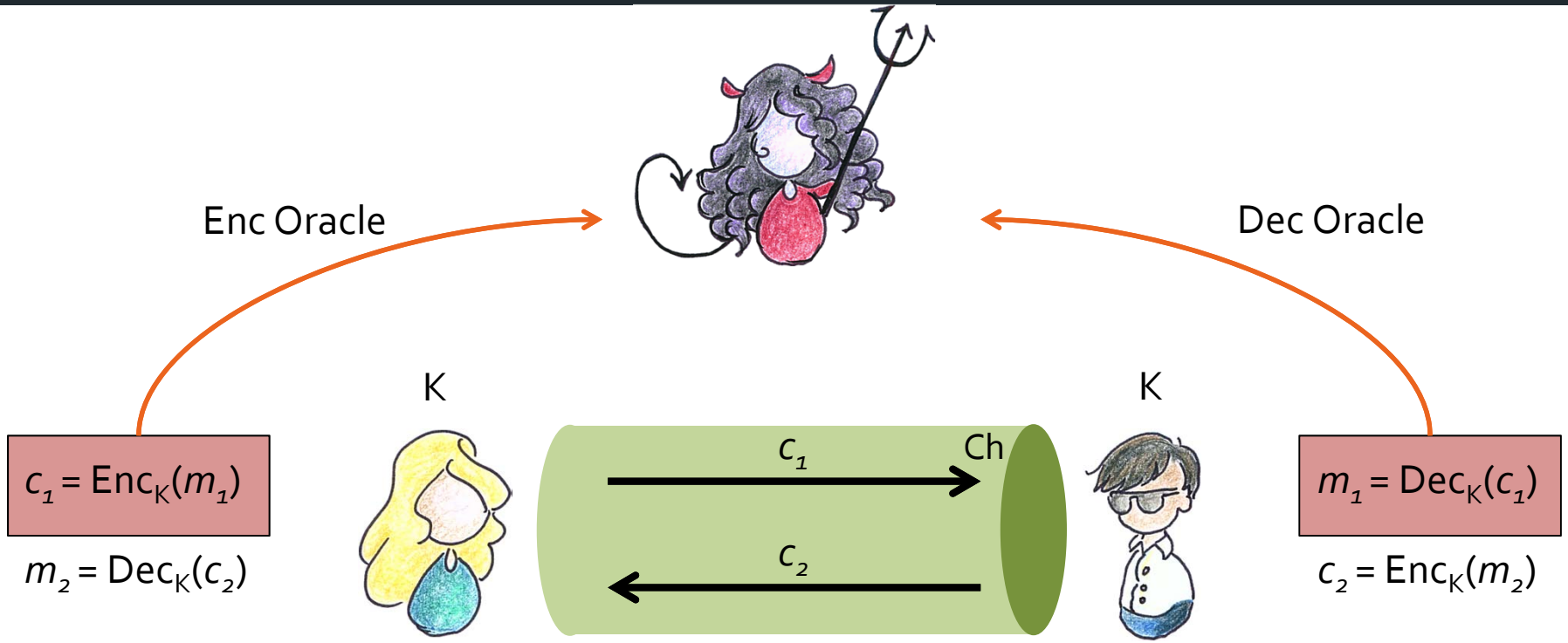
Security for Symmetric Encryption – Integrity



INT-PTXT
(Bellare-Namprempre, 2000)

INT-CTXT
(Bellare, Rogaway, 2000)

Security for Symmetric Encryption – AE

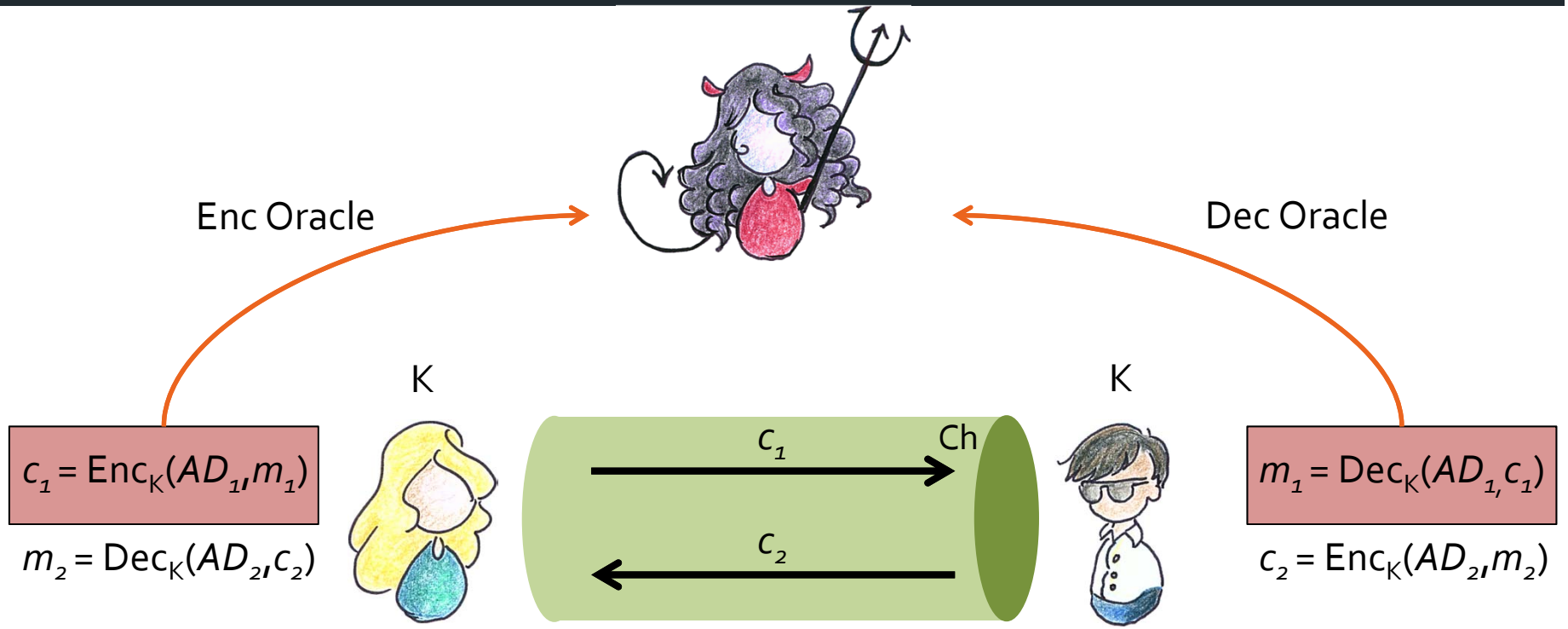


INT-PTXT
(Bellare-Namprempre, 2000)

Authenticated Encryption
IND-CPA + INT-CTXT
(➔IND-CCA)

INT-CTXT
(Bellare, Rogaway, 2000)

Security for Symmetric Encryption – AEAD



Authenticated Encryption with Associated Data

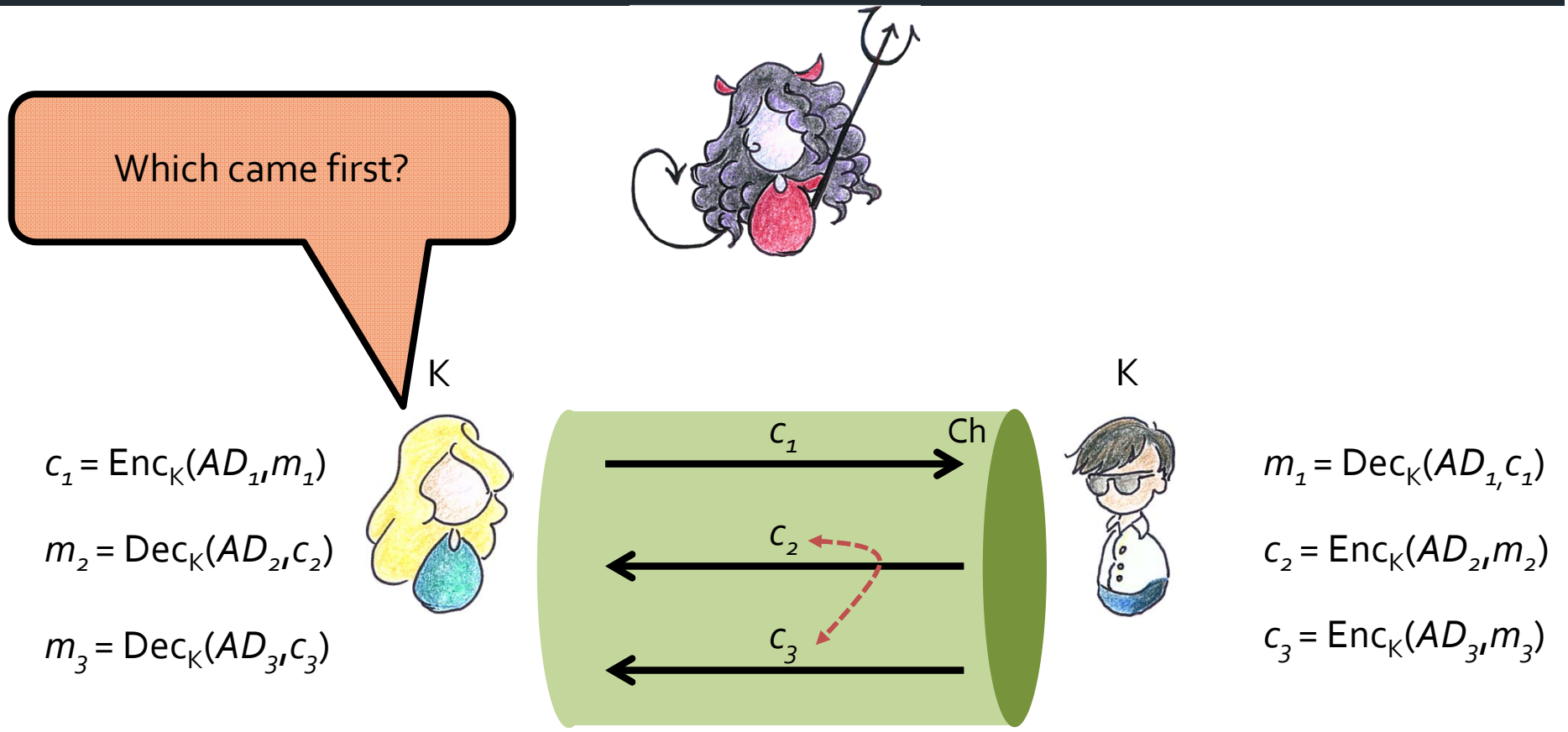
AE security for message *m*

Integrity for associated data *AD*

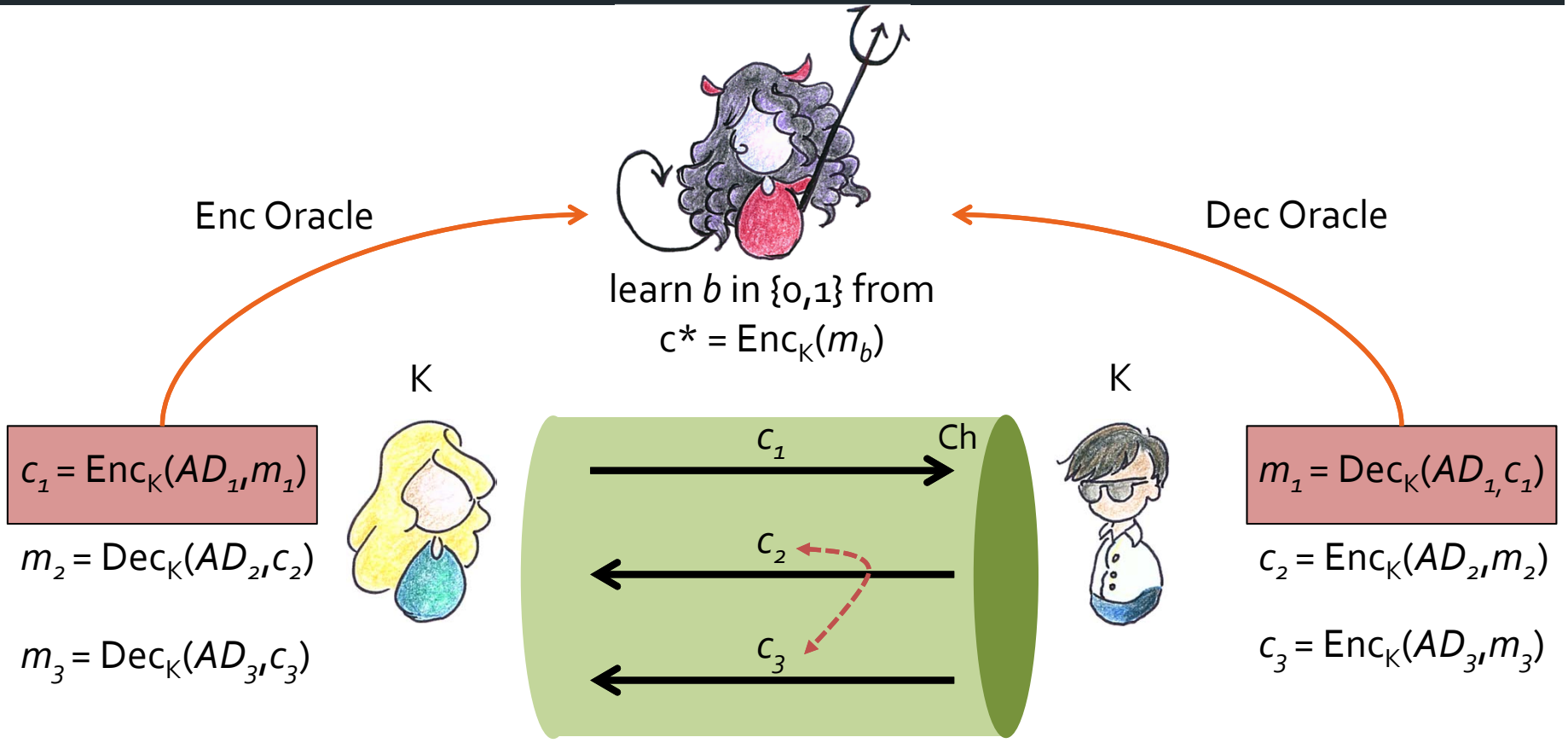
Strong binding between *c* and *AD*

(Rogaway 2002)

Security for Symmetric Encryption – stateful AEAD



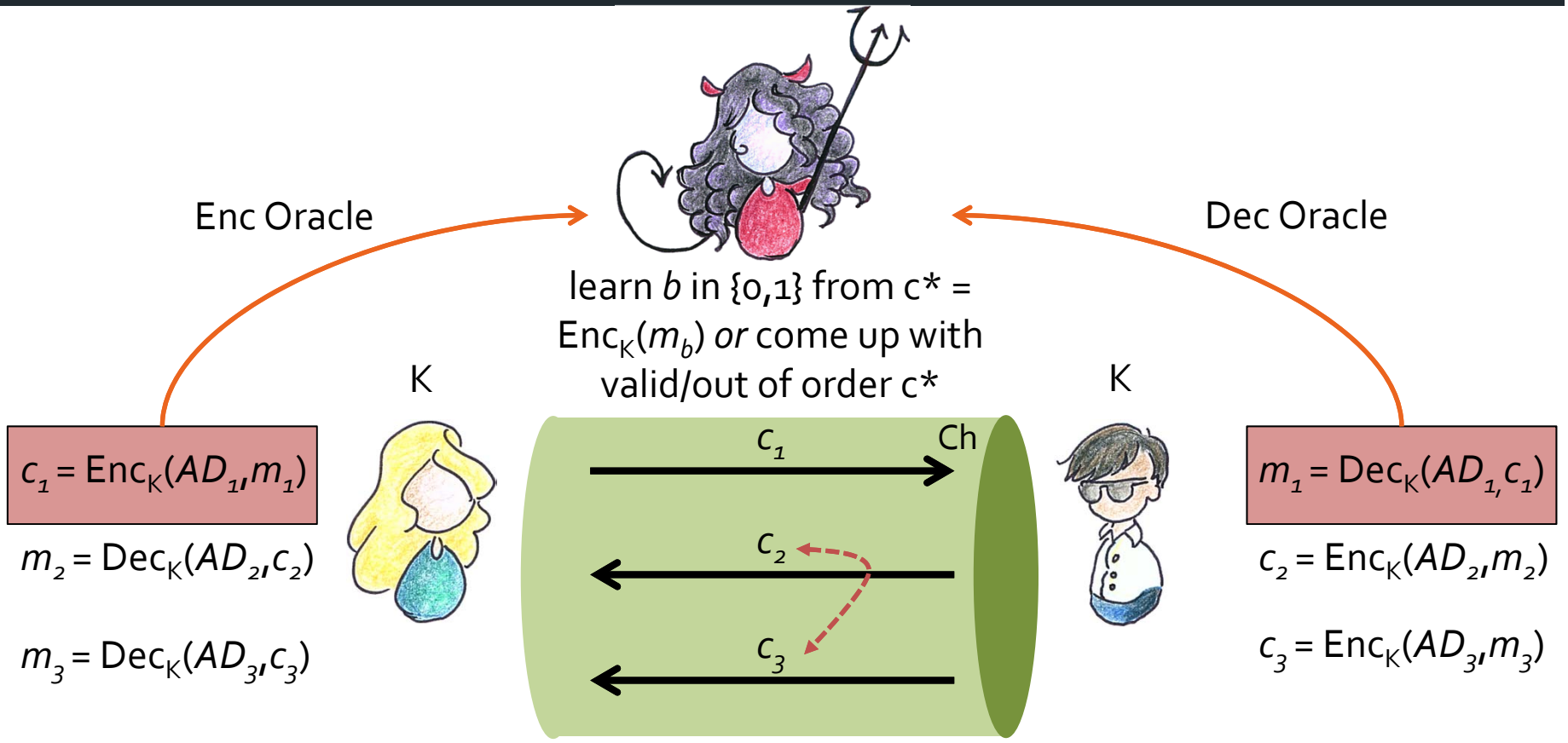
Security for Symmetric Encryption – stateful AE(AD)



IND-sfCCA

(Bellare-Kohno-Namprempre, 2002)

Security for Symmetric Encryption – stateful AE(AD)



IND-sfCCA

Stateful AEAD

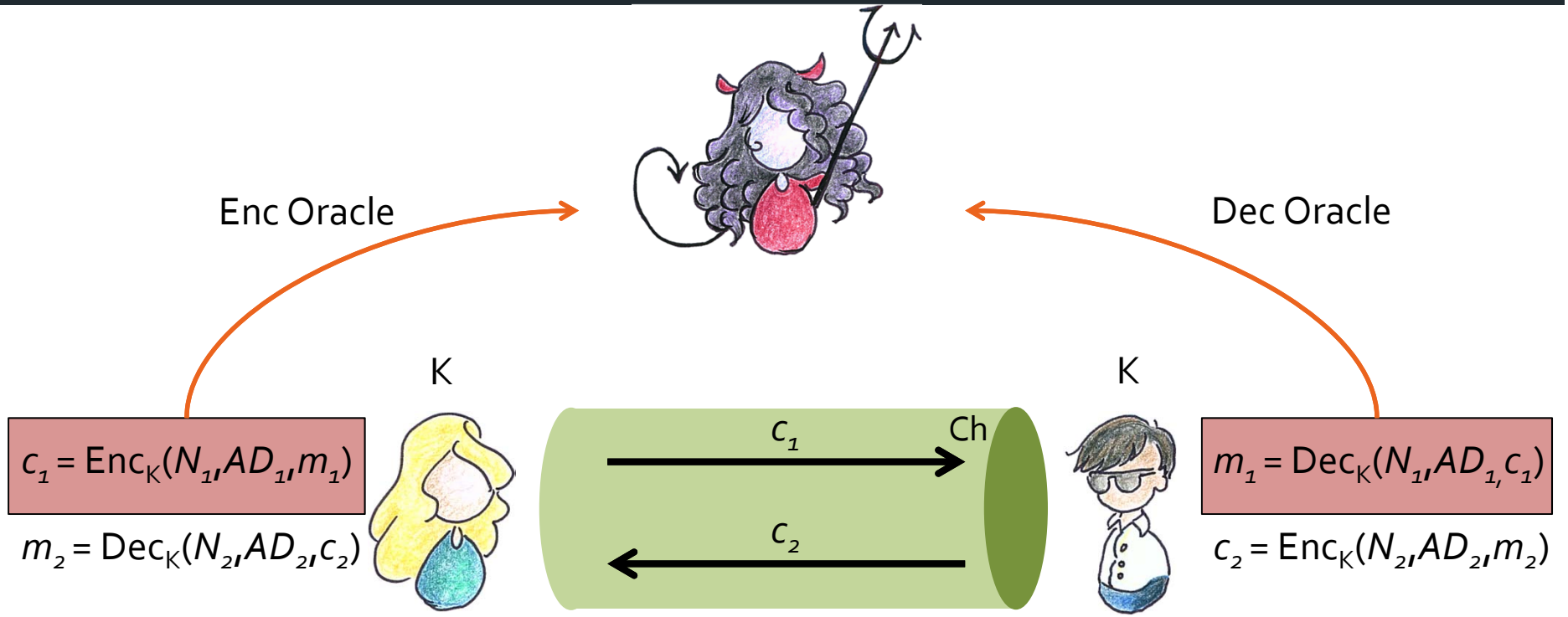
(Bellare-Kohno-Namprempre, 2002)

INT-sfCTXT

INT-sfPTXT

(Brzuska-Smart-Warinschi-Watson, 2013)

Security for Symmetric Encryption – nonce-based AEAD



Nonce-based Authenticated Encryption with Associated Data

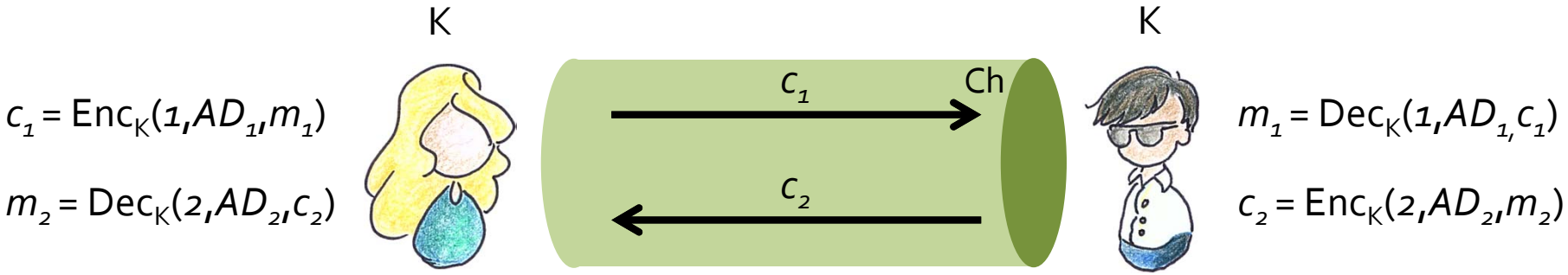
As per AEAD, but with additional input *N* to Enc and Dec algorithms

Adversary may arbitrarily specify *N*, but "no repeats" rule in Enc queries

Enc and Dec can now be *stateless* and *deterministic*

(Rogaway 2004)

From nonce-based AEAD to a basic secure channel

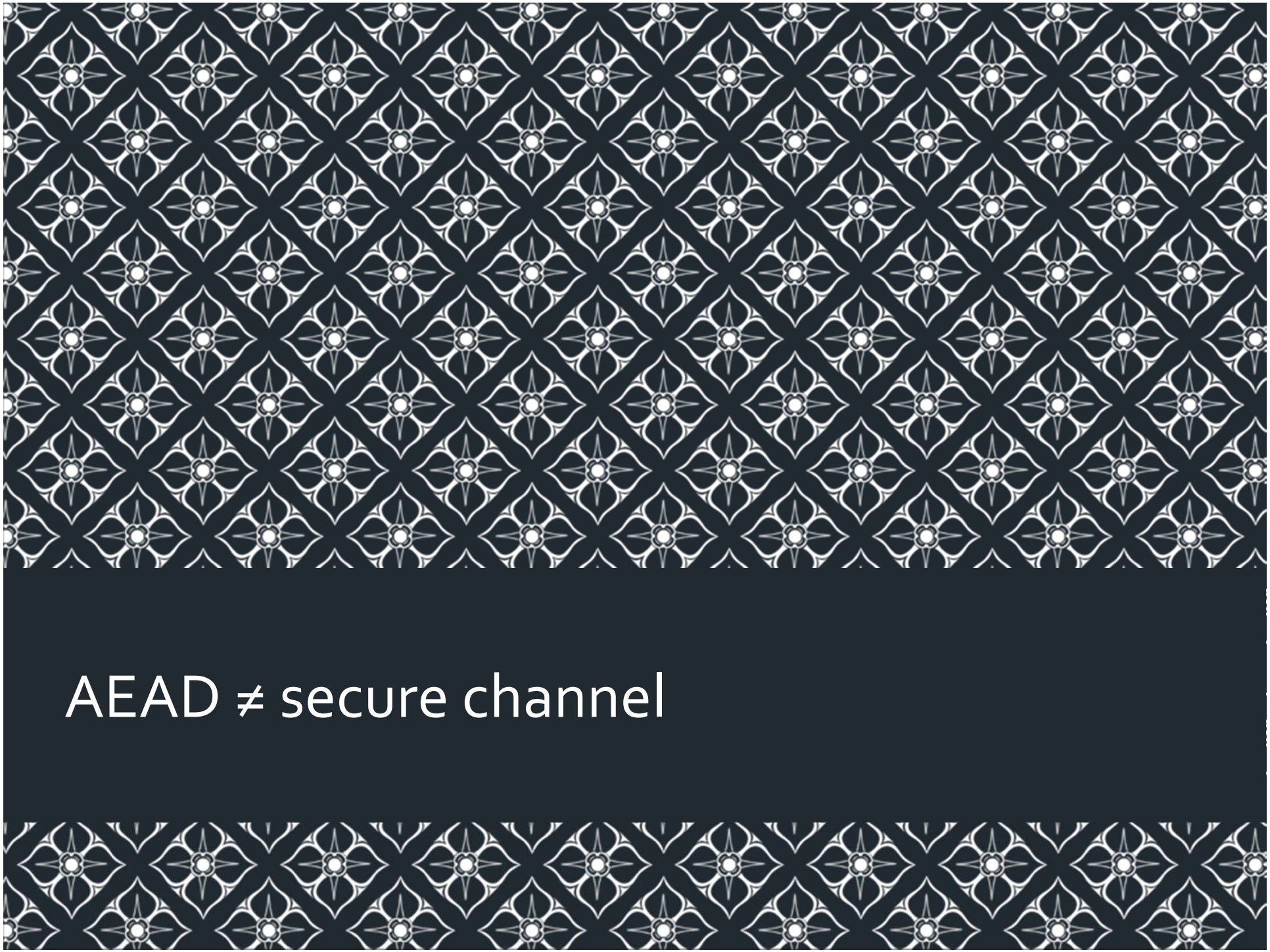


Nonce-based AEAD scheme to build a basic secure channel:
Sender uses sequence of counter values for nonces.
Receiver maintains local copy of counter.
Integrity properties of AEAD catch reordering/deletion attacks.

CAESAR

- CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness.
- Initiated by Dan Bernstein, supported by committee of experts.
- Main goal is the design of a *portfolio* of **AE schemes.**
- CAESAR has involved dozens of person-years of effort and led to a major uptick in research activity.

- **It seems that most of the cryptographic community has settled on nonce-based AEAD as their design target.**


AEAD ≠ secure channel

AEAD ≠ secure channel

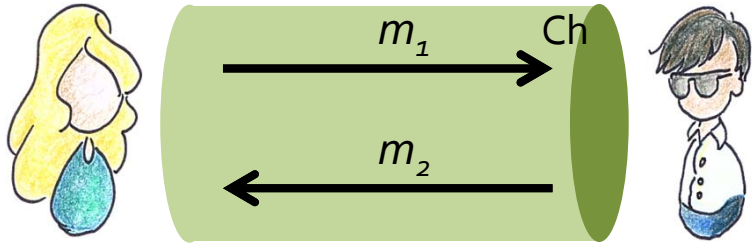
- Recall our application developer:
  - Perhaps he wants a drop-in replacement for TCP that's secure.
  - Actually, she might *just* want to send and receive some atomic messages and not a TCP-like stream.
- To what extent does AEAD meet these requirements?
- It might meet some of them, but not the complete list of possible – and conflicting – requirements we highlighted earlier.

AEAD ≠ secure channel

Enc(.,.,.)

+

Dec(.,.,.)



There's a significant semantic gap between AEAD's functionality and raw security guarantees, and the things a developer expects a secure channel to provide.

An example of the gap: cookie cutters

Bhargavan, Delignat-Lavaud, Fournet, Pironti, Strub 2014: cookie cutter attack on "HTTP over SSL/TLS".

- Attacker forces part of the HTTP header (e.g., cookie) to be cut off.
- Partial message/header arrives and might be misinterpreted.

Cookie cutters

Why doesn't this violate the proven integrity of SSL/TLS encryption?

```
6.2.1. Fragmentation

The record layer fragments information blocks
into TLSPlaintext records [...].  Client
message boundaries are not preserved in the
record layer (i.e., multiple client messages
of the same ContentType MAY be coalesced into
a single TLSPlaintext record, or a single
message MAY be fragmented across several
records).
```

RFC 5246 (TLS v1.2)

Cookie cutters

Why doesn't this violate the proven integrity of SSL/TLS encryption?

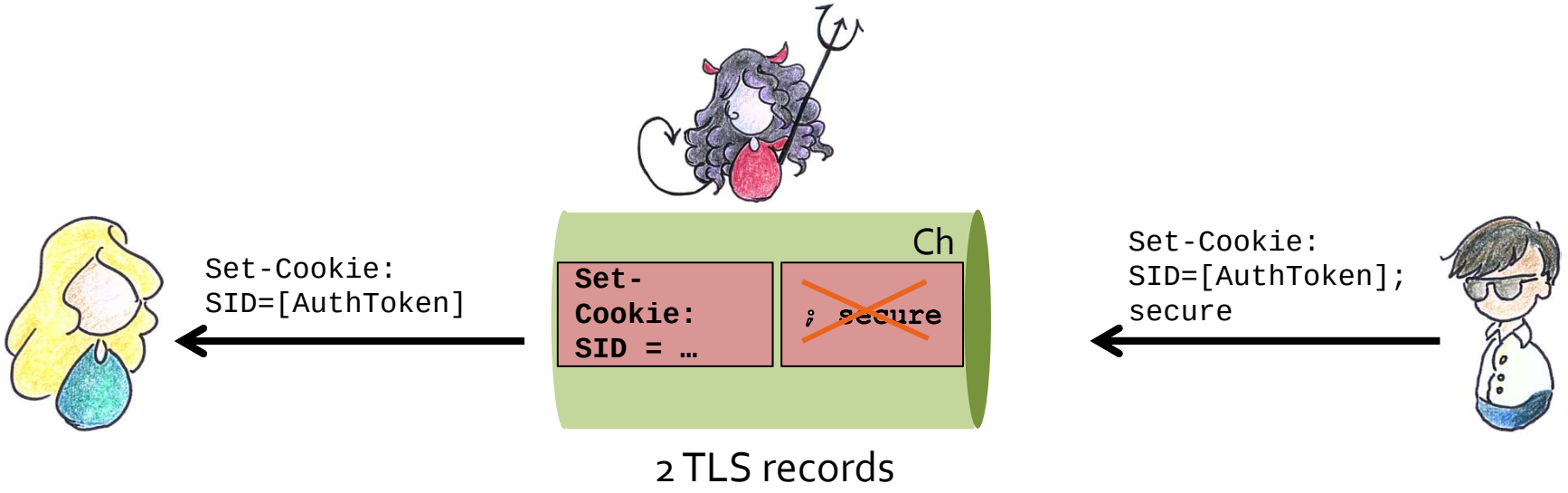
6.2.1. Fragmentation

The record layer fragments information blocks into TLSPlaintext records [...]. Client message boundaries are not preserved in the record layer (i.e., multiple client messages of the same ContentType MAY be coalesced into a single TLSPlaintext record, **or a single message MAY be fragmented across several records**).

RFC 5246 (TLS v1.2)

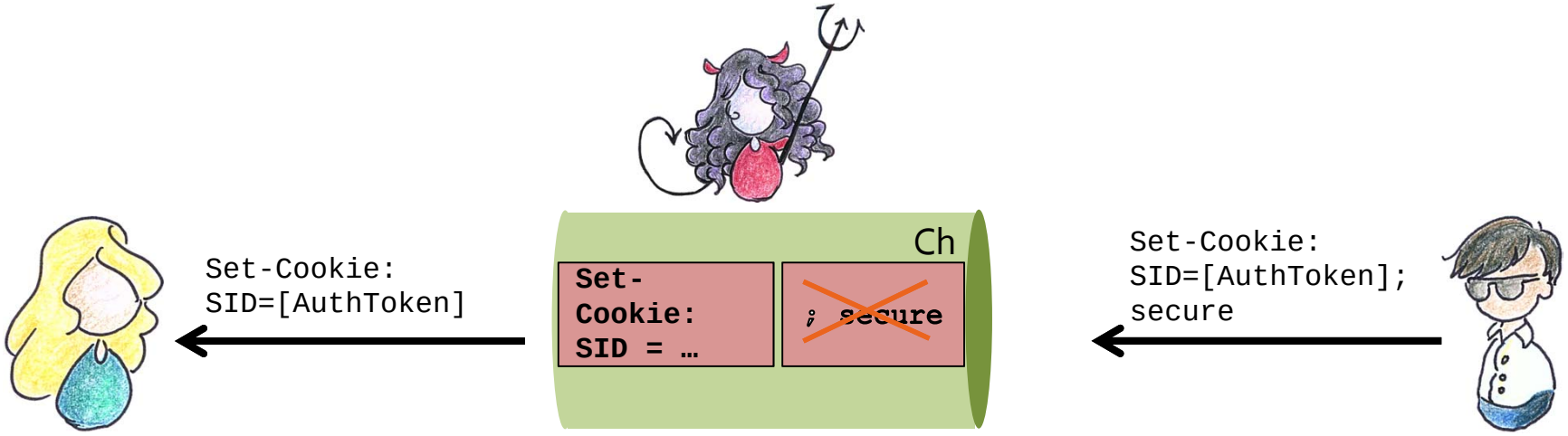
Cookie cutters

- So SSL/TLS can (and will) fragment when *sending.*
- Protocols like SSH have to handle fragmentation when *receiving* (but not usually when sending) – also a source of problems…



Cookie cutters

- It's up to the calling application to deal with message boundaries if it wants to use SSL/TLS for atomic message delivery.
- The cookie cutter attack relies on a buggy browser that does not check for correct HTTP message termination.
- This happens in practice –evidence that developers do not fully understand the interface provided by SSL/TLS.



What lies ahead

What lies ahead in the next two lectures

- Detailed discussion of symmetric crypto used in SSH and its security (failings).
- Ditto for SSL/TLS.
- Building better models for SSH-like and streaming secure channels.

"Now this is not the end. It is not even the beginning of the end. But it is, perhaps, the end of the beginning."

Closing remarks