

Establishing Secure Connections

A Cryptographer's Perspective and the Case of TLS 1.3



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Felix Günther

Technische Universität Darmstadt, Germany

based on joint work with

Jacqueline Brendel, Benjamin Dowling, Marc Fischlin, Britta Hale, Tibor Jager,
Christian Janson, Sebastian Lauer, Giorgia Azzurra Marson, Sogol Mazaheri,
Kenneth G. Paterson, Benedikt Schmidt, Douglas Stebila, Bogdan Warinschi



TECHNISCHE
UNIVERSITÄT
DARMSTADT



Cryptoplexity
Cryptography & Complexity Theory
Technische Universität Darmstadt
www.cryptoplexity.de

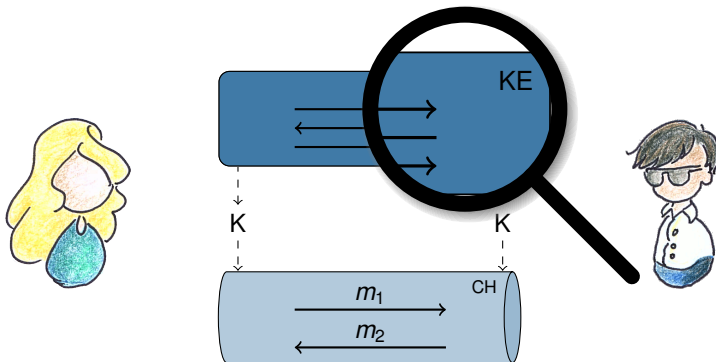


CROSSING

Secure Connections – Everywhere

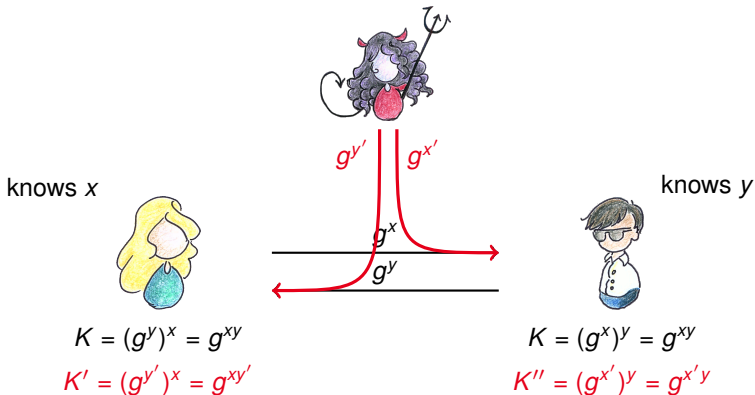


Secure Connections – Cryptographically



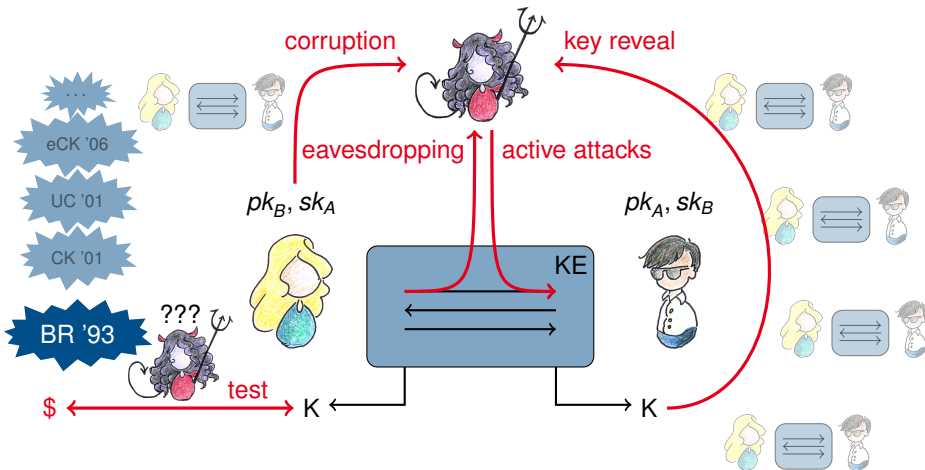
drawings by *Giorgia Azzurra Marson*

Key Exchange à la Diffie–Hellman (1976)

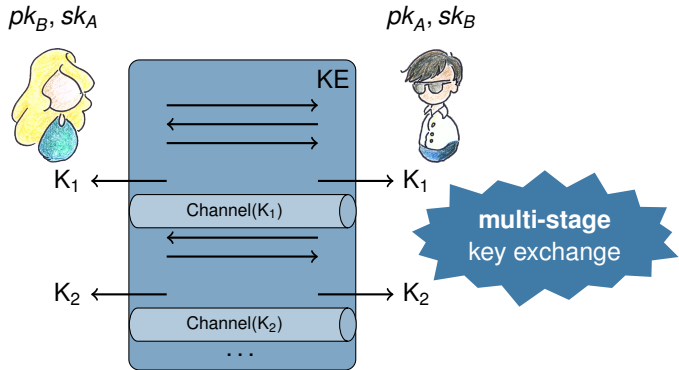


- ▶ **key secrecy:** given only g^x , g^y , key $K = g^{xy}$ remains secret
- ▶ **no authentication:** susceptible to man-in-the-middle attack

Key Exchange Security à la Bellare–Rogaway (1993)



But what if... ?



- ▶ key exchange establishes more than one key?
- ▶ ... even uses the intermediary keys within the key exchange or channel?
- ▶ not covered by classical key exchange models



QUIC (“Quick UDP Internet Connections”, Google 2013)

- ▶ “low-latency transport protocol with security equivalent to TLS”
- ▶ Diffie–Hellman-based key exchange
- ▶ aims at 0-RTT, i.e., immediately encrypts under intermediate key K_1
- ▶ later rekeys to forward-secret K_2
- ▶ intermediate key K_1 used to establish K_2 (i.e., in KE part)



Fischlin, Günther

Multi-Stage Key Exchange and the Case of Google’s QUIC Protocol

ACM CCS 2014

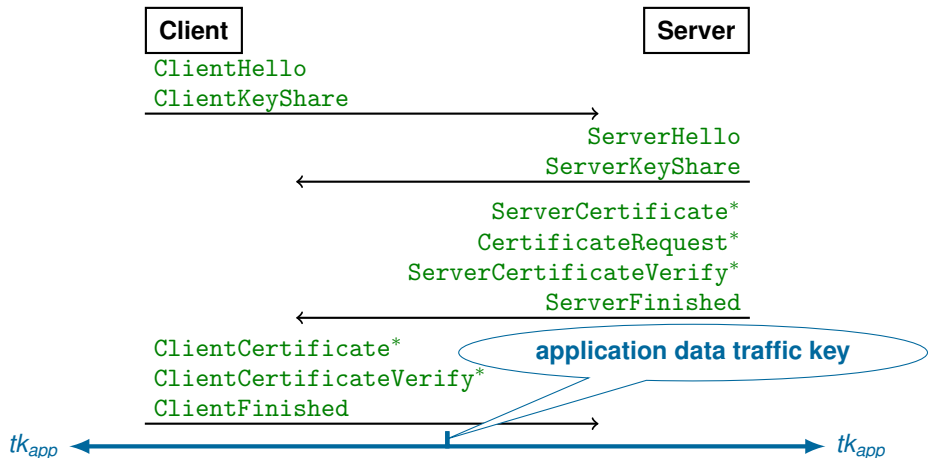
TLS 1.3

- ▶ next TLS version, **currently being specified**
 - ▶ now in **IETF Working Group Last Call (WGLC)**
 - ▶ latest: draft-18, Oct 2016

- ▶ several **substantial cryptographic changes** (compared to TLS 1.2), incl.
 1. **encrypting some handshake messages** with intermediate session key
 2. **deprecating some crypto algorithms** (RC4, SHA-1, key transport, MtEE, etc.)
 3. using only **AEAD schemes** for the record layer encryption
 4. providing reduced-latency **0-RTT handshake**
 5. ...

TLS 1.3 Full Handshake (simplified)

draft-ietf-tls-tls13-10 (Oct 2015)



... actually, there is more ...

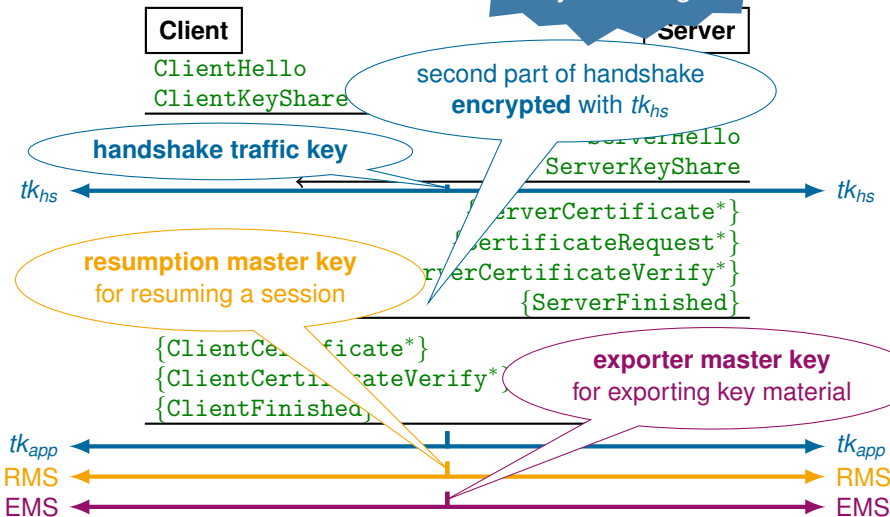
TLS 1.3 Full Handshake (still simplified)

draft-ietf-tls-tls13-10 (Oct 2015)



TECHNISCHE
UNIVERSITÄT
DARMSTADT

multi-stage
key exchange



Multi-Stage Key Exchange Analyses of TLS 1.3 Handshake Protocol Candidates

- ▶ full (DH) and preshared-key (resumption) handshakes (draft-10 & earlier)



Dowling, Fischlin, Günther, Stebila

A Cryptographic Analysis of the TLS 1.3 ... Handshake Protocol ...

ACM CCS 2015, TRON workshop @ NDSS 2016

- ▶ 0-RTT handshake, DH-based (draft-12) & PSK-based (draft-14)



Fischlin, Günther

Replay Attacks on Zero Round-Trip Time: The Case of the TLS 1.3 Handshake Candidates

IEEE EuroS&P 2017

- ▶ analyses of work-in-progress drafts (i.e., not definitive)
 - ▶ contribution to and involved in working group discussion
 - ▶ and part of a **great community effort of many people**



STANDARD UNDER CONSTRUCTION

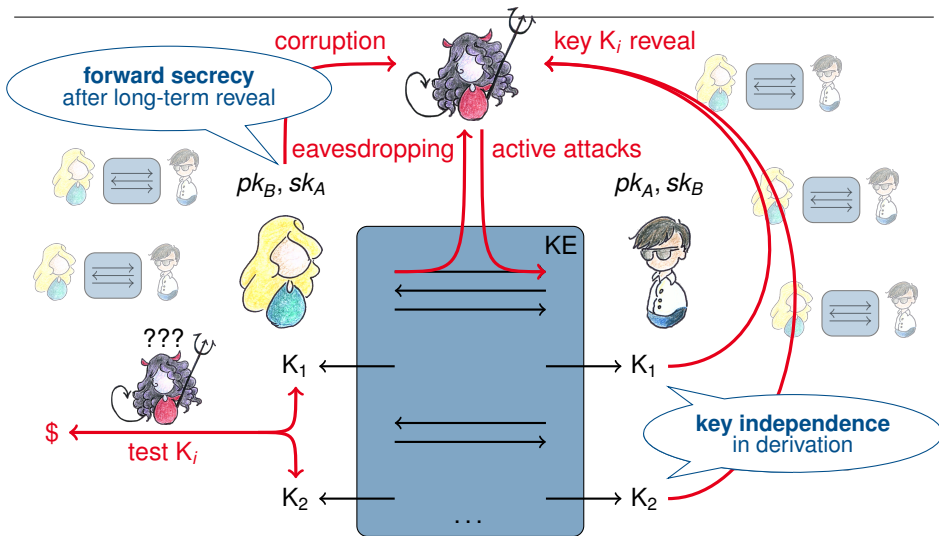
... and Many More Analyses

- ▶ Arai, Matsuo [CELLOS] (TLS mailing list 2016): [ProVerif Analysis](#)
- ▶ Badertscher, Matt, Maurer, Rogaway, Tackmann (ProvSec 2015): [Record Layer](#)
- ▶ Bellare, Tackmann (Crypto 2016): [Security of AES-GCM Usage](#)
- ▶ Beurdouche, Bhargavan, Blanchet, Delignat-Lavaud, Fournet, Ishtiaq, Kobeissi, Kohlweiss, Pan, Protzenko, Rastogi, Swamy, Zanella-Bguelin, Zinzindohoué [INRIA/Microsoft] (TRON 2016, S&P 2017, ...): [Verified Implementations of Handshake and Record Layer](#)
- ▶ Bhargavan, Brzuska, Fournet, Green, Kohlweiss, Zanella-Beguellin (S&P 2016): [Downgrade Resilience](#)
- ▶ Cremers, Horvat, Scott, van der Merwe (S&P 2016): [Tamarin Analysis](#)
- ▶ Jager, Schwenk, Somorovsky (CCS 2015): [Bleichenbacher's Attack](#)
- ▶ Kohlweiss, Maurer, Onete, Tackmann, Venturi (ePrint 2015): [Constructive Crypto](#)
- ▶ Krawczyk, Wee (EuroS&P 2016): [OPTLS](#)
- ▶ Krawczyk (CCS 2016): [Unilateral-to-Mutual Authentication Compiler](#)
- ▶ Li, Xu, Zhang, Feng, Hu (S&P 2016): [Multi-Handshake Security](#)
- ▶ ...

(alphabetical order)

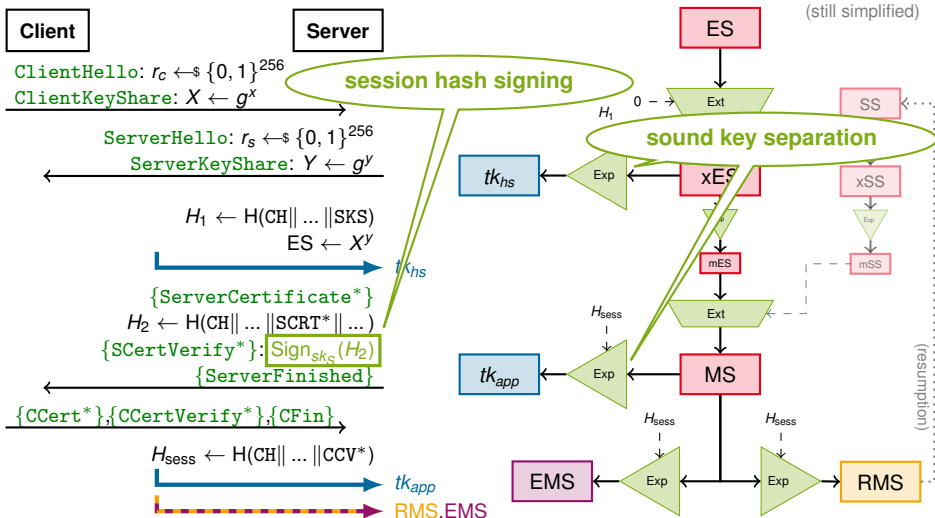
Multi-Stage Key Exchange Security

game-based model, "provable security" paradigm



TLS 1.3 Handshake Security

draft-10 Full Handshake



TLS 1.3 Handshake Security

draft-10 Full Handshake

We show that the draft-10 **full (EC)DHE handshake** establishes

- ▶ **random-looking keys** (tk_{hs} , tk_{app} , **RMS**, **EMS**)
tolerating adversary that corrupts other users and reveals other session keys
- ▶ **forward secrecy** for all these keys
- ▶ **concurrent security** of anonymous, unilateral, mutual authentication
- ▶ **key independence** (leakage of traffic/resumption/exporter keys in same session does not compromise each other's security)

assuming

- ▶ **collision-resistant** hashing
- ▶ **unforgeable** signatures
- ▶ **HKDF** is pseudorandom function
- ▶ **PRF-ODH** assumption holds

standard key exchange security
under **standard(-model) assumptions**



Brendel, Fischlin, Günther, Janson

PRF-ODH: Relations, Instantiations, and Impossibility Results

TLS 1.3 Handshake Security


Further Modes & Beyond



TECHNISCHE
UNIVERSITÄT
DARMSTADT

- ▶ **PSK/PSK-DHE handshake** (draft-10)
 - ▶ similar results as for full handshake
 - ▶ DHE variant enables **forward secrecy**

 - ▶ **0-RTT handshake** (draft-12/14)
 - ▶ 0-RTT messages/key can be **replayed**
 - ▶ **weaker forward secrecy** guarantees

 - ▶ **Key confirmation properties** (draft-10)
 - ▶ assurance that communication partner actually **holds the shared key**
-  Fischlin, Günther, Schmidt, Warinschi
Key Confirmation in Key Exchange: A Formal Treatment and Implications for TLS 1.3
IEEE S&P 2016

More Key Exchange Challenges



▶ **TLS 1.3: Post-handshake messages & Early (0.5-RTT) server data**

- ▶ post-handshake late client authentication, key updates, and more
- ▶ early server data before handshake is over
- ▶ changing authentication of session key in use
- ▶ beyond what classical key exchange models capture

[Krawczyk'16]: can work as
Unilateral-to-Mutual Compiler

▶ **Signal: Ratcheting in Secure Messaging**

- ▶ frequent key updates / new session key with every message
- ▶ advanced security properties, future/post-compromise security

[Cohn-Gordon CDGS'17]
security proof in MSKE model

▶ **Forward-secret 0-RTT key exchange**

- ▶ in current designs, forward secrecy is sacrificed in 0-RTT modes
- ▶ new idea: leverage puncturable forward-secret encryption [Green, Miers'15]
- ▶ enables fully forward-secret 0-RTT (generically from any HIBKEM)

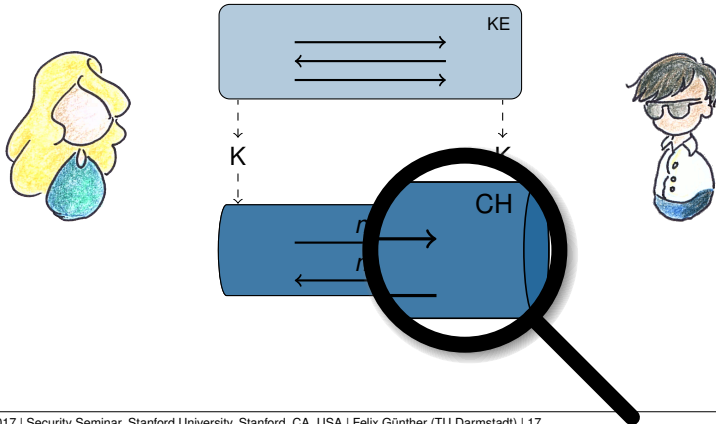


Günther, Hale, Jager, Lauer

0-RTT Key Exchange with Full Forward Secrecy

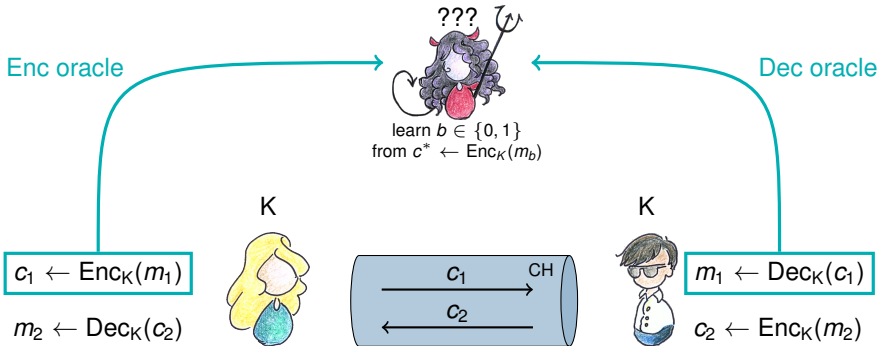
Eurocrypt 2017

Secure Connections – Cryptographically



On the Origin of Channel Models

Confidentiality

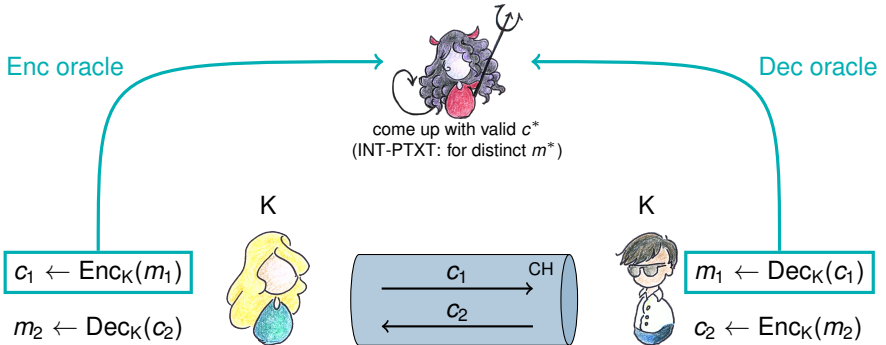


IND-CPA
[Goldwasser, Micali'84]

IND-CCA
[Naor, Yung'90], [Rackoff, Simon'91]

On the Origin of Channel Models

Integrity



Authenticated Encryption

IND-CPA + INT-CTXT
(\Rightarrow IND-CCA)

INT-PTXT

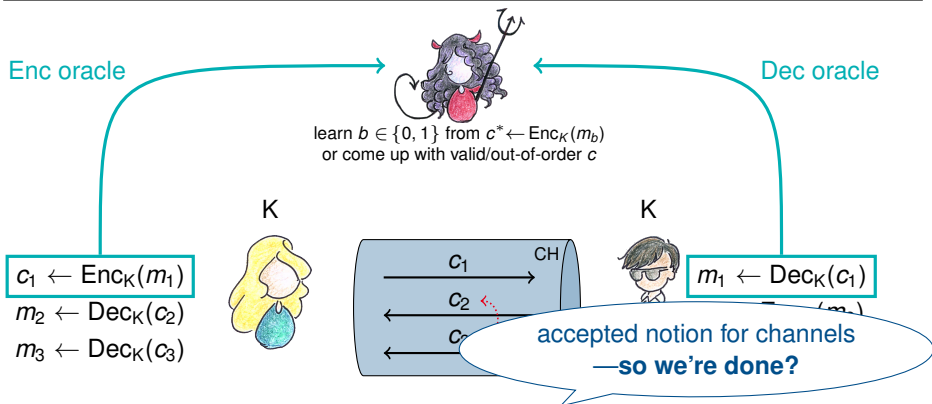
[Bellare, Namprempre'00]

INT-CTXT

[Bellare, Rogaway'00]

On the Origin of Channel Models

Stateful Authenticated Encryption



Stateful Authenticated Encryption

used to analyze SSH

IND-sfCCA

[Bellare, Kohno, Namprempre'02]

INT-sfCTXT

[Albrecht, Paterson, Watson'09]: **plaintext recovery attack against SSH**
(SSH Binary Packet Protocol with CBC-mode Encode-then-Encrypt&MAC)

- ▶ **adversary feeds ciphertext in *block-wise*** (via TCP fragmentation)
- ▶ observable MAC failure can be used to leak plaintext → **confidentiality break**

Wait. . .

- ▶ SSH was proven IND-sfCCA and INT-sfCTXT secure! [BKN'02]
- ▶ . . . but these only allow ***atomic* ciphertexts in Dec oracle**



On the Origin of Channel Models

Symmetric Encryption Supporting Fragmentation



Symmetric Encryption Supporting Fragmentation

[Boldyreva, Degabriele, Paterson, Stam'12]

- ▶ general security model for **ciphertext fragmentation**
- ▶ standard Enc algorithm (and left-or-right oracle)
- ▶ Dec algorithm obtains **ciphertext fragments**, reassembles **original messages**

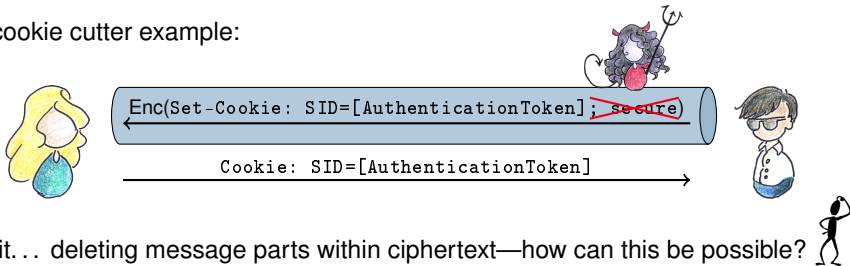
Are we there yet?

Attack on TLS

Cutting Cookies

[Bhargavan, Delignat-Lavaud, Fournet, Pironti, Strub'14]: **cookie cutter attack**

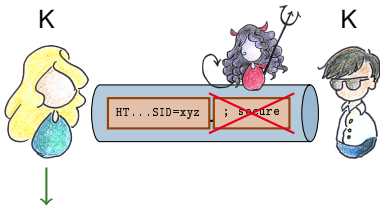
- ▶ attacker **truncates TLS connection** by closing underlying TCP connection
- ▶ forces part of the HTTP header (e.g., cookie) to be cut off
- ▶ **partial message/header arrives** and might be misinterpreted
- ▶ cookie cutter example:



Wait... deleting message parts within ciphertext—how can this be possible?

Cookie Cutter Attack

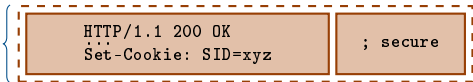
A Closer Look



HTTP/1.1 200 OK
...
Set-Cookie: SID=xyz

2 TLS records

```
c ← Enc(HTTP/1.1 200 OK  
...  
Set-Cookie: SID=xyz; secure)  
  
#include <openssl/ssl.h>  
SSL_write("HTTP/1.1 200 OK  
...  
Set-Cookie: SID=xyz; secure")
```



- ▶ fragmentation in TLS is **implementation-specific**
- ▶ adversary can potentially enforce a split at any point
→ receiver sees **arbitrarily fragmented messages / no message boundaries**

An Interface Misunderstanding: Data Is a Stream!

... and TLS is not alone

- ▶ That behavior is actually okay—and specified:

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

- ▶ TLS never promised to treat messages atomically!
- ▶ indeed, many important channel protocols treat data as a stream
 - ▶ TLS
 - ▶ SSH tunnel-mode
 - ▶ QUIC

- ▶ so, there's a **gap** between what
channel models capture



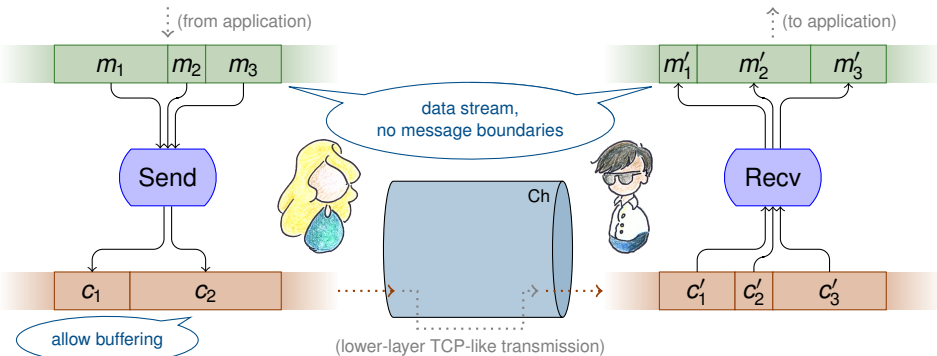
AE \neq secure channel

and channels expose to the application

Stream-Based Channels

Intuition and Security Notions

 Fischlin, Günther, Marson, Paterson
Data Is a Stream: Security of Stream-Based Channels
Crypto 2015



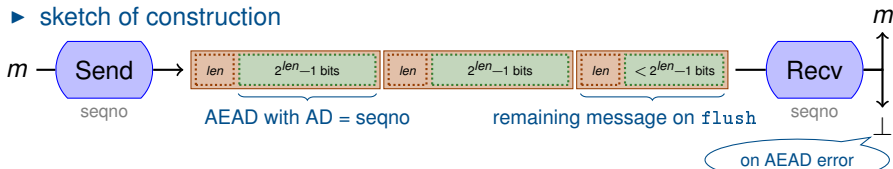
- ▶ adapted confidentiality and integrity notions for the stream-based setting

Stream-Based Channels

Generic Construction

- ▶ secure stream-based channels can be built
 - ▶ based on authenticated encryption with associated data (AEAD)
 - ▶ achieving strong IND-CCFA confidentiality
 - ▶ achieving strong INT-CST integrity

▶ sketch of construction



- ▶ close to TLS record layer design using AEAD (providing some validation)
 - ✓ sequence number authenticated, but not sent
 - ✓ sent length field, unauthenticated (in TLS 1.3)
 - ✗ TLS additionally includes, e.g., content type (sent authenticated)

Further Properties

- ▶ Length-hiding [Paterson, Ristenpart, Shrimpton'11] for streams?
- ▶ Multiplexing of data (explicitly in QUIC, implicitly in TLS)
- ▶ How to safely encode atomic messages in a stream?
(upcoming extended version)

TLS 1.3 Record Protocol

- ▶ employs several traffic keys in the same protocol (for handshake + data)
- ▶ key switching requires care to prevent truncation attacks [miTLS team]



Günther, Mazaheri

A Formal Treatment of Multi-key Channels

[miTLS team'16]: verified
TLS 1.3 Record Layer implementation

Conclusions

- ▶ **basic properties** of key exchange and secure channels are **well-understood** ?
- ▶ but **advanced properties** pose **new challenges** for security models

▶ in this talk:

- ▶ **multi-stage key exchange** (QUIC, TLS 1.3)
- ▶ **stream-based channels** (generic, TLS)
- ▶ **positive:** interaction of **crypto, formal methods, and engineering** communities in development of TLS 1.3

