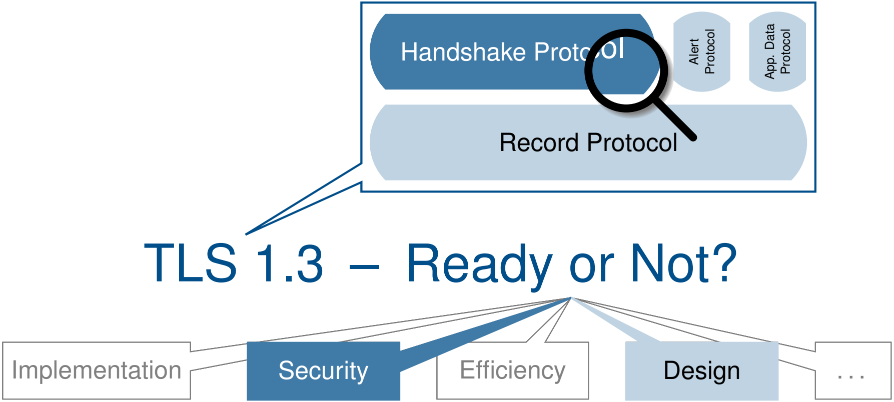
A Cryptographic Analysis of the TLS 1.3 draft-10 Full and Pre-shared Key Handshake Protocol

Felix Günther

Technische Universität Darmstadt, Germany

joint work with Benjamin Dowling, Marc Fischlin, and Douglas Stebila

TRON(?)



TLS 1.3 – Ready or Not?

Our answer: Yes (almost).

Security (Analyses) of TLS (< 1.3)

(arbitrary selection from recent years)

trunc. handshake [GMP+,MSW]	2008	2008	**TLS 1.2**
		2009	Insecure Renegotiation [RayDis]
record protocol (LHAE) [PRS]	2011	2011	BEAST [DuoRiz]
full TLS-DHE (ACCE) [JKSS]	2012	2012	CRIME [DuoRiz]
verified miTLS impl. [BFK+]	2013	2013	Lucky 13 [AlFPat]
TLS-DH, TLS-RSA-CCA [KSS]			RC4 biases [ABP+]
multiple ciphersuites [KPW]			
TLS 1.2 handshake [BFK+]	2014	2014	Triple Handshake [BDF+]
pre-shared key suites [LSY+]			Heartbleed [Cod]
(de-)constructing TLS [KMO+]			POODLE [MDK]
		2015	SMACK + FREAK [BBD+]
			Logjam [ABD+]
		2016	SLOTH [BhaLeu]

TLS 1.3

- next TLS version, currently being specified (latest: `draft-11`, Dec 2015)
- several substantial cryptographic changes (compared to TLS 1.2), incl.
  1. encrypting some handshake messages with intermediate session key
  2. signing the entire transcript when authenticating
  3. including handshake message hashes in key calculations
  4. generating `Finished` messages with seperate key
  5. deprecating some crypto algorithms (RC4, SHA-1, key transport, MtEE, etc.)
  6. using only AEAD schemes for the record layer encryption
  7. switch to HKDF for key derivation
  8. providing reduced-latency 0-RTT handshake
- in large part meant to address previous attacks and design weaknesses
- analysis can check absence of unexpected cryptographic weaknesses — desirably before standardization

Our Scope

TECHNISCHE
UNIVERSITÄT
DARMSTADT

- `draft-10` (Oct 2015)
  - updating our earlier analysis of `draft-05` and `draft-dh` (of May 2015, @CCS 2015)
  - TLS 1.3 is work in progress
  - contribution to ongoing discussion rather than definitive analysis of TLS 1.3

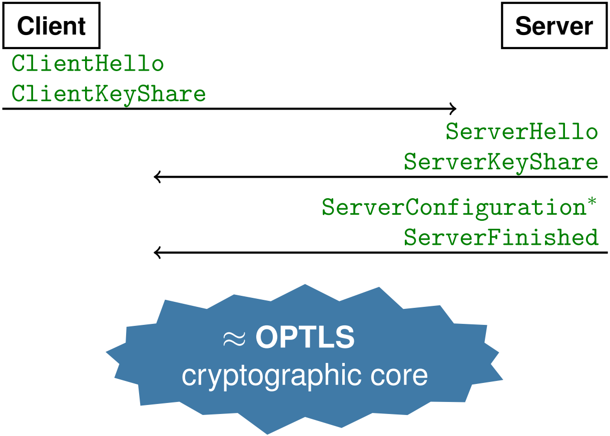
STANDARD UNDER CONSTRUCTION

- focus on full and preshared-key handshakes (separately)
  - **(EC)DHE full** handshake
  - **PSK** / **PSK-(EC)DHE** preshared-key/resumption handshake
  - don't capture 0-RTT handshake

- we don't analyze the Record Protocol
  - but follow a compositional approach that allows independent treatment (see later)

TLS 1.3 Full Handshake (simplified)

`draft-ietf-tls-tls13-10`



Client

Server

ClientHello

ClientKeyShare

ServerHello

ServerKeyShare

ServerConfiguration*

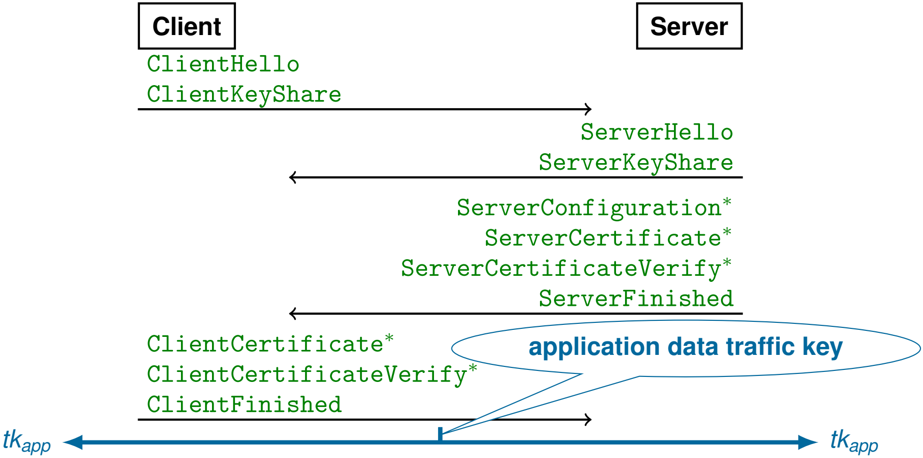
ServerFinished

≈ **OPTLS**

cryptographic core

TLS 1.3 Full Handshake (simplified)

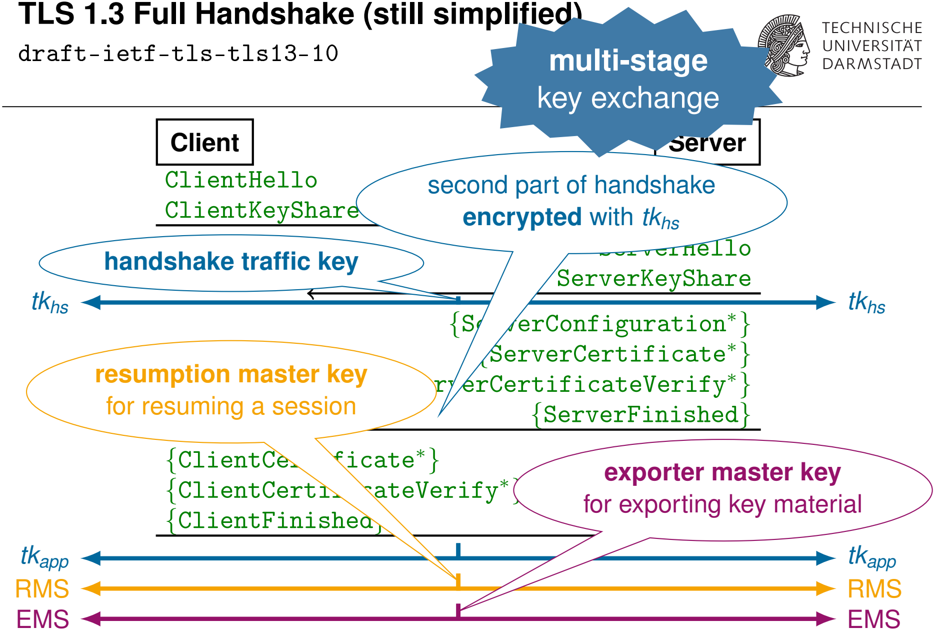
`draft-ietf-tls-tls13-10`



… actually, there is more …

TLS 1.3 Full Handshake (still simplified)

`draft-ietf-tls-tls13-10`



multi-stage key exchange

Client

Server

ClientHello
ClientKeyShare

second part of handshake **encrypted** with tk_{hs}

ServerHello
ServerKeyShare

handshake traffic key

tk_{hs} tk_{hs}

{ServerConfiguration*}
{ServerCertificate*}
{ServerCertificateVerify*}
{ServerFinished}

resumption master key for resuming a session

{ClientCertificate*}
{ClientCertificateVerify*}
{ClientFinished}

exporter master key for exporting key material

tk_{app} tk_{app}

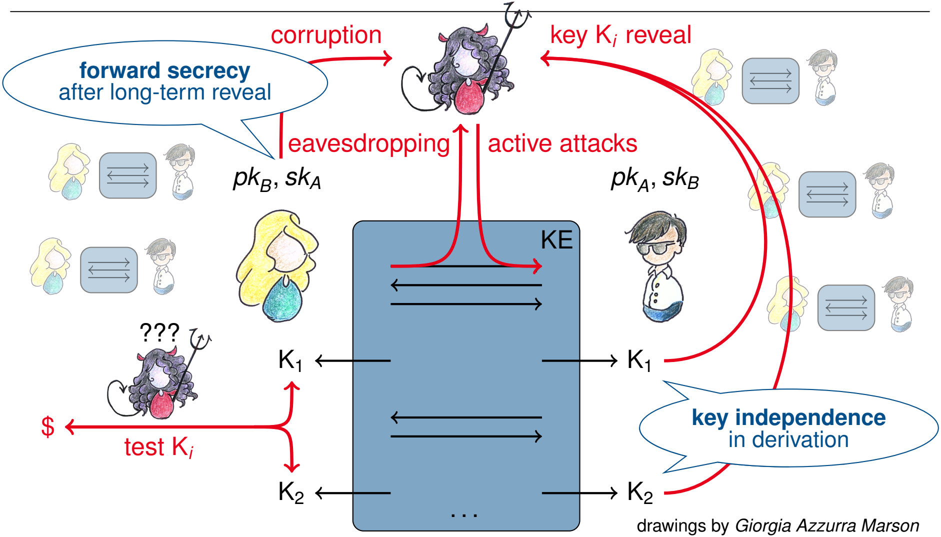
RMS RMS

EMS EMS

Multi-Stage Key Exchange (Security)

(Fischlin, Günther @ CCS 2014)

game-based model, “provable security” paradigm



drawings by *Giorgia Azzurra Marson*

Modeling Multi-Stage Key Exchange

Further Aspects

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Extensions in This Work

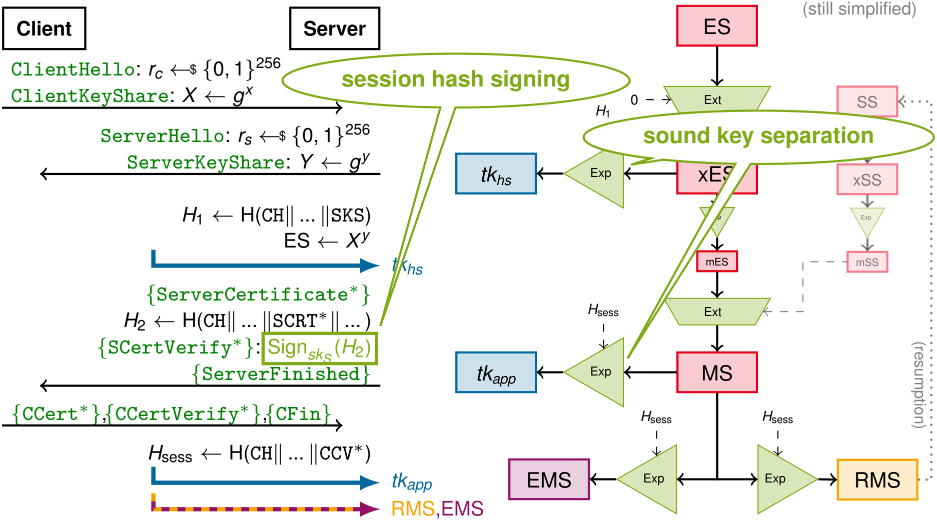
- **unauthenticated keys/stages** (beyond unilateral/mutual authentication)
  TLS 1.3: neither server nor client send a certificate
- **concurrent execution of different authentication types**
  TLS 1.3: anonymous, server authenticates, server+client authenticate
- **post-specified peers**
  TLS 1.3: parties learn peer's identity (= *pk*) only within handshake
- **pre-shared secret key variant**
  TLS 1.3: PSK/PSK-DHE handshake modes from preshared secrets (RMS)

Secret Compromise Paradigm

- We consider leakage of:
  - **long-term/static secret keys** (signing keys of server/client)
    high potential of compromise, necessary to model forward secrecy
  - **session keys** (traffic keys tk_{hs} and tk_{app}, RMS, EMS)
    outputs of handshake used *outside* the key exchange for encryption, resumption, exporting
- We do not permit leakage of:
  - **ephemeral secret keys** (DH exponents, signature randomness)
  - **internal values / session state** (master secrets, intermediate values)
    TLS 1.3 full/PSK handshakes not designed to be secure against such compromise
  - **semi-static secret keys** (s in semi-static g^s used for 0-RTT)
    security of full/PSK handshakes independent of this value
    but: in analysis of **0-RTT handshake** this type of leakage needs to be considered!

Security of the `draft-10` Full Handshake

(still simplified)



Client **Server**

ClientHello: $r_c \xleftarrow{\$} \{0,1\}^{256}$

ClientKeyShare: $X \leftarrow g^x$

ServerHello: $r_s \xleftarrow{\$} \{0,1\}^{256}$

ServerKeyShare: $Y \leftarrow g^y$

$H_1 \leftarrow \mathrm{H}(\mathtt{CH}\|\ldots\|\mathtt{SKS})$

$\mathrm{ES} \leftarrow X^y$

tk_{hs}

$\{\mathtt{ServerCertificate}^*\}$

$H_2 \leftarrow \mathrm{H}(\mathtt{CH}\|\ldots\|\mathtt{SCRT}^*\|\ldots)$

$\{\mathtt{SCertVerify}^*\}$: $\boxed{\mathrm{Sign}_{sk_S}(H_2)}$

$\{\mathtt{ServerFinished}\}$

$\{\mathtt{CCert}^*\}$,$\{\mathtt{CCertVerify}^*\}$,$\{\mathtt{CFin}\}$

$H_{\mathrm{sess}} \leftarrow \mathrm{H}(\mathtt{CH}\|\ldots\|\mathtt{CCV}^*)$

tk_{app}

RMS,EMS

session hash signing

sound key separation

Security of the `draft-10` Full Handshake

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

- random-looking keys (tk_{hs}, tk_{app}, RMS, EMS) with adversary allowed to corrupt other users and reveal 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
- Decisional Diffie–Hellman is hard
- HKDF is pseudorandom function

} **standard KE security under standard assumptions**

Security of the `draft-10` PSK Handshakes

PSK

- random-looking keys (tk_{hs}, tk_{app}, EMS)
- mutual authentication (down to RMS)
- key independence
- no forward secrecy

PSK-DHE

- random-looking keys (tk_{hs}, tk_{app}, EMS)
- mutual authentication (down to RMS)
- key independence
- forward secrecy for all keys

Under similar standard assumptions:

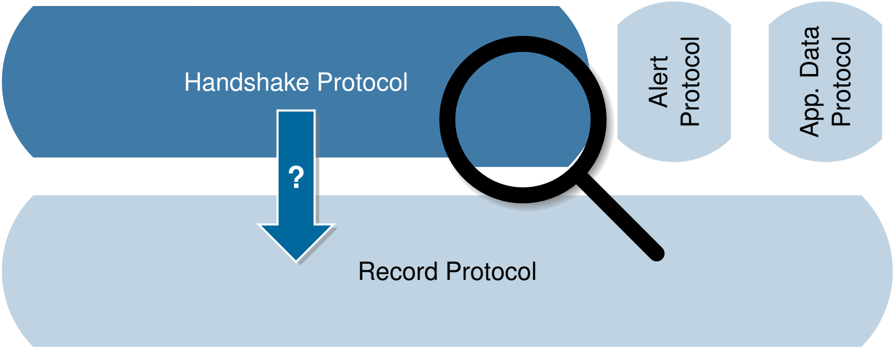
PSK:

- collision-resistant hashing
- HKDF is pseudorandom function

PSK-DHE:

- collision-resistant hashing
- HKDF is pseudorandom function
- HMAC is unforgeable
- Decisional Diffie–Hellman is hard

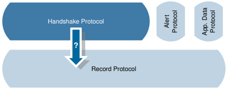
Composition



- we established security of the keys derived in the full and PSK handshakes
- what about the usage of those keys, e.g., in the Record Protocol?

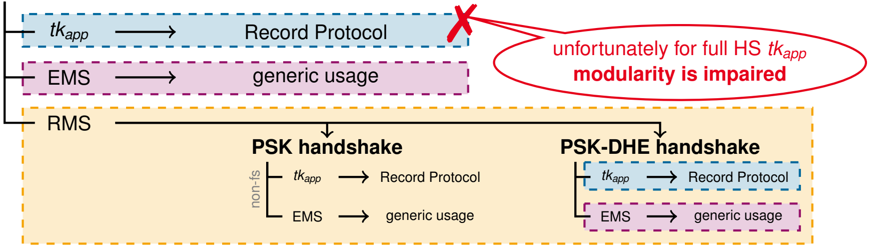
Composition

- we follow a modular, compositional approach
  (extending [FG'14])



- we show: using final, forward-secret keys in any symmetric-key protocol is safe
- i.e., Record Protocol can be analyzed independently
- also captures use of exported EMS and RMS for resumption (cascading)

full (EC)DHE handshake



tk_{app} ⟶ Record Protocol

EMS ⟶ generic usage

unfortunately for full HS tk_{app}
modularity is impaired

RMS

PSK handshake

non-fs

tk_{app} ⟶ Record Protocol

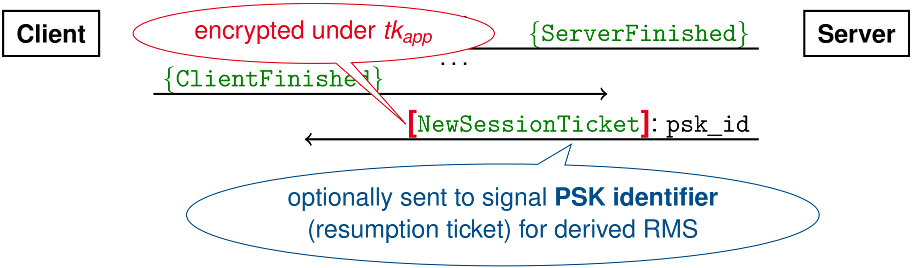
EMS ⟶ generic usage

PSK-DHE handshake

tk_{app} ⟶ Record Protocol

EMS ⟶ generic usage

The NewSessionTicket Issue



Client

Server

encrypted under tk_{app}

{ServerFinished}

...

{ClientFinished}

[NewSessionTicket]: psk_id

optionally sent to signal **PSK identifier** (resumption ticket) for derived RMS

- final/main **session key tk_{app} used within handshake**
- reminds of TLS 1.2 Finished message (requiring monolithic/special analysis)
- in similar spirit as current WG discussion of not changing $tk_{hs} \rightarrow tk_{app}$

- note: there is no immediate attack arising from this . . .
- . . . but means handshake design does not achieve generic KE security
- violates modularity between handshake and record layer (in draft-10)
- draft-11: less clear whether part of handshake, can be sent much later

The NewSessionTicket Issue

Effects and Potential Alternatives (if part of handshake)

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Multi-stage key exchange model allows to separate:

- ✓ **key secrecy** for tk_{app} still given
- ✗ **generic composition** using tk_{app} not possible
  - prevents modular combination with independent analysis of Record Protocol
  - requires analysis to be reworked for changes/new aspects in Record Protocol

Potential alternatives:

1. Send `NewSessionTicket` earlier encrypted under tk_{hs}.
   - precludes some usage scenarios, particularly (server) state encoding in ticket
2. Send `NewSessionTicket` as final message, encrypted under tk_{hs}.
   - tk_{hs} only implicitly authenticated, but RMS is anyway
3. Send `NewSessionTicket` as final message, encrypted under new tk_{nst}.
   - keeps authentication level, requires extra key switching
   - may be extendable to “control channel” for post-handshake messages (`draft-11`)

Don’t advocate a particular option, balancing of constraints best left to TLS WG.

Main Comments on TLS 1.3 from Our Analysis

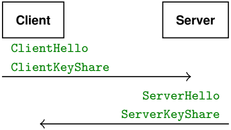
TECHNISCHE
UNIVERSITÄT
DARMSTADT

1. **Soundness of key separation**
   - ► separate keys for handshake and application data encryption*
   - ► allows to achieve standard key secrecy notions using standard assumptions
2. **Key independence**
   - ► unique labels in key derivation
   - ► neither key affected by other's compromise → allows compositional approach
3. **Session hash in online signatures**
   - ► full transcript signed in `CertificateVerify` messages
   - ► makes proof easier and allows for standard assumptions
4. **Encryption of handshake messages**
   - ► tk_{hs} secure against passive adversaries, hence can indeed increase privacy
   - ► we confirm there are no negative effects on main key secrecy goal
5. `NewSessionTicket` **encrypted under application traffic key*** (in handshake)
   - ► violates modularity between handshake and record layer
   - ► prevents generic composition for tk_{app} of full handshake

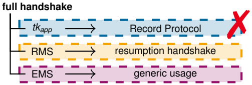
Summary

We

- analyze TLS 1.3 `draft-10`
  full (EC)DHE, PSK, and PSK-DHE handshake
  in an extended multi-stage key exchange model



- establish standard key secrecy notions
  - with forward secrecy (for full/PSK-DHE)
  - running all authentication modes concurrently
  - under standard assumptions



- extend composition result for modular analysis
- exhibit `NewSessionTicket` message (in handshake) violates modularity

full versions @ IACR ePrint

- `http://ia.cr/2016/081` (`draft-10`)
- `http://ia.cr/2015/914` (`draft-05` + `draft-dh`)

Thank You!