ETH zürich

Applied Cryptography
Spring Semester 2023

Lectures 33 and 34
Transport Layer Security

Felix Günther

Applied Cryptography Group

https://appliedcrypto.ethz.ch/

Overview of this lecture

ETH zürich

Part I Introduction to TLS

- The Transport Layer Security (TLS) protocol: history, design, and flaws.
- Why TLS 1.3 and what does it change?

Part II TLS 1.3 Design & Security Analyses

- TLS 1.3: the technical details
- Understanding the security of TLS 1.3: some research bits

Part I

Introduction to TLS

Transport Layer Security (TLS)

ETH zürich

TLS allows client/server applications to communicate over the Internet in a way that is designed to prevent eavesdropping, tampering, and message forgery.

TLS 1.3 [RFC 8446]

Transport Layer Security (TLS)

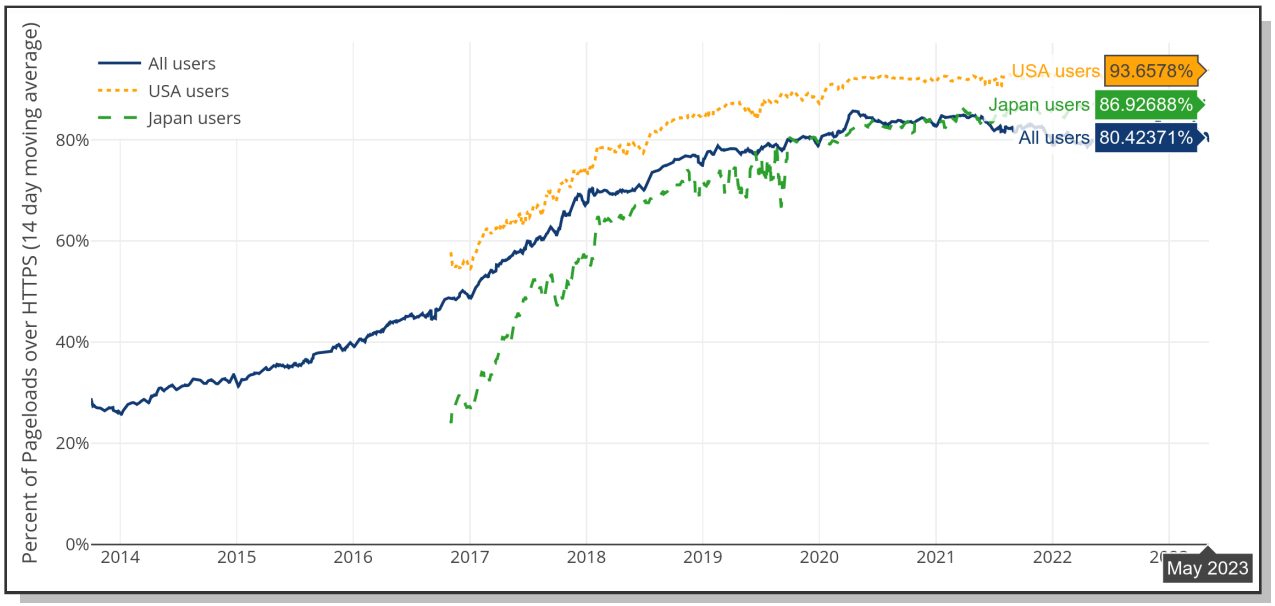
ETH zürich

TLS allows client/server applications to communicate over the Internet in a way that is designed to prevent eavesdropping, tampering, and message forgery.

TLS 1.3 [RFC 8446]

HTTPS page loads

- 80–93% Firefox (May 23)
  (Telemetry)



Transport Layer Security (TLS)

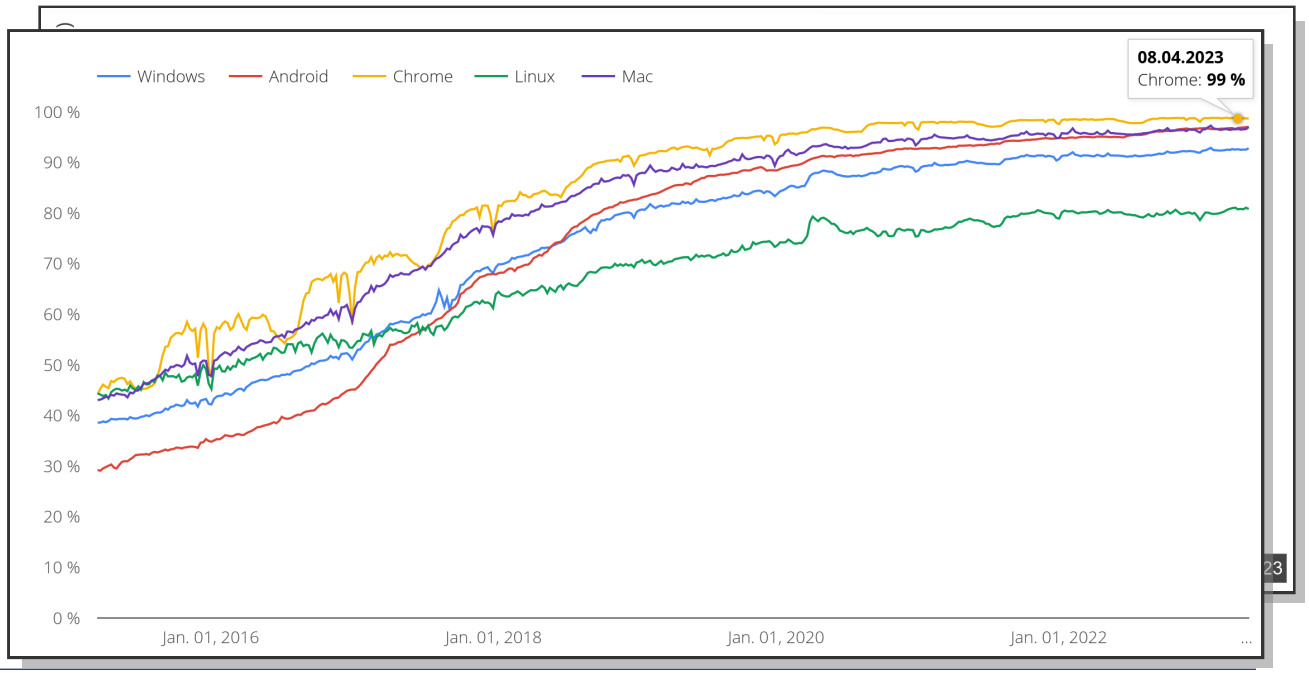
ETH zürich

TLS allows client/server applications to communicate over the Internet in a way that is designed to prevent eavesdropping, tampering, and message forgery.

TLS 1.3 [RFC 8446]

HTTPS page loads

- 80–93% Firefox (May 23)
  (Telemetry)
- 81–99% Chrome (May 23)
  (Transparency Report)



Transport Layer Security (TLS)

The SSL/TLS history . . .

ETH zürich

Transport Layer Security (TLS)

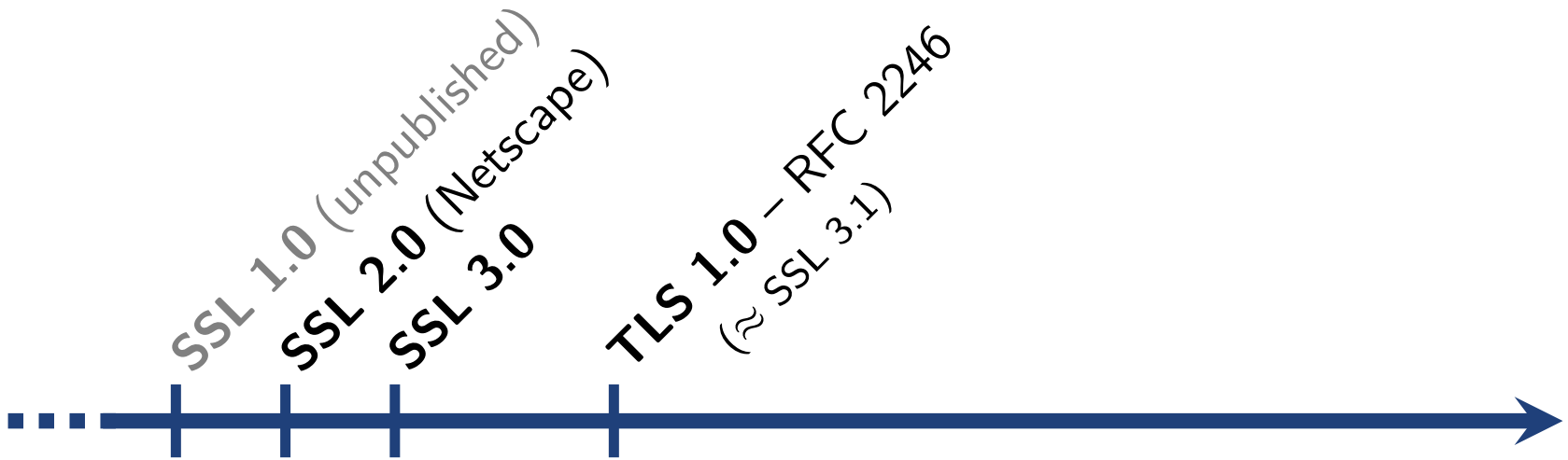
The SSL/TLS history . . .

ETH zürich

Transport Layer Security (TLS)

The SSL/TLS history . . .

ETH zürich

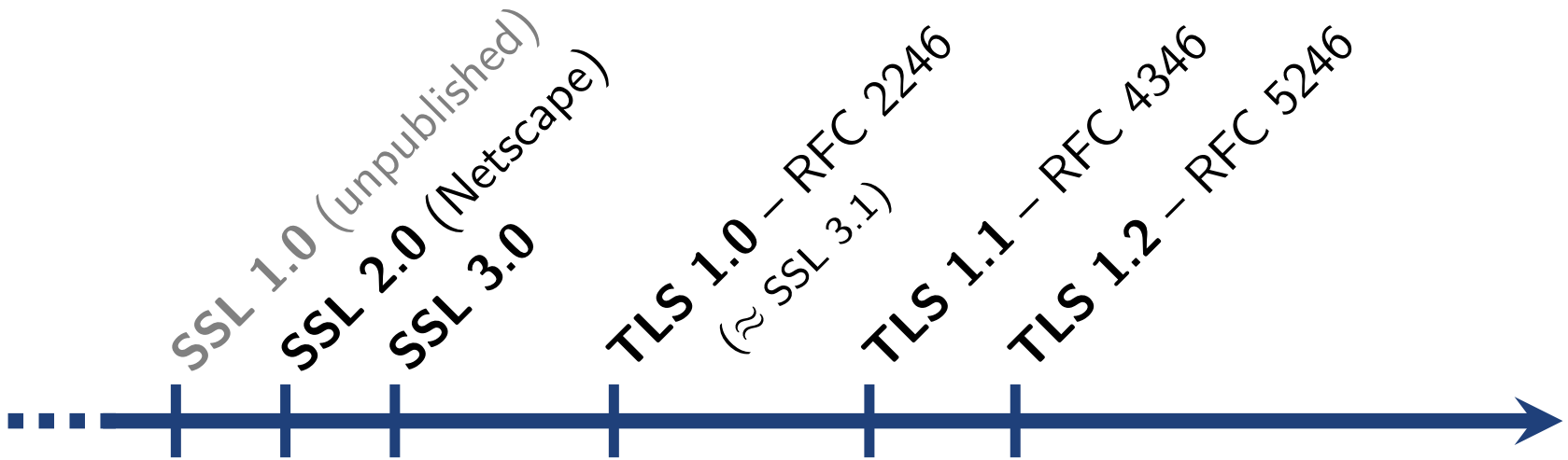


adopted by IETF TLS working group
main editor: Eric Rescorla

Transport Layer Security (TLS)

The SSL/TLS history . . .

ETH zürich



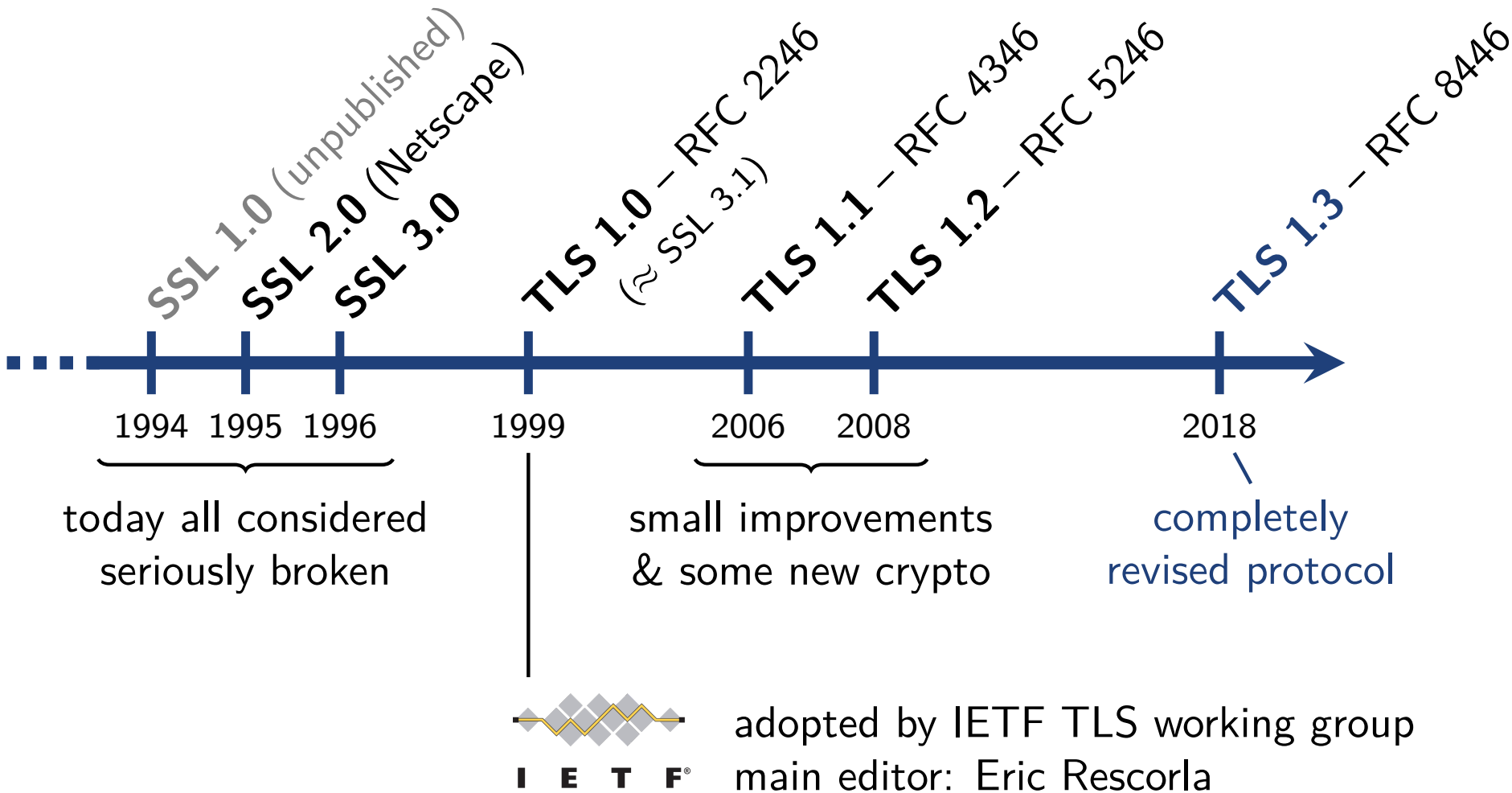
I E T F®

adopted by IETF TLS working group
main editor: Eric Rescorla

Transport Layer Security (TLS)

The SSL/TLS history . . .

ETH zürich



The TLS Protocol

High-level Goals

ETH zürich

"The primary goal of TLS is to provide a **secure channel between two peers**"

TLS 1.3 [RFC 8446]

The TLS Protocol

High-level Goals

ETH zürich

"The primary goal of TLS is to provide a **secure channel between two peers**"

TLS 1.3 [RFC 8446]

- **Authentication**
  - **server** side of the channel is **always authenticated**
  - **client** side is **optionally authenticated**
  - via **asymmetric crypto** (e.g., signatures) or a symmetric **pre-shared key**

The TLS Protocol

High-level Goals

ETH zürich

"The primary goal of TLS is to provide a **secure channel between two peers**"

TLS 1.3 [RFC 8446]

- **Authentication**
  - **server** side of the channel is **always authenticated**
  - **client** side is **optionally authenticated**
  - via **asymmetric crypto** (e.g., signatures) or a symmetric **pre-shared key**
- **Confidentiality**
  - **data** sent over the channel is **only visible to the endpoints**
  - TLS does **not hide the length** of the data it transmits (but allows padding)

The TLS Protocol

High-level Goals

ETH zürich

"The primary goal of TLS is to provide a **secure channel between two peers**"

TLS 1.3 [RFC 8446]

- **Authentication**
  - **server** side of the channel is **always authenticated**
  - **client** side is **optionally authenticated**
  - via **asymmetric crypto** (e.g., signatures) or a symmetric **pre-shared key**
- **Confidentiality**
  - **data** sent over the channel is **only visible to the endpoints**
  - TLS does **not hide the length** of the data it transmits (but allows padding)
- **Integrity**
  - **data** sent over the channel **cannot be modified** without detection

The TLS Protocol

High-level Goals

ETH zürich

"The primary goal of TLS is to provide a **secure channel between two peers**"

TLS 1.3 [RFC 8446]

- **Authentication**
  - **server** side of the channel is **always authenticated**
  - **client** side is **optionally authenticated**
  - via **asymmetric crypto** (e.g., signatures) or a symmetric **pre-shared key**
- **Confidentiality**
  - **data** sent over the channel is **only visible to the endpoints**
  - TLS does **not hide the length** of the data it transmits (but allows padding)
- **Integrity**
  - **data** sent over the channel **cannot be modified** without detection
- security in the face of **attacker who has complete control of the network**
- only requirement from underlying transport: reliable, in-order data stream

Transport Layer Security (TLS)

Main Components (overly simplified)

ETH zürich

Client

Server

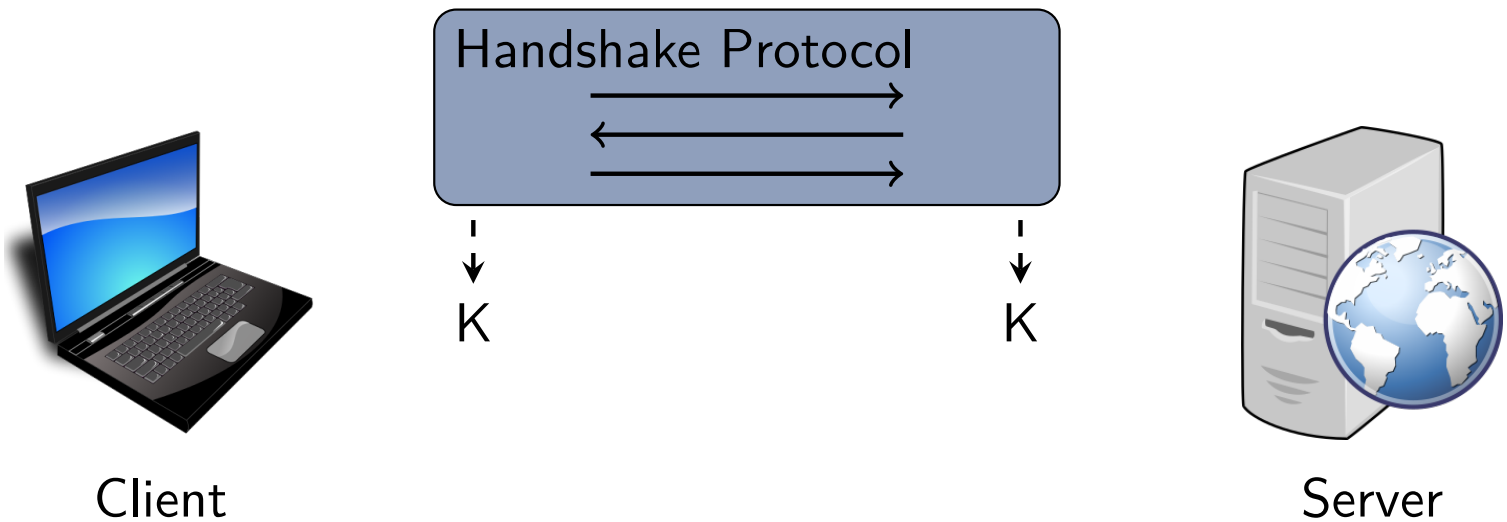
Transport Layer Security (TLS)

Main Components (overly simplified)

ETH zürich

Handshake Protocol:

- negotiate security parameters ("cipher suite")
- authenticate peers
- establish key material for data protection



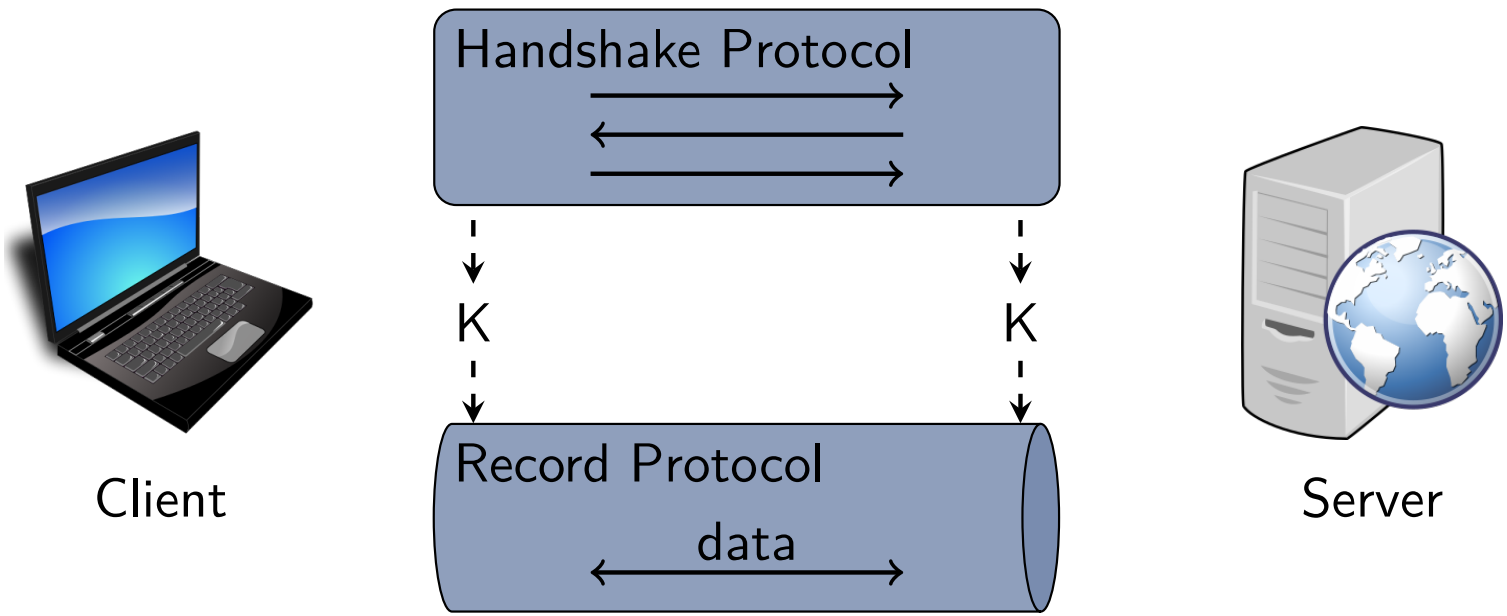
Transport Layer Security (TLS)

Main Components (overly simplified)

ETH zürich

Handshake Protocol:

- negotiate security parameters ("cipher suite")
- authenticate peers
- establish key material for data protection



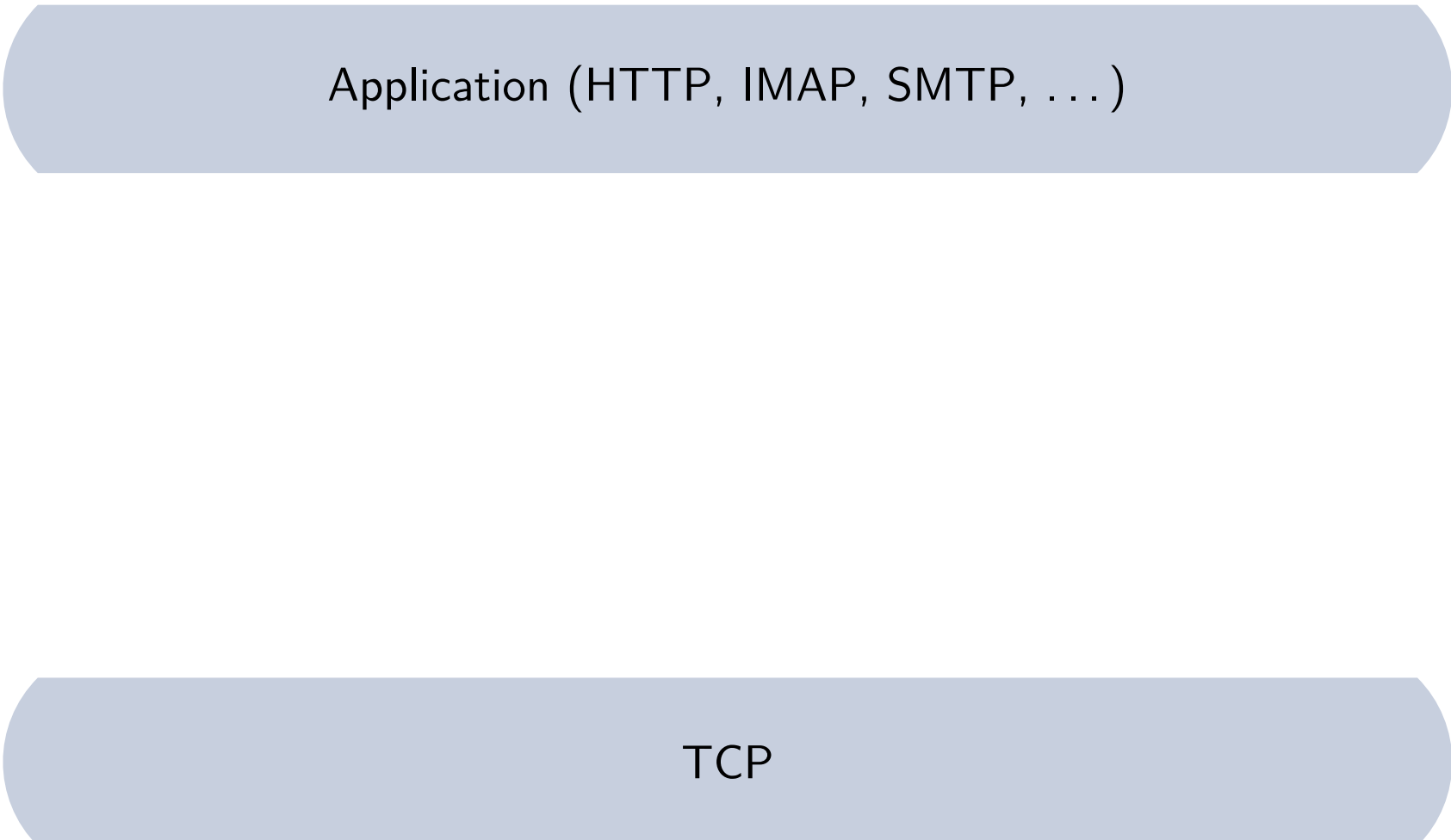
Record Protocol:

- protect data using key material from handshake
- ensuring confidentiality and integrity

The TLS Protocol

Architecture within Network Stack

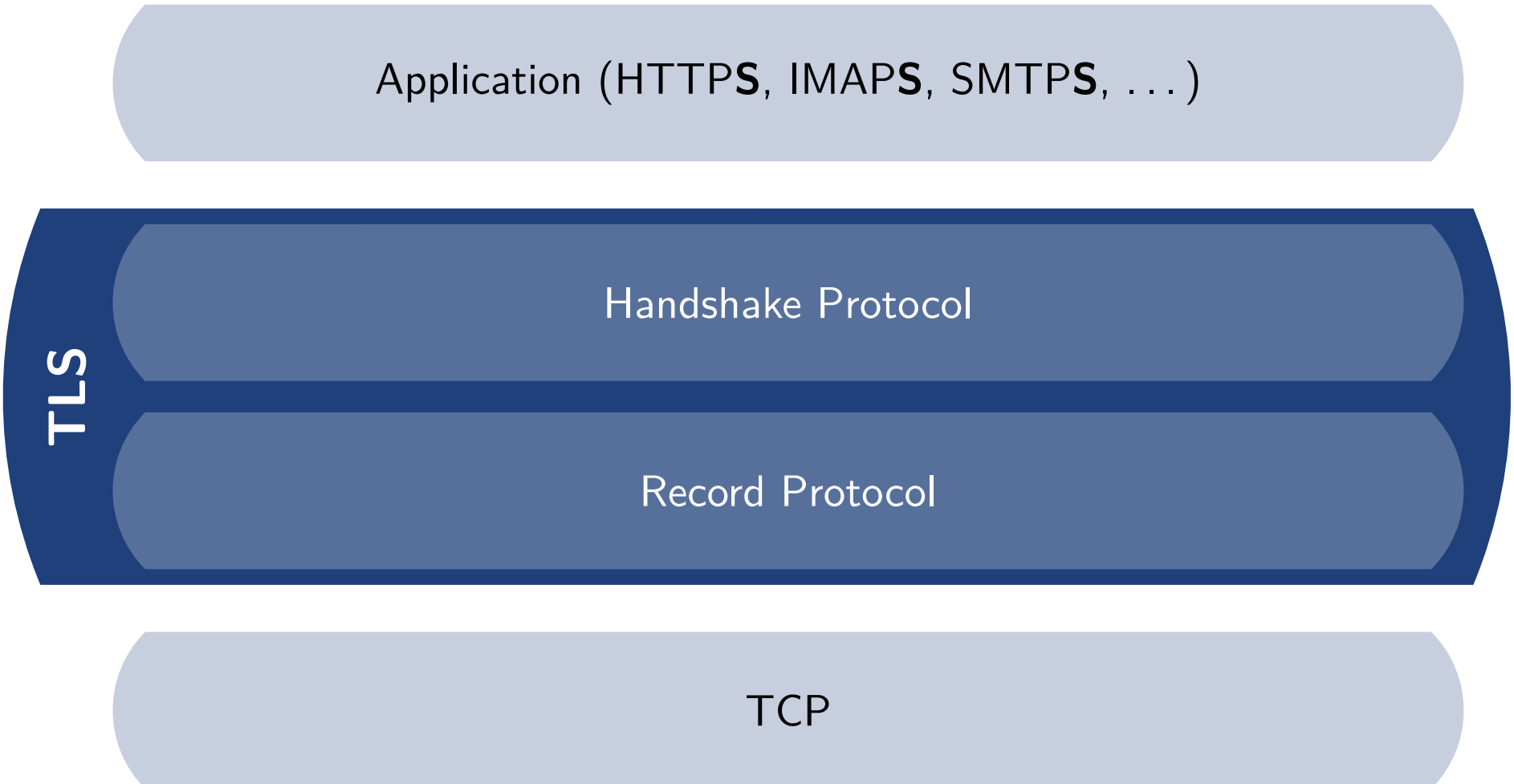
ETH zürich



The TLS Protocol

Architecture within Network Stack

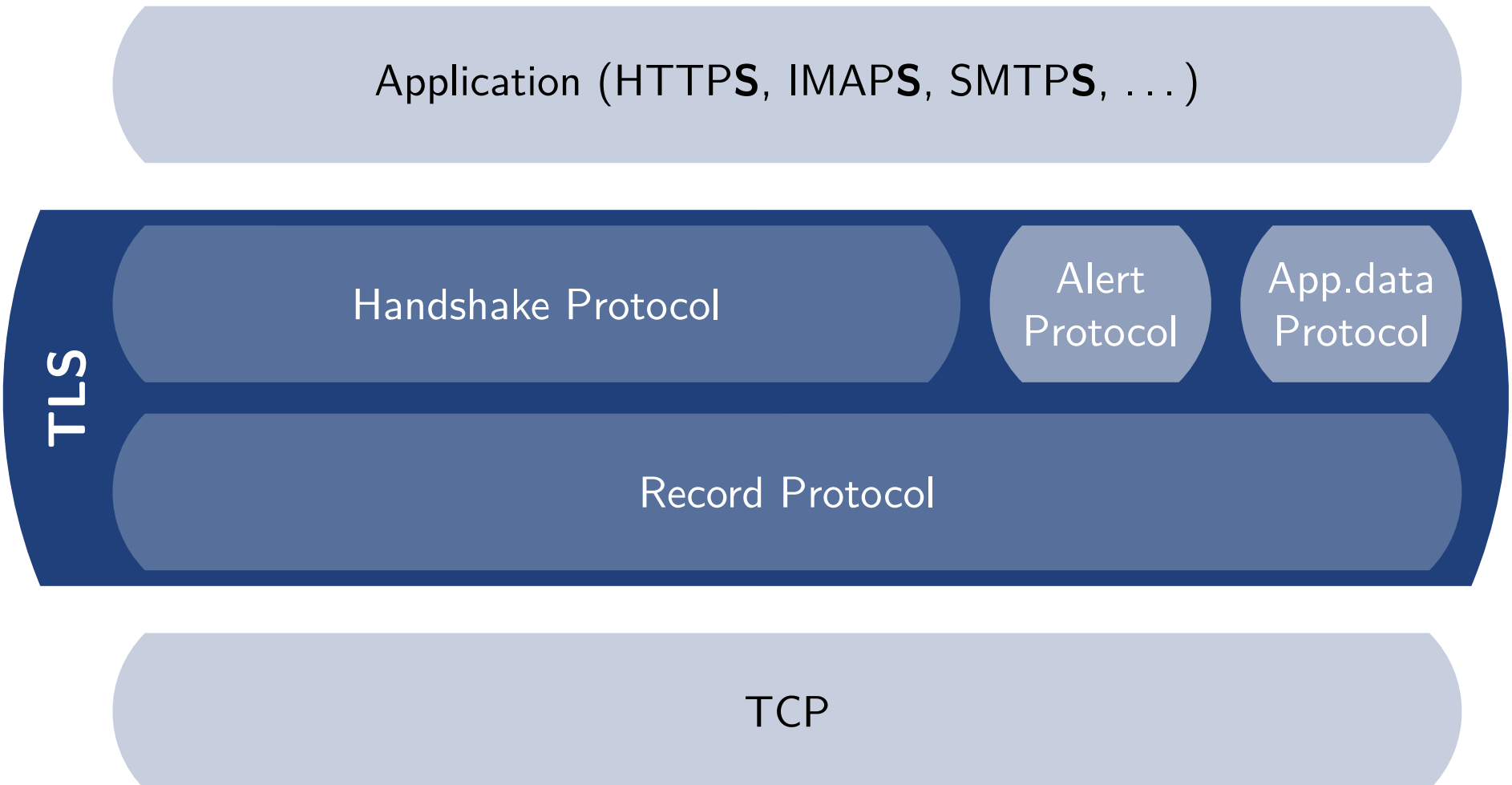
ETH zürich



The TLS Protocol

Architecture within Network Stack

ETH zürich



The TLS Protocol

Cryptographic Components

ETH zürich

- TLS is a **"self-negotiating"** protocol
- handshake first of all agrees on TLS version and cipher suite to use

The TLS Protocol

Cryptographic Components

ETH zürich

- TLS is a **"self-negotiating"** protocol
- handshake first of all agrees on TLS version and cipher suite to use

- **Cipher suites:** client proposes list, server picks
- fixes crypto algorithms to be used for that session
- format (up to TLS 1.2): `TLS_KEX_AUT_WITH_CIP_MAC`

The TLS Protocol

Cryptographic Components

ETH zürich

- TLS is a **"self-negotiating"** protocol
- handshake first of all agrees on TLS version and cipher suite to use

- **Cipher suites:** client proposes list, server picks
- fixes crypto algorithms to be used for that session
- format (up to TLS 1.2): `TLS_KEX_AUT_WITH_CIP_MAC`

Key Exchange

`RSA` `DHE` `ECDHE` `PSK` ...

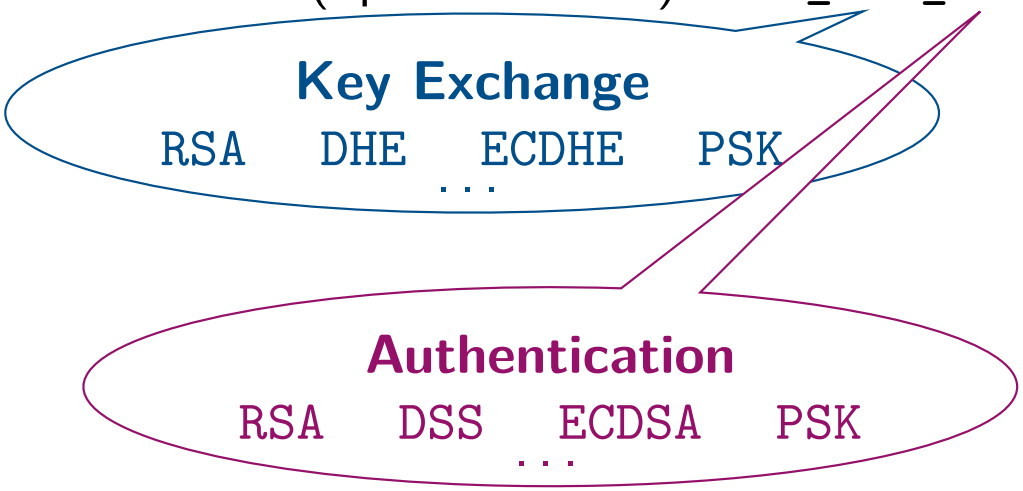
The TLS Protocol

Cryptographic Components

ETH zürich

- TLS is a **"self-negotiating"** protocol
- handshake first of all agrees on TLS version and cipher suite to use

- **Cipher suites:** client proposes list, server picks
- fixes crypto algorithms to be used for that session
- format (up to TLS 1.2): `TLS_KEX_AUT_WITH_CIP_MAC`



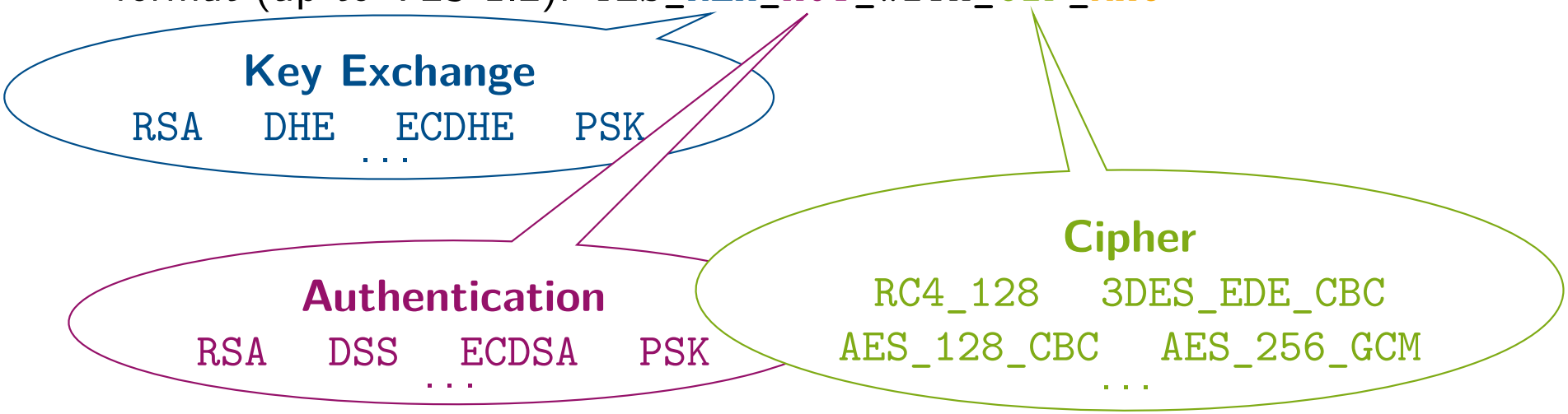
The TLS Protocol

Cryptographic Components

ETH zürich

- TLS is a **"self-negotiating"** protocol
- handshake first of all agrees on TLS version and cipher suite to use

- **Cipher suites:** client proposes list, server picks
- fixes crypto algorithms to be used for that session
- format (up to TLS 1.2): `TLS_KEX_AUT_WITH_CIP_MAC`



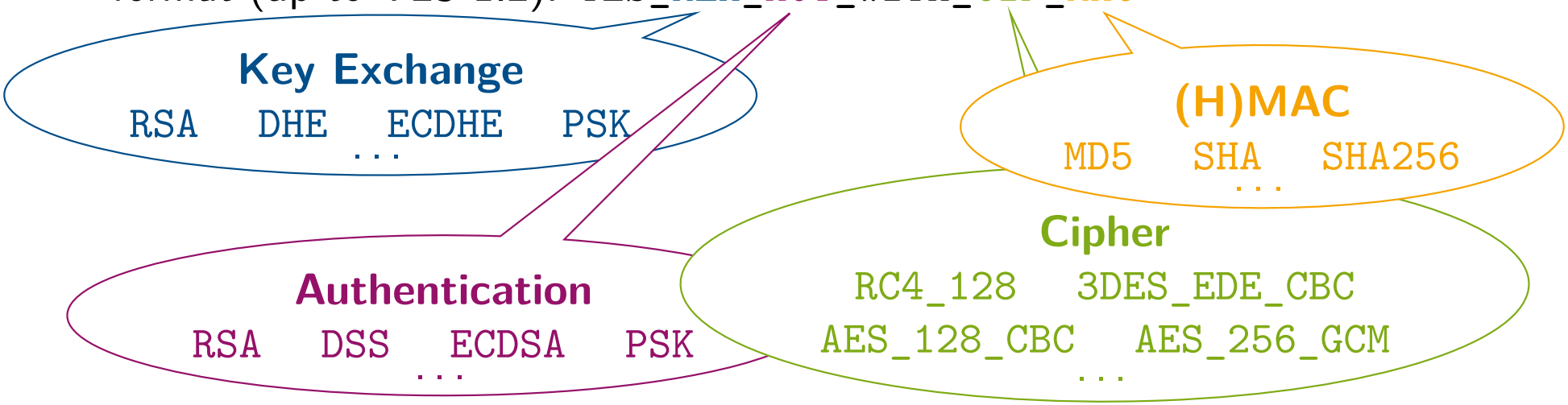
The TLS Protocol

Cryptographic Components

ETH zürich

- TLS is a **"self-negotiating"** protocol
- handshake first of all agrees on TLS version and cipher suite to use

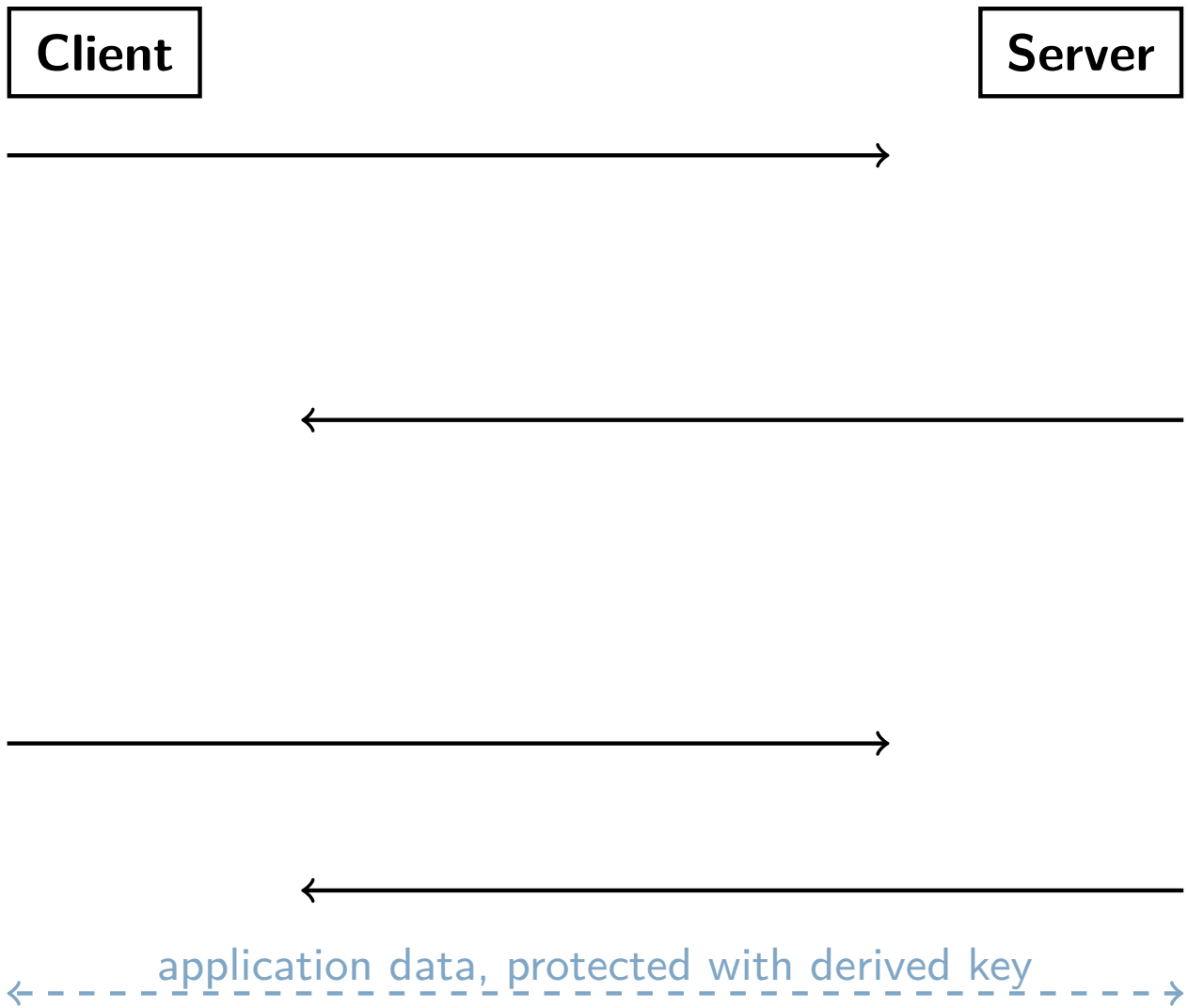
- **Cipher suites:** client proposes list, server picks
- fixes crypto algorithms to be used for that session
- format (up to TLS 1.2): `TLS_KEX_AUT_WITH_CIP_MAC`



The TLS Protocol

TLS 1.2 Handshake Protocol Structure

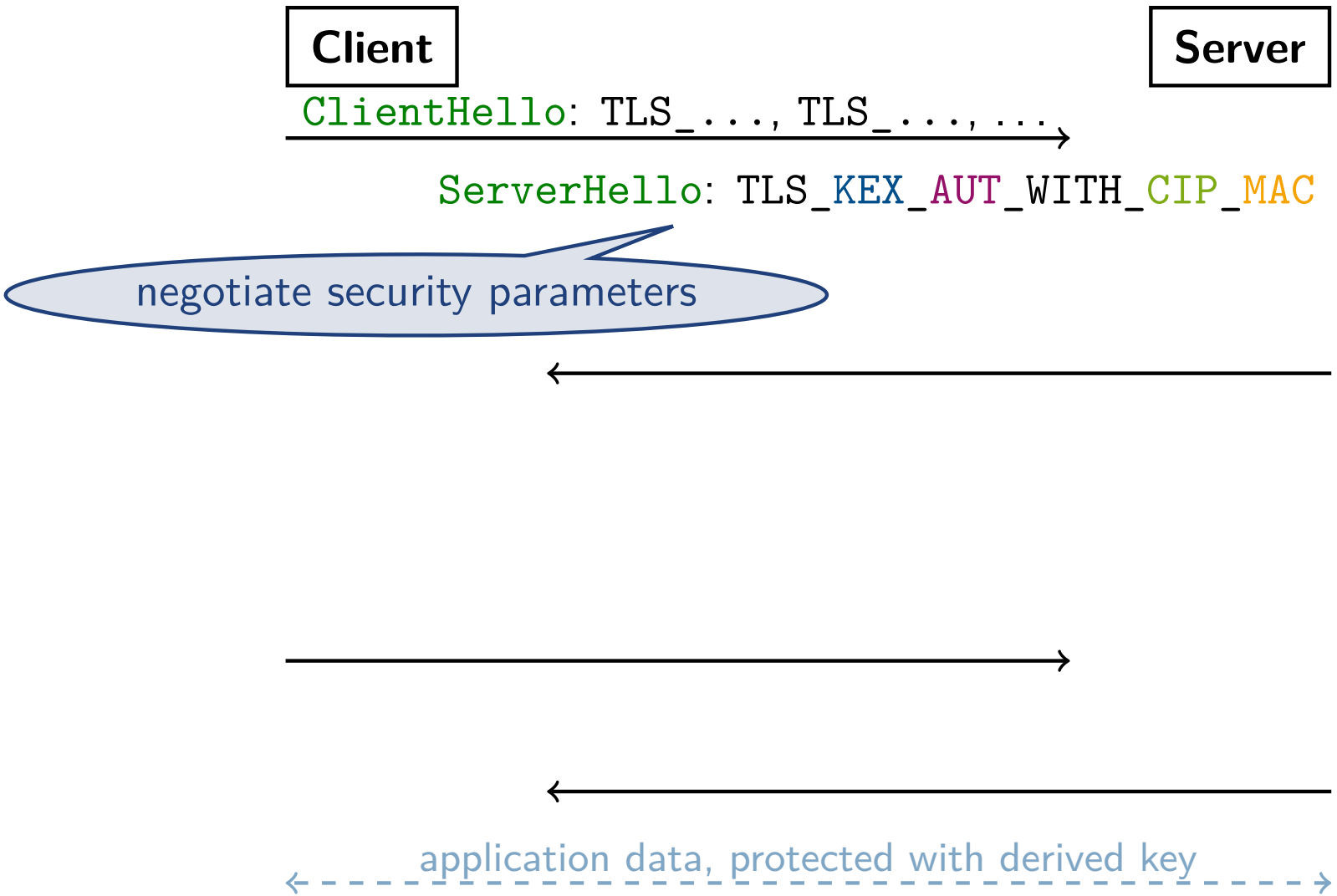
ETH zürich



The TLS Protocol

TLS 1.2 Handshake Protocol Structure

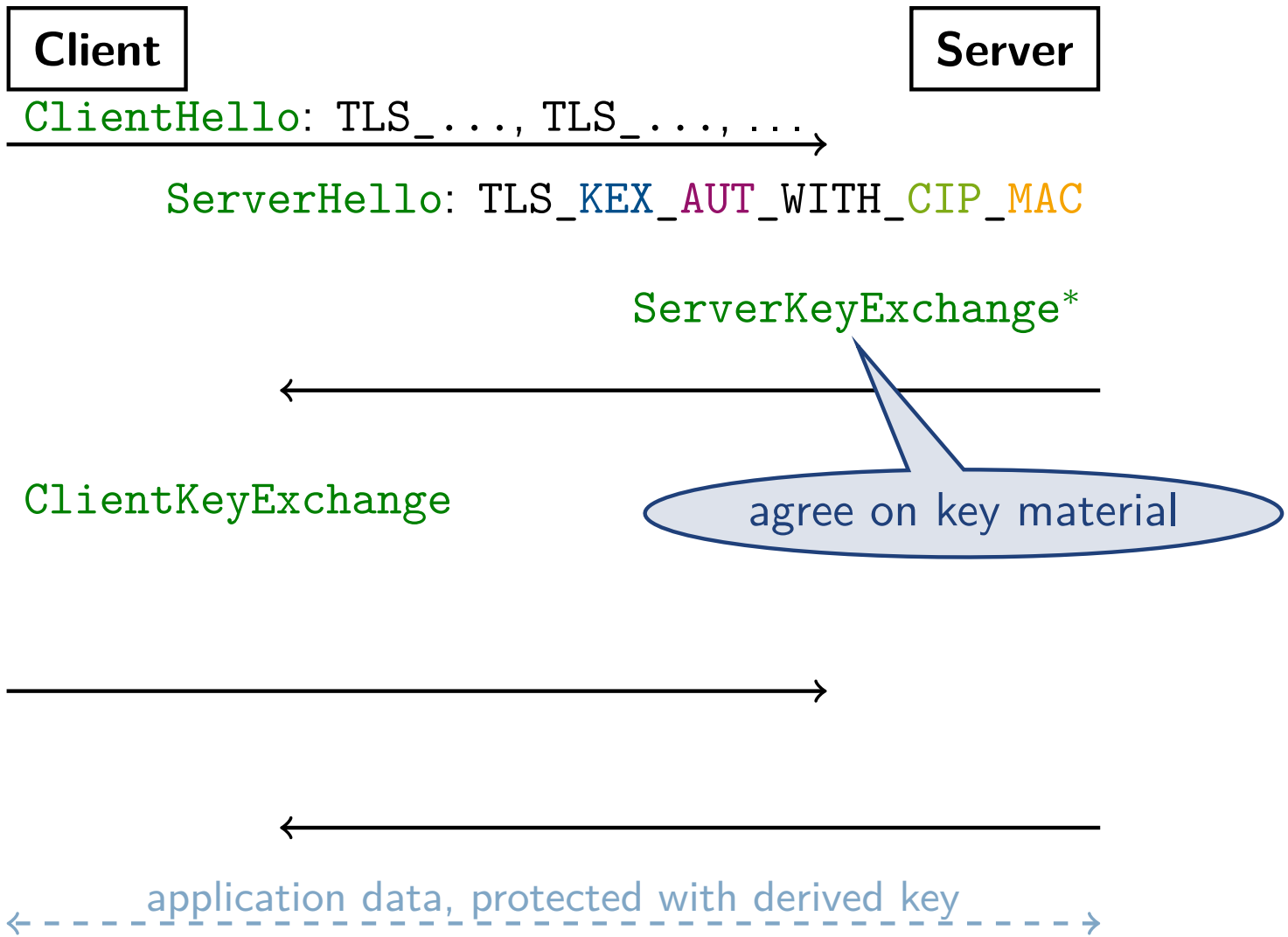
ETH zürich



The TLS Protocol

TLS 1.2 Handshake Protocol Structure

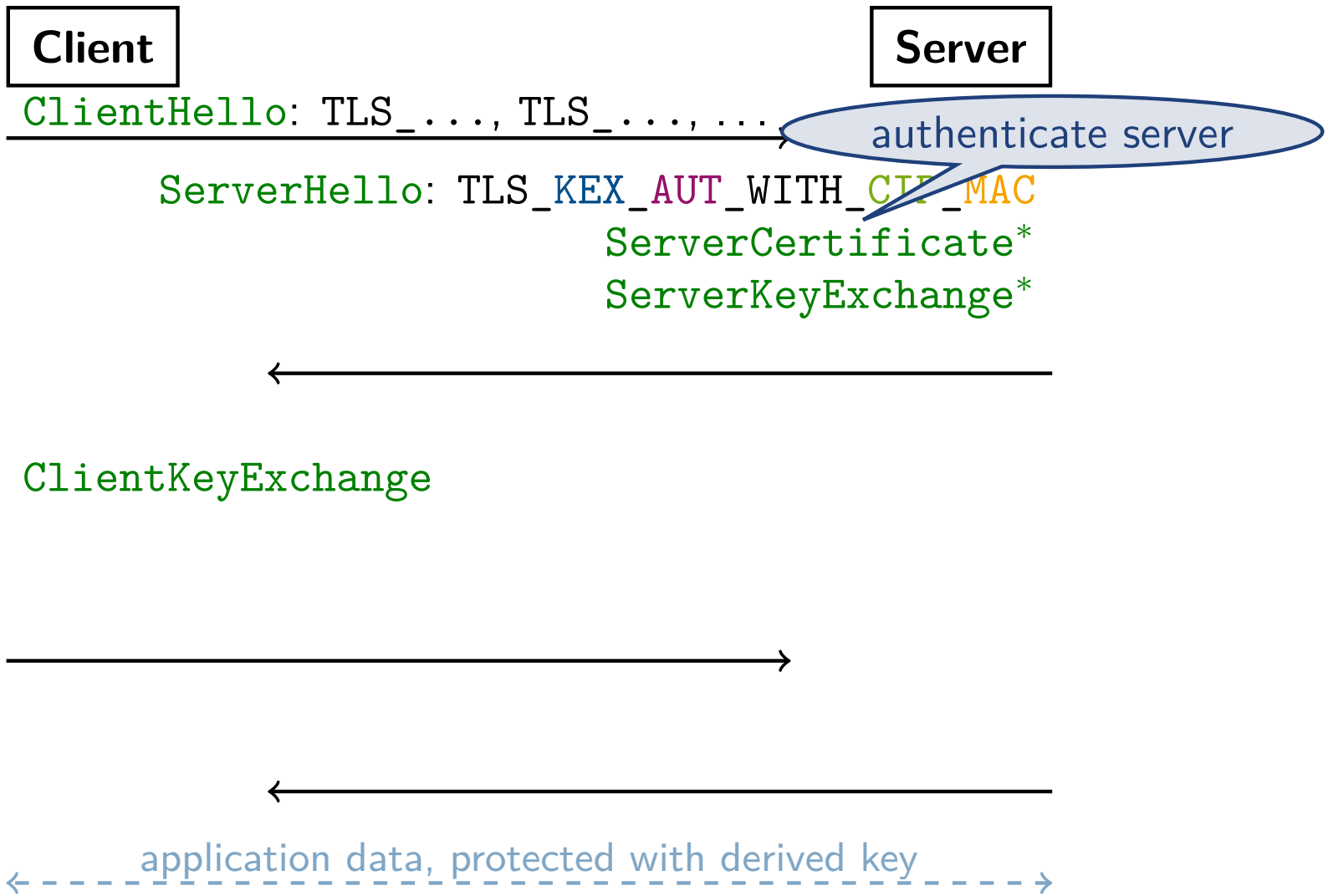
ETH zürich



The TLS Protocol

TLS 1.2 Handshake Protocol Structure

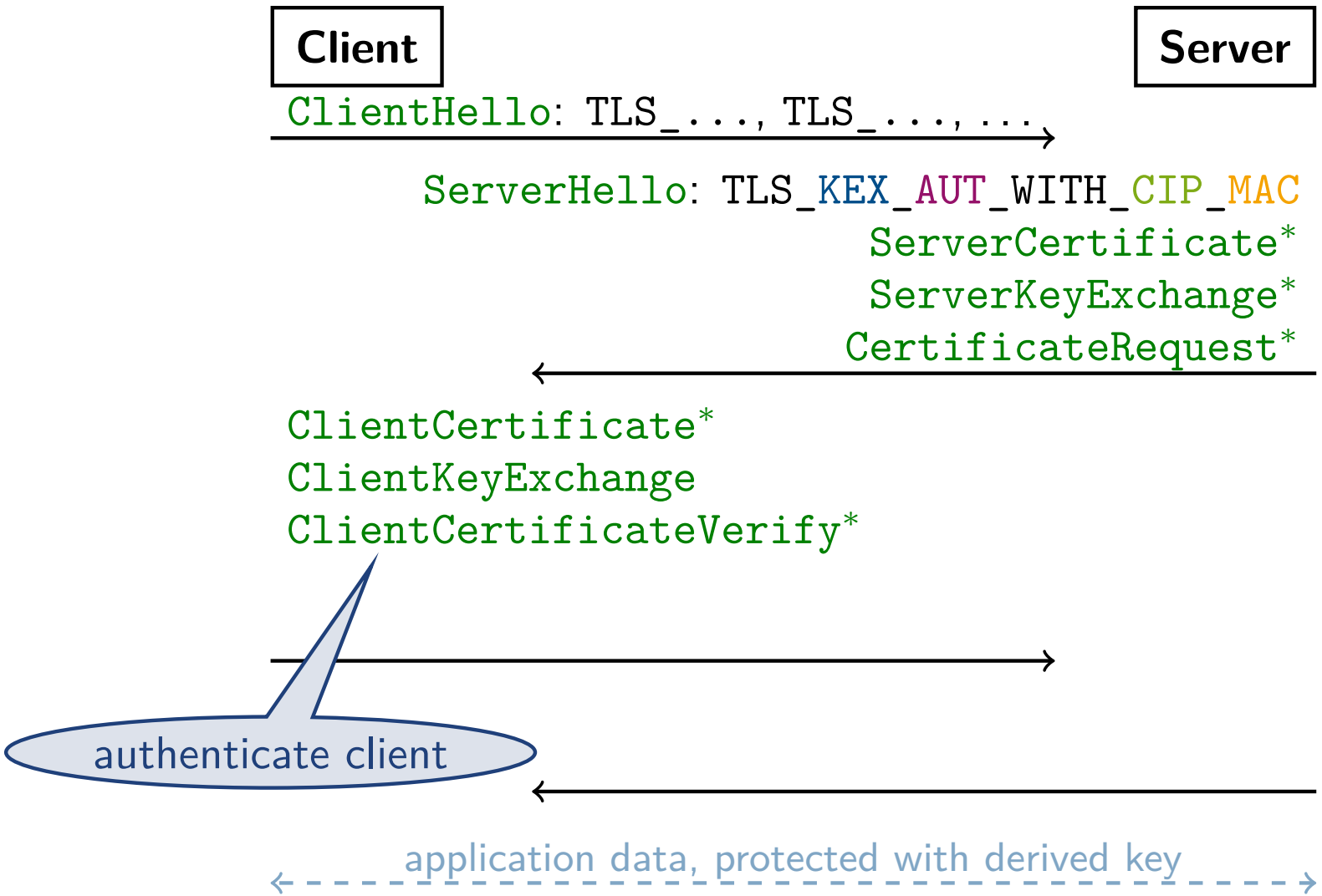
ETH zürich



The TLS Protocol

TLS 1.2 Handshake Protocol Structure

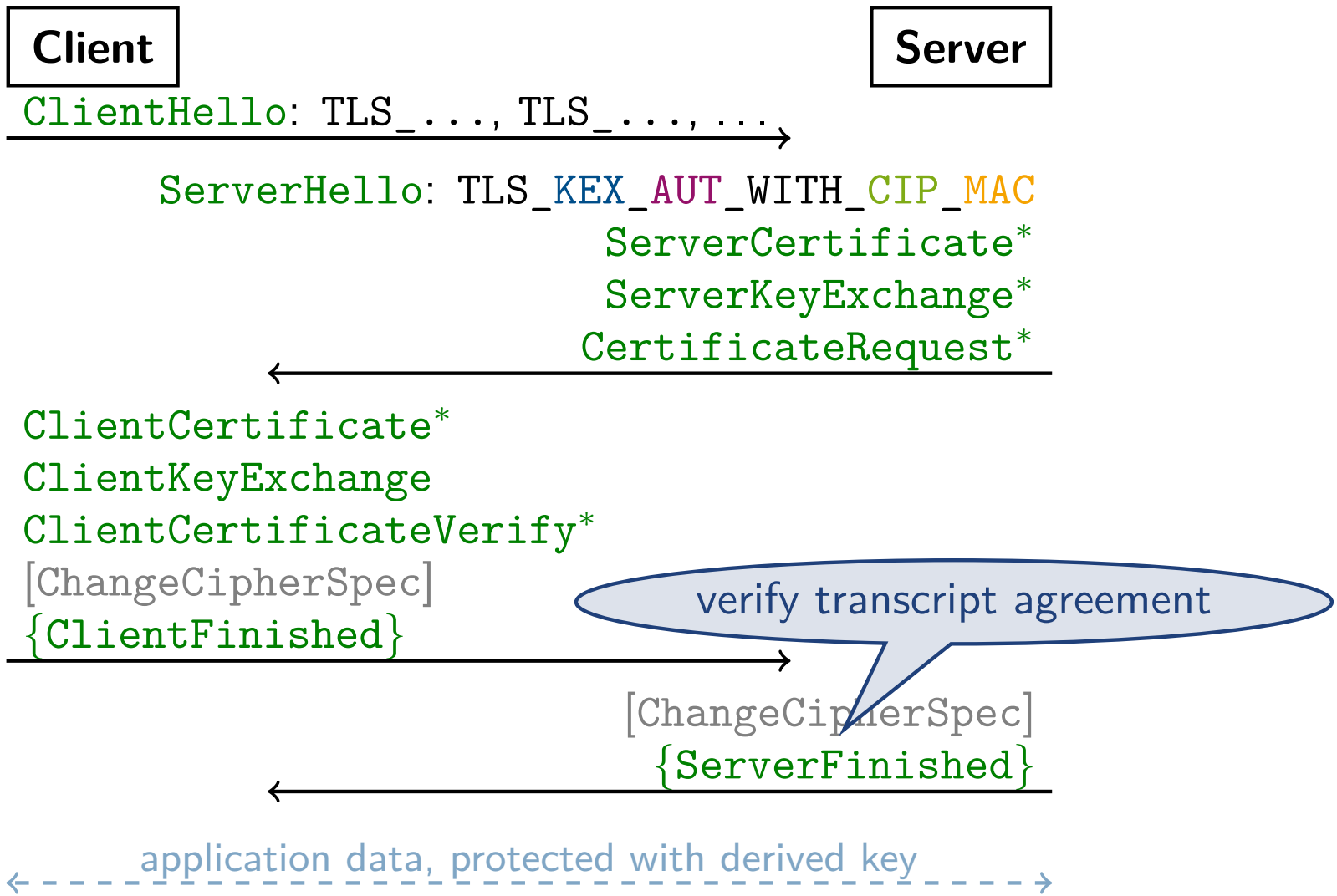
ETH zürich



The TLS Protocol

TLS 1.2 Handshake Protocol Structure

ETH zürich



The TLS Protocol

TLS 1.2 Example: TLS_RSA_WITH_AES_128_CBC_SHA — Handshake

ETH zürich

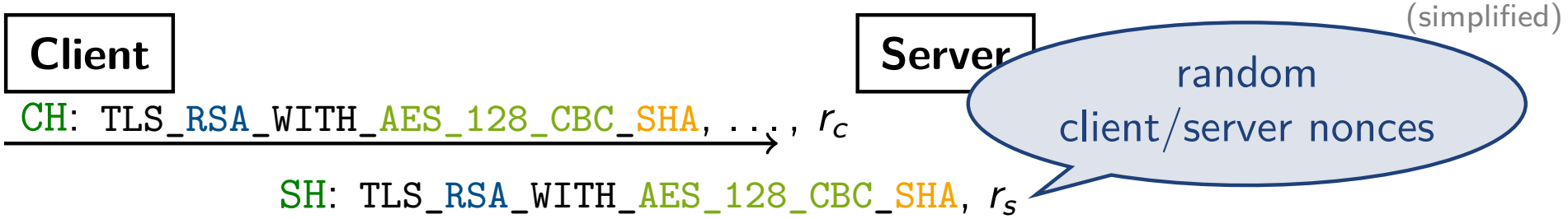
(simplified)

The TLS Protocol

TLS 1.2 Example: TLS_RSA_WITH_AES_128_CBC_SHA — Handshake

ETH zürich

(simplified)

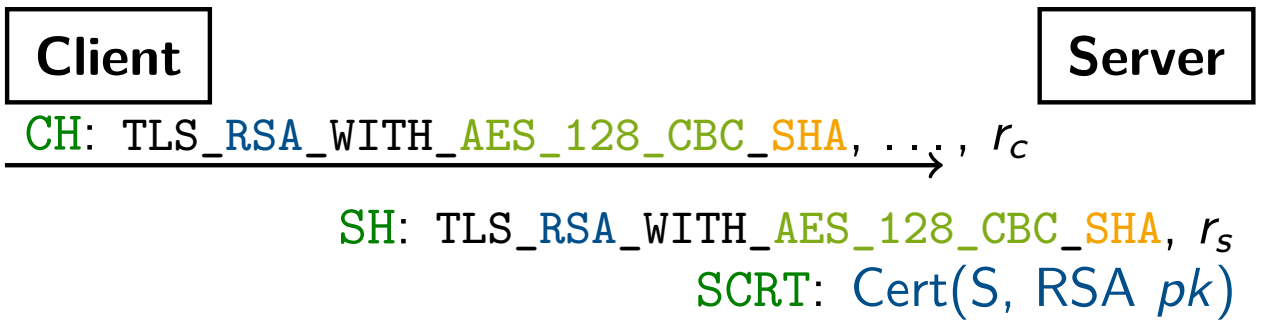


The TLS Protocol

TLS 1.2 Example: TLS_RSA_WITH_AES_128_CBC_SHA — Handshake

ETH zürich

(simplified)

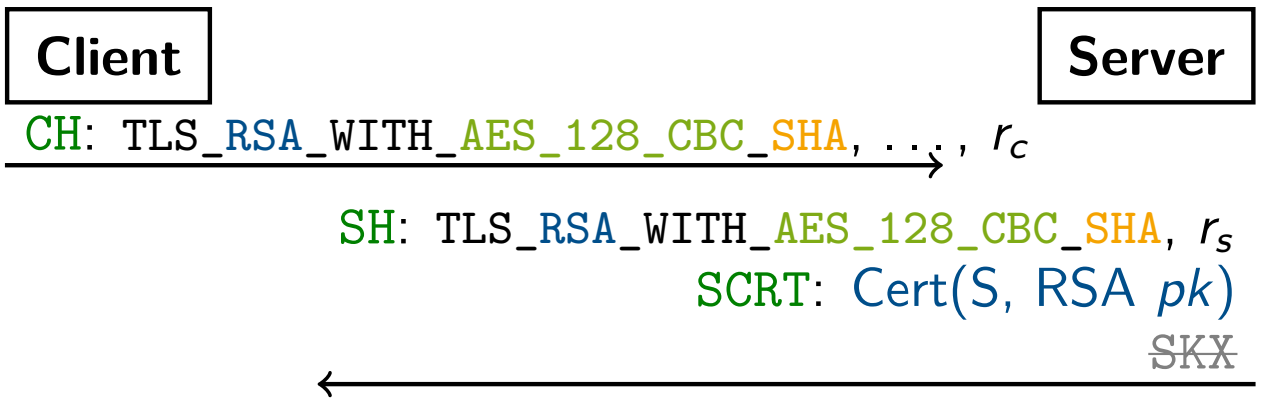


The TLS Protocol

TLS 1.2 Example: TLS_RSA_WITH_AES_128_CBC_SHA — Handshake

ETH zürich

(simplified)

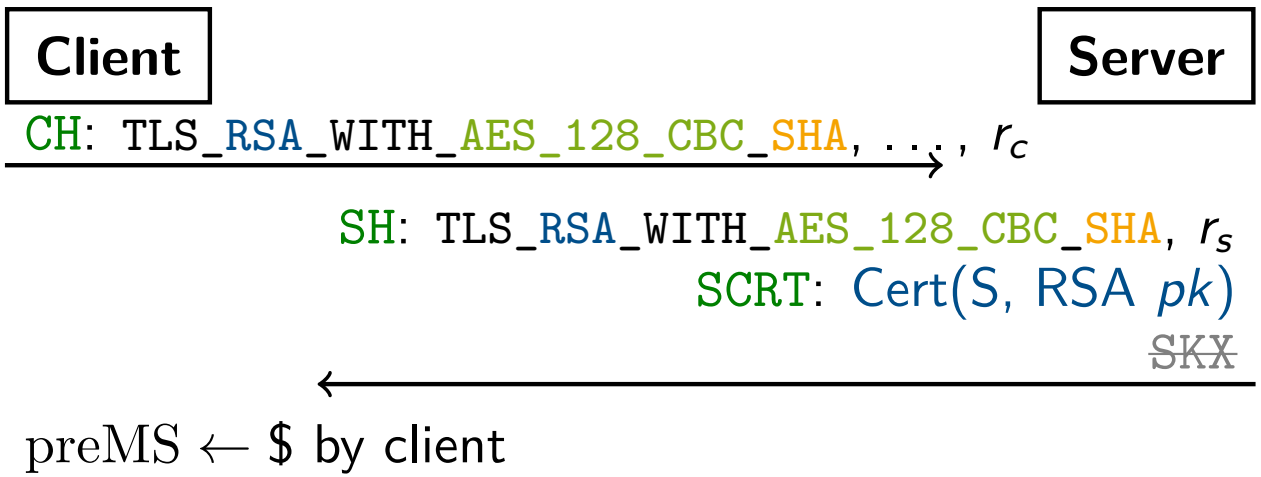


The TLS Protocol

TLS 1.2 Example: TLS_RSA_WITH_AES_128_CBC_SHA — Handshake

ETH zürich

(simplified)

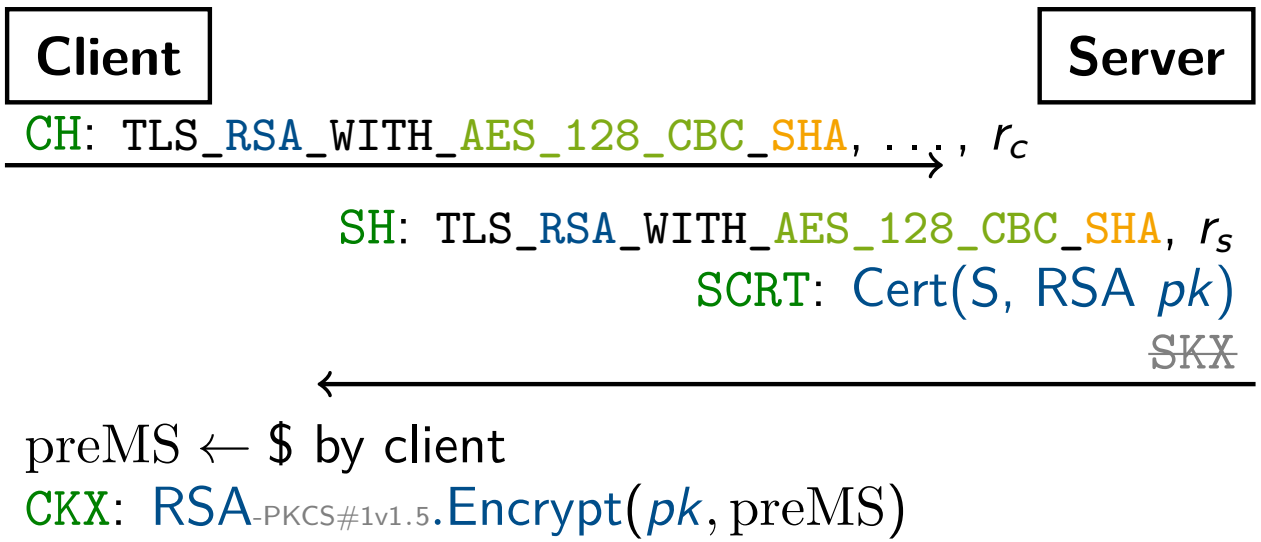


The TLS Protocol

TLS 1.2 Example: TLS_RSA_WITH_AES_128_CBC_SHA — Handshake

ETH zürich

(simplified)

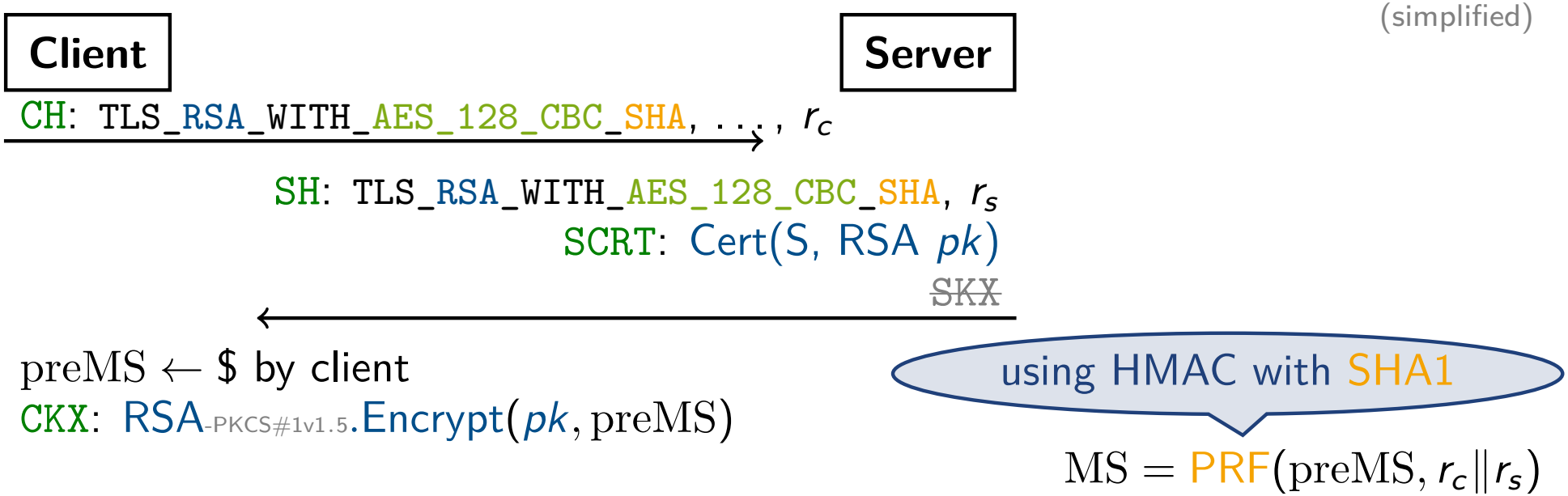


The TLS Protocol

TLS 1.2 Example: TLS_RSA_WITH_AES_128_CBC_SHA — Handshake

ETH zürich

(simplified)

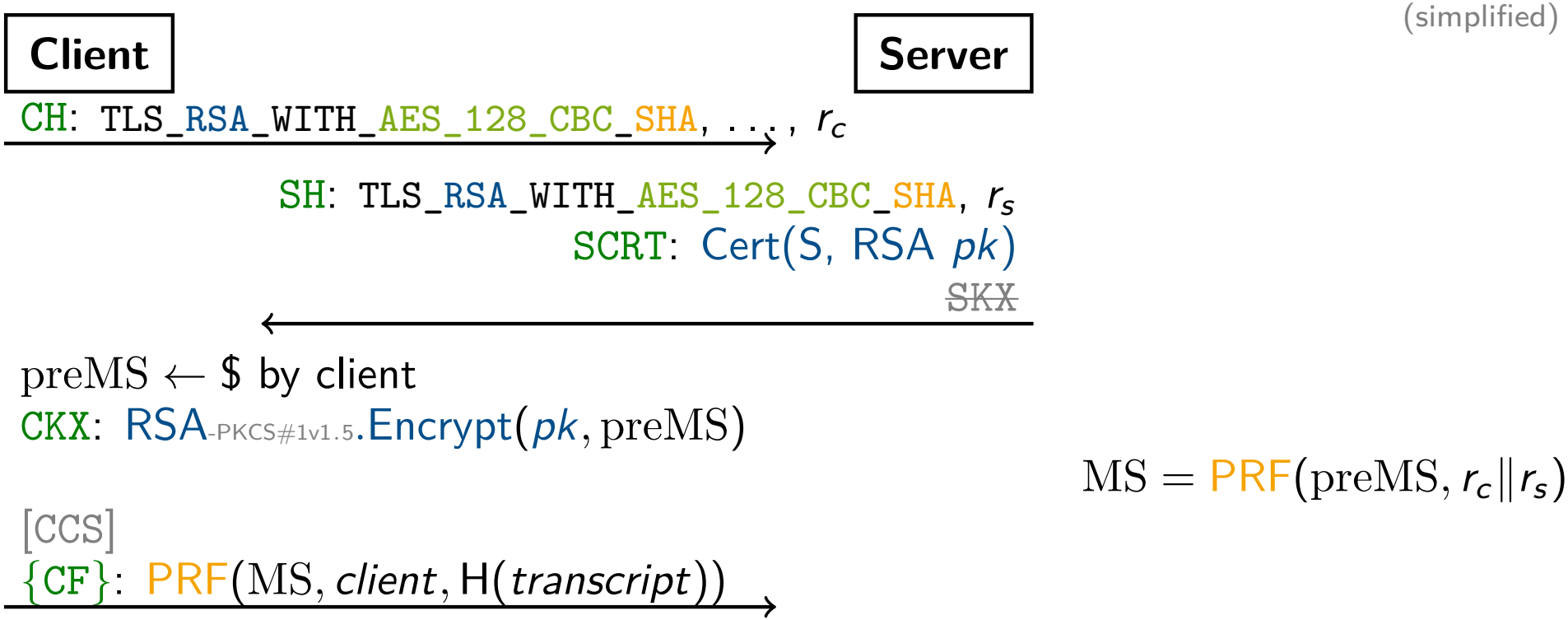


The TLS Protocol

TLS 1.2 Example: TLS_RSA_WITH_AES_128_CBC_SHA — Handshake

ETH zürich

(simplified)

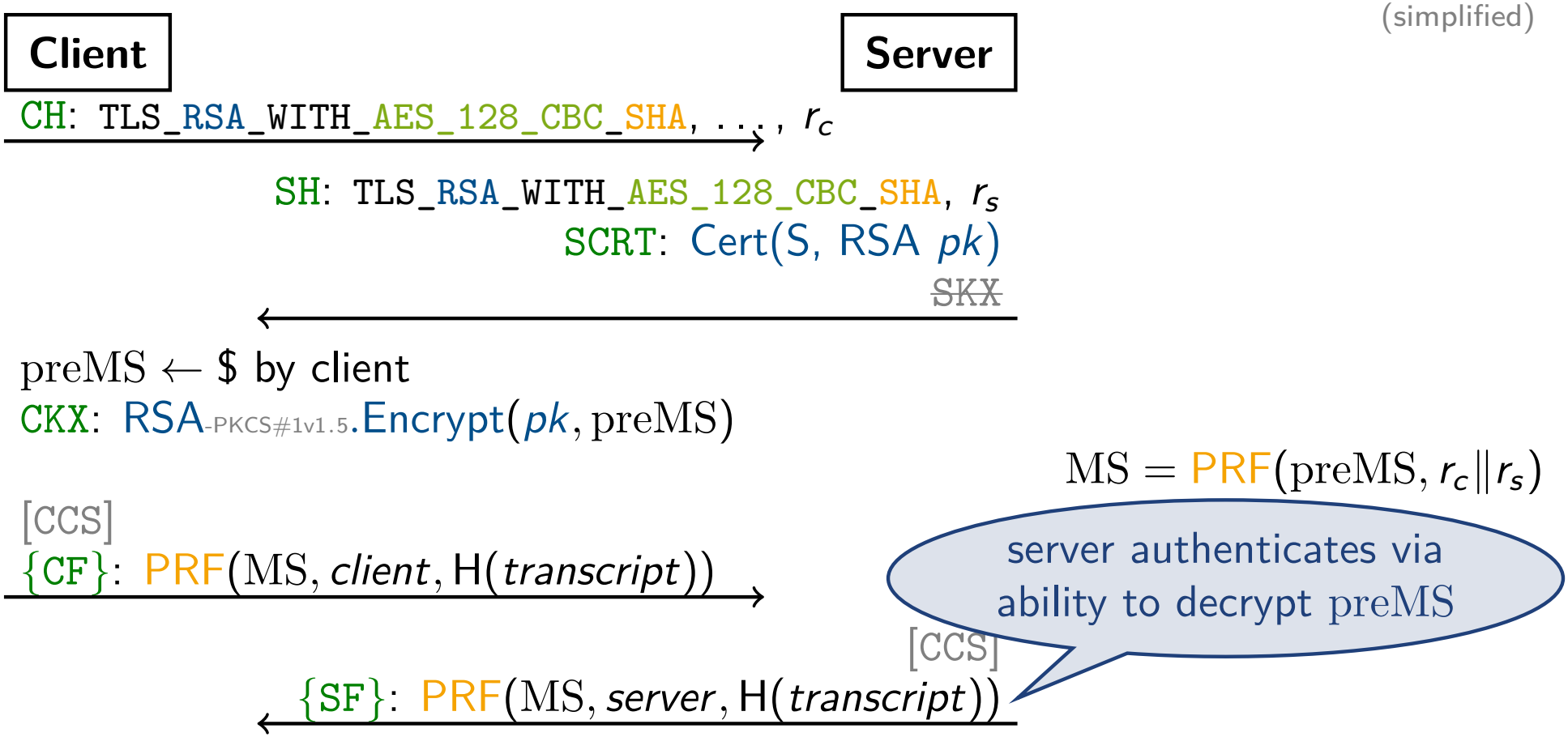


The TLS Protocol

TLS 1.2 Example: TLS_RSA_WITH_AES_128_CBC_SHA — Handshake

ETH zürich

(simplified)

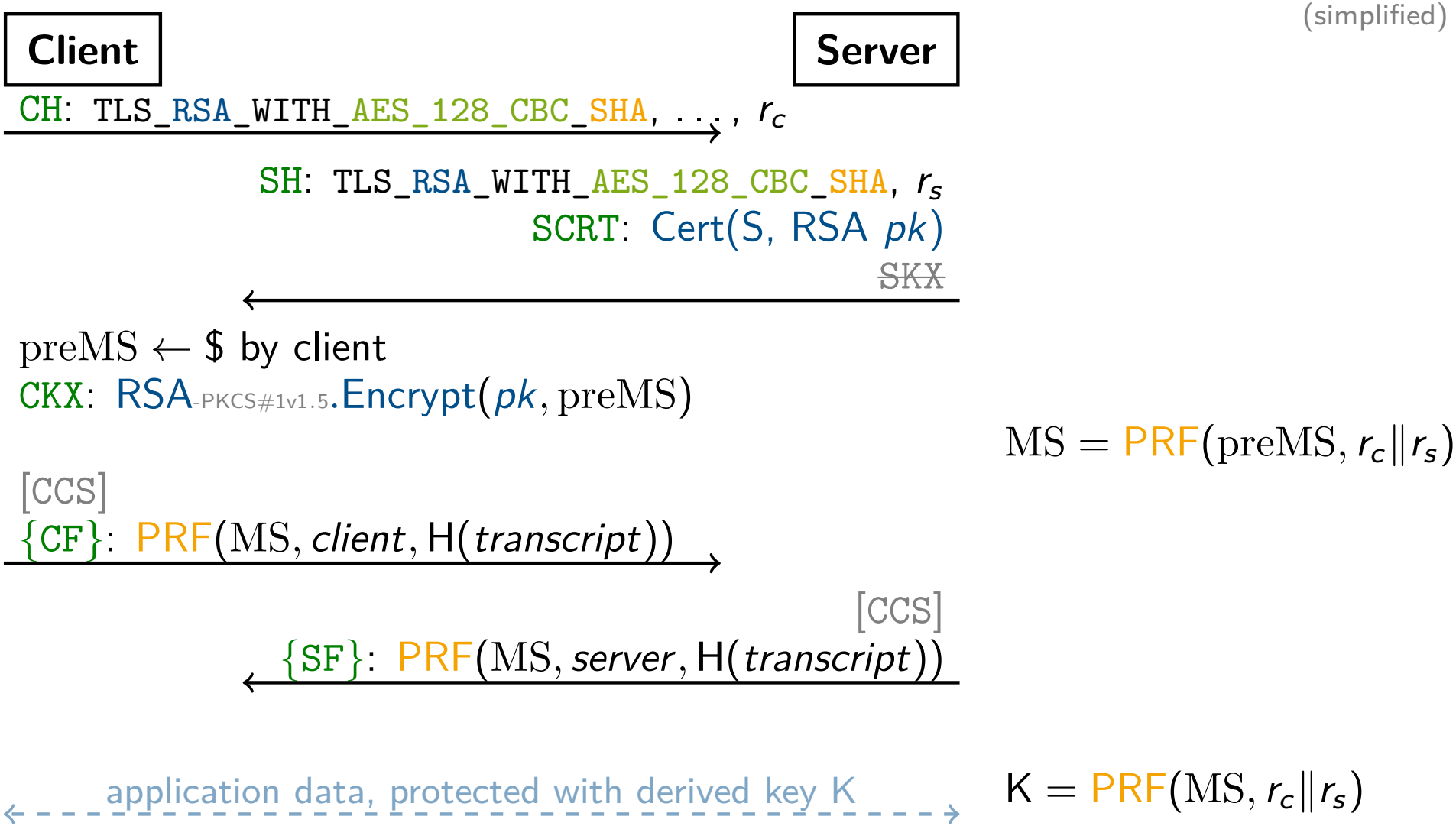


The TLS Protocol

TLS 1.2 Example: TLS_RSA_WITH_AES_128_CBC_SHA — Handshake

ETH zürich

(simplified)



The TLS Protocol

TLS 1.2 Example: TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 — Handshake

ETH zürich

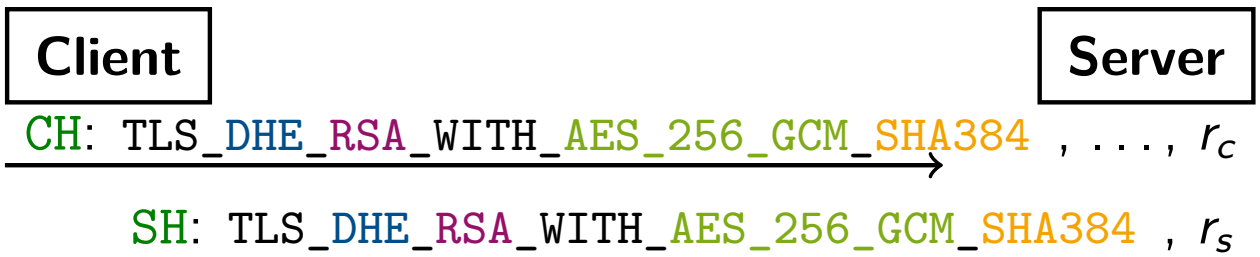
(simplified)

The TLS Protocol

TLS 1.2 Example: TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 — Handshake

ETH zürich

(simplified)

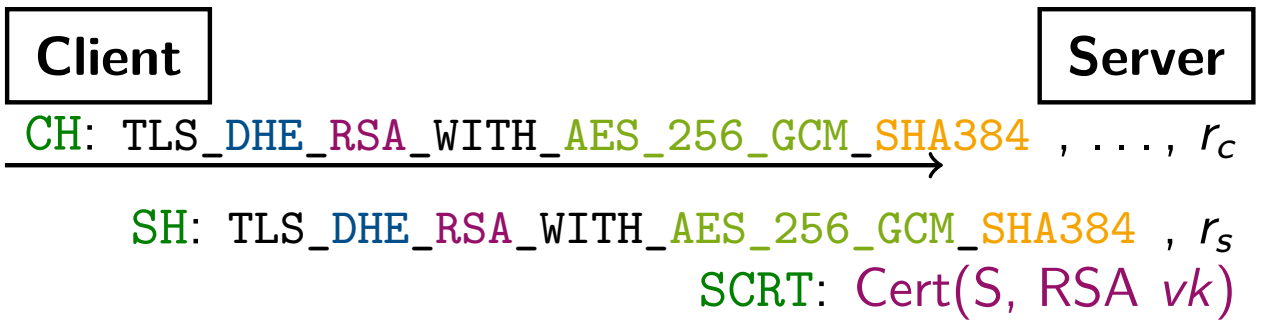


The TLS Protocol

TLS 1.2 Example: TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 — Handshake

ETH zürich

(simplified)

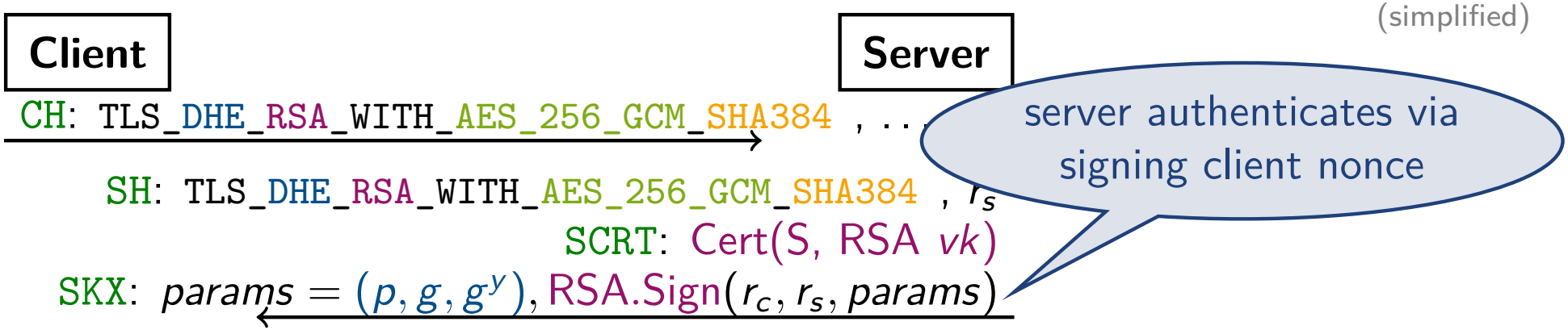


The TLS Protocol

TLS 1.2 Example: TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 — Handshake

ETH zürich

(simplified)

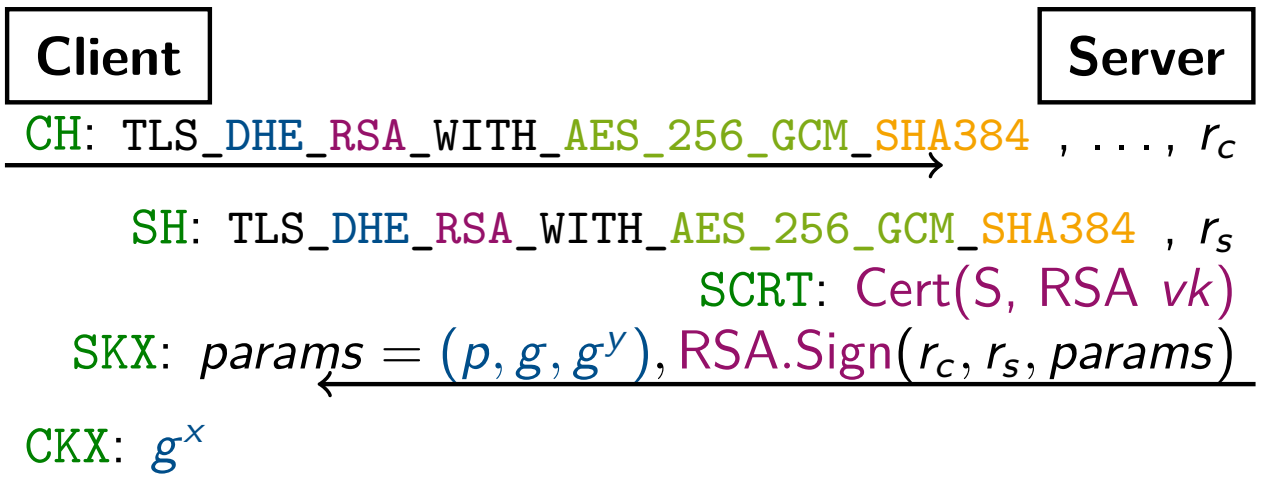


The TLS Protocol

TLS 1.2 Example: TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 — Handshake

ETH zürich

(simplified)



The TLS Protocol

TLS 1.2 Example: TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 — Handshake

ETH zürich

(simplified)

Client		Server
CH: TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 , ..., r_c	$\longrightarrow$	
		SH: TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 , r_s
		SCRT: Cert(S, RSA vk)
	$\longleftarrow$	SKX: $params = (p, g, g^y)$, RSA.Sign$(r_c, r_s, params)$
CKX: g^x		

$\text{preMS} = g^{xy}$

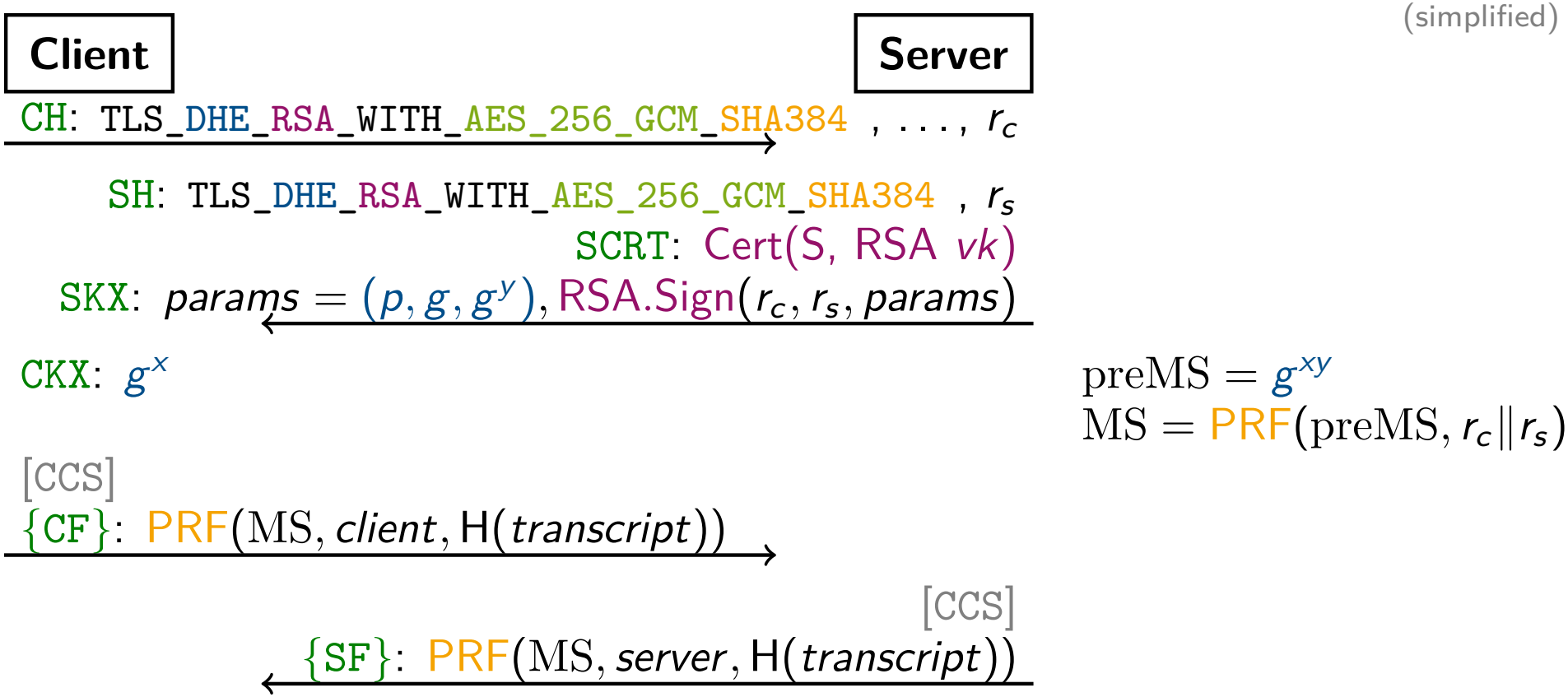
$\text{MS} = \text{PRF}(\text{preMS}, r_c \| r_s)$

The TLS Protocol

TLS 1.2 Example: TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 — Handshake

ETH zürich

(simplified)



The TLS Protocol

TLS 1.2 Example: TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 — Handshake

ETH zürich

(simplified)

Client		Server
CH: TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 , ..., r_c	→	
	←	SH: TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 , r_s
	←	SCRT: Cert(S, RSA vk)
	←	SKX: $params = (p, g, g^y)$, RSA.Sign($r_c, r_s, params$)
CKX: g^x	→	
		$\text{preMS} = g^{xy}$
		$\text{MS} = \text{PRF}(\text{preMS}, r_c \| r_s)$
[CCS]		
{CF}: PRF(MS, *client*, H(*transcript*))	→	
	←	[CCS]
	←	{SF}: PRF(MS, *server*, H(*transcript*))
		$\text{K} = \text{PRF}(\text{MS}, r_c \| r_s)$

The TLS Protocol

TLS 1.2 Example: TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 — Handshake

ETH zürich

(simplified)

Client		Server
CH: TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 , ..., r_c	→	
	←	SH: TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 , r_s
	←	SCRT: Cert(S, RSA vk)
	←	SKX: $params = (p, g, g^y)$, RSA.Sign($r_c, r_s, params$)
CKX: g^x	→	
		$\text{preMS} = g^{xy}$
		$\text{MS} = \text{PRF}(\text{preMS}, r_c \| r_s)$
[CCS]		
{CF}: PRF(MS, *client*, H(*transcript*))	→	
	←	[CCS]
	←	{SF}: PRF(MS, *server*, H(*transcript*))
		$\text{K} = \text{PRF}(\text{MS}, r_c \| r_s)$
application data, protected with derived key K	↔	

The TLS Protocol

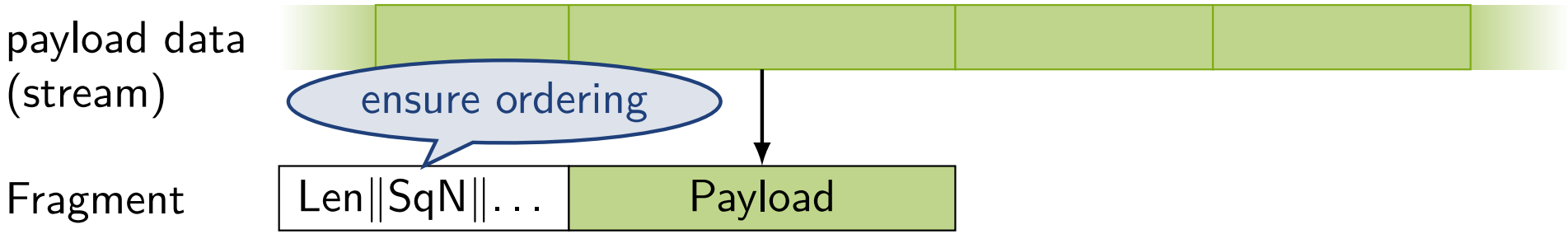
TLS 1.2 Record Protocol Structure

ETH zürich

payload data (stream)

The TLS Protocol

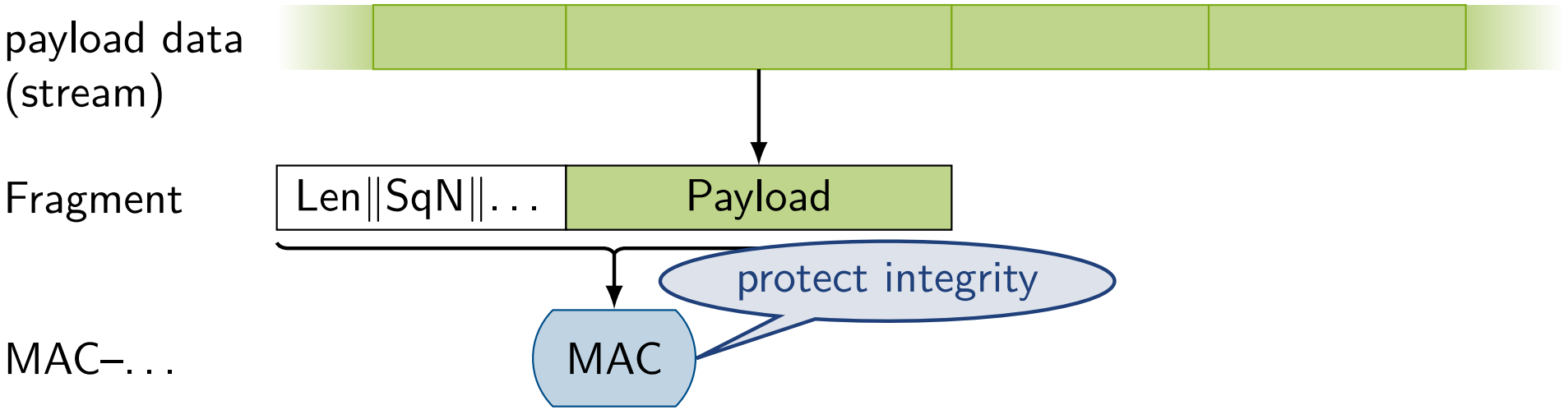
TLS 1.2 Record Protocol Structure



The TLS Protocol

TLS 1.2 Record Protocol Structure

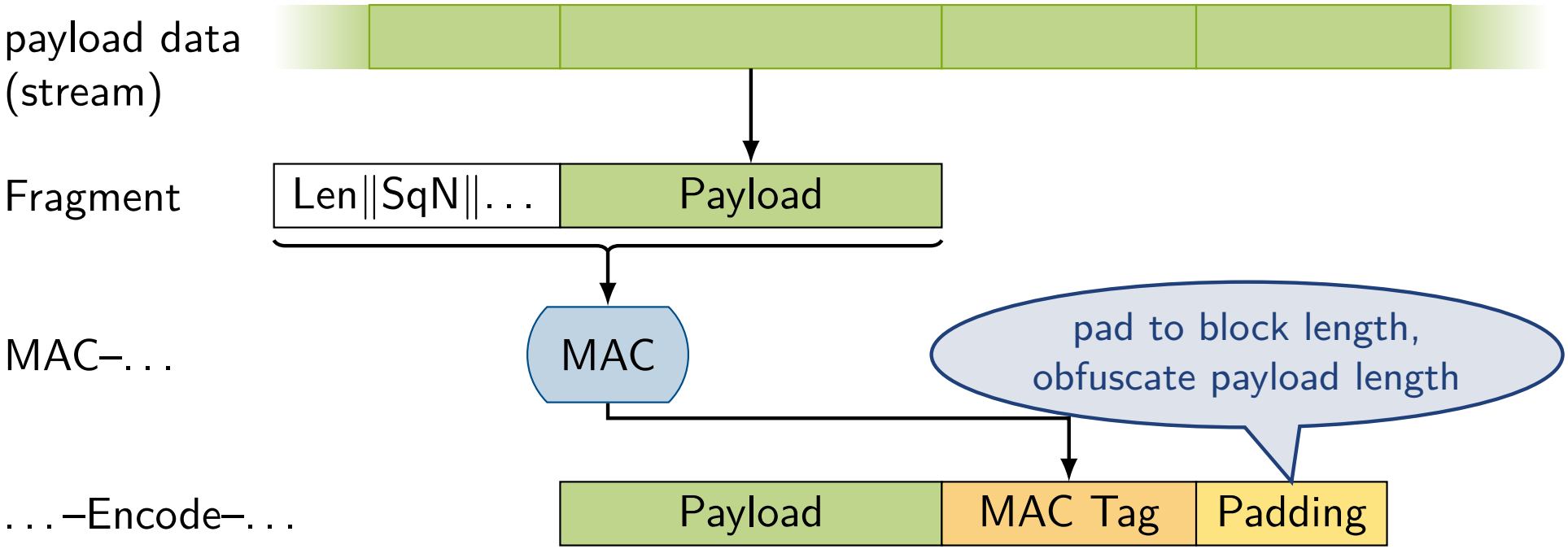
ETH zürich



The TLS Protocol

TLS 1.2 Record Protocol Structure

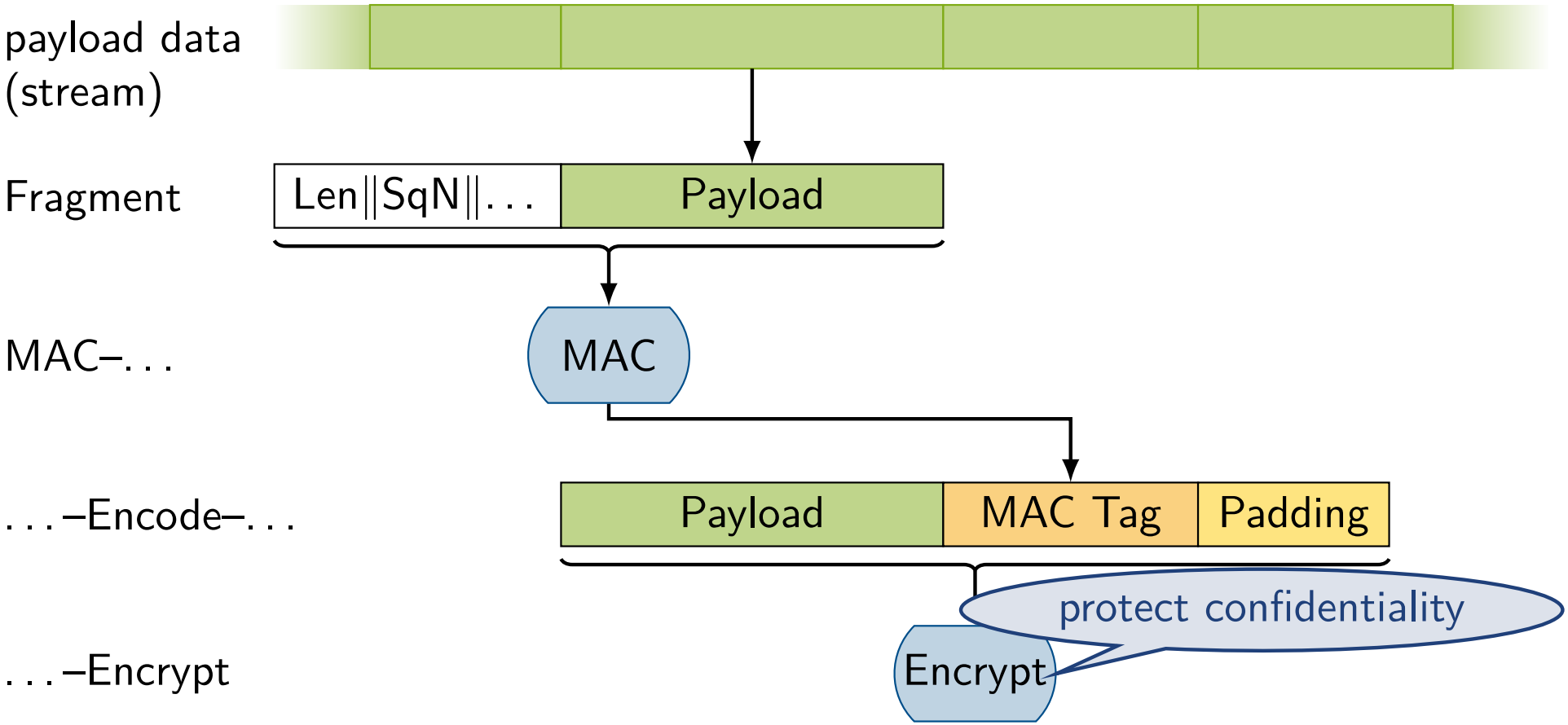
ETH zürich



The TLS Protocol

TLS 1.2 Record Protocol Structure

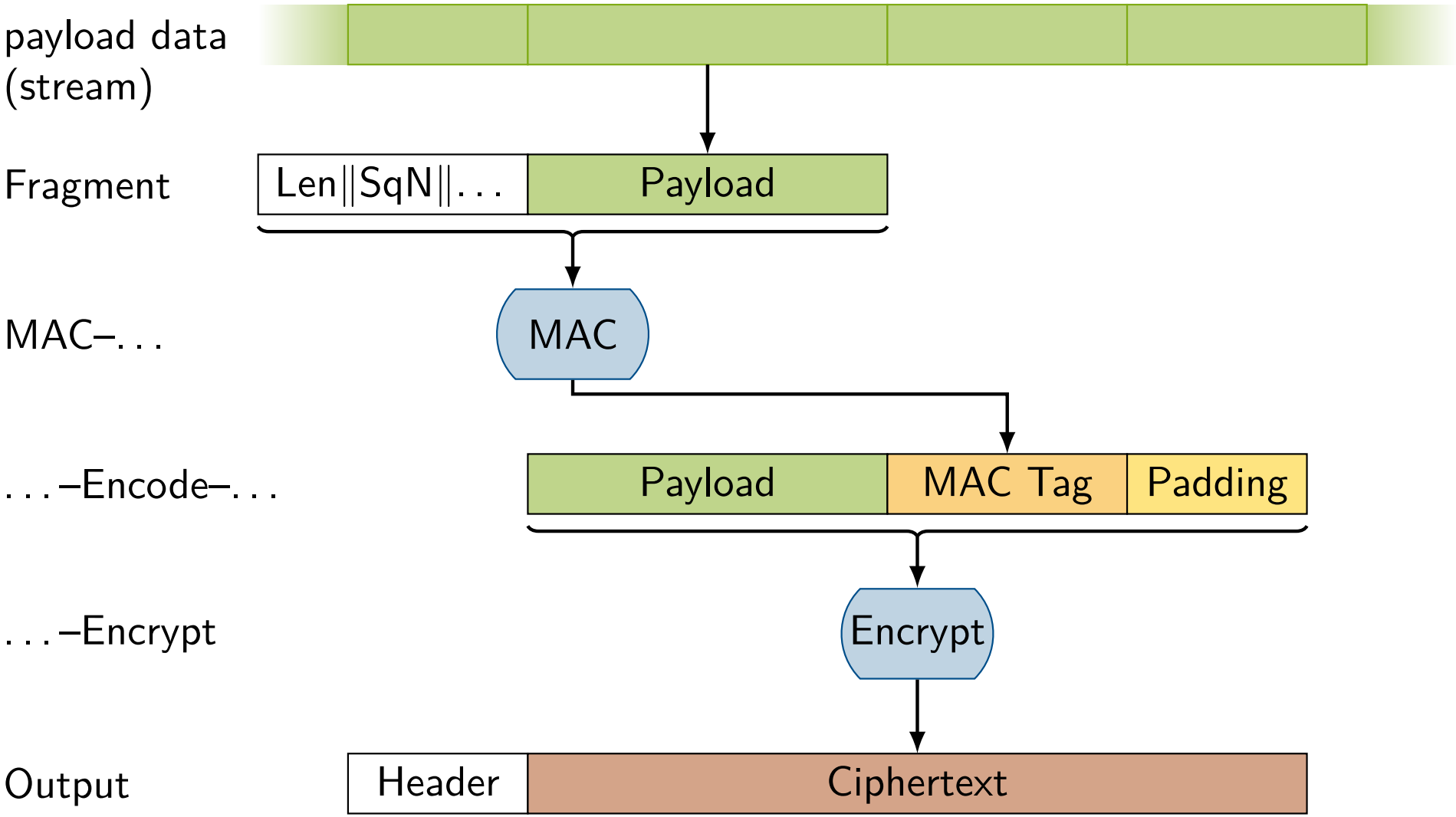
ETH zürich



The TLS Protocol

TLS 1.2 Record Protocol Structure

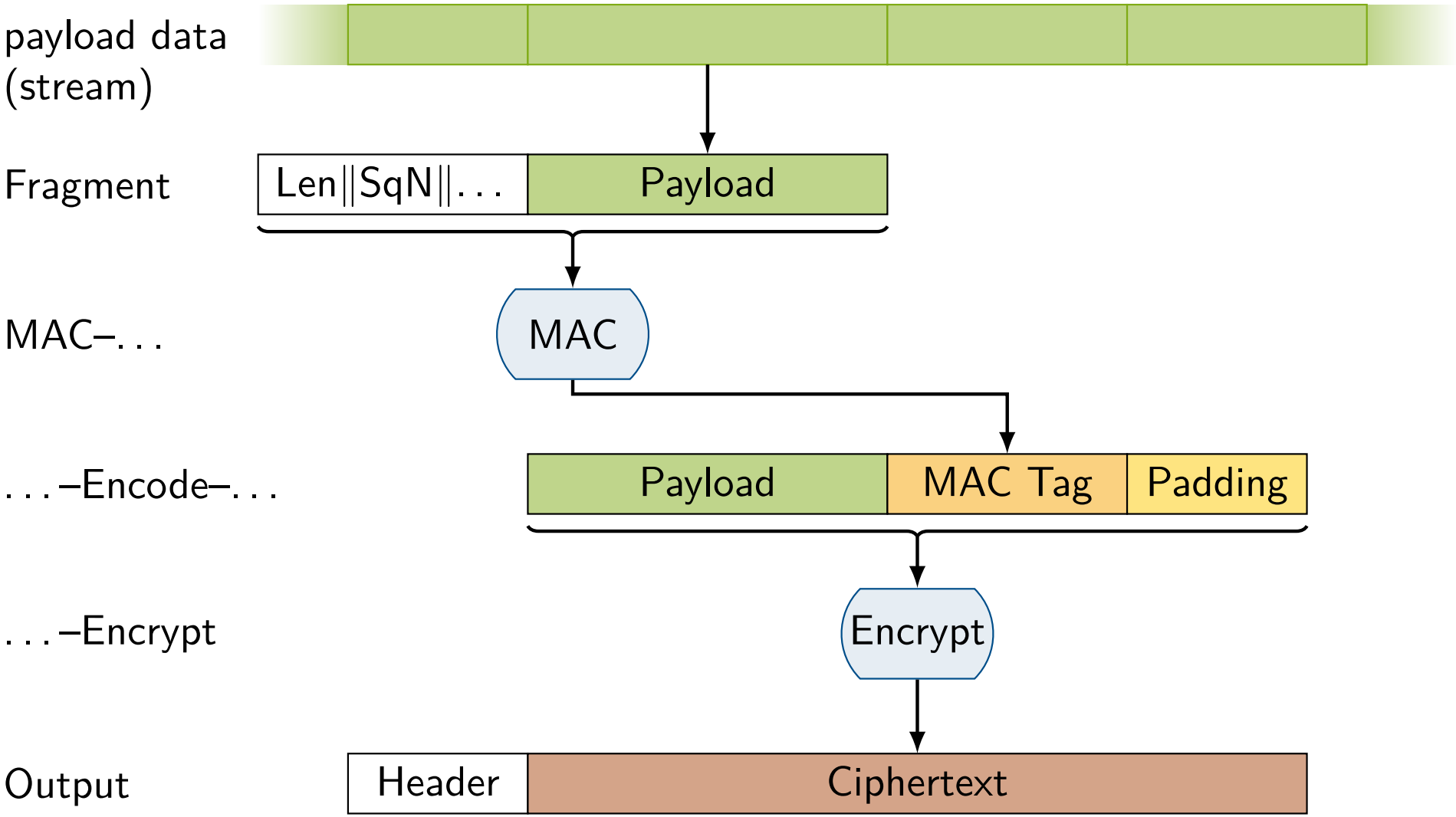
ETH zürich



The TLS Protocol

TLS 1.2 Example: TLS_RSA_WITH_AES_128_CBC_SHA — Record Protocol

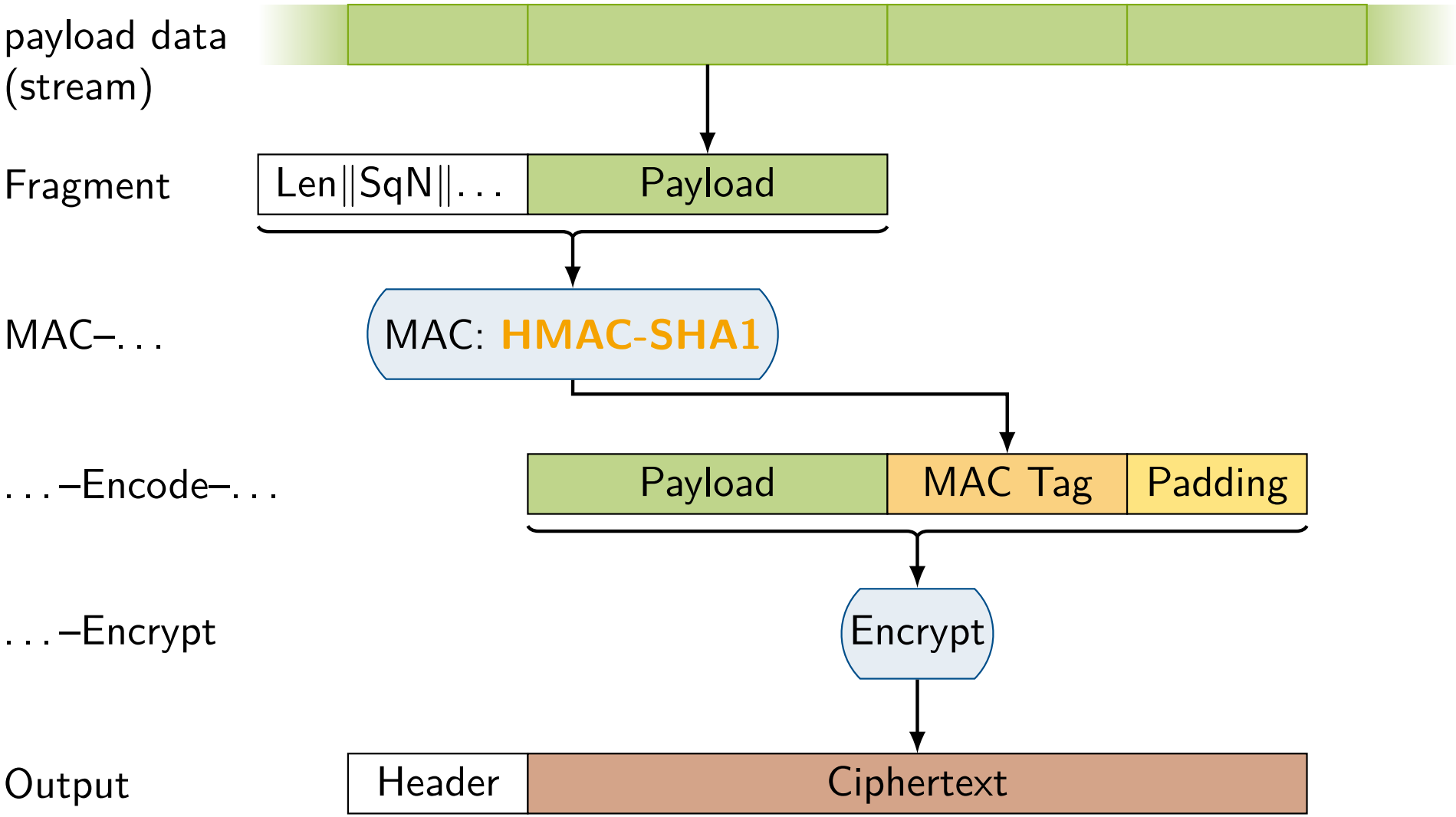
ETH zürich



The TLS Protocol

TLS 1.2 Example: TLS_RSA_WITH_AES_128_CBC_SHA — Record Protocol

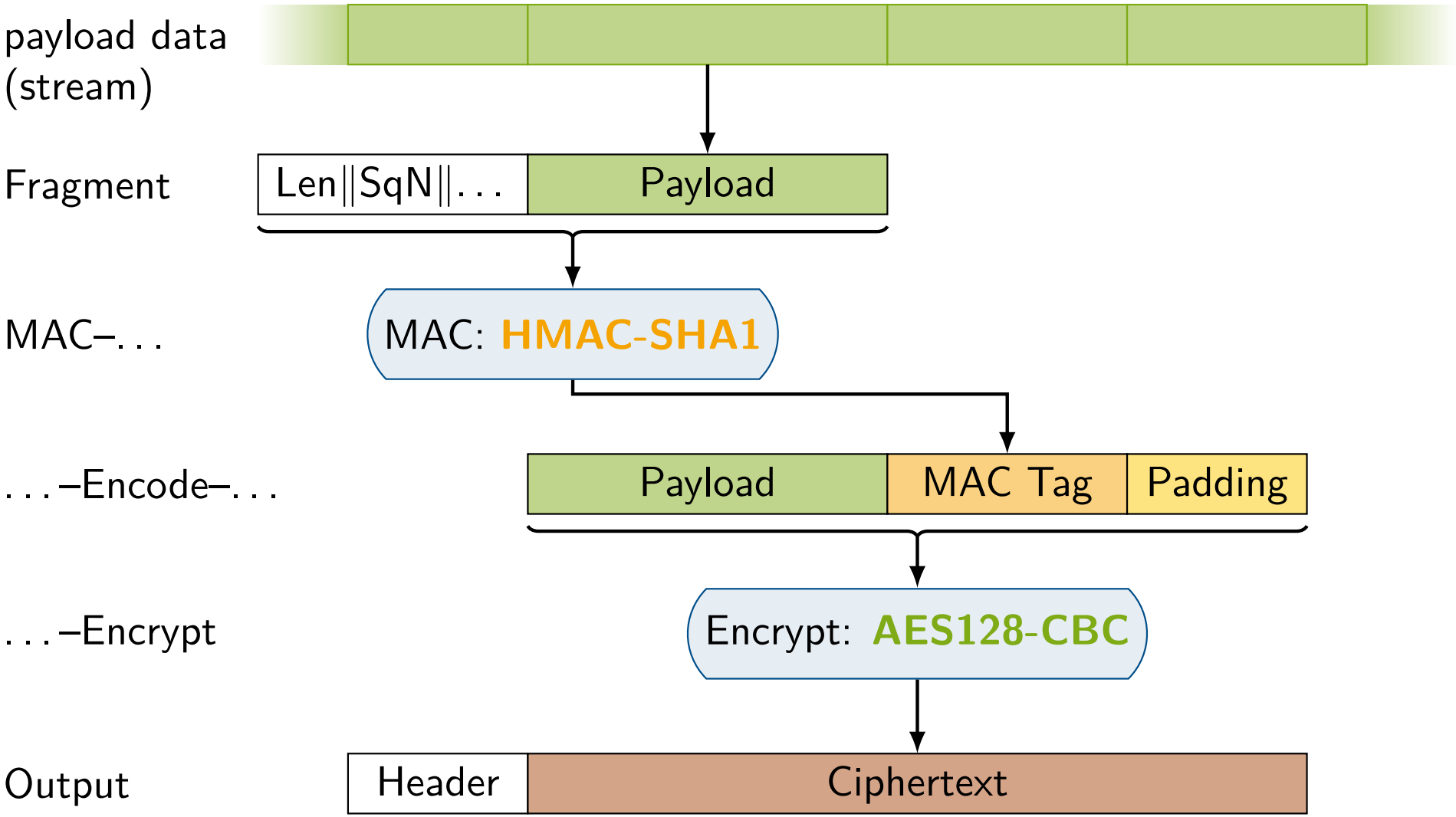
ETH zürich



The TLS Protocol

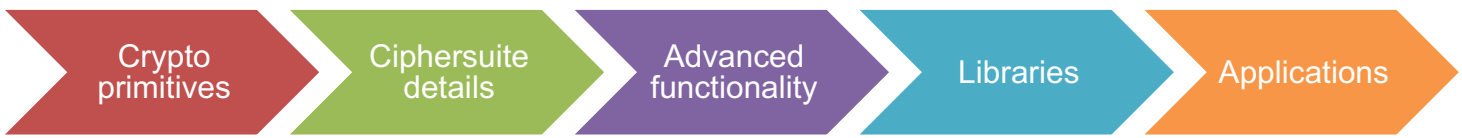
TLS 1.2 Example: TLS_RSA_WITH_AES_128_CBC_SHA — Record Protocol

ETH zürich



Components of TLS

Slides by Douglas Stebila



- RSA, DSA, ECDSA
- Diffie–Hellman, ECDH
- HMAC
- MD5, SHA1, SHA-2
- DES, 3DES, RC4, AES
- Export grade

- Data structures
- Key derivation
- Encryption modes, IVs
- Padding

- Alerts & errors
- Certification / revocation
- Negotiation
- Renegotiation
- Session resumption
- Key reuse
- Compression
- State machine

- OpenSSL
- LibreSSL, BoringSSL
- NSS
- GnuTLS
- SChannel
- Java JSSE
- Everest / miTLS
- s2n

- Web browsers: Chrome, Firefox, IE/Edge, Safari
- Web servers: Apache, IIS, nginx, node, …
- Application SDKs
- Certificates
- Protocols
  - HTTP, IMAP, ..

https://www.douglas.stebila.ca/research/presentations/tls-attacks/

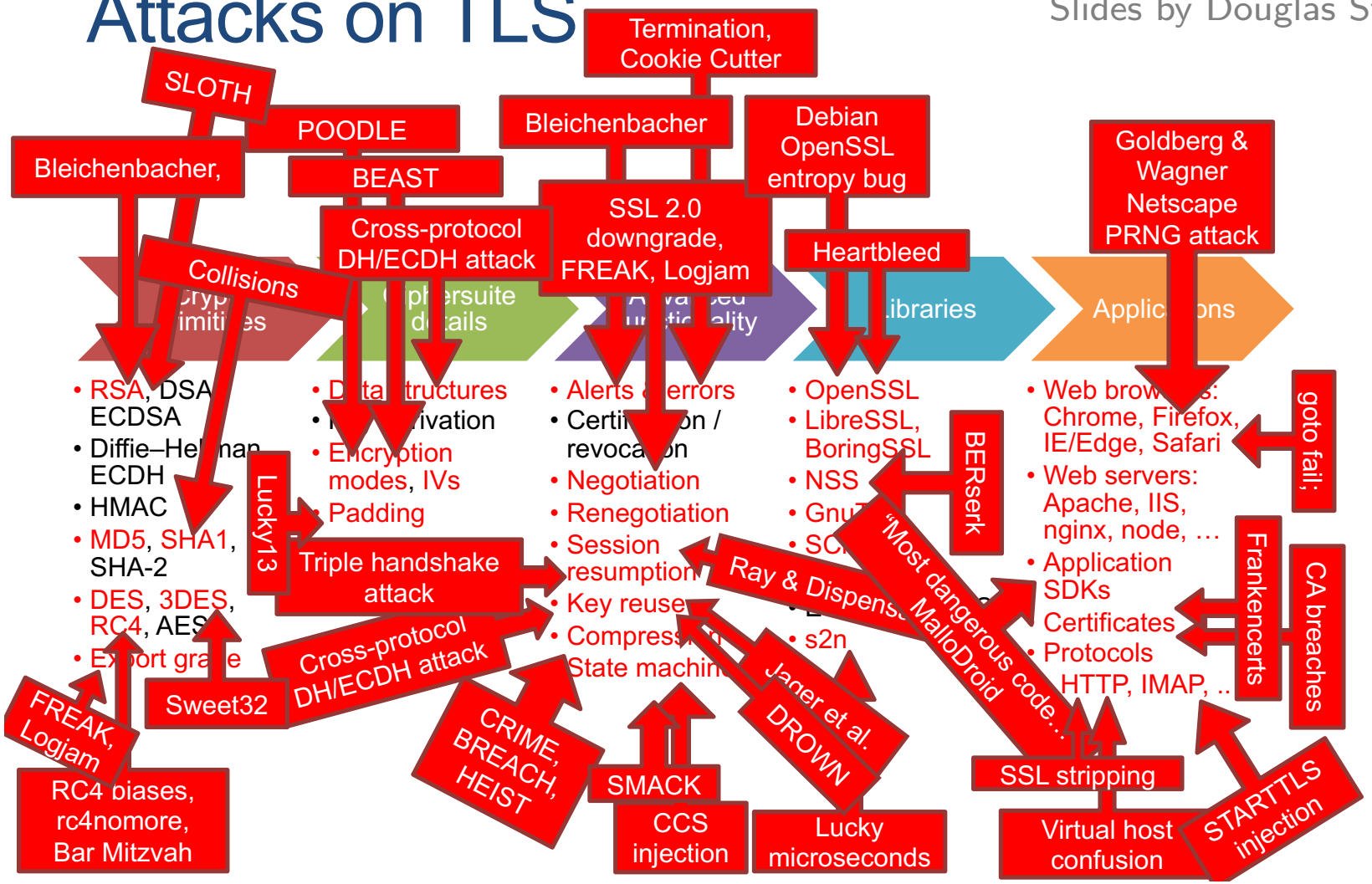
The TLS Protocol

A complex protocol with many subtly interacting sub-components
"What could possibly go wrong?"

ETH zürich

Attacks on TLS

Slides by Douglas Stebila



https://www.douglas.stebila.ca/research/presentations/tls-attacks/

TLS Security Issues

@Core Crypto: PKCS#1v1.5 and Bleichenbacher
— see PKE Lectures

ETH zürich

- core issue: RSA PKCS#1v1.5 is **not CCA-secure**

- **RSA key transport**
  - client selects secret random pre-master secret preMS
  - encrypts with the server's public key:

$$\mathsf{RSA}_{\text{-PKCS\#1v1.5}}.\mathsf{Encrypt}(pk, \mathrm{preMS}) = (00\|02\|padding\|00\|\mathrm{preMS})^e \mod N$$

  - server decrypts, checks padding, computes key from preMS
  - RFC doesn't specify exact process, but signals decryption error

- decryption error signal can be transformed into a **decryption oracle** [Ble98]

TLS Security Issues

@Core Crypto: PKCS#1v1.5 and Bleichenbacher
— see PKE Lectures

ETH zürich

- core issue: RSA PKCS#1v1.5 is **not CCA-secure**

- **RSA key transport**
  - client selects secret random pre-master secret preMS
  - encrypts with the server's public key:

$$\mathsf{RSA}_{\text{-PKCS\#1v1.5}}.\mathsf{Encrypt}(pk, \mathrm{preMS}) = (00\|02\|\mathit{padding}\|00\|\mathrm{preMS})^e \mod N$$

  - server decrypts, checks padding, computes key from preMS
  - RFC doesn't specify exact process, but signals decryption error

- decryption error signal can be transformed into a **decryption oracle** [Ble98]

- instead of switch to CCA-secure RSA-OAEP, TLS tried hiding error signal
- but Bleichenbacher-style attacks came back repeatedly
- to make things worse: re-use of RSA keys across different SSL/TLS versions
  - downgrade TLS version & attack there – e.g., DROWN attack [Avi+16]

TLS Security Issues

@Core Crypto: MAC-Encode-Encrypt and Lucky13 — see AEAD Lectures, Extra Slides

ETH zürich

- core issue: (good) MAC –then– (good) Encrypt $\neq$ **CCA-secure** AE [BN00]
- **MAC–then–AES-CBC Decryption**
  - decrypt ciphertext to obtain Payload $\|$ MAC Tag $\|$ Padding
  - remove padding — what if padding is incorrect?
  - check MAC
- A padding oracle
  - in a modified ciphertext, either the padding check fails...
  - ... or the MAC check fails
  - if the two are distinguishable: padding oracle
  - can lift a padding oracle to a **decryption oracle** [Vau02] (conditions apply)

TLS Security Issues

@Core Crypto: MAC-Encode-Encrypt and Lucky13
— see AEAD Lectures, Extra Slides

ETH zürich

- core issue: (good) MAC –then– (good) Encrypt $\neq$ **CCA-secure** AE [BN00]
- **MAC–then–AES-CBC Decryption**
  - decrypt ciphertext to obtain Payload $\|$ MAC Tag $\|$ Padding
  - remove padding — what if padding is incorrect?
  - check MAC
- A padding oracle
  - in a modified ciphertext, either the padding check fails. . .
  - . . . or the MAC check fails
  - if the two are distinguishable: padding oracle
  - can lift a padding oracle to a **decryption oracle** [Vau02] (conditions apply)
- instead of switch to CCA-secure Enc-then-MAC, TLS tried hiding error signal
  - "compute MAC w/ zero padding"
  - "leaves a [non-exploitable] small timing channel"
  - **Lucky13** [AP13]: HMAC timing difference still big enough
  - really need constant time—which is extremely difficult!

TLS Security Issues

@Protocol Design: Weak DH Negotiation and Logjam

ETH zürich

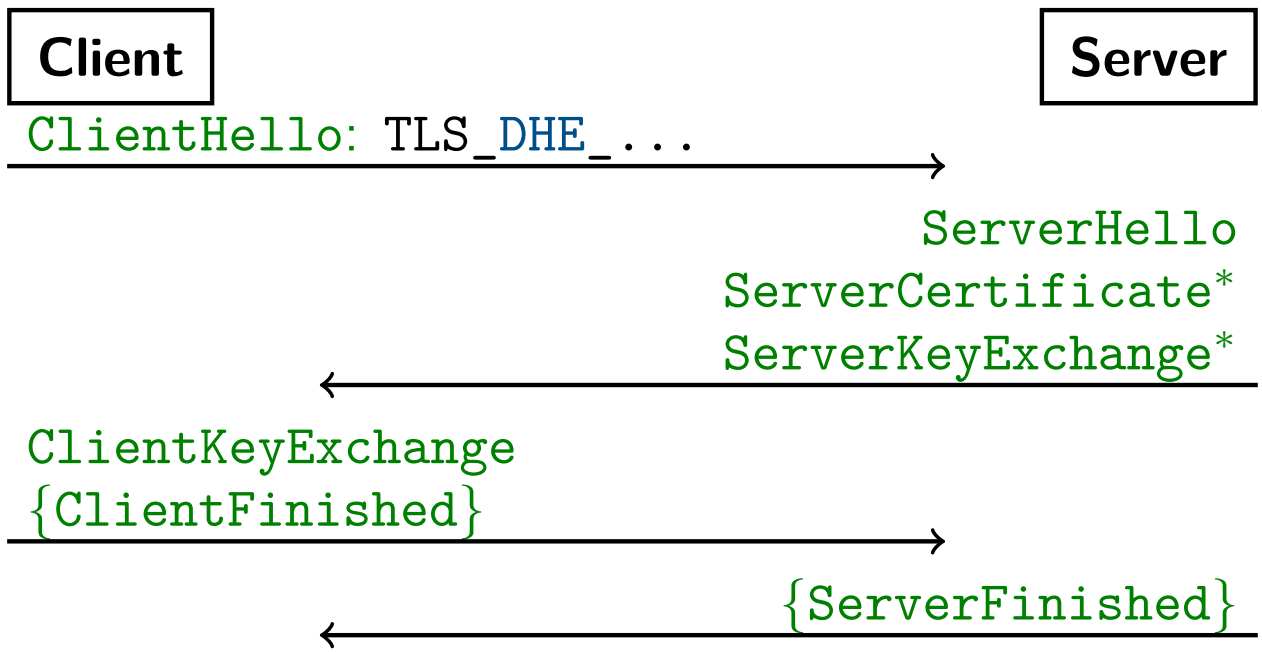
- core issue: weak algorithms make strong ones fail through **downgrades**

TLS Security Issues

@Protocol Design: Weak DH Negotiation and Logjam

ETH zürich

- core issue: weak algorithms make strong ones fail through **downgrades**



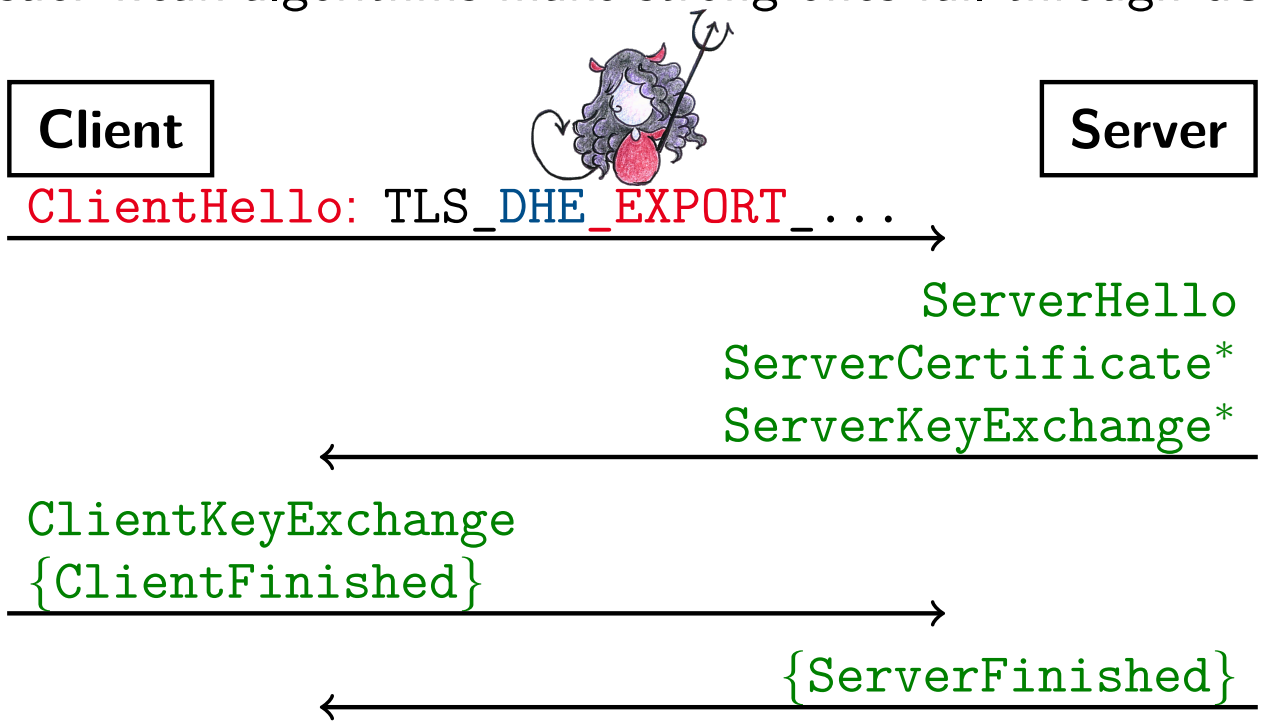
drawings by *Giorgia Azzurra Marson*

TLS Security Issues

@Protocol Design: Weak DH Negotiation and Logjam

ETH zürich

- core issue: weak algorithms make strong ones fail through **downgrades**



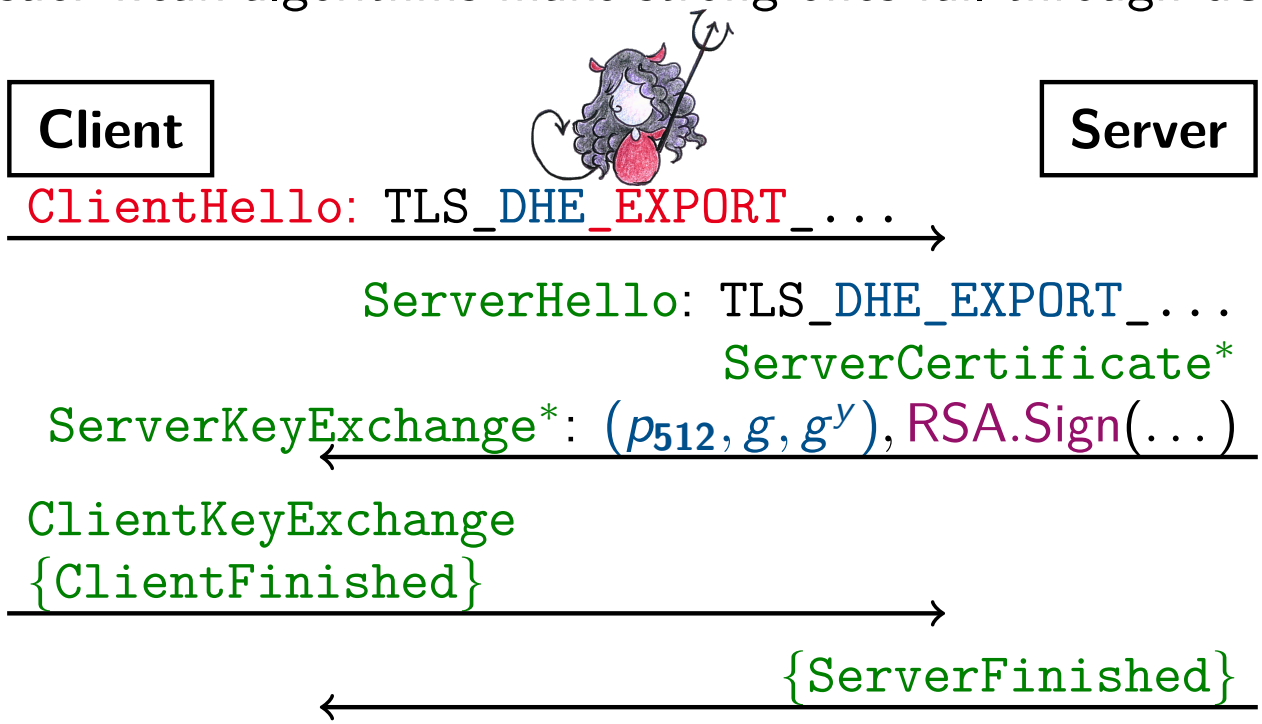
drawings by *Giorgia Azzurra Marson*

TLS Security Issues

@Protocol Design: Weak DH Negotiation and Logjam

ETH zürich

- core issue: weak algorithms make strong ones fail through **downgrades**



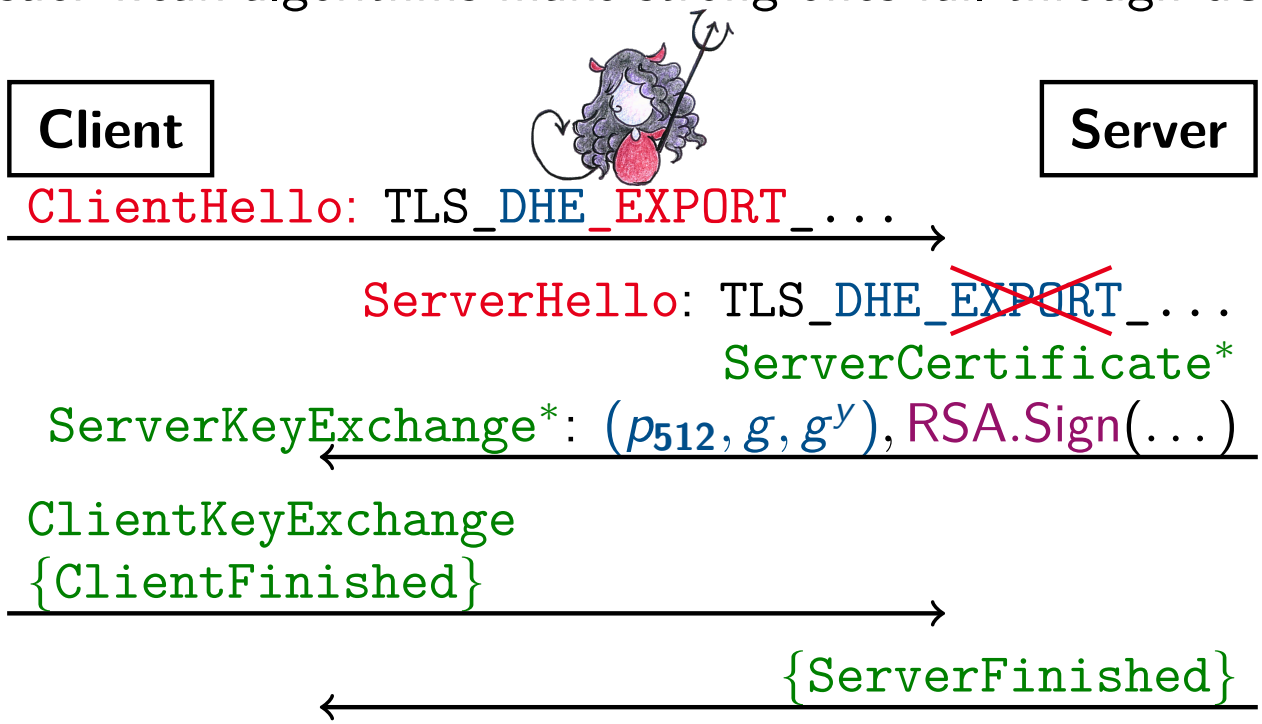
drawings by *Giorgia Azzurra Marson*

TLS Security Issues

@Protocol Design: Weak DH Negotiation and Logjam

ETH zürich

- core issue: weak algorithms make strong ones fail through **downgrades**



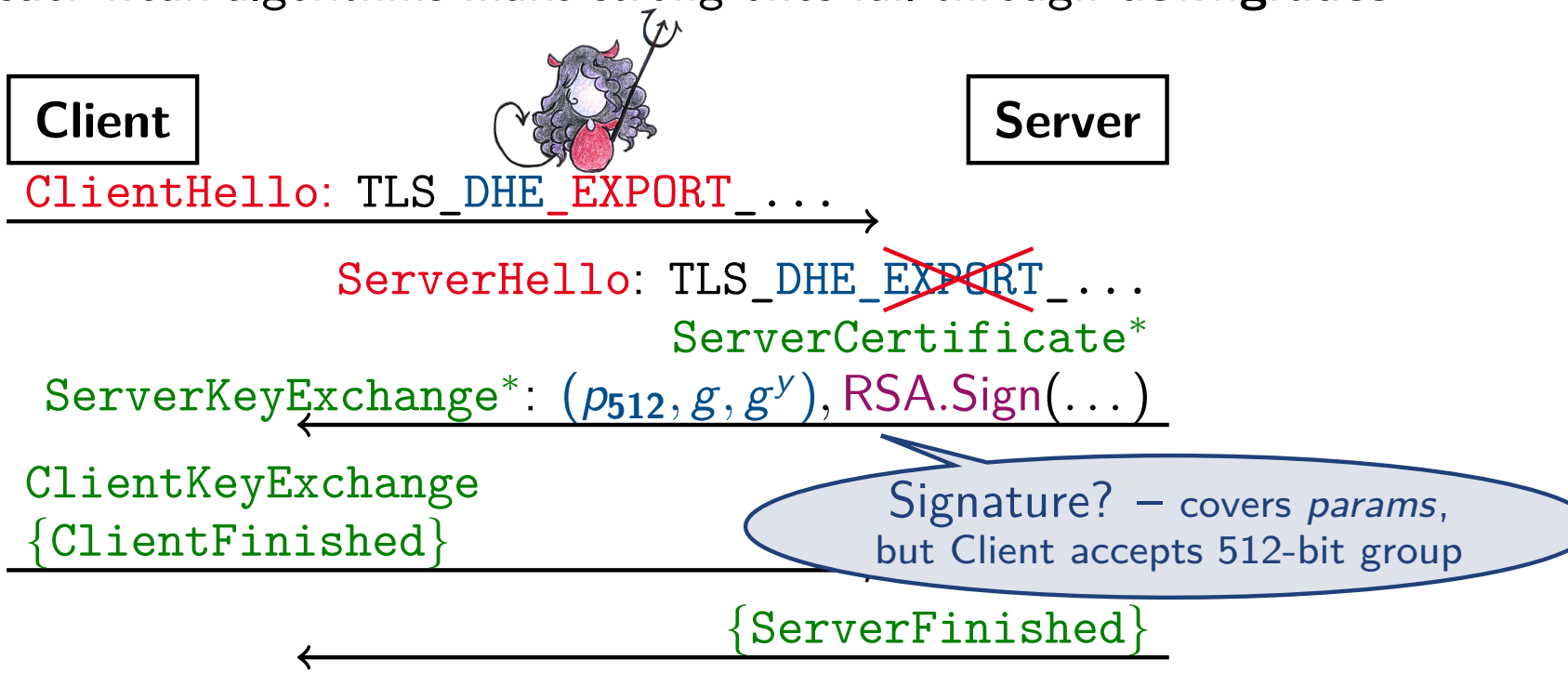
drawings by *Giorgia Azzurra Marson*

TLS Security Issues

@Protocol Design: Weak DH Negotiation and Logjam

ETH zürich

- core issue: weak algorithms make strong ones fail through **downgrades**



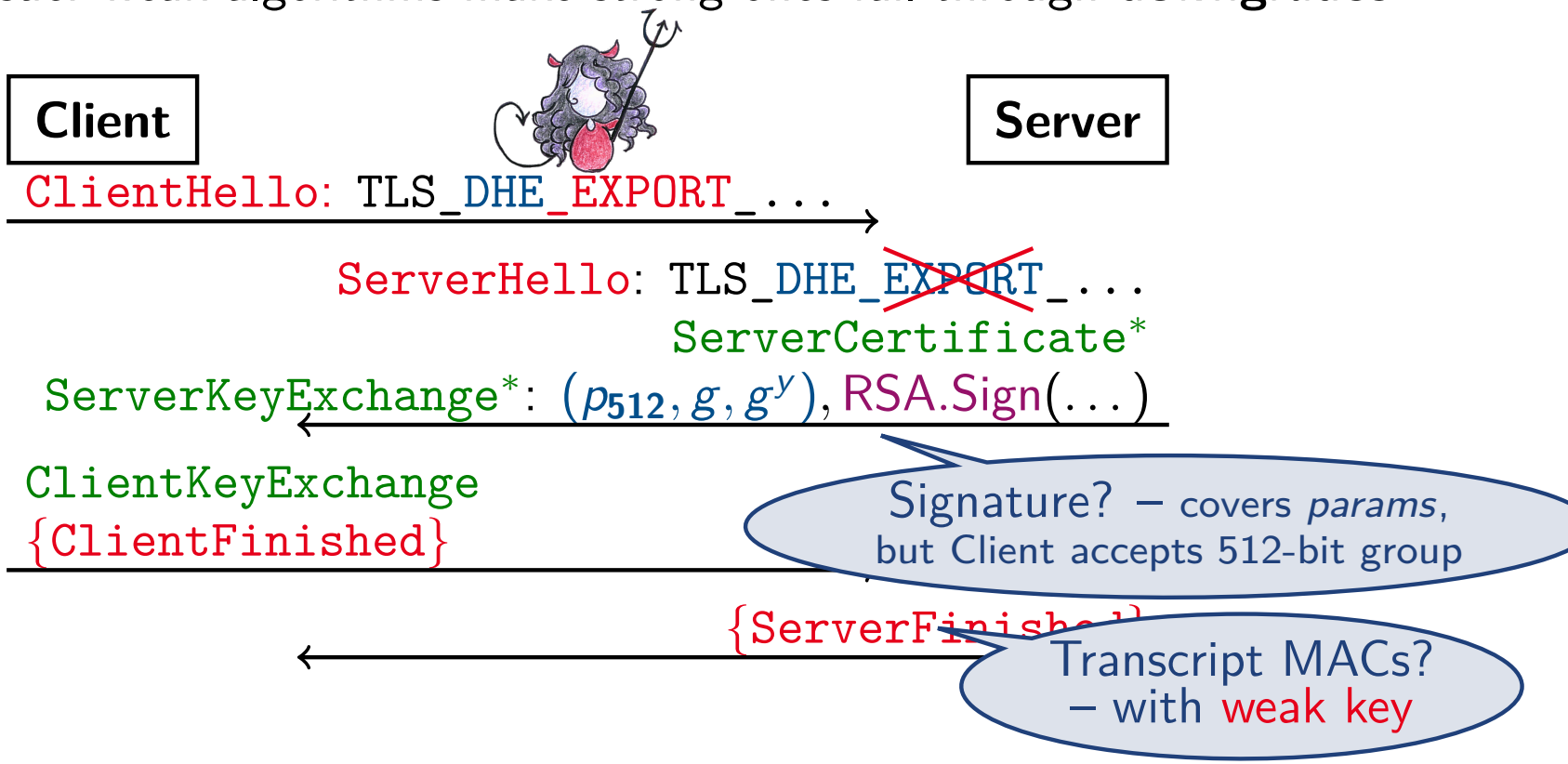
drawings by *Giorgia Azzurra Marson*

TLS Security Issues

@Protocol Design: Weak DH Negotiation and Logjam

ETH zürich

- core issue: weak algorithms make strong ones fail through **downgrades**



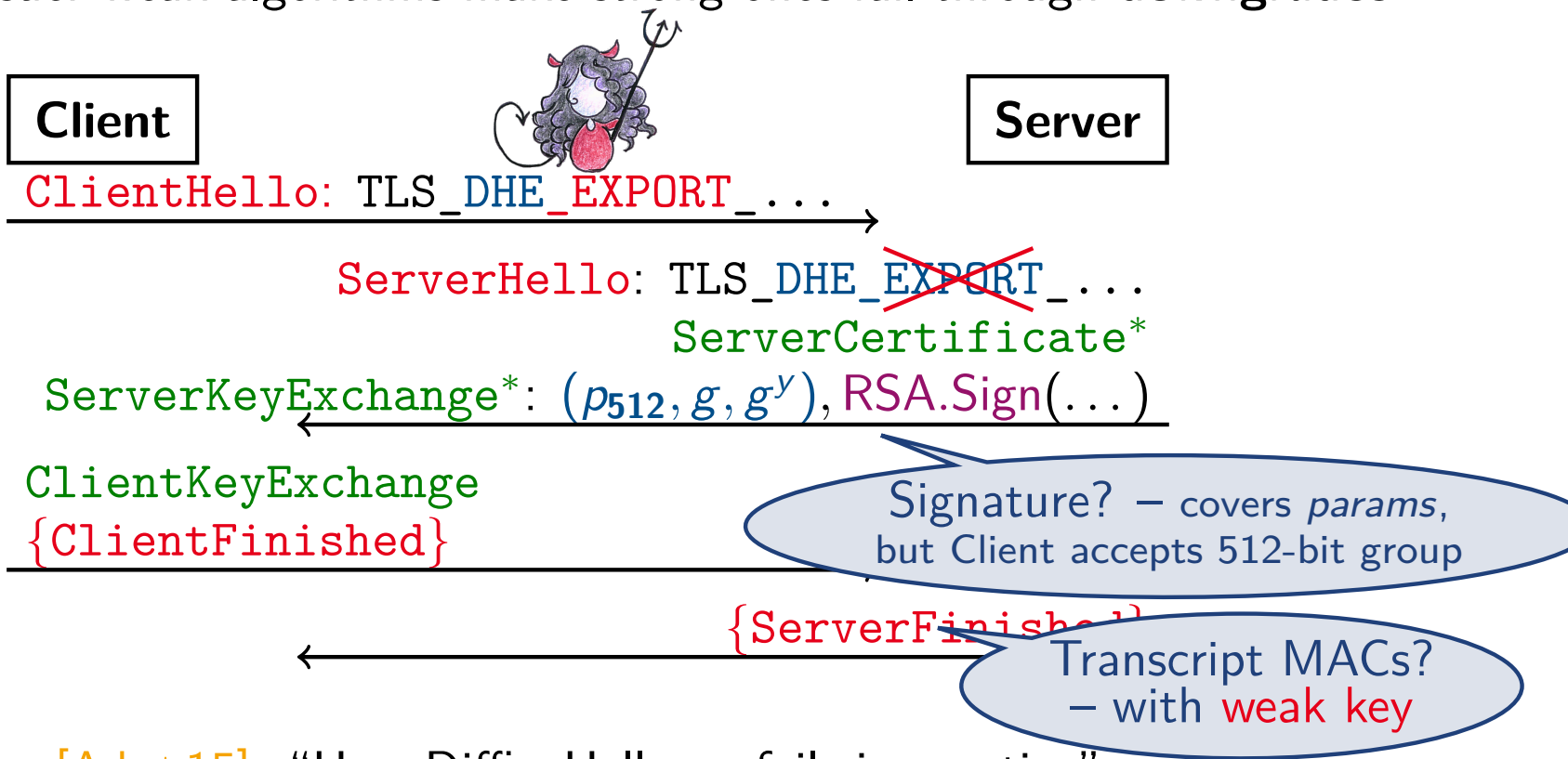
drawings by *Giorgia Azzurra Marson*

TLS Security Issues

@Protocol Design: Weak DH Negotiation and Logjam

ETH zürich

- core issue: weak algorithms make strong ones fail through **downgrades**



- **Logjam** [Adr+15]: "How Diffie–Hellman fails in practice"
  - server impersonation through (also) supporting weak DH groups

drawings by *Giorgia Azzurra Marson*

TLS Security Issues

@Implementation: Buffers and Heartbleed

ETH zürich

- core issue: **buffer over-read** in OpenSSL

TLS Security Issues

@Implementation: Buffers and Heartbleed

ETH zürich

- core issue: **buffer over-read** in OpenSSL
- **Heartbeat** extension (RFC 6520)
  - client sends "ping back those 4 bytes: `00 01 02 03`"
  - server responds "`00 01 02 03`"

TLS Security Issues

@Implementation: Buffers and Heartbleed

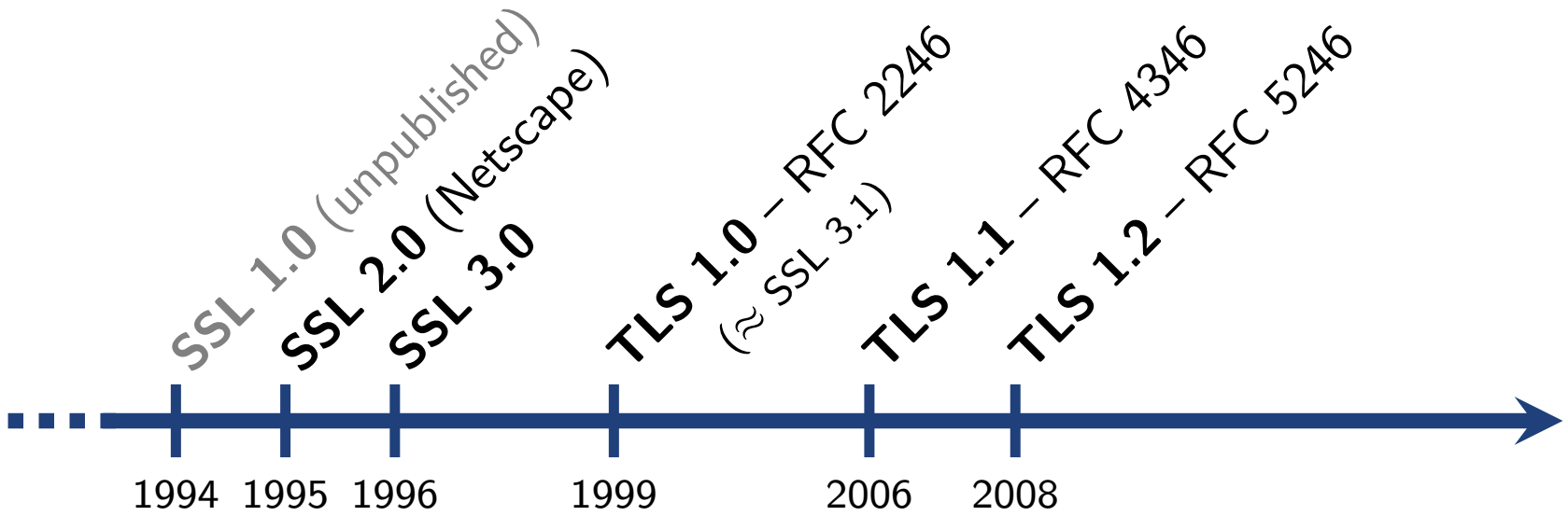
ETH zürich

- core issue: **buffer over-read** in OpenSSL
- **Heartbeat** extension (RFC 6520)
  - client sends "ping back those 4 bytes: `00 01 02 03`"
  - server responds "`00 01 02 03`"
- **Heartbleed** attack [Hea]
  - client sends "ping back those **16 Kbytes**: `00 01 02 03`"
  - server responds "`00 01 02 03 ...<memory dump>`"
  - possibly including sensitive data like server private key etc.
- high severity & public attention — and a catchy logo

Transport Layer Security (TLS)

A new chapter: TLS 1.3

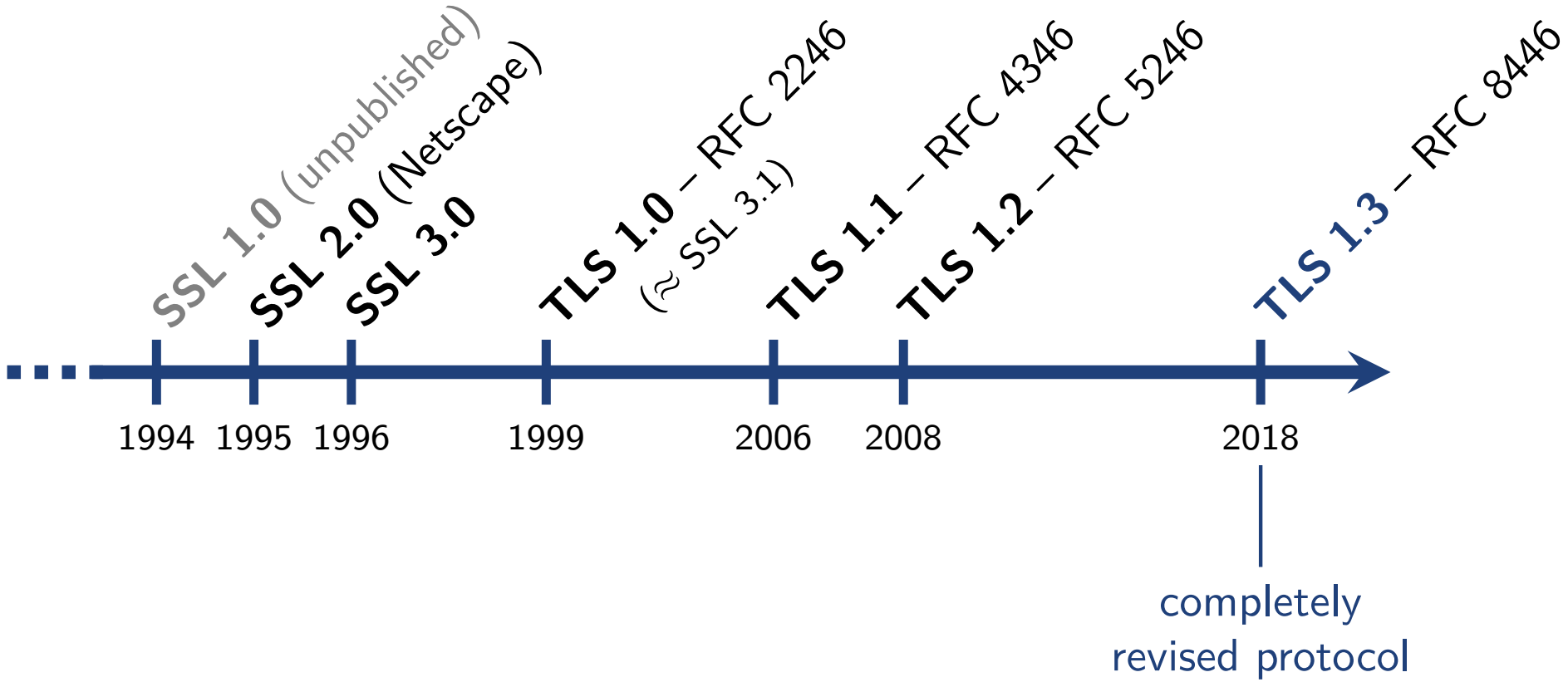
ETH zürich



Transport Layer Security (TLS)

A new chapter: TLS 1.3

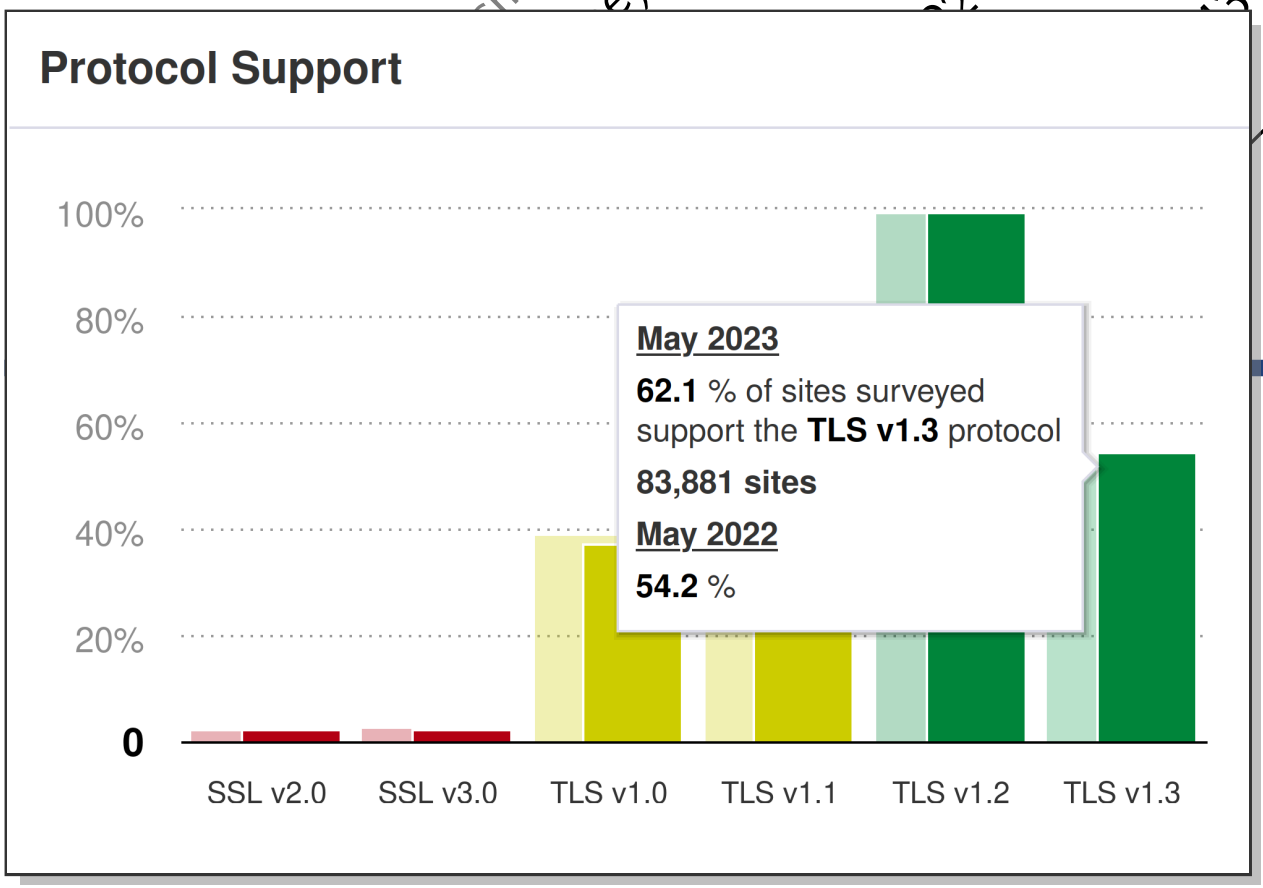
ETH zürich

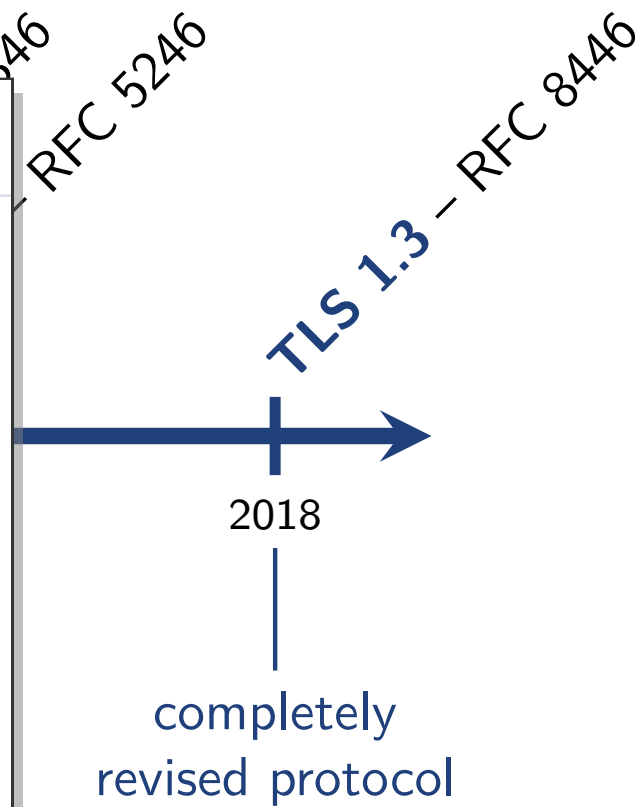


Transport Layer Security (TLS)

A new chapter: TLS 1.3

ETH zürich



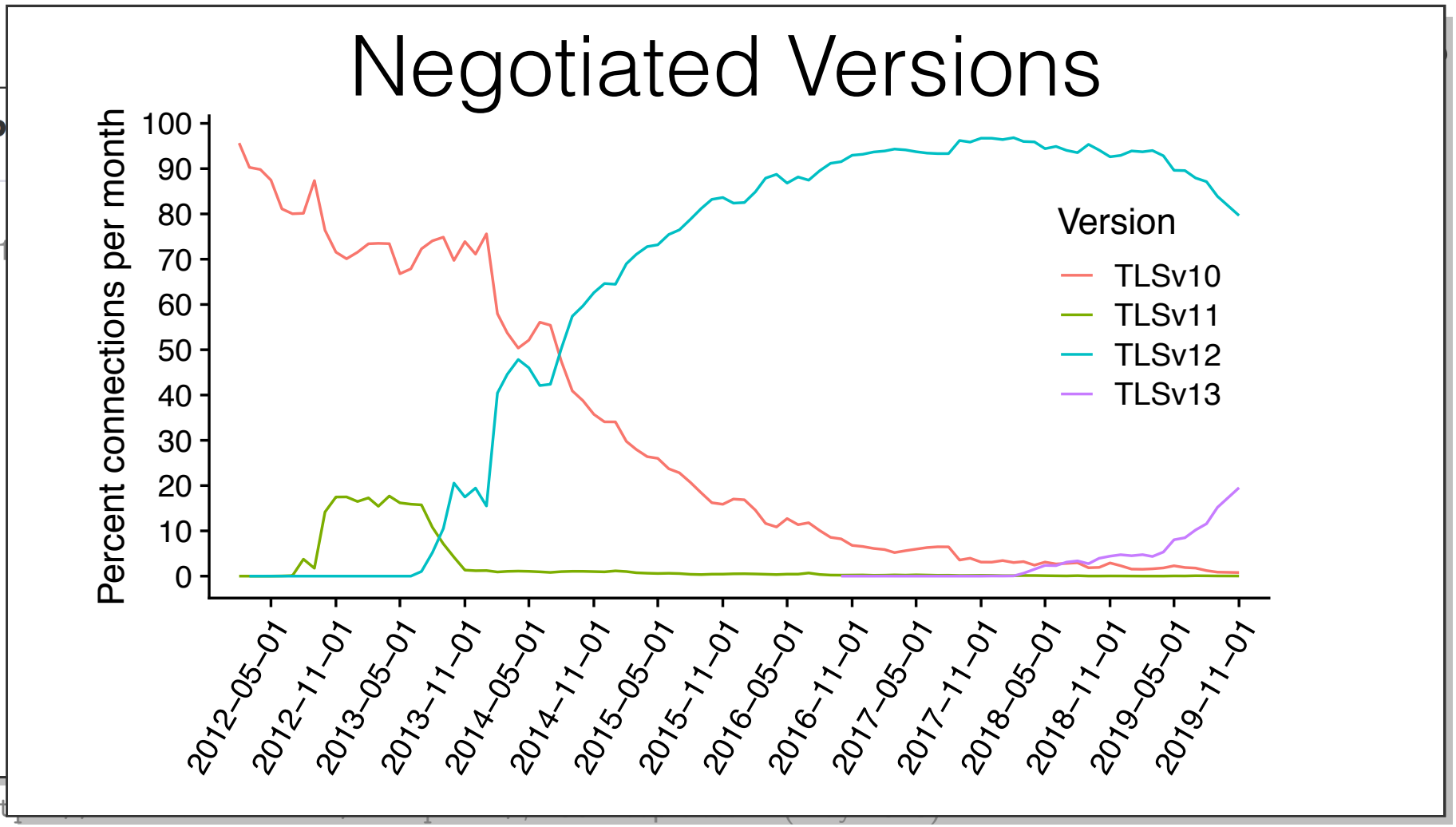


https://www.ssllabs.com/ssl-pulse/, 150K top sites (May 2023)

Transport Layer Security (TLS)

A new chapter: TLS 1.3

ETH zürich



ICSI SSL Notary (Johanna Amann @ RWC 2020)

TLS 1.3

Design Goals

ETH zürich

- **Clean up:** get rid of flawed and unused crypto & features
- **Improve latency:** for main handshake and repeated connections (while maintaining security)
- **Improve privacy:** hide as much of the handshake as possible
- **Continuity:** maintain interoperability with previous versions and support existing important use cases

TLS 1.3

Design Goals

ETH zürich

- **Clean up:** get rid of flawed and unused crypto & features
- **Improve latency:** for main handshake and repeated connections (while maintaining security)
- **Improve privacy:** hide as much of the handshake as possible
- **Continuity:** maintain interoperability with previous versions and support existing important use cases
- **Security Assurance (added later):** have supporting analyses for changes

TLS 1.3

Main changes (from TLS 1.2)

ETH zürich

Clean up

TLS 1.3

Main changes (from TLS 1.2)

ETH zürich

Clean up

- removed **legacy and broken crypto**
  - ciphers: (3)DES, RC4, . . . , MtE (CBC & generally) — only AEAD remains
  - hash functions: MD5, SHA1
  - authentication: Kerberos, RSA PKCS#1v1.5 key transport
  - custom (EC)DHE groups

TLS 1.3

Main changes (from TLS 1.2)

ETH zürich

Clean up

- removed **legacy and broken crypto**
  - ciphers: (3)DES, RC4, . . . , MtE (CBC & generally) — only AEAD remains
  - hash functions: MD5, SHA1
  - authentication: Kerberos, RSA PKCS#1v1.5 key transport
  - custom (EC)DHE groups
- removed **broken features**
  - compression
  - renegotiation (but added key updates + late client auth)

TLS 1.3

Main changes (from TLS 1.2)

ETH zürich

Clean up

- removed **legacy and broken crypto**
  - ciphers: (3)DES, RC4, . . . , MtE (CBC & generally) — only AEAD remains
  - hash functions: MD5, SHA1
  - authentication: Kerberos, RSA PKCS#1v1.5 key transport
  - custom (EC)DHE groups
- removed **broken features**
  - compression
  - renegotiation (but added key updates + late client auth)
- removed **static RSA/DH**: public-key crypto = forward security

TLS 1.3

Main changes (from TLS 1.2)

ETH zürich

Clean up

- removed **legacy and broken crypto**
  - ciphers: (3)DES, RC4, . . . , MtE (CBC & generally) — only AEAD remains
  - hash functions: MD5, SHA1
  - authentication: Kerberos, RSA PKCS#1v1.5 key transport
  - custom (EC)DHE groups
- removed **broken features**
  - compression
  - renegotiation (but added key updates + late client auth)
- removed **static RSA/DH**: public-key crypto = forward security

quite some resistance from enterprises doing passive inspection

TLS 1.3

Main changes (from TLS 1.2)

ETH zürich

Clean up

- removed **legacy and broken crypto**
  - ciphers: (3)DES, RC4, . . . , MtE (CBC & generally) — only AEAD remains
  - hash functions: MD5, SHA1
  - authentication: Kerberos, RSA PKCS#1v1.5 key transport
  - custom (EC)DHE groups
- removed **broken features**
  - compression
  - renegotiation (but added key updates + late client auth)
- removed **static RSA/DH**: public-key crypto = forward security
- clean **key derivation** based on Extract-then-Expand HKDF

TLS 1.3

Main changes (from TLS 1.2)

ETH zürich

Clean up

- removed **legacy and broken crypto**
  - ciphers: (3)DES, RC4, . . . , MtE (CBC & generally) — only AEAD remains
  - hash functions: MD5, SHA1
  - authentication: Kerberos, RSA PKCS#1v1.5 key transport
  - custom (EC)DHE groups
- removed **broken features**
  - compression
  - renegotiation (but added key updates + late client auth)
- removed **static RSA/DH**: public-key crypto = forward security
- clean **key derivation** based on Extract-then-Expand HKDF
- **hardened negotiation** of version/cipher suite against downgrades

ETH zürich

Improve latency

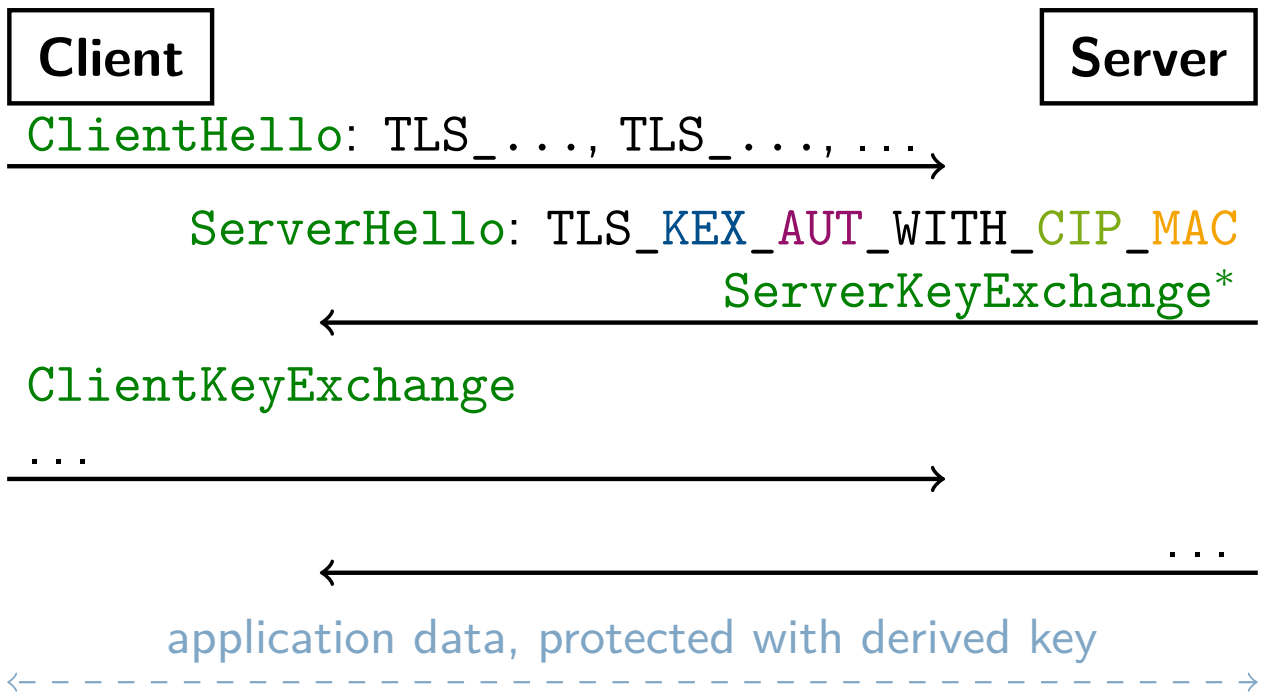
TLS 1.3

Main changes (from TLS 1.2)

ETH zürich

Improve latency

- TLS $\leq$ 1.2 is slow: 2 round trips before client can send data



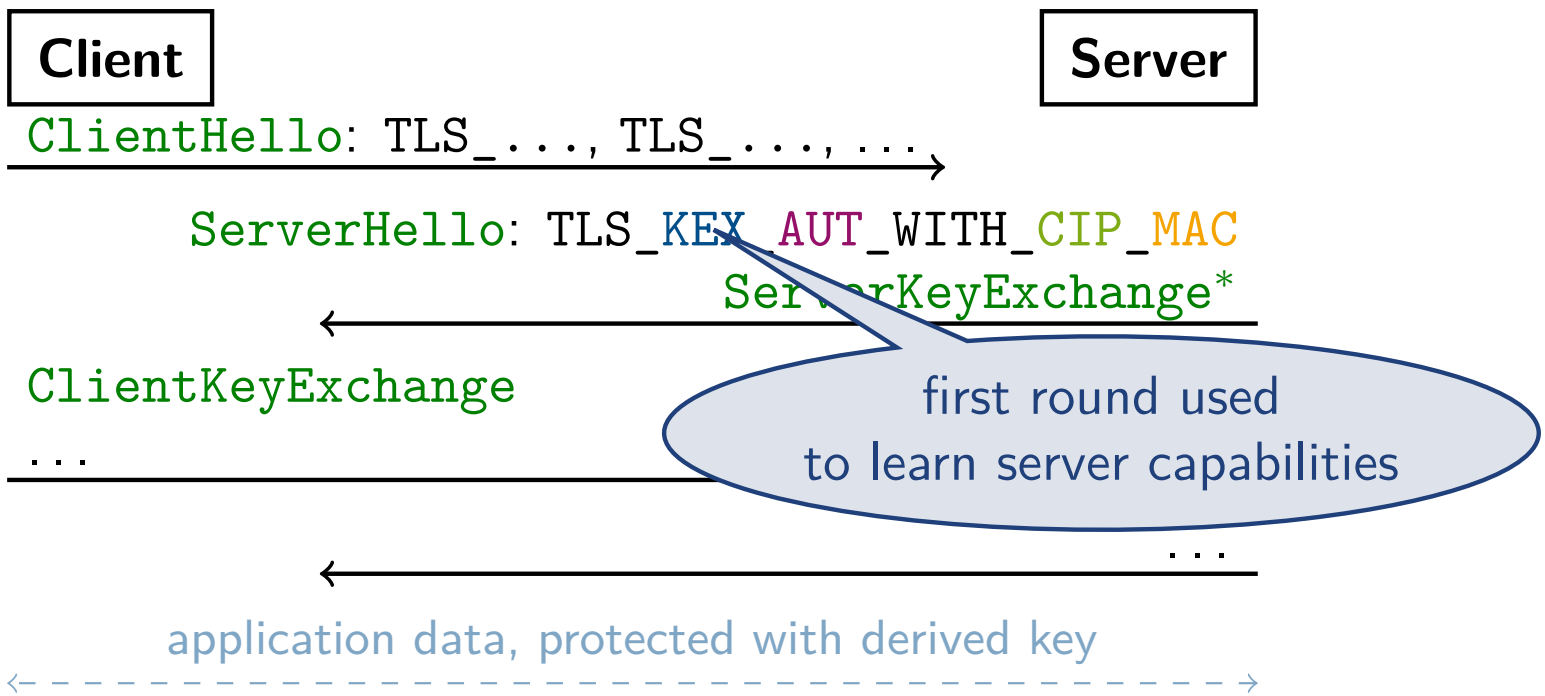
TLS 1.3

Main changes (from TLS 1.2)

ETH zürich

Improve latency

- TLS $\leq$ 1.2 is slow: 2 round trips before client can send data



TLS 1.3

Main changes (from TLS 1.2)

ETH zürich

Improve latency

- TLS $\leq$ 1.2 is slow: 2 round trips before client can send data

- TLS 1.3: **full handshake in 1 round trip**
  - feature reduction $\rightarrow$ always do (EC)DHE
  - client speculatively sends several DH shares in supported groups
  - server picks one, replies with its share, and key can be already derived

TLS 1.3

Main changes (from TLS 1.2)

ETH zürich

Improve latency

- TLS $\leq$ 1.2 is slow: 2 round trips before client can send data
- TLS 1.3: **full handshake in 1 round trip**
  - feature reduction $\rightarrow$ always do (EC)DHE
  - client speculatively sends several DH shares in supported groups
  - server picks one, replies with its share, and key can be already derived
- **0-RTT handshake** when resuming previous connection
  - client+server keep shared resumption secret (PSK)
  - client derives a key from that and can immediately encrypt data
  - but: 0-RTT *sacrifices* replay protection (see extra slides)

TLS 1.3

Main changes (from TLS 1.2)

ETH zürich

Improve privacy

TLS 1.3

Main changes (from TLS 1.2)

ETH zürich

Improve privacy

- TLS $\leq$ 1.2: entire handshake in the clear (incl. certificates, extensions)

TLS 1.3

Main changes (from TLS 1.2)

ETH zürich

Improve privacy

- TLS $\leq$ 1.2: entire handshake in the clear (incl. certificates, extensions)
- TLS 1.3: **encrypts almost all handshake messages**
  - derive separate key early to protect handshake messages
  - provides security against passive/active attackers (for server/client)

TLS 1.3

Main changes (from TLS 1.2)

ETH zürich

Improve privacy

- TLS $\leq$ 1.2: entire handshake in the clear (incl. certificates, extensions)
- TLS 1.3: **encrypts almost all handshake messages**
  - derive separate key early to protect handshake messages
  - provides security against passive/active attackers (for server/client)

Continuity

TLS 1.3

Main changes (from TLS 1.2)

ETH zürich

Improve privacy

- TLS $\leq$ 1.2: entire handshake in the clear (incl. certificates, extensions)
- TLS 1.3: **encrypts almost all handshake messages**
  - derive separate key early to protect handshake messages
  - provides security against passive/active attackers (for server/client)

Continuity

- e.g.: remove complex renegotiation, but keep features (key update + client auth)

TLS 1.3

Main changes (from TLS 1.2)

ETH zürich

Improve privacy

- TLS $\leq$ 1.2: entire handshake in the clear (incl. certificates, extensions)
- TLS 1.3: **encrypts almost all handshake messages**
  - derive separate key early to protect handshake messages
  - provides security against passive/active attackers (for server/client)

Continuity

- e.g.: remove complex renegotiation, but keep features (key update + client auth)
- interoperability (idea): let `ClientHello` look like TLS $<$ 1.3

TLS 1.3

Main changes (from TLS 1.2)

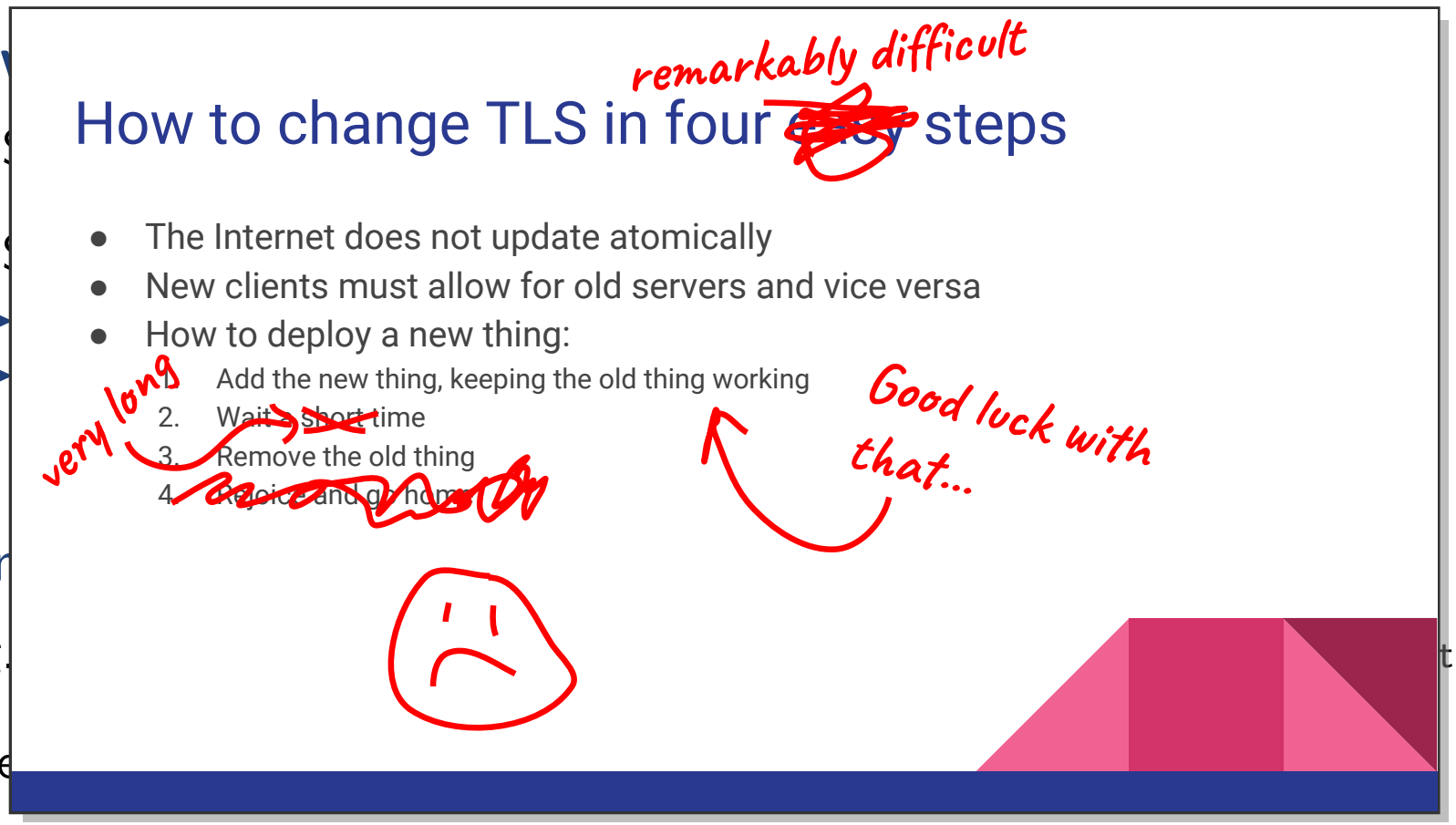
ETH zürich

Improv

- TL
- TL
  -
  -

Contin

- e.g. t auth)
- inte



(David Benjamin @ RWC 2018: "TLS Ecosystem Woes: Why your Crypto isn't Real World yet")

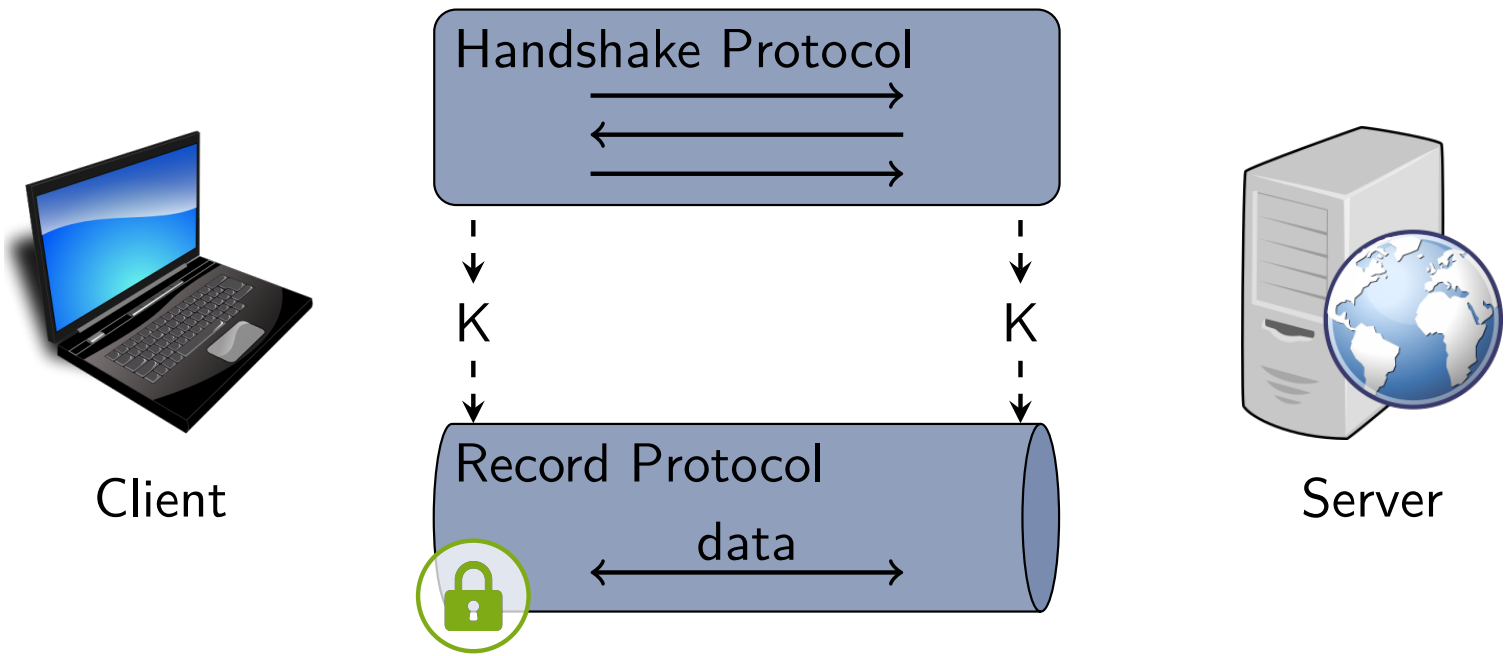
ETH zürich

Part II

TLS 1.3
Design & Security Analyses

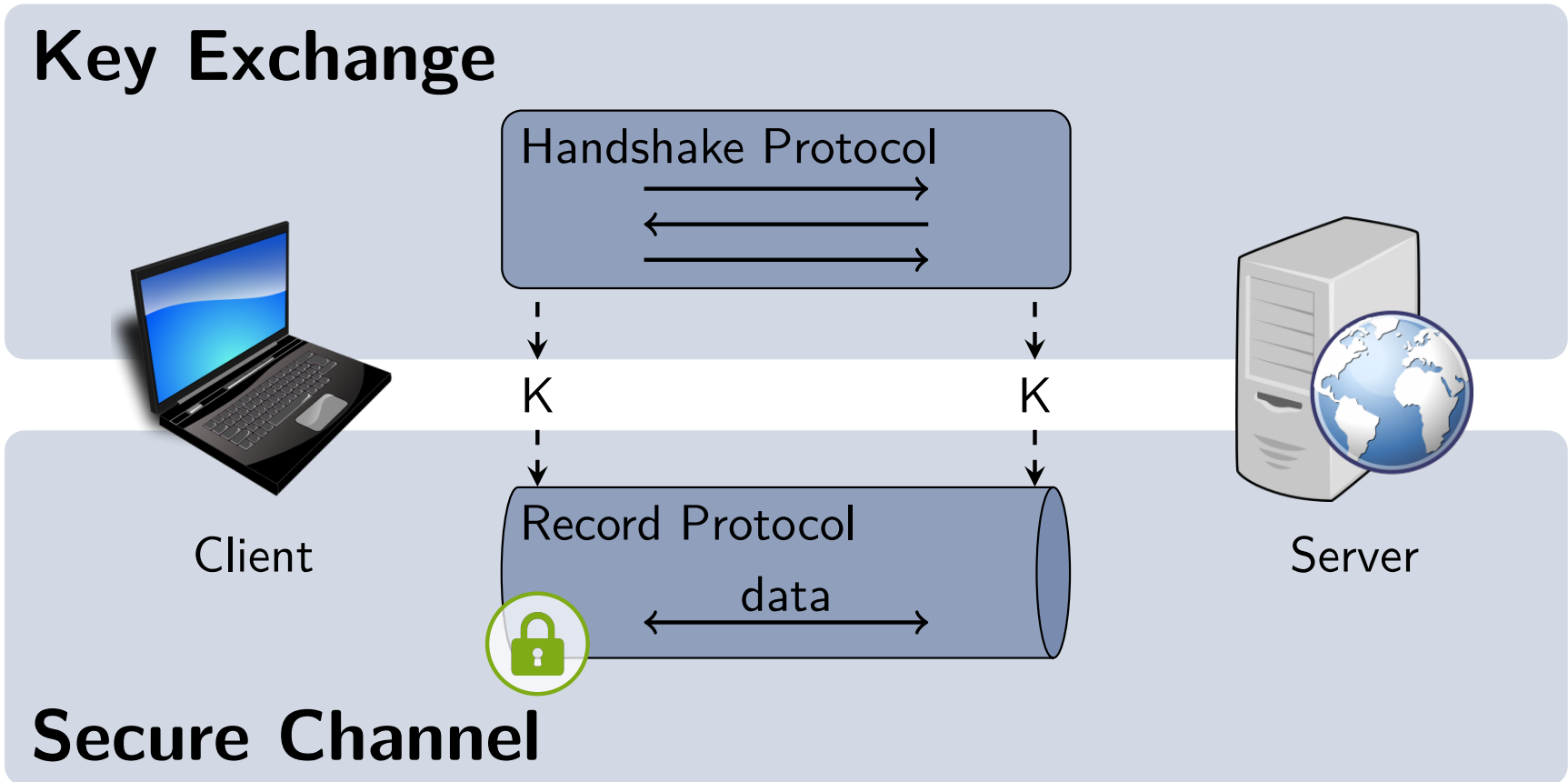
The Cryptographic Core

ETH zürich



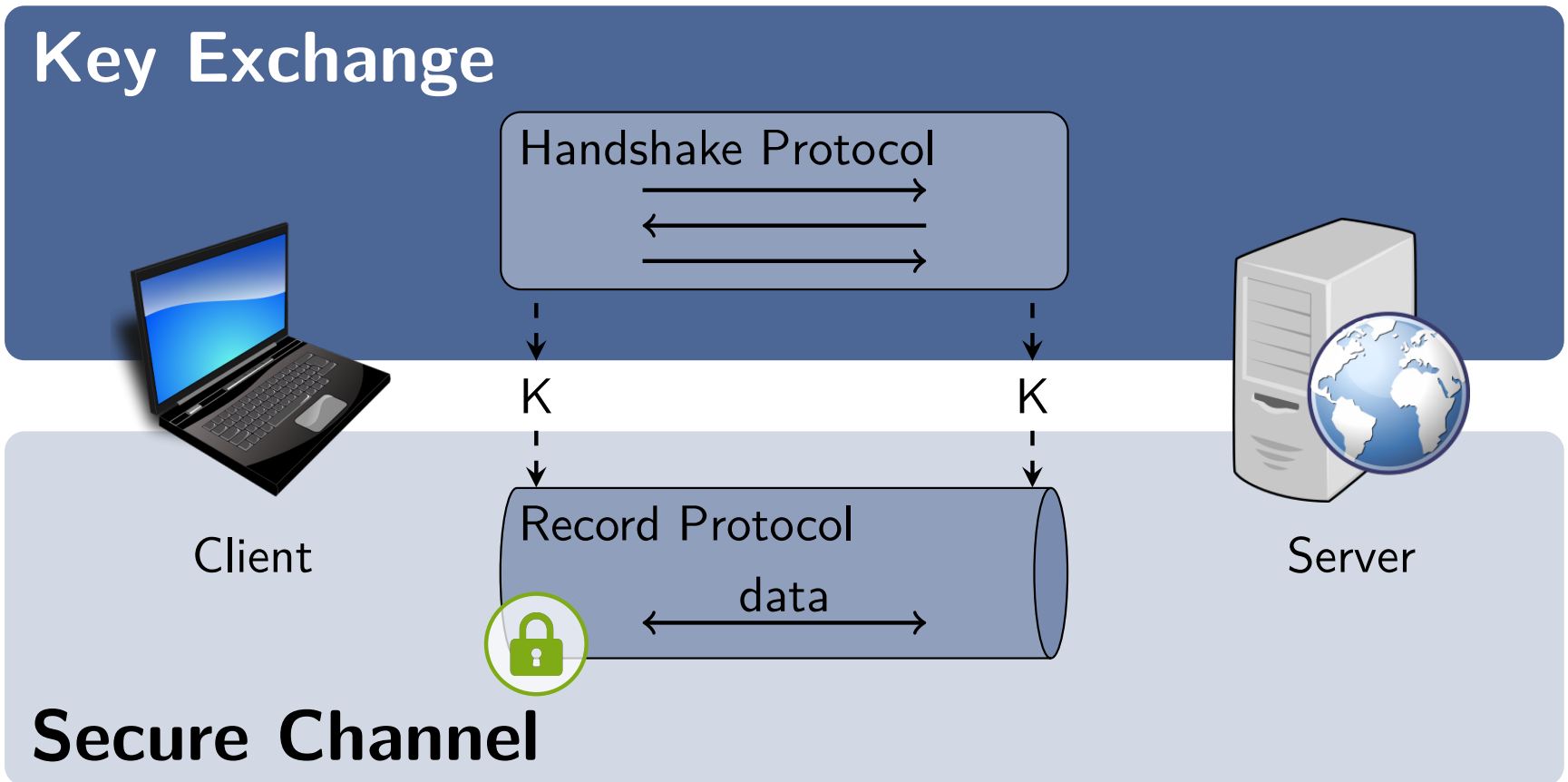
The Cryptographic Core

ETH zürich



The Cryptographic Core

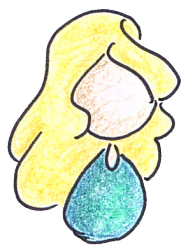
ETH zürich



Key Exchange à la Diffie–Hellman

Diffie, Hellman: New directions in cryptography. 1976 [DH76]
— see PKE Lectures

ETH zürich

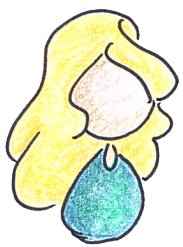


Key Exchange à la Diffie–Hellman

Diffie, Hellman: New directions in cryptography. 1976 [DH76]
— see PKE Lectures

ETH zürich

knows x



g^x

g^y

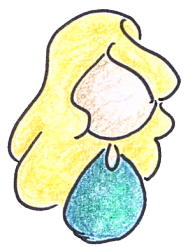
knows y

Key Exchange à la Diffie–Hellman

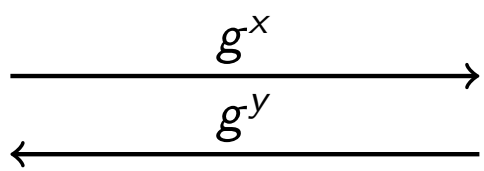
Diffie, Hellman: New directions in cryptography. 1976 [DH76]
— see PKE Lectures

ETH zürich

knows x



knows y



$g^{xy} = (g^y)^x$

$K = \mathrm{KDF}(g^{xy})$

KDF: Key Derivation Function

$g^{xy} = (g^x)^y$

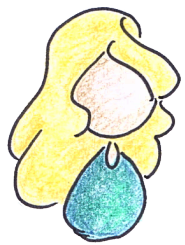
$K = \mathrm{KDF}(g^{xy})$

Key Exchange à la Diffie–Hellman

Diffie, Hellman: New directions in cryptography. 1976 [DH76]
— see PKE Lectures

ETH zürich

knows x



g^x

g^y

knows y

$g^{xy} = (g^y)^x$

$K = \mathrm{KDF}(g^{xy})$

KDF: Key Derivation Function

$g^{xy} = (g^x)^y$

$K = \mathrm{KDF}(g^{xy})$

- key secrecy: given only g^x and g^y, key K remains secret (indist. from )

Key Exchange à la Diffie–Hellman

Diffie, Hellman: New directions in cryptography. 1976 [DH76]
— see PKE Lectures

ETH zürich

DDH: g^a g^b g^{ab} / g^a g^b g^r ?

knows x knows y

g^x

g^y

$g^{xy} = (g^y)^x$
$K = \mathrm{KDF}(g^{xy})$

KDF: Key Derivation Function

$g^{xy} = (g^x)^y$
$K = \mathrm{KDF}(g^{xy})$

- key secrecy: given only g^x and g^y, key K remains secret (indist. from )

Key Exchange à la Diffie–Hellman

Diffie, Hellman: New directions in cryptography. 1976 [DH76]
— see PKE Lectures

ETH zürich

DDH: g^a g^b g^{ab} / g^a g^b g^r ?

knows x

$g^{xy} = (g^y)^x$

$K = \mathrm{KDF}(g^{xy})$

g^a

g^b

KDF: Key Derivation Function

knows y

$g^{xy} = (g^x)^y$

$K = \mathrm{KDF}(g^{xy})$

- key secrecy: given only g^x and g^y, key K remains secret (indist. from )

Key Exchange à la Diffie–Hellman

Diffie, Hellman: New directions in cryptography. 1976 [DH76]
— see PKE Lectures

ETH zürich

DDH: g^a g^b g^{ab} / g^a g^b g^r ?

knows x

knows y

g^a

g^b

$g^{xy} = g^?$

$K = \mathrm{KDF}(g^{xy})$

KDF: Key Derivation Function

$g^{xy} = g^?$

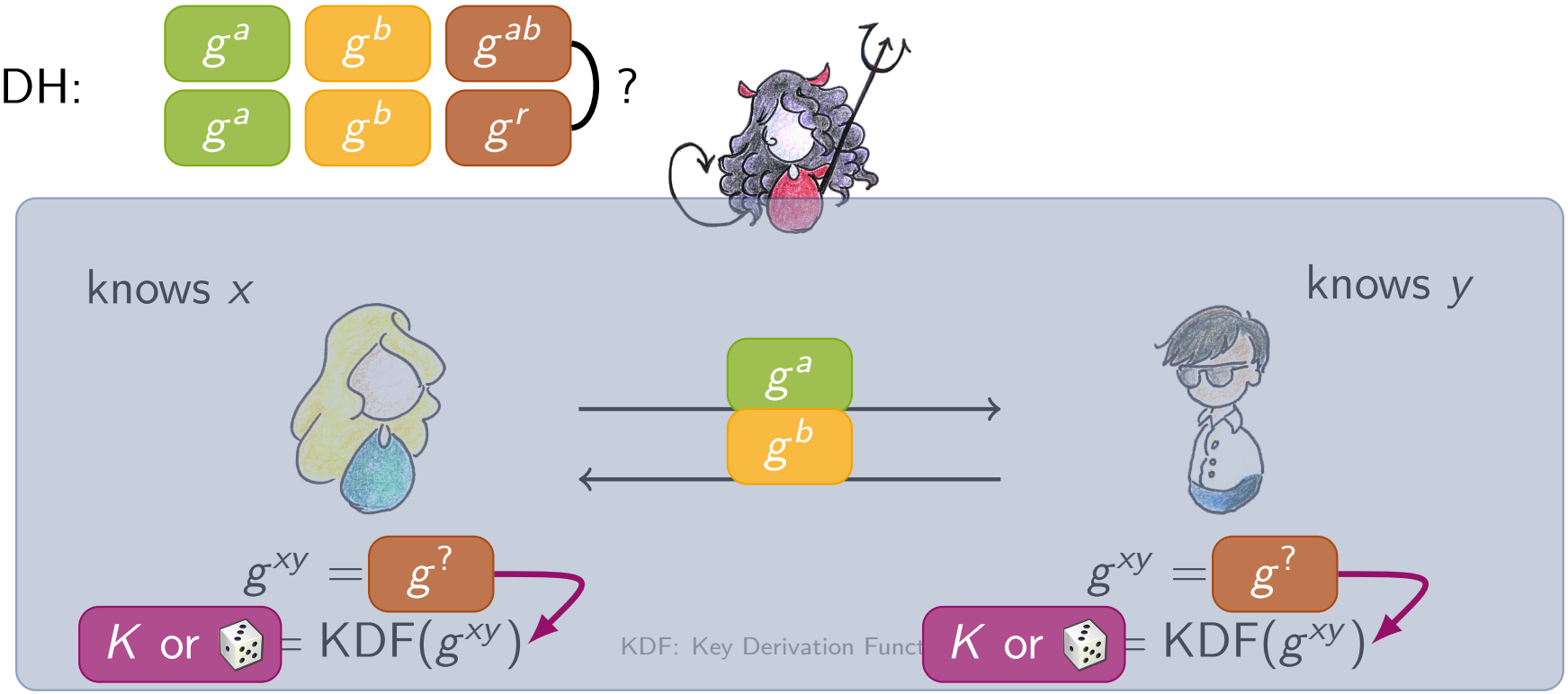
$K = \mathrm{KDF}(g^{xy})$

- key secrecy: given only g^x and g^y, key K remains secret (indist. from )

Key Exchange à la Diffie–Hellman

Diffie, Hellman: New directions in cryptography. 1976 [DH76]
— see PKE Lectures

ETH zürich

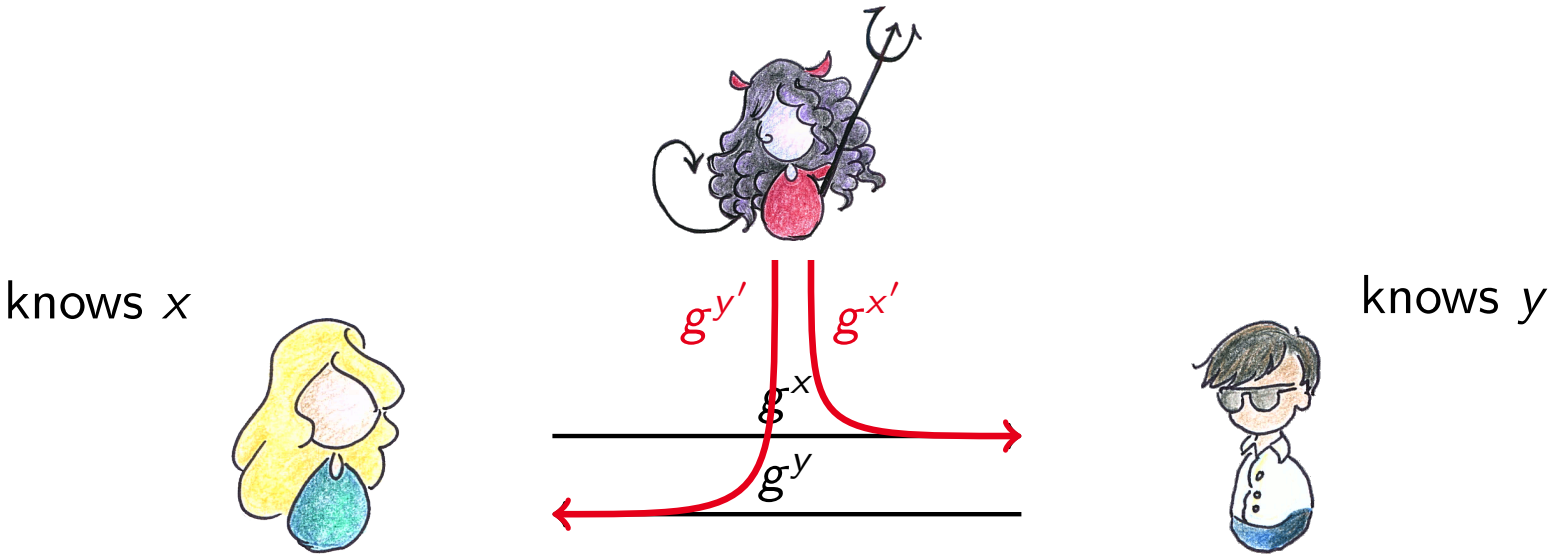


- key secrecy: given only g^x and g^y, key K remains secret (indist. from)

Key Exchange à la Diffie–Hellman

Diffie, Hellman: New directions in cryptography. 1976 [DH76]
— see PKE Lectures

ETH zürich



$g^{xy} = (g^y)^x$
$K = \mathrm{KDF}(g^{xy})$
$K' = \mathrm{KDF}(g^{xy'})$

KDF: Key Derivation Function

$g^{xy} = (g^x)^y$
$K = \mathrm{KDF}(g^{xy})$
$K'' = \mathrm{KDF}(g^{x'y})$

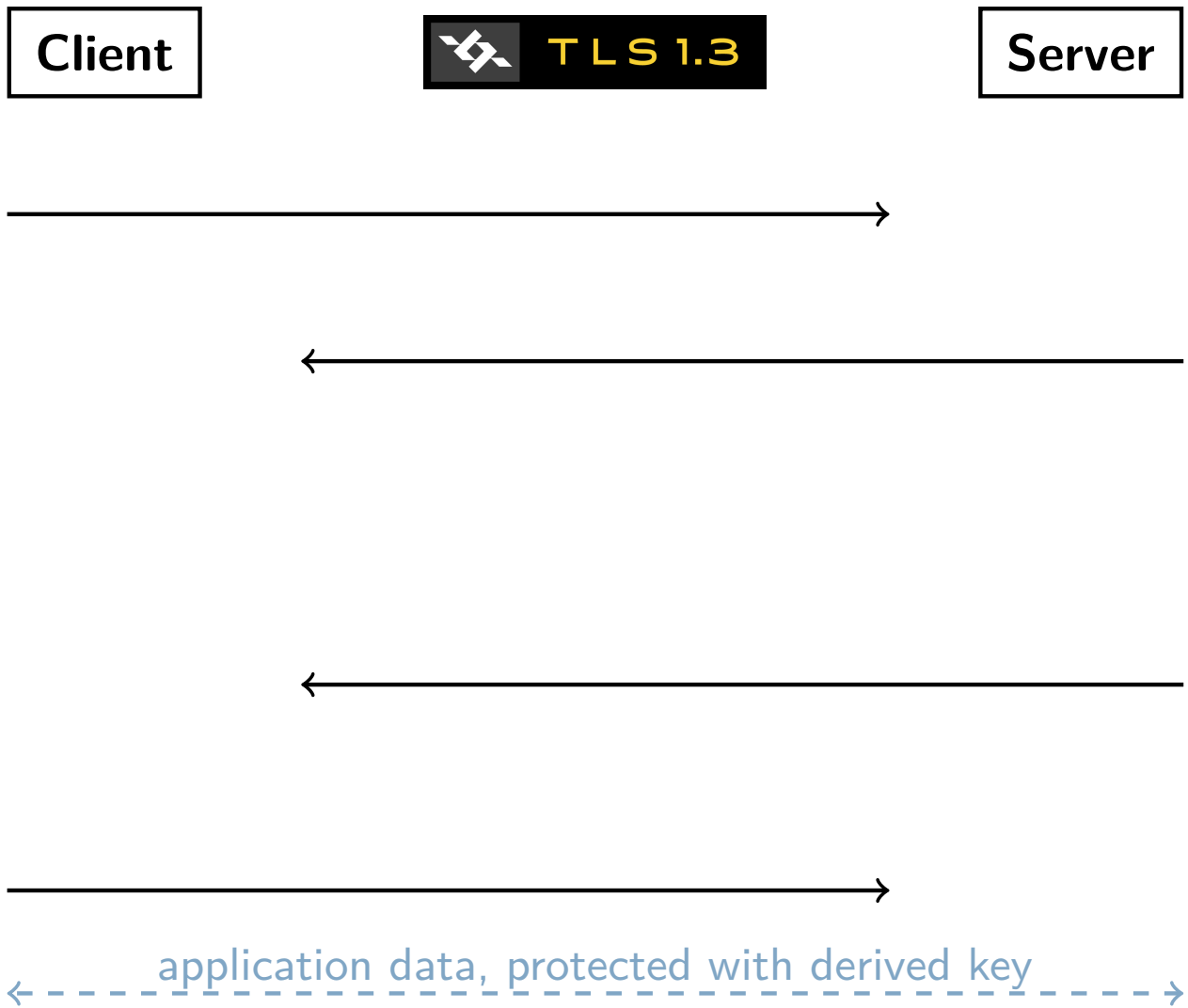
- key secrecy: given only g^x and g^y, key K remains secret (indist. from )
- just one of many building blocks (no security against MitM, . . .)

The TLS 1.3 Handshake

Full (EC)DHE Mode

ETH zürich

(simplified)

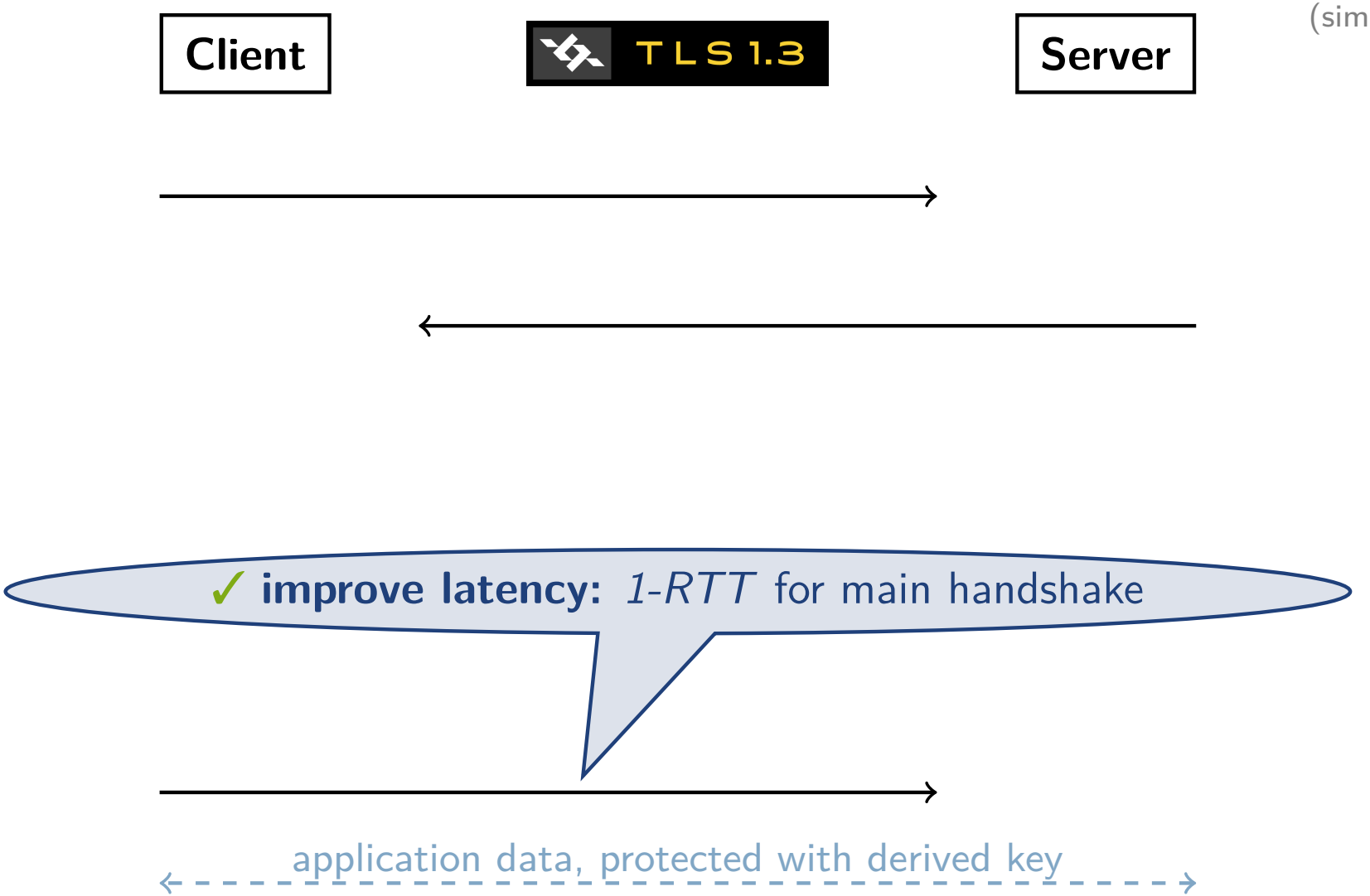


The TLS 1.3 Handshake

Full (EC)DHE Mode

ETH zürich

(simplified)

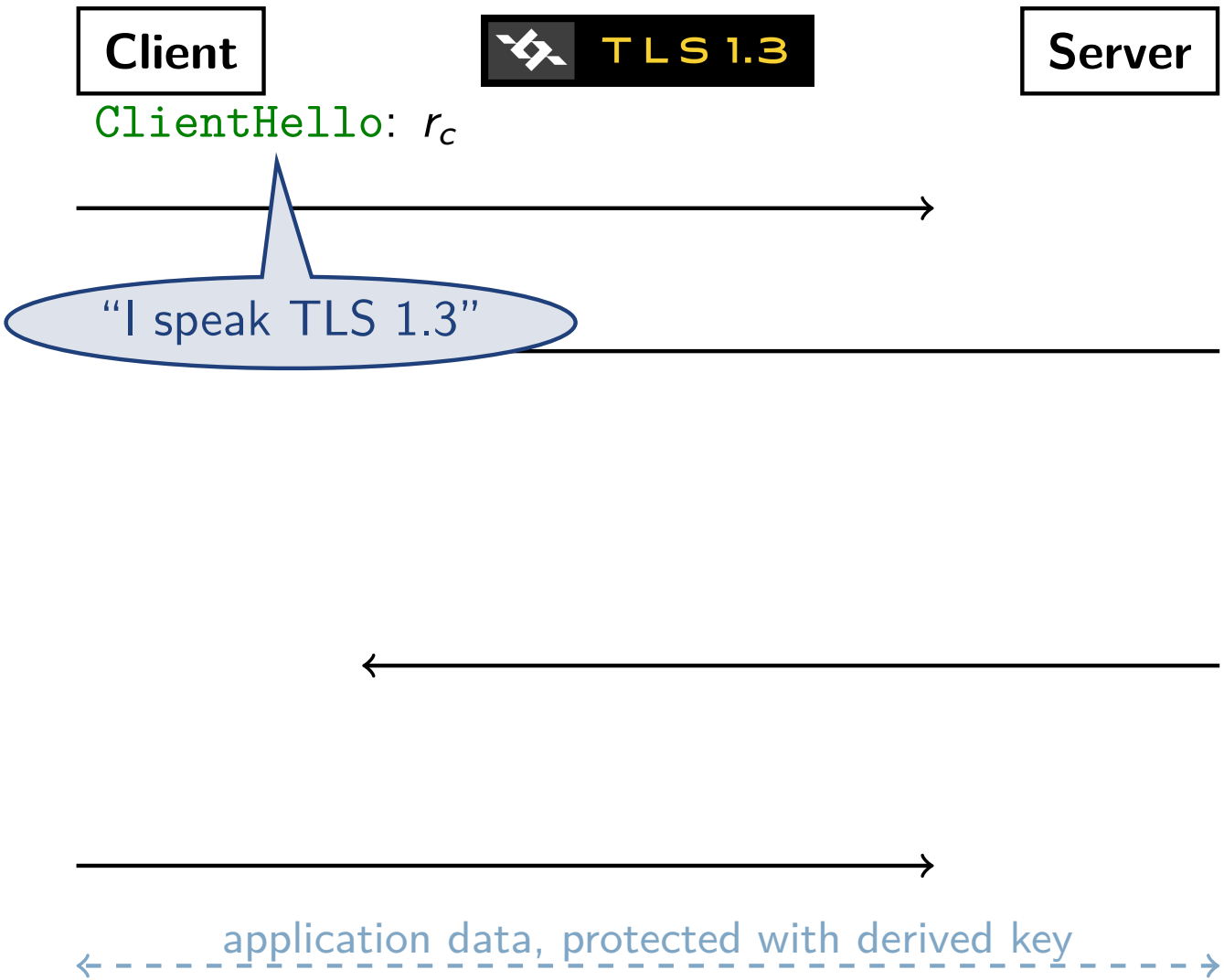


The TLS 1.3 Handshake

Full (EC)DHE Mode

ETH zürich

(simplified)

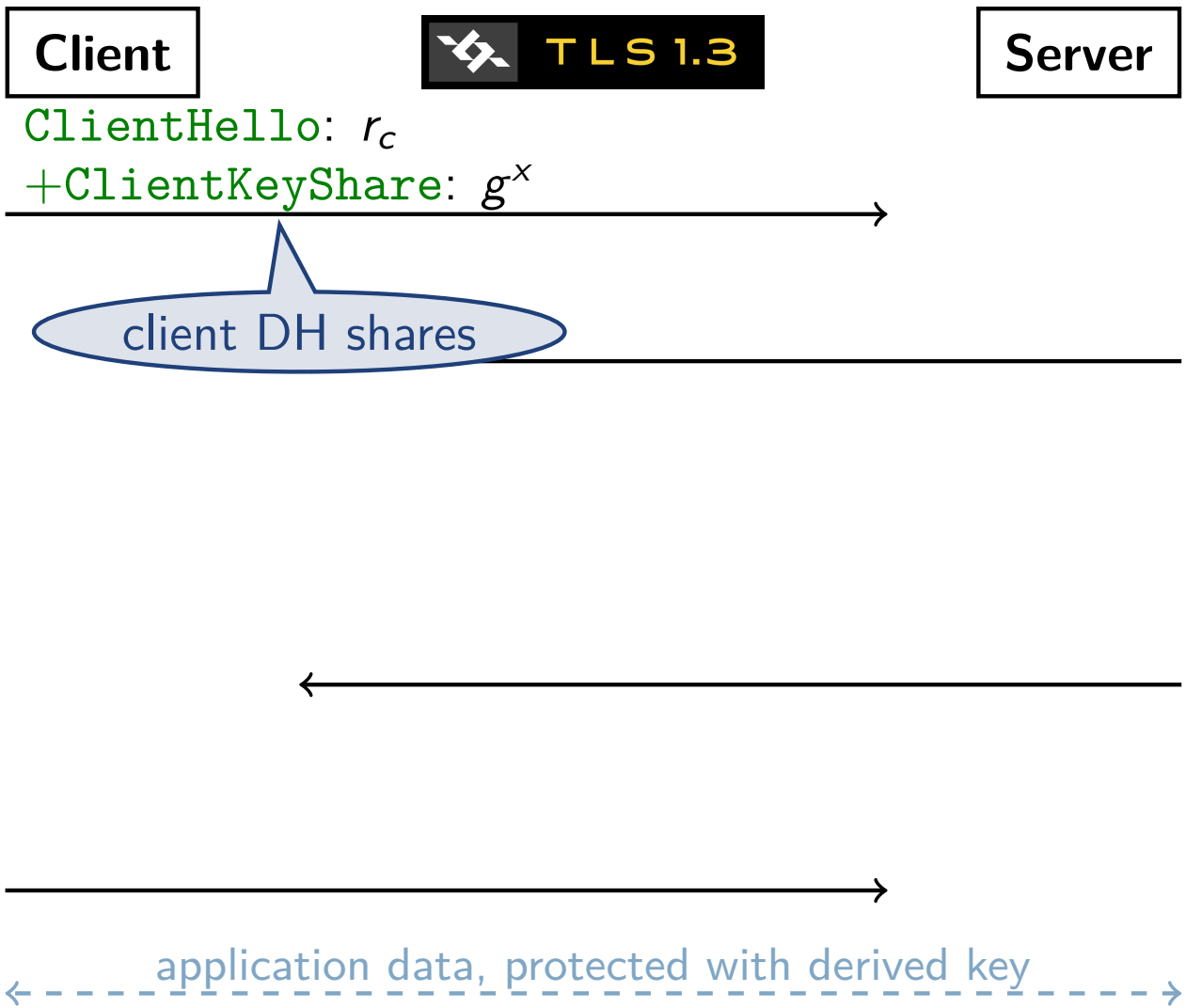


The TLS 1.3 Handshake

Full (EC)DHE Mode

ETH zürich

(simplified)

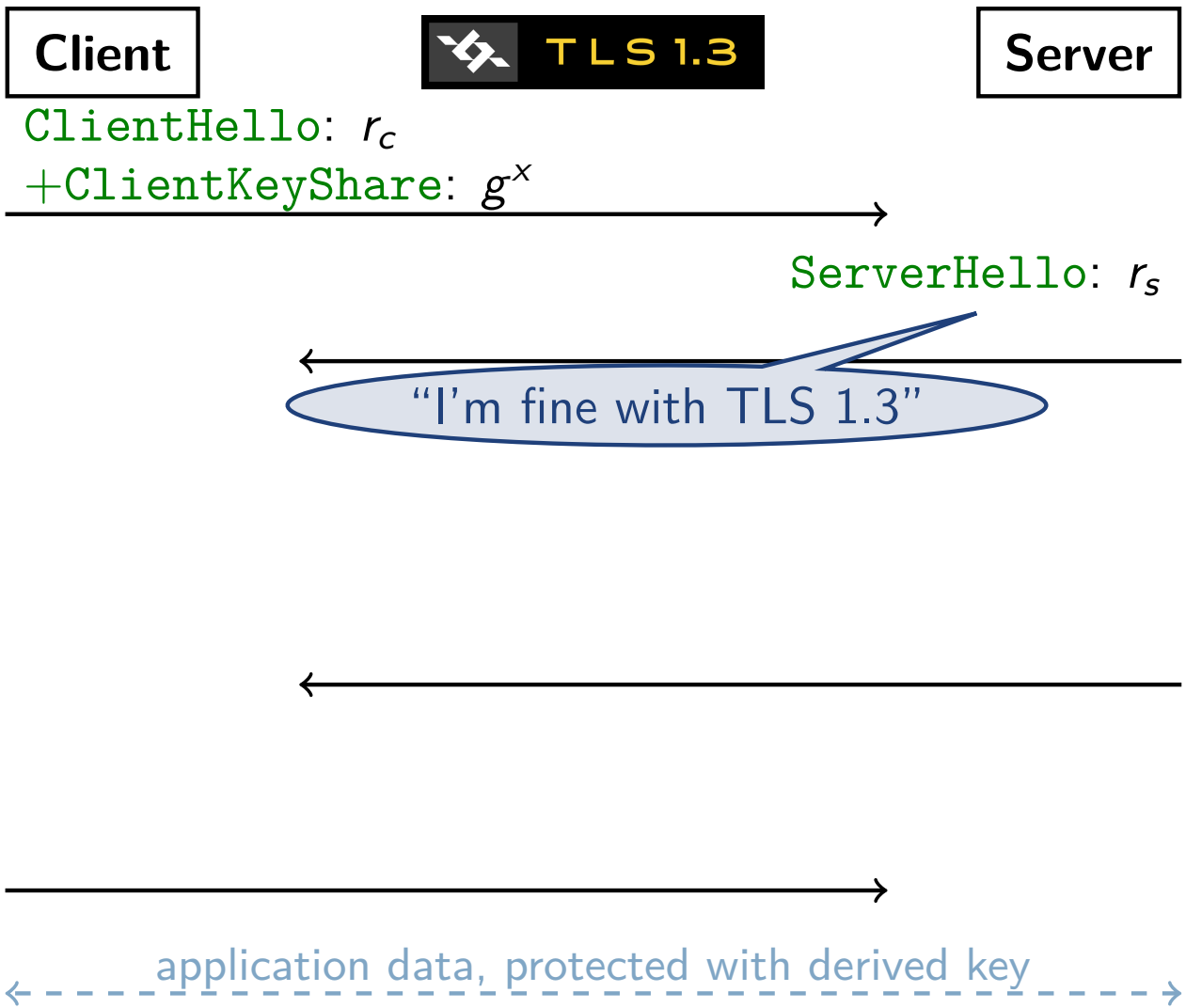


The TLS 1.3 Handshake

Full (EC)DHE Mode

ETH zürich

(simplified)

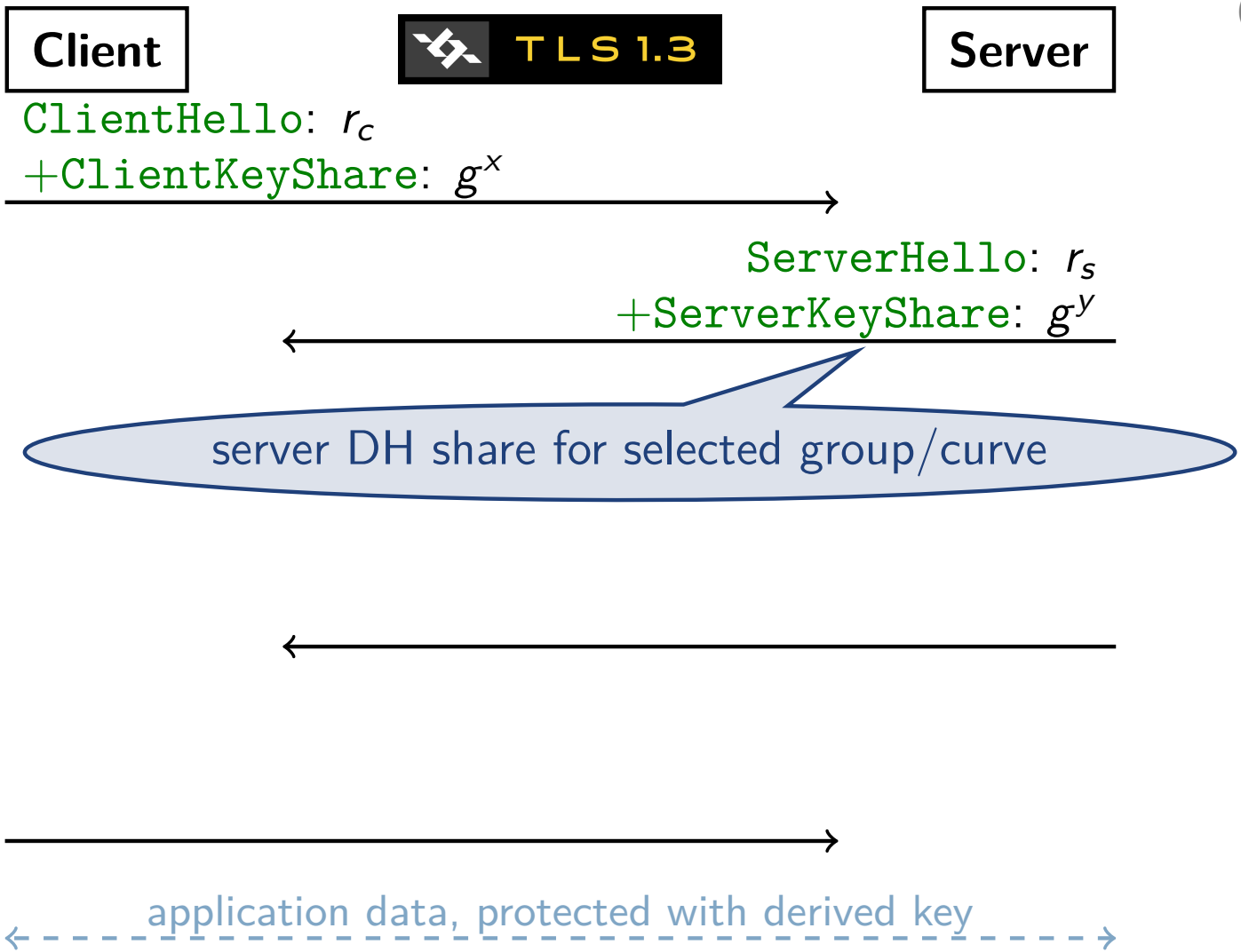


The TLS 1.3 Handshake

Full (EC)DHE Mode

ETH zürich

(simplified)

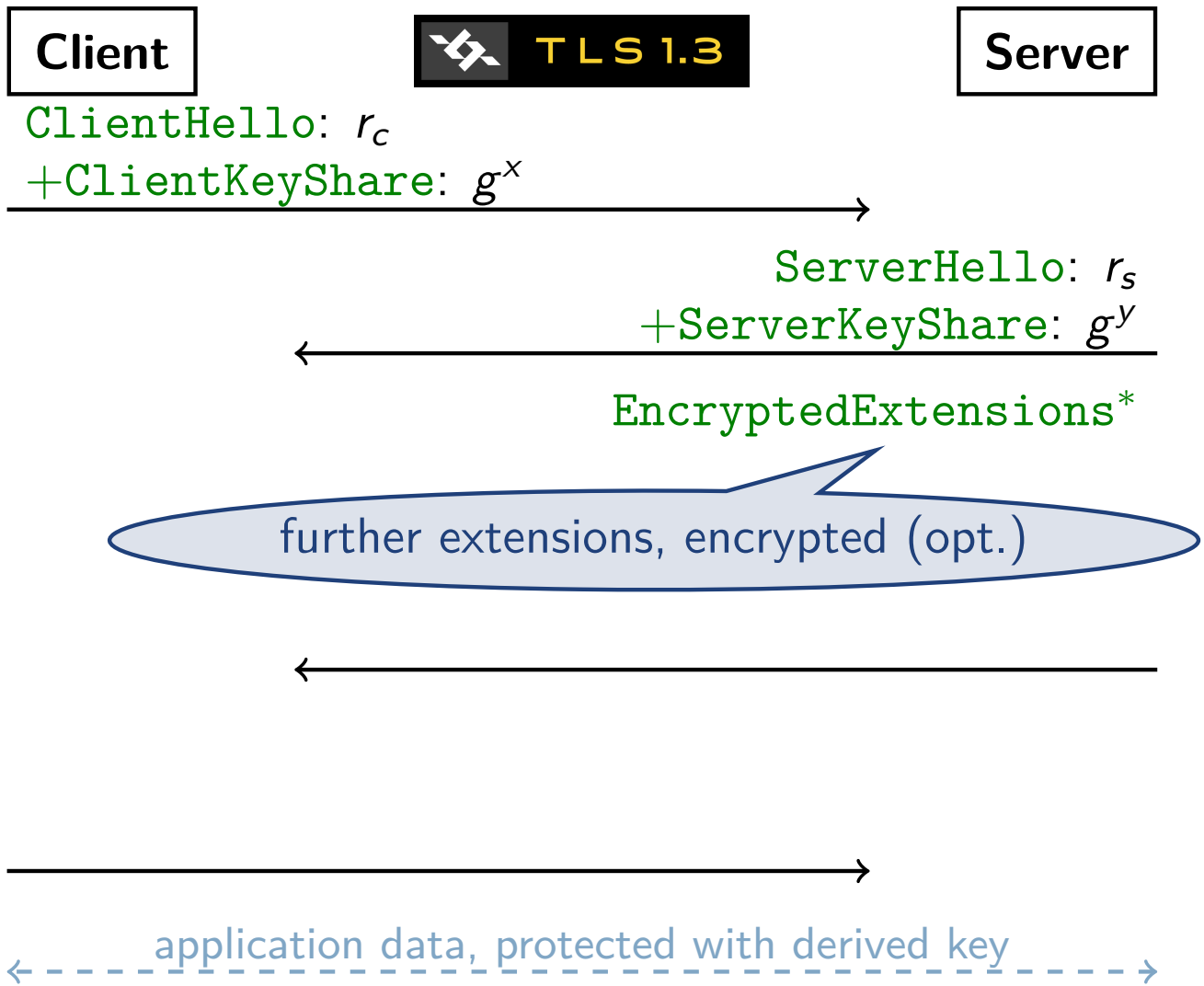


The TLS 1.3 Handshake

Full (EC)DHE Mode

ETH zürich

(simplified)

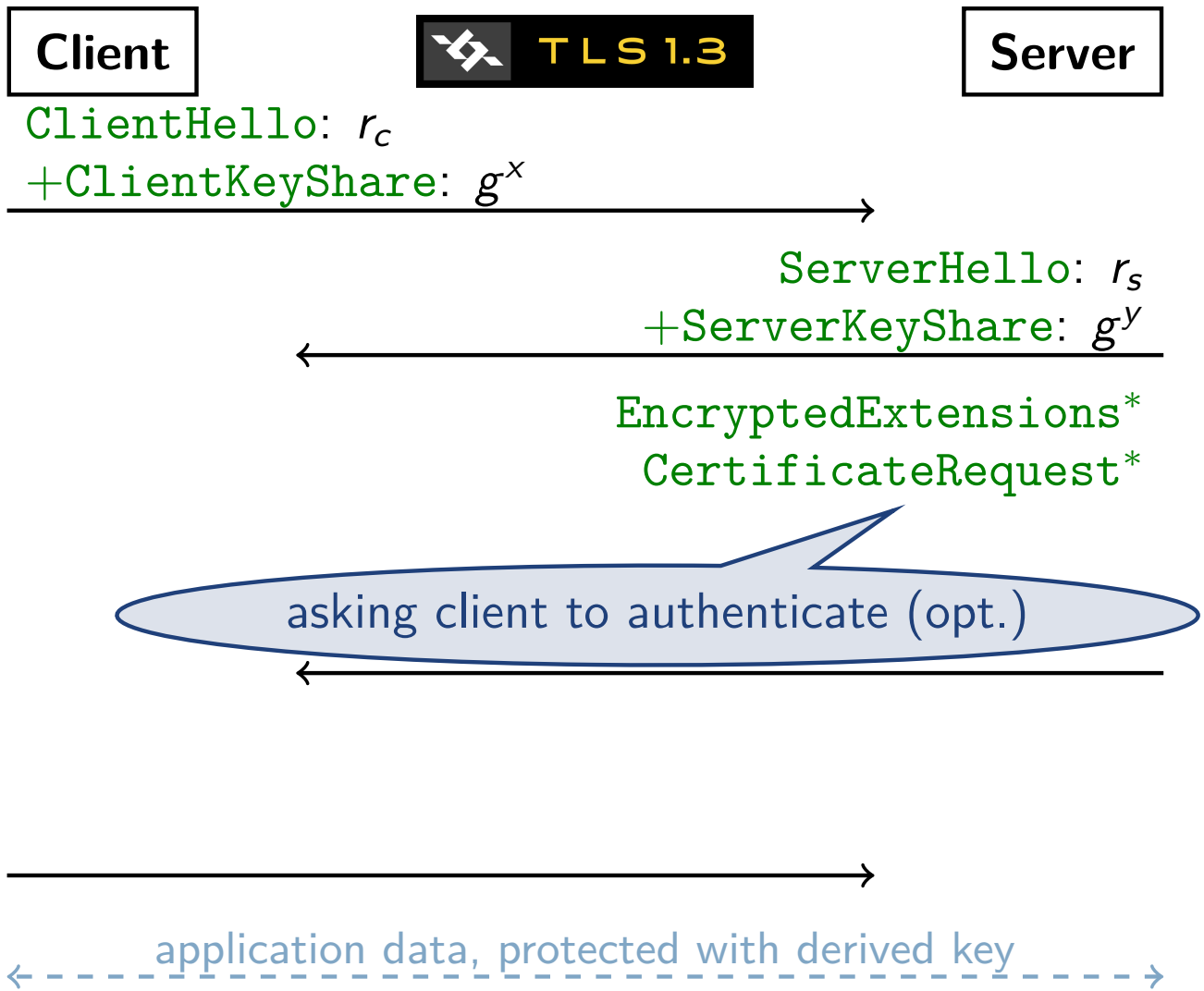


The TLS 1.3 Handshake

Full (EC)DHE Mode

ETH zürich

(simplified)

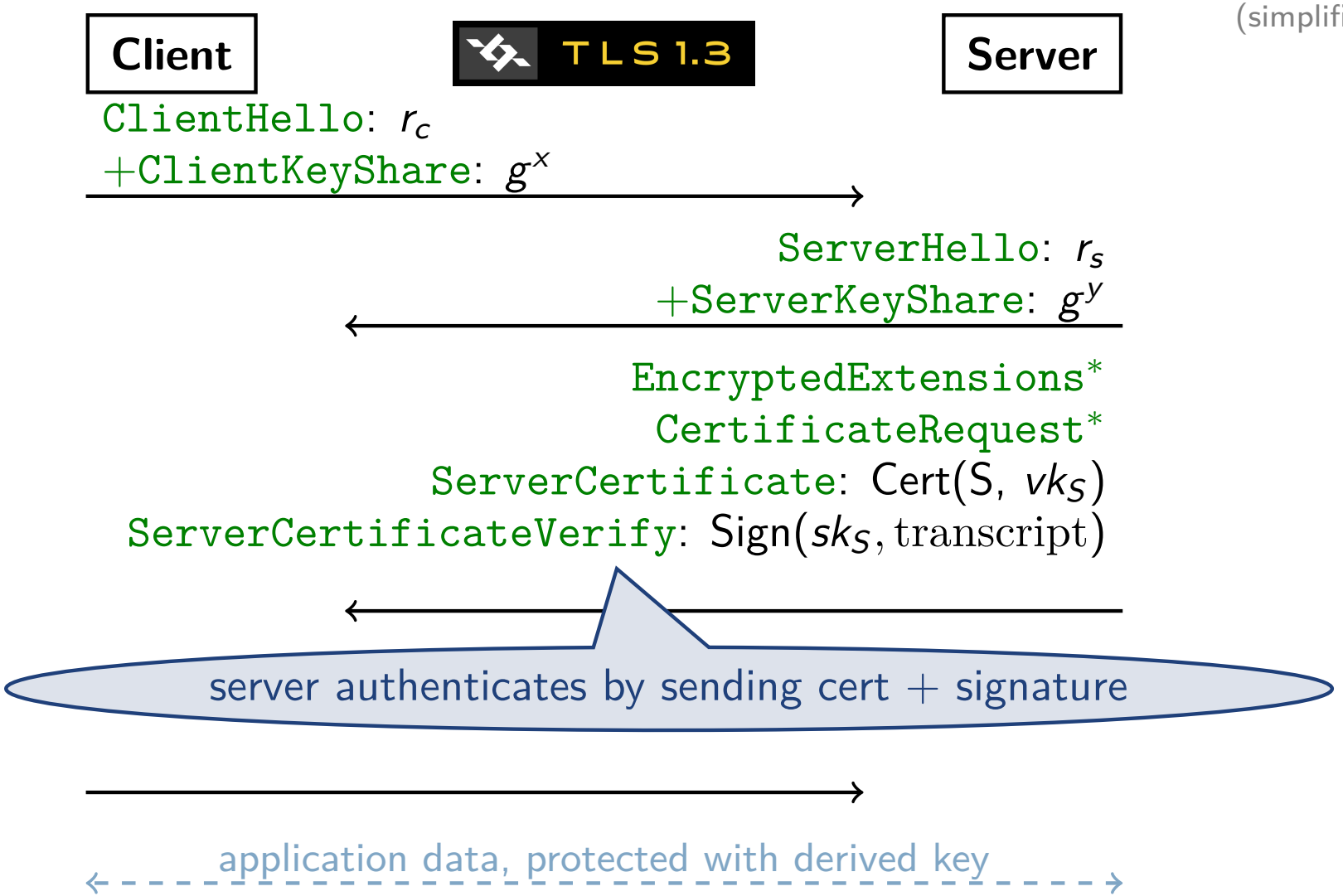


The TLS 1.3 Handshake

Full (EC)DHE Mode

ETH zürich

(simplified)

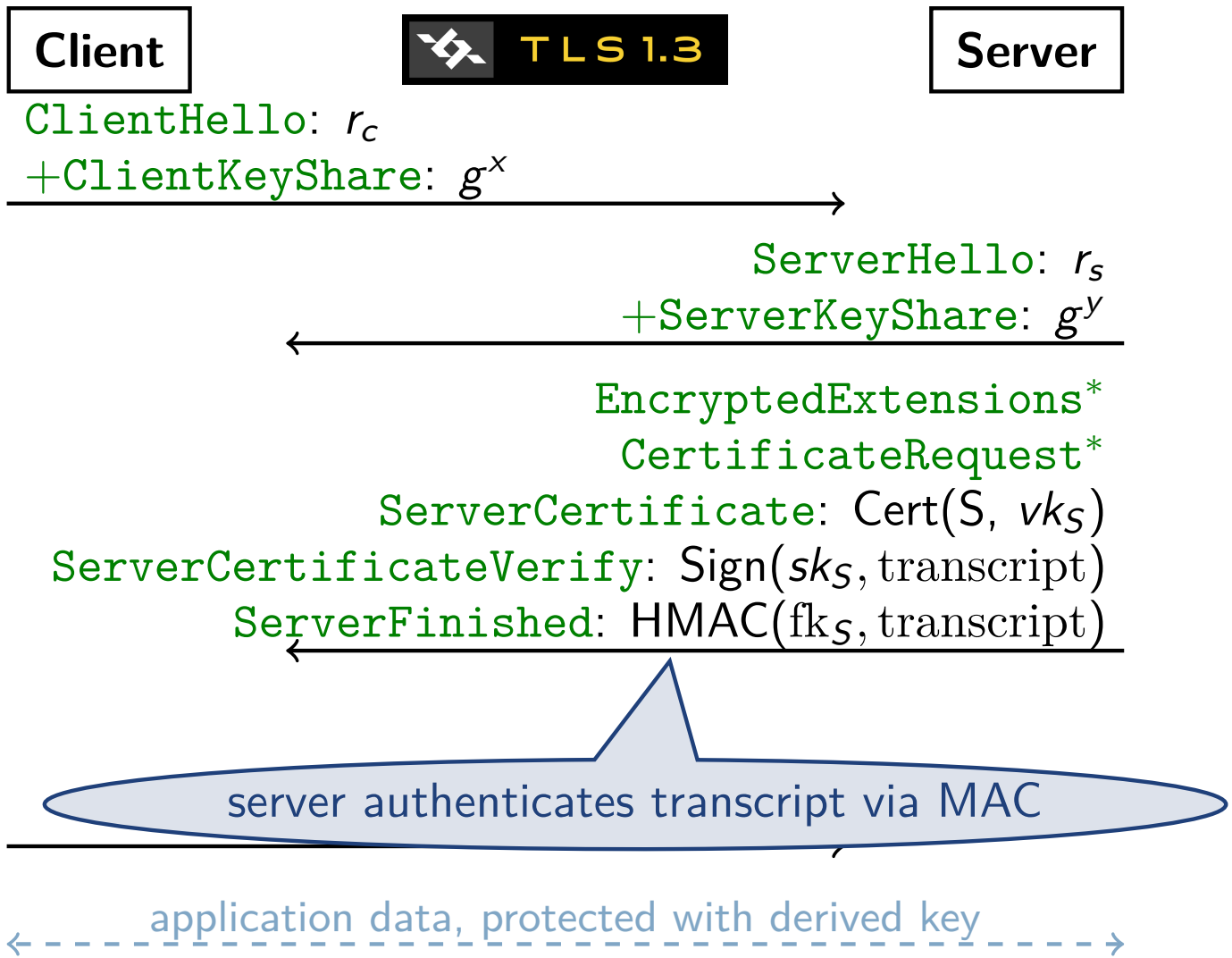


The TLS 1.3 Handshake

Full (EC)DHE Mode

ETH zürich

(simplified)

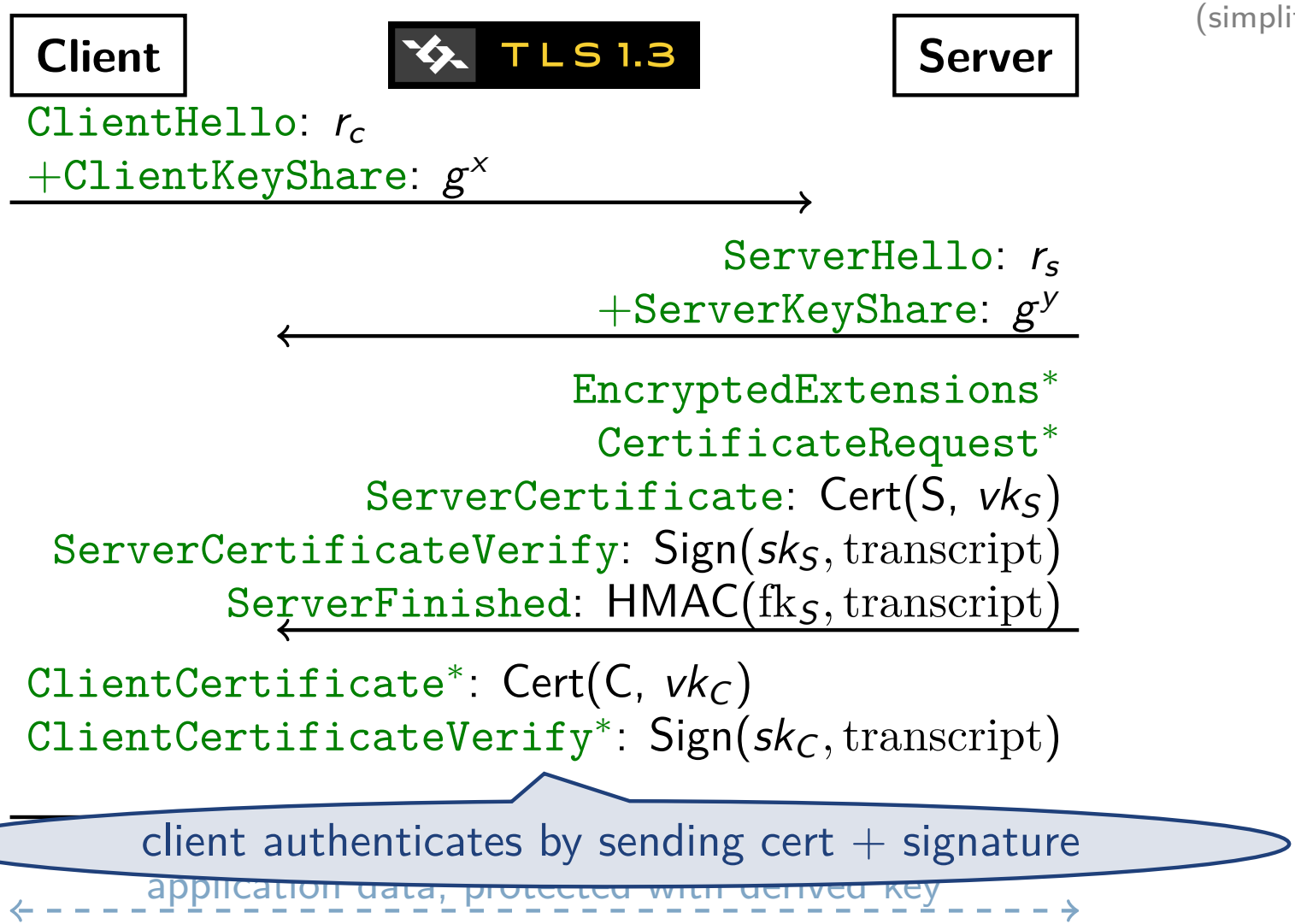


The TLS 1.3 Handshake

Full (EC)DHE Mode

ETH zürich

(simplified)

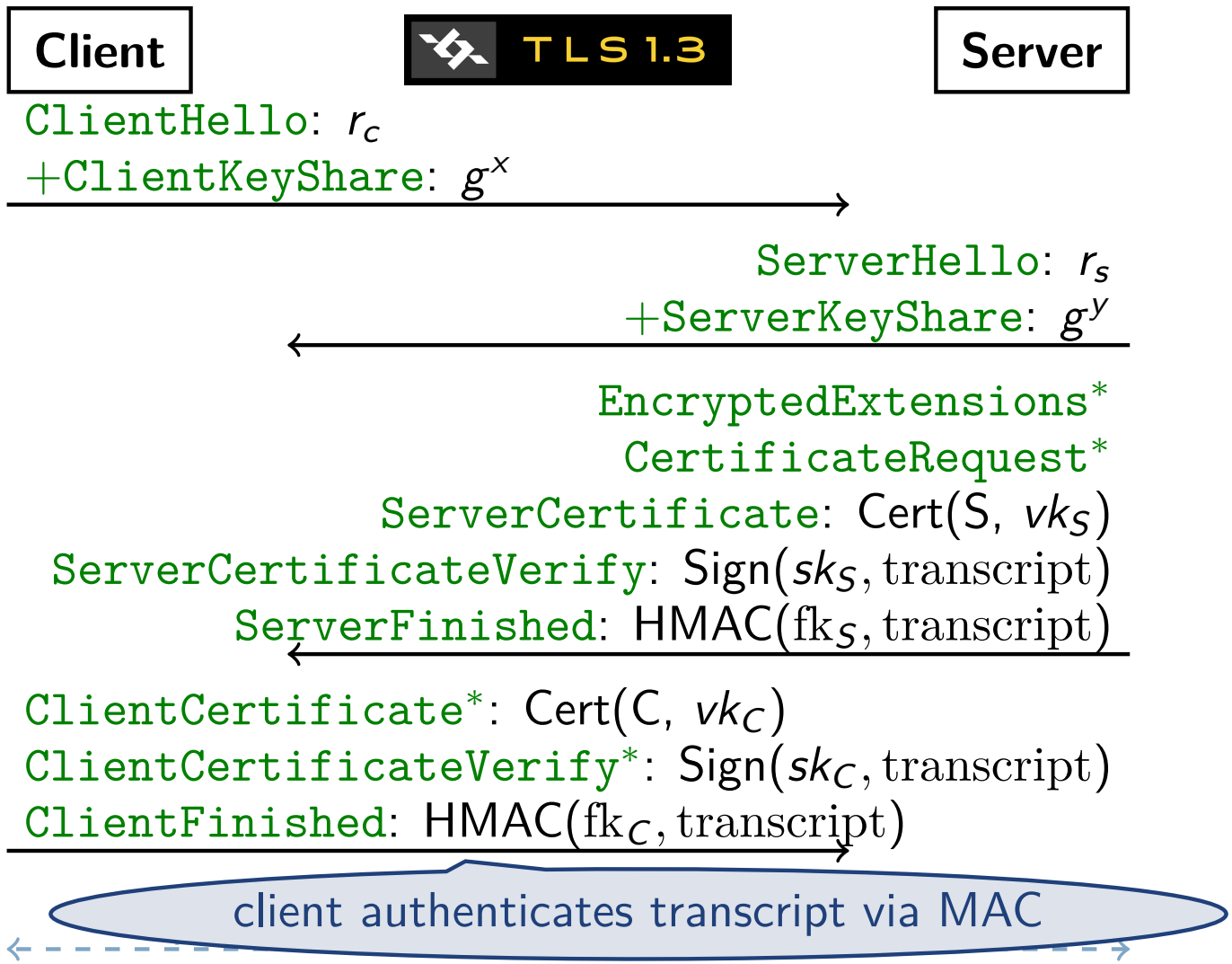


The TLS 1.3 Handshake

Full (EC)DHE Mode

ETH zürich

(simplified)

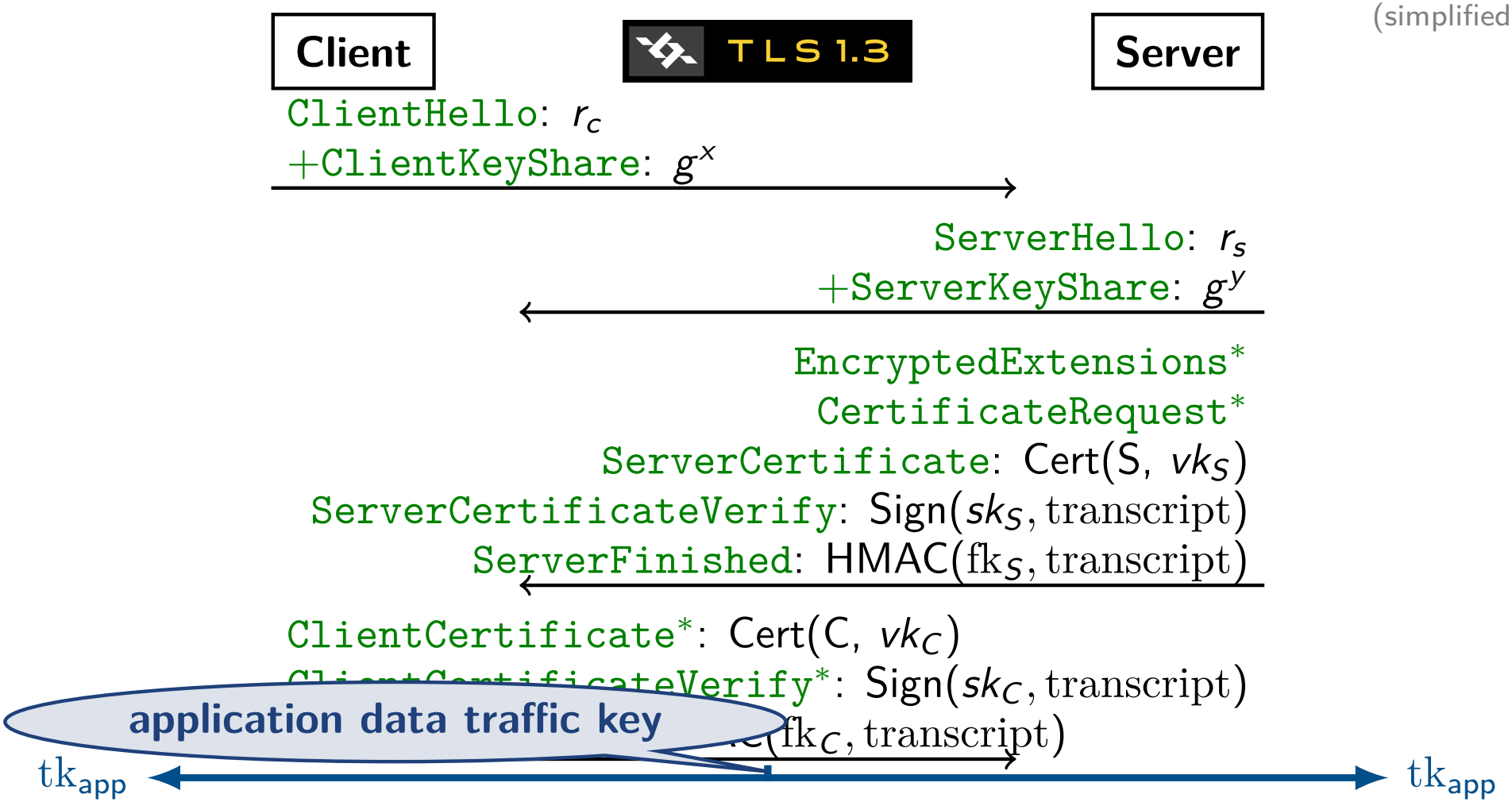


The TLS 1.3 Handshake

Full (EC)DHE Mode

ETH zürich

(simplified)

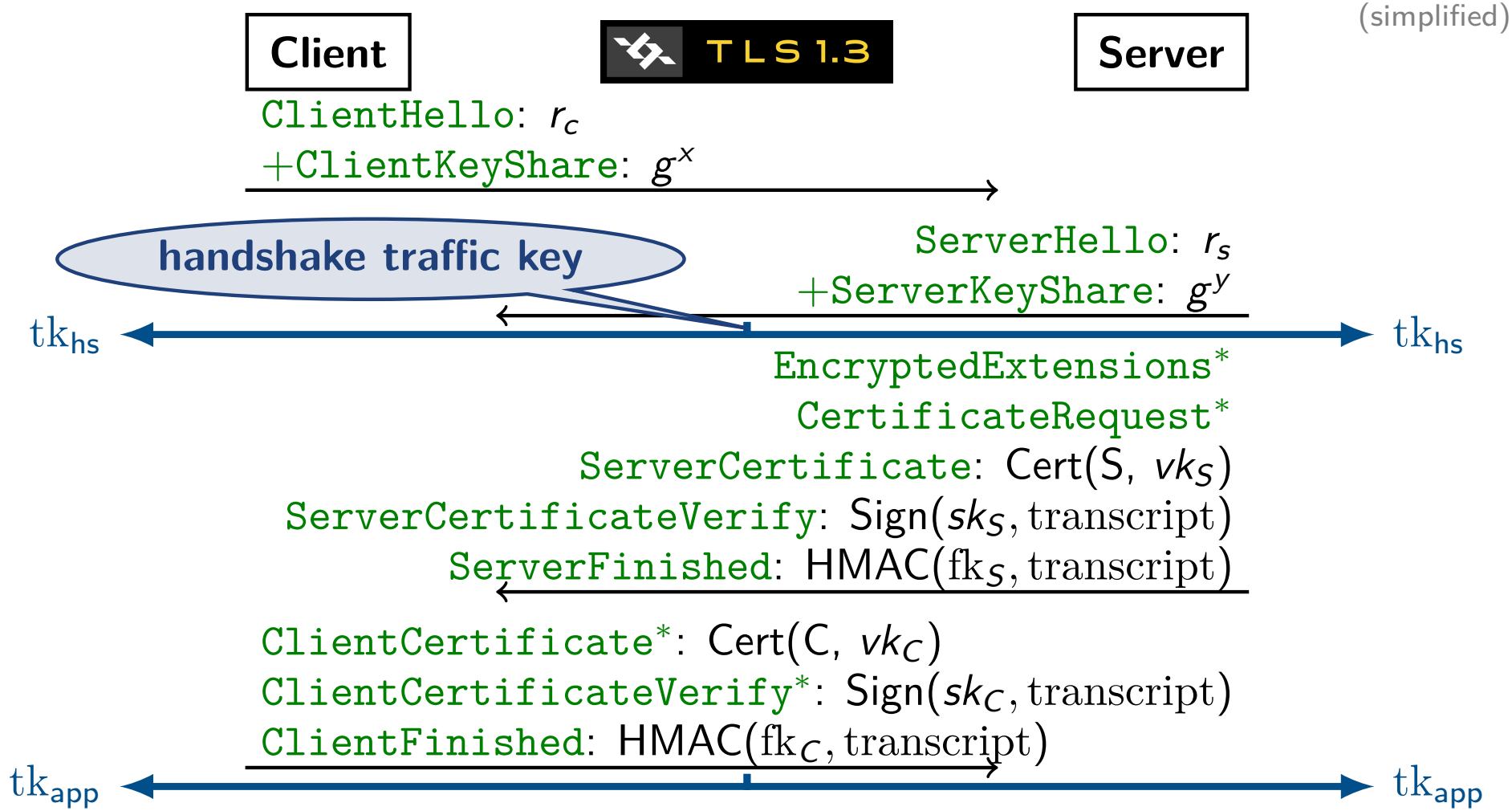


The TLS 1.3 Handshake

Full (EC)DHE Mode

ETH zürich

(simplified)

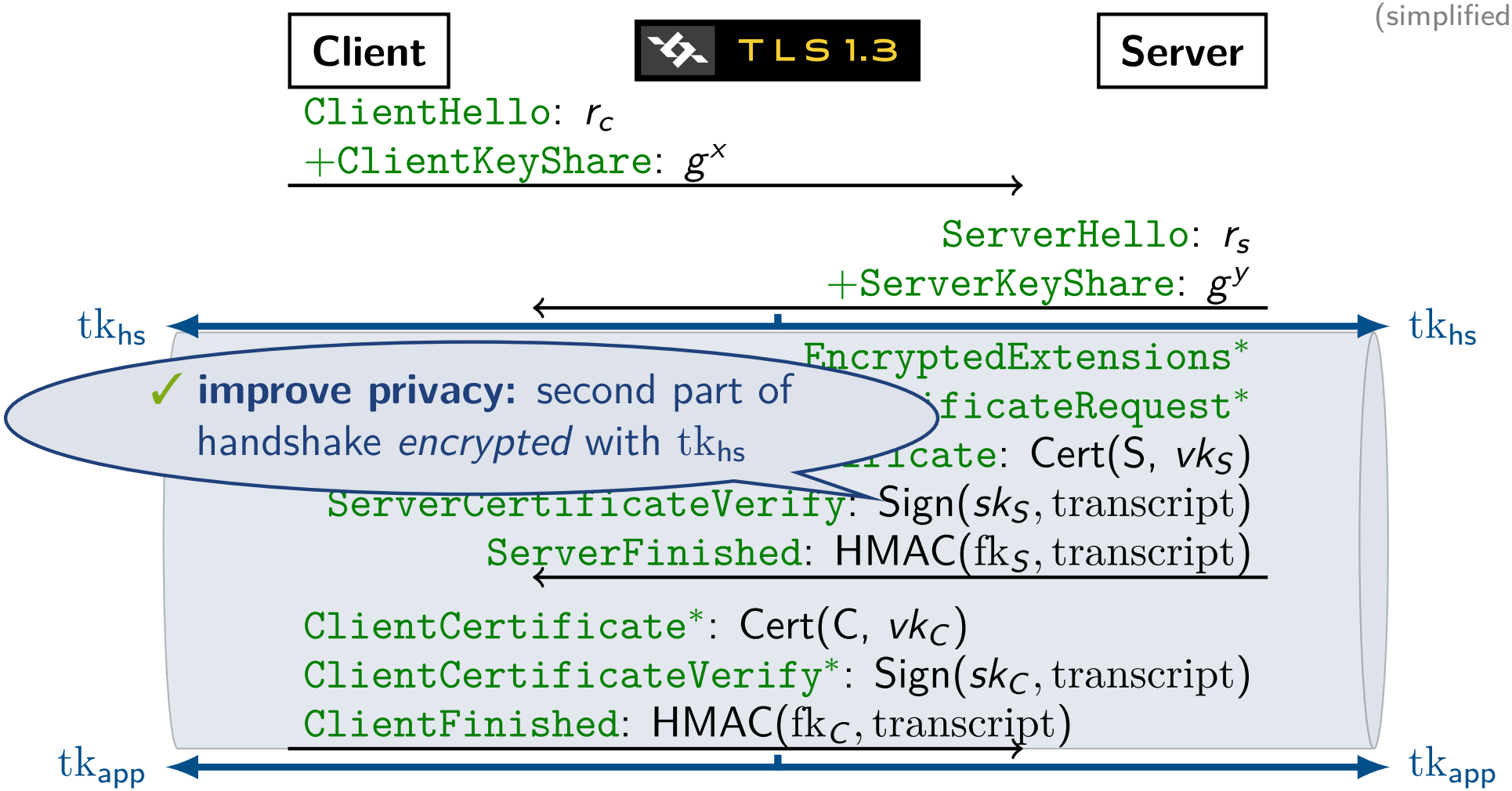


The TLS 1.3 Handshake

Full (EC)DHE Mode

ETH zürich

(simplified)

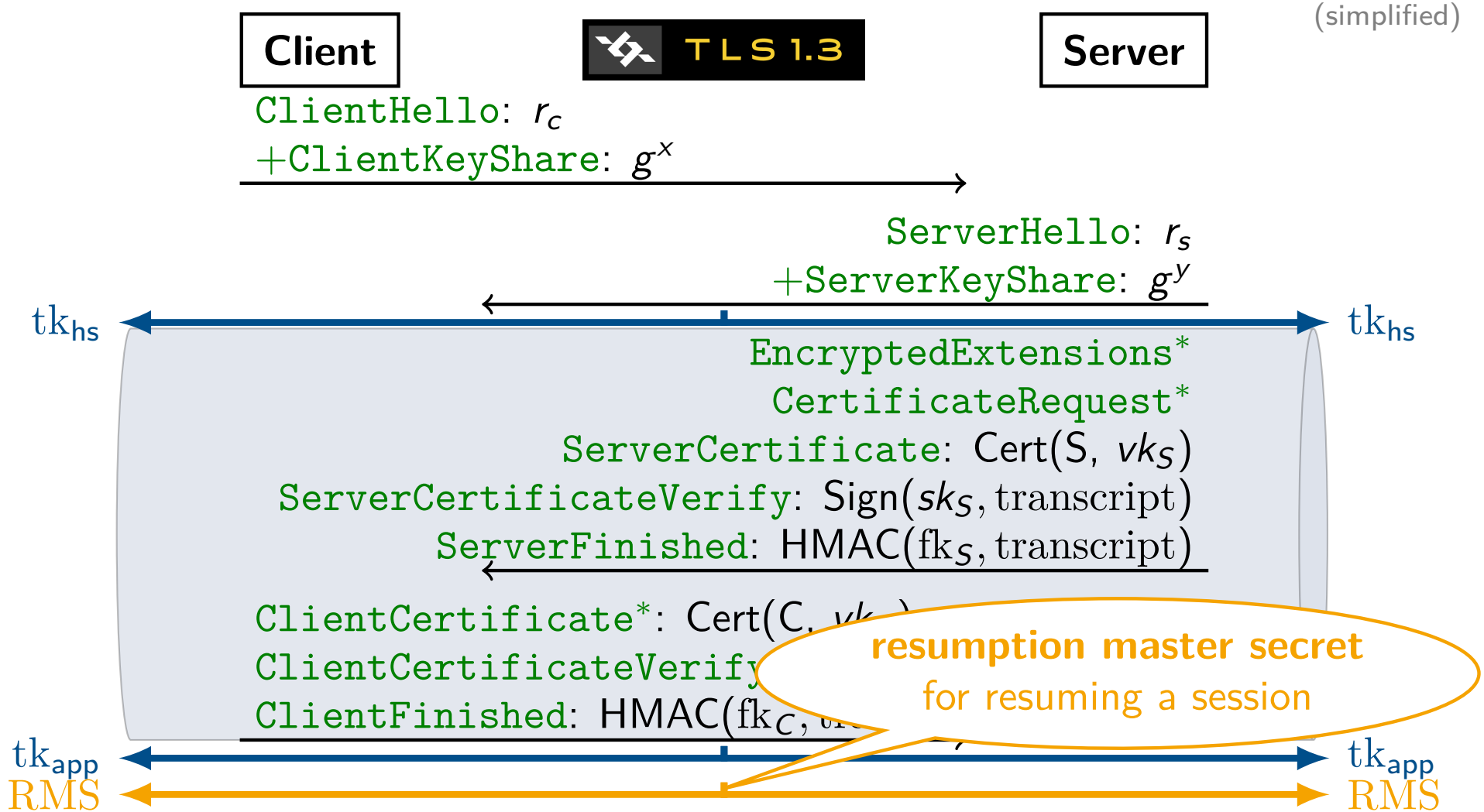


The TLS 1.3 Handshake

Full (EC)DHE Mode

ETH zürich

(simplified)

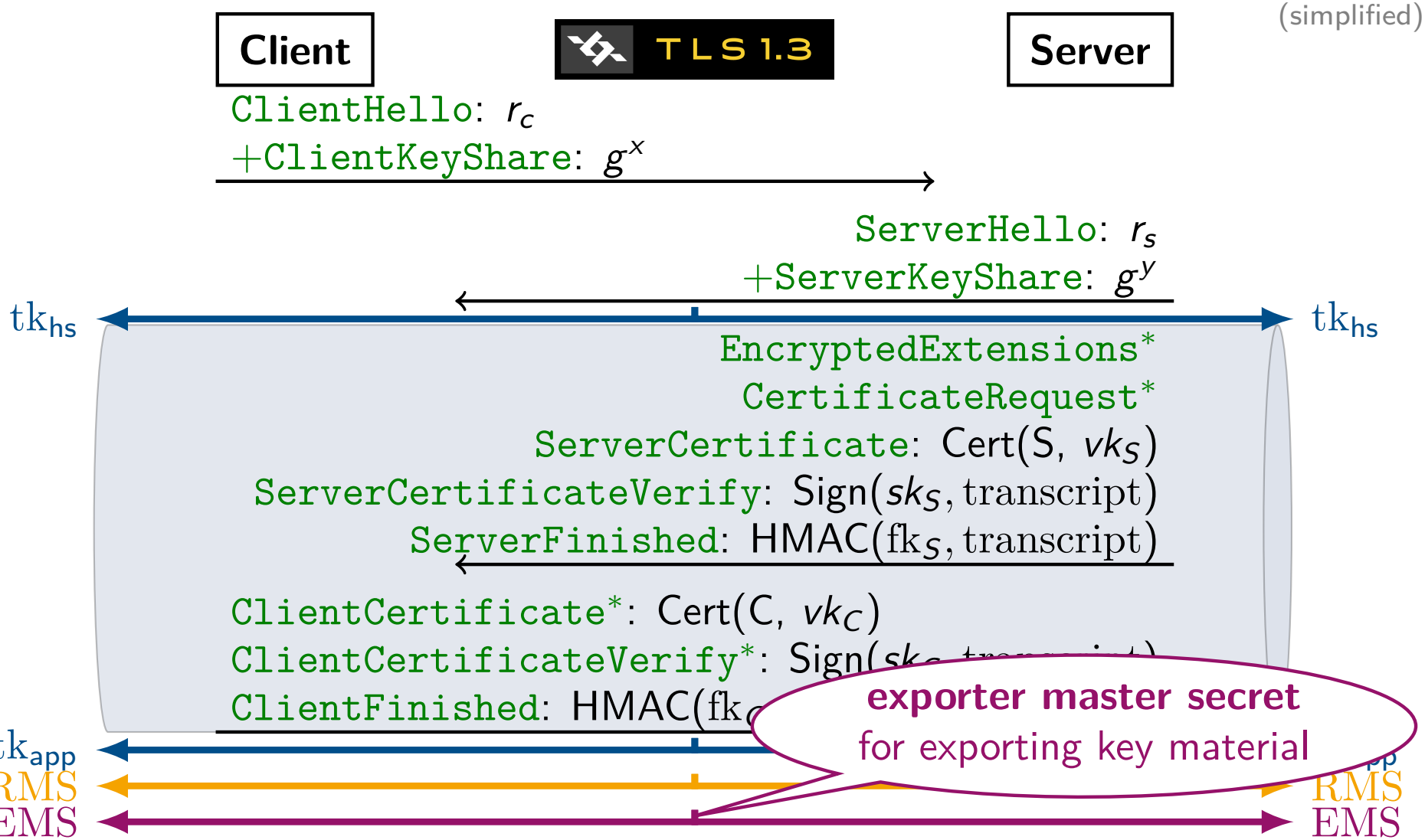


The TLS 1.3 Handshake

Full (EC)DHE Mode

ETH zürich

(simplified)

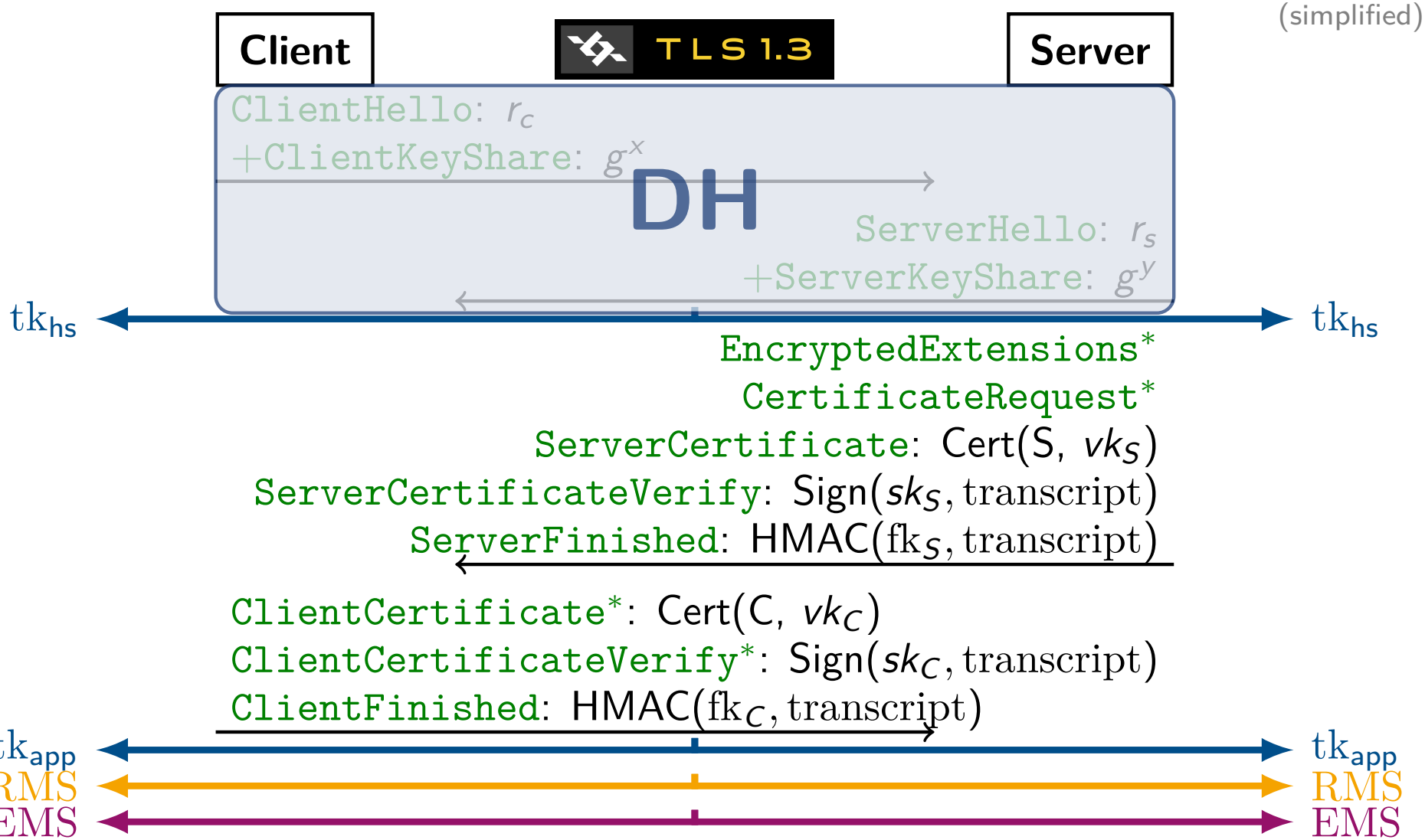


The TLS 1.3 Handshake

Full (EC)DHE Mode

ETH zürich

(simplified)

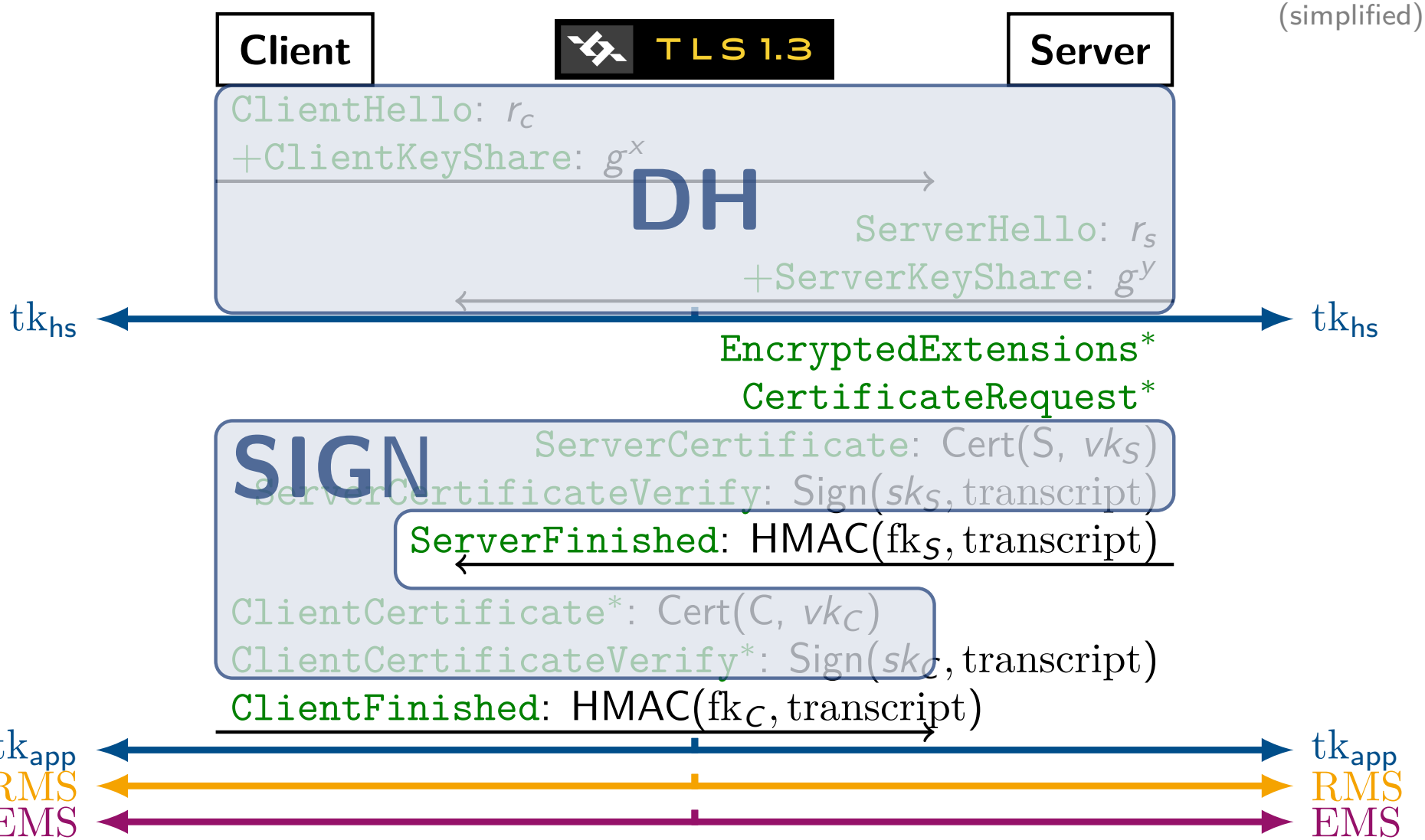


The TLS 1.3 Handshake

Full (EC)DHE Mode

ETH zürich

(simplified)

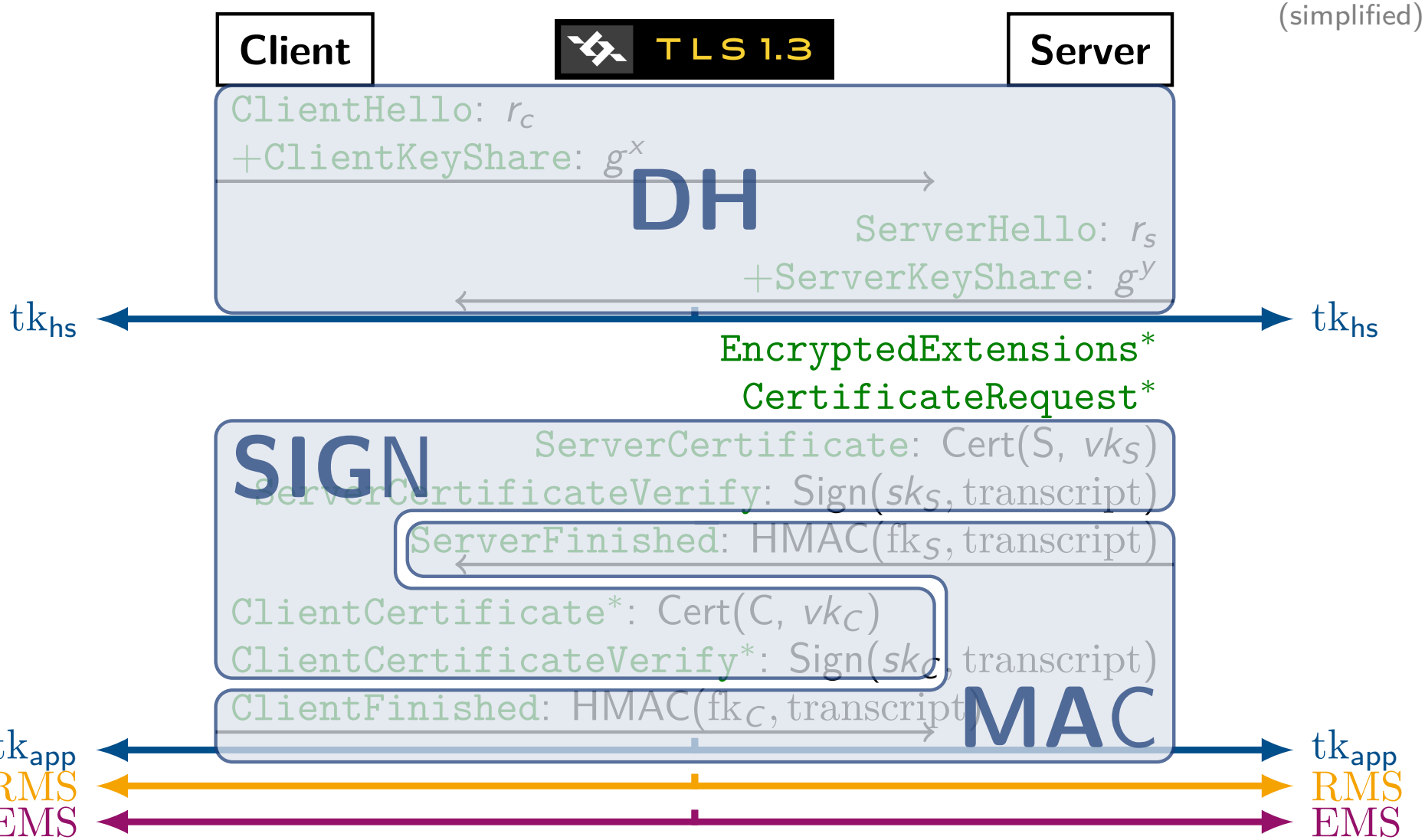


The TLS 1.3 Handshake

Full (EC)DHE Mode ≈ **SIGMA**: (DH w/) **SIG**n-and-**MA**c [Kra03]

ETH zürich

(simplified)

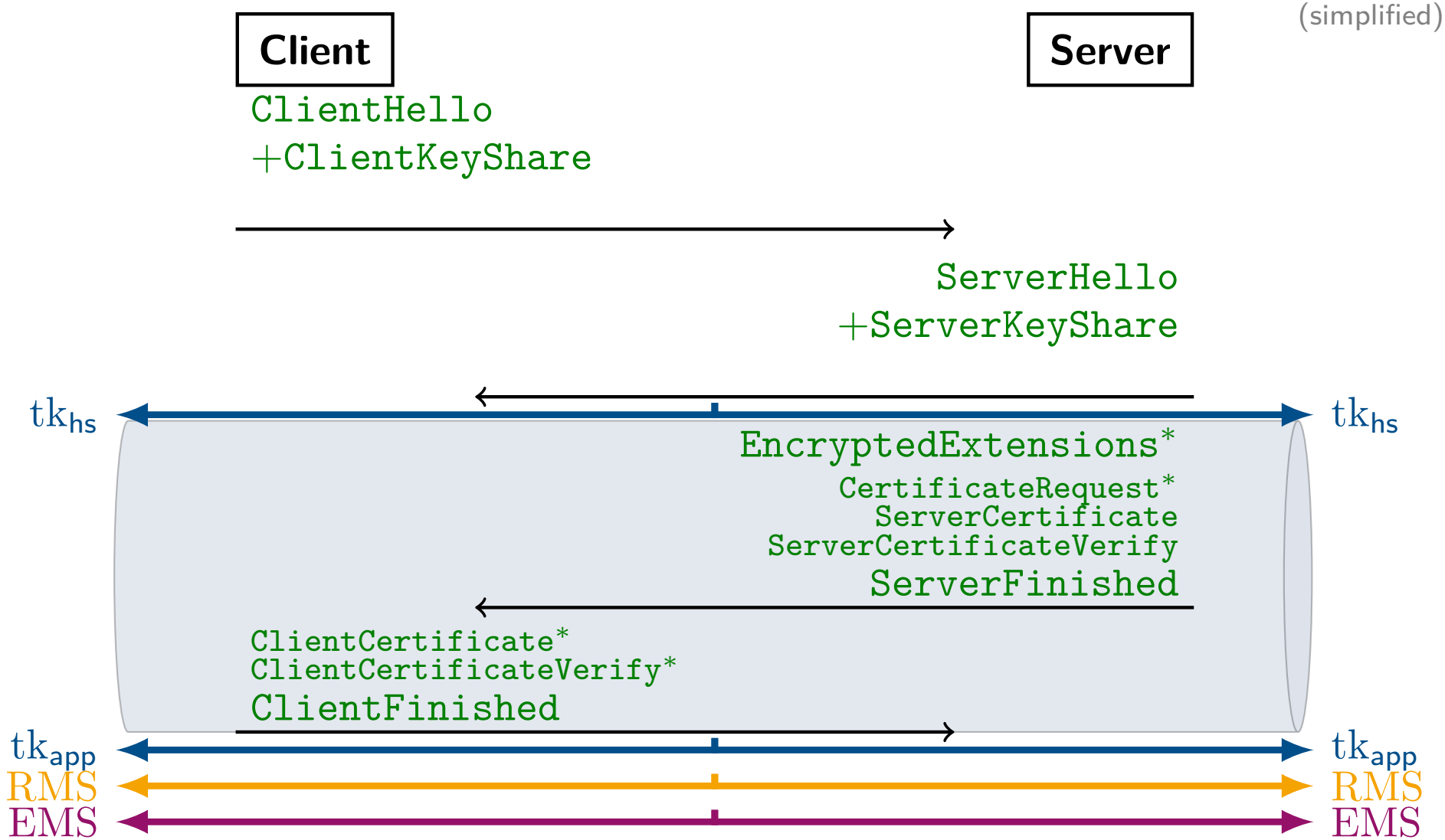


The TLS 1.3 Handshake

PSK / PSK-(EC)DHE Resumption Mode

ETH zürich

(simplified)

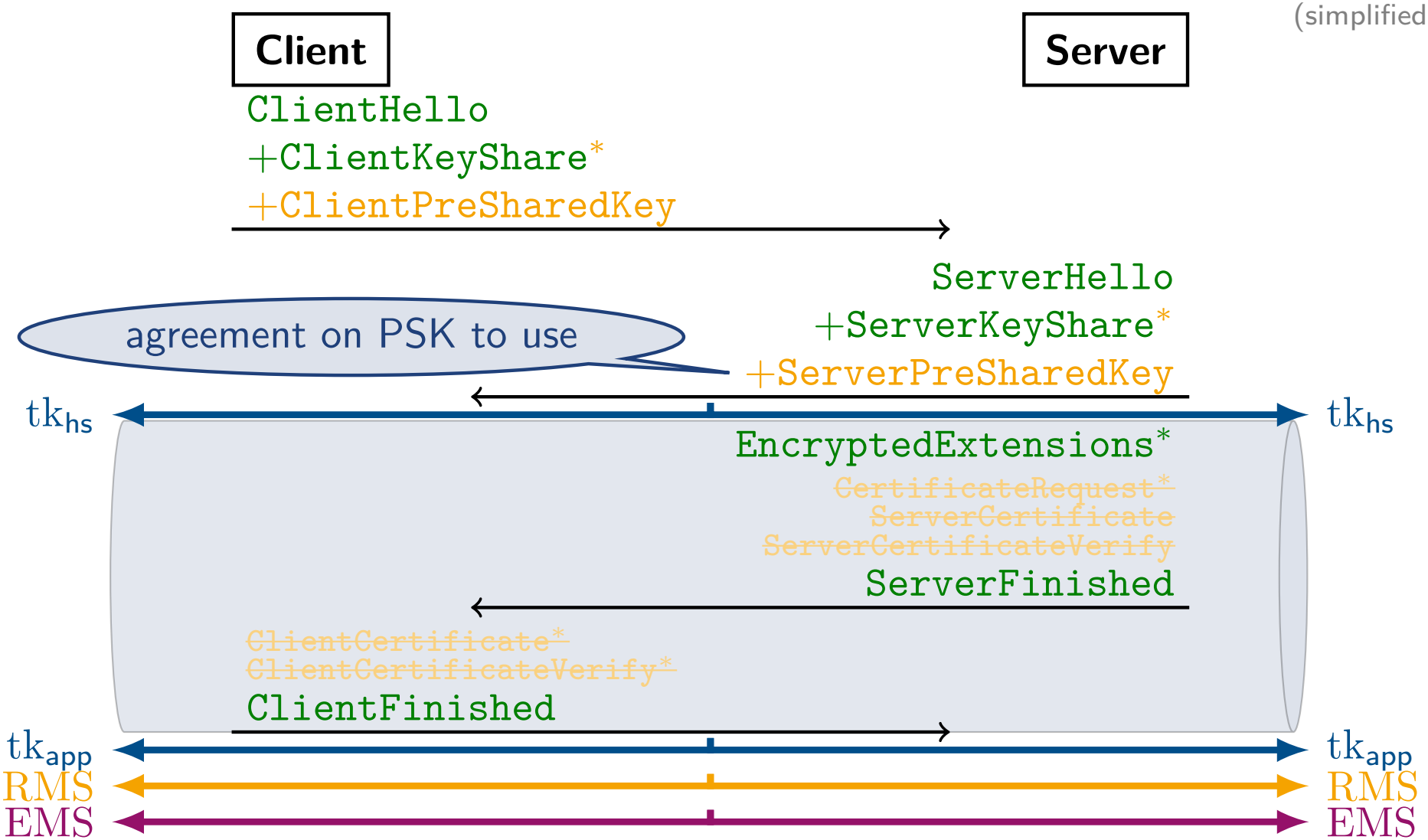


The TLS 1.3 Handshake

PSK / PSK-(EC)DHE Resumption Mode

ETH zürich

(simplified)

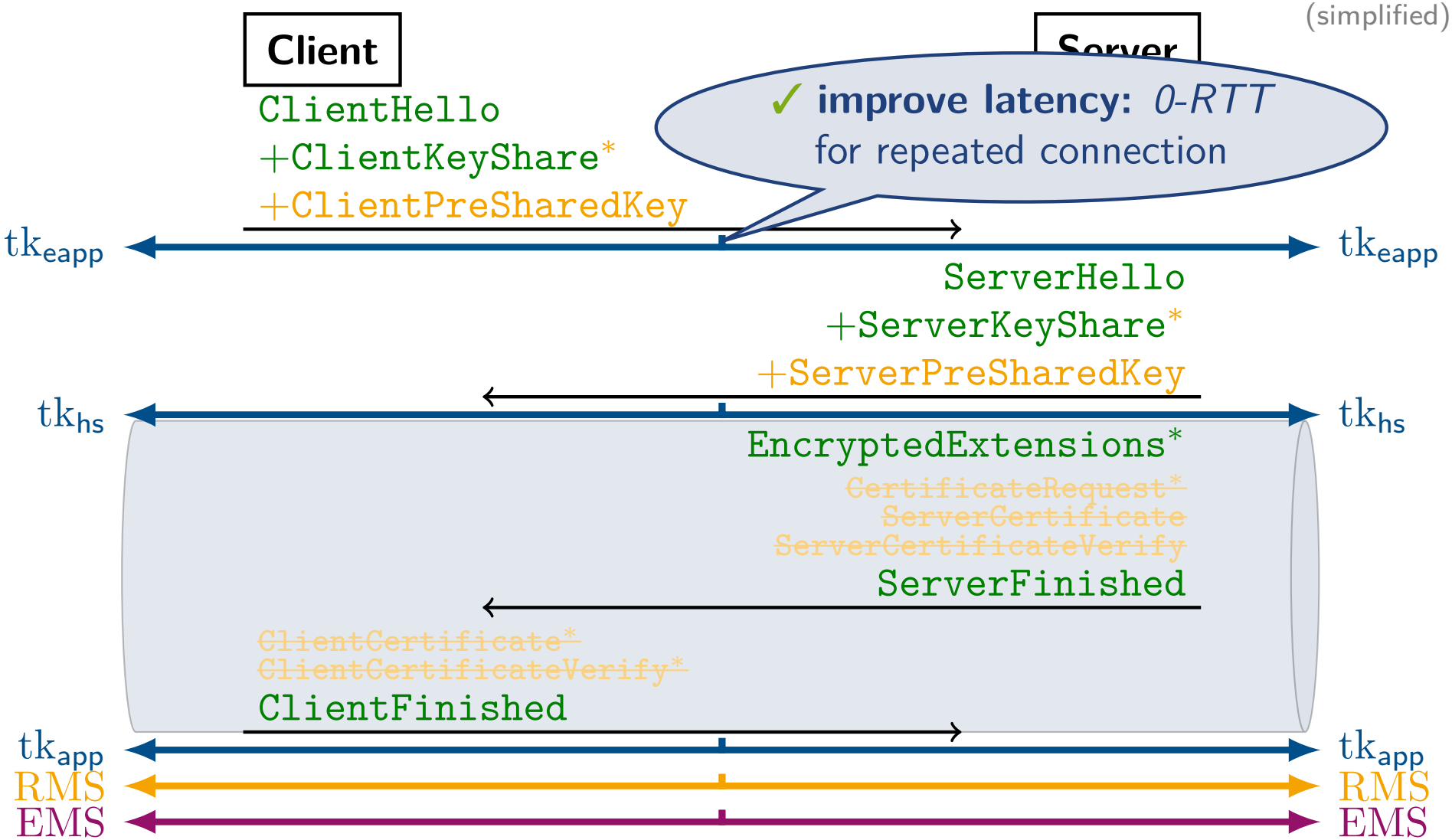


The TLS 1.3 Handshake

PSK / PSK-(EC)DHE Resumption Mode

ETH zürich

(simplified)



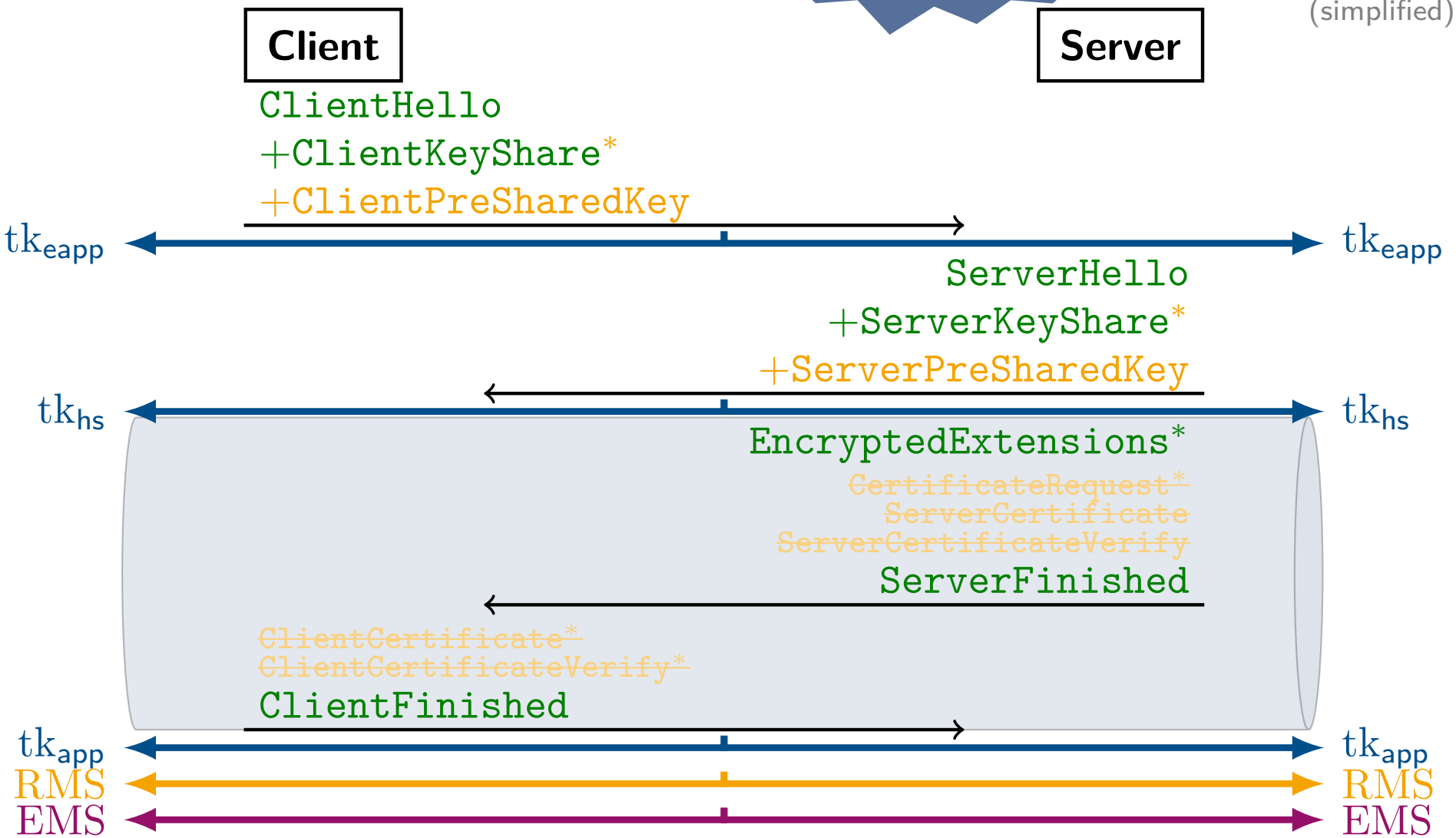
The TLS 1.3 Handshake

PSK / PSK-(EC)DHE Resumption Mode

multi-stage key exchange

ETH zürich

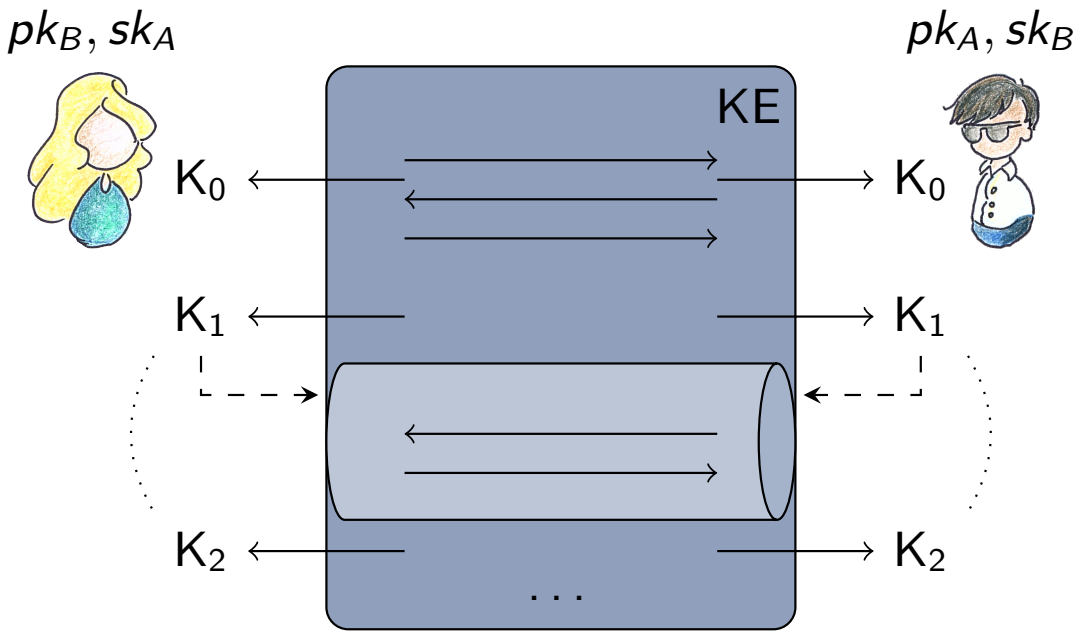
(simplified)



Multi-Stage Key Exchange Security

ETH zürich

(extending [Bellare–Rogaway93])

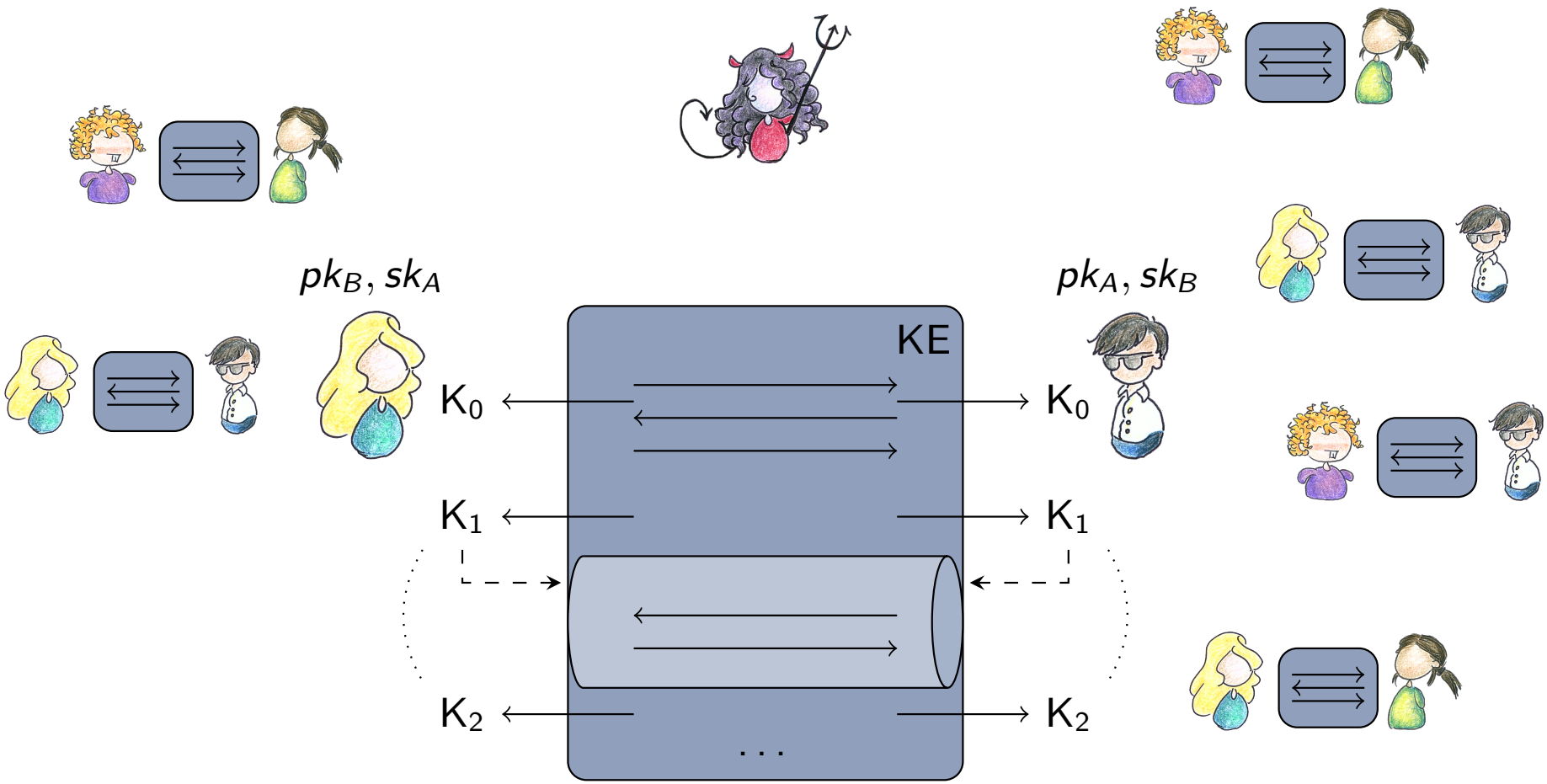


[FG14], [Dow+15], [Dow+16], [FG17], [Gün18], [Dow+21]

Multi-Stage Key Exchange Security

ETH zürich

(extending [Bellare–Rogaway93])

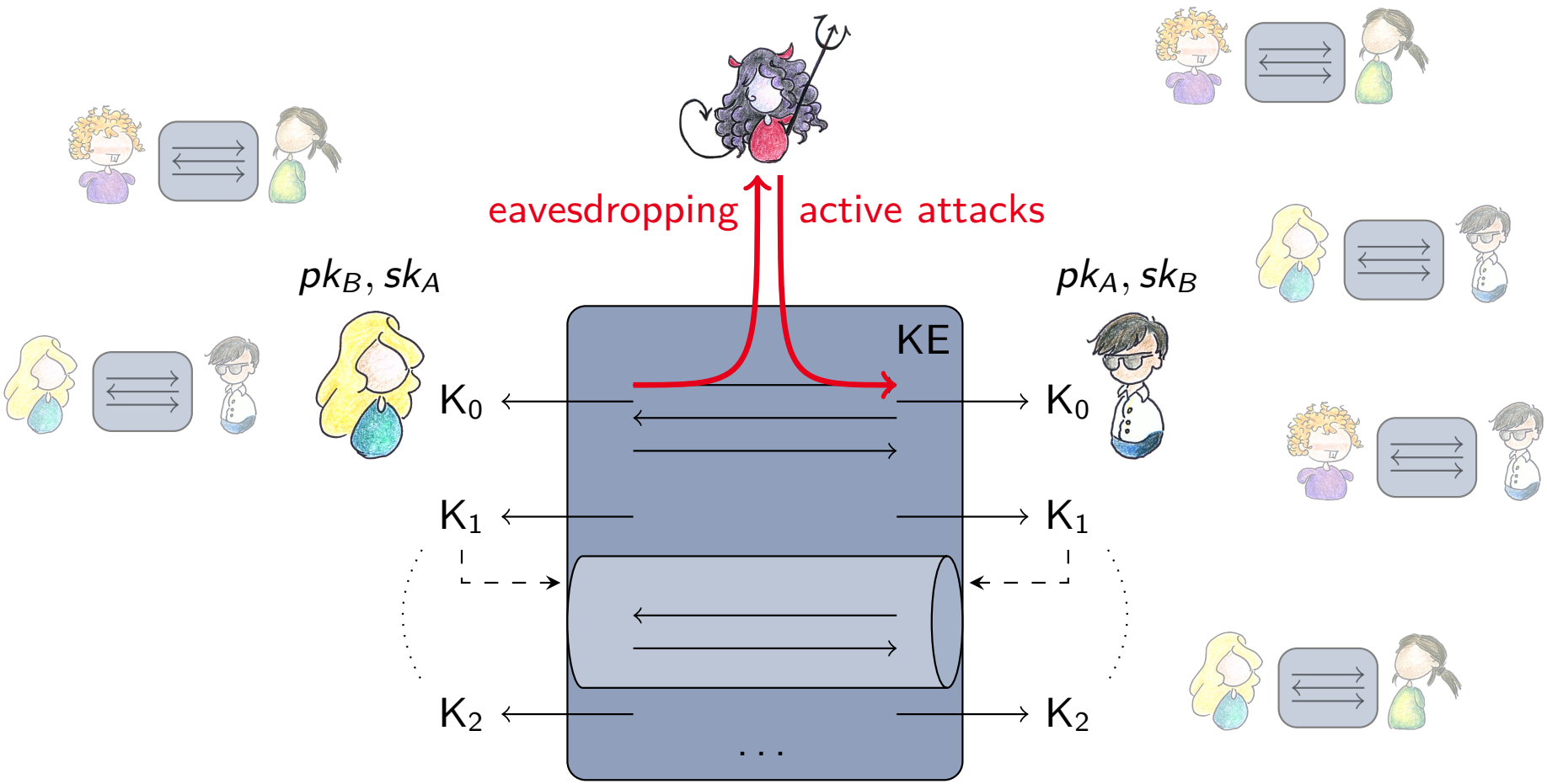


[FG14], [Dow+15], [Dow+16], [FG17], [Gün18], [Dow+21]

Multi-Stage Key Exchange Security

ETH zürich

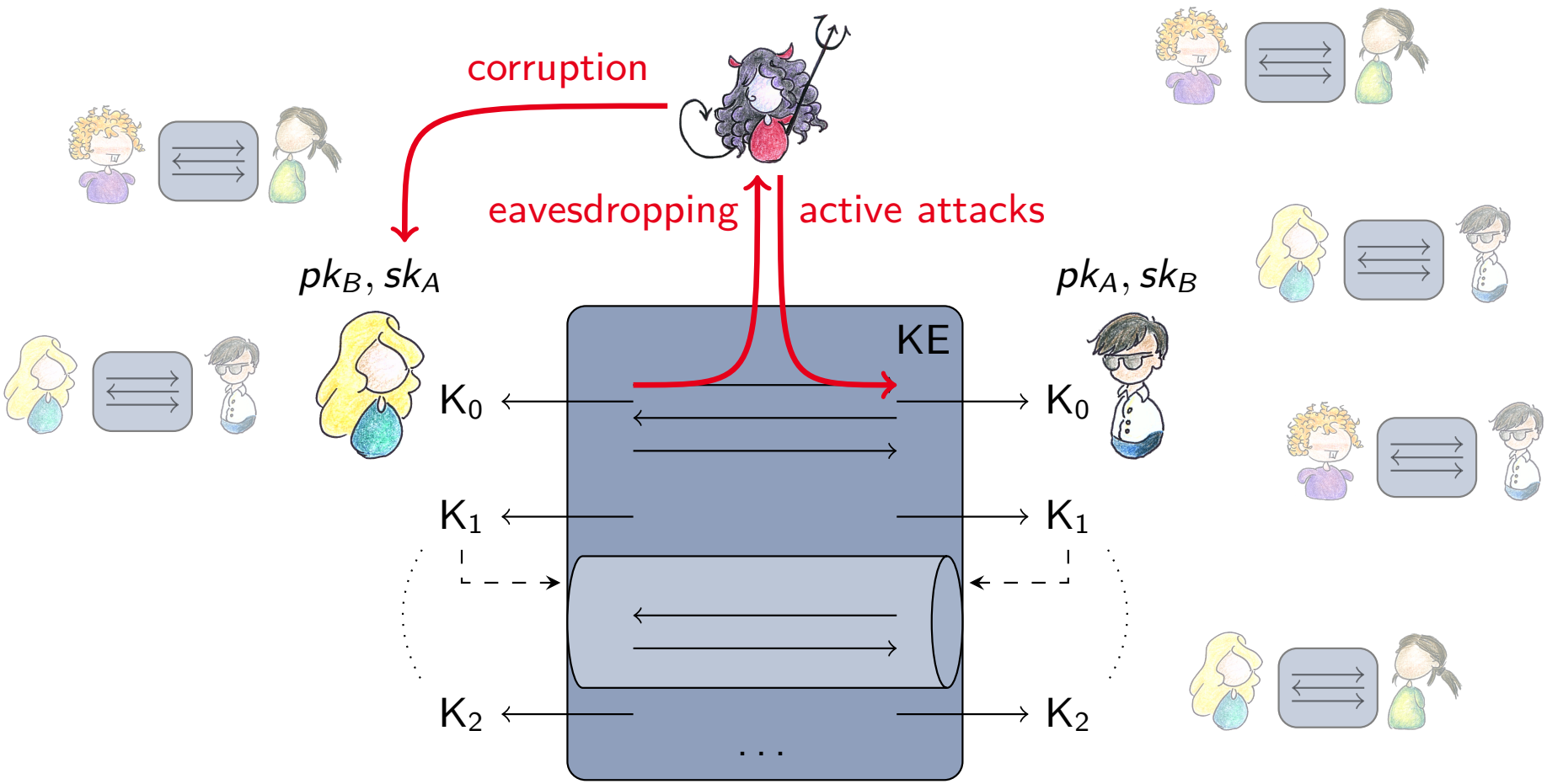
(extending [Bellare–Rogaway93])



[FG14], [Dow+15], [Dow+16], [FG17], [Gün18], [Dow+21]

Multi-Stage Key Exchange Security

(extending [Bellare–Rogaway93])

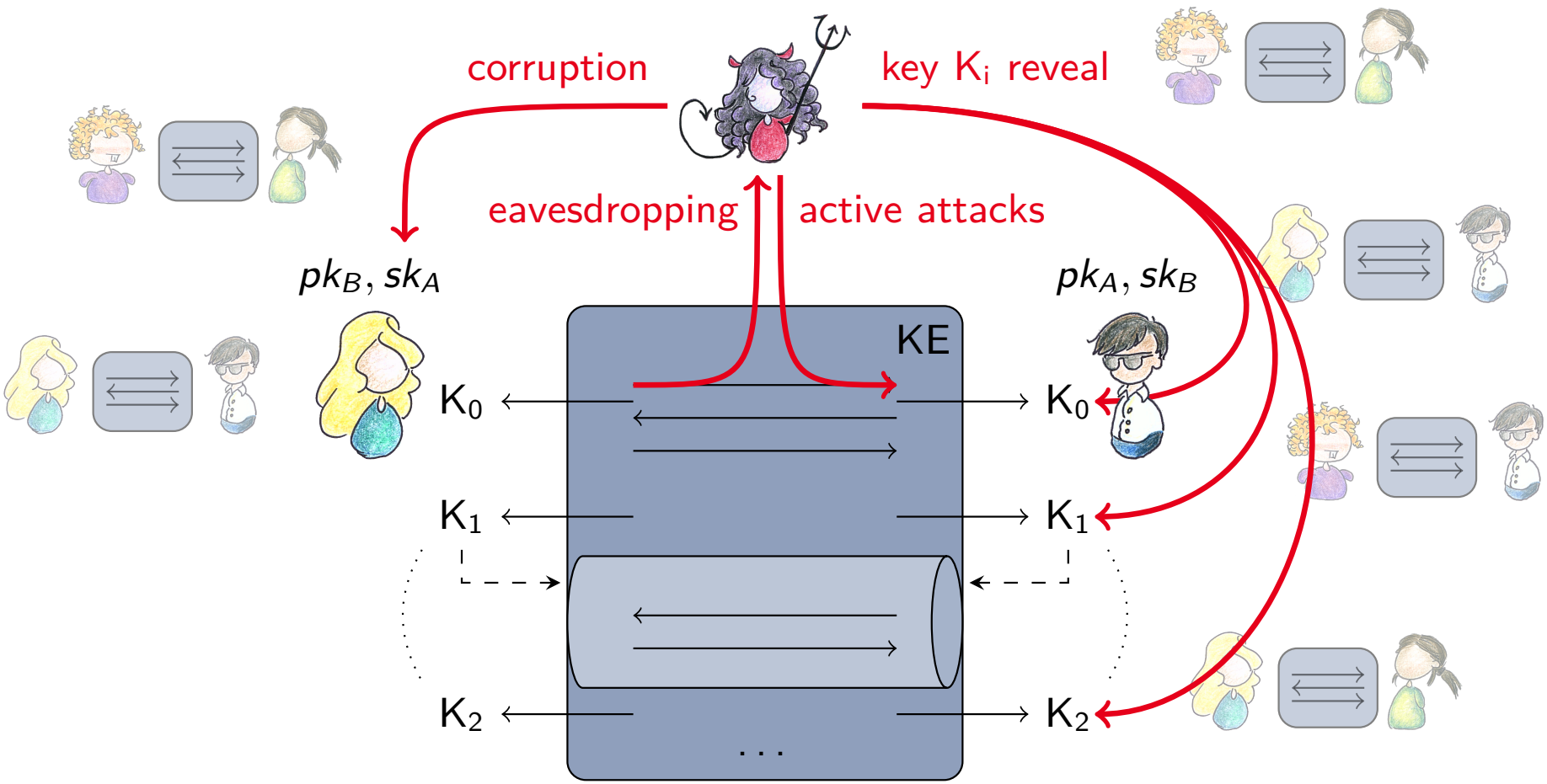


[FG14], [Dow+15], [Dow+16], [FG17], [Gün18], [Dow+21]

Multi-Stage Key Exchange Security

ETH zürich

(extending [Bellare–Rogaway93])

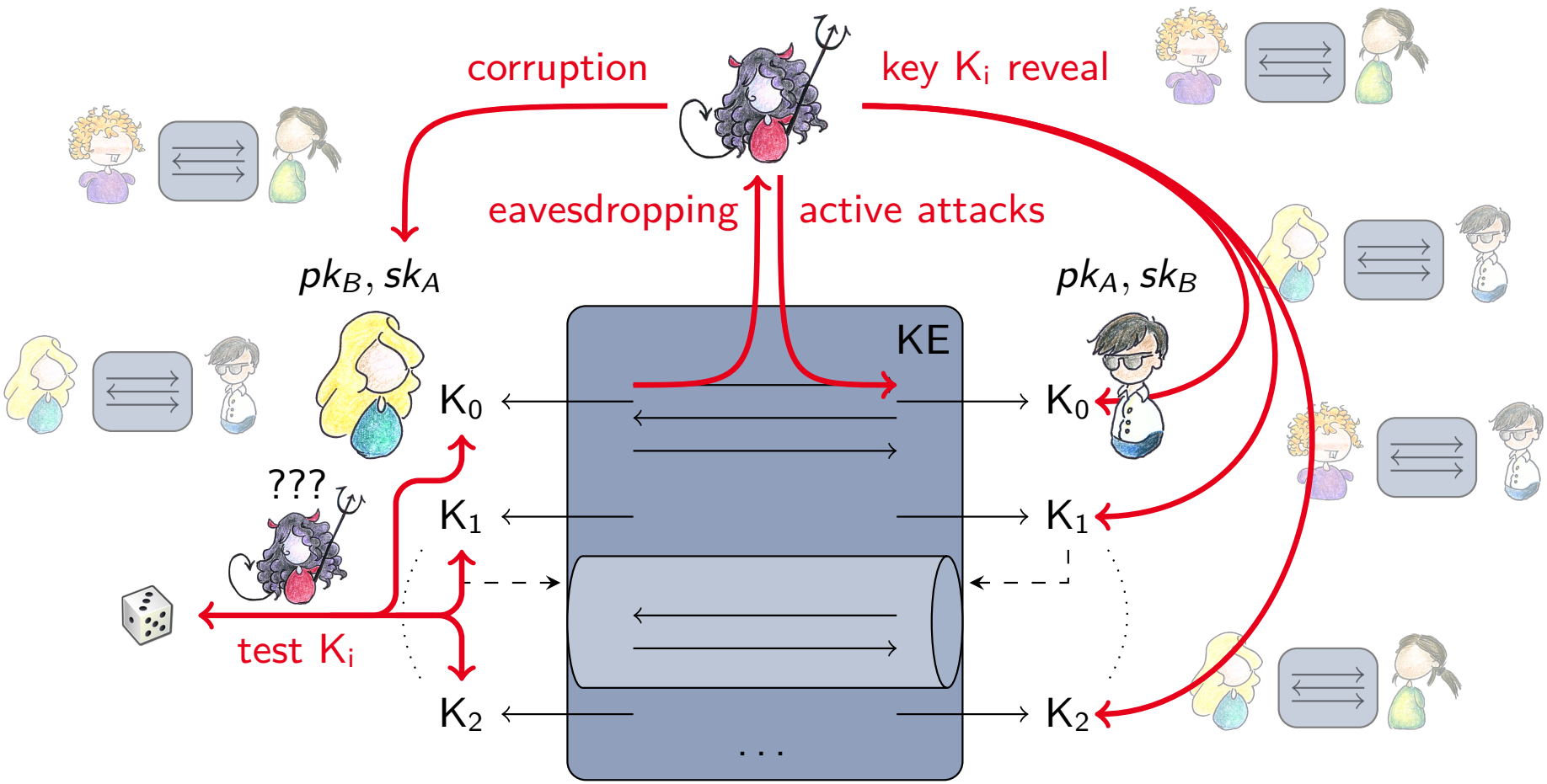


[FG14], [Dow+15], [Dow+16], [FG17], [Gün18], [Dow+21]

Multi-Stage Key Exchange Security

ETH zürich

(extending [Bellare–Rogaway93])



[FG14], [Dow+15], [Dow+16], [FG17], [Gün18], [Dow+21]

Modeling Key Exchange Security

The communication/security model: a simplified example

ETH zürich

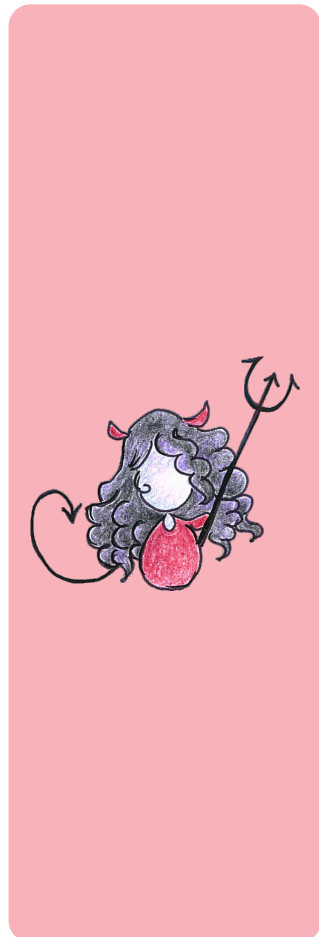
$$\mathrm{KE}(\mathit{own\ id}, \mathit{peer\ id}, \mathit{sk}_{\mathit{oid}}, \mathit{pk}_{\mathit{pid}}, \mathit{msg\ in}, \dots) \mapsto (\mathit{msg\ out}, \mathit{status}, K, \dots)$$

Modeling Key Exchange Security

The communication/security model: a simplified example

ETH zürich

$$\mathrm{KE}(\mathit{own\ id}, \mathit{peer\ id}, \mathit{sk}_{\mathit{oid}}, \mathit{pk}_{\mathit{pid}}, \mathit{msg\ in}, \ldots) \mapsto (\mathit{msg\ out}, \mathit{status}, K, \ldots)$$



Modeling Key Exchange Security

The communication/security model: a simplified example

ETH zürich

$$\mathrm{KE}(\mathit{own\ id}, \mathit{peer\ id}, \mathit{sk}_{\mathit{oid}}, \mathit{pk}_{\mathit{pid}}, \mathit{msg\ in}, \ldots) \mapsto (\mathit{msg\ out}, \mathit{status}, K, \ldots)$$

sk_A

sk_B

sk_C

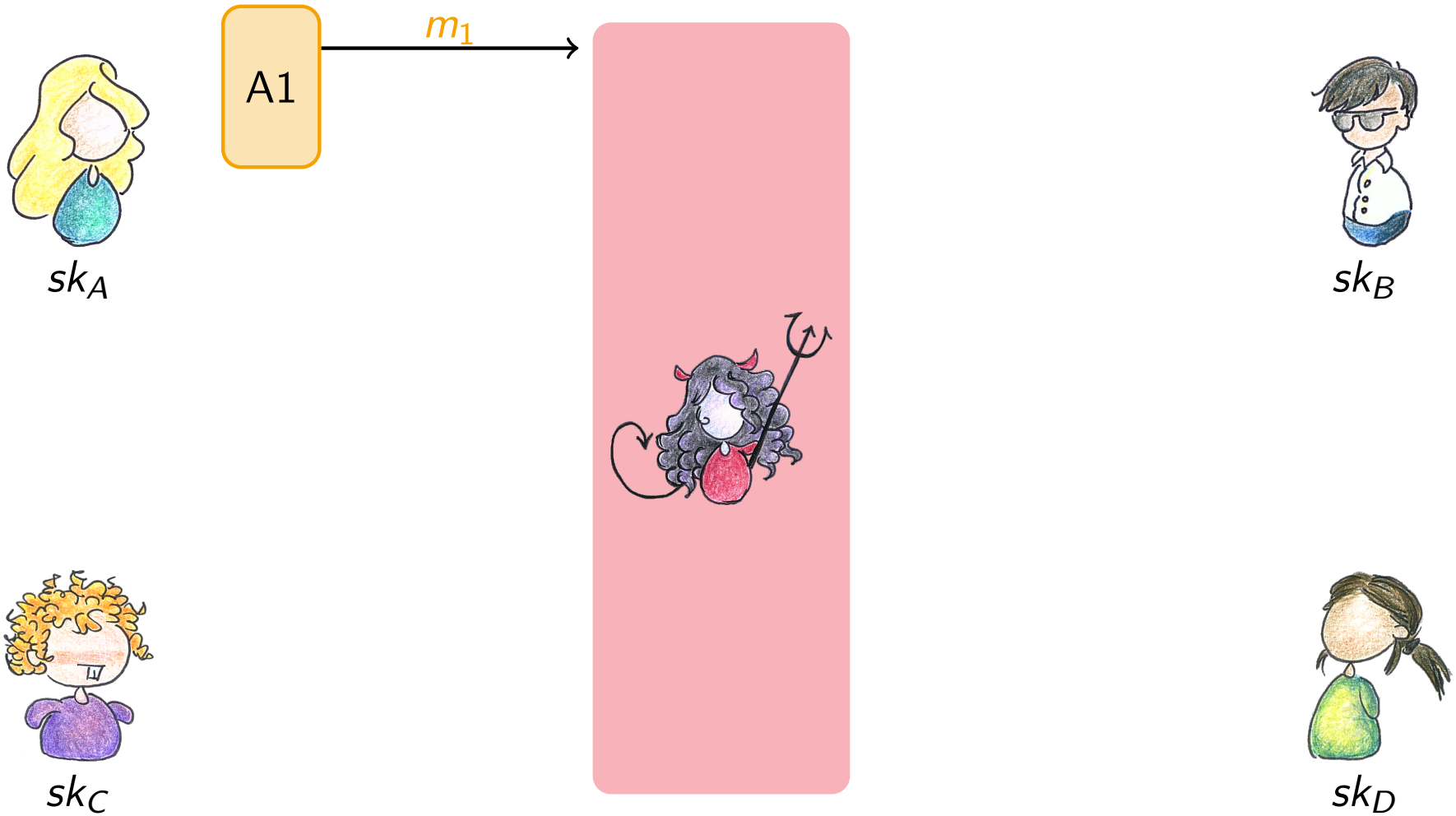
sk_D

Modeling Key Exchange Security

The communication/security model: a simplified example

ETH zürich

$$\mathrm{KE}(\textit{own id}, \textit{peer id}, sk_{oid}, pk_{pid}, \textit{msg in}, \dots) \mapsto (\textit{msg out}, \textit{status}, K, \dots)$$

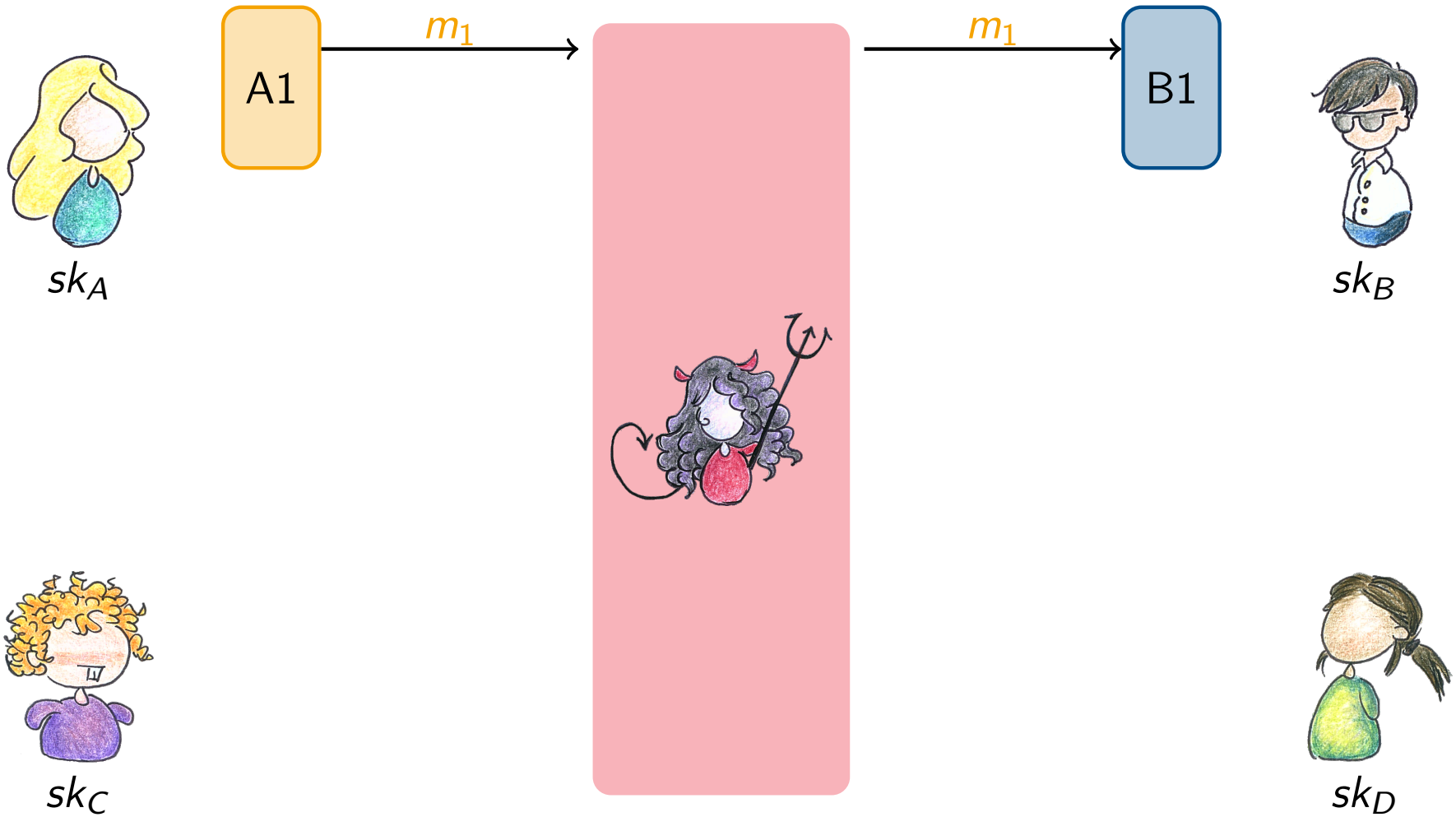


Modeling Key Exchange Security

The communication/security model: a simplified example

ETH zürich

$$\mathrm{KE}(\textit{own id}, \textit{peer id}, sk_{oid}, pk_{pid}, \textit{msg in}, \dots) \mapsto (\textit{msg out}, \textit{status}, K, \dots)$$

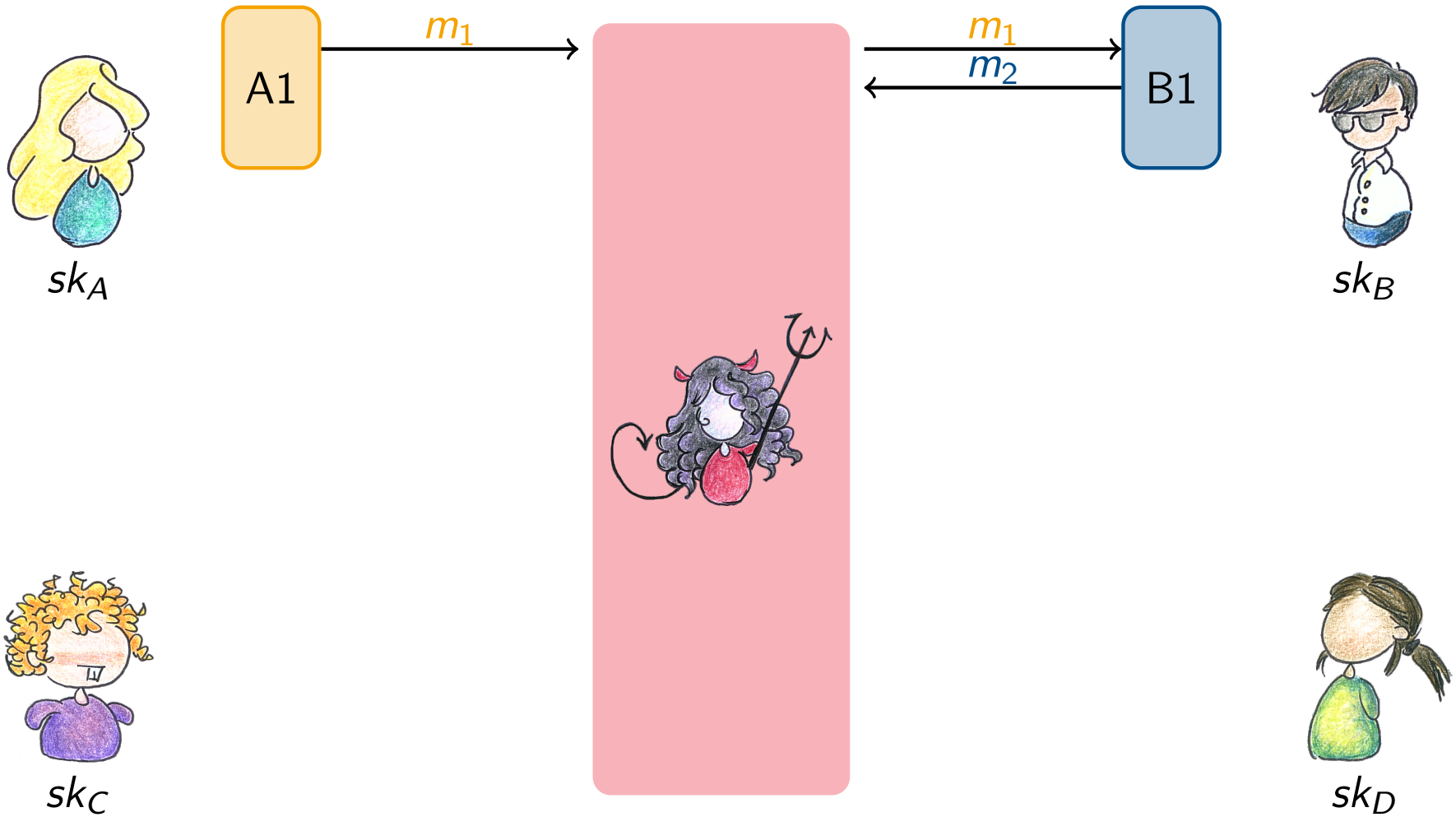


Modeling Key Exchange Security

The communication/security model: a simplified example

ETH zürich

$$\mathrm{KE}(\mathit{own\ id}, \mathit{peer\ id}, \mathit{sk}_{\mathit{oid}}, \mathit{pk}_{\mathit{pid}}, \mathit{msg\ in}, \ldots) \mapsto (\mathit{msg\ out}, \mathit{status}, K, \ldots)$$

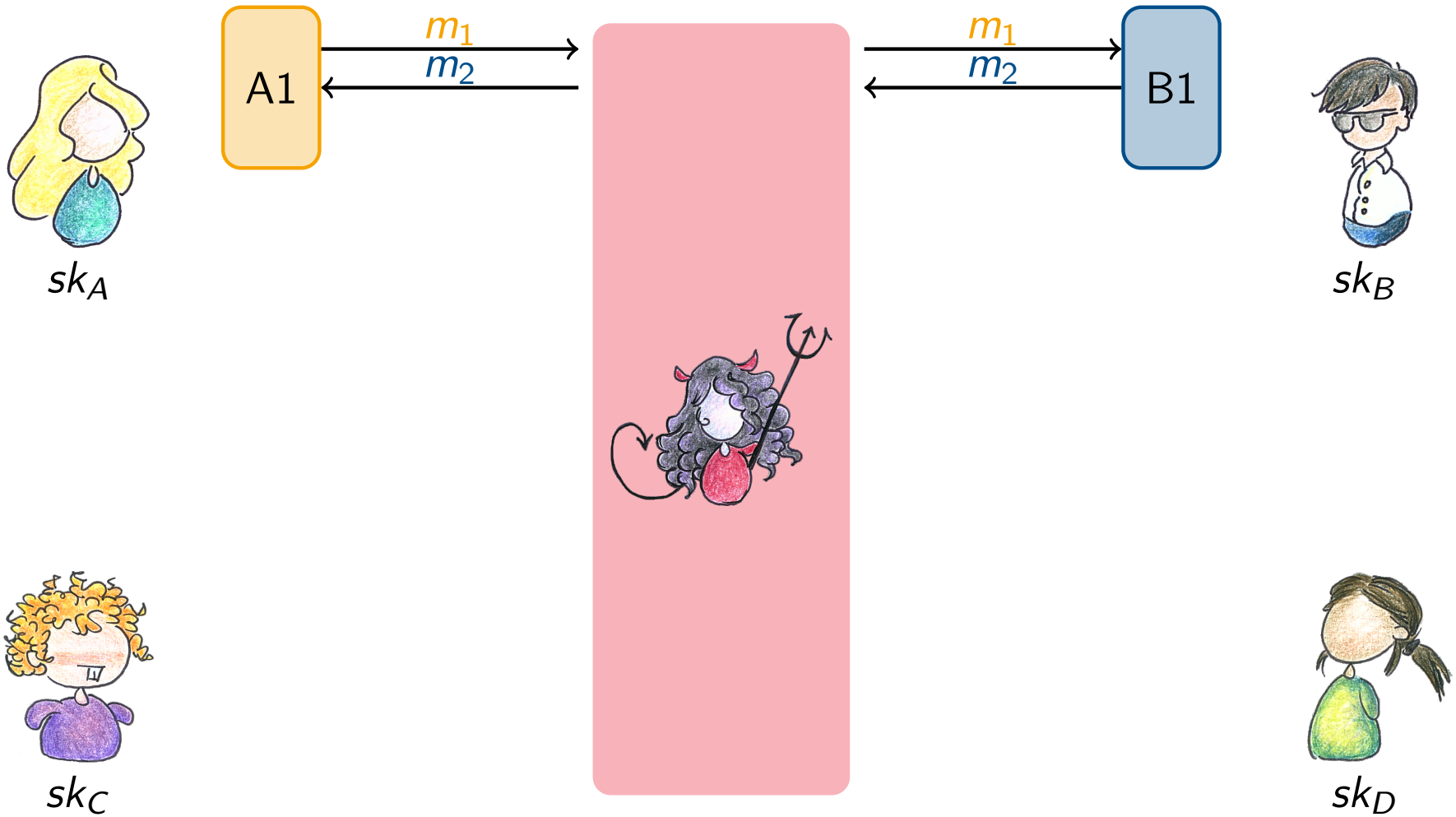


Modeling Key Exchange Security

The communication/security model: a simplified example

ETH zürich

$$\mathrm{KE}(\textit{own id}, \textit{peer id}, sk_{oid}, pk_{pid}, \textit{msg in}, \dots) \mapsto (\textit{msg out}, \textit{status}, K, \dots)$$

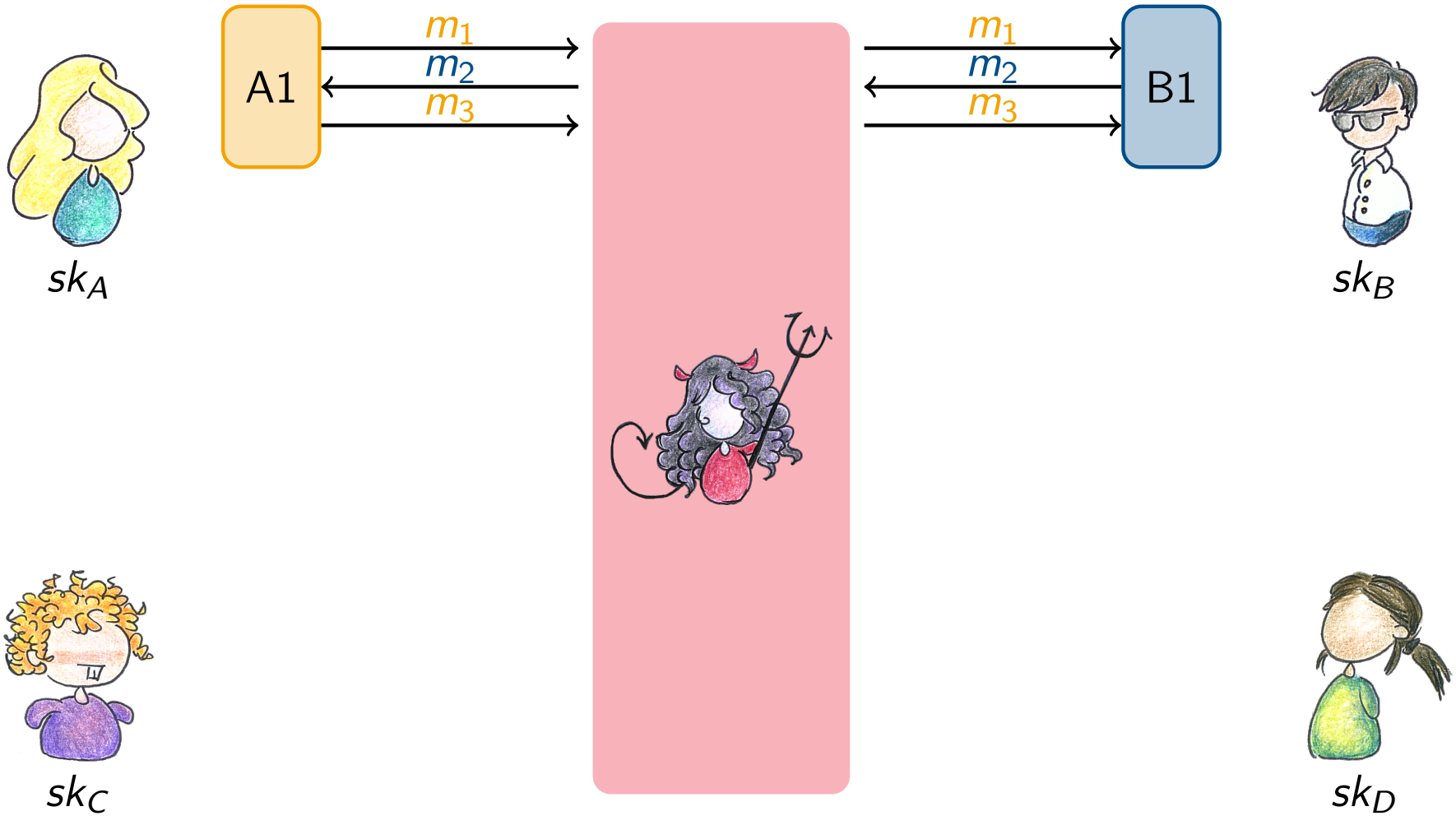


Modeling Key Exchange Security

The communication/security model: a simplified example

ETH zürich

$$\mathrm{KE}(\textit{own id}, \textit{peer id}, sk_{oid}, pk_{pid}, \textit{msg in}, \dots) \mapsto (\textit{msg out}, \textit{status}, K, \dots)$$

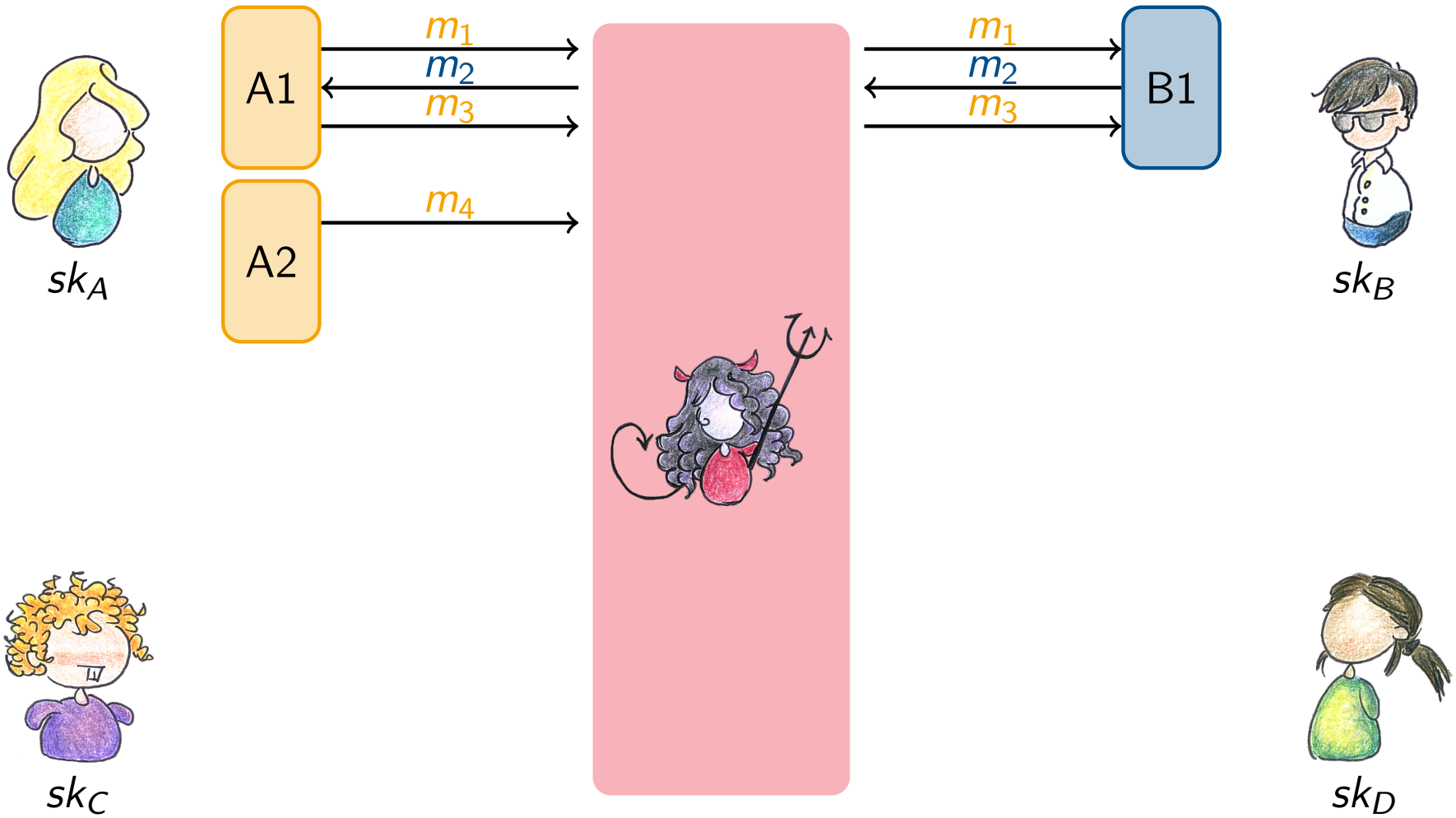


Modeling Key Exchange Security

The communication/security model: a simplified example

ETH zürich

$$\mathrm{KE}(\textit{own id}, \textit{peer id}, sk_{oid}, pk_{pid}, \textit{msg in}, \dots) \mapsto (\textit{msg out}, \textit{status}, K, \dots)$$

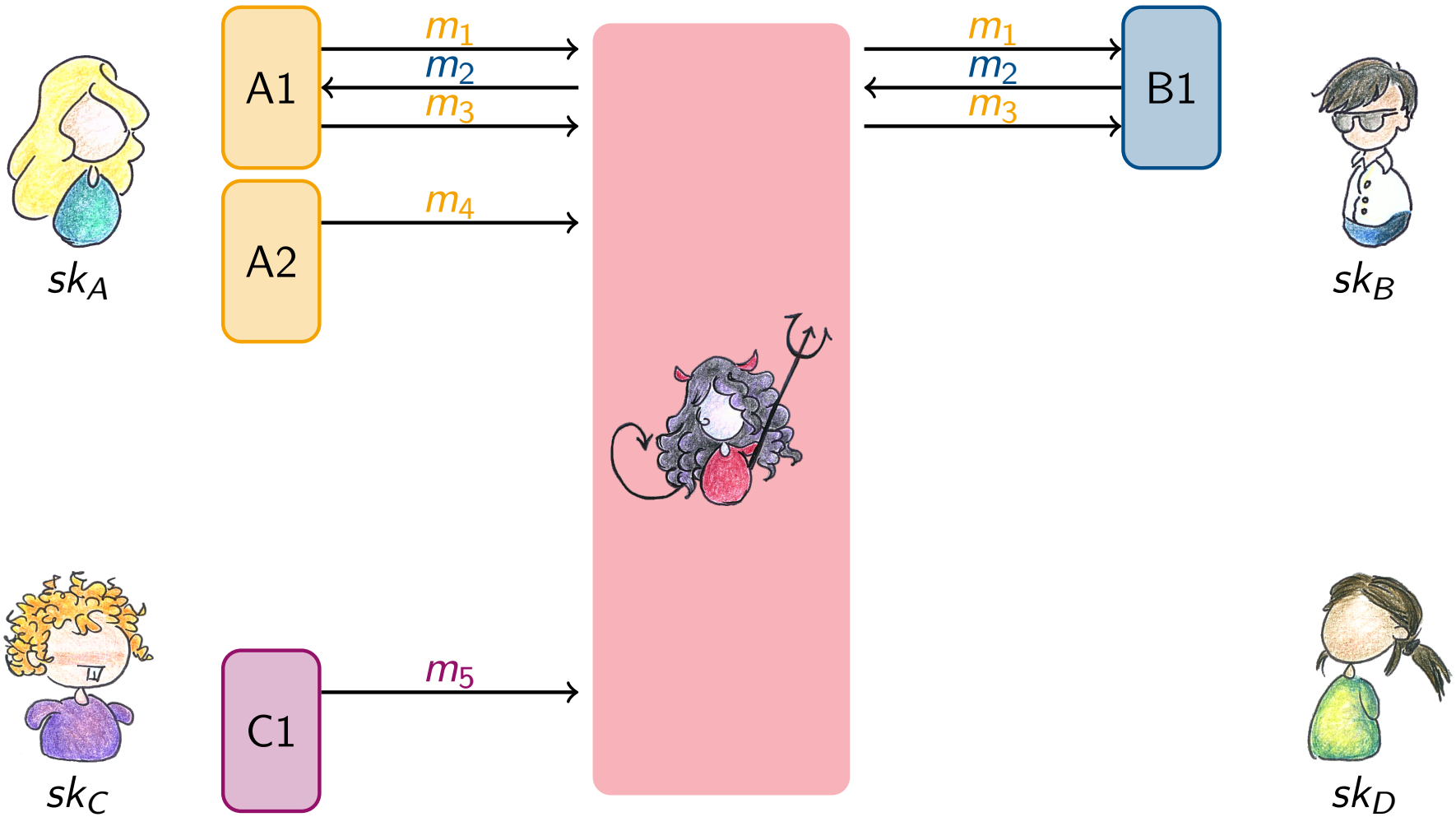


Modeling Key Exchange Security

The communication/security model: a simplified example

ETH zürich

$$\mathrm{KE}(\textit{own id}, \textit{peer id}, sk_{oid}, pk_{pid}, \textit{msg in}, \dots) \mapsto (\textit{msg out}, \textit{status}, K, \dots)$$

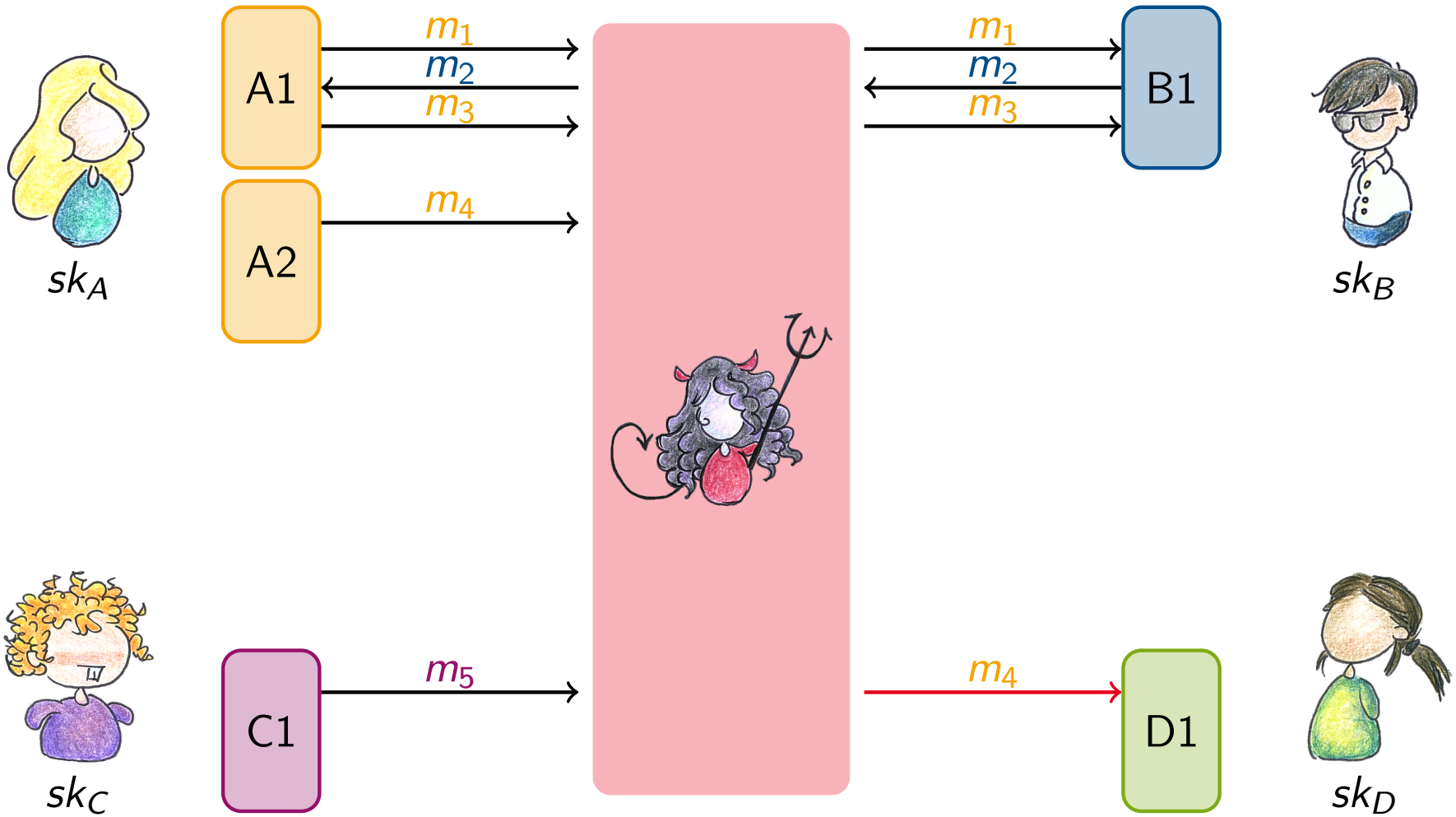


Modeling Key Exchange Security

The communication/security model: a simplified example

ETH zürich

$$\mathrm{KE}(\textit{own id}, \textit{peer id}, sk_{oid}, pk_{pid}, \textit{msg in}, \dots) \mapsto (\textit{msg out}, \textit{status}, K, \dots)$$

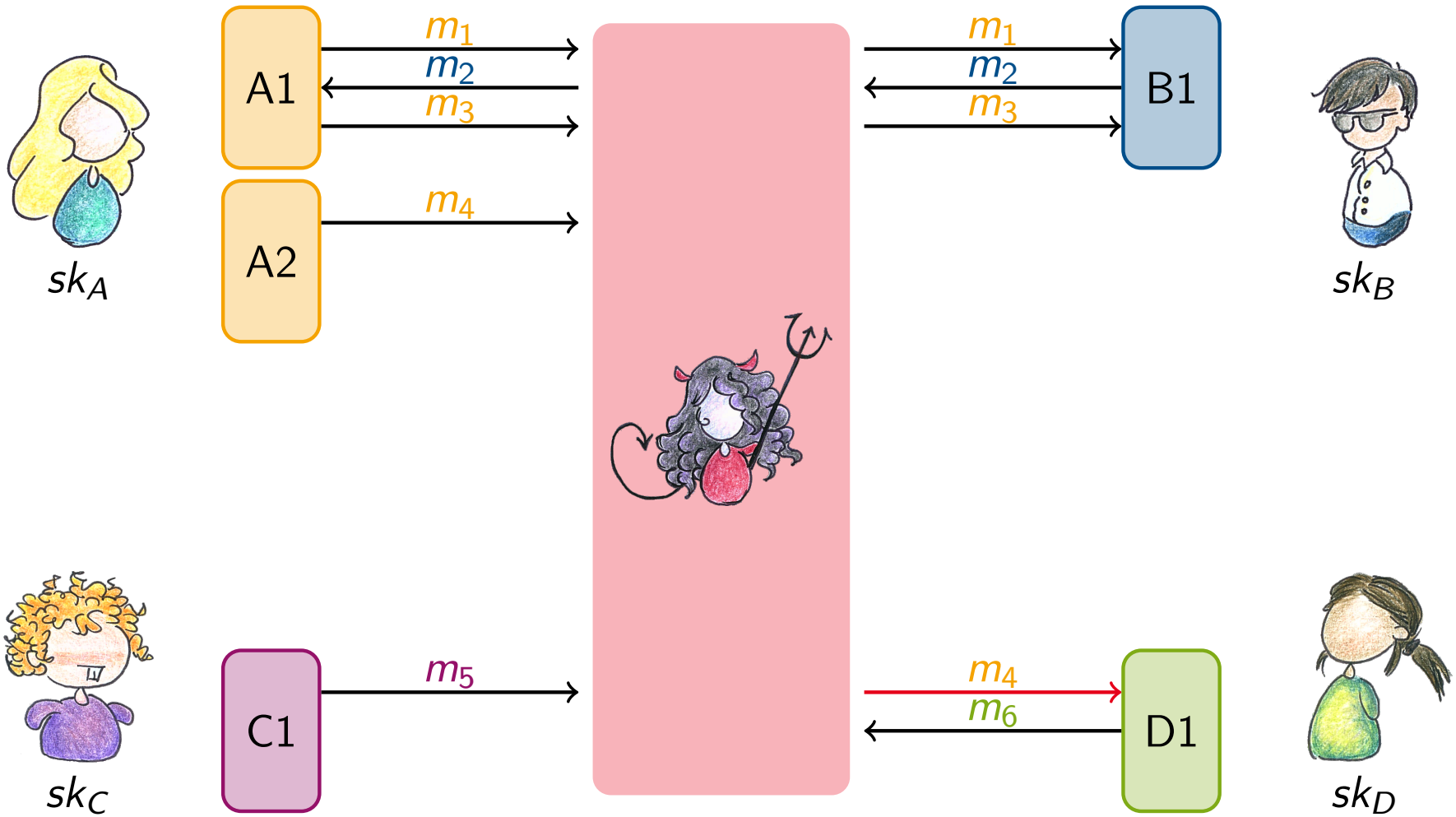


Modeling Key Exchange Security

The communication/security model: a simplified example

ETH zürich

$$\mathrm{KE}(\textit{own id}, \textit{peer id}, sk_{oid}, pk_{pid}, \textit{msg in}, \dots) \mapsto (\textit{msg out}, \textit{status}, K, \dots)$$

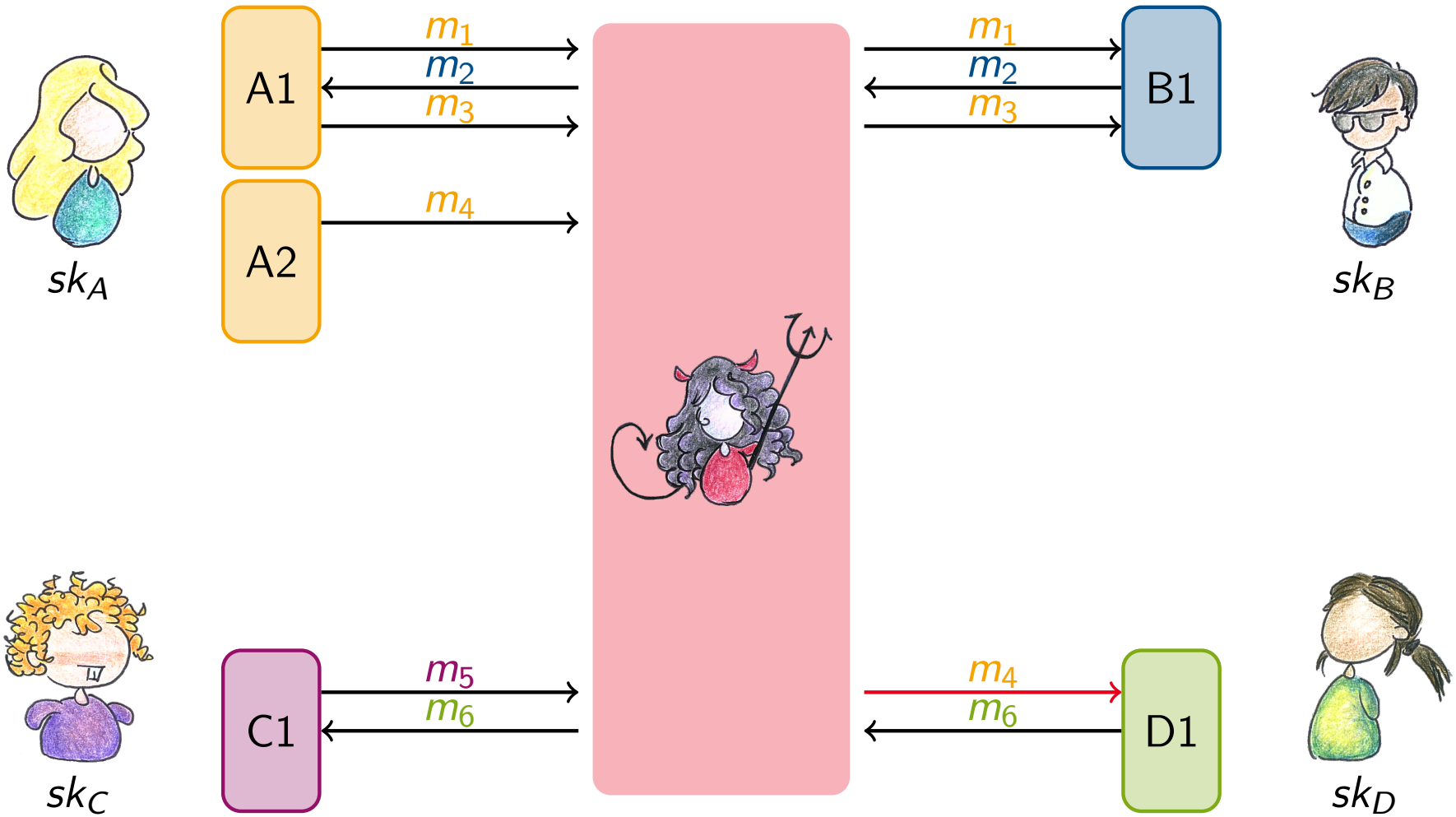


Modeling Key Exchange Security

The communication/security model: a simplified example

ETH zürich

$$\mathrm{KE}(\textit{own id}, \textit{peer id}, sk_{oid}, pk_{pid}, \textit{msg in}, \dots) \mapsto (\textit{msg out}, \textit{status}, K, \dots)$$

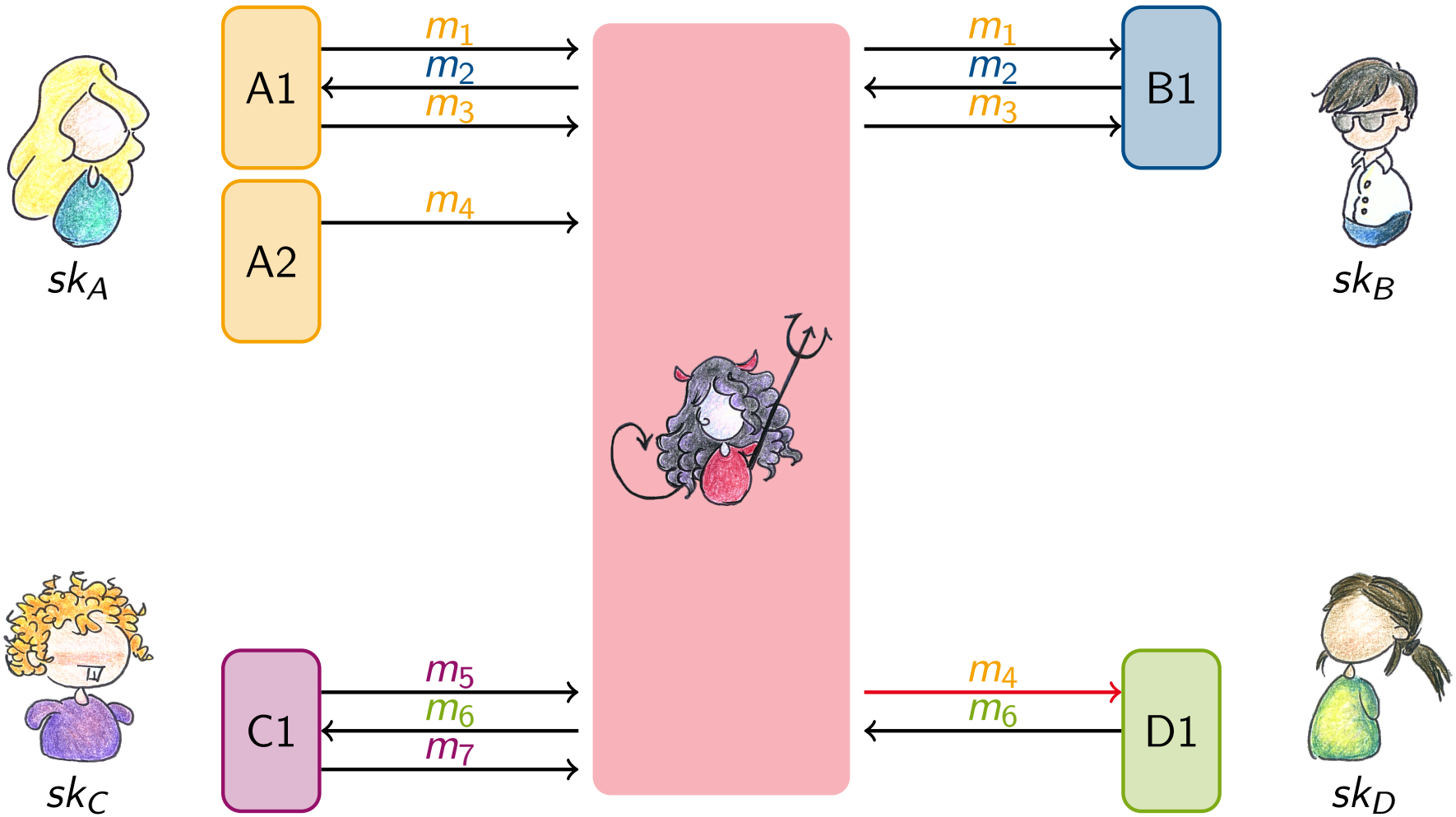


Modeling Key Exchange Security

The communication/security model: a simplified example

ETH zürich

$$\mathrm{KE}(\textit{own id}, \textit{peer id}, sk_{oid}, pk_{pid}, \textit{msg in}, \dots) \mapsto (\textit{msg out}, \textit{status}, K, \dots)$$

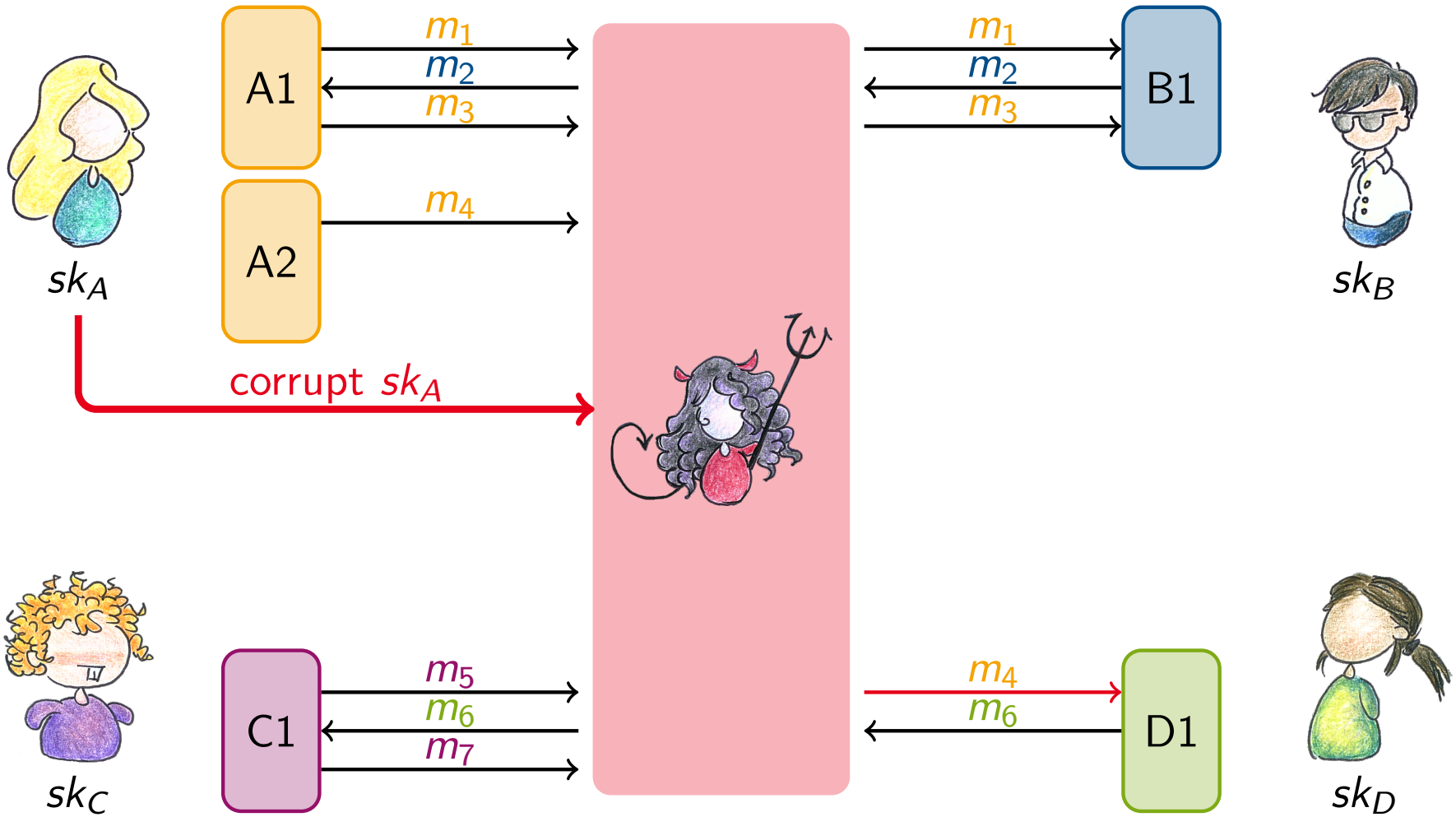


Modeling Key Exchange Security

The communication/security model: a simplified example

ETH zürich

$$\mathrm{KE}(\textit{own id}, \textit{peer id}, sk_{oid}, pk_{pid}, \textit{msg in}, \dots) \mapsto (\textit{msg out}, \textit{status}, K, \dots)$$

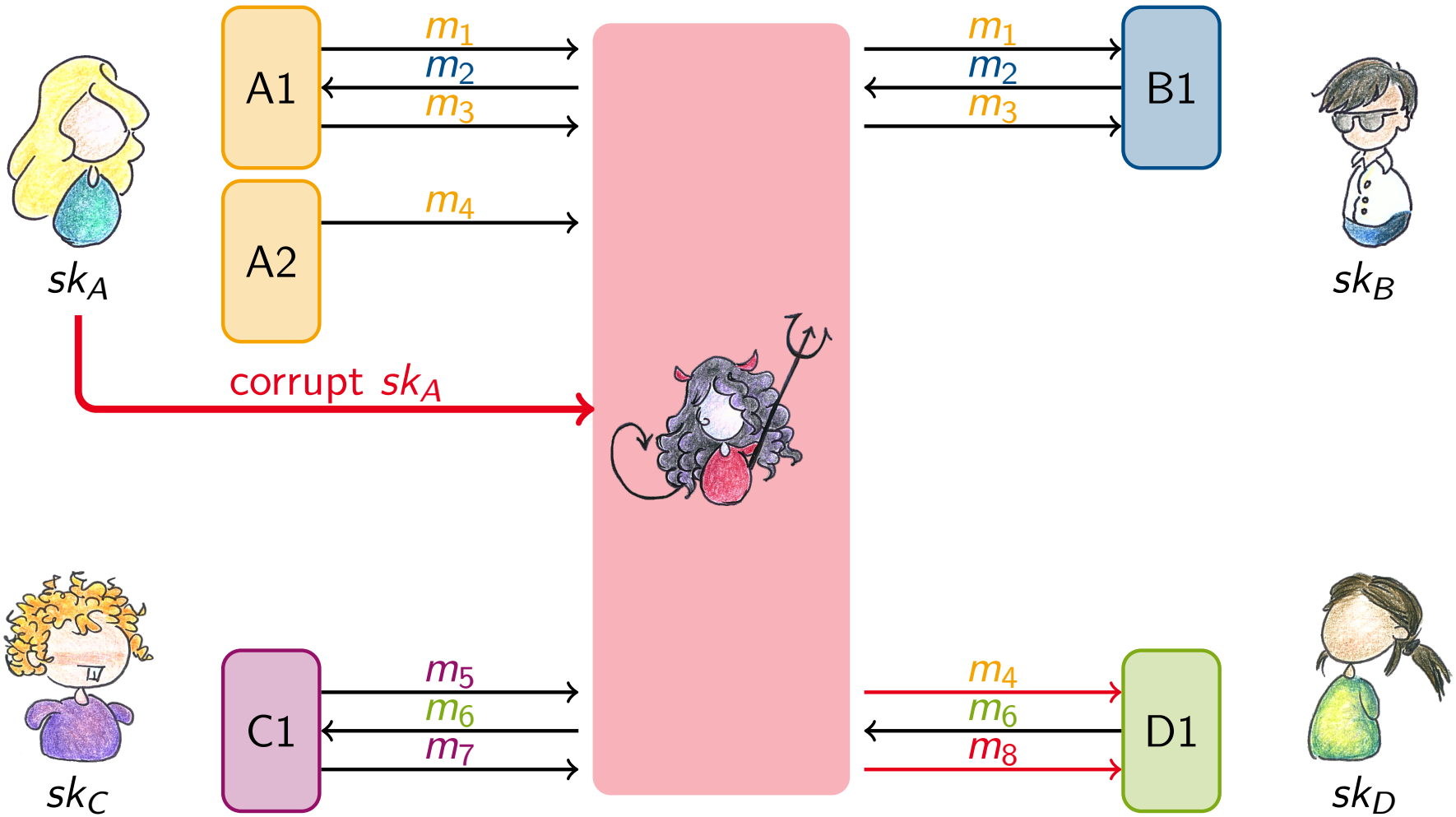


Modeling Key Exchange Security

The communication/security model: a simplified example

ETH zürich

$$\mathrm{KE}(\textit{own id}, \textit{peer id}, sk_{oid}, pk_{pid}, \textit{msg in}, \dots) \mapsto (\textit{msg out}, \textit{status}, K, \dots)$$

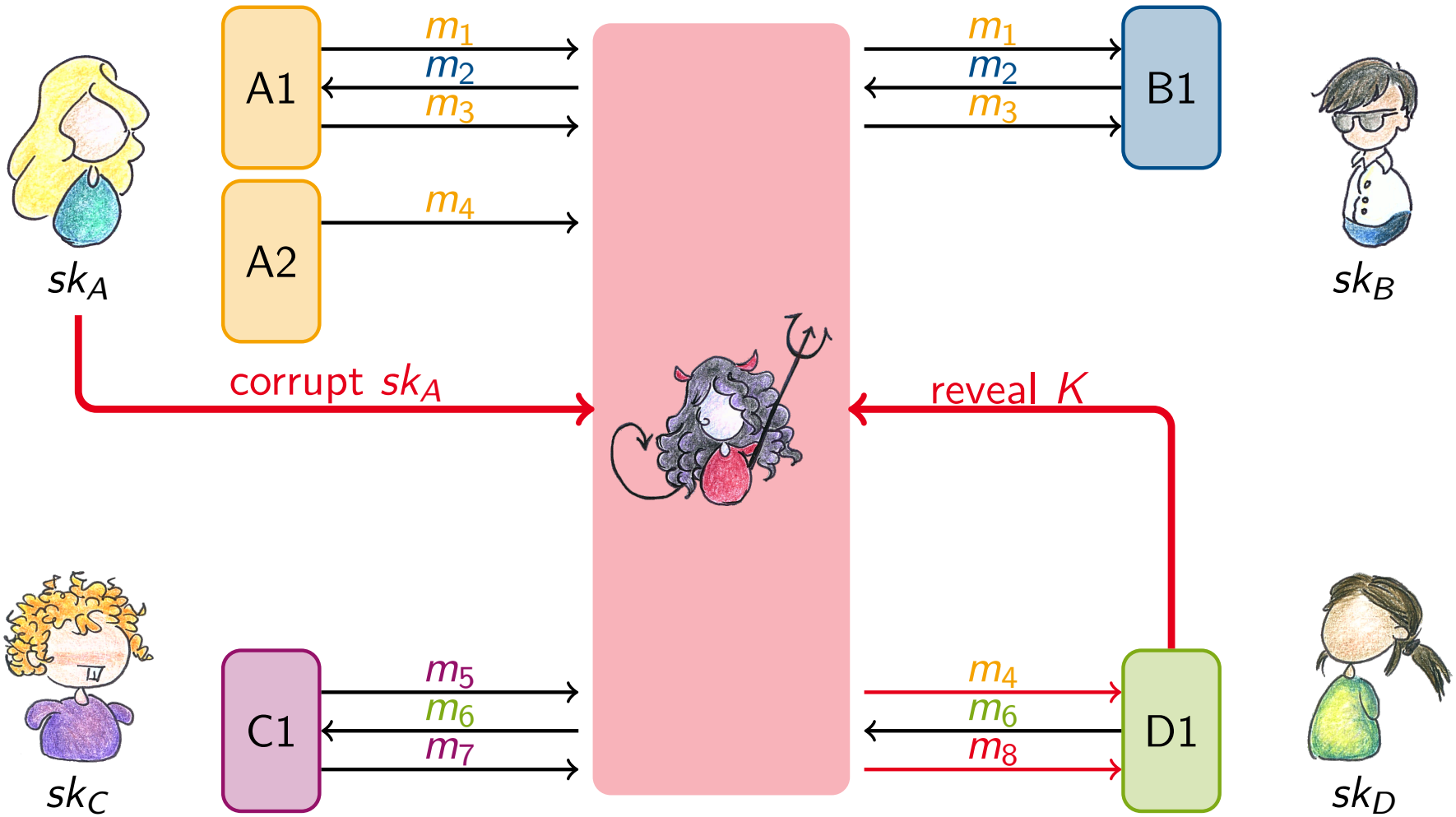


Modeling Key Exchange Security

The communication/security model: a simplified example

ETH zürich

$$\mathrm{KE}(\textit{own id}, \textit{peer id}, sk_{oid}, pk_{pid}, \textit{msg in}, \dots) \mapsto (\textit{msg out}, \textit{status}, K, \dots)$$

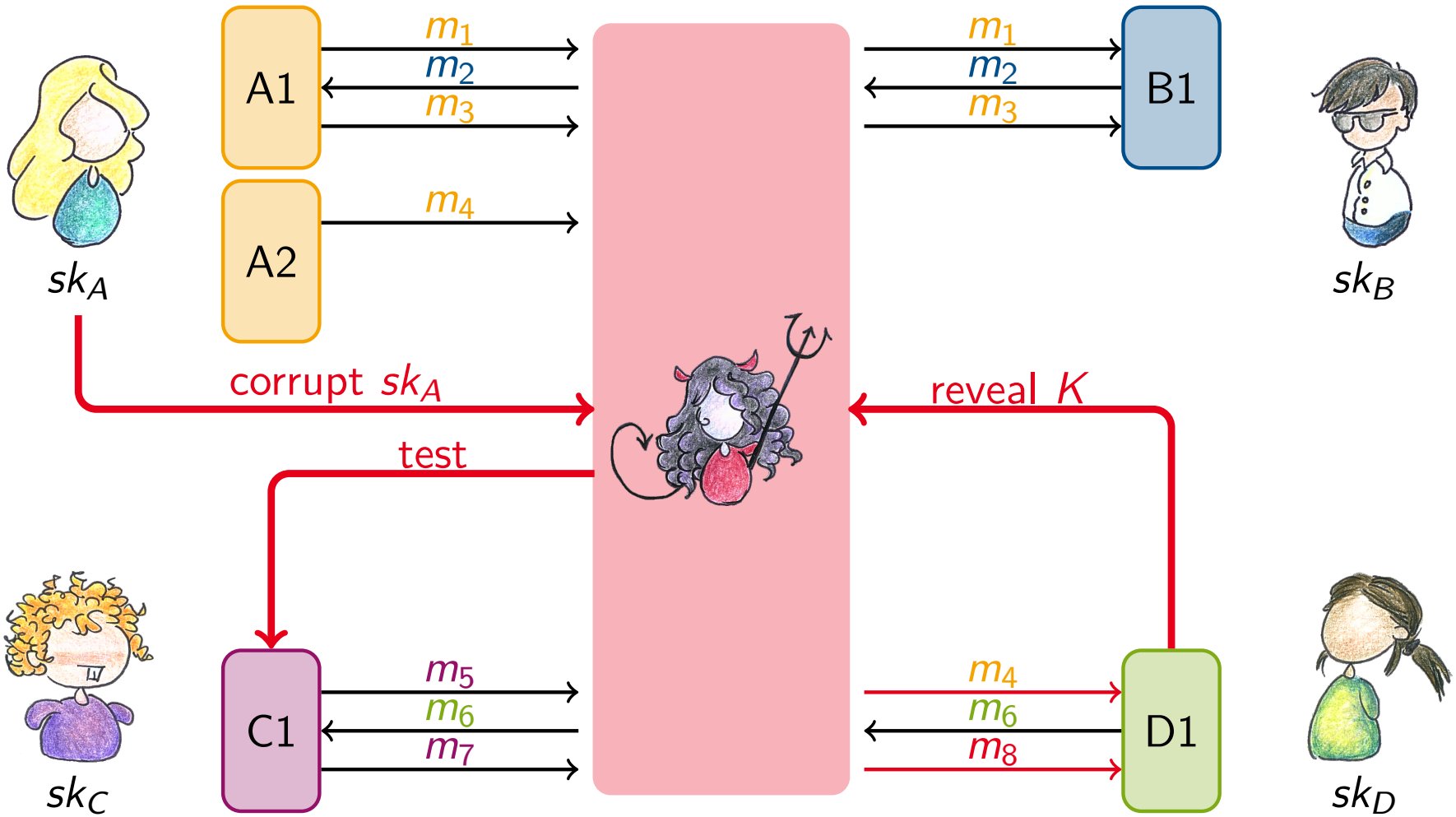


Modeling Key Exchange Security

The communication/security model: a simplified example

ETH zürich

$$\mathrm{KE}(\textit{own id}, \textit{peer id}, sk_{oid}, pk_{pid}, \textit{msg in}, \dots) \mapsto (\textit{msg out}, \textit{status}, K, \dots)$$



The TLS 1.3 Handshake

PSK-(EC)DHE 0-RTT

ETH zürich

(still simplified)

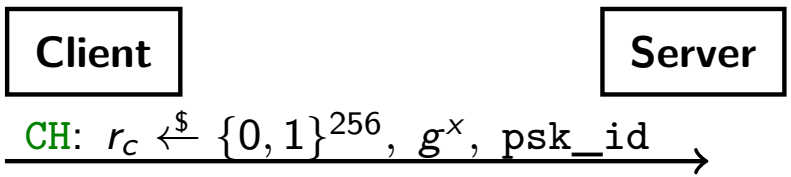
Client **Server**

The TLS 1.3 Handshake

PSK-(EC)DHE 0-RTT

ETH zürich

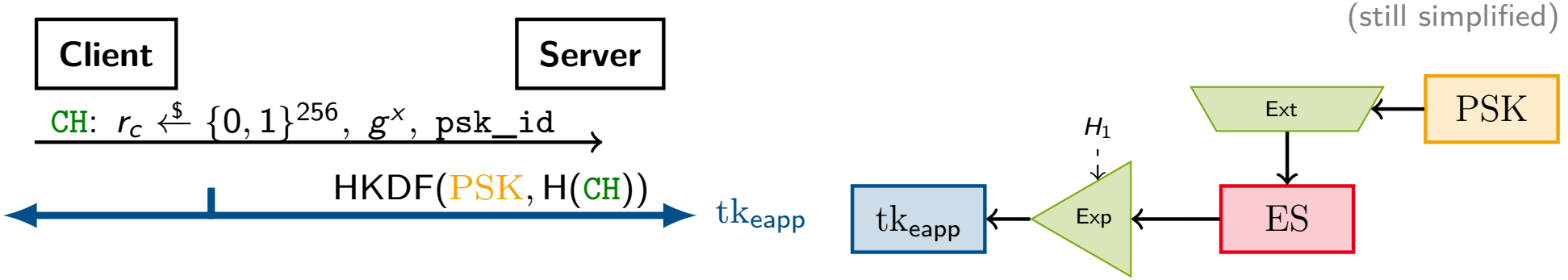
(still simplified)



The TLS 1.3 Handshake

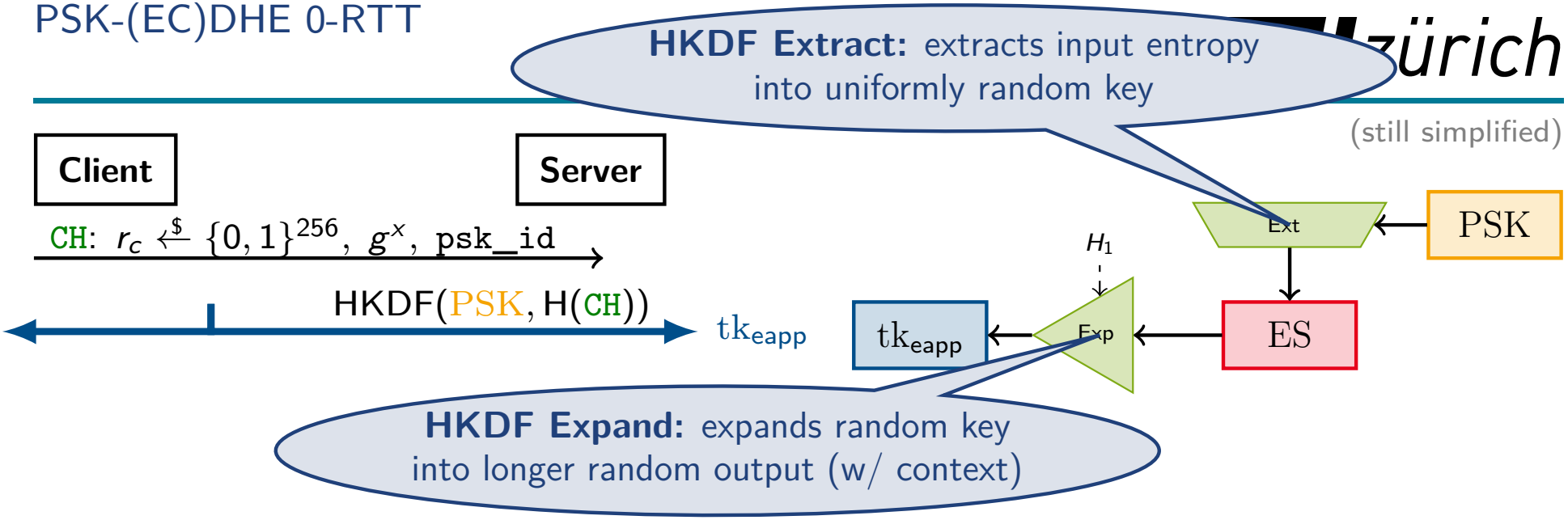
PSK-(EC)DHE 0-RTT

(still simplified)



The TLS 1.3 Handshake

PSK-(EC)DHE 0-RTT



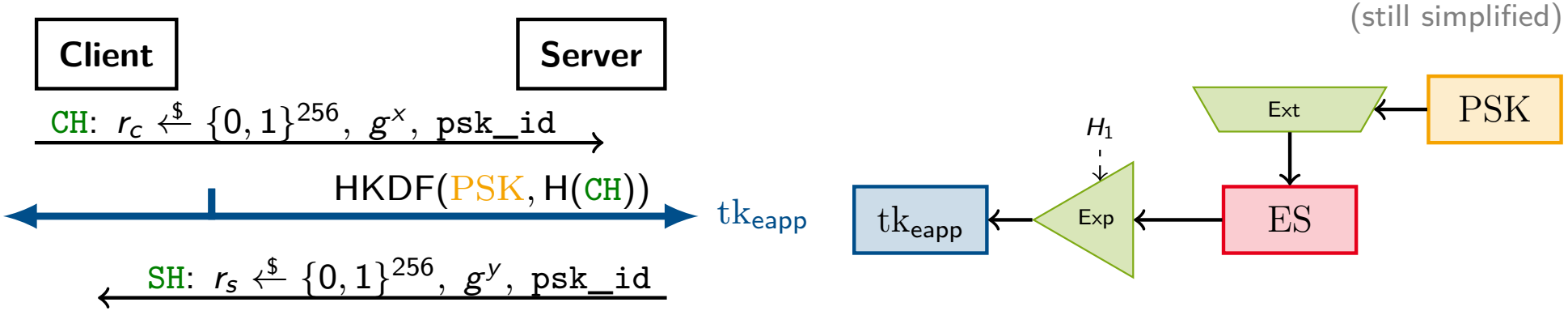
HKDF (recap)

- HKDF.Extract($salt$, $keymaterial$) = HMAC($salt$, $keymaterial$)
- HKDF.Expand(key, $context$, $length$) = $T(1) \| \ldots \| T(N)$
  where $T(0)$ = empty string (zero length)
  $T(i) = \text{HMAC}(key, T(i-1) \| context \| i)$

The TLS 1.3 Handshake

PSK-(EC)DHE 0-RTT

(still simplified)

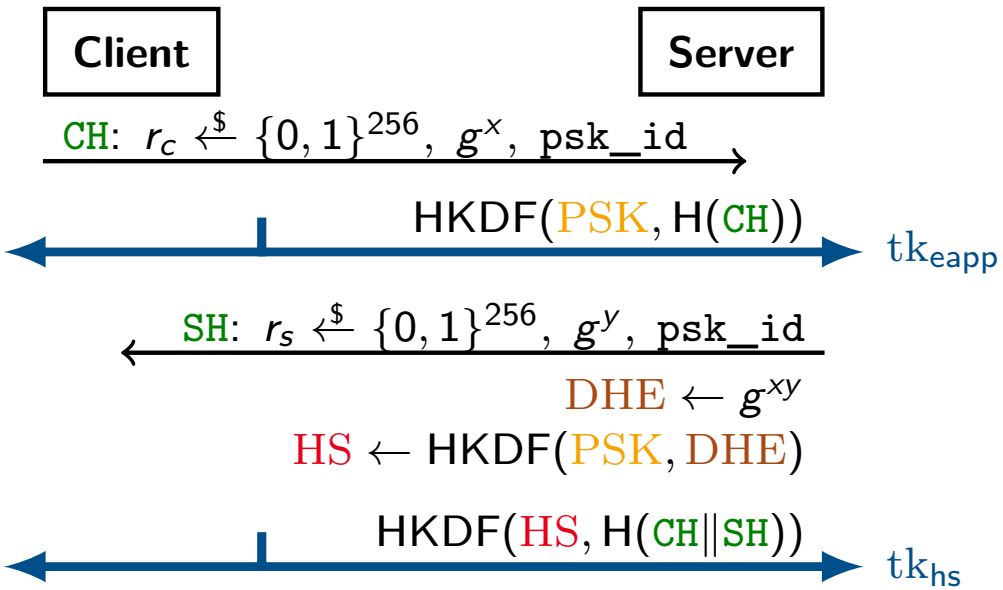


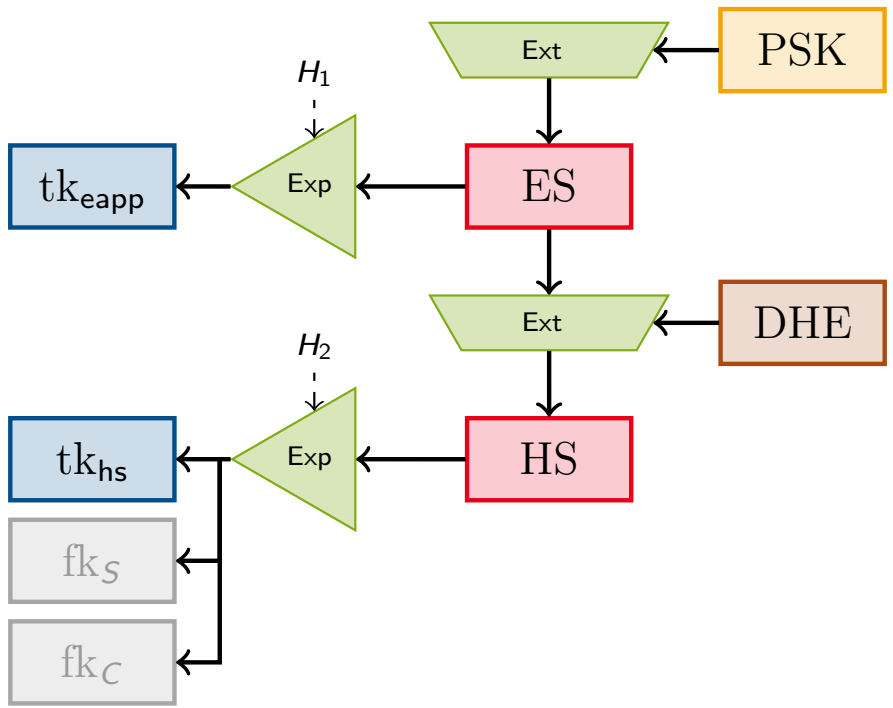
The TLS 1.3 Handshake

PSK-(EC)DHE 0-RTT

ETH zürich

(still simplified)



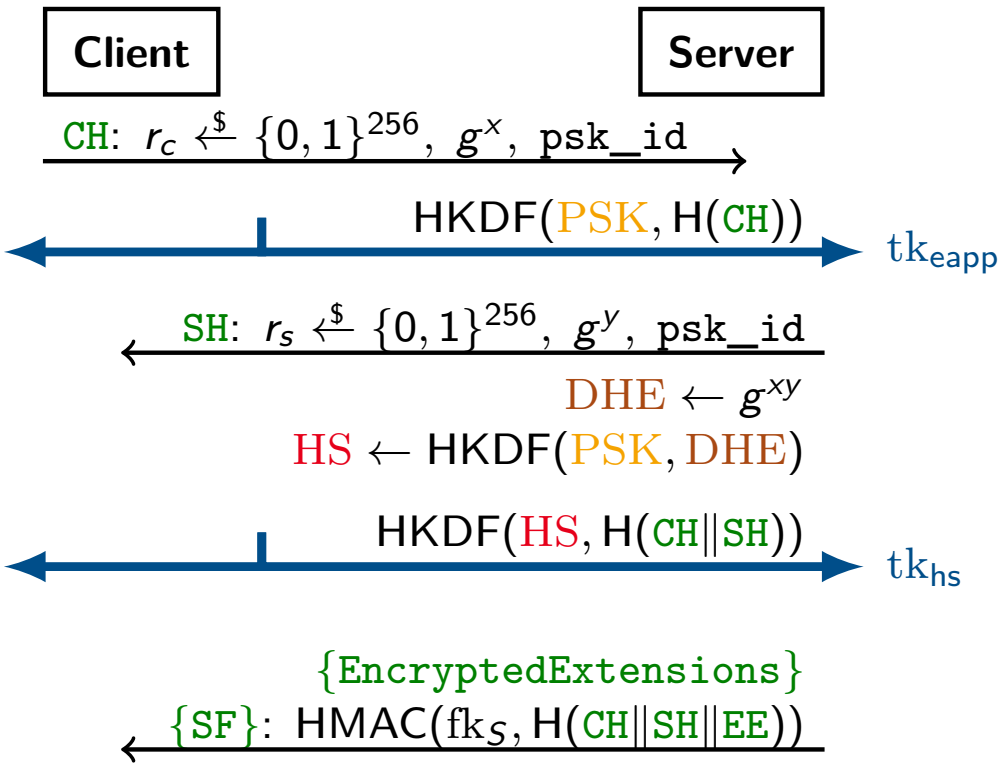


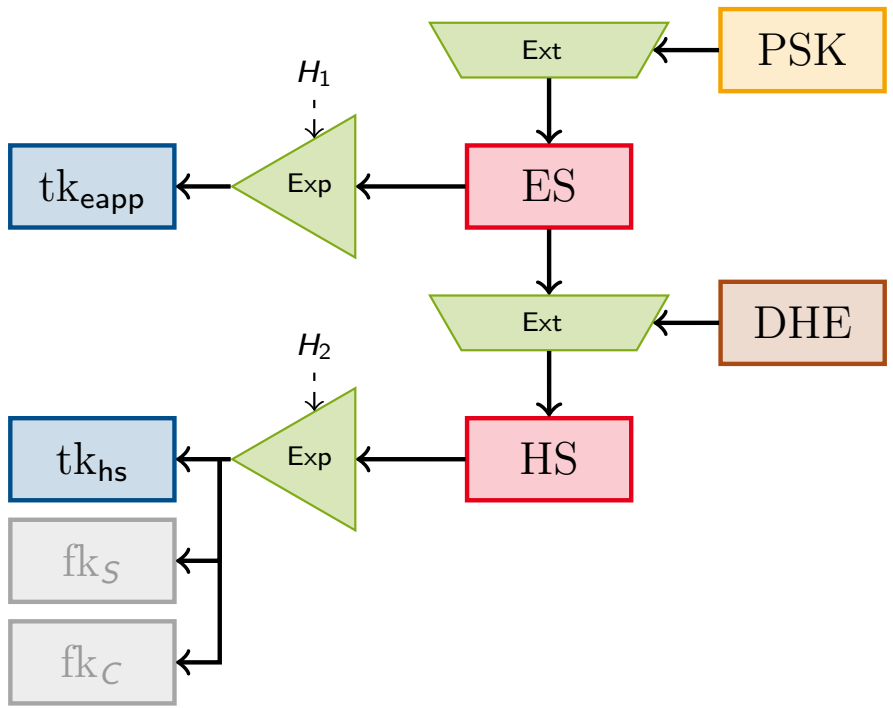
The TLS 1.3 Handshake

PSK-(EC)DHE 0-RTT

ETH zürich

(still simplified)



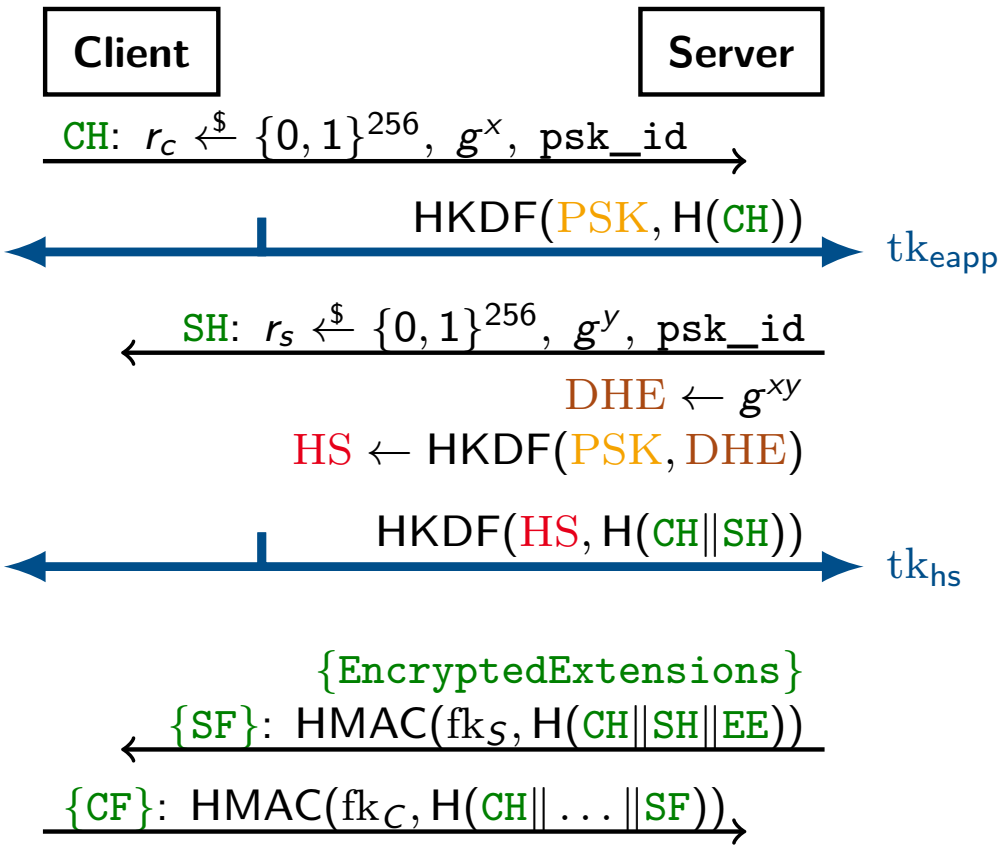


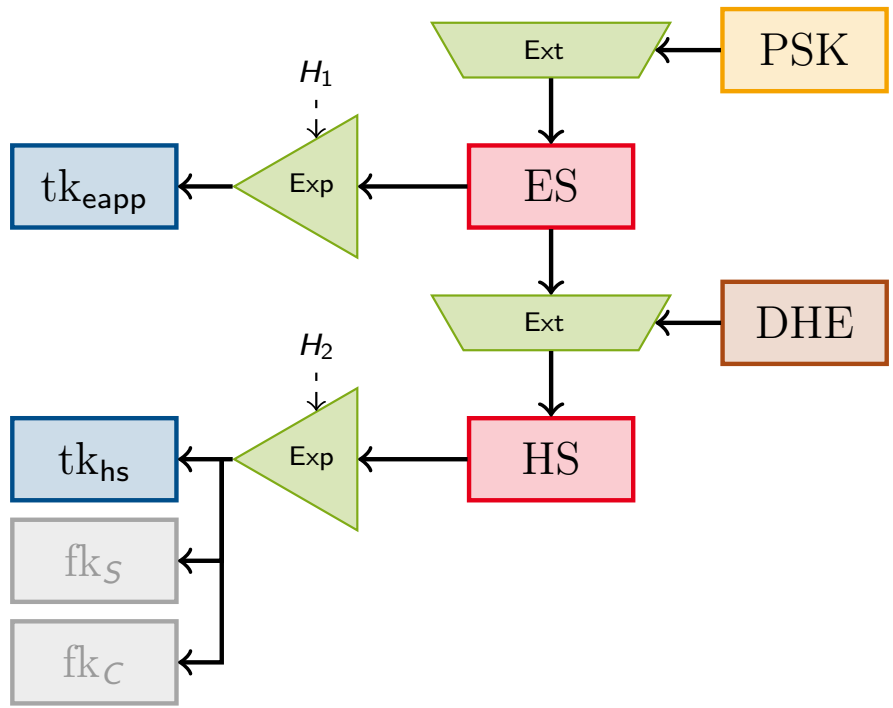
The TLS 1.3 Handshake

PSK-(EC)DHE 0-RTT

ETH zürich

(still simplified)



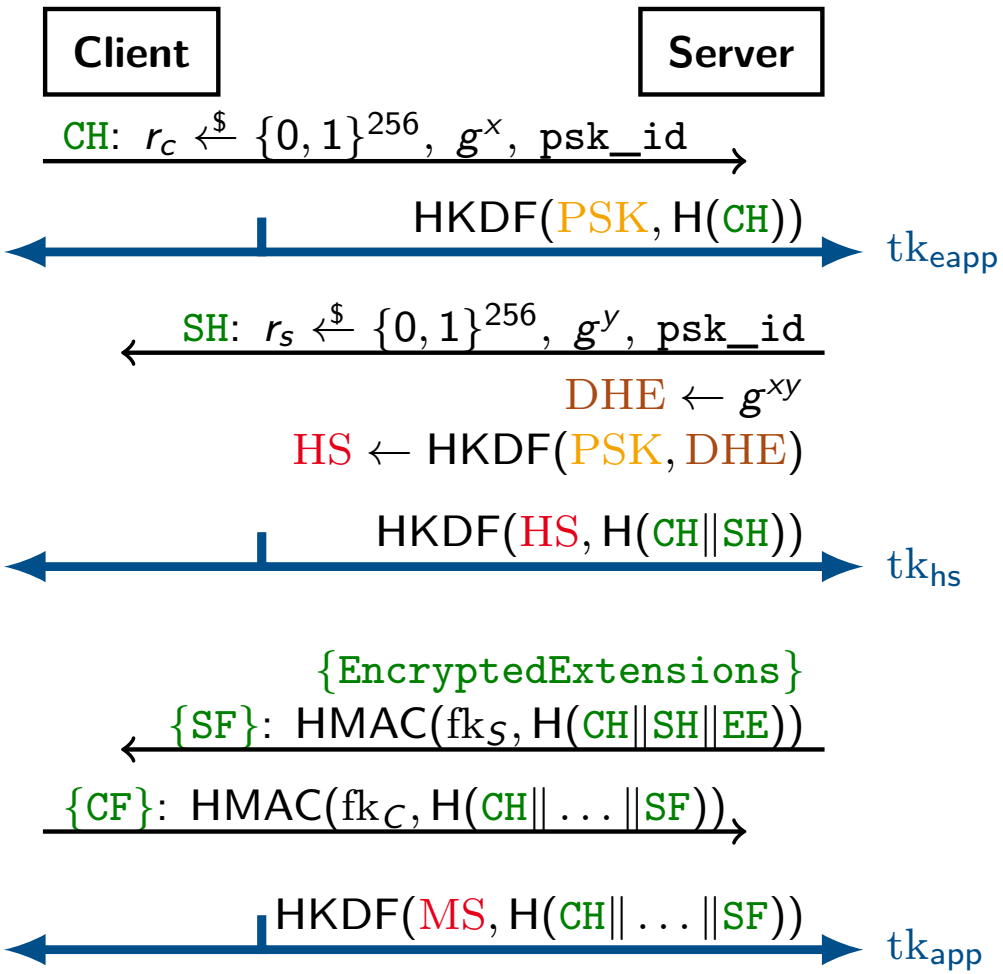


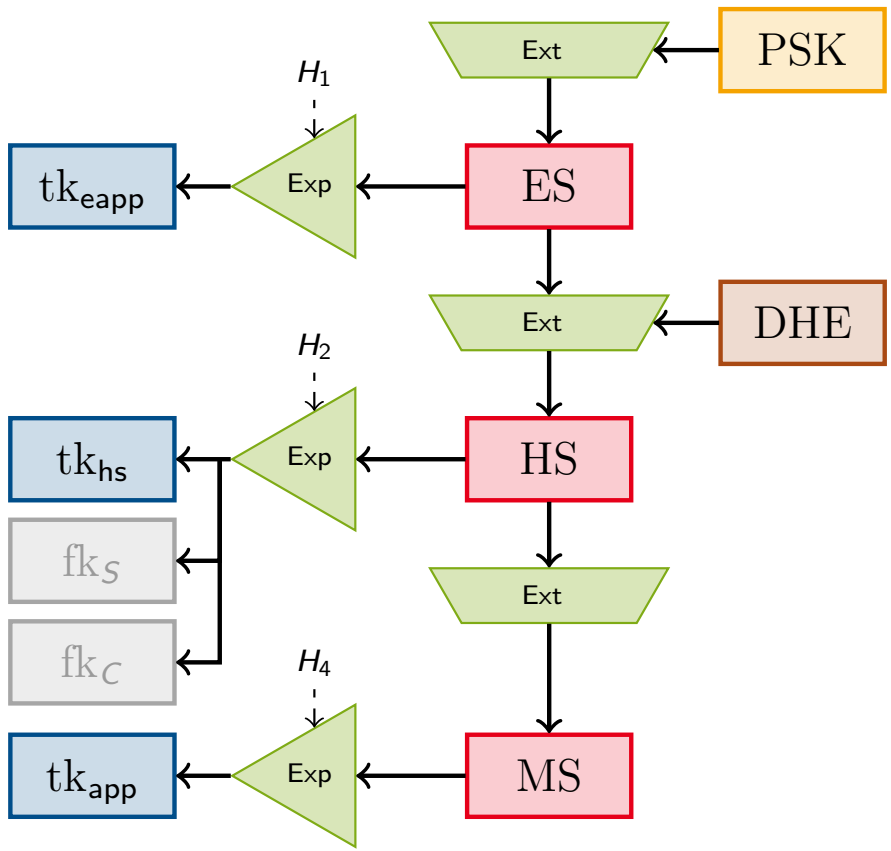
The TLS 1.3 Handshake

PSK-(EC)DHE 0-RTT

ETH zürich

(still simplified)



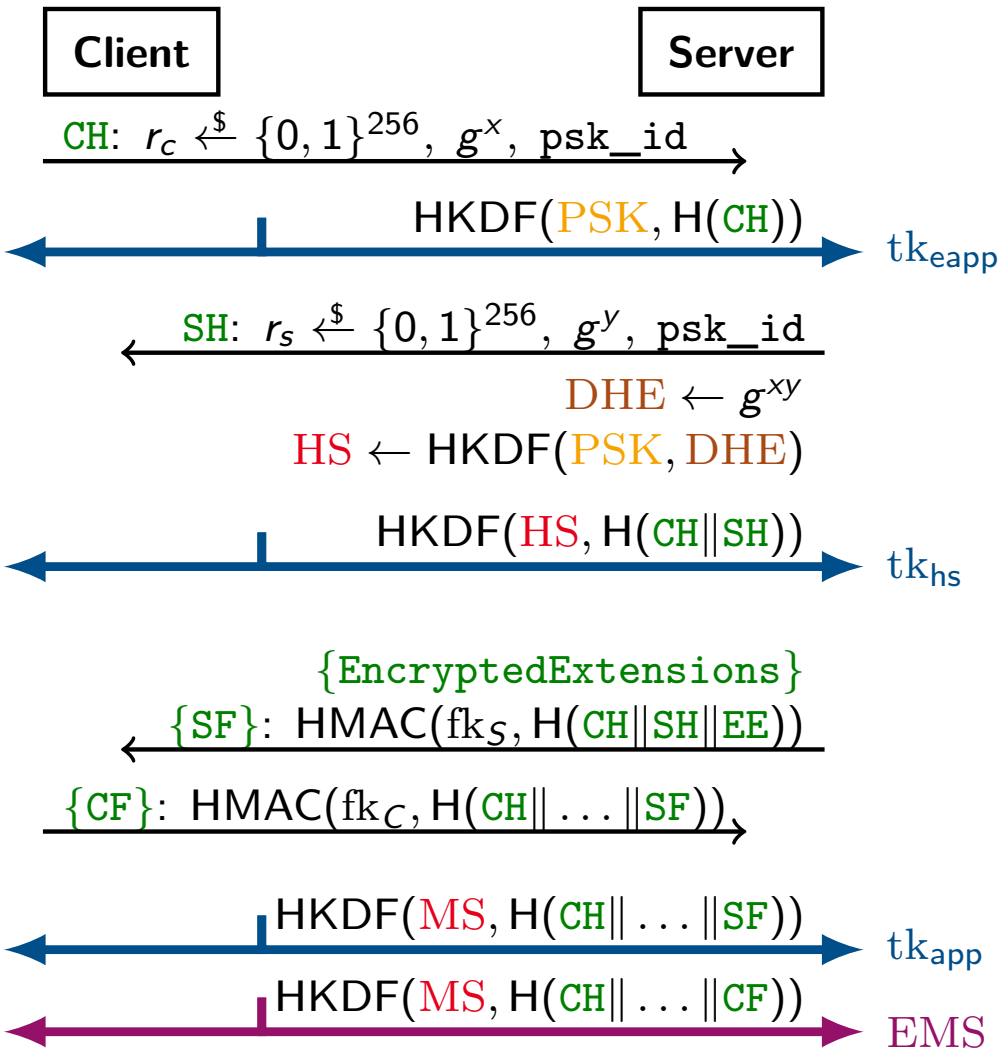


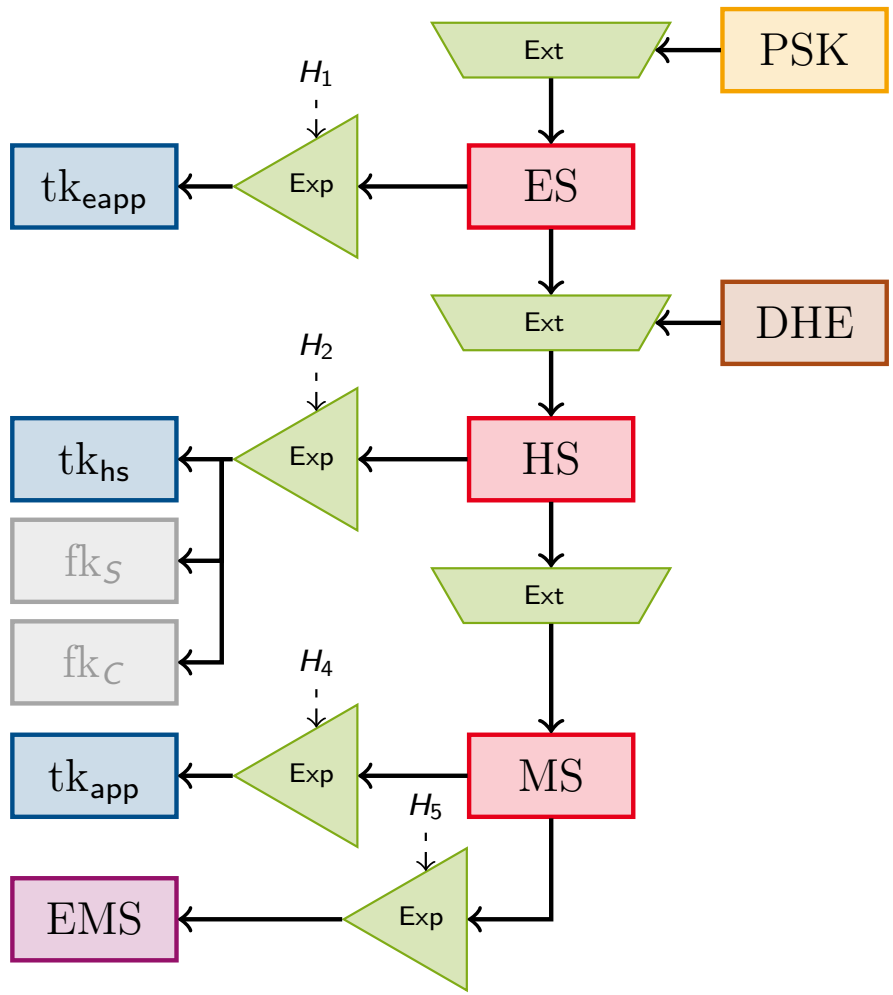
The TLS 1.3 Handshake

PSK-(EC)DHE 0-RTT

ETH zürich

(still simplified)





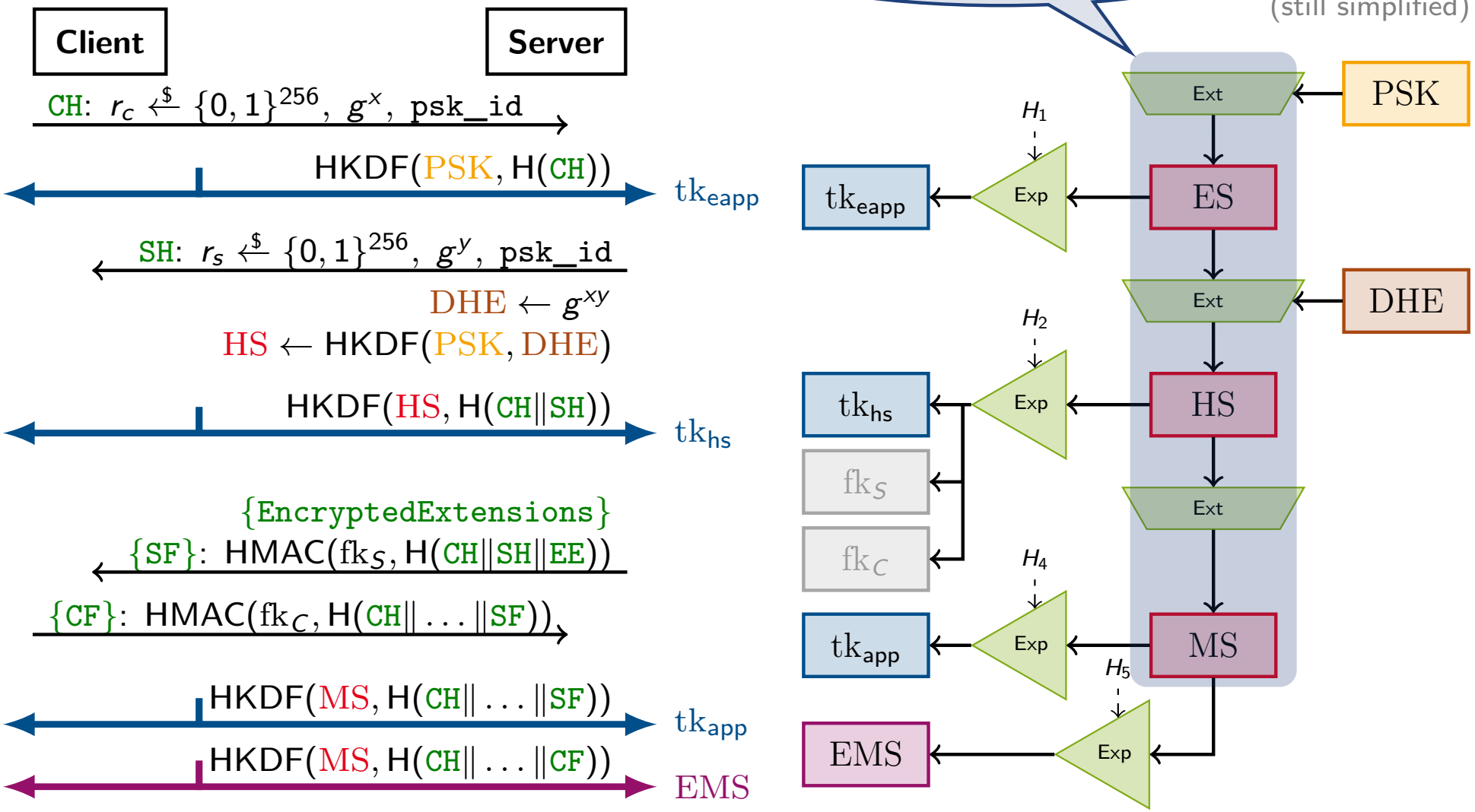
The TLS 1.3 Handshake

PSK-(EC)DHE 0-RTT

zürich

key schedule: core accumulates secret inputs

(still simplified)

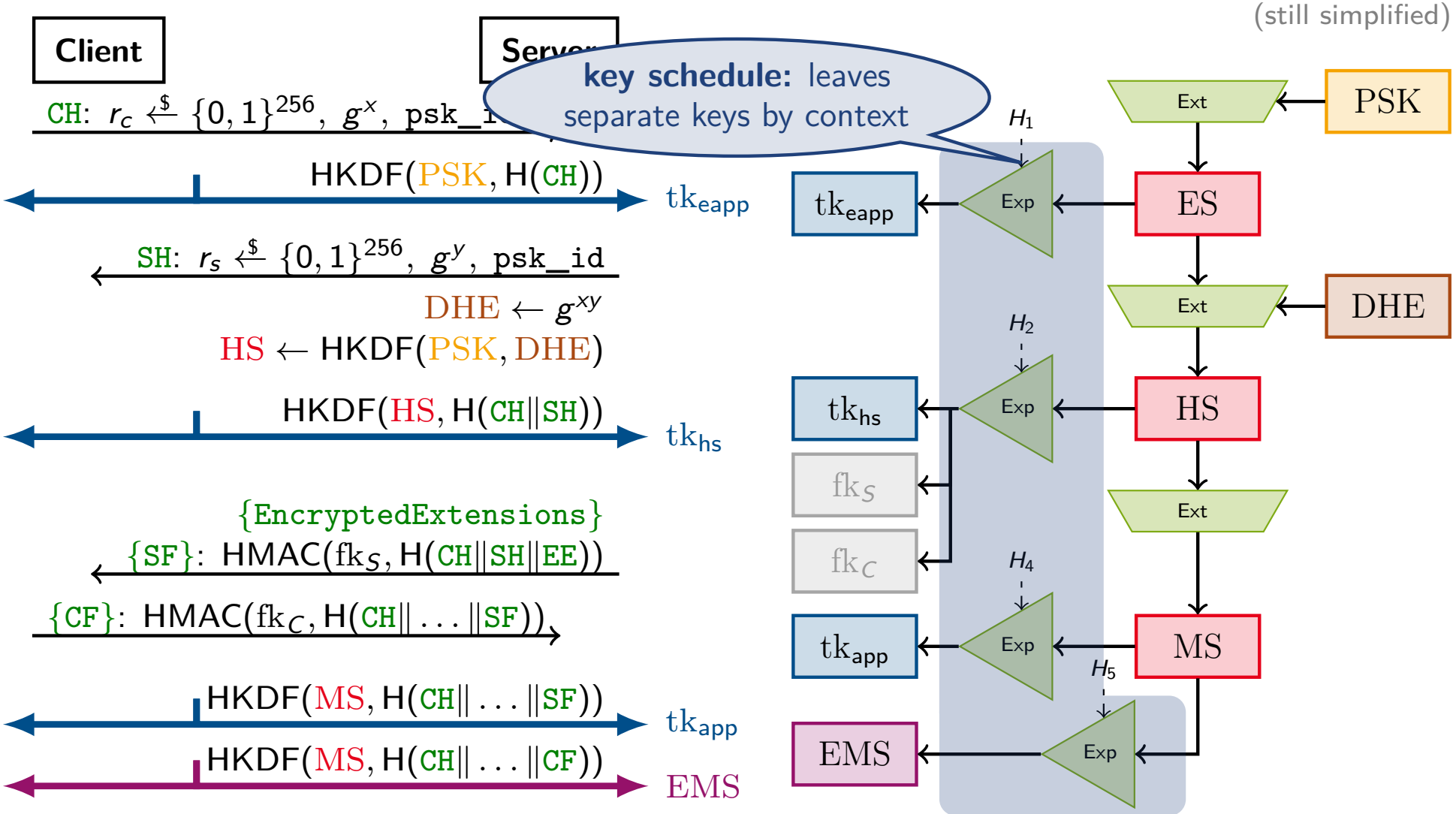


The TLS 1.3 Handshake

PSK-(EC)DHE 0-RTT

ETH zürich

(still simplified)

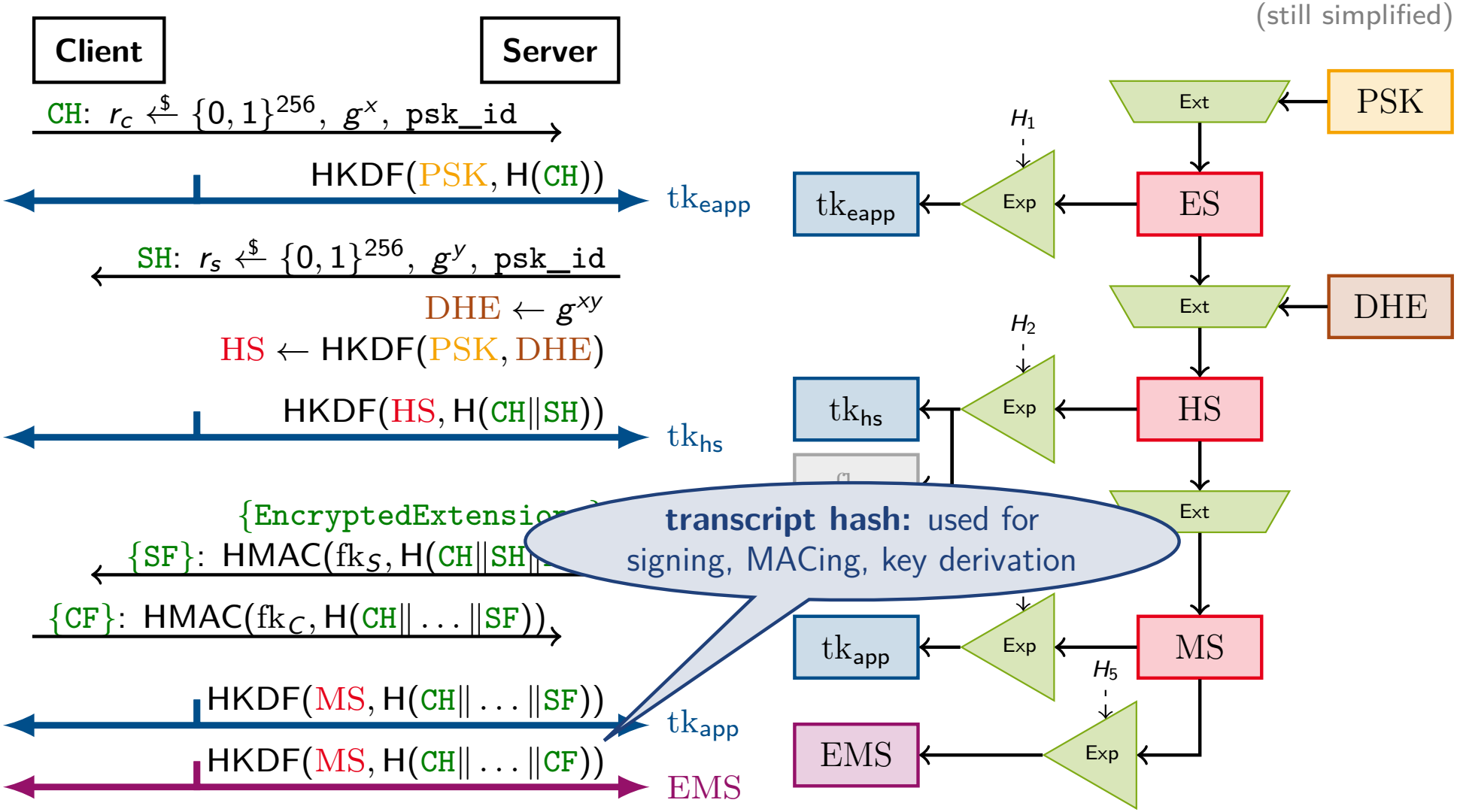


The TLS 1.3 Handshake

PSK-(EC)DHE 0-RTT

ETH zürich

(still simplified)



TLS 1.3 Handshake Security

TLS 1.3 PSK-(EC)DHE 0-RTT as Multi-Stage Key Exchange

ETH zürich

TLS 1.3 Handshake Security

TLS 1.3 PSK-(EC)DHE 0-RTT as Multi-Stage Key Exchange

ETH zürich

Theorem 6.4. *The TLS 1.3 PSK-(EC)DHE 0-RTT handshake is* Multi-Stage*-secure with properties* (M, AUTH, FS, USE, REPLAY).

$$
\begin{aligned}
\mathsf{Adv}^{\text{Multi-Stage},\mathcal{D}}_{\texttt{TLS1.3-PSK-(EC)DHE-0RTT},\mathcal{A}} \leq 8n_s \cdot \Bigg(& \mathsf{Adv}^{\text{COLL}}_{\mathsf{H},\mathcal{B}_1} \\
& + n_p n_s \cdot \Big(\mathsf{Adv}^{\text{dual-PRF-sec}}_{\mathsf{HKDF.Extract},\mathcal{B}_2} + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_3} \\
& \qquad + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Extract},\mathcal{B}_4} + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_5} \\
& \qquad + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_6} + \mathsf{Adv}^{\text{EUF-CMA}}_{\mathsf{HMAC},\mathcal{B}_7} \\
& \qquad + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_8} + \mathsf{Adv}^{\text{EUF-CMA}}_{\mathsf{HMAC},\mathcal{B}_9} \Big) \\
& + n_p n_s \cdot \Big(\mathsf{Adv}^{\text{dual-PRF-sec}}_{\mathsf{HKDF.Extract},\mathcal{B}_{10}} + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_{11}} \Big) \\
& + n_s \cdot \Big(\mathsf{Adv}^{\text{dual-snPRF-ODH}}_{\mathsf{HKDF.Extract},\mathbb{G},\mathcal{B}_{12}} + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_{13}} \\
& \qquad + 2\mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_{14}} + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Extract},\mathcal{B}_{15}} \\
& \qquad + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_{16}} \Big) \Bigg).
\end{aligned}
$$

[Dow+21]

TLS 1.3 Handshake Security

TLS 1.3 PSK-(EC)DHE 0-RTT as Multi-Stage Key Exchange

ETH zürich

The **TLS 1.3 PSK-(EC)DHE 0-RTT** handshake provides

- random-looking secret keys

Theorem 6.4. *The TLS 1.3 PSK-(EC)DHE 0-RTT handshake is* **Multi-Stage-*secure*** *with properties* (M, AUTH, FS, USE, REPLAY).

$$
\begin{aligned}
\mathsf{Adv}^{\text{Multi-Stage},\mathcal{D}}_{\texttt{TLS1.3-PSK-(EC)DHE-0RTT},\mathcal{A}} \leq 8n_s \cdot \Big(& \mathsf{Adv}^{\text{COLL}}_{\mathsf{H},\mathcal{B}_1} \\
& + n_p n_s \cdot \Big(\mathsf{Adv}^{\text{dual-PRF-sec}}_{\mathsf{HKDF.Extract},\mathcal{B}_2} + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_3} \\
& \qquad + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Extract},\mathcal{B}_4} + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_5} \\
& \qquad + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_6} + \mathsf{Adv}^{\text{EUF-CMA}}_{\mathsf{HMAC},\mathcal{B}_7} \\
& \qquad + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_8} + \mathsf{Adv}^{\text{EUF-CMA}}_{\mathsf{HMAC},\mathcal{B}_9} \Big) \\
& + n_p n_s \cdot \Big(\mathsf{Adv}^{\text{dual-PRF-sec}}_{\mathsf{HKDF.Extract},\mathcal{B}_{10}} + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_{11}} \Big) \\
& + n_s \cdot \Big(\mathsf{Adv}^{\text{dual-snPRF-ODH}}_{\mathsf{HKDF.Extract},\mathbb{G},\mathcal{B}_{12}} + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_{13}} \\
& \qquad + 2\mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_{14}} + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Extract},\mathcal{B}_{15}} \\
& \qquad + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_{16}} \Big) \Big).
\end{aligned}
$$

[Dow+21]

TLS 1.3 Handshake Security

TLS 1.3 PSK-(EC)DHE 0-RTT as Multi-Stage Key Exchange

ETH zürich

The **TLS 1.3 PSK-(EC)DHE 0-RTT** handshake provides

- random-looking secret keys
- forward security
  for non–0-RTT keys

Theorem 6.4. *The TLS 1.3 PSK-(EC)DHE 0-RTT handshake is* **Multi-Stage-*secure*** *with properties* (M, AUTH, **FS**, USE, REPLAY).

$$
\begin{aligned}
\mathsf{Adv}^{\text{Multi-Stage},\mathcal{D}}_{\texttt{TLS1.3-PSK-(EC)DHE-0RTT},\mathcal{A}} \leq 8n_s \cdot \Big(& \mathsf{Adv}^{\text{COLL}}_{\mathsf{H},\mathcal{B}_1} \\
& + n_p n_s \cdot \Big(\mathsf{Adv}^{\text{dual-PRF-sec}}_{\mathsf{HKDF.Extract},\mathcal{B}_2} + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_3} \\
& \qquad + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Extract},\mathcal{B}_4} + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_5} \\
& \qquad + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_6} + \mathsf{Adv}^{\text{EUF-CMA}}_{\mathsf{HMAC},\mathcal{B}_7} \\
& \qquad + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_8} + \mathsf{Adv}^{\text{EUF-CMA}}_{\mathsf{HMAC},\mathcal{B}_9} \Big) \\
& + n_p n_s \cdot \Big(\mathsf{Adv}^{\text{dual-PRF-sec}}_{\mathsf{HKDF.Extract},\mathcal{B}_{10}} + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_{11}} \Big) \\
& + n_s \cdot \Big(\mathsf{Adv}^{\text{dual-snPRF-ODH}}_{\mathsf{HKDF.Extract},\mathbb{G},\mathcal{B}_{12}} + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_{13}} \\
& \qquad + 2\mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_{14}} + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Extract},\mathcal{B}_{15}} \\
& \qquad + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_{16}} \Big) \Big).
\end{aligned}
$$

[Dow+21]

TLS 1.3 Handshake Security

TLS 1.3 PSK-(EC)DHE 0-RTT as Multi-Stage Key Exchange

ETH zürich

The **TLS 1.3 PSK-(EC)DHE 0-RTT** handshake provides

- random-looking secret keys
- forward security
  for non–0-RTT keys
- mutual authentication wrt. PSK

Theorem 6.4. *The TLS 1.3 PSK-(EC)DHE 0-RTT handshake is* **Multi-Stage-*secure*** *with properties* (M, **AUTH**, **FS**, USE, REPLAY).

$$
\begin{aligned}
\mathsf{Adv}^{\text{Multi-Stage},\mathcal{D}}_{\texttt{TLS1.3-PSK-(EC)DHE-0RTT},\mathcal{A}} \leq 8n_s \cdot \Big(& \mathsf{Adv}^{\text{COLL}}_{\mathsf{H},\mathcal{B}_1} \\
& + n_p n_s \cdot \Big(\mathsf{Adv}^{\text{dual-PRF-sec}}_{\mathsf{HKDF.Extract},\mathcal{B}_2} + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_3} \\
& \qquad + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Extract},\mathcal{B}_4} + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_5} \\
& \qquad + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_6} + \mathsf{Adv}^{\text{EUF-CMA}}_{\mathsf{HMAC},\mathcal{B}_7} \\
& \qquad + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_8} + \mathsf{Adv}^{\text{EUF-CMA}}_{\mathsf{HMAC},\mathcal{B}_9} \Big) \\
& + n_p n_s \cdot \Big(\mathsf{Adv}^{\text{dual-PRF-sec}}_{\mathsf{HKDF.Extract},\mathcal{B}_{10}} + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_{11}} \Big) \\
& + n_s \cdot \Big(\mathsf{Adv}^{\text{dual-snPRF-ODH}}_{\mathsf{HKDF.Extract},\mathbb{G},\mathcal{B}_{12}} + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_{13}} \\
& \qquad + 2\mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_{14}} + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Extract},\mathcal{B}_{15}} \\
& \qquad + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_{16}} \Big) \Big).
\end{aligned}
$$

[Dow+21]

TLS 1.3 Handshake Security

TLS 1.3 PSK-(EC)DHE 0-RTT as Multi-Stage Key Exchange

ETH zürich

The **TLS 1.3 PSK-(EC)DHE 0-RTT** handshake provides

- random-looking secret keys
- forward security
  for non–0-RTT keys
- mutual authentication wrt. PSK
- key independence

Theorem 6.4. *The TLS 1.3 PSK-(EC)DHE 0-RTT handshake is* **Multi-Stage-*secure*** *with properties* (M, **AUTH**, **FS**, USE, REPLAY).

$$
\begin{aligned}
\mathsf{Adv}^{\text{Multi-Stage},\mathcal{D}}_{\texttt{TLS1.3-PSK-(EC)DHE-0RTT},\mathcal{A}} \leq 8n_s \cdot \Big(& \mathsf{Adv}^{\text{COLL}}_{\mathsf{H},\mathcal{B}_1} \\
& + n_p n_s \cdot \Big(\mathsf{Adv}^{\text{dual-PRF-sec}}_{\mathsf{HKDF.Extract},\mathcal{B}_2} + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_3} \\
& \qquad + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Extract},\mathcal{B}_4} + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_5} \\
& \qquad + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_6} + \mathsf{Adv}^{\text{EUF-CMA}}_{\mathsf{HMAC},\mathcal{B}_7} \\
& \qquad + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_8} + \mathsf{Adv}^{\text{EUF-CMA}}_{\mathsf{HMAC},\mathcal{B}_9} \Big) \\
& + n_p n_s \cdot \Big(\mathsf{Adv}^{\text{dual-PRF-sec}}_{\mathsf{HKDF.Extract},\mathcal{B}_{10}} + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_{11}} \Big) \\
& + n_s \cdot \Big(\mathsf{Adv}^{\text{dual-snPRF-ODH}}_{\mathsf{HKDF.Extract},\mathbb{G},\mathcal{B}_{12}} + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_{13}} \\
& \qquad + 2\mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_{14}} + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Extract},\mathcal{B}_{15}} \\
& \qquad + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_{16}} \Big) \Big).
\end{aligned}
$$

[Dow+21]

TLS 1.3 Handshake Security

TLS 1.3 PSK-(EC)DHE 0-RTT as Multi-Stage Key Exchange

ETH zürich

The **TLS 1.3 PSK-(EC)DHE 0-RTT** handshake provides

- random-looking secret keys
- forward security for non–0-RTT keys
- mutual authentication wrt. PSK
- key independence
- replayable 0-RTT keys

Theorem 6.4. *The TLS 1.3 PSK-(EC)DHE 0-RTT handshake is* **Multi-Stage-*secure*** *with properties* (M, **AUTH**, **FS**, USE, **REPLAY**).

$$
\begin{aligned}
\mathsf{Adv}^{\text{Multi-Stage},\mathcal{D}}_{\texttt{TLS1.3-PSK-(EC)DHE-0RTT},\mathcal{A}} \leq 8n_s \cdot \Big(& \mathsf{Adv}^{\text{COLL}}_{\text{H},\mathcal{B}_1} \\
& + n_p n_s \cdot \Big(\mathsf{Adv}^{\text{dual-PRF-sec}}_{\text{HKDF.Extract},\mathcal{B}_2} + \mathsf{Adv}^{\text{PRF-sec}}_{\text{HKDF.Expand},\mathcal{B}_3} \\
& \quad + \mathsf{Adv}^{\text{PRF-sec}}_{\text{HKDF.Extract},\mathcal{B}_4} + \mathsf{Adv}^{\text{PRF-sec}}_{\text{HKDF.Expand},\mathcal{B}_5} \\
& \quad + \mathsf{Adv}^{\text{PRF-sec}}_{\text{HKDF.Expand},\mathcal{B}_6} + \mathsf{Adv}^{\text{EUF-CMA}}_{\text{HMAC},\mathcal{B}_7} \\
& \quad + \mathsf{Adv}^{\text{PRF-sec}}_{\text{HKDF.Expand},\mathcal{B}_8} + \mathsf{Adv}^{\text{EUF-CMA}}_{\text{HMAC},\mathcal{B}_9} \Big) \\
& + n_p n_s \cdot \Big(\mathsf{Adv}^{\text{dual-PRF-sec}}_{\text{HKDF.Extract},\mathcal{B}_{10}} + \mathsf{Adv}^{\text{PRF-sec}}_{\text{HKDF.Expand},\mathcal{B}_{11}} \Big) \\
& + n_s \cdot \Big(\mathsf{Adv}^{\text{dual-snPRF-ODH}}_{\text{HKDF.Extract},\mathbb{G},\mathcal{B}_{12}} + \mathsf{Adv}^{\text{PRF-sec}}_{\text{HKDF.Expand},\mathcal{B}_{13}} \\
& \quad + 2\mathsf{Adv}^{\text{PRF-sec}}_{\text{HKDF.Expand},\mathcal{B}_{14}} + \mathsf{Adv}^{\text{PRF-sec}}_{\text{HKDF.Extract},\mathcal{B}_{15}} \\
& \quad + \mathsf{Adv}^{\text{PRF-sec}}_{\text{HKDF.Expand},\mathcal{B}_{16}} \Big) \Big).
\end{aligned}
$$

[Dow+21]

TLS 1.3 Handshake Security

TLS 1.3 PSK-(EC)DHE 0-RTT as Multi-Stage Key Exchange

ETH zürich

The **TLS 1.3 PSK-(EC)DHE 0-RTT** handshake provides

- random-looking secret keys
- forward security for non–0-RTT keys
- mutual authentication wrt. PSK
- key independence
- replayable 0-RTT keys

assuming . . .

Theorem 6.4. *The TLS 1.3 PSK-(EC)DHE 0-RTT handshake is* **Multi-Stage-*secure*** *with properties* (M, **AUTH**, **FS**, USE, **REPLAY**).

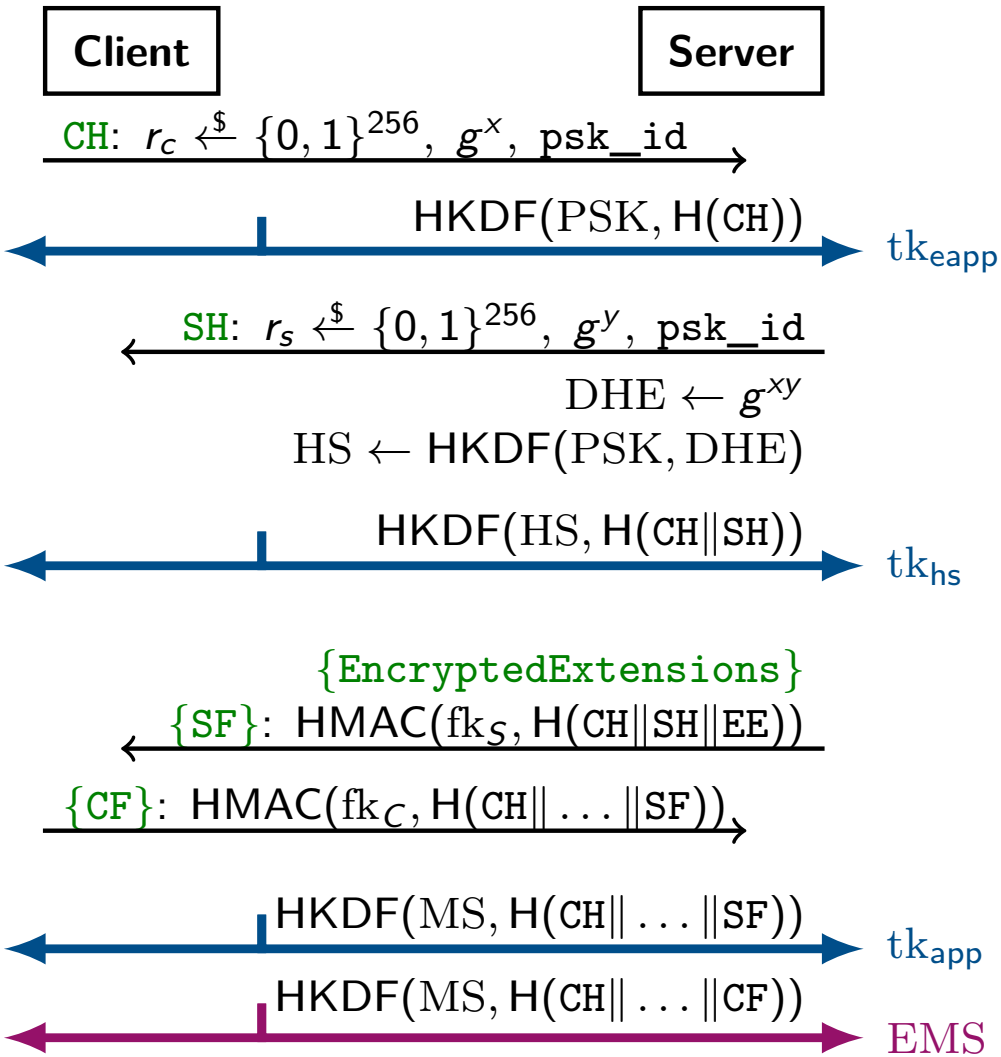
$$
\begin{aligned}
\mathsf{Adv}^{\text{Multi-Stage},\mathcal{D}}_{\texttt{TLS1.3-PSK-(EC)DHE-0RTT},\mathcal{A}} \leq\ & 8n_s \cdot \Big(\mathsf{Adv}^{\text{COLL}}_{\mathsf{H},\mathcal{B}_1} \\
& + n_p n_s \cdot \Big(\mathsf{Adv}^{\text{dual-PRF-sec}}_{\mathsf{HKDF.Extract},\mathcal{B}_2} + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_3} \\
& \qquad + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Extract},\mathcal{B}_4} + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_5} \\
& \qquad + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_6} + \mathsf{Adv}^{\text{EUF-CMA}}_{\mathsf{HMAC},\mathcal{B}_7} \\
& \qquad + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_8} + \mathsf{Adv}^{\text{EUF-CMA}}_{\mathsf{HMAC},\mathcal{B}_9} \Big) \\
& + n_p n_s \cdot \Big(\mathsf{Adv}^{\text{dual-PRF-sec}}_{\mathsf{HKDF.Extract},\mathcal{B}_{10}} + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_{11}} \Big) \\
& + n_s \cdot \Big(\mathsf{Adv}^{\text{dual-snPRF-ODH}}_{\mathsf{HKDF.Extract},\mathbb{G},\mathcal{B}_{12}} + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_{13}} \\
& \qquad + 2\mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_{14}} + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Extract},\mathcal{B}_{15}} \\
& \qquad + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_{16}} \Big) \Big).
\end{aligned}
$$

[Dow+21]

TLS 1.3 Handshake Security

TLS 1.3 PSK-(EC)DHE 0-RTT as Multi-Stage Key Exchange

ETH zürich



Theorem 6.4. *The TLS 1.3 PSK-(EC)DHE 0-RTT handshake is* **Multi-Stage-*secure*** *with properties* (M, **AUTH**, **FS**, USE, **REPLAY**).

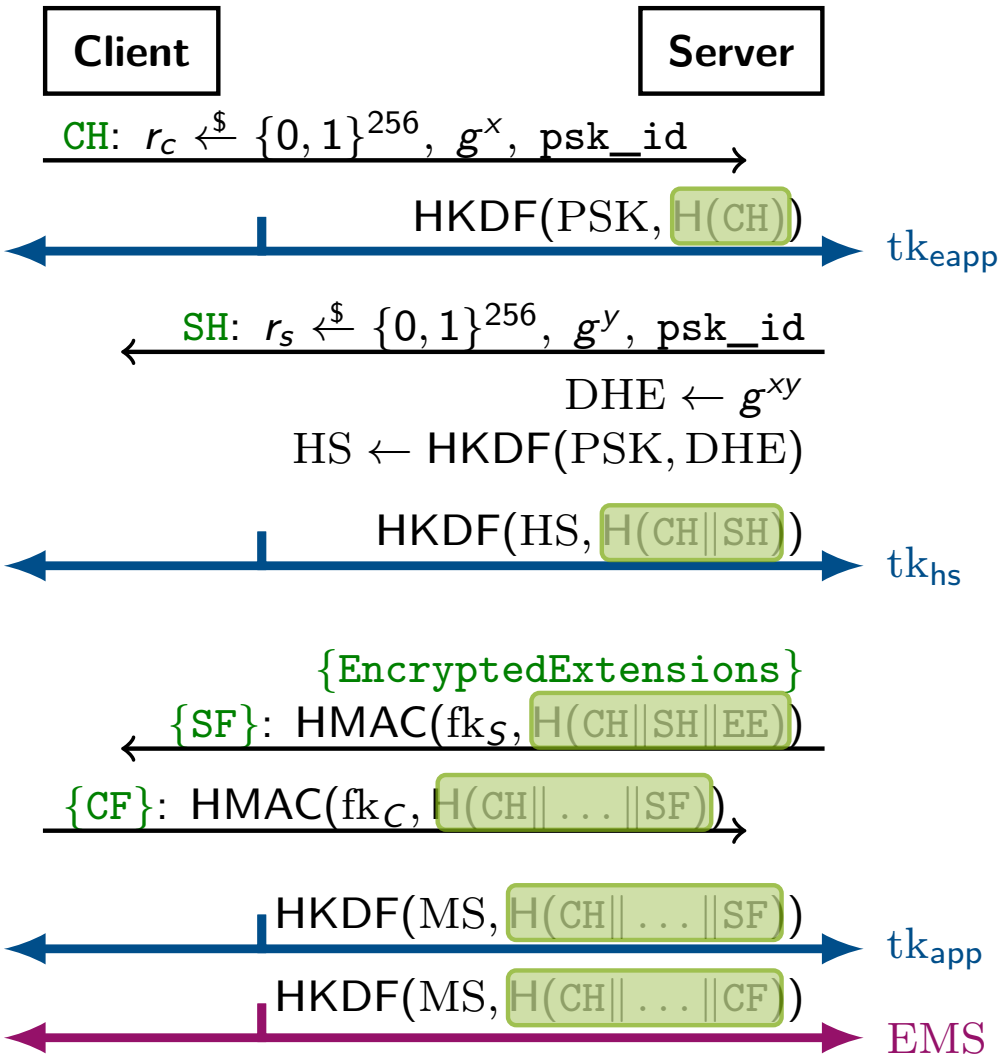
$$
\begin{aligned}
\mathsf{Adv}^{\text{Multi-Stage},\mathcal{D}}_{\text{TLS1.3-PSK-(EC)DHE-0RTT},\mathcal{A}} \leq\ & 8n_s \cdot \Big(\mathsf{Adv}^{\text{COLL}}_{\mathsf{H},\mathcal{B}_1} \\
& + n_p n_s \cdot \Big(\mathsf{Adv}^{\text{dual-PRF-sec}}_{\mathsf{HKDF.Extract},\mathcal{B}_2} + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_3} \\
& \quad + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Extract},\mathcal{B}_4} + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_5} \\
& \quad + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_6} + \mathsf{Adv}^{\text{EUF-CMA}}_{\mathsf{HMAC},\mathcal{B}_7} \\
& \quad + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_8} + \mathsf{Adv}^{\text{EUF-CMA}}_{\mathsf{HMAC},\mathcal{B}_9} \Big) \\
& + n_p n_s \cdot \Big(\mathsf{Adv}^{\text{dual-PRF-sec}}_{\mathsf{HKDF.Extract},\mathcal{B}_{10}} + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_{11}} \Big) \\
& + n_s \cdot \Big(\mathsf{Adv}^{\text{dual-snPRF-ODH}}_{\mathsf{HKDF.Extract},\mathbb{G},\mathcal{B}_{12}} + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_{13}} \\
& \quad + 2\mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_{14}} + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Extract},\mathcal{B}_{15}} \\
& \quad + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_{16}} \Big) \Big).
\end{aligned}
$$

[Dow+21]

TLS 1.3 Handshake Security

TLS 1.3 PSK-(EC)DHE 0-RTT as Multi-Stage Key Exchange

ETH zürich



Theorem 6.4. *The TLS 1.3 PSK-(EC)DHE 0-RTT handshake is* **Multi-Stage-*secure*** *with properties* (M, **AUTH**, **FS**, USE, **REPLAY**).

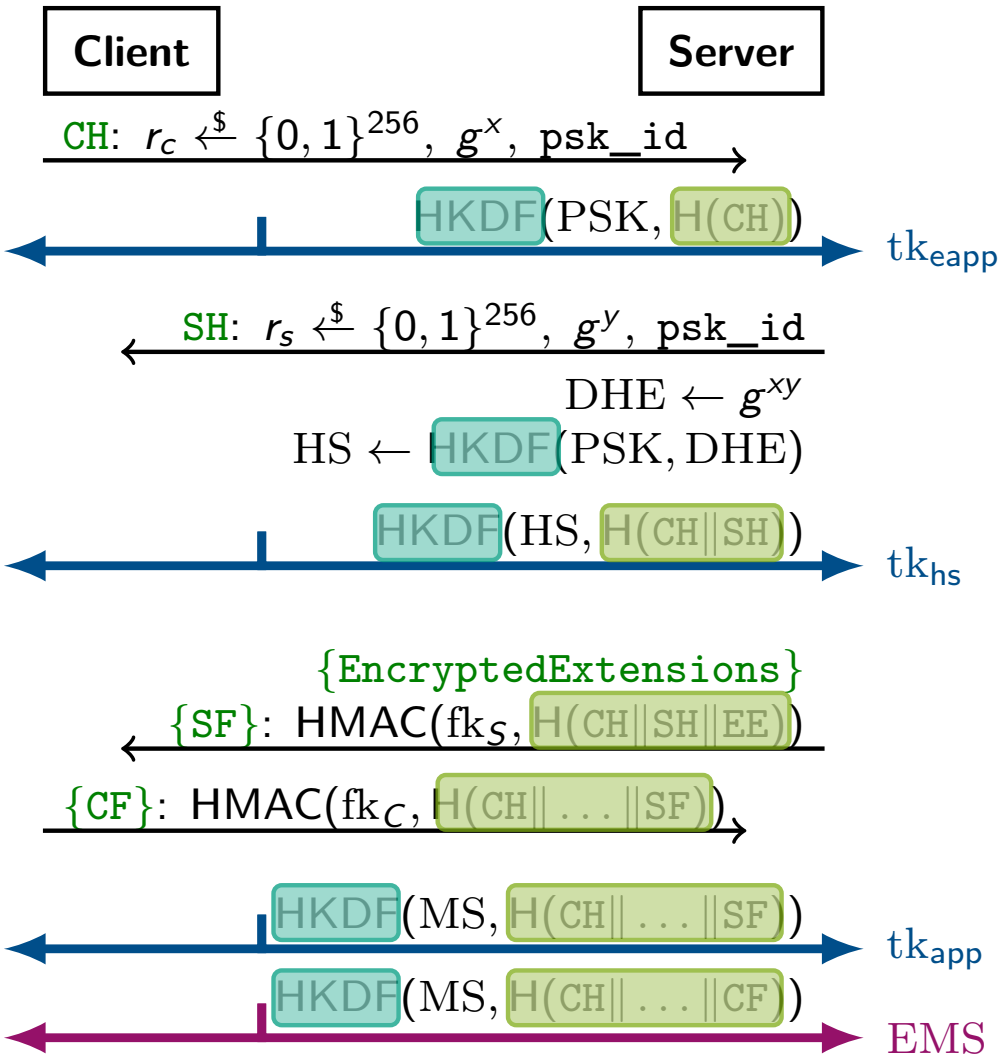
$$
\begin{aligned}
\mathsf{Adv}^{\text{Multi-Stage},\mathcal{D}}_{\text{TLS1.3-PSK-(EC)DHE-0RTT},\mathcal{A}} \leq\ & 8n_s \cdot \Big(\mathsf{Adv}^{\mathsf{COLL}}_{\mathsf{H},\mathcal{B}_1} \\
& + n_p n_s \cdot \Big(\mathsf{Adv}^{\text{dual-PRF-sec}}_{\mathsf{HKDF.Extract},\mathcal{B}_2} + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_3} \\
& \quad + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Extract},\mathcal{B}_4} + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_5} \\
& \quad + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_6} + \mathsf{Adv}^{\text{EUF-CMA}}_{\mathsf{HMAC},\mathcal{B}_7} \\
& \quad + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_8} + \mathsf{Adv}^{\text{EUF-CMA}}_{\mathsf{HMAC},\mathcal{B}_9} \Big) \\
& + n_p n_s \cdot \Big(\mathsf{Adv}^{\text{dual-PRF-sec}}_{\mathsf{HKDF.Extract},\mathcal{B}_{10}} + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_{11}} \Big) \\
& + n_s \cdot \Big(\mathsf{Adv}^{\text{dual-snPRF-ODH}}_{\mathsf{HKDF.Extract},\mathbb{G},\mathcal{B}_{12}} + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_{13}} \\
& \quad + 2\mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_{14}} + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Extract},\mathcal{B}_{15}} \\
& \quad + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_{16}} \Big) \Big).
\end{aligned}
$$

[Dow+21]

TLS 1.3 Handshake Security

TLS 1.3 PSK-(EC)DHE 0-RTT as Multi-Stage Key Exchange

ETH zürich



Theorem 6.4. *The TLS 1.3 PSK-(EC)DHE 0-RTT handshake is* **Multi-Stage-*secure*** *with properties* (M, **AUTH**, **FS**, USE, **REPLAY**).

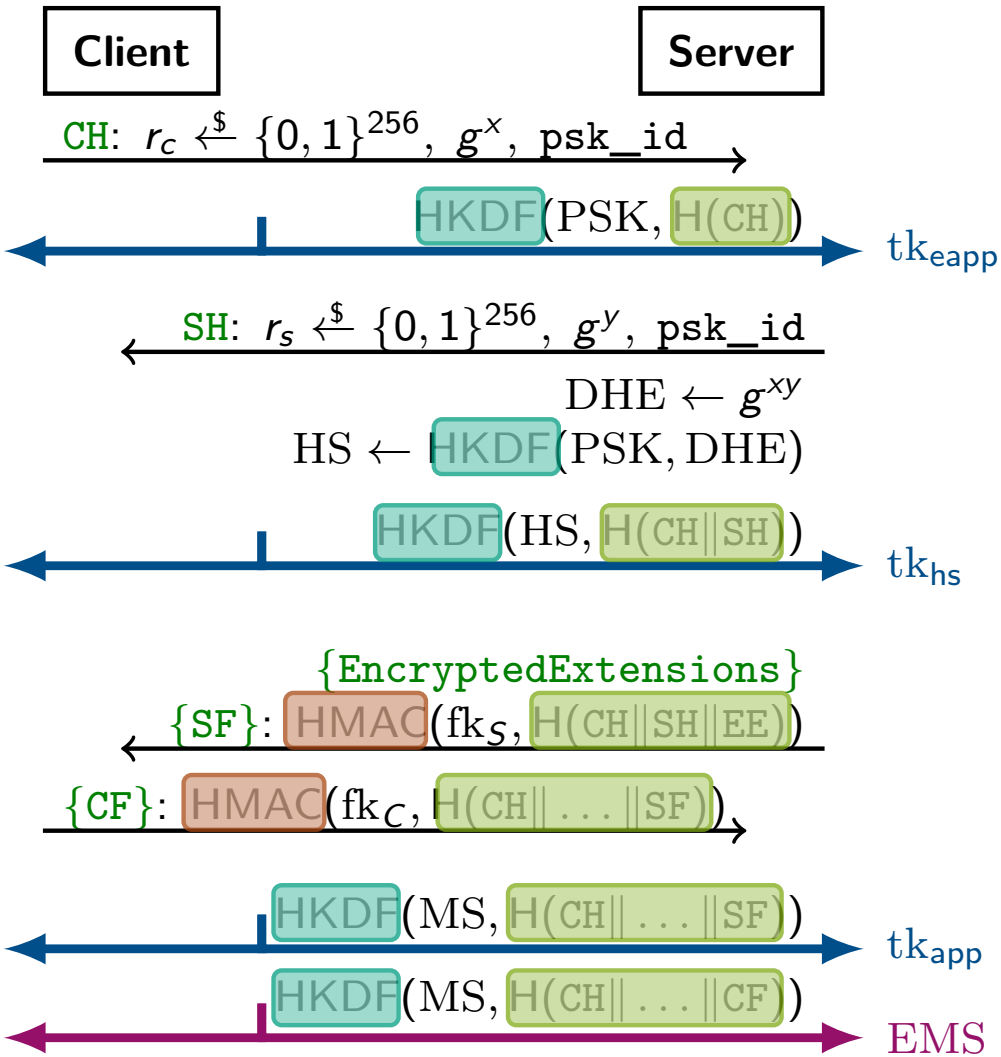
$$
\begin{aligned}
\mathsf{Adv}^{\text{Multi-Stage},\mathcal{D}}_{\text{TLS1.3-PSK-(EC)DHE-0RTT},\mathcal{A}} \leq\ & 8n_s \cdot \Big(\mathsf{Adv}^{\text{COLL}}_{\mathsf{H},\mathcal{B}_1} \\
& + n_p n_s \cdot \Big(\mathsf{Adv}^{\text{dual-PRF-sec}}_{\mathsf{HKDF.Extract},\mathcal{B}_2} + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_3} \\
& \quad + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Extract},\mathcal{B}_4} + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_5} \\
& \quad + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_6} + \mathsf{Adv}^{\text{EUF-CMA}}_{\mathsf{HMAC},\mathcal{B}_7} \\
& \quad + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_8} + \mathsf{Adv}^{\text{EUF-CMA}}_{\mathsf{HMAC},\mathcal{B}_9} \Big) \\
& + n_p n_s \cdot \Big(\mathsf{Adv}^{\text{dual-PRF-sec}}_{\mathsf{HKDF.Extract},\mathcal{B}_{10}} + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_{11}} \Big) \\
& + n_s \cdot \Big(\mathsf{Adv}^{\text{dual-snPRF-ODH}}_{\mathsf{HKDF.Extract},\mathbb{G},\mathcal{B}_{12}} + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_{13}} \\
& \quad + 2\mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_{14}} + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Extract},\mathcal{B}_{15}} \\
& \quad + \mathsf{Adv}^{\text{PRF-sec}}_{\mathsf{HKDF.Expand},\mathcal{B}_{16}} \Big) \Big).
\end{aligned}
$$

[Dow+21]

TLS 1.3 Handshake Security

TLS 1.3 PSK-(EC)DHE 0-RTT as Multi-Stage Key Exchange

ETH zürich



Theorem 6.4. *The TLS 1.3 PSK-(EC)DHE 0-RTT handshake is* **Multi-Stage-*secure*** *with properties* (M, **AUTH**, **FS**, USE, **REPLAY**).

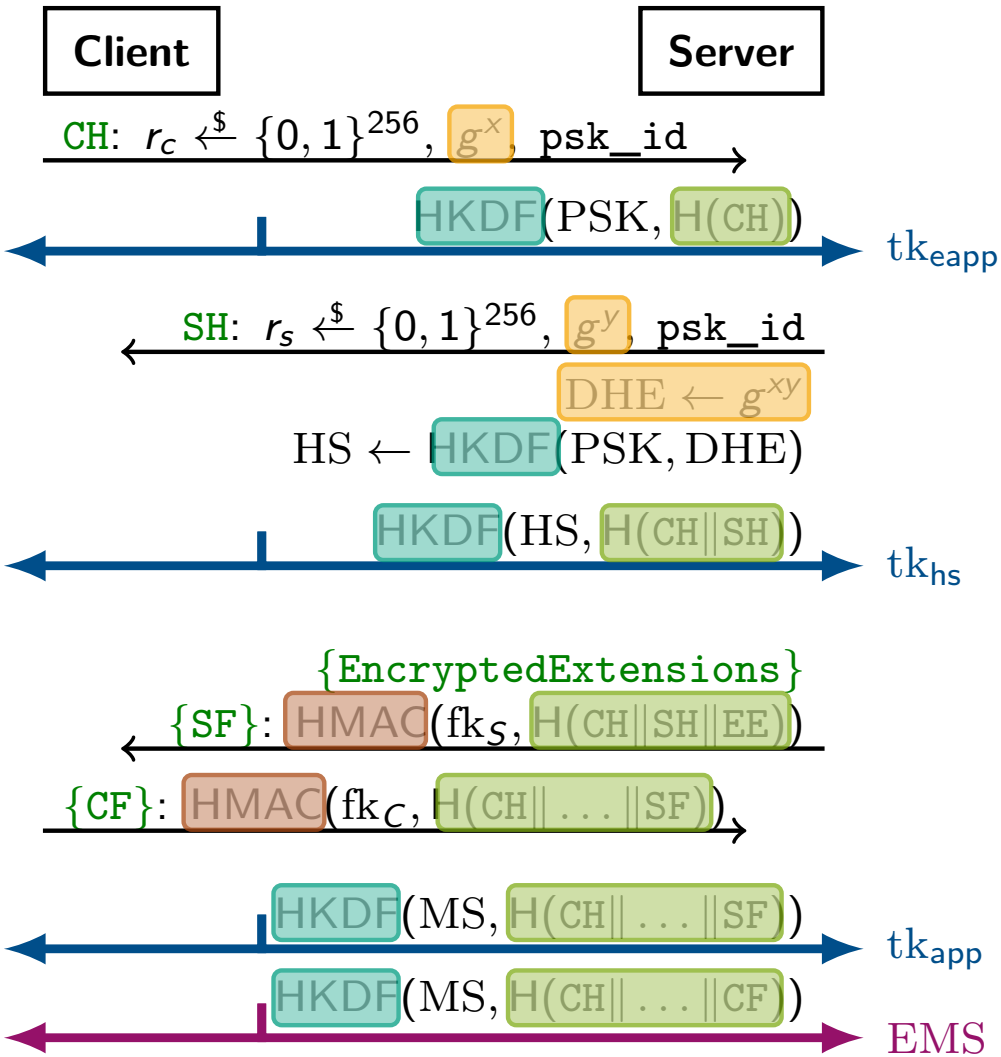
$$
\begin{aligned}
\mathrm{Adv}^{\text{Multi-Stage},\mathcal{D}}_{\text{TLS1.3-PSK-(EC)DHE-0RTT},\mathcal{A}} \leq\ & 8n_s \cdot \Big(\mathrm{Adv}^{\text{COLL}}_{\mathrm{H},\mathcal{B}_1} \\
& + n_p n_s \cdot \Big(\mathrm{Adv}^{\text{dual-PRF-sec}}_{\text{HKDF.Extract},\mathcal{B}_2} + \mathrm{Adv}^{\text{PRF-sec}}_{\text{HKDF.Expand},\mathcal{B}_3} \\
& \quad + \mathrm{Adv}^{\text{PRF-sec}}_{\text{HKDF.Extract},\mathcal{B}_4} + \mathrm{Adv}^{\text{PRF-sec}}_{\text{HKDF.Expand},\mathcal{B}_5} \\
& \quad + \mathrm{Adv}^{\text{PRF-sec}}_{\text{HKDF.Expand},\mathcal{B}_6} + \mathrm{Adv}^{\text{EUF-CMA}}_{\text{HMAC},\mathcal{B}_7} \\
& \quad + \mathrm{Adv}^{\text{PRF-sec}}_{\text{HKDF.Expand},\mathcal{B}_8} + \mathrm{Adv}^{\text{EUF-CMA}}_{\text{HMAC},\mathcal{B}_9} \Big) \\
& + n_p n_s \cdot \Big(\mathrm{Adv}^{\text{dual-PRF-sec}}_{\text{HKDF.Extract},\mathcal{B}_{10}} + \mathrm{Adv}^{\text{PRF-sec}}_{\text{HKDF.Expand},\mathcal{B}_{11}} \Big) \\
& + n_s \cdot \Big(\mathrm{Adv}^{\text{dual-snPRF-ODH}}_{\text{HKDF.Extract},\mathbb{G},\mathcal{B}_{12}} + \mathrm{Adv}^{\text{PRF-sec}}_{\text{HKDF.Expand},\mathcal{B}_{13}} \\
& \quad + 2\mathrm{Adv}^{\text{PRF-sec}}_{\text{HKDF.Expand},\mathcal{B}_{14}} + \mathrm{Adv}^{\text{PRF-sec}}_{\text{HKDF.Extract},\mathcal{B}_{15}} \\
& \quad + \mathrm{Adv}^{\text{PRF-sec}}_{\text{HKDF.Expand},\mathcal{B}_{16}} \Big) \Big) .
\end{aligned}
$$

[Dow+21]

TLS 1.3 Handshake Security

TLS 1.3 PSK-(EC)DHE 0-RTT as Multi-Stage Key Exchange

ETH zürich



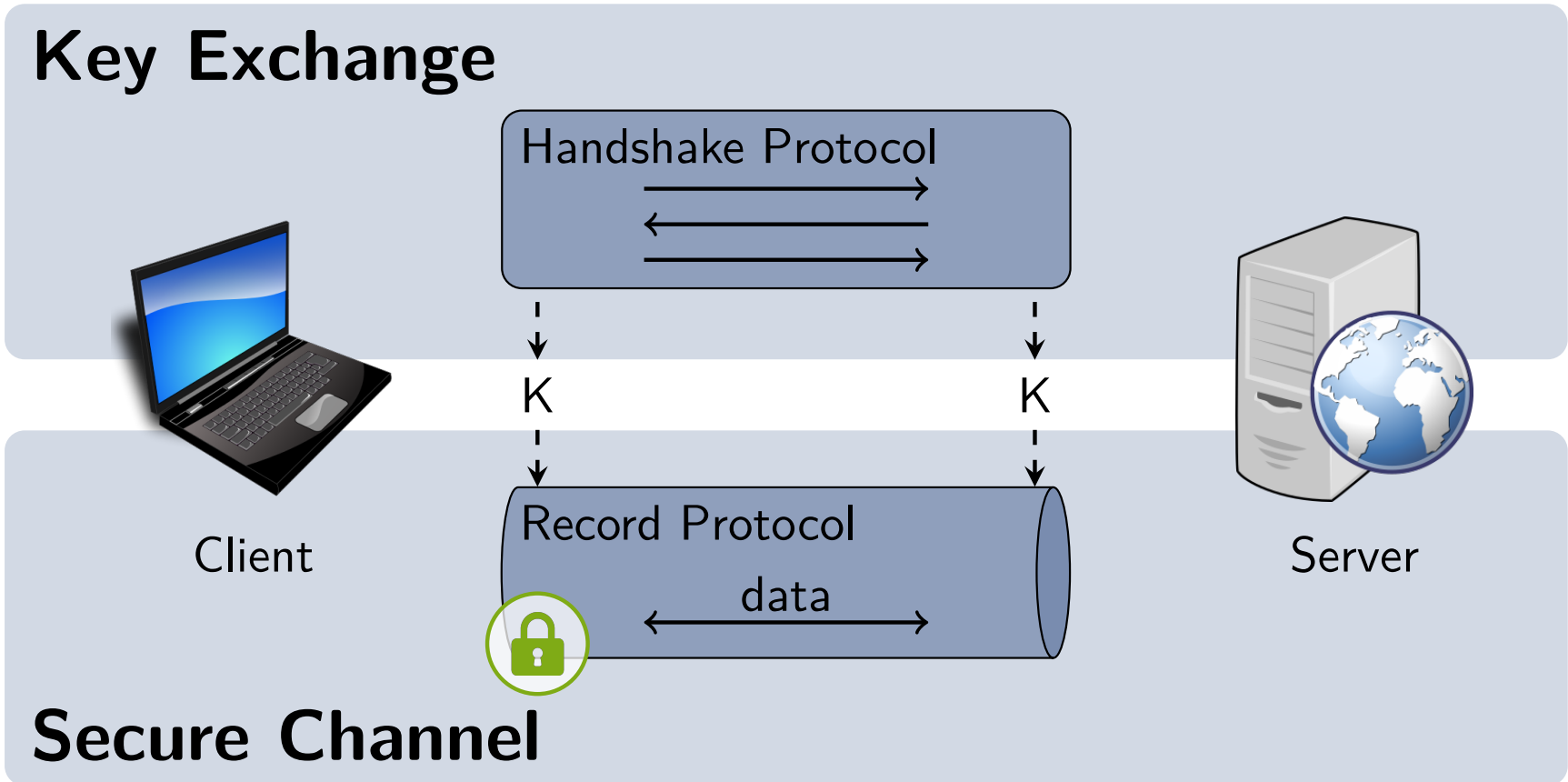
Theorem 6.4. *The TLS 1.3 PSK-(EC)DHE 0-RTT handshake is* **Multi-Stage-*secure*** *with properties* (M, **AUTH**, **FS**, USE, **REPLAY**).

$$\begin{aligned}
\mathsf{Adv}^{\text{Multi-Stage},\mathcal{D}}_{\text{TLS1.3-PSK-(EC)DHE-ORTT},\mathcal{A}} \leq\ & 8n_s \cdot \Big(\mathsf{Adv}^{\text{COLL}}_{H,\mathcal{B}_1} \\
& + n_p n_s \cdot \Big(\mathsf{Adv}^{\text{dual-PRF-sec}}_{\text{HKDF.Extract},\mathcal{B}_2} + \mathsf{Adv}^{\text{PRF-sec}}_{\text{HKDF.Expand},\mathcal{B}_3} \\
& \quad + \mathsf{Adv}^{\text{PRF-sec}}_{\text{HKDF.Extract},\mathcal{B}_4} + \mathsf{Adv}^{\text{PRF-sec}}_{\text{HKDF.Expand},\mathcal{B}_5} \\
& \quad + \mathsf{Adv}^{\text{PRF-sec}}_{\text{HKDF.Expand},\mathcal{B}_6} + \mathsf{Adv}^{\text{EUF-CMA}}_{\text{HMAC},\mathcal{B}_7} \\
& \quad + \mathsf{Adv}^{\text{PRF-sec}}_{\text{HKDF.Expand},\mathcal{B}_8} + \mathsf{Adv}^{\text{EUF-CMA}}_{\text{HMAC},\mathcal{B}_9} \Big) \\
& + n_p n_s \cdot \Big(\mathsf{Adv}^{\text{dual-PRF-sec}}_{\text{HKDF.Extract},\mathcal{B}_{10}} + \mathsf{Adv}^{\text{PRF-sec}}_{\text{HKDF.Expand},\mathcal{B}_{11}} \Big) \\
& + n_s \cdot \Big(\mathsf{Adv}^{\text{dual-snPRF-ODH}}_{\text{HKDF.Extract},\mathbb{G},\mathcal{B}_{12}} + \mathsf{Adv}^{\text{PRF-sec}}_{\text{HKDF.Expand},\mathcal{B}_{13}} \\
& \quad + 2\mathsf{Adv}^{\text{PRF-sec}}_{\text{HKDF.Expand},\mathcal{B}_{14}} + \mathsf{Adv}^{\text{PRF-sec}}_{\text{HKDF.Extract},\mathcal{B}_{15}} \\
& \quad + \mathsf{Adv}^{\text{PRF-sec}}_{\text{HKDF.Expand},\mathcal{B}_{16}} \Big) \Big).
\end{aligned}$$

[Dow+21]

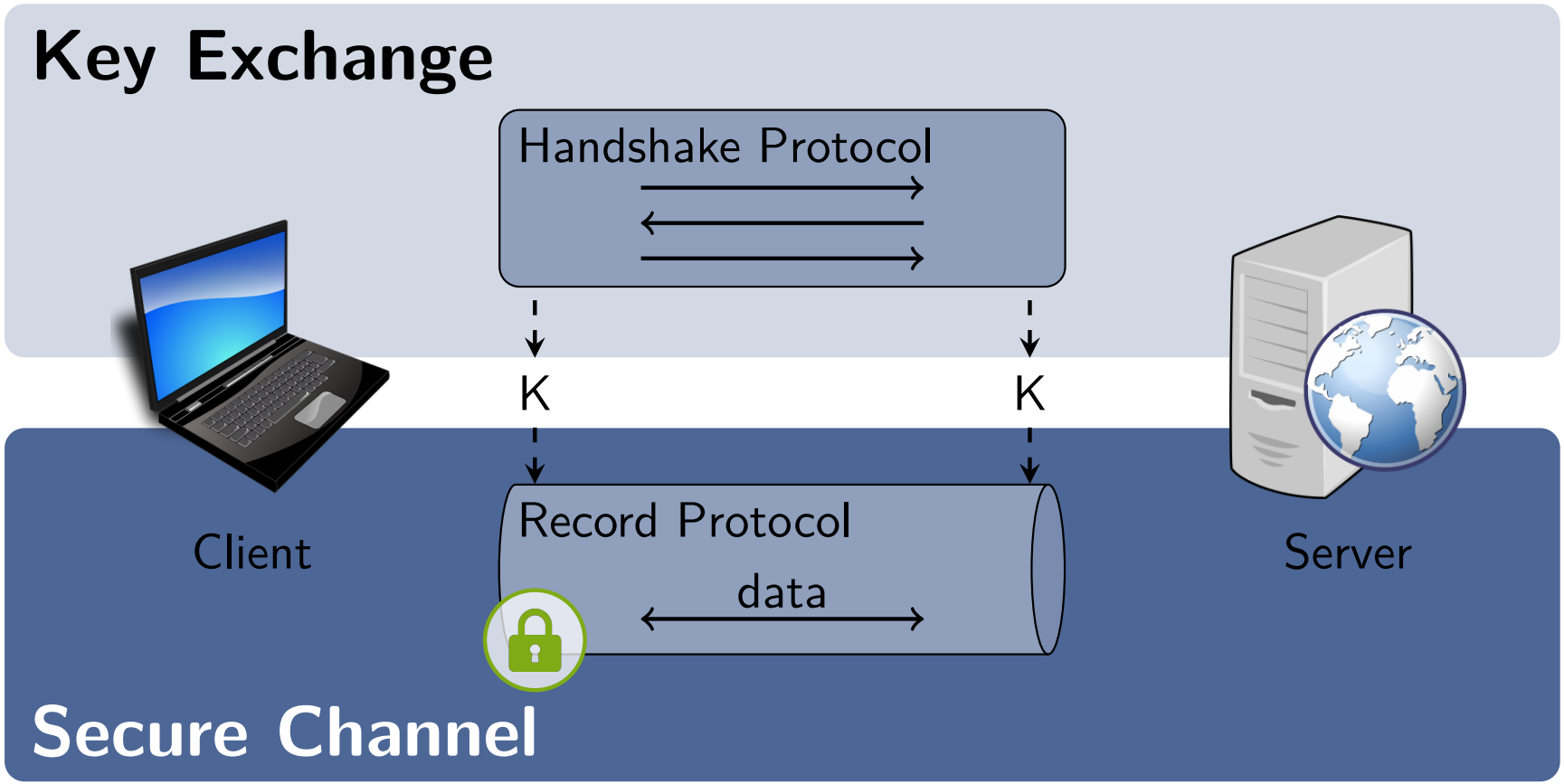
The Cryptographic Core

ETH zürich



The Cryptographic Core

ETH zürich



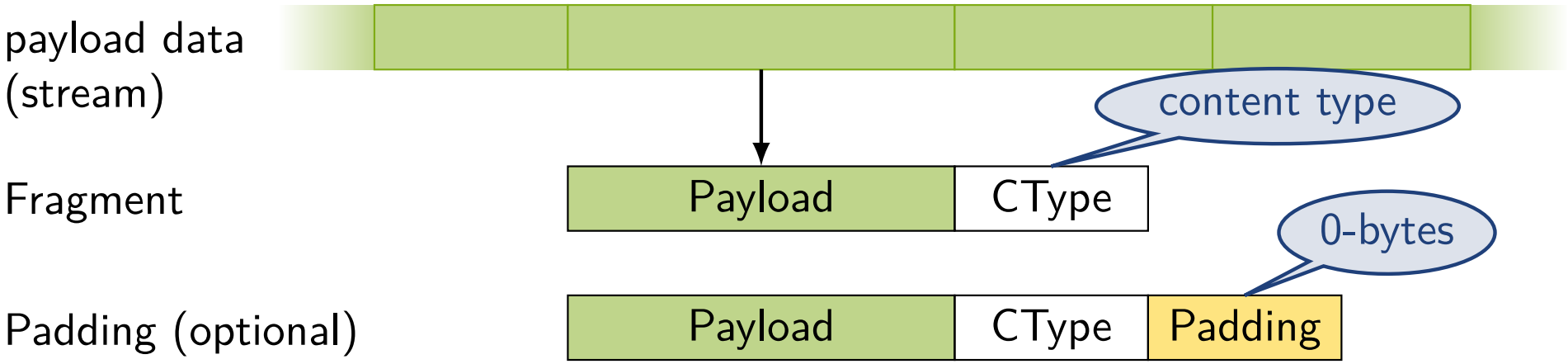
The TLS 1.3 Record Protocol

ETH zürich

payload data
(stream)

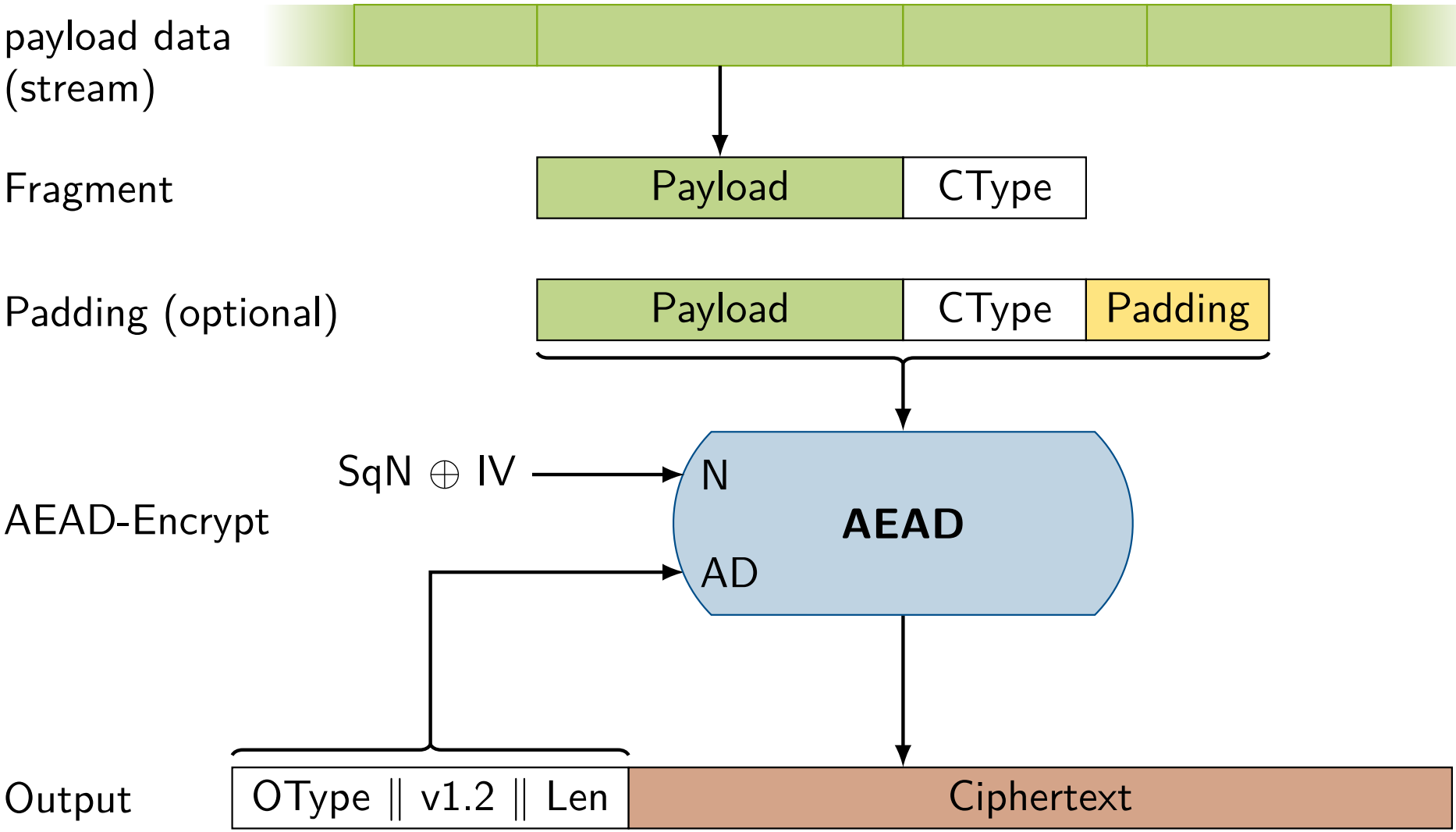
The TLS 1.3 Record Protocol

ETH zürich



The TLS 1.3 Record Protocol

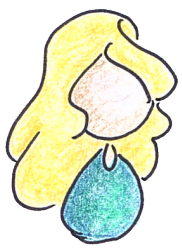
ETH zürich



Channel Security: Stateful Auth. Encryption

Originating from [BKN02]

ETH zürich

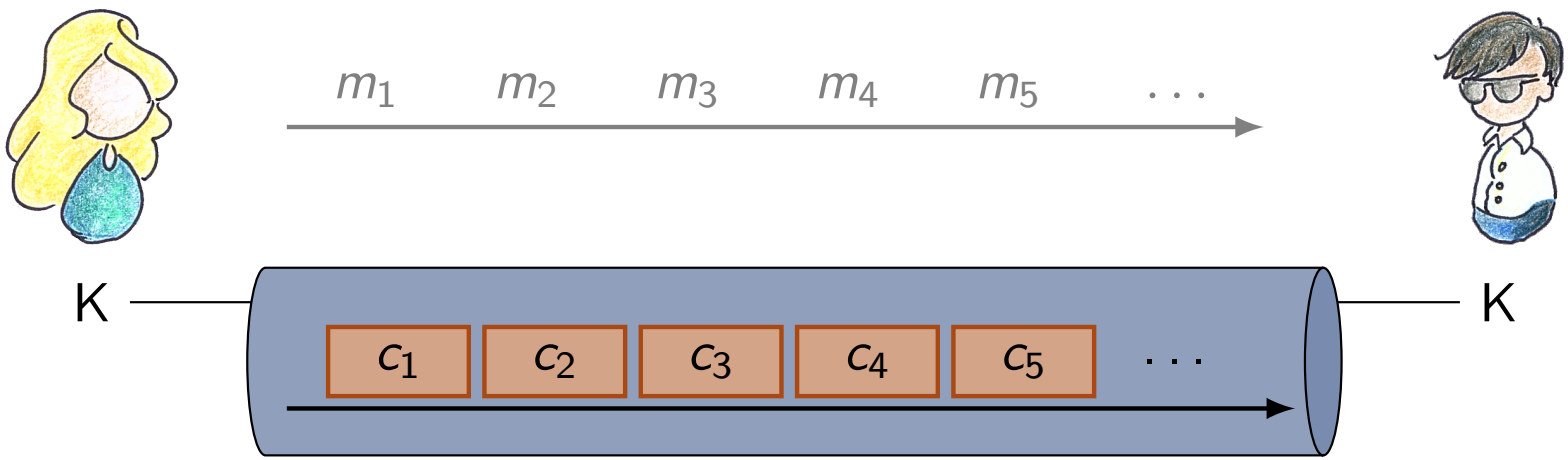


$m_1 \quad m_2 \quad m_3 \quad m_4 \quad m_5 \quad \ldots$

Channel Security: Stateful Auth. Encryption

Originating from [BKN02]

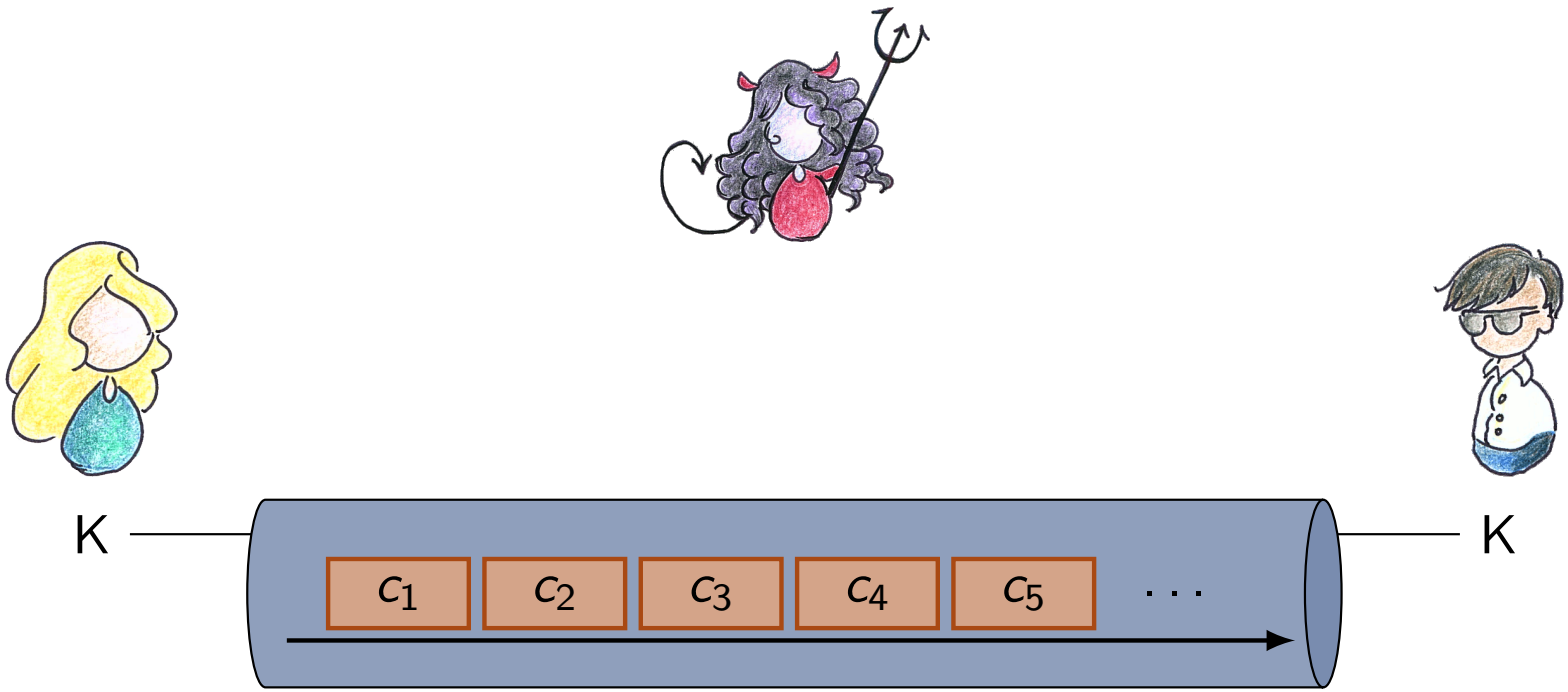
ETH zürich



Channel Security: Stateful Auth. Encryption

Originating from [BKN02]

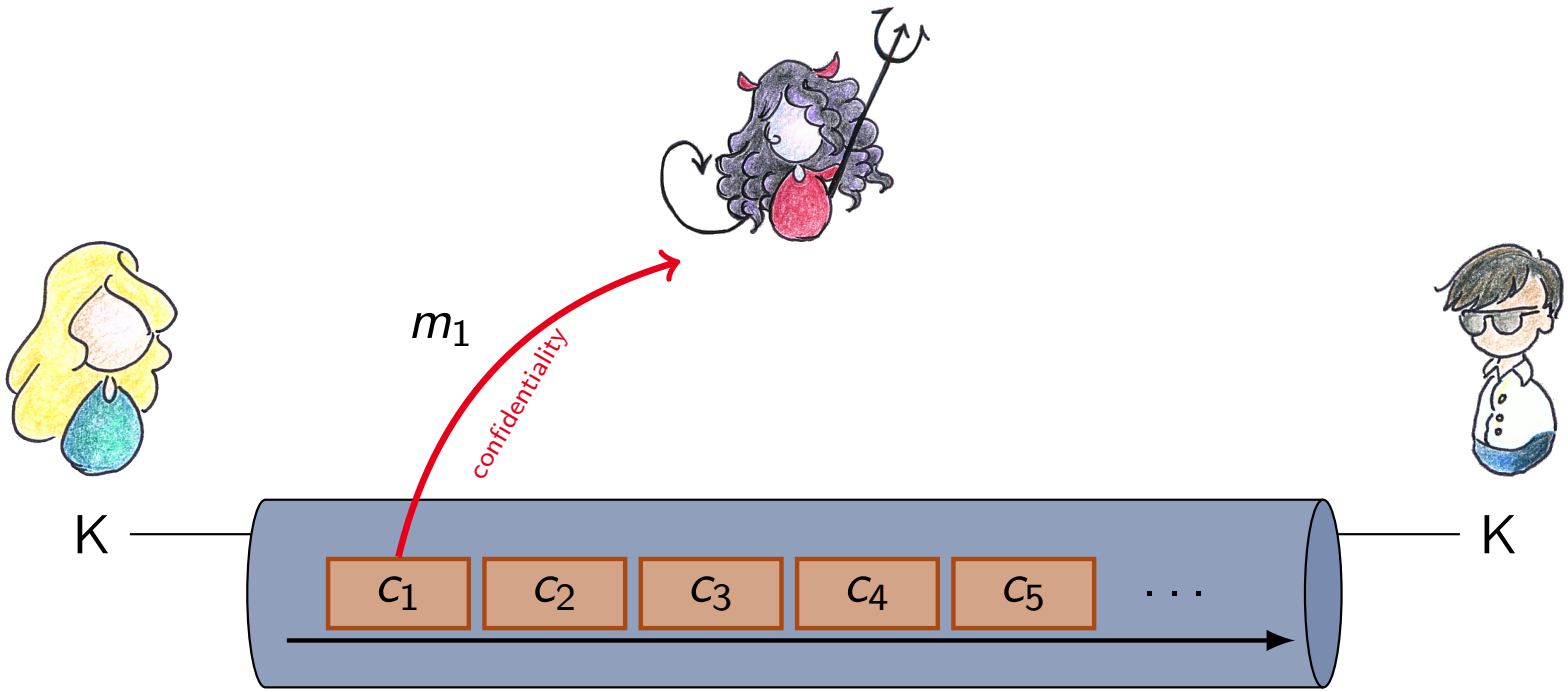
ETH zürich



Channel Security: Stateful Auth. Encryption

Originating from [BKN02]

ETH zürich



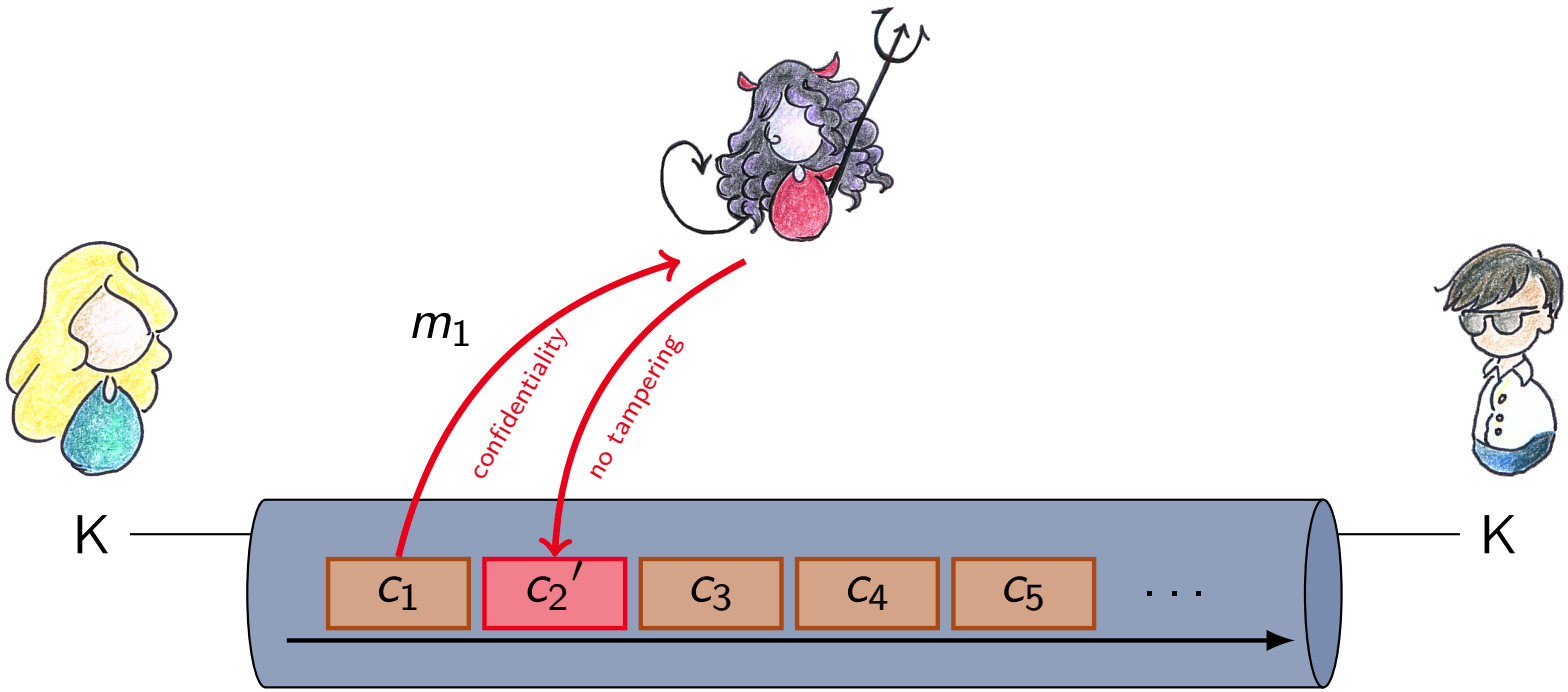
IND-sfCPA (passive confidentiality)

IND-sfCCA (active confidentiality)

Channel Security: Stateful Auth. Encryption

Originating from [BKN02]

ETH zürich



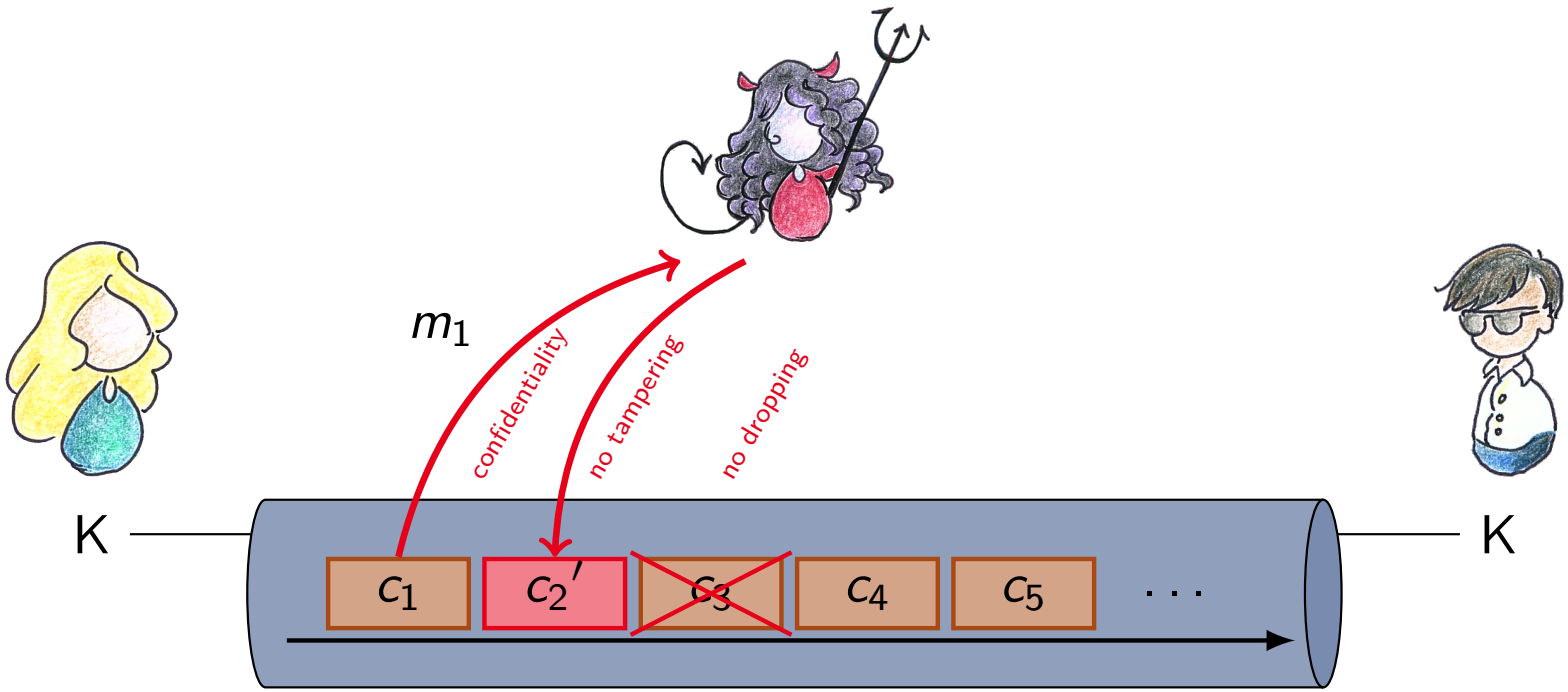
IND-sfCPA (passive confidentiality)

IND-sfCCA (active confidentiality)

Channel Security: Stateful Auth. Encryption

Originating from [BKN02]

ETH zürich



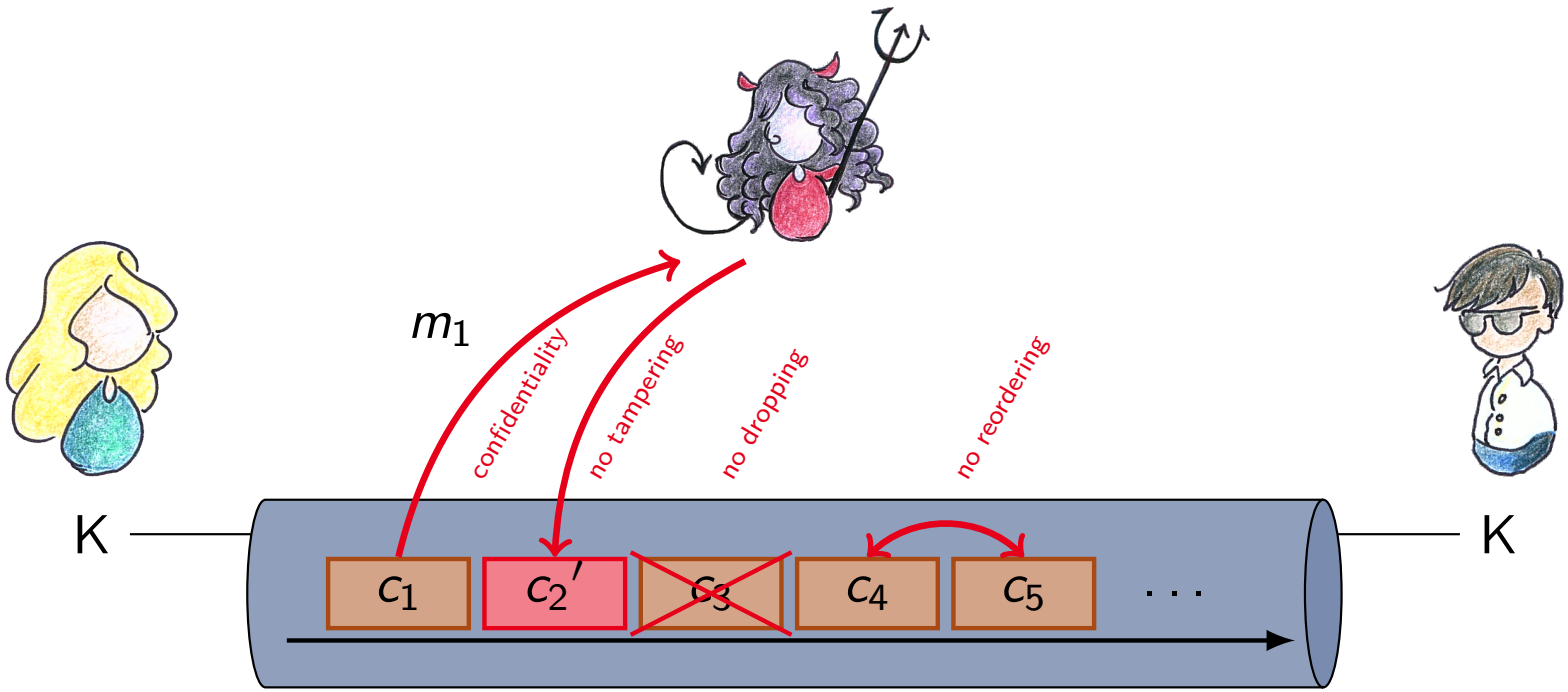
IND-sfCPA (passive confidentiality)

IND-sfCCA (active confidentiality)

Channel Security: Stateful Auth. Encryption

Originating from [BKN02]

ETH zürich



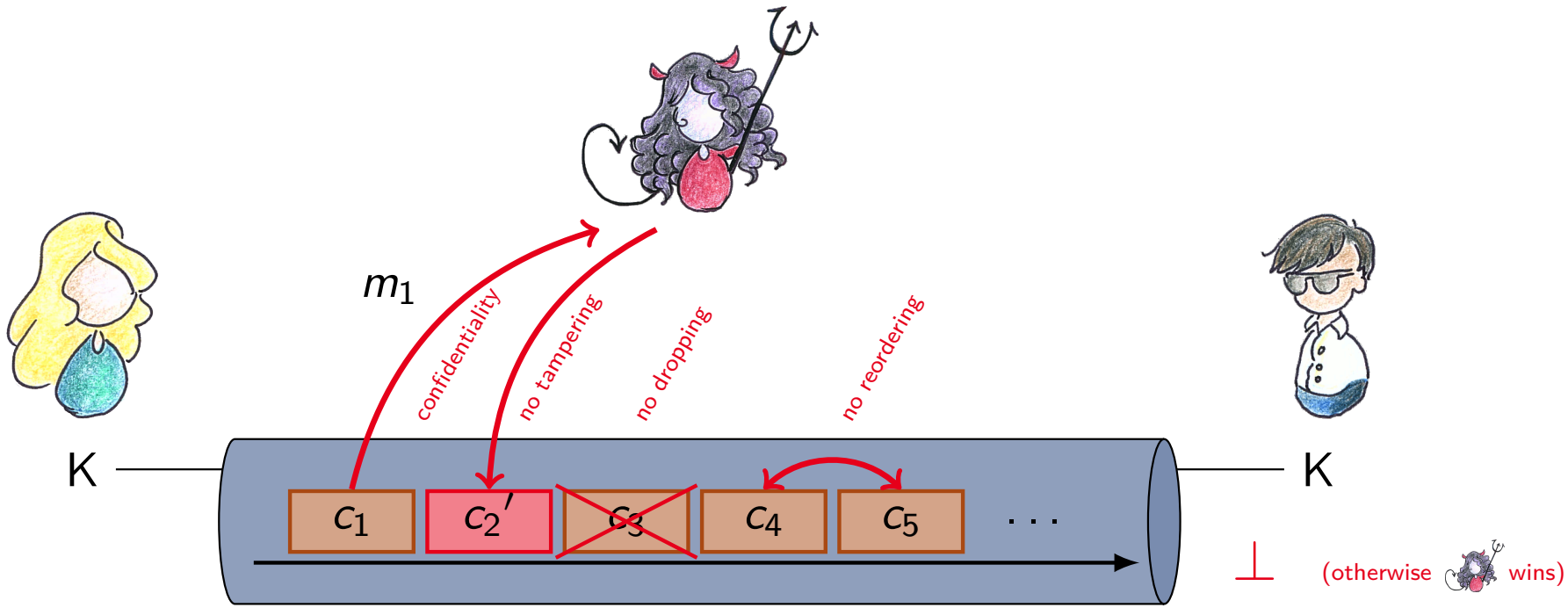
IND-sfCPA (passive confidentiality)

IND-sfCCA (active confidentiality)

Channel Security: Stateful Auth. Encryption

Originating from [BKN02]

ETH zürich



IND-sfCPA (passive confidentiality)

IND-sfCCA (active confidentiality)

INT-sfPTXT (plaintext integrity)

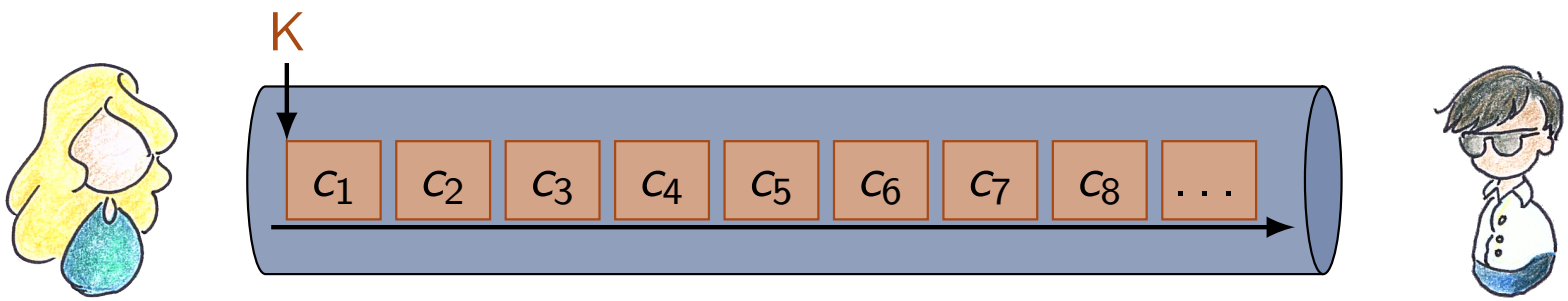
INT-sfCTXT (ciphertext integrity)

TLS 1.3 Record Protocol Security

. . . a **multi-key** channel [GM17]

ETH zürich

- classically: 1 key ✓

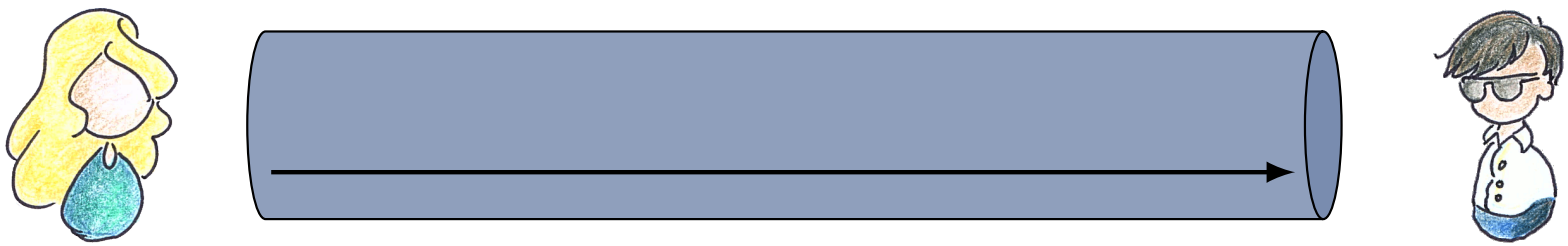


TLS 1.3 Record Protocol Security

. . . a **multi-key** channel [GM17]

ETH zürich

- classically: 1 key ✓
- **TLS 1.3**, **Signal**, **QUIC**, . . . : keys updated during channel operation

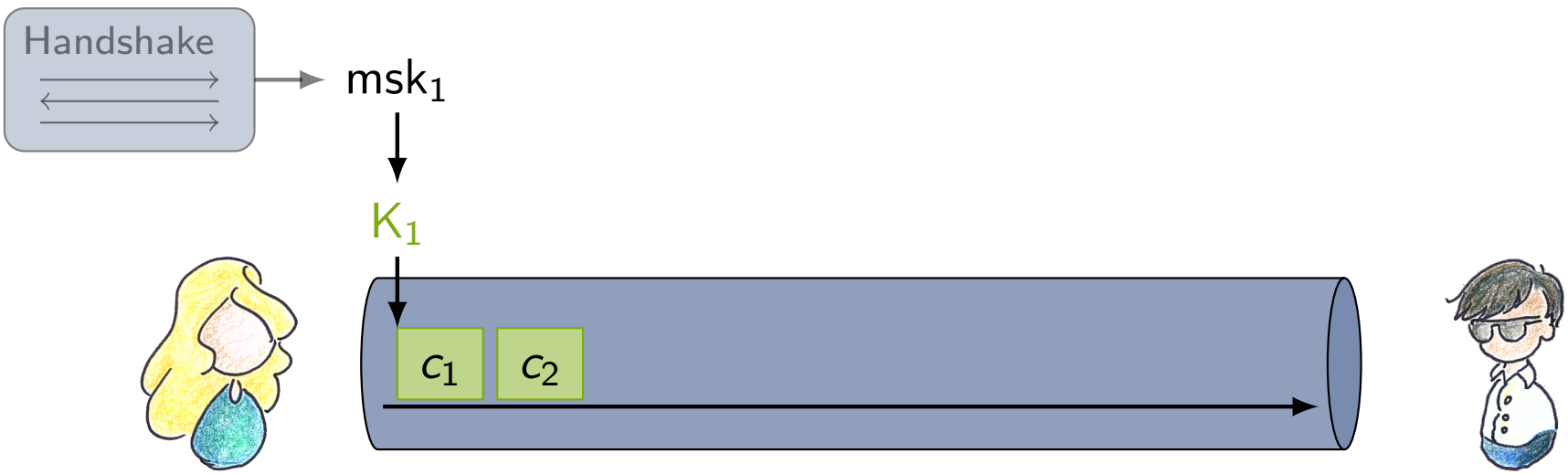


TLS 1.3 Record Protocol Security

. . . a **multi-key** channel [GM17]

ETH zürich

- classically: 1 key ✓
- **TLS 1.3**, **Signal**, **QUIC**, . . . : keys updated during channel operation

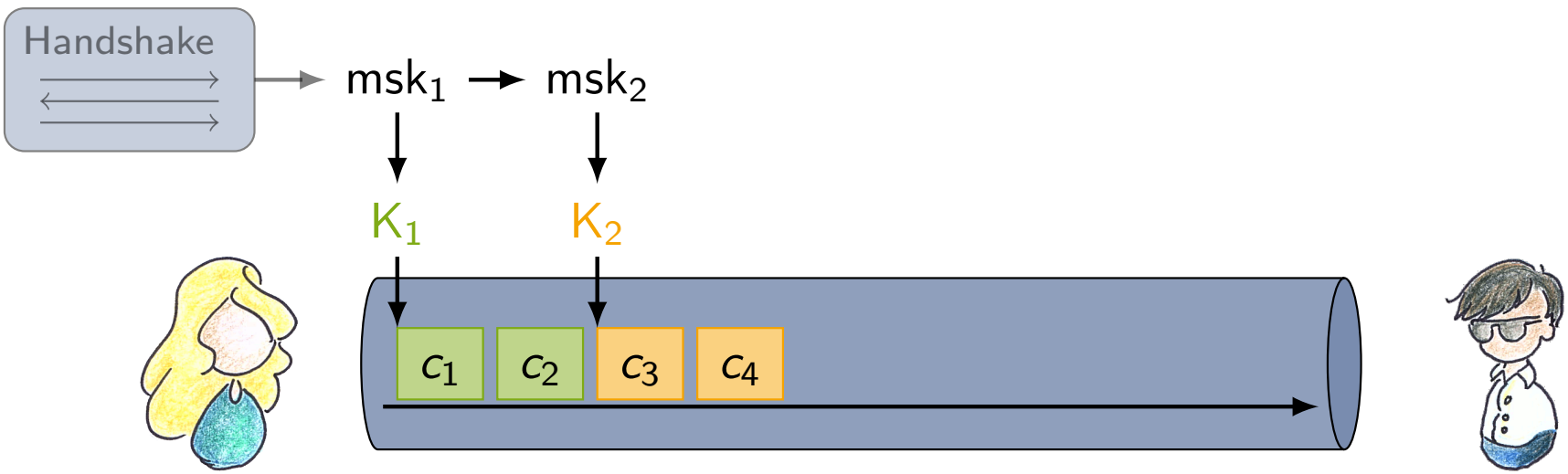


TLS 1.3 Record Protocol Security

. . . a **multi-key** channel [GM17]

ETH zürich

- classically: 1 key ✓
- **TLS 1.3**, **Signal**, **QUIC**, . . . : keys updated during channel operation

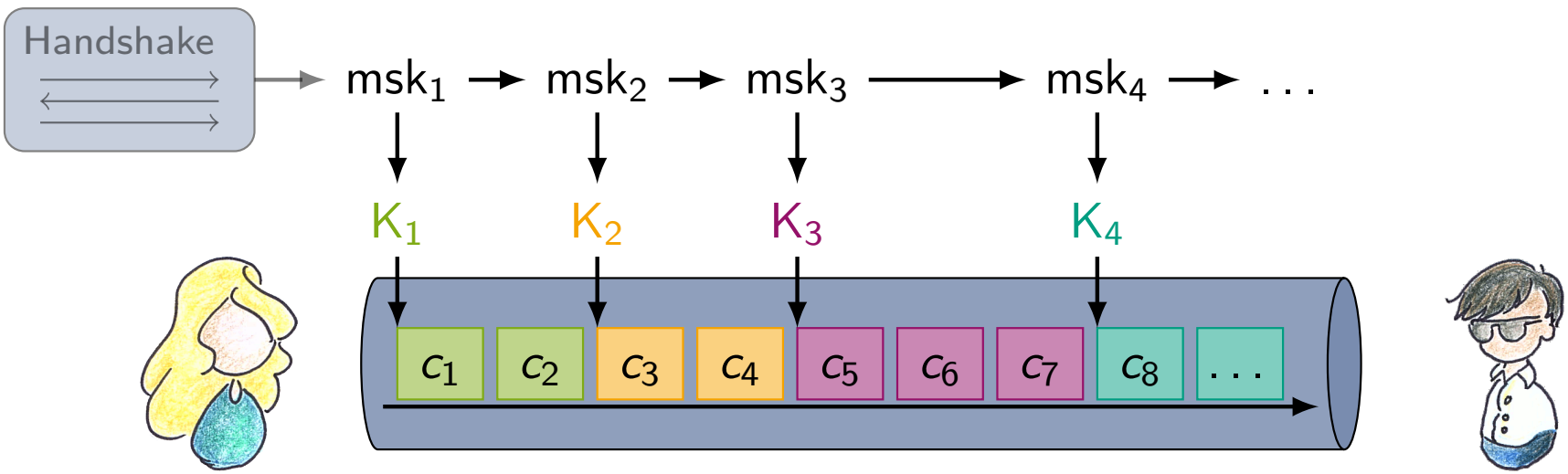


TLS 1.3 Record Protocol Security

. . . a **multi-key** channel [GM17]

ETH zürich

- classically: 1 key ✓
- **TLS 1.3**, **Signal**, **QUIC**, . . . : keys updated during channel operation

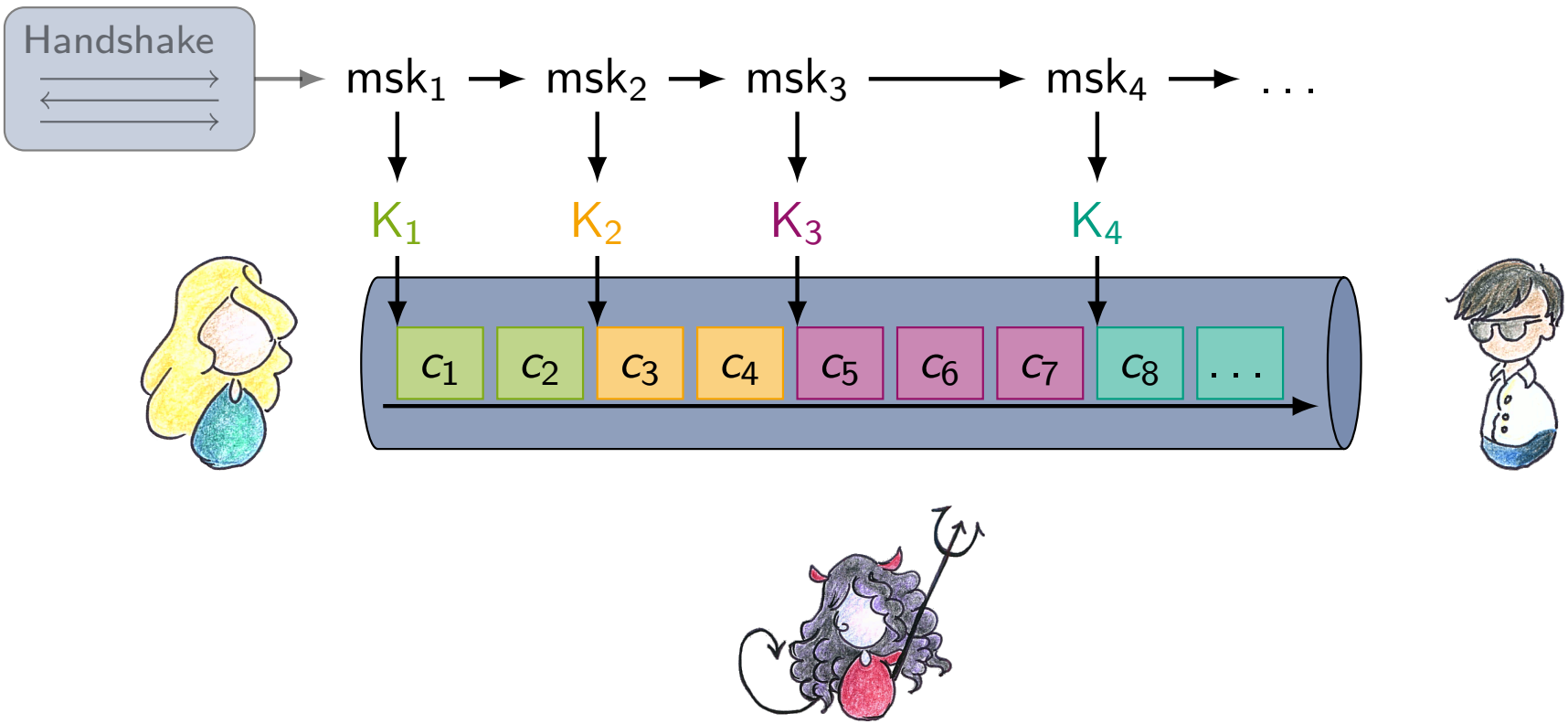


TLS 1.3 Record Protocol Security

. . . a **multi-key** channel [GM17]

ETH zürich

- classically: 1 key ✓
- **TLS 1.3**, **Signal**, **QUIC**, . . . : keys updated during channel operation

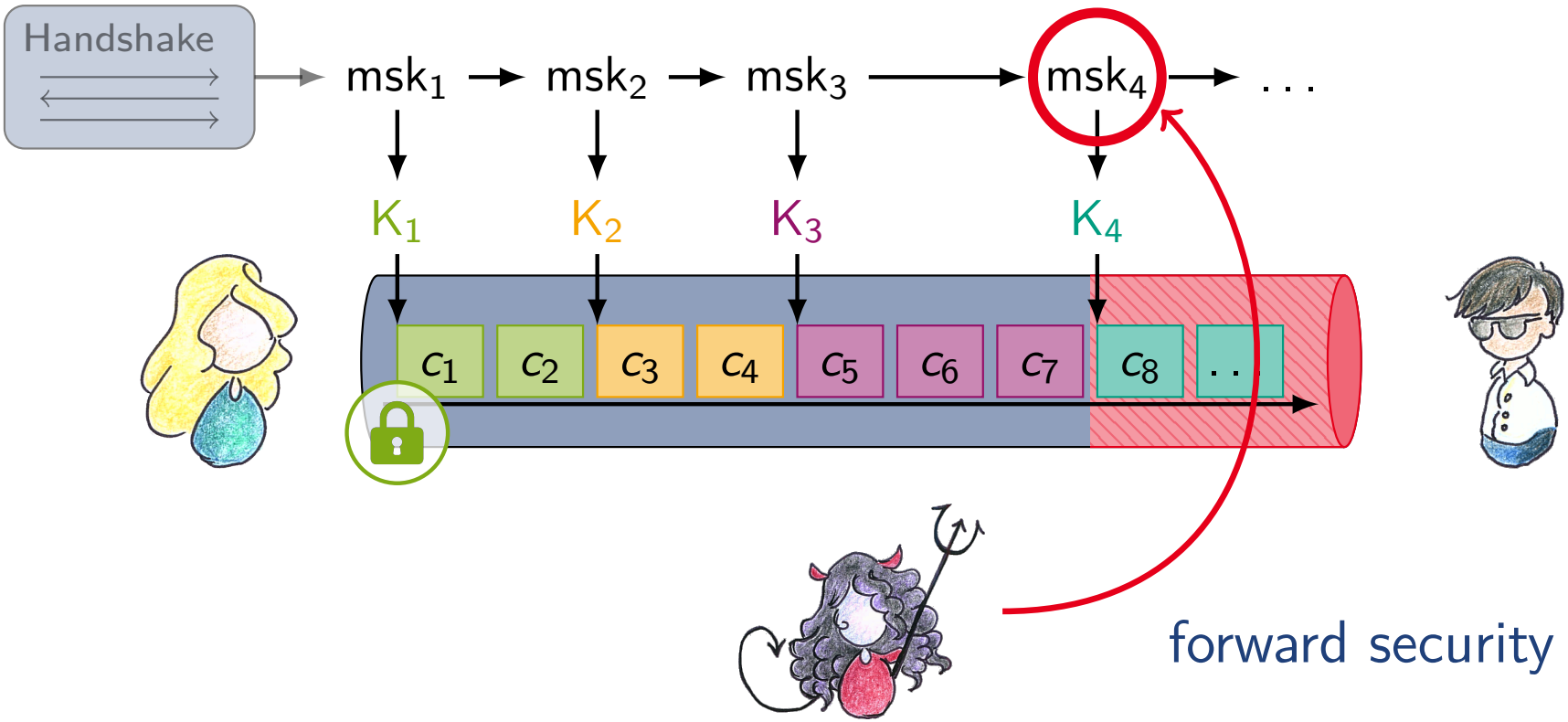


TLS 1.3 Record Protocol Security

. . . a **multi-key** channel [GM17]

ETH zürich

- classically: 1 key ✓
- **TLS 1.3**, **Signal**, **QUIC**, . . . : keys updated during channel operation

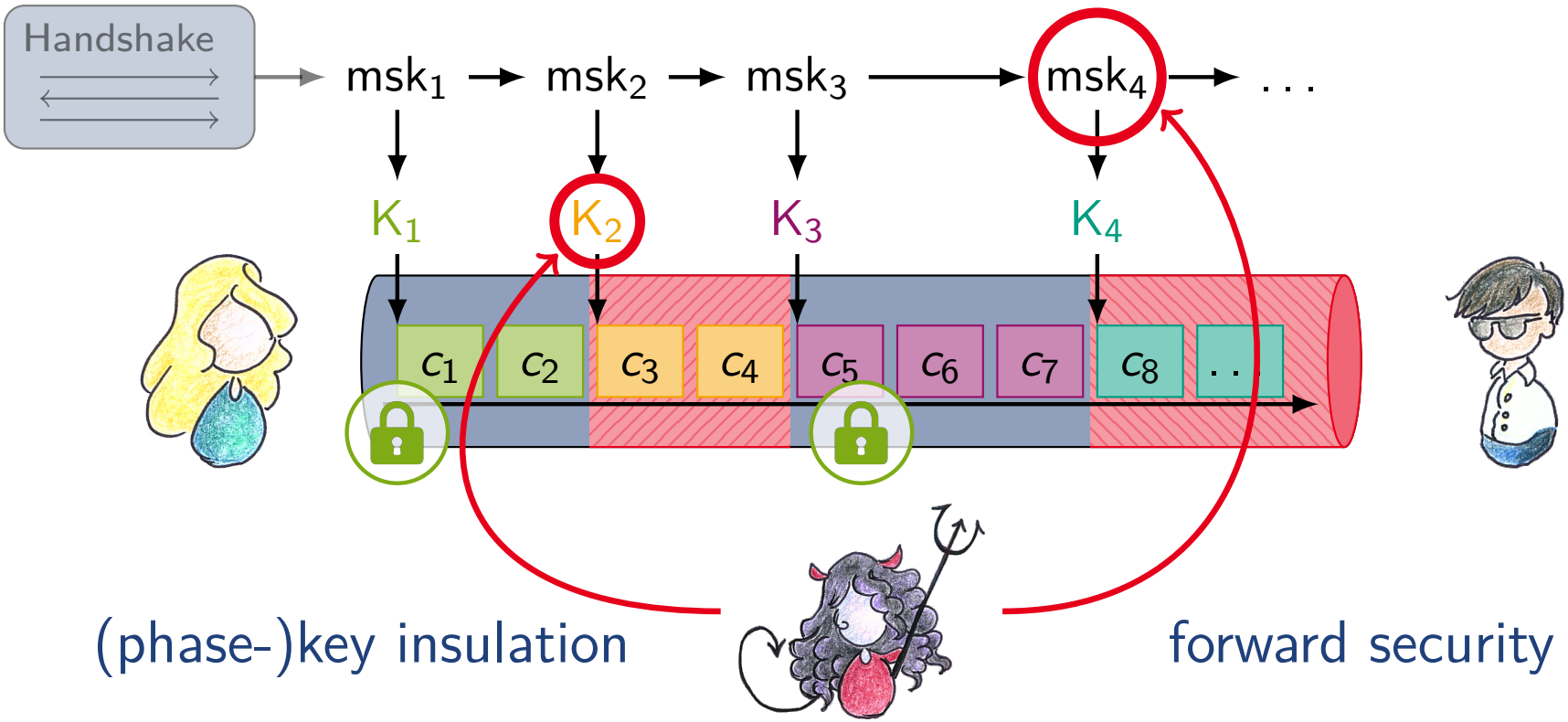


TLS 1.3 Record Protocol Security

. . . a **multi-key** channel [GM17]

ETH zürich

- classically: 1 key ✓
- **TLS 1.3**, **Signal**, **QUIC**, . . . : keys updated during channel operation



DTLS 1.3 Record Protocol Security

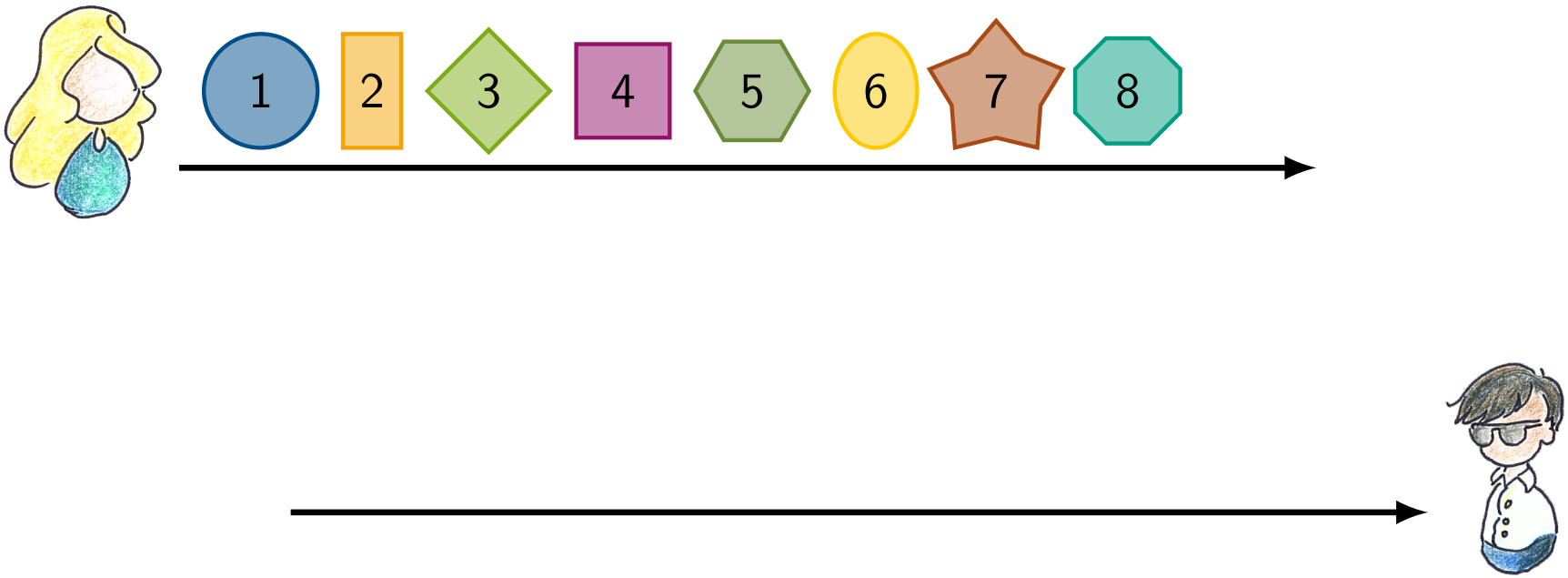
. . . a **robust** channel over **unreliable transport** UDP [FGJ20]

ETH zürich

DTLS 1.3 Record Protocol Security

. . . a **robust** channel over **unreliable transport** UDP [FGJ20]

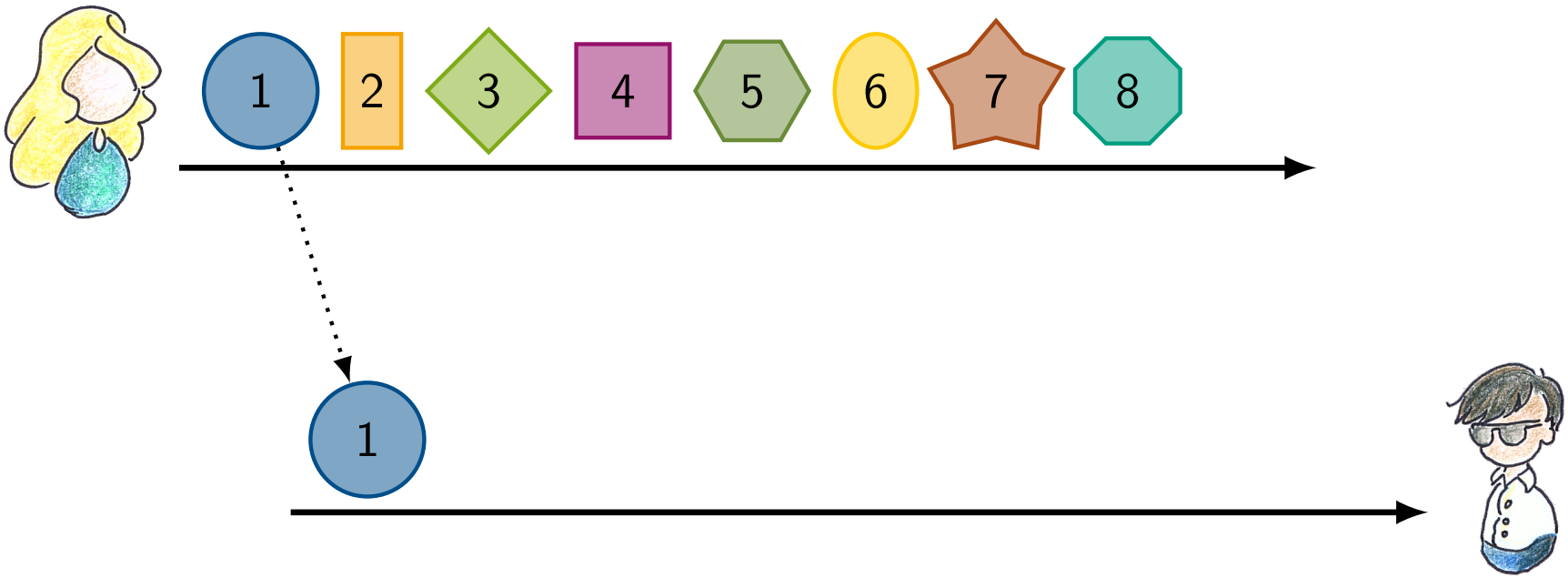
ETH zürich



DTLS 1.3 Record Protocol Security

. . . a **robust** channel over **unreliable transport** UDP [FGJ20]

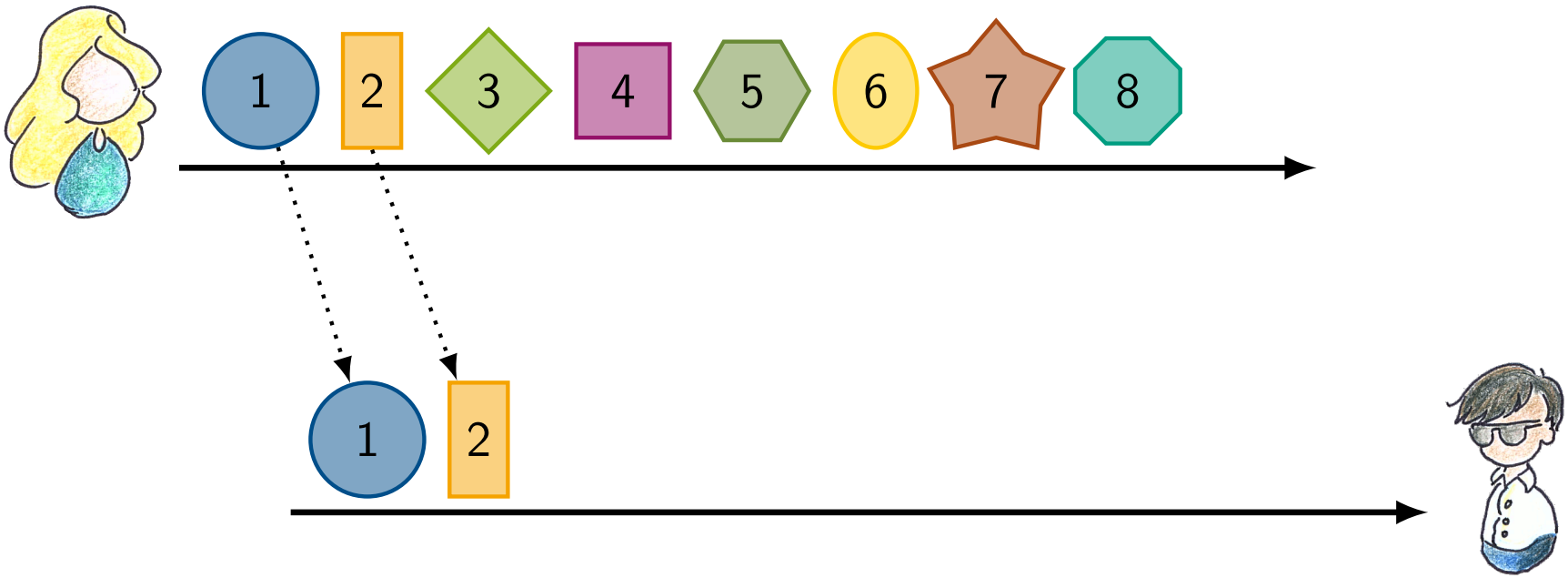
ETH zürich



DTLS 1.3 Record Protocol Security

. . . a **robust** channel over **unreliable transport** UDP [FGJ20]

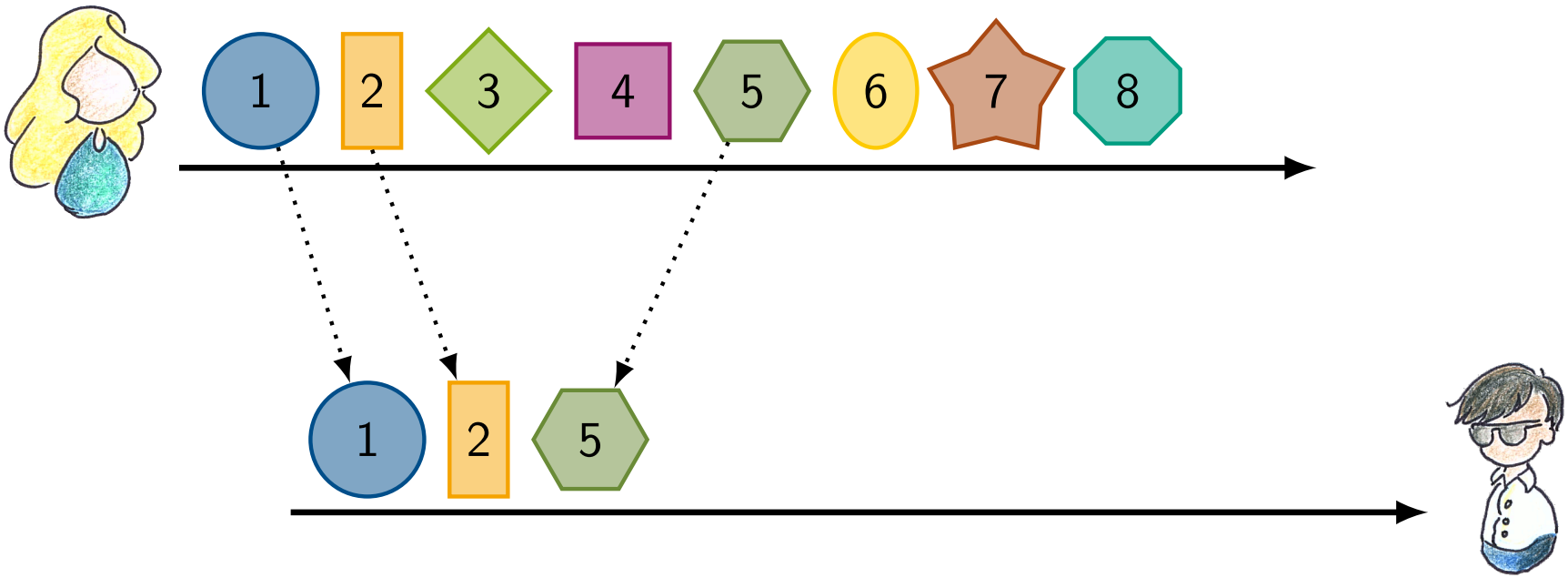
ETH zürich



DTLS 1.3 Record Protocol Security

. . . a **robust** channel over **unreliable transport** UDP [FGJ20]

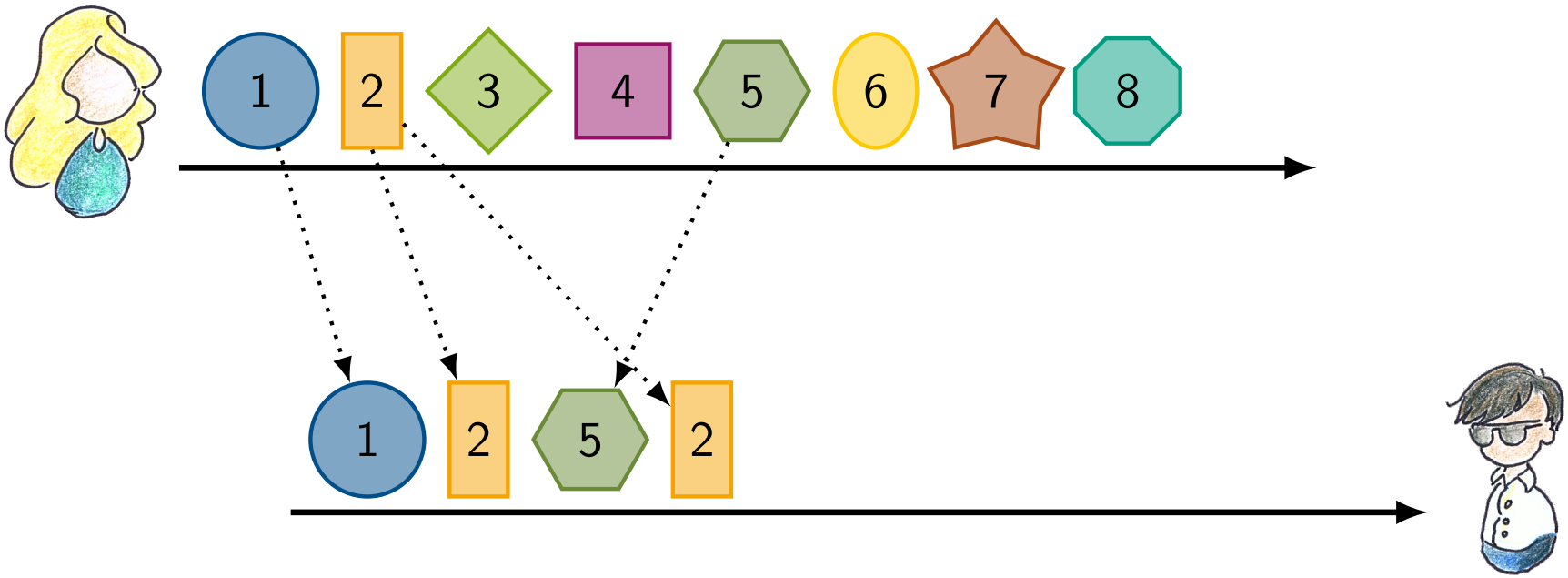
ETH zürich



DTLS 1.3 Record Protocol Security

. . . a **robust** channel over **unreliable transport** UDP [FGJ20]

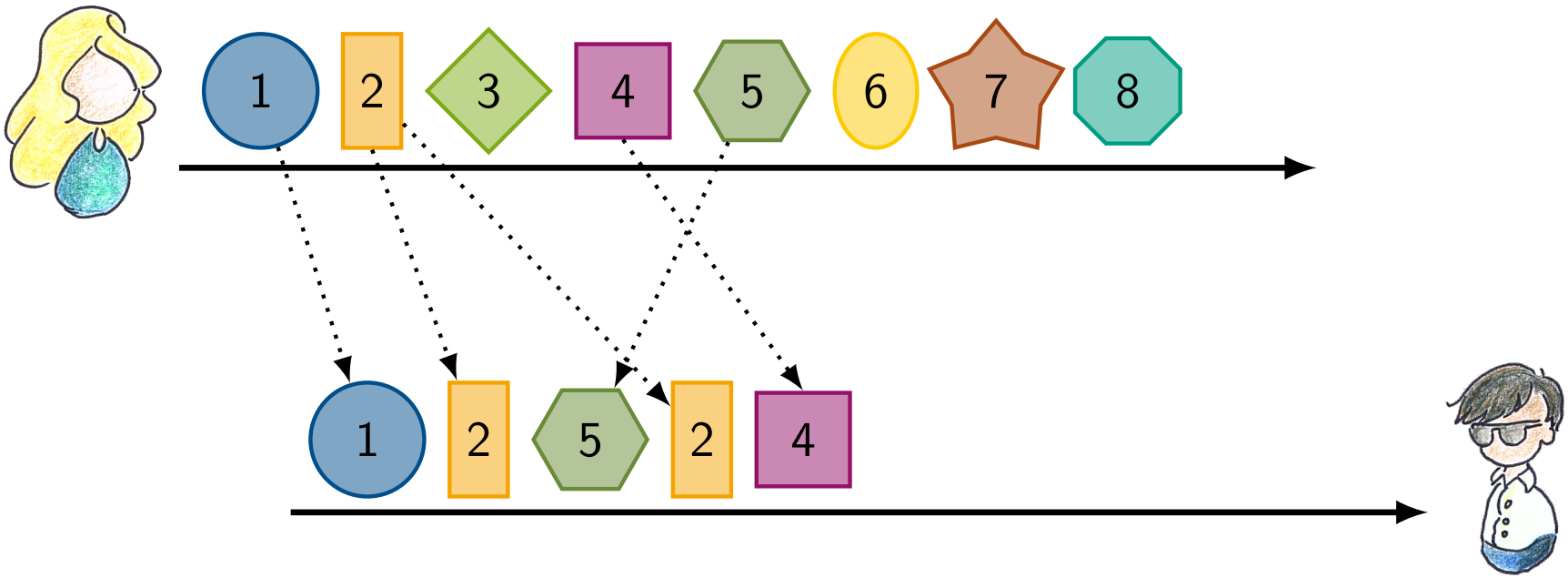
ETH zürich



DTLS 1.3 Record Protocol Security

. . . a **robust** channel over **unreliable transport** UDP [FGJ20]

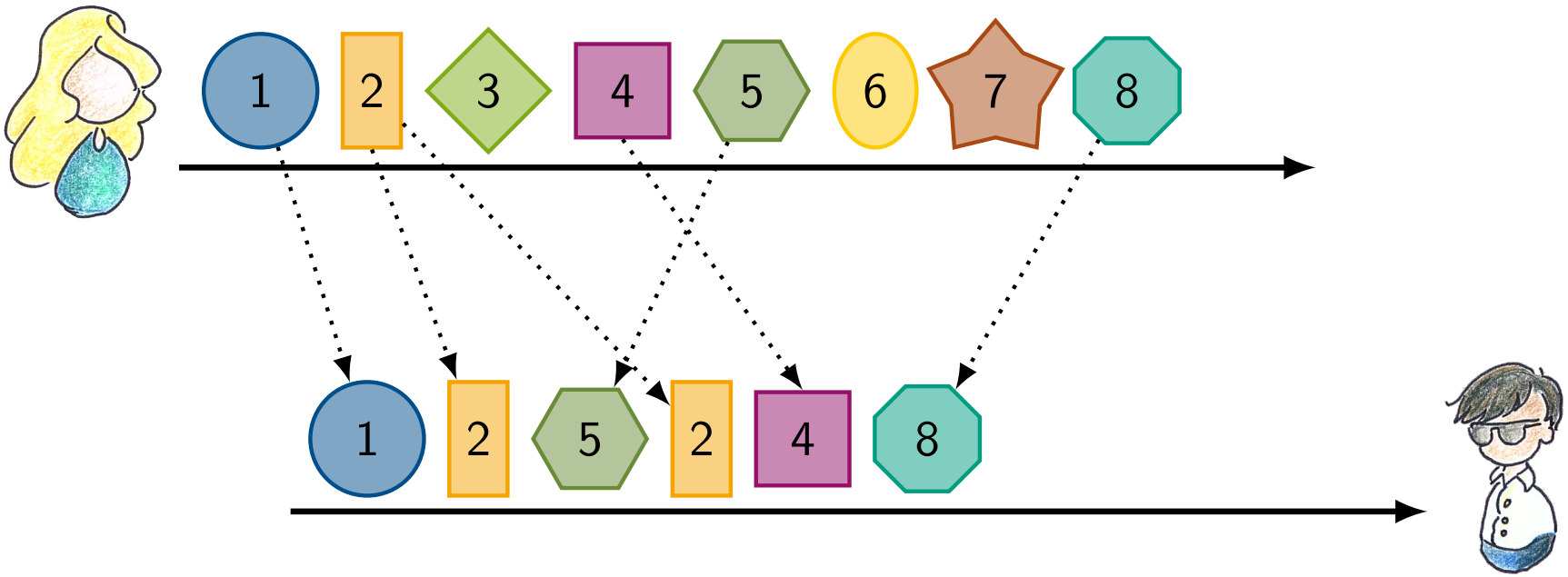
ETH zürich



DTLS 1.3 Record Protocol Security

. . . a **robust** channel over **unreliable transport** UDP [FGJ20]

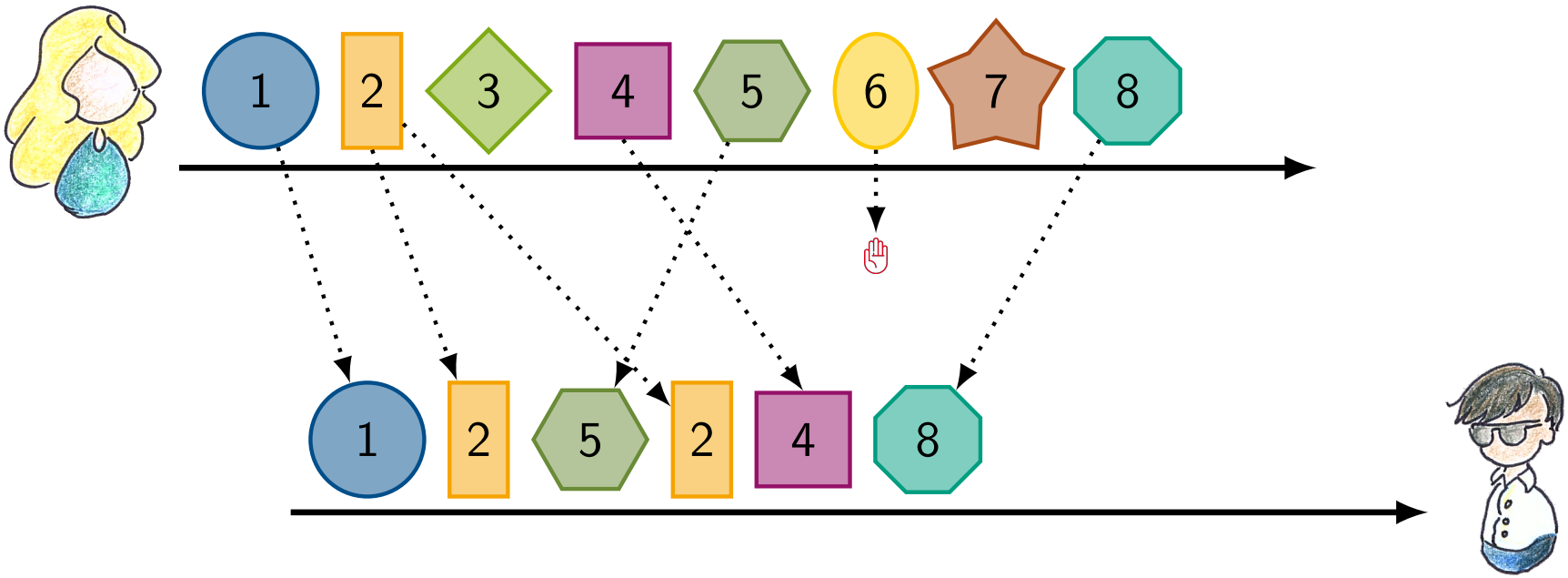
ETH zürich



DTLS 1.3 Record Protocol Security

. . . a **robust** channel over **unreliable transport** UDP [FGJ20]

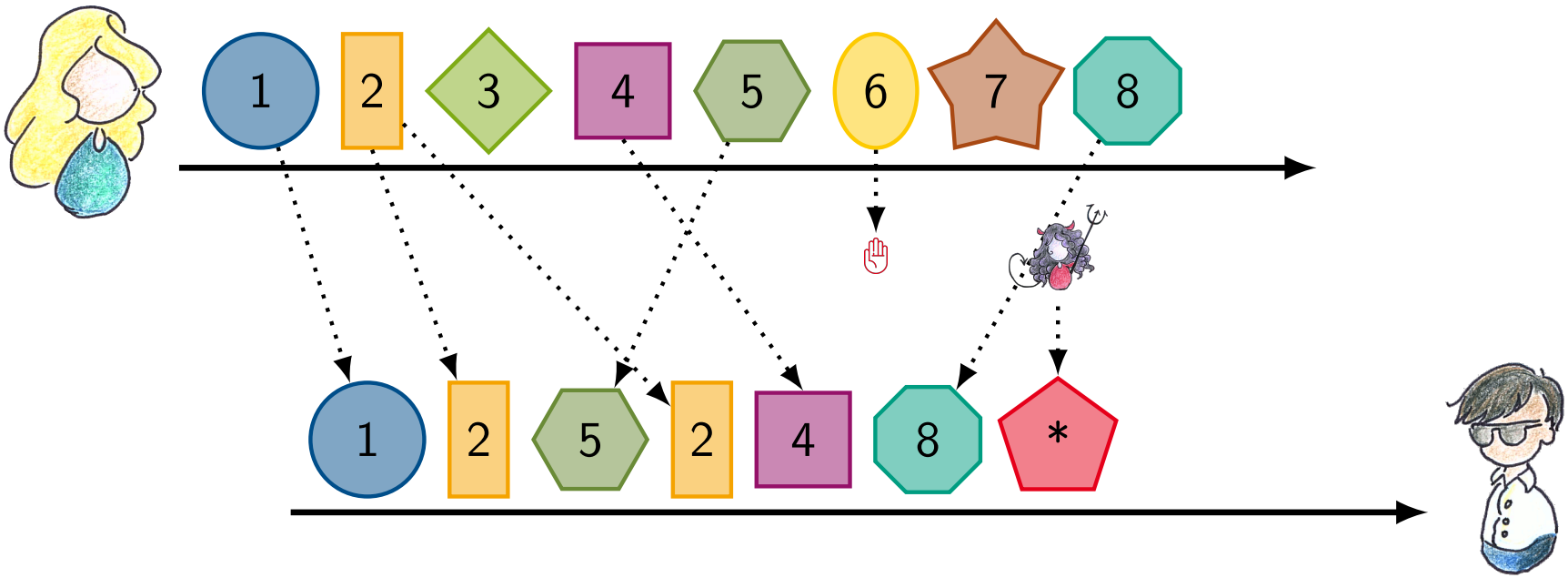
ETH zürich



DTLS 1.3 Record Protocol Security

. . . a **robust** channel over **unreliable transport** UDP [FGJ20]

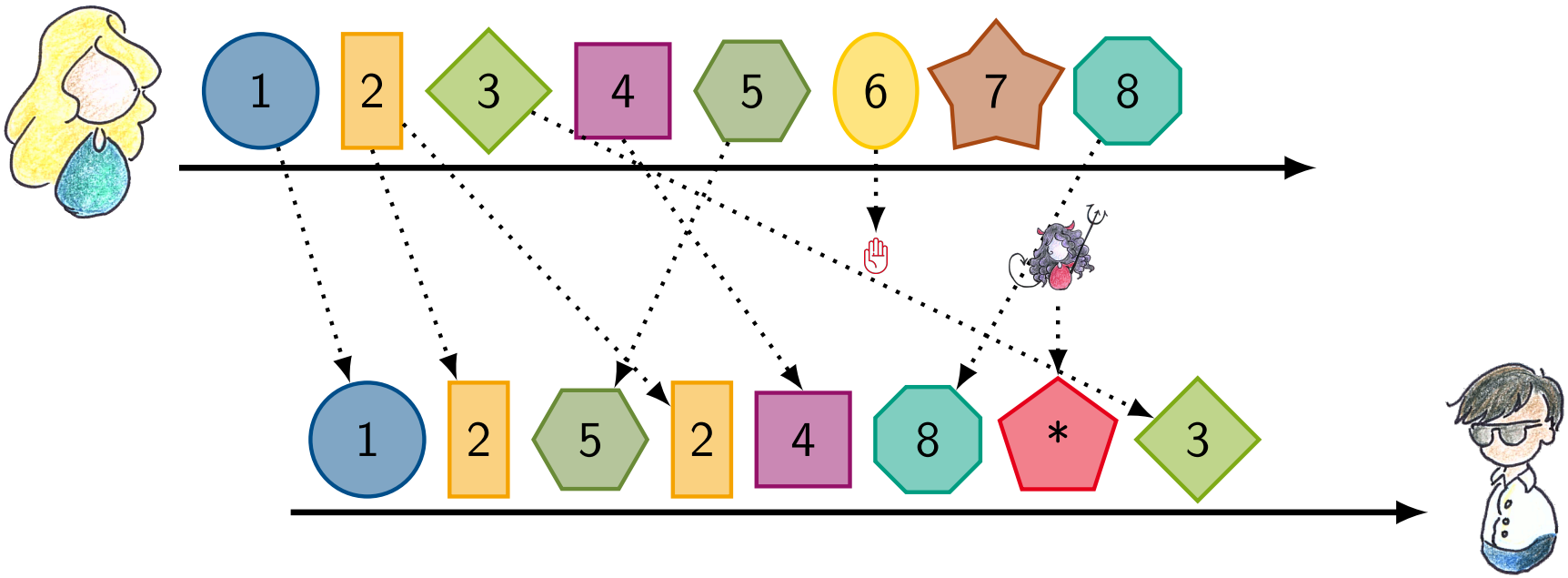
ETH zürich



DTLS 1.3 Record Protocol Security

. . . a **robust** channel over **unreliable transport** UDP [FGJ20]

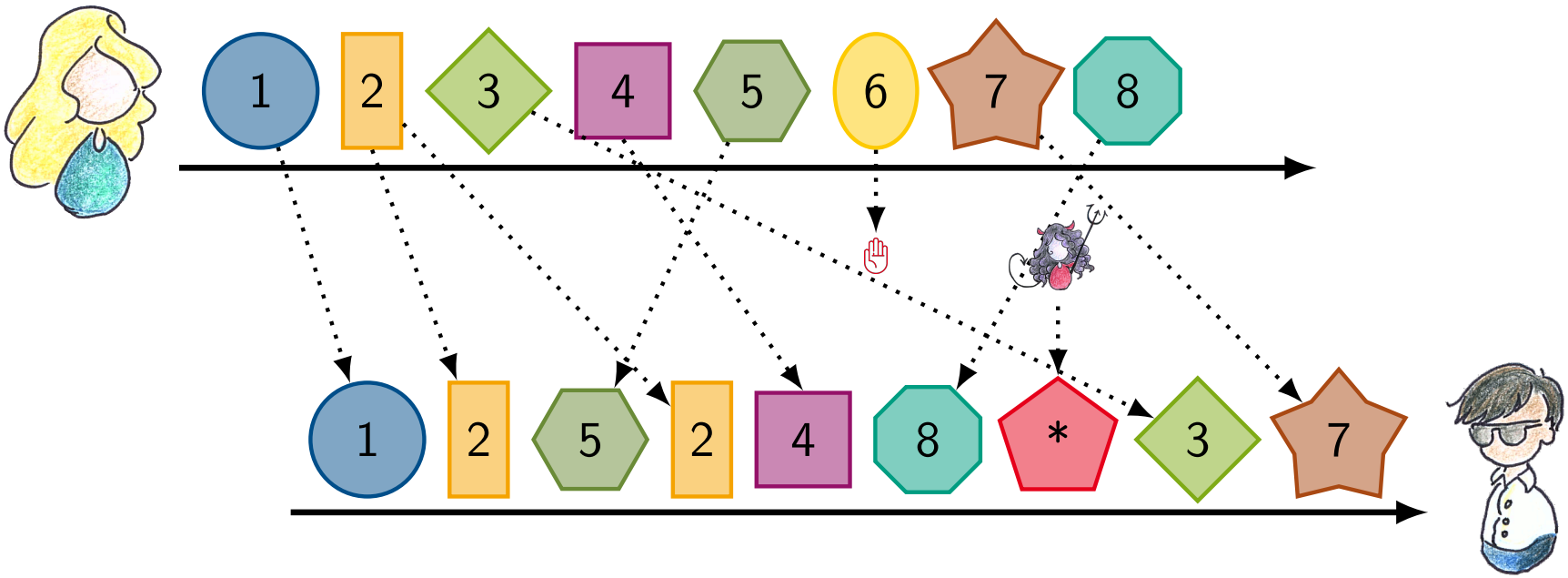
ETH zürich



DTLS 1.3 Record Protocol Security

. . . a **robust** channel over **unreliable transport** UDP [FGJ20]

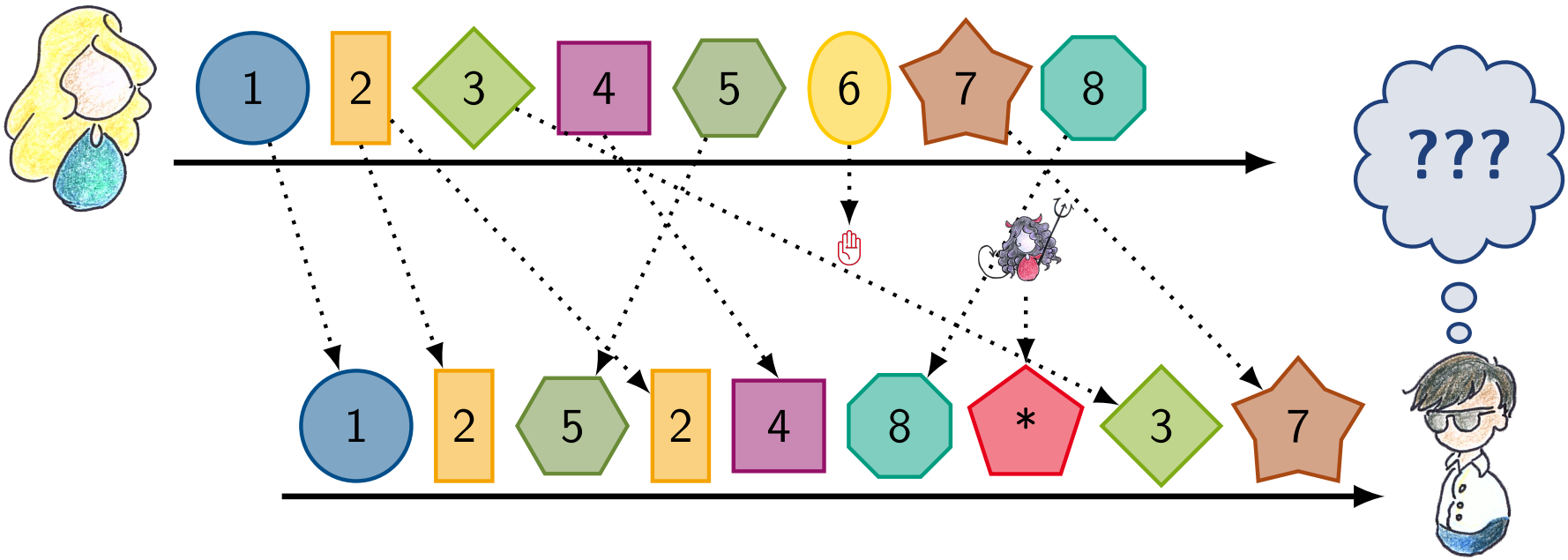
ETH zürich



DTLS 1.3 Record Protocol Security

. . . a **robust** channel over **unreliable transport** UDP [FGJ20]

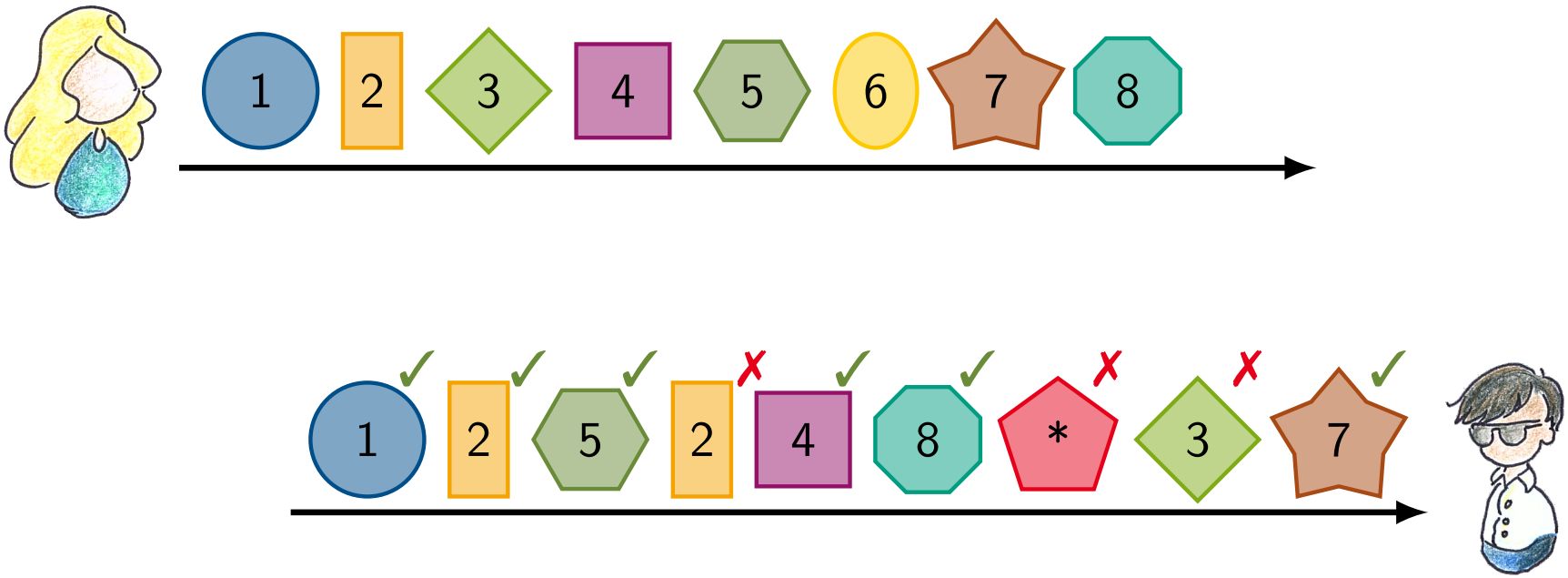
ETH zürich



DTLS 1.3 Record Protocol Security

. . . a **robust** channel over **unreliable transport** UDP [FGJ20]

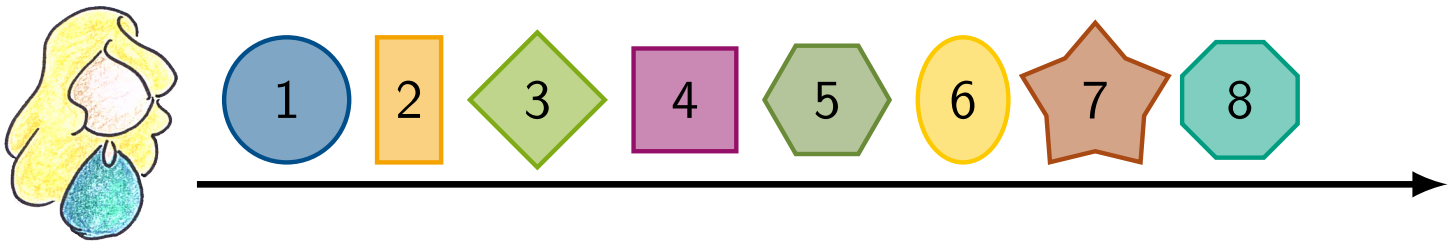
ETH zürich

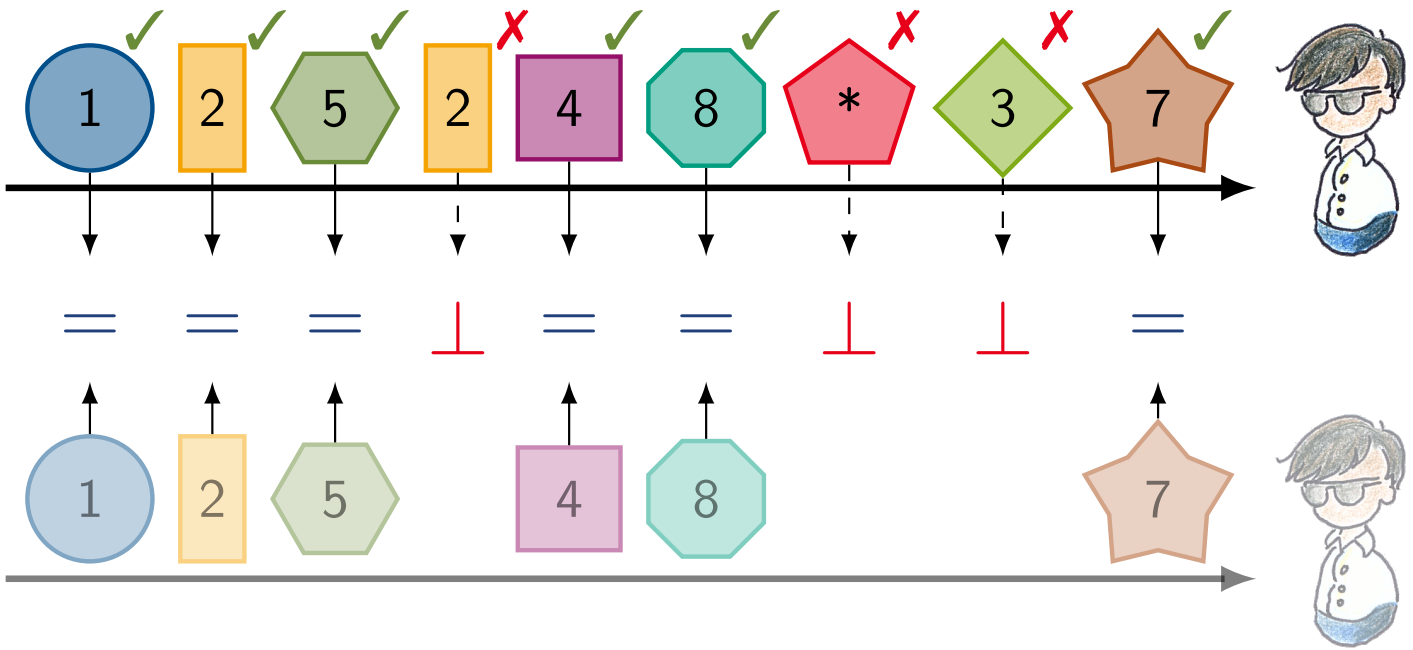


DTLS 1.3 Record Protocol Security

. . . a **robust** channel over **unreliable transport** UDP [FGJ20]

ETH zürich

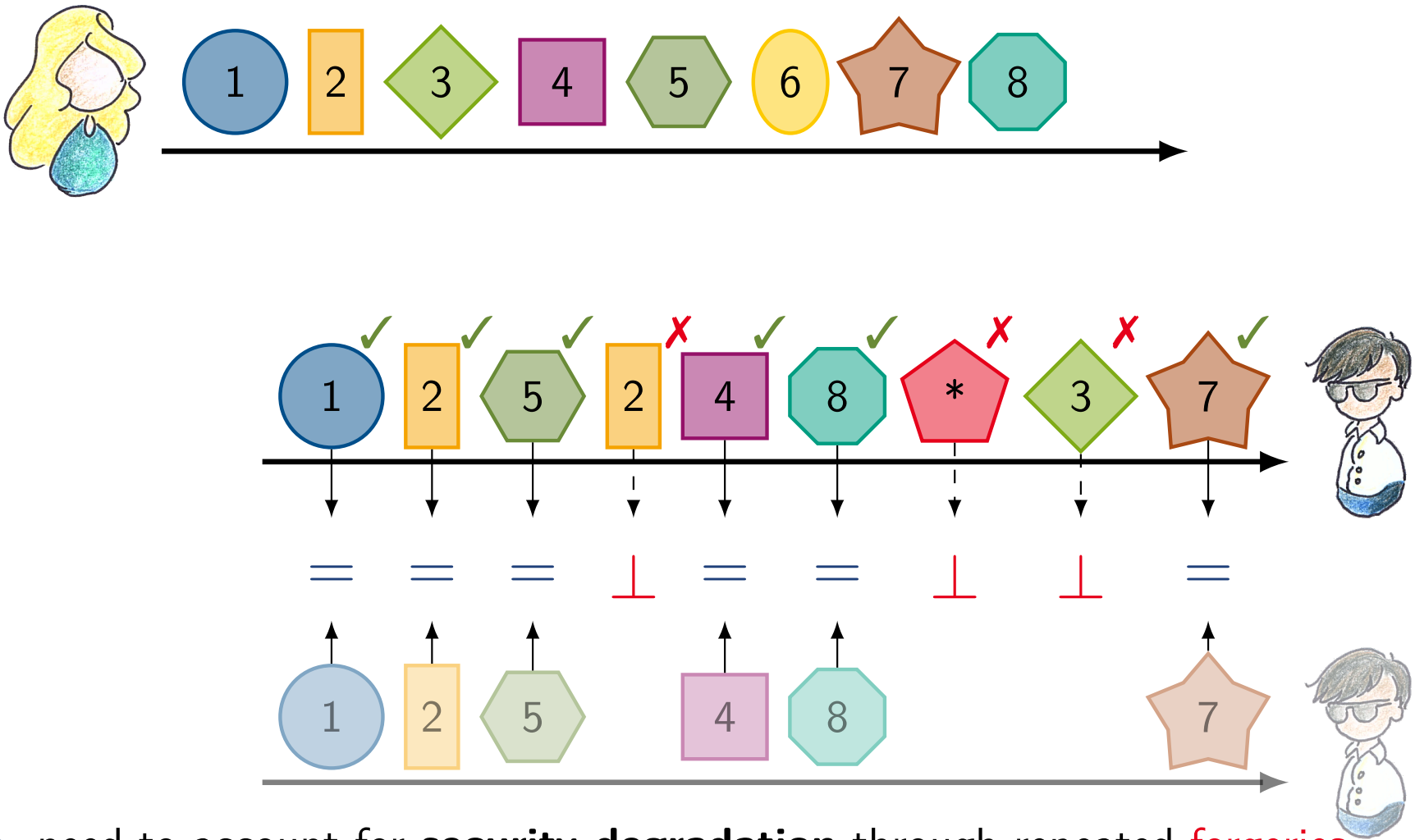




DTLS 1.3 Record Protocol Security

. . . a **robust** channel over **unreliable transport** UDP [FGJ20]

ETH zürich



- need to account for **security degradation** through repeated forgeries

Conclusions

TLS 1.3 Security Analysis

Many more analyses. . .

(a selection from the standardization process)

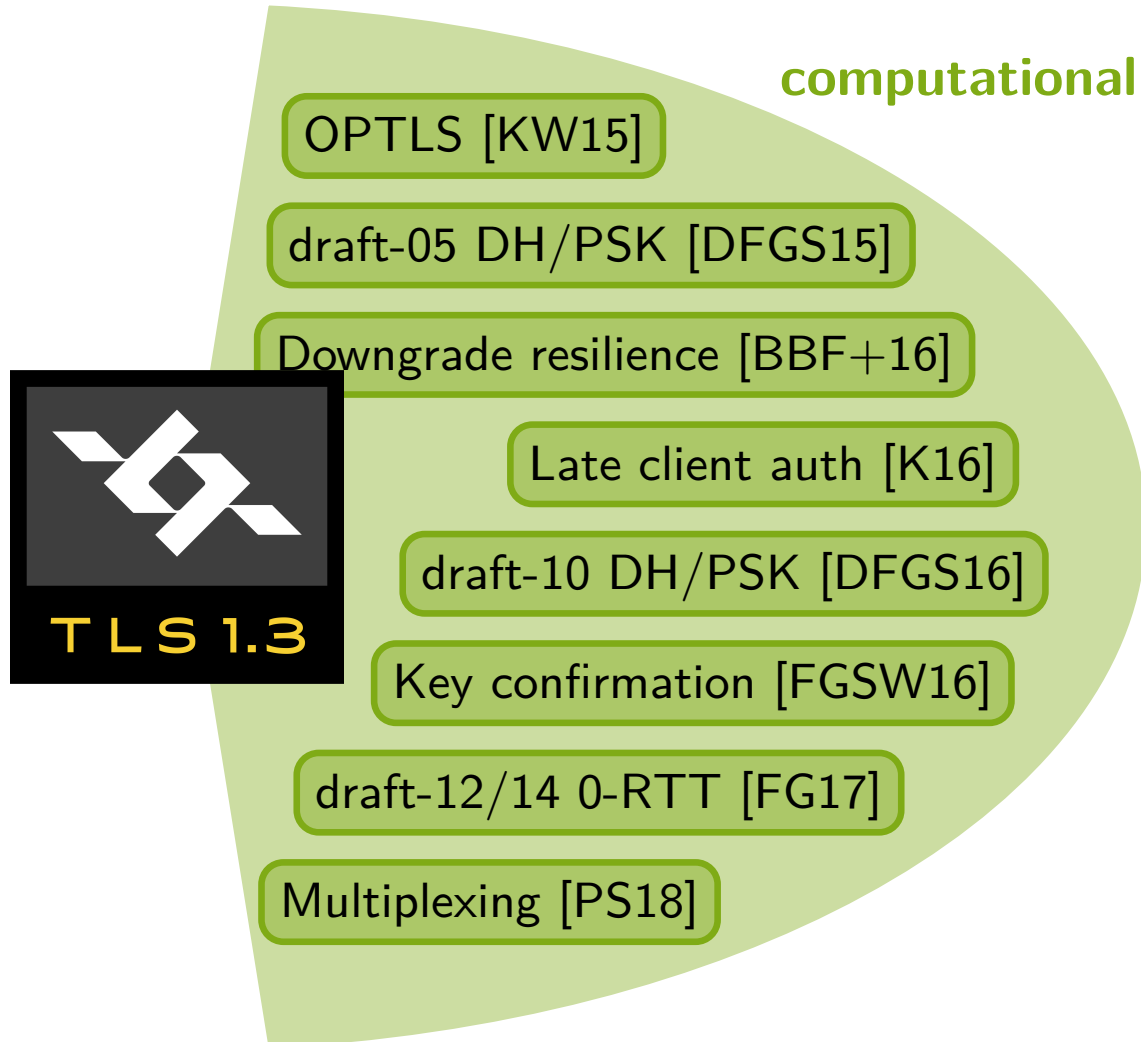
ETH zürich

TLS 1.3 Security Analysis

Many more analyses. . .

(a selection from the standardization process)

ETH zürich

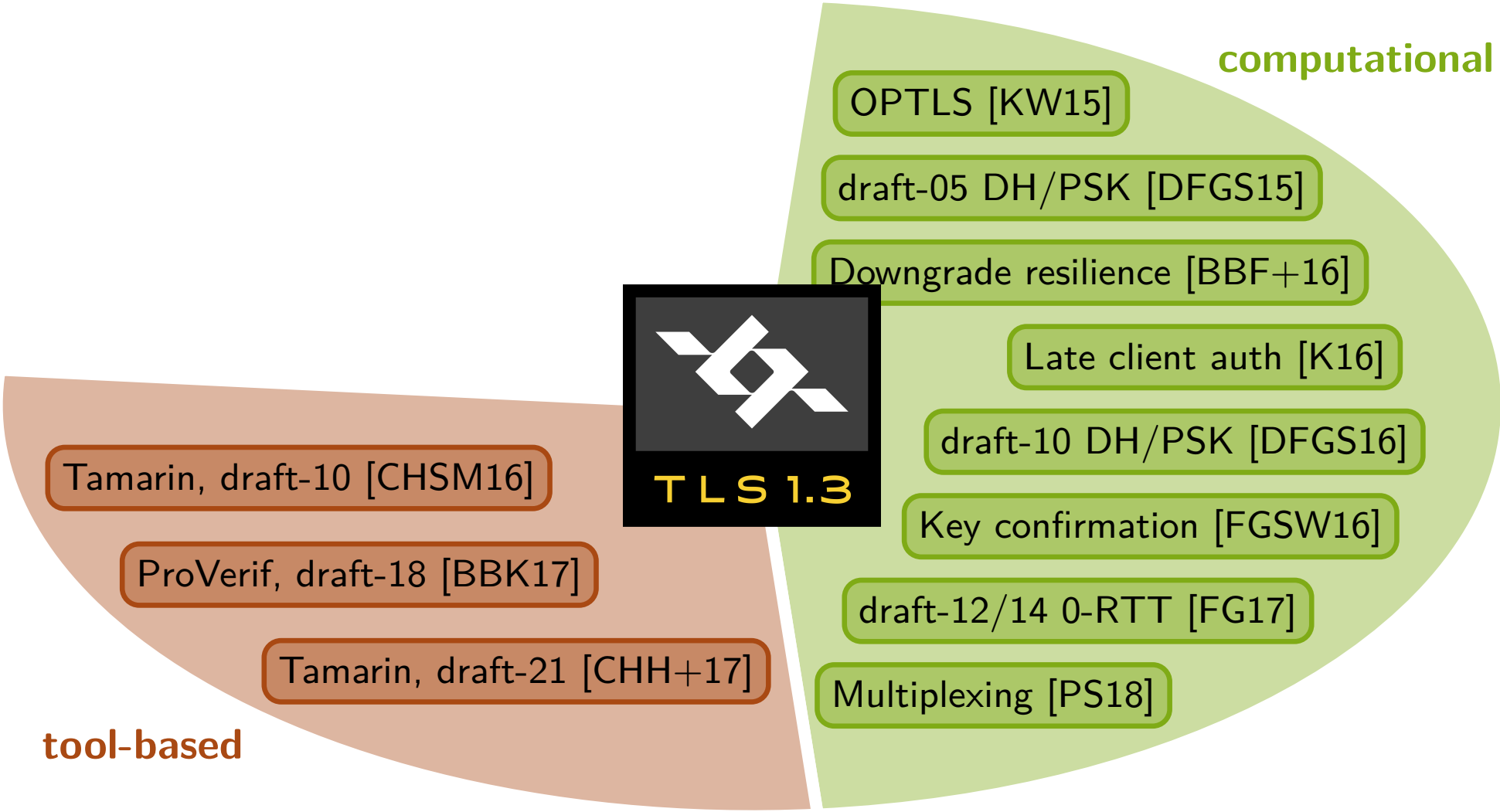


TLS 1.3 Security Analysis

Many more analyses. . .

(a selection from the standardization process)

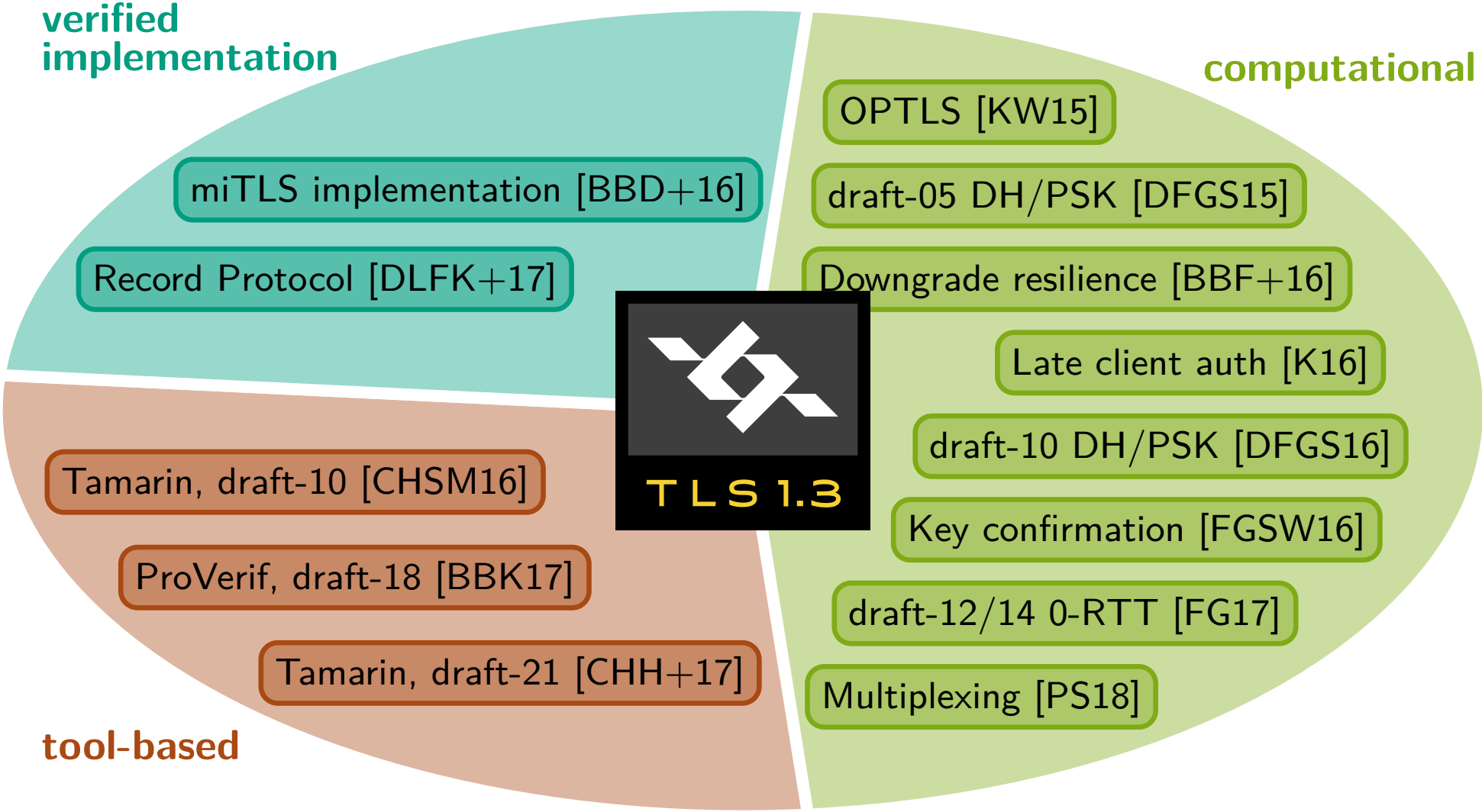
ETH zürich



TLS 1.3 Security Analysis

Many more analyses. . .
(a selection from the standardization process)

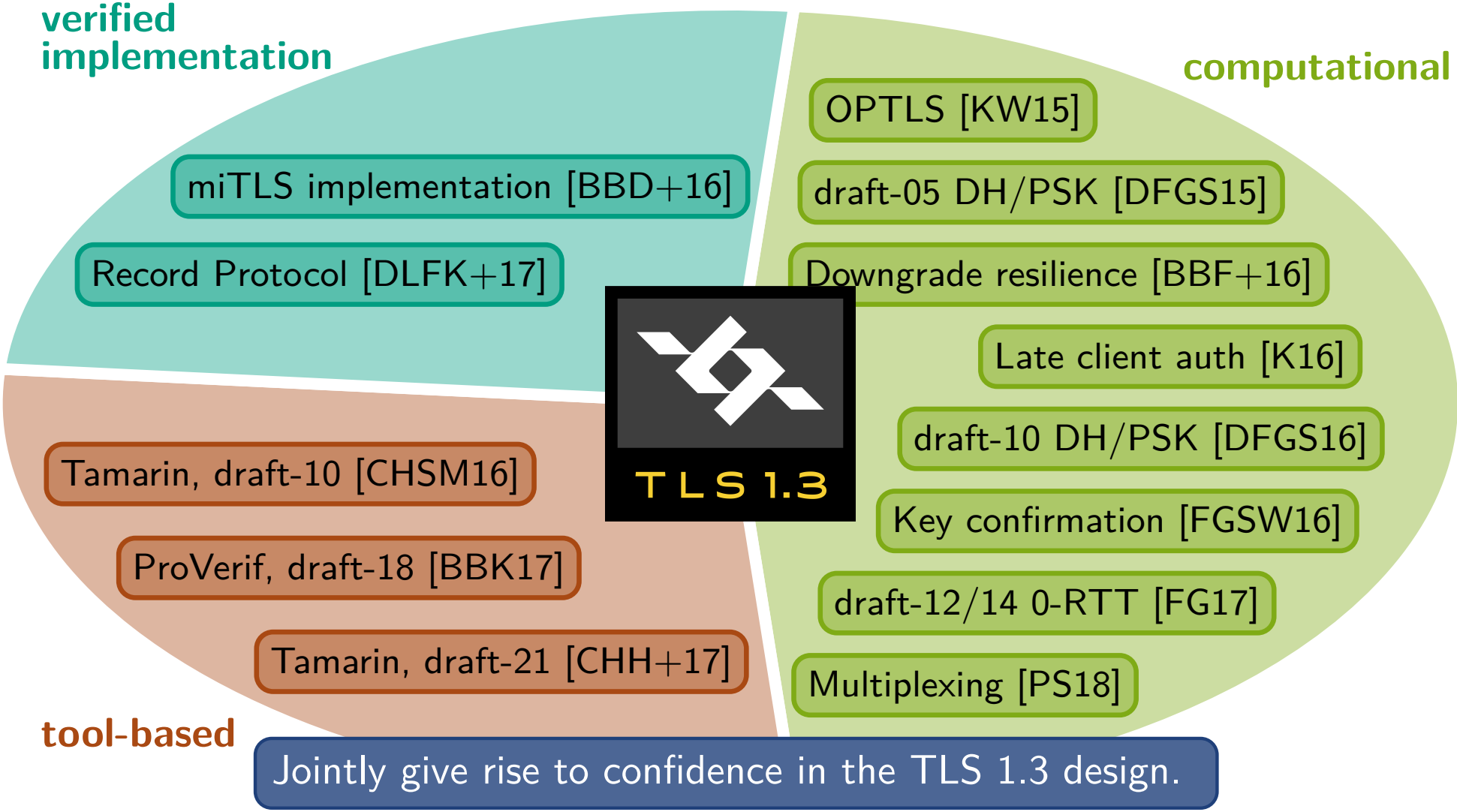
ETH zürich



TLS 1.3 Security Analysis

Many more analyses. . .
(a selection from the standardization process)

ETH zürich



TLS 1.3 Security Analysis

All good?

ETH zürich

TLS 1.3 Security Analysis

All good?

ETH zürich

Selfie: reflections on TLS 1.3 with PSK

Nir Drucker and Shay Gueron

University of Haifa, Israel,
and
Amazon, Seattle, USA

Abstract. TLS 1.3 allows two parties to establish a shared session key from an out-of-band agreed Pre Shared Key (PSK). The PSK is used to mutually authenticate the parties, under the assumption that it is not shared with others. This allows the parties to skip the certificate verification steps, saving bandwidth, communication rounds, and latency.
We identify a security vulnerability in this TLS 1.3 path, by showing a new reflection attack that we call "Selfie". The Selfie attack breaks the mutual authentication. It leverages the fact that TLS does not mandate explicit authentication of the server and the client in every message.
The paper explains the root cause of this TLS 1.3 vulnerability, demonstrates the Selfie attack on the TLS implementation of OpenSSL and proposes appropriate mitigation.
The attack is surprising because it breaks some assumptions and uncovers an interesting gap in the existing TLS security proofs. We explain the gap in the model assumptions and subsequently in the security proofs. We also provide an enhanced Multi-Stage Key Exchange (MSKE) model that captures the additional required assumptions of TLS 1.3 in its current state. The resulting security claims in the case of external PSKs are accordingly different.

Keywords: TLS 1.3 · Selfie Attack · Reflection attack · Network security · Multi-Stage Key Exchange model

Selfie attack [DG21]

TLS 1.3 Security Analysis

All good?

ETH zürich

Selfie: reflections on TLS 1.3 with PSK

Nir Drucker and Shay Gueron

University of Haifa, Israel,
and
Amazon, Seattle, USA

Abstract. TLS 1.3 allows two parties to establish a shared session key from an out-of-band agreed Pre Shared Key (PSK). The PSK is used to mutually authenticate the parties, under the assumption that it is not shared with others. This allows the parties to skip the certificate verification steps, saving bandwidth, communication rounds, and latency.
We identify a security vulnerability in this TLS 1.3 path, by showing a new reflection attack that we call "Selfie". The Selfie attack breaks the mutual authentication. It leverages the fact that TLS does not mandate explicit authentication of the server and the client in every message.
The paper explains the root cause of this TLS 1.3 vulnerability, demonstrates the Selfie attack on the TLS implementation of OpenSSL and proposes appropriate mitigation.
The attack is surprising because it breaks some assumptions and uncovers an interesting gap in the existing TLS security proofs. We explain the gap in the model assumptions and subsequently in the security proofs. We also provide an enhanced Multi-Stage Key Exchange (MSKE) model that captures the additional required assumptions of TLS 1.3 in its current state. The resulting security claims in the case of external PSKs are accordingly different.

Keywords: TLS 1.3 · Selfie Attack · Reflection attack · Network security · Multi-Stage Key Exchange model

Selfie attack [DG21]

Models are just models. . .
("All models are wrong but some are useful." — George Box)

Conclusions

ETH zürich

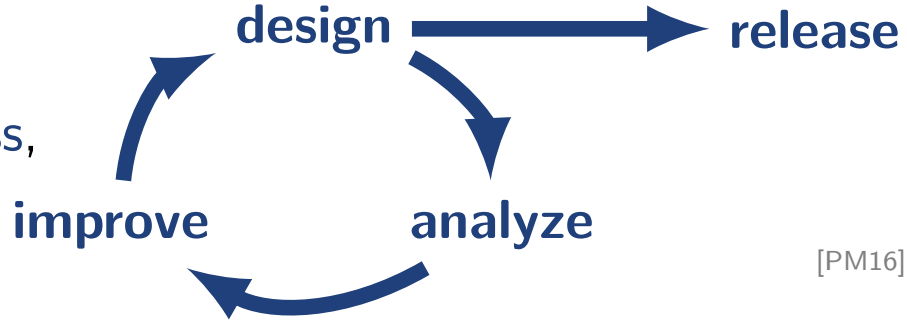
Conclusions

TLS 1.3

ETH zürich

TLS 1.3

- commendable standardization process, highly interactive



[PM16]

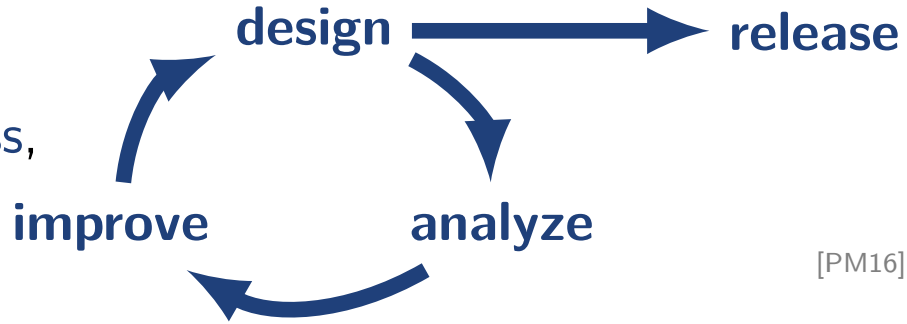
Conclusions

TLS 1.3

ETH zürich

TLS 1.3

- commendable standardization process, highly interactive
- sound cryptographic design
  - improving substantially over prior versions
  - yet with possibly easy-to-misuse 0-RTT mode



[PM16]

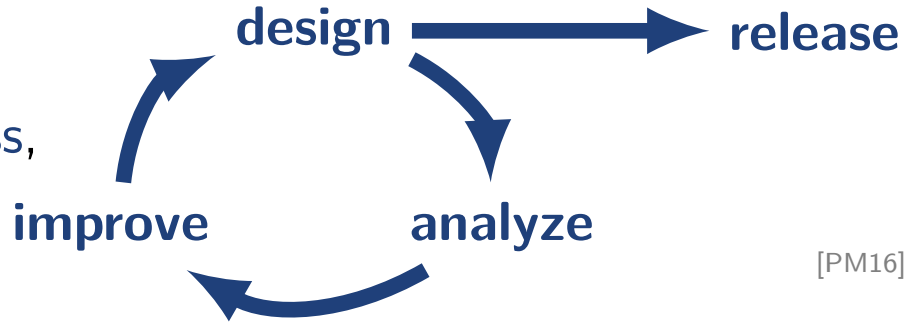
Conclusions

TLS 1.3

ETH zürich

TLS 1.3

- commendable standardization process, highly interactive
- sound cryptographic design
  - improving substantially over prior versions
  - yet with possibly easy-to-misuse 0-RTT mode



[PM16]

Design & Research

- crypto protocol design is highly complex
  - even when from "boring crypto" components (that's a plus!)
  - even when looking only at the "cryptographic core"

References I

ETH zürich

[Adr+15] D. Adrian, K. Bhargavan, Z. Durumeric, P. Gaudry, M. Green, J. A. Halderman, N. Heninger, D. Springall, E. Thomé, L. Valenta, B. VanderSloot, E. Wustrow, S. Zanella-Béguelin, and P. Zimmermann. "Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice". In: *ACM CCS 2015*. Ed. by I. Ray, N. Li, and C. Kruegel. ACM Press, Oct. 2015, pp. 5–17.

[AP13] N. J. AlFardan and K. G. Paterson. "Lucky Thirteen: Breaking the TLS and DTLS Record Protocols". In: *2013 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2013, pp. 526–540.

[Avi+16] N. Aviram, S. Schinzel, J. Somorovsky, N. Heninger, M. Dankel, J. Steube, L. Valenta, D. Adrian, J. A. Halderman, V. Dukhovni, E. Käsper, S. Cohney, S. Engels, C. Paar, and Y. Shavitt. "DROWN: Breaking TLS with SSLv2". In: *25th USENIX Security Symposium*. Aug. 2016.

[BKN02] M. Bellare, T. Kohno, and C. Namprempre. "Authenticated Encryption in SSH: Provably Fixing The SSH Binary Packet Protocol". In: *ACM CCS 2002*. Ed. by V. Atluri. ACM Press, Nov. 2002, pp. 1–11.

[BN00] M. Bellare and C. Namprempre. "Authenticated Encryption: Relations among notions and analysis of the generic composition paradigm". In: *ASIACRYPT 2000*. Ed. by T. Okamoto. Vol. 1976. LNCS. Springer, Heidelberg, Dec. 2000, pp. 531–545.

[BR94] M. Bellare and P. Rogaway. "Entity Authentication and Key Distribution". In: *CRYPTO'93*. Ed. by D. R. Stinson. Vol. 773. LNCS. Springer, Heidelberg, Aug. 1994, pp. 232–249.

[Ble98] D. Bleichenbacher. "Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS #1". In: *CRYPTO'98*. Ed. by H. Krawczyk. Vol. 1462. LNCS. Springer, Heidelberg, Aug. 1998, pp. 1–12.

References II

ETH zürich

[DH76] W. Diffie and M. E. Hellman. "New Directions in Cryptography". In: *IEEE Transactions on Information Theory* 22.6 (1976), pp. 644–654.

[Dow+15] B. Dowling, M. Fischlin, F. Günther, and D. Stebila. "A Cryptographic Analysis of the TLS 1.3 Handshake Protocol Candidates". In: *ACM CCS 2015*. Ed. by I. Ray, N. Li, and C. Kruegel. ACM Press, Oct. 2015, pp. 1197–1210.

[Dow+16] B. Dowling, M. Fischlin, F. Günther, and D. Stebila. *A Cryptographic Analysis of the TLS 1.3 draft-10 Full and Pre-shared Key Handshake Protocol*. Cryptology ePrint Archive, Report 2016/081. `https://eprint.iacr.org/2016/081`. 2016.

[Dow+21] B. Dowling, M. Fischlin, F. Günther, and D. Stebila. "A Cryptographic Analysis of the TLS 1.3 Handshake Protocol". In: *Journal of Cryptology* 34.4 (Oct. 2021), p. 37.

[DG21] N. Drucker and S. Gueron. "Selfie: reflections on TLS 1.3 with PSK". In: *Journal of Cryptology* 34.3 (July 2021), p. 27.

[FG14] M. Fischlin and F. Günther. "Multi-Stage Key Exchange and the Case of Google's QUIC Protocol". In: *ACM CCS 2014*. Ed. by G.-J. Ahn, M. Yung, and N. Li. ACM Press, Nov. 2014, pp. 1193–1204.

[FG17] M. Fischlin and F. Günther. "Replay Attacks on Zero Round-Trip Time: The Case of the TLS 1.3 Handshake Candidates". In: *2017 IEEE European Symposium on Security and Privacy, EuroS&P 2017*. Paris, France: IEEE, 2017, pp. 60–75.

[FGJ20] M. Fischlin, F. Günther, and C. Janson. *Robust Channels: Handling Unreliable Networks in the Record Layers of QUIC and DTLS 1.3*. Available as Cryptology ePrint Archive Report 2020/718, `https://eprint.iacr.org/2020/718`. 2020.

References III

ETH zürich

[Gün18] F. Günther. "Modeling Advanced Security Aspects of Key Exchange and Secure Channel Protocols". http://tuprints.ulb.tu-darmstadt.de/7162/. PhD thesis. Darmstadt, Germany: Technische Universität Darmstadt, 2018.

[GM17] F. Günther and S. Mazaheri. "A Formal Treatment of Multi-key Channels". In: *CRYPTO 2017, Part III*. Ed. by J. Katz and H. Shacham. Vol. 10403. LNCS. Springer, Heidelberg, Aug. 2017, pp. 587–618.

[Hea] *Heartbleed bug*. http://heartbleed.com/. 2014.

[Kra03] H. Krawczyk. "SIGMA: The "SIGn-and-MAc" Approach to Authenticated Diffie-Hellman and Its Use in the IKE Protocols". In: *CRYPTO 2003*. Ed. by D. Boneh. Vol. 2729. LNCS. Springer, Heidelberg, Aug. 2003, pp. 400–425.

[PM16] K. G. Paterson and T. van der Merwe. "Reactive and Proactive Standardisation of TLS". In: *Security Standardisation Research: Third International Conference (SSR 2016)*. Ed. by L. Chen, D. A. McGrew, and C. J. Mitchell. Vol. 10074. Lecture Notes in Computer Science. Gaithersburg, MD, USA: Springer, 2016, pp. 160–186.

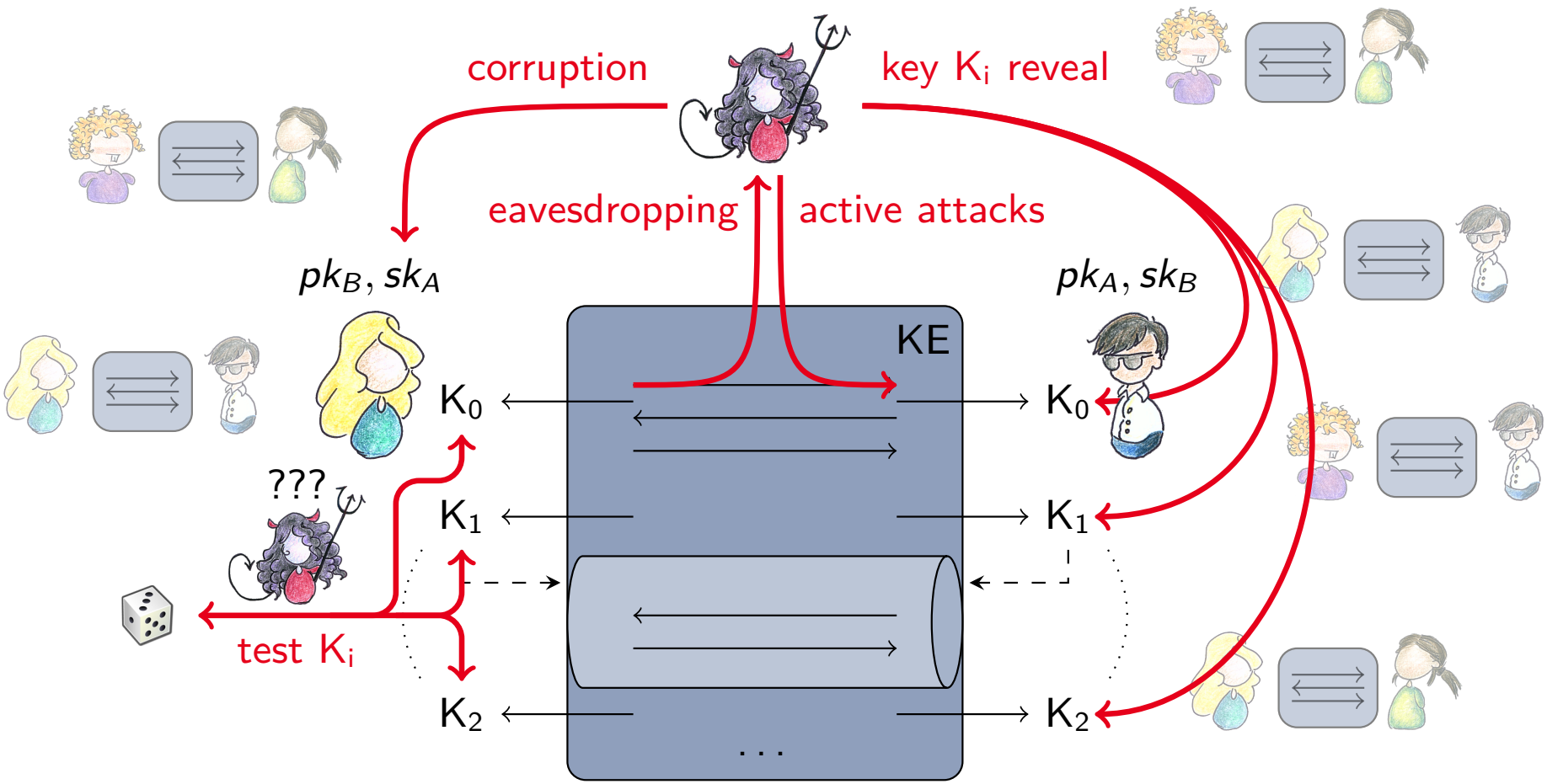
[Vau02] S. Vaudenay. "Security Flaws Induced by CBC Padding - Applications to SSL, IPSEC, WTLS...". In: *EUROCRYPT 2002*. Ed. by L. R. Knudsen. Vol. 2332. LNCS. Springer, Heidelberg, 2002, pp. 534–546.

Extra Slides

Multi-Stage Key Exchange Security

ETH zürich

(extending [Bellare–Rogaway93])

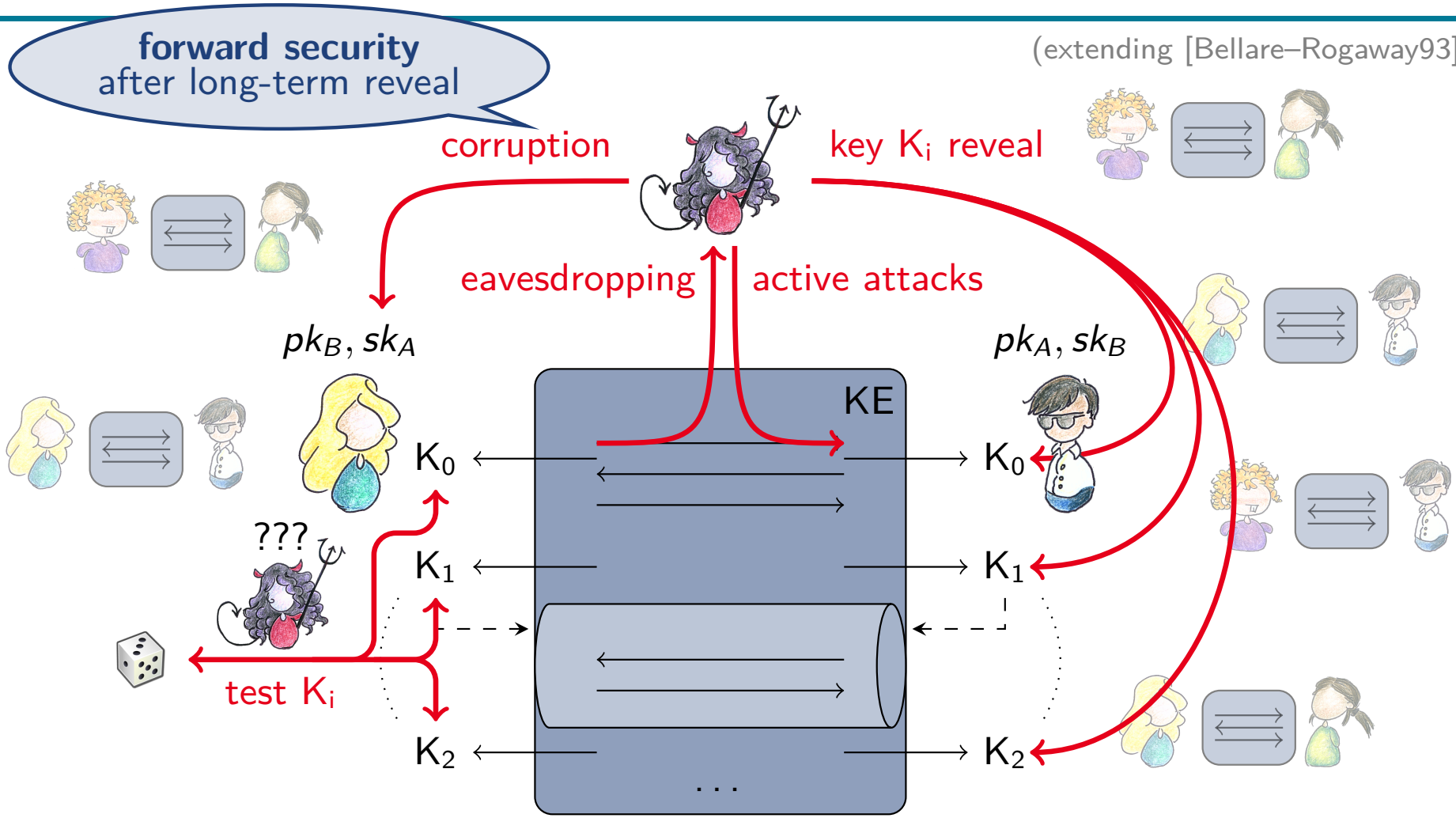


[FG14], [Dow+15], [Dow+16], [FG17], [Gün18], [Dow+21]

Multi-Stage Key Exchange Security

ETH zürich

(extending [Bellare–Rogaway93])

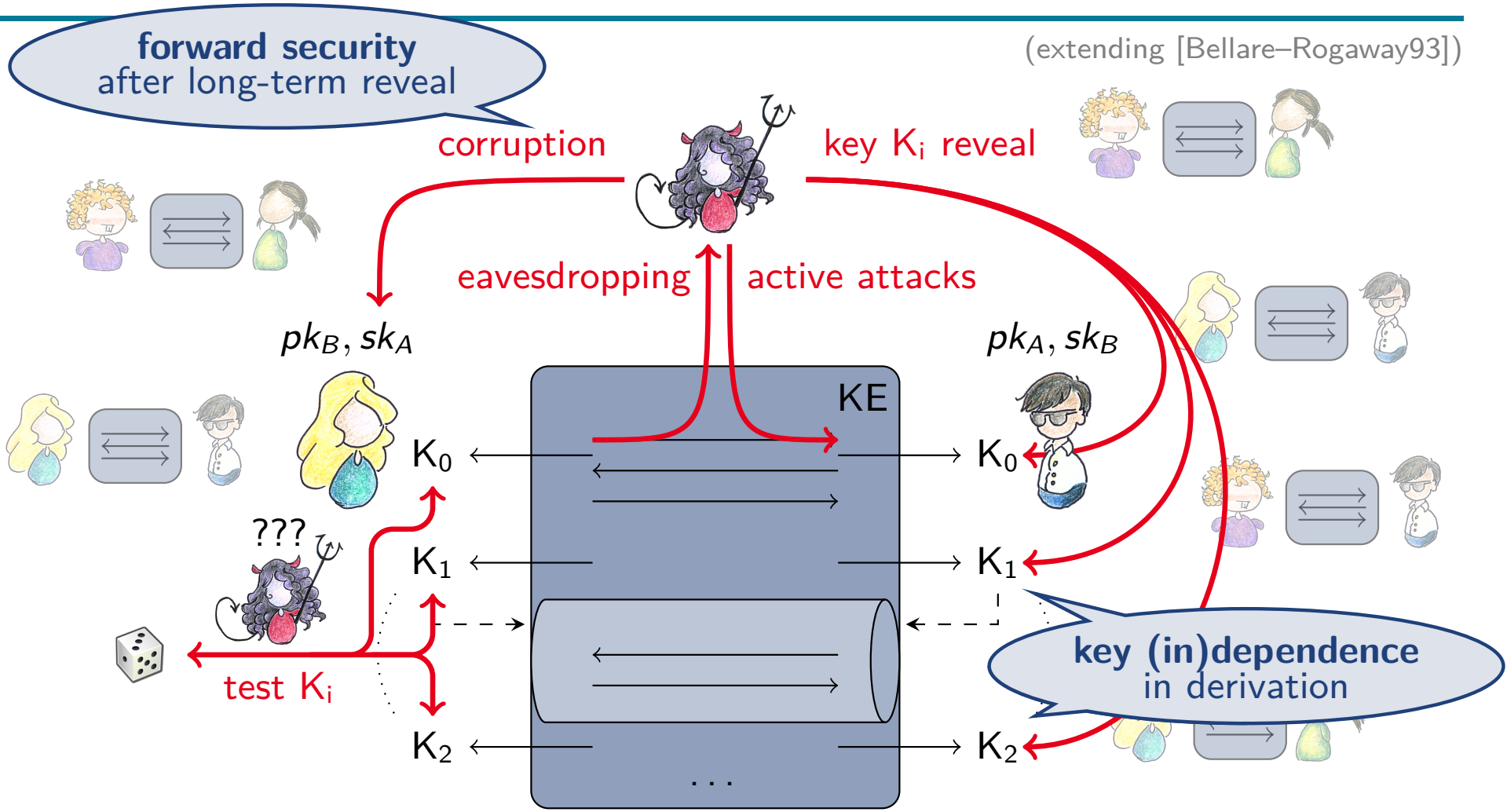


[FG14], [Dow+15], [Dow+16], [FG17], [Gün18], [Dow+21]

Multi-Stage Key Exchange Security

ETH zürich

(extending [Bellare–Rogaway93])

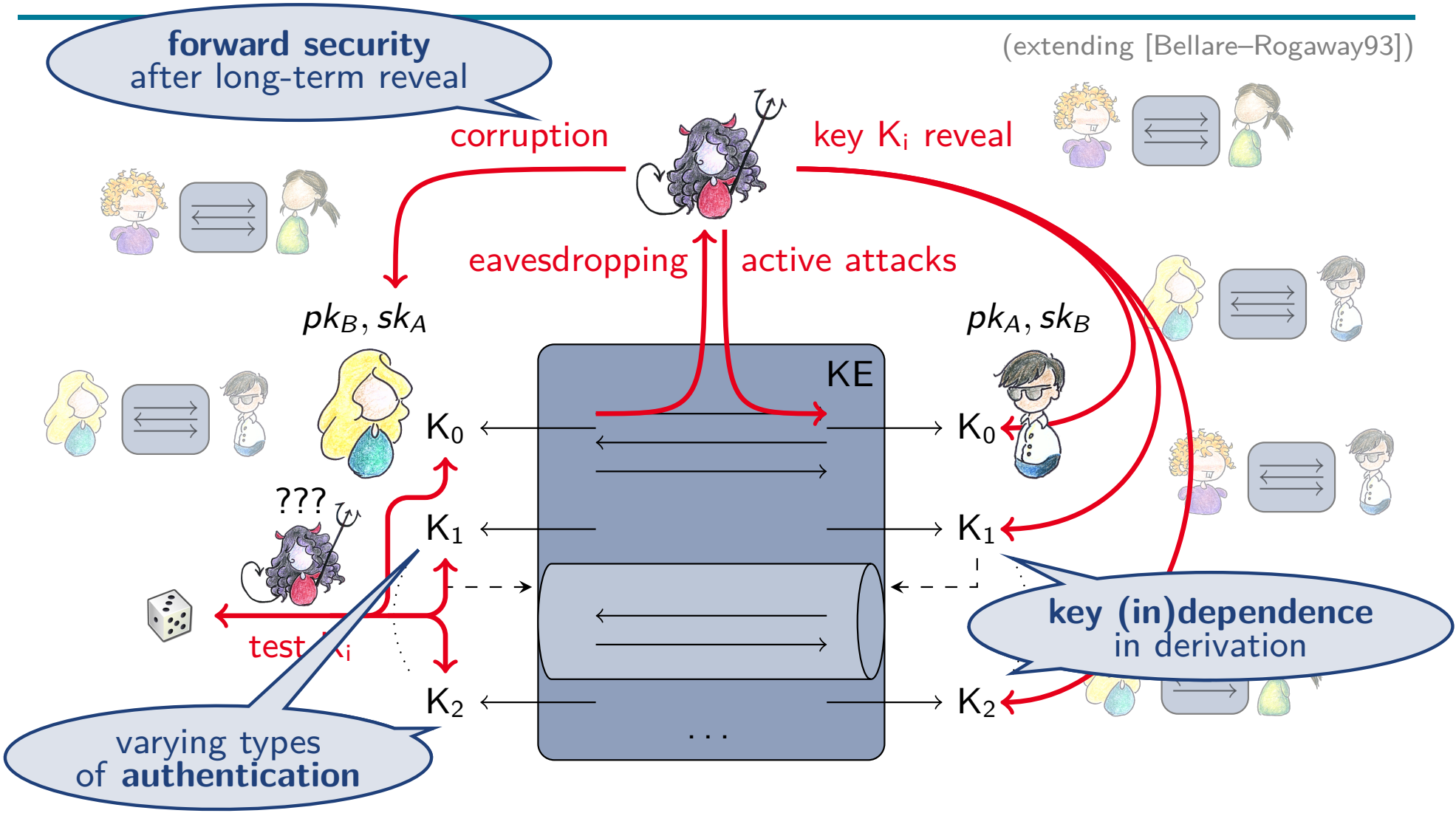


[FG14], [Dow+15], [Dow+16], [FG17], [Gün18], [Dow+21]

Multi-Stage Key Exchange Security

ETH zürich

(extending [Bellare–Rogaway93])

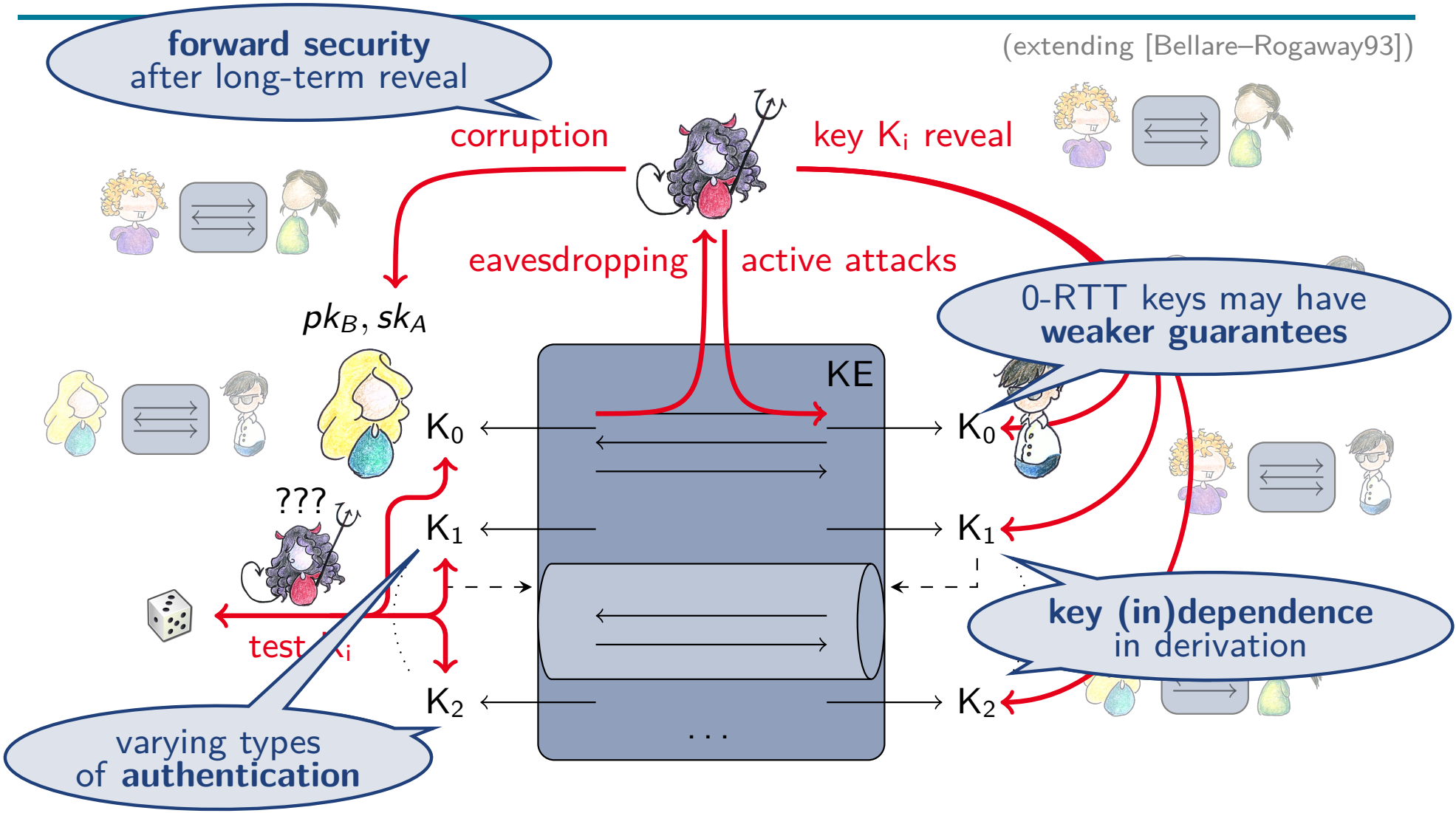


[FG14], [Dow+15], [Dow+16], [FG17], [Gün18], [Dow+21]

Multi-Stage Key Exchange Security

ETH zürich

(extending [Bellare–Rogaway93])

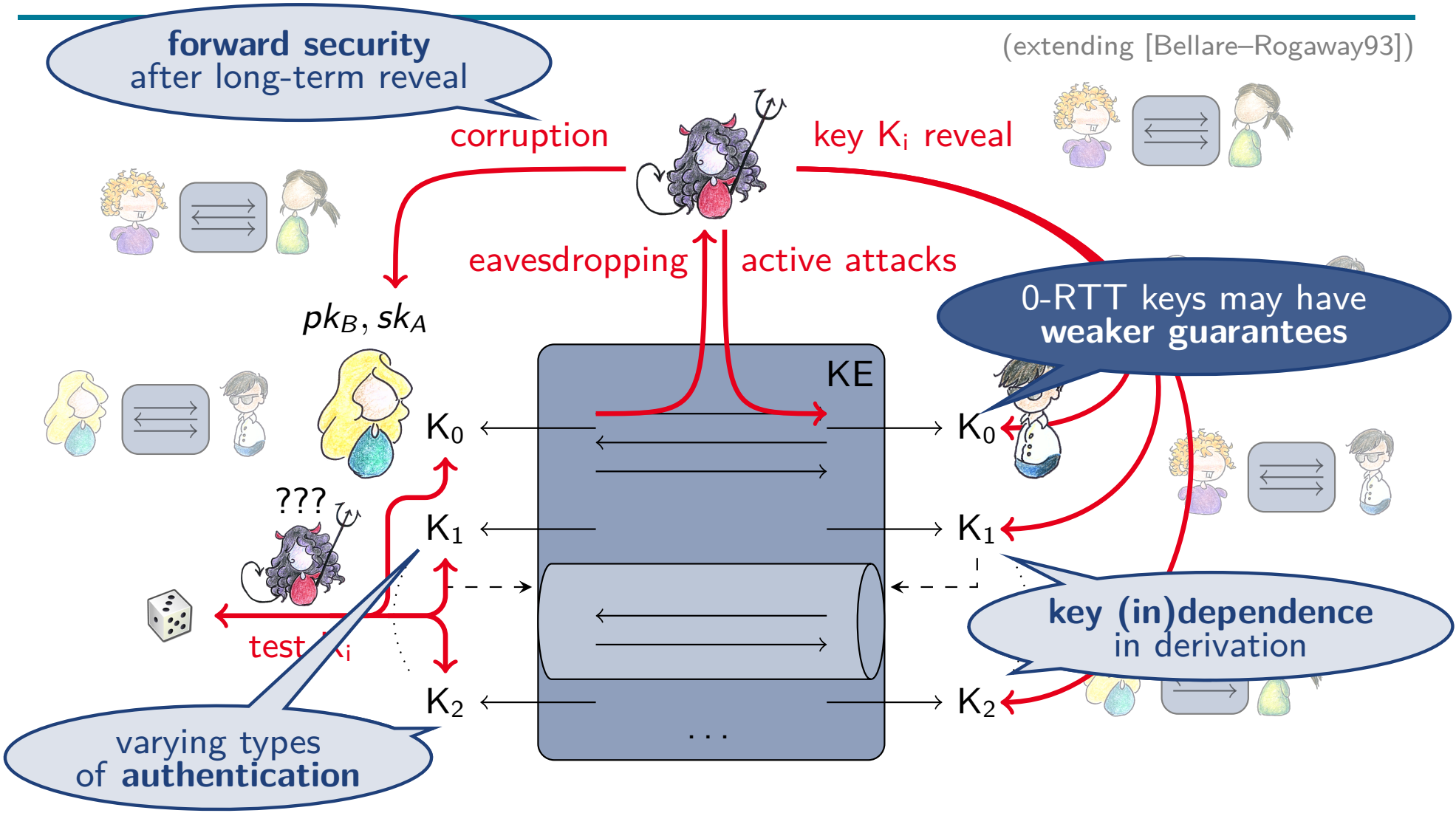


[FG14], [Dow+15], [Dow+16], [FG17], [Gün18], [Dow+21]

Multi-Stage Key Exchange Security

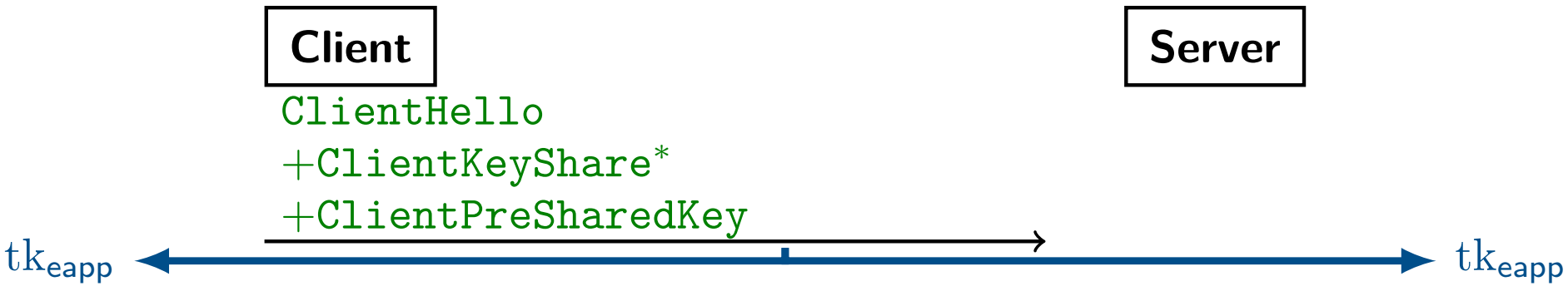
ETH zürich

(extending [Bellare–Rogaway93])



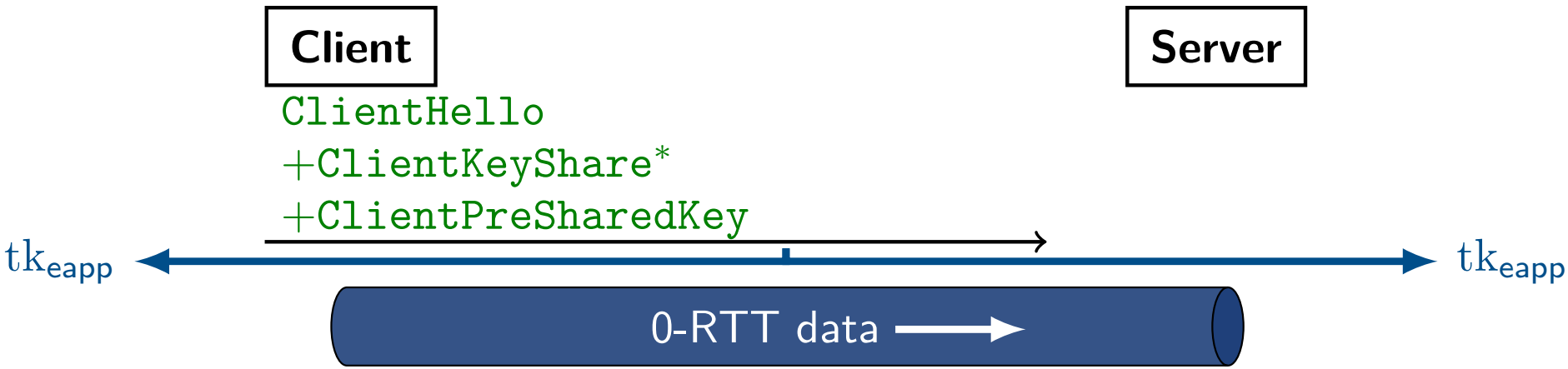
[FG14], [Dow+15], [Dow+16], [FG17], [Gün18], [Dow+21]

0-RTT and Replays



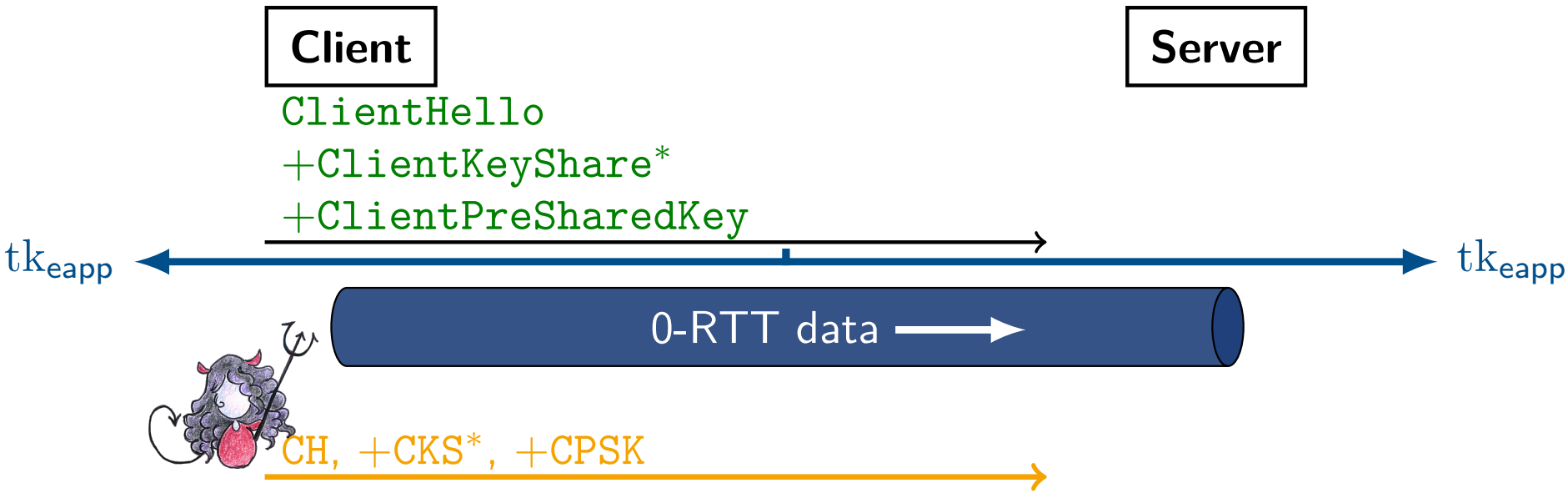
0-RTT and Replays

ETH zürich



- allows client to send data without waiting for server reply
- but without server input, how does server know the request is fresh?

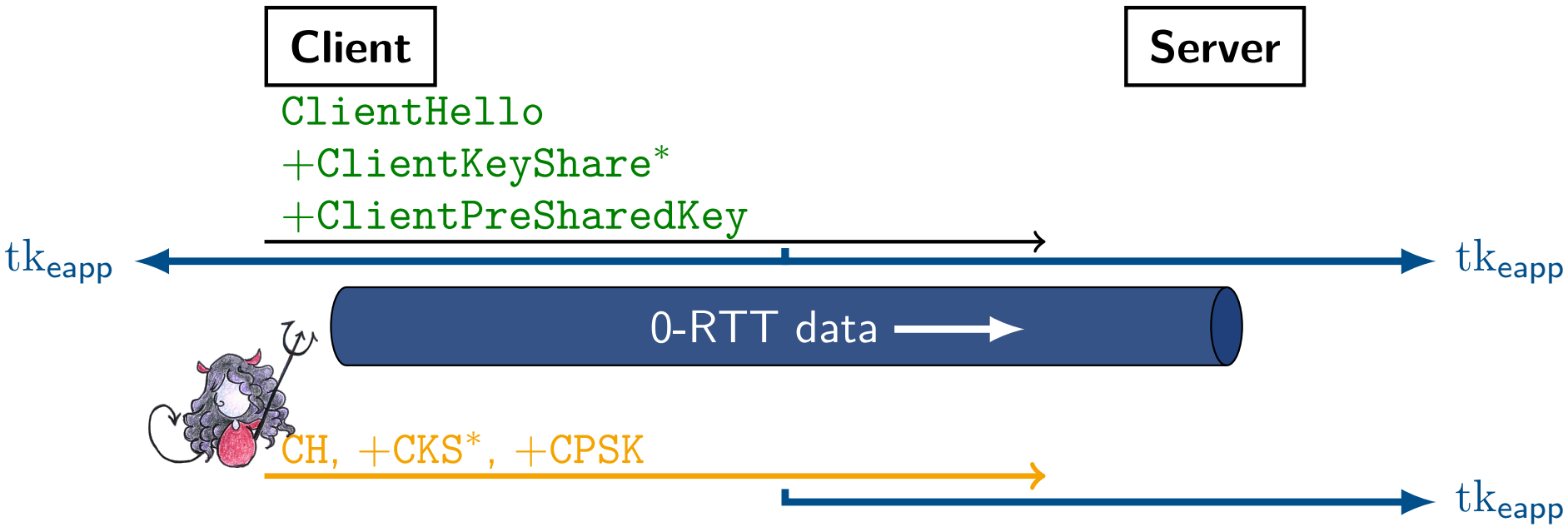
0-RTT and Replays



- allows client to send data without waiting for server reply
- but without server input, how does server know the request is fresh?

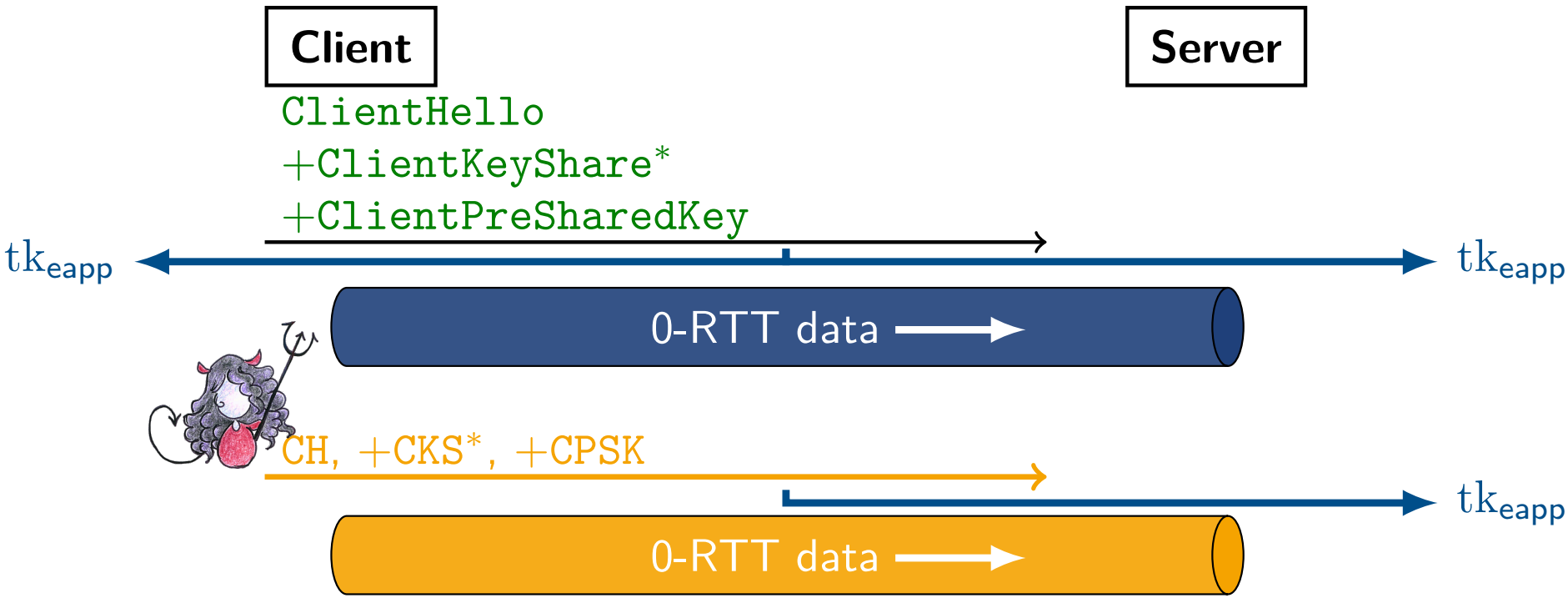
0-RTT and Replays

ETH zürich



- allows client to send data without waiting for server reply
- but without server input, how does server know the request is fresh?

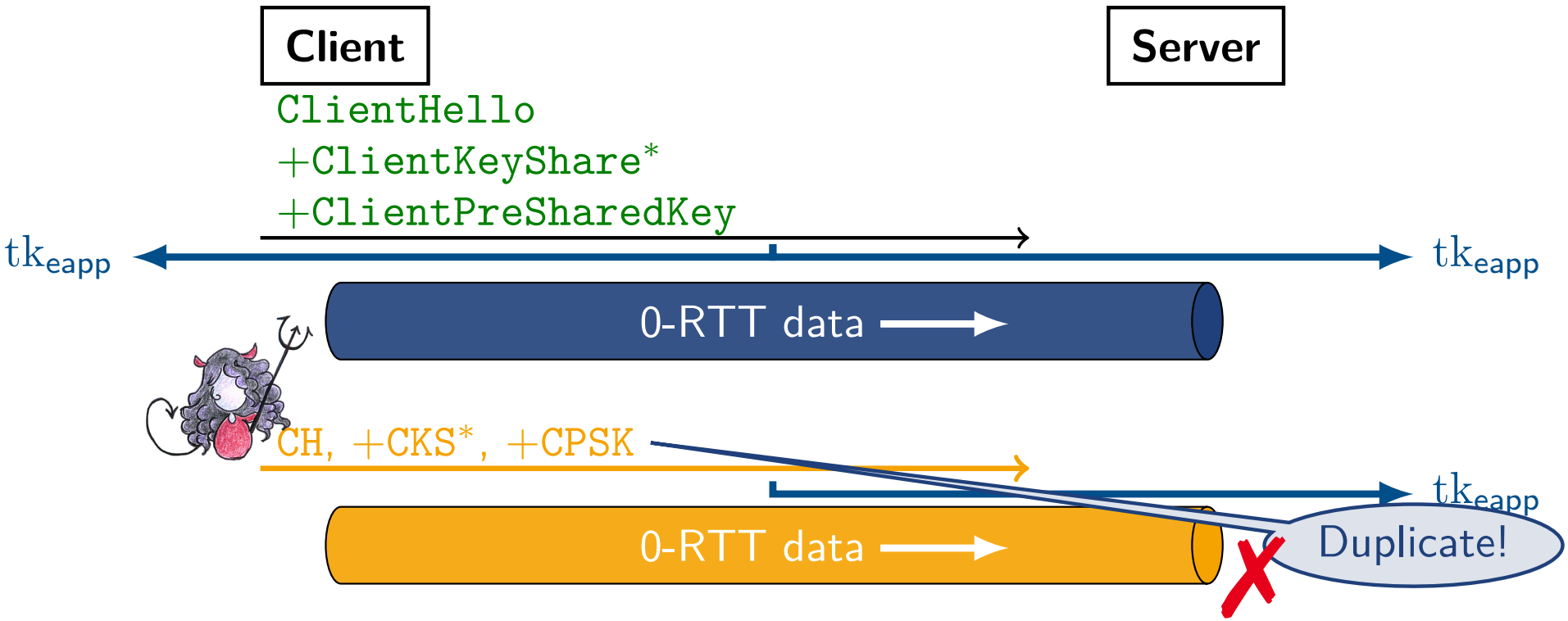
0-RTT and Replays



- allows client to send data without waiting for server reply
- but without server input, how does server know the request is fresh?
- adversary can **replay** `ClientHello` together with 0-RTT data

0-RTT and Replays

ETH zürich

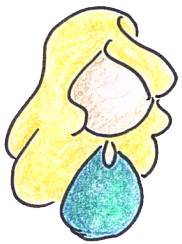


- allows client to send data without waiting for server reply
- but without server input, how does server know the request is fresh?
- adversary can **replay** `ClientHello` together with 0-RTT data
- idea: remember `ClientHello` identifier and reject duplicates

0-RTT and Replays

Generic Replay Attack on 0-RTT (by Daniel Kahn Gillmore)

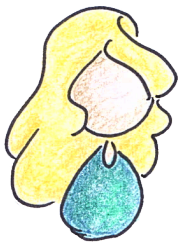
ETH zürich



0-RTT and Replays

Generic Replay Attack on 0-RTT (by Daniel Kahn Gillmore)

ETH zürich

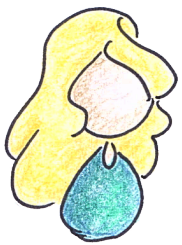


0-RTT KE msg

0-RTT and Replays

Generic Replay Attack on 0-RTT (by Daniel Kahn Gillmore)

ETH zürich



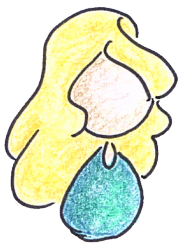
0-RTT KE msg

0-RTT request

0-RTT and Replays

Generic Replay Attack on 0-RTT (by Daniel Kahn Gillmore)

ETH zürich



0-RTT KE msg

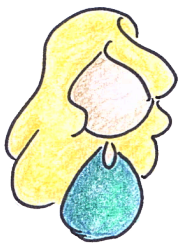
0-RTT request

process request

0-RTT and Replays

Generic Replay Attack on 0-RTT (by Daniel Kahn Gillmore)

ETH zürich



0-RTT KE msg

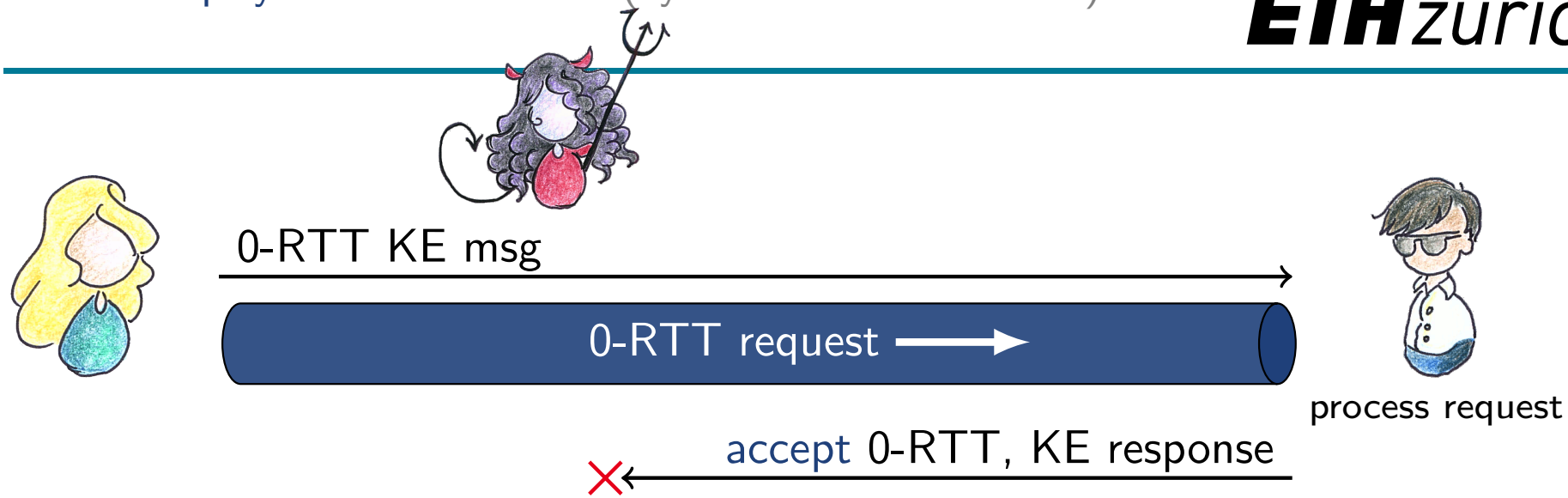
0-RTT request

process request

accept 0-RTT, KE response

0-RTT and Replays

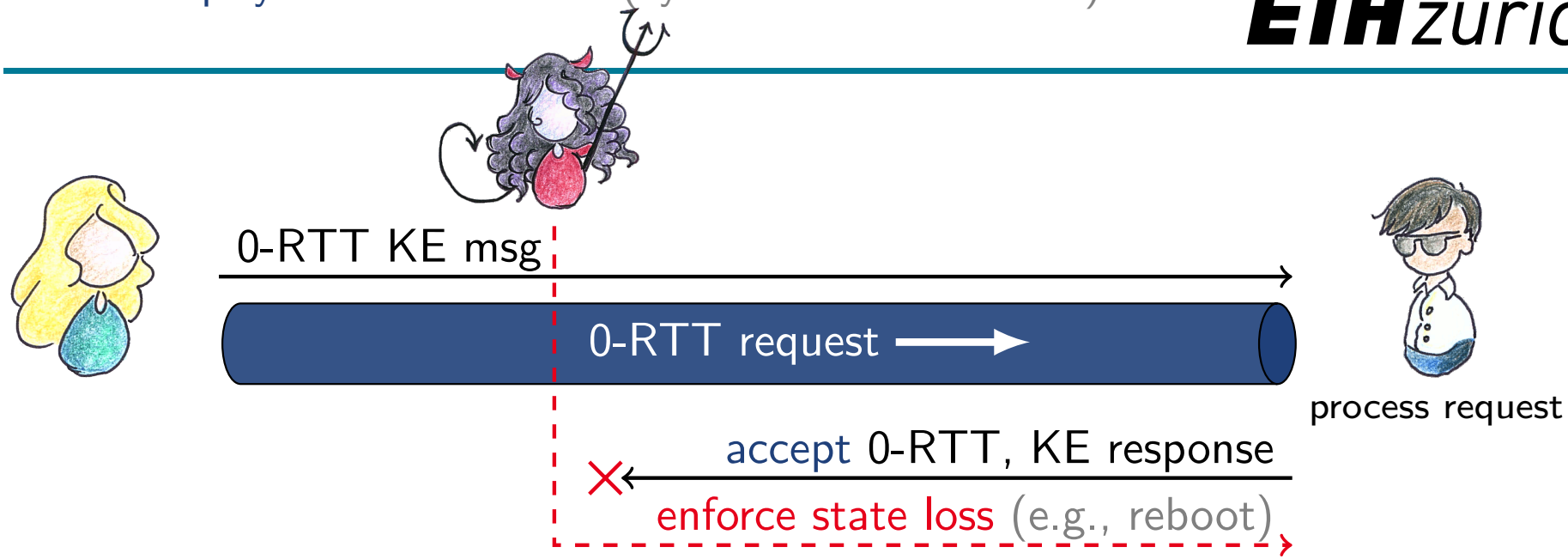
Generic Replay Attack on 0-RTT (by Daniel Kahn Gillmore)



0-RTT and Replays

Generic Replay Attack on 0-RTT (by Daniel Kahn Gillmore)

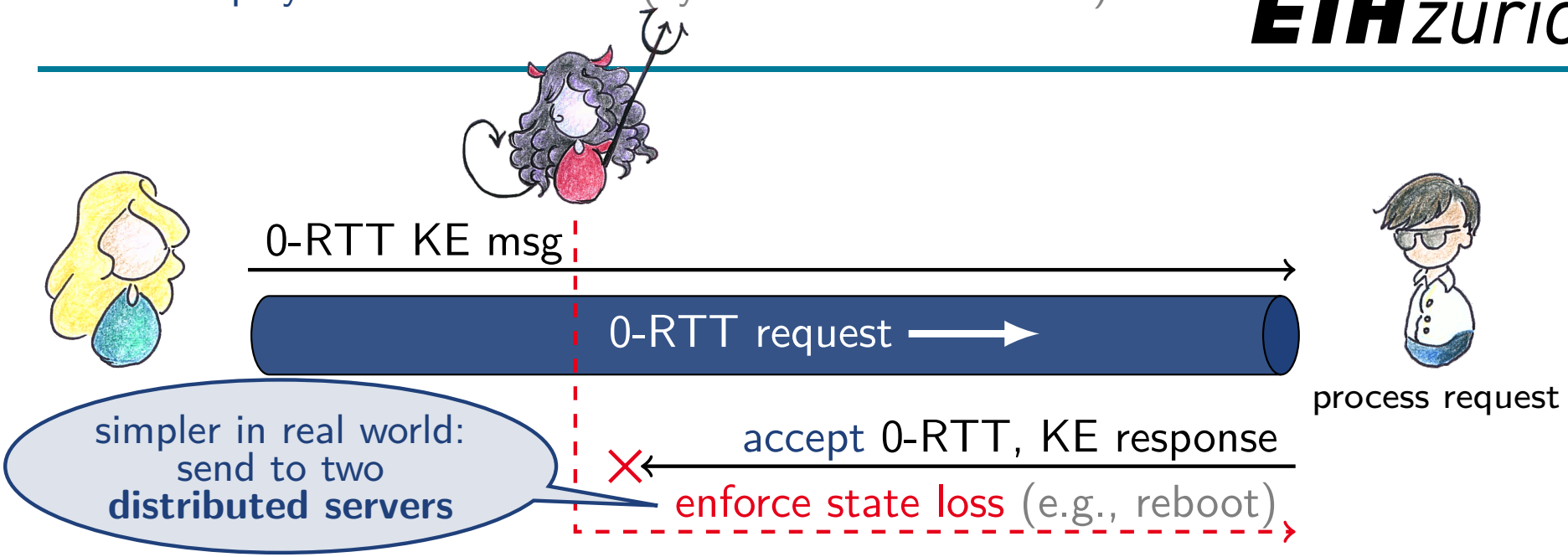
ETH zürich



0-RTT and Replays

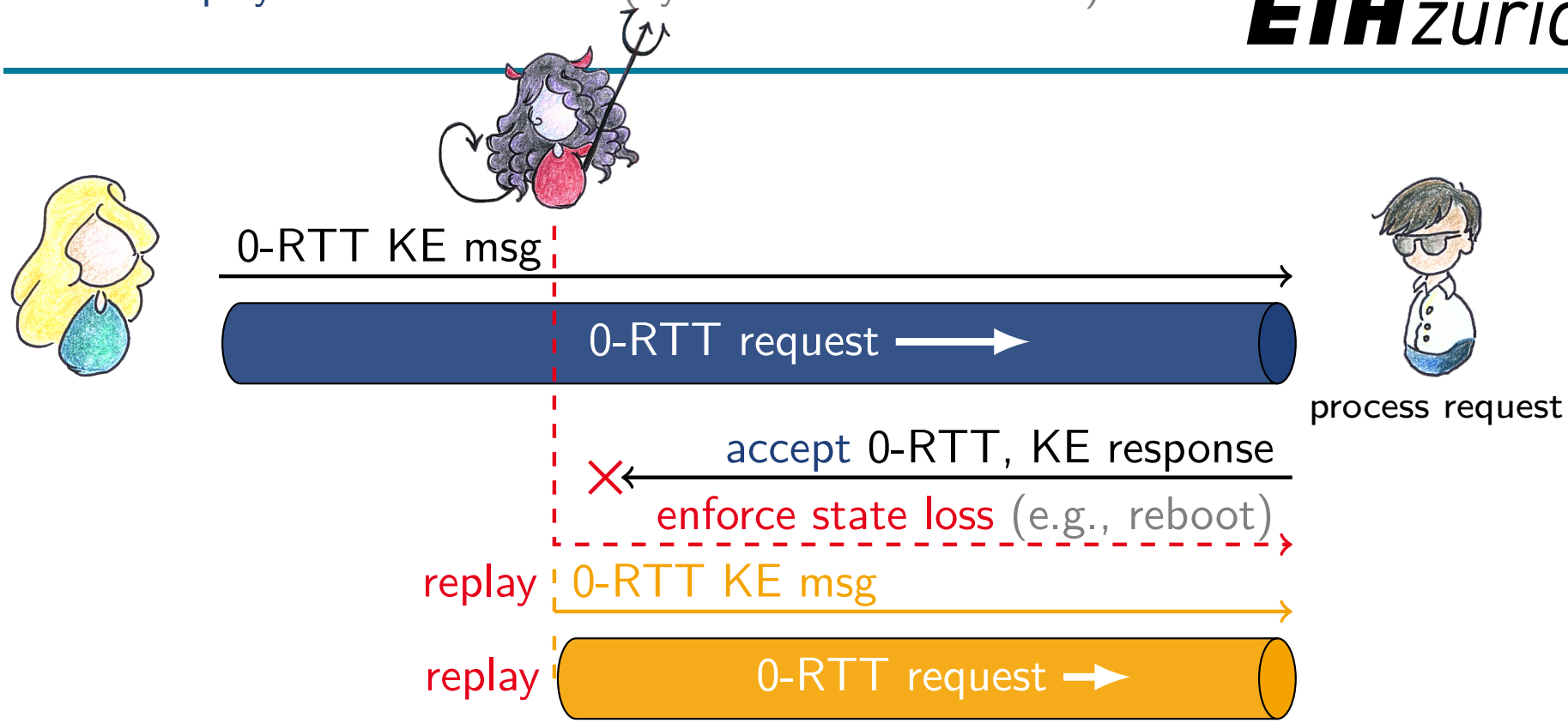
Generic Replay Attack on 0-RTT (by Daniel Kahn Gillmore)

ETH zürich



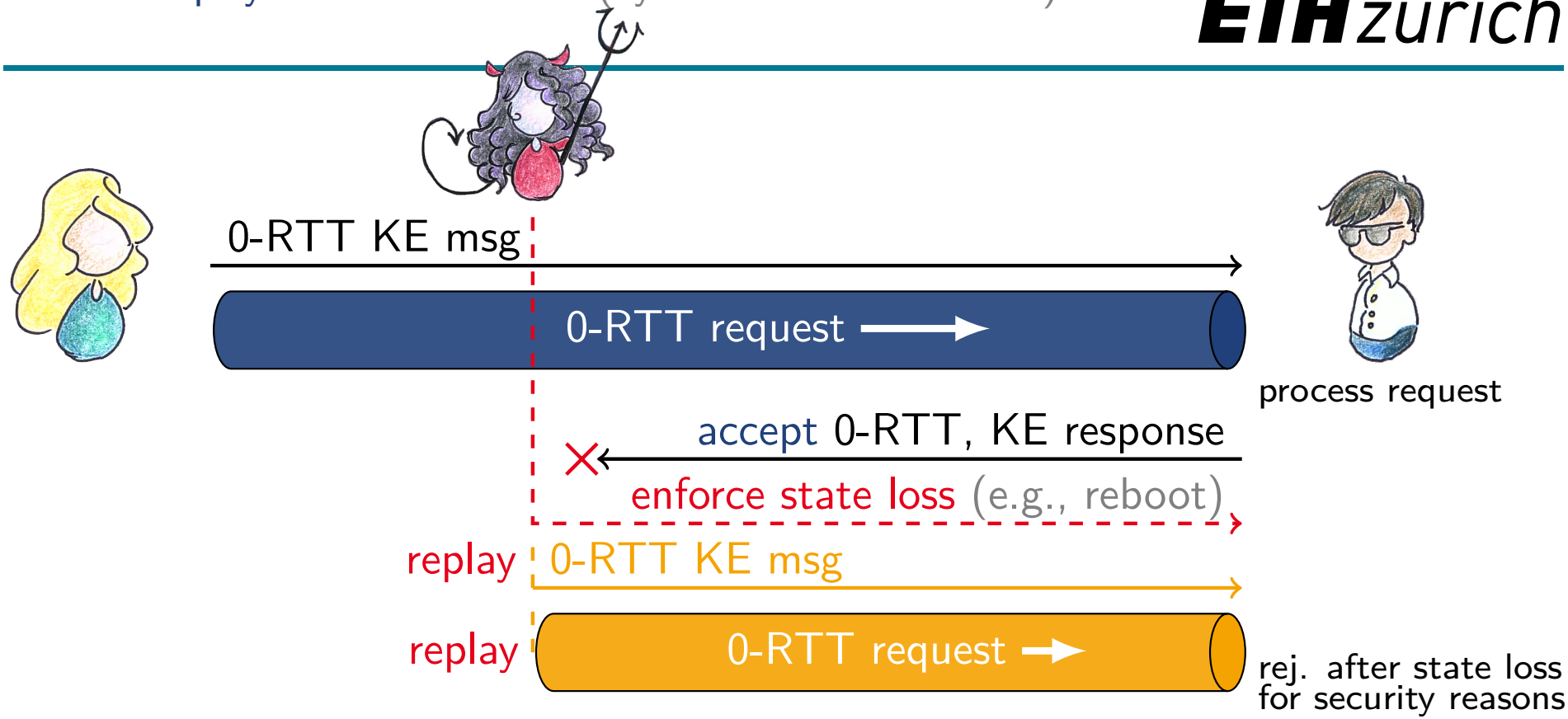
0-RTT and Replays

Generic Replay Attack on 0-RTT (by Daniel Kahn Gillmore)



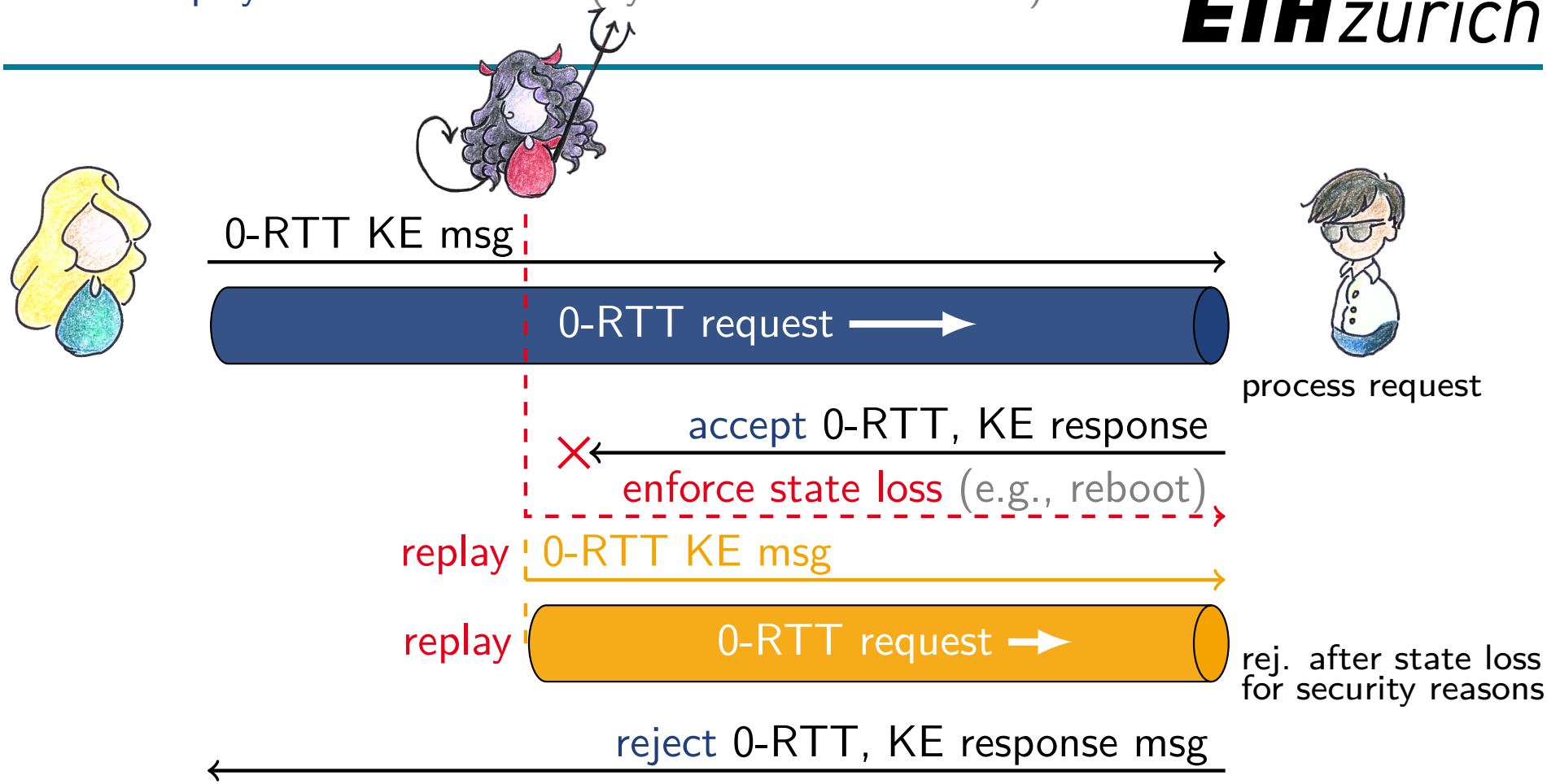
0-RTT and Replays

Generic Replay Attack on 0-RTT (by Daniel Kahn Gillmore)



0-RTT and Replays

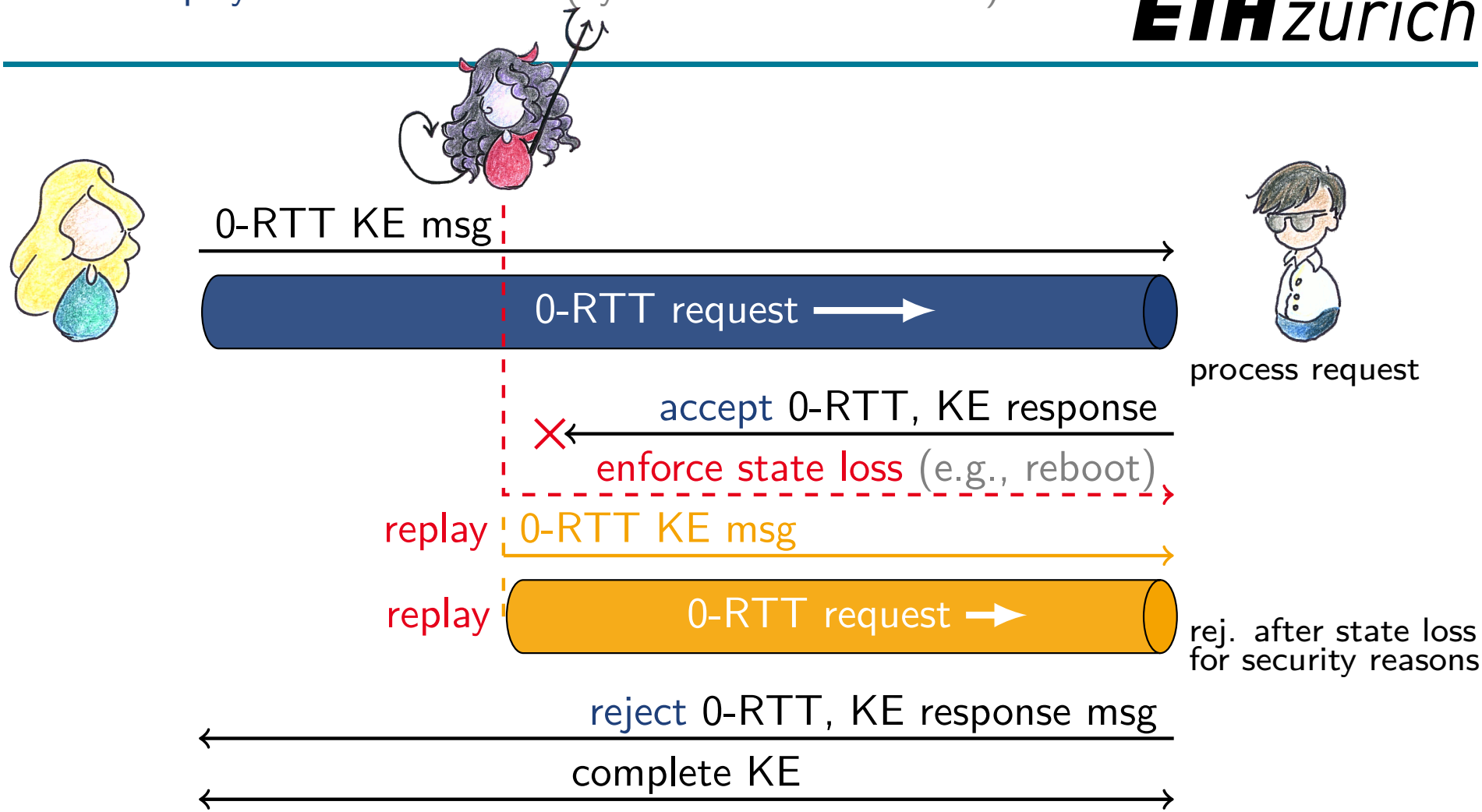
Generic Replay Attack on 0-RTT (by Daniel Kahn Gillmore)



0-RTT and Replays

Generic Replay Attack on 0-RTT (by Daniel Kahn Gillmore)

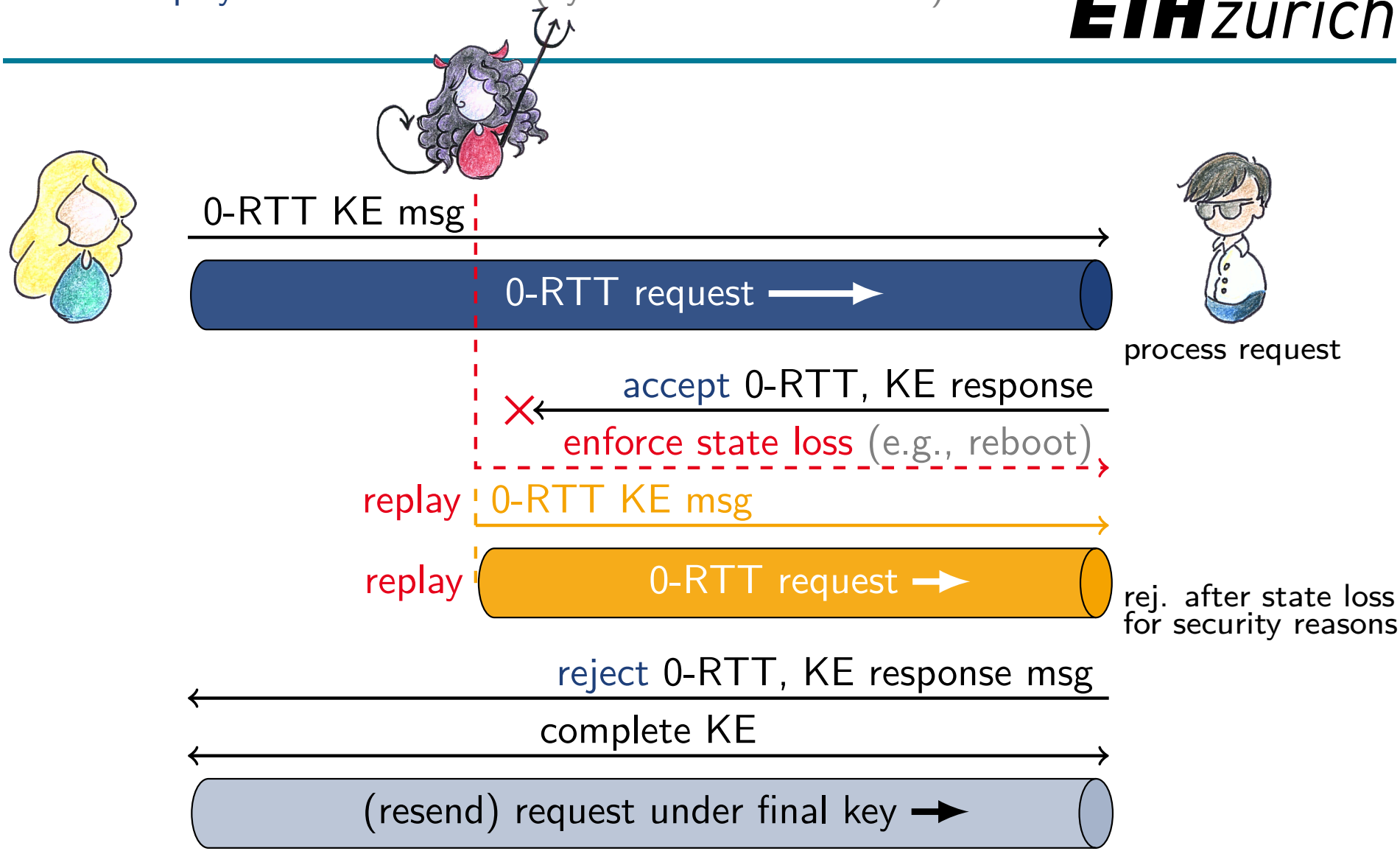
ETH zürich



0-RTT and Replays

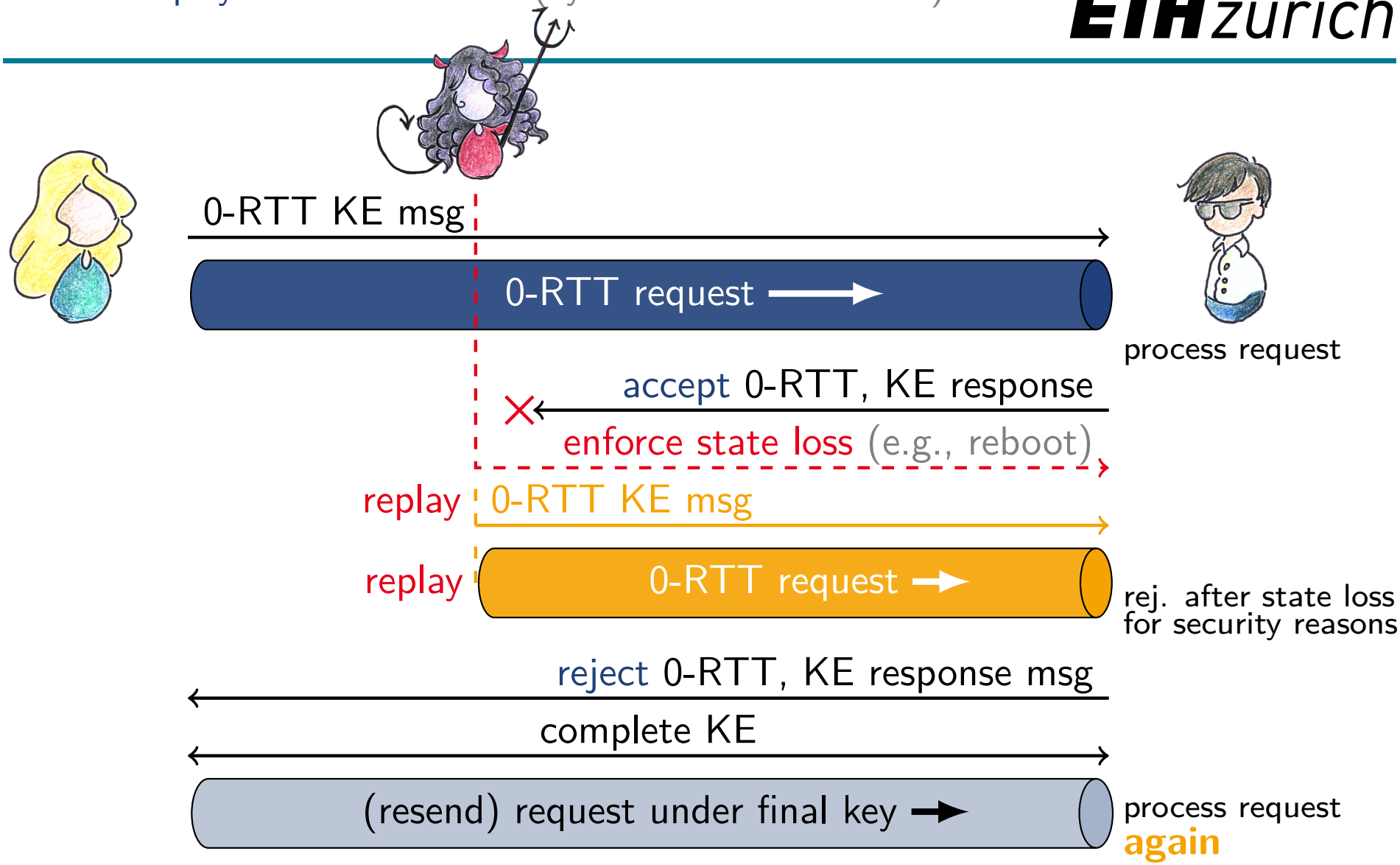
Generic Replay Attack on 0-RTT (by Daniel Kahn Gillmore)

ETH zürich



0-RTT and Replays

Generic Replay Attack on 0-RTT (by Daniel Kahn Gillmore)



0-RTT and Replays

TLS 1.3's take on replays

ETH zürich

0-RTT and Replays

TLS 1.3's take on replays

ETH zürich

TLS does not provide inherent replay protection for 0-RTT data.

[Simple duplicates] can be prevented by sharing state to guarantee that the 0-RTT data is accepted at most once.

Servers SHOULD provide that level of replay safety by implementing one of the methods described in this section [. . .] [RFC 8446, Section 8]

0-RTT and Replays

TLS 1.3's take on replays

ETH zürich

TLS does not provide inherent replay protection for 0-RTT data.

[Simple duplicates] can be prevented by sharing state to guarantee that the 0-RTT data is accepted at most once.

Servers SHOULD provide that level of replay safety by implementing one of the methods described in this section [...] [RFC 8446, Section 8]

- **suggested mechanisms**
  - single-use tickets: allow each RMS to be used only once (simplest)
  - `ClientHello` recording: reject by unique identifier
  - freshness checks: reject based on `ClientHello` time
- very much leaves things to the application/implementer...