the **OPTLS** protocol and **TLS** 1.3

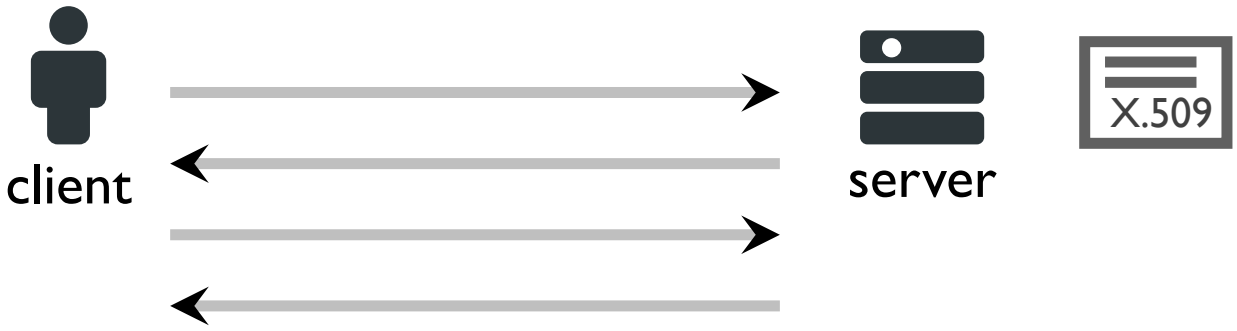
Hugo Krawczyk **IBM**

Hoeteck Wee **ENS**

TLS = **lingua franca** of crypto on the Internet

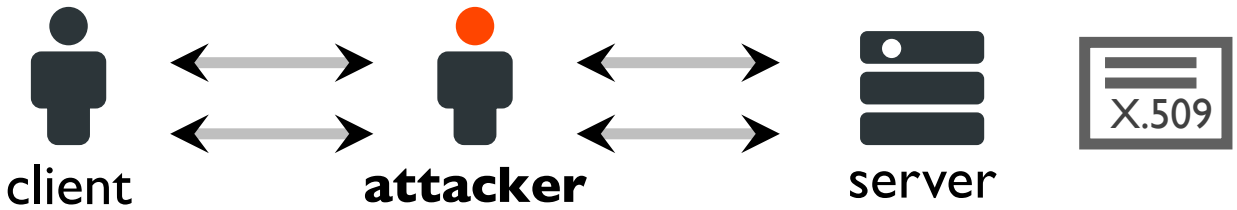
HTTPS, 802.1x, VPNs, email, VoIP, ...

TLS: transport layer security



goal: secure channel

goal: secure channel



cannot

- inject forged data into the stream (**authenticity**)
- distinguish data stream from random bytes (**confidentiality**)

TLS

history. 20 years of attacks, fixes, and extensions

- netscape's **SSL** (1994) ... **TLS** 1.2 (2008) ...

TLS 1.3

history. 20 years of attacks, fixes, and extensions

TLS 1.3. clean-up

- improved security and privacy, e.g. forward secrecy
- reduced latency: 1-rtt ; 0-rtt for repeat connections

TLS 1.3 and **OPTLS**

history. 20 years of attacks, fixes, and extensions

TLS 1.3. clean-up

- improved security and privacy, e.g. forward secrecy
- reduced latency: 1-rtt ; 0-rtt for repeat connections

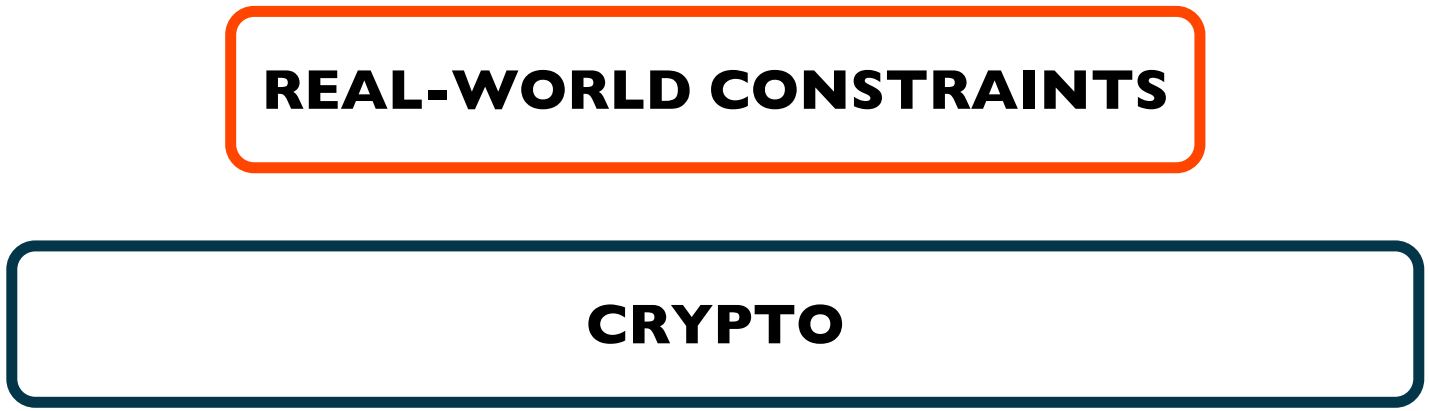
OPTLS. a simple suite of protocols developed to serve as the **crypto core** of **TLS** 1.3 handshake

our **philosophy**

CRYPTO

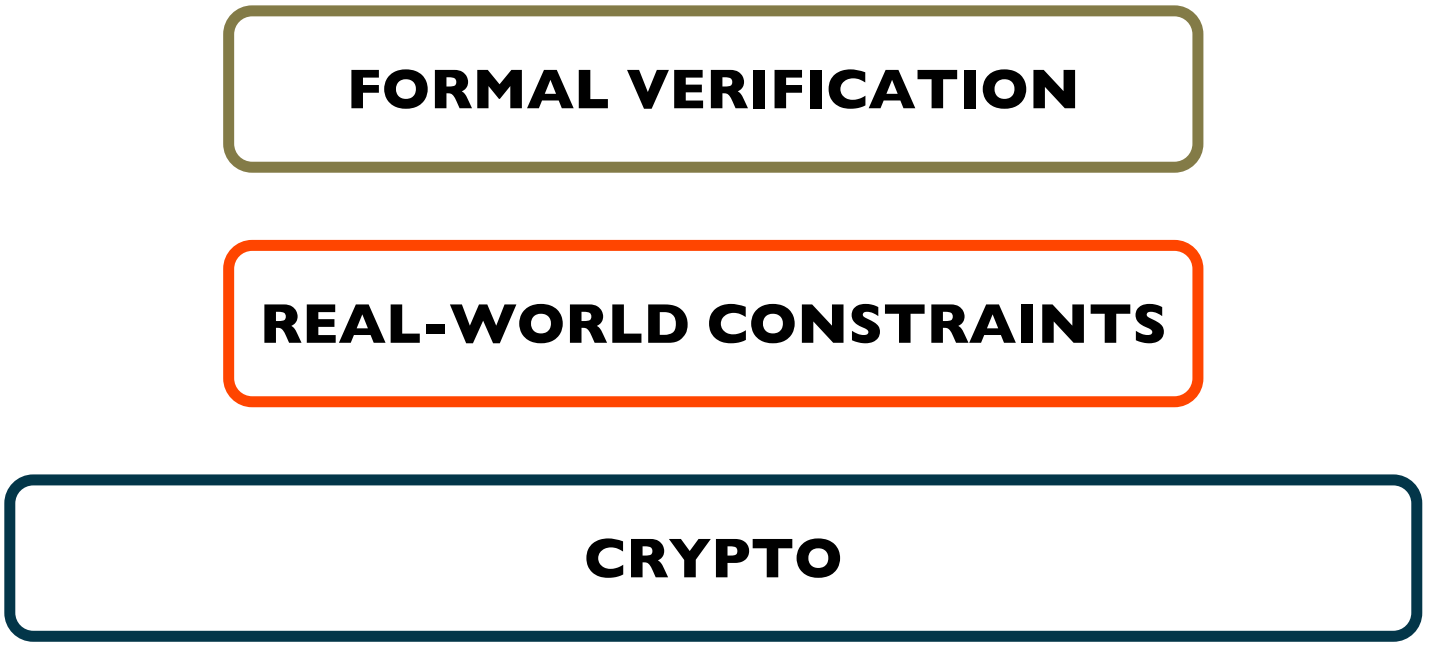
simple + modular + uniform **crypto core** as foundations

our **philosophy**



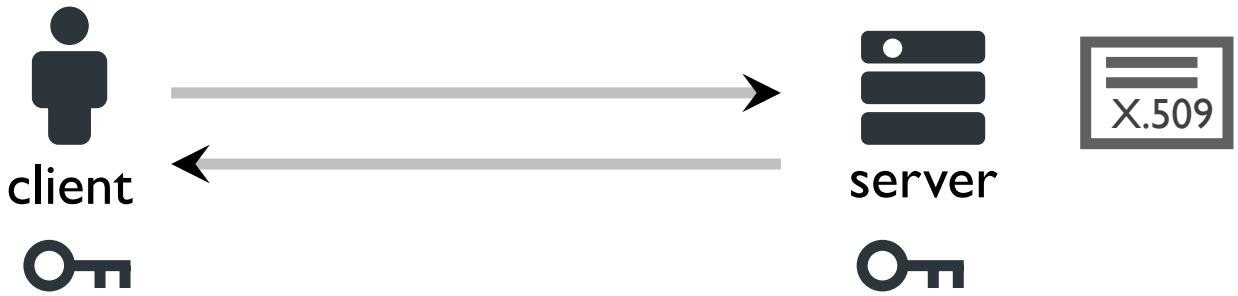
simple + modular + uniform **crypto core** as foundations

our **philosophy**

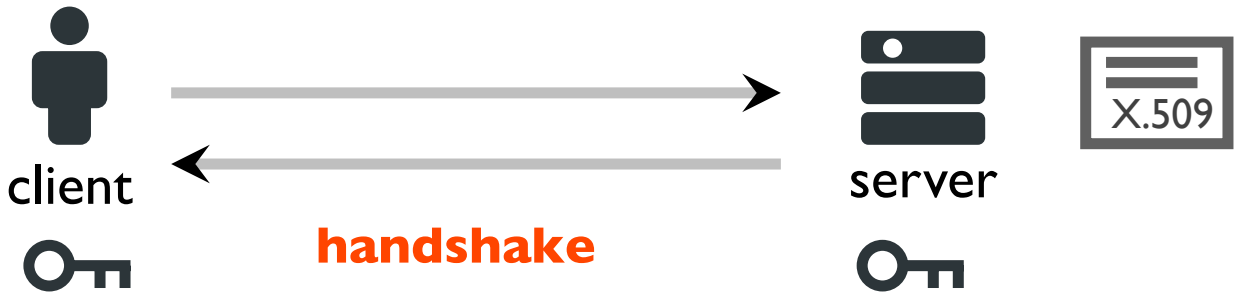


simple + modular + uniform **crypto core** as foundations

goal: secure key exchange



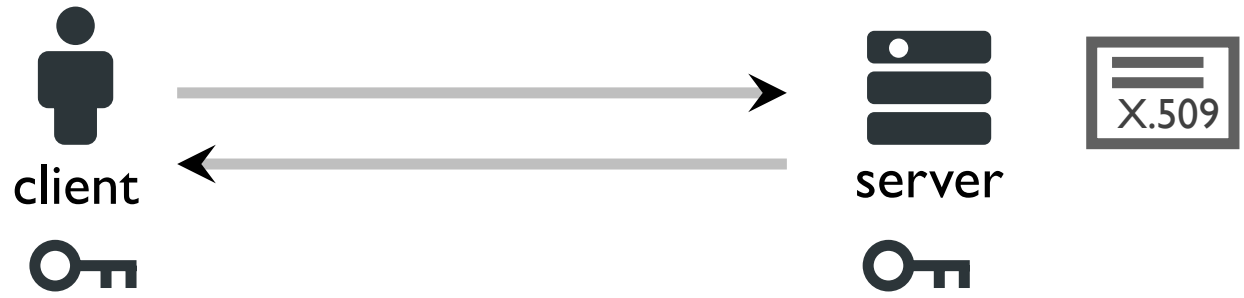
goal: secure key exchange



+ authenticated encryption **=** secure channel

record layer

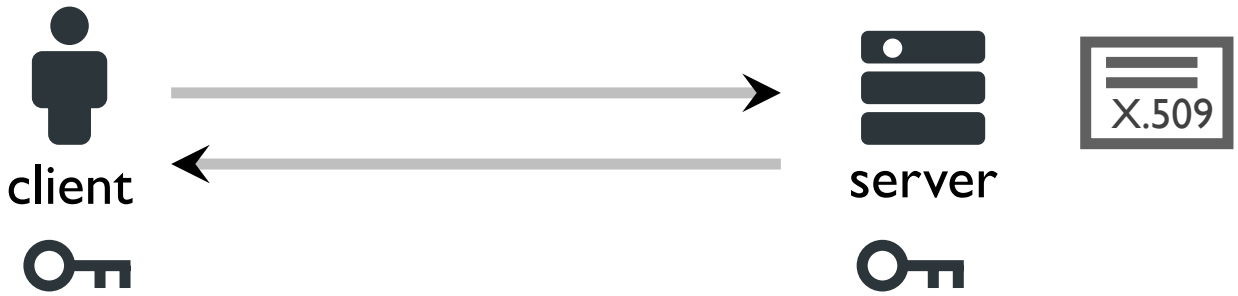
goal: secure key exchange



security. if a client completes with an honest server as its peer

- **agreement.** ∃ a server session with the same transcript
- **confidentiality.** the key is indistinguishable from random

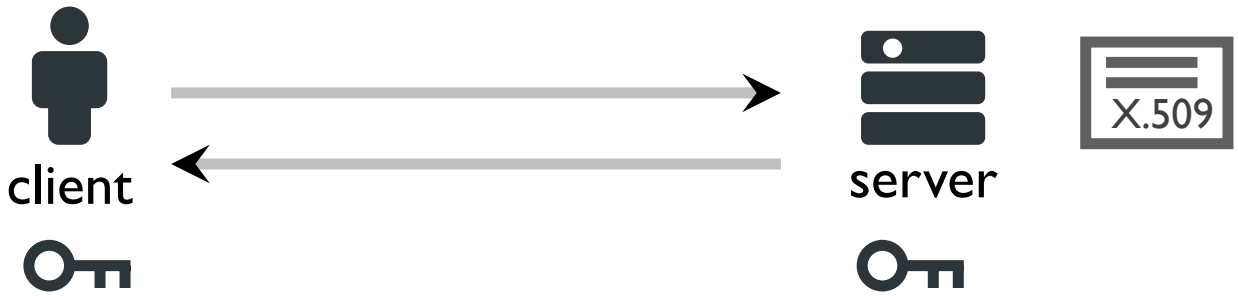
goal: secure key exchange



agreement + **confidentiality**

= **fundamental** requirements

goal: secure key exchange



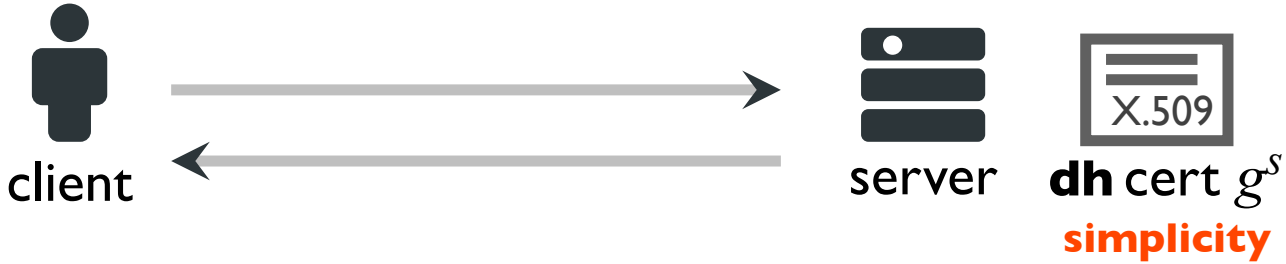
agreement + **confidentiality**

= **fundamental** requirements

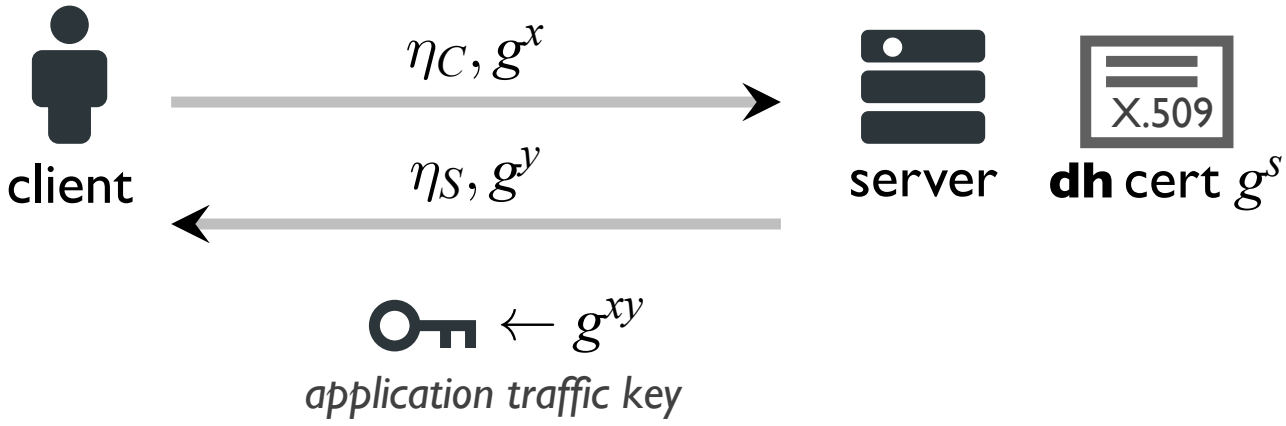
on which we can layer additional functionality/properties

e.g. client auth, key sync security

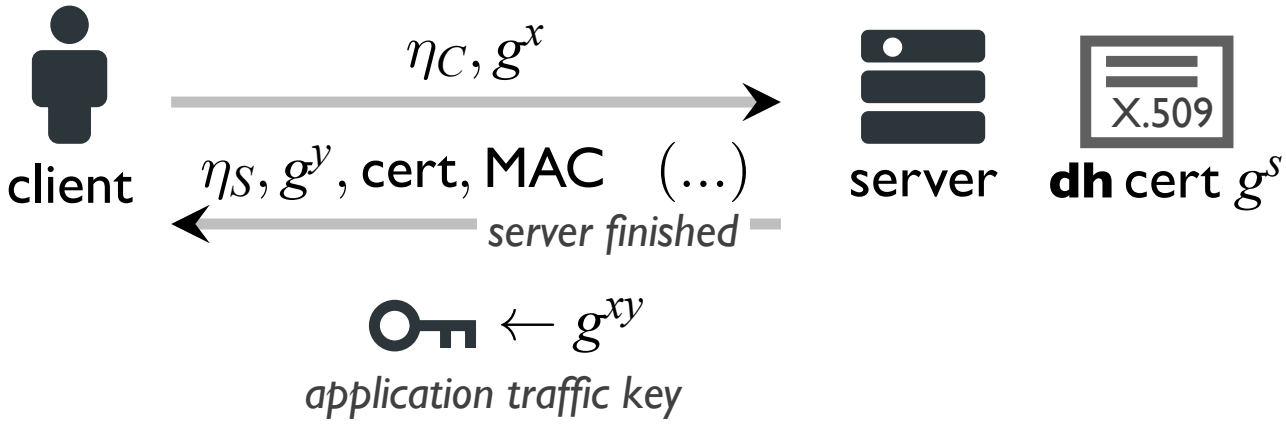
OPTLS



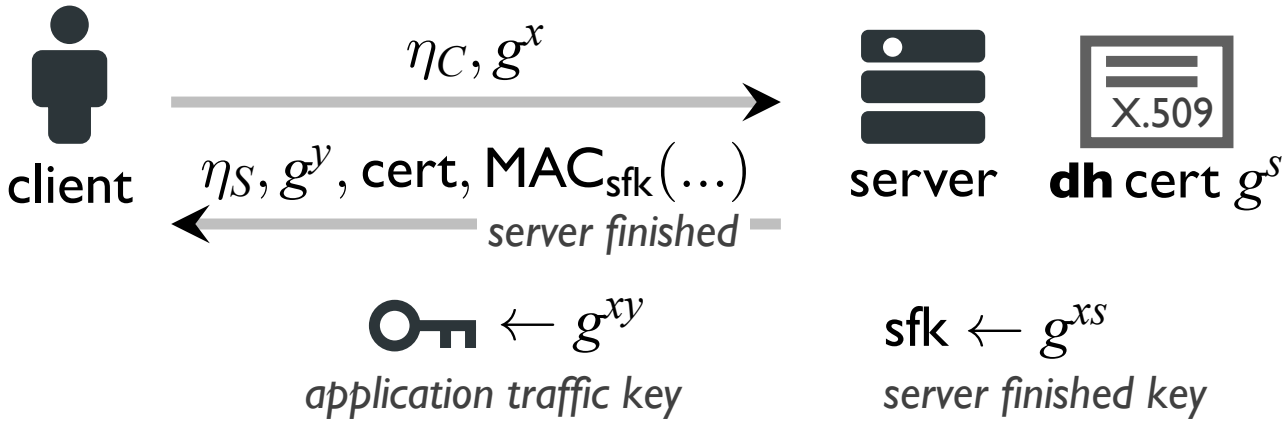
OPTLS


η_C, g^x
η_S, g^y
client
server
X.509
dh cert g^s
$\leftarrow g^{xy}$
application traffic key

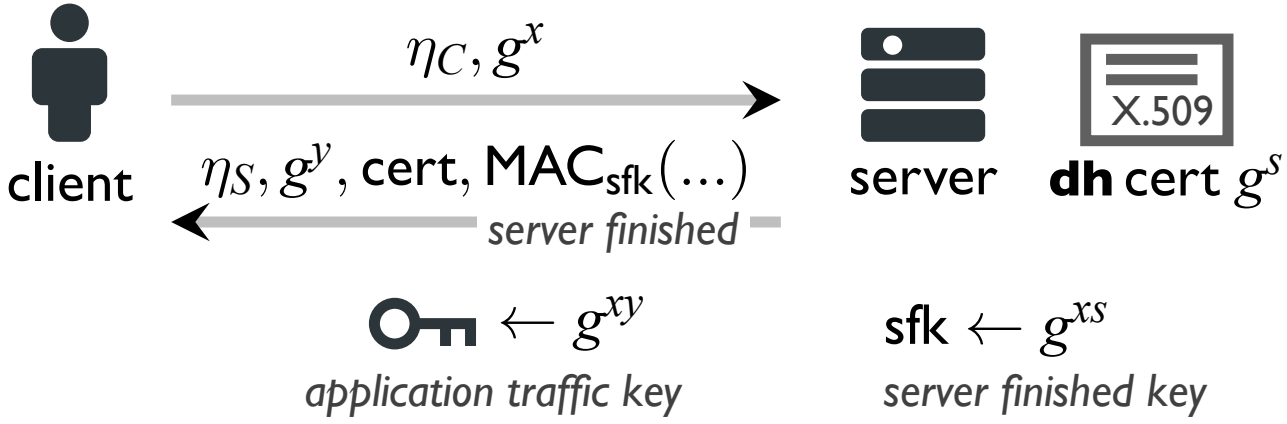
OPTLS


η_C, g^x
client
η_S, g^y, cert, MAC $(\ldots)$
server finished
server
X.509
dh cert g^s
$\leftarrow g^{xy}$
application traffic key

OPTLS


client
η_C, g^x
η_S, g^y, cert, $\mathsf{MAC}_{\mathsf{sfk}}(\ldots)$
server finished
server
X.509
dh cert g^s
$\leftarrow g^{xy}$
application traffic key
sfk $\leftarrow g^{xs}$
server finished key

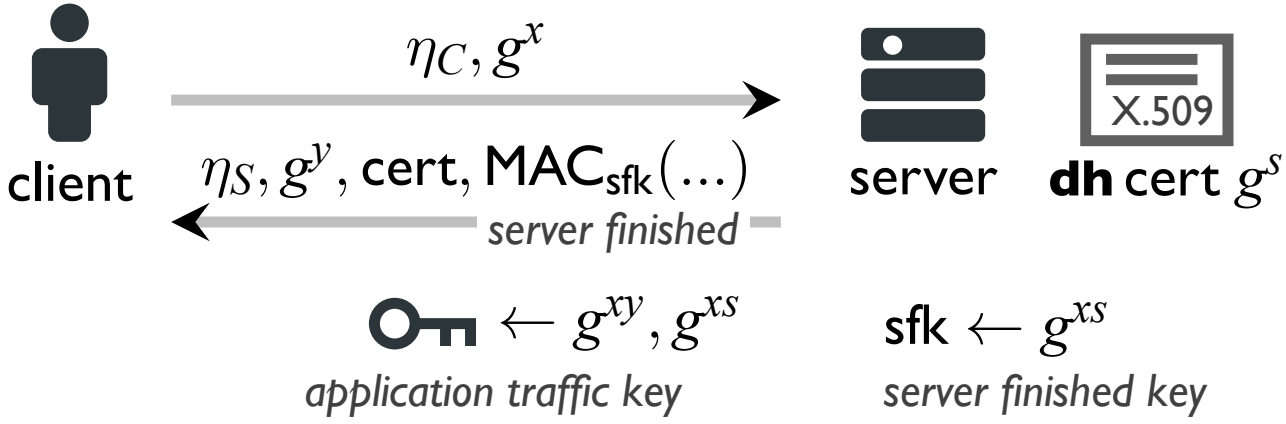
OPTLS



– **agreement.** **i.** g^s via cert, **ii.** transcript via MAC

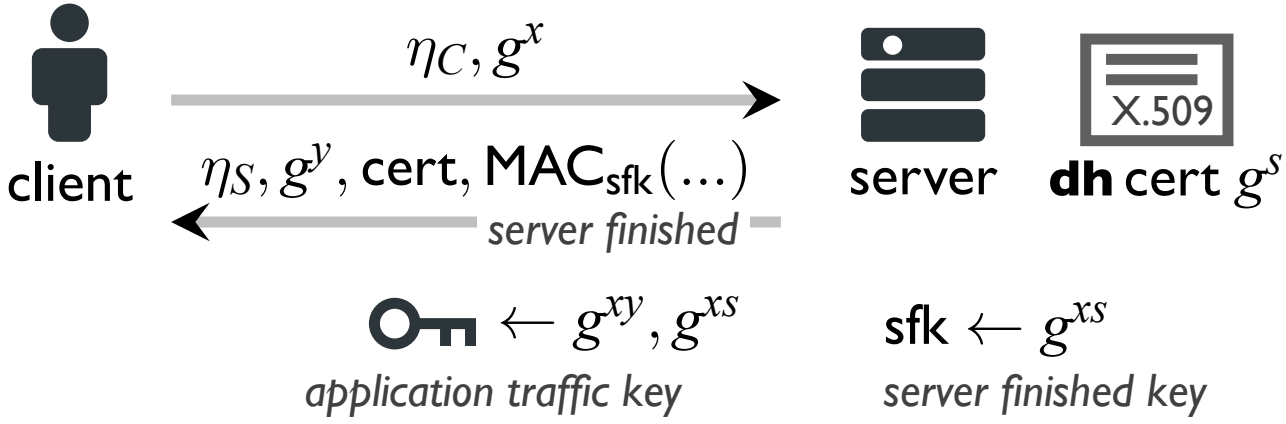
two-layer authentication

OPTLS



- **agreement.** **i.** g^s via cert, **ii.** transcript via MAC
- **confidentiality.**

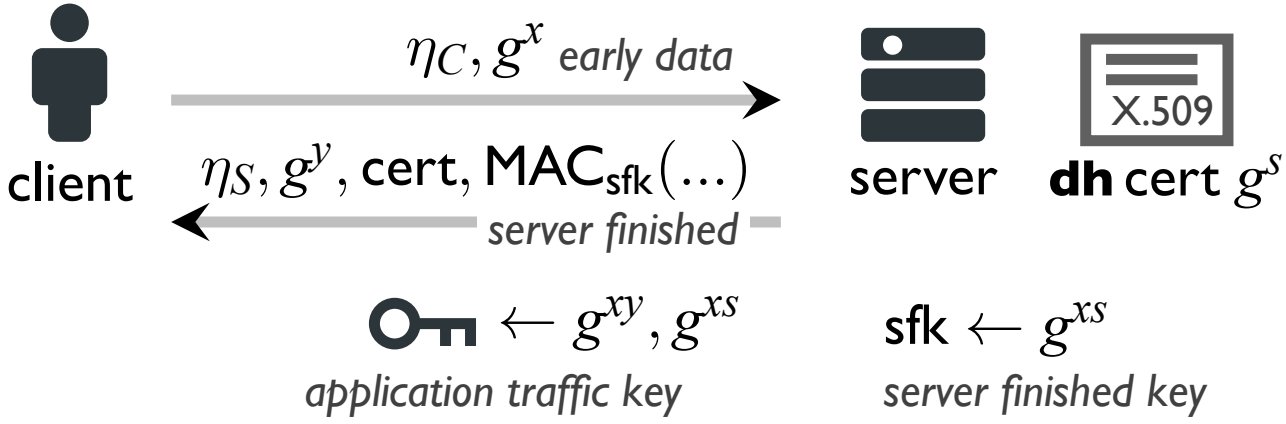
OPTLS



- **agreement.** **i.** g^s via cert, **ii.** transcript via MAC
- **confidentiality.** even if s or y is compromised

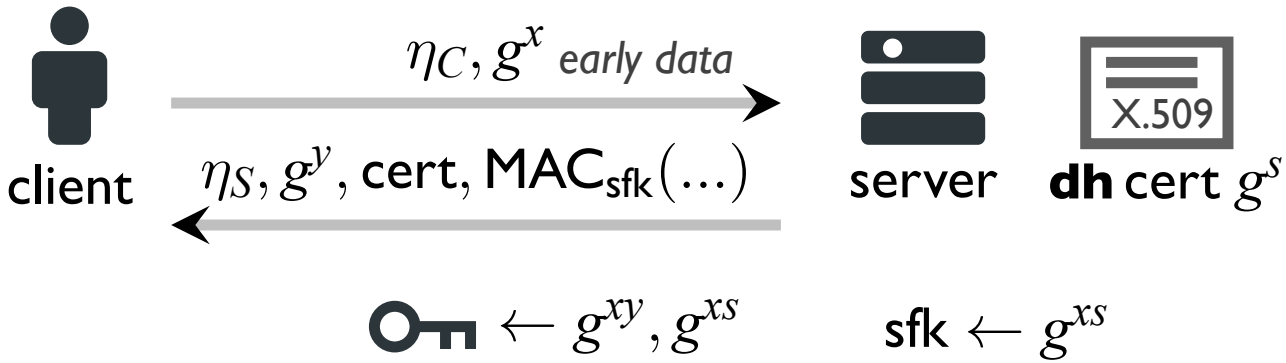
  forward secrecy + resilience to exposure of y

OPTLS



- **agreement.** **i.** g^s via cert, **ii.** transcript via MAC
- **confidentiality.** even if s or y is compromised
- 0**-rtt.** client encrypts early data using g^{xs} *no forward secrecy*

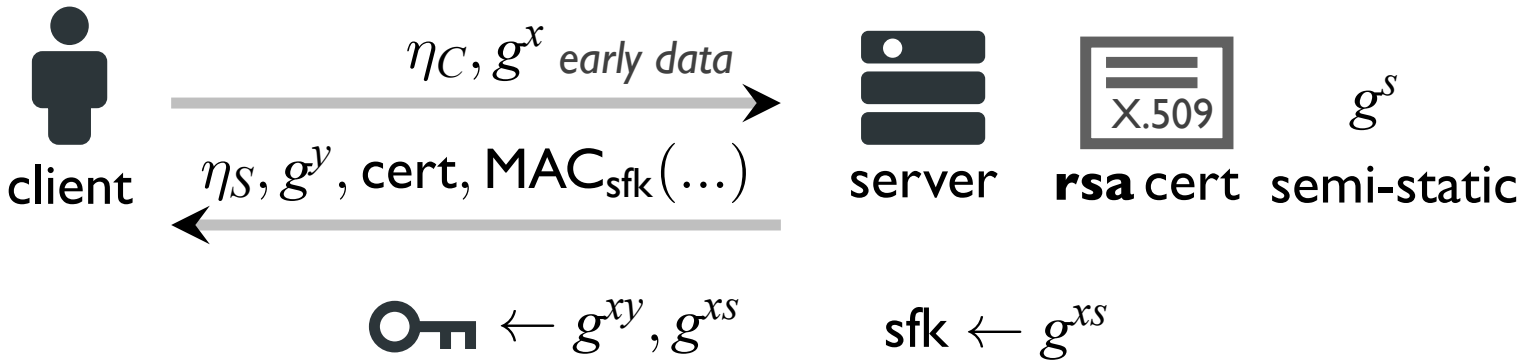
OPTLS: basic protocol



next. 4 modes corresponding to **TLS** settings

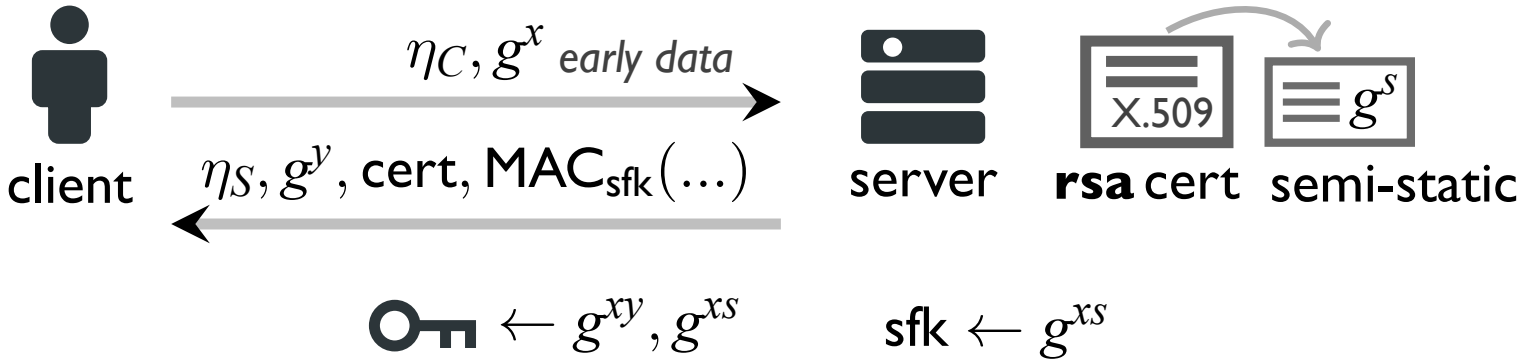
– i.e. **rsa** certs and pre-shared keys

OPTLS: 4 modes



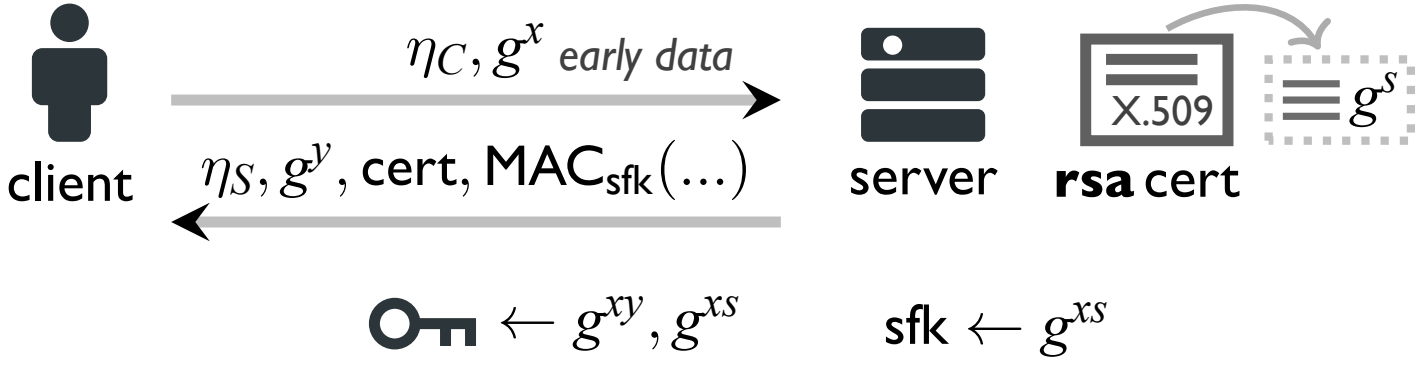
1. 1**-rtt semi-static.**

OPTLS: 4 modes



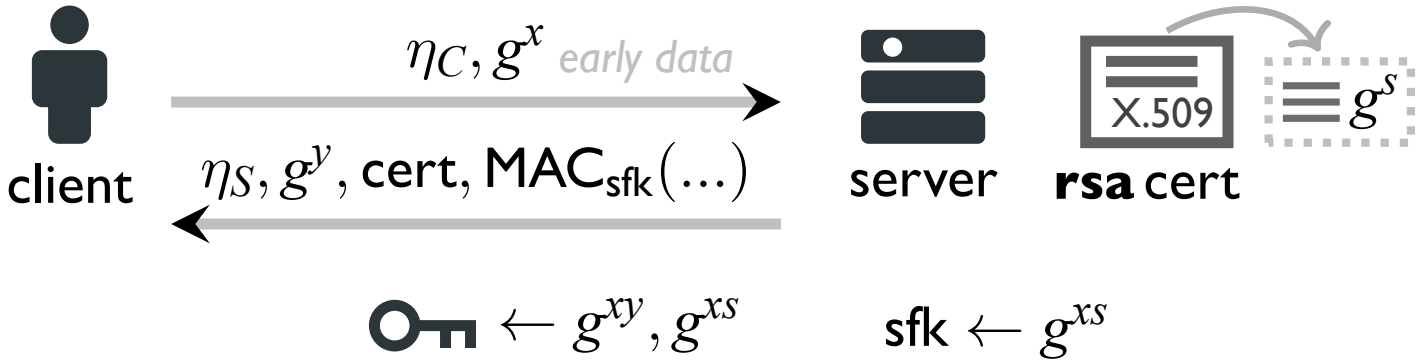
1. 1**-rtt semi-static.** server signs **semi-static** g^s

OPTLS: 4 modes



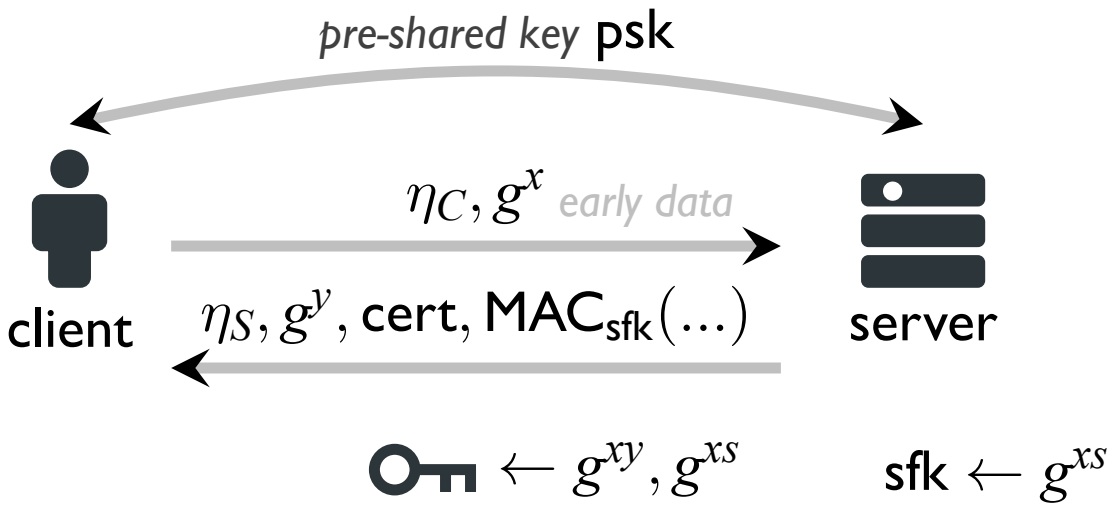
1. 1**-rtt semi-static.** server signs **semi-static** g^s
2. 1**-rtt non-static.**

OPTLS: 4 modes



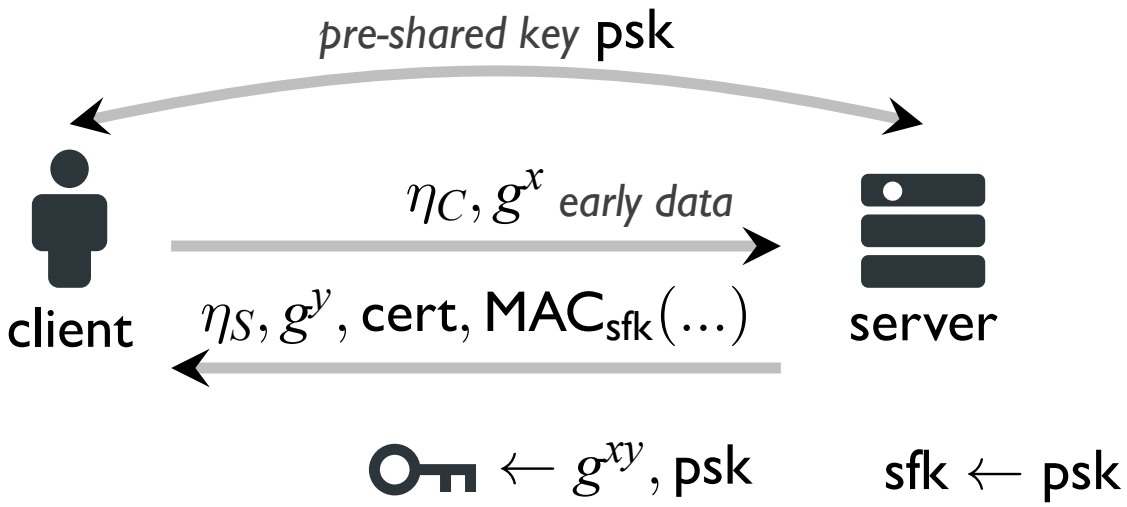
1. 1**-rtt semi-static.** server signs **semi-static** g^s
2. 1**-rtt non-static.** server signs **ephemeral** $g^s = g^y$

OPTLS: 4 modes



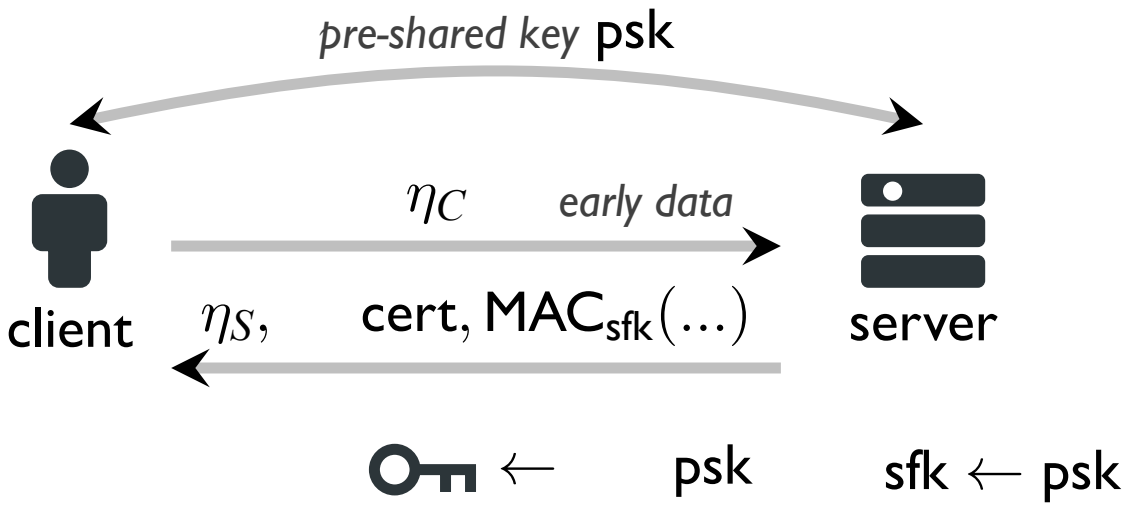
1. **1-rtt semi-static.** server signs **semi-static** g^s
2. **1-rtt non-static.** server signs **ephemeral** $g^s = g^y$
3. **psk-dhe.**

OPTLS: 4 modes



1. **1-rtt semi-static.** server signs **semi-static** g^s
2. **1-rtt non-static.** server signs **ephemeral** $g^s = g^y$
3. **psk-dhe.** uses psk in place of g^{xs}

OPTLS: 4 modes



1. **1-rtt semi-static.** server signs **semi-static** g^s
2. **1-rtt non-static.** server signs **ephemeral** $g^s = g^y$
3. **psk-dhe.** uses psk in place of g^{xs}
4. **psk.** psk only *fast, but no forward secrecy*

OPTLS: key derivation

es *ephemeral secret*		ss *static secret*	
g^{xy}		g^{xs}	**1-rtt semi-static**
g^{xy}	=	g^{xs}	**1-rtt non-static**
g^{xy}		psk	**psk-dhe**
psk	=	psk	**psk**

OPTLS: key derivation

es

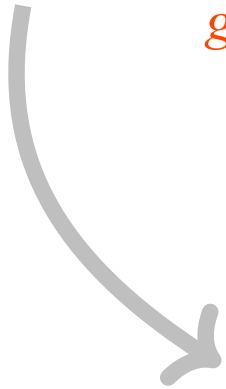
ephemeral secret

g^{xy} or psk

ss

static secret

g^{xs} or psk



sfk

server finished key

edk

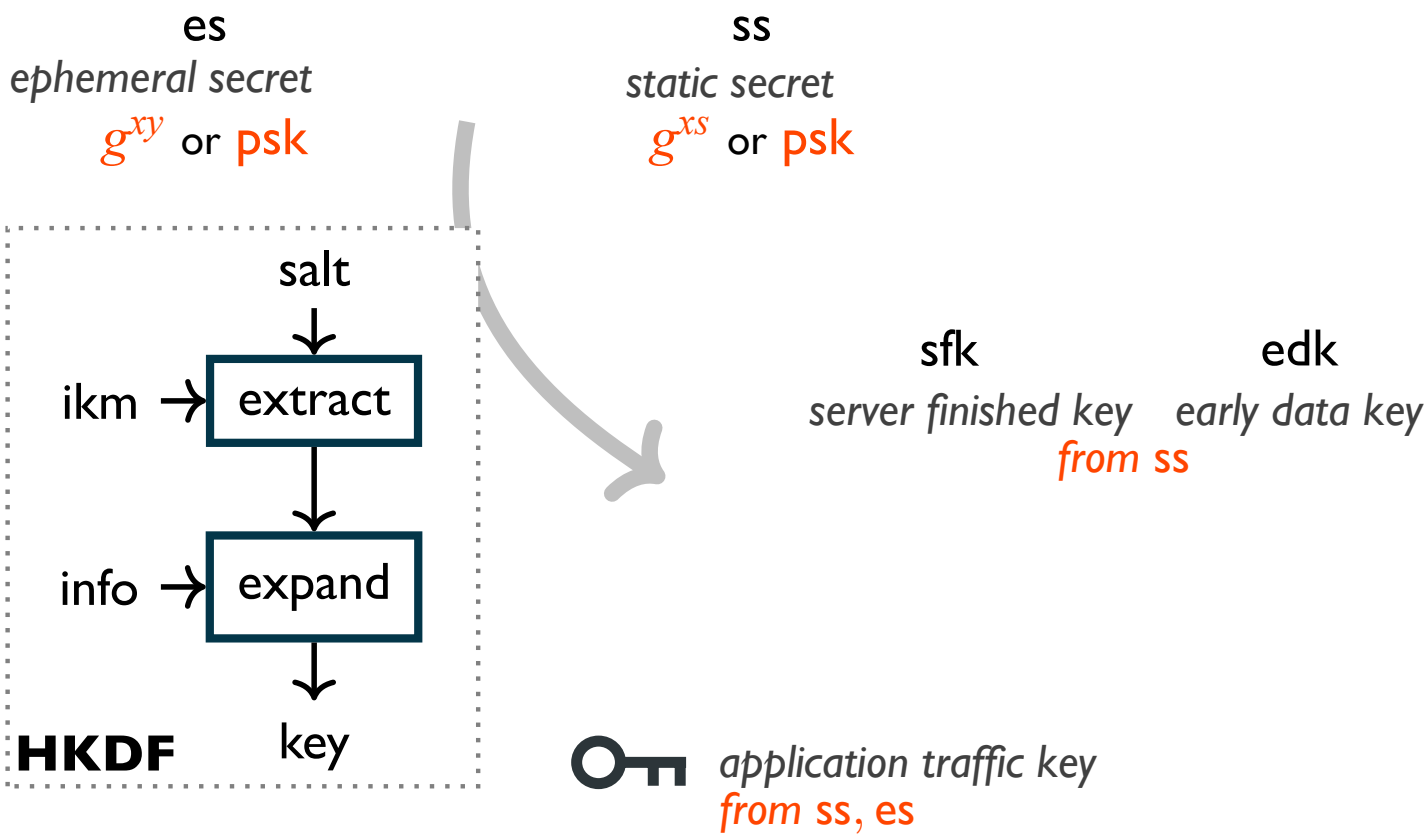
early data key

from ss

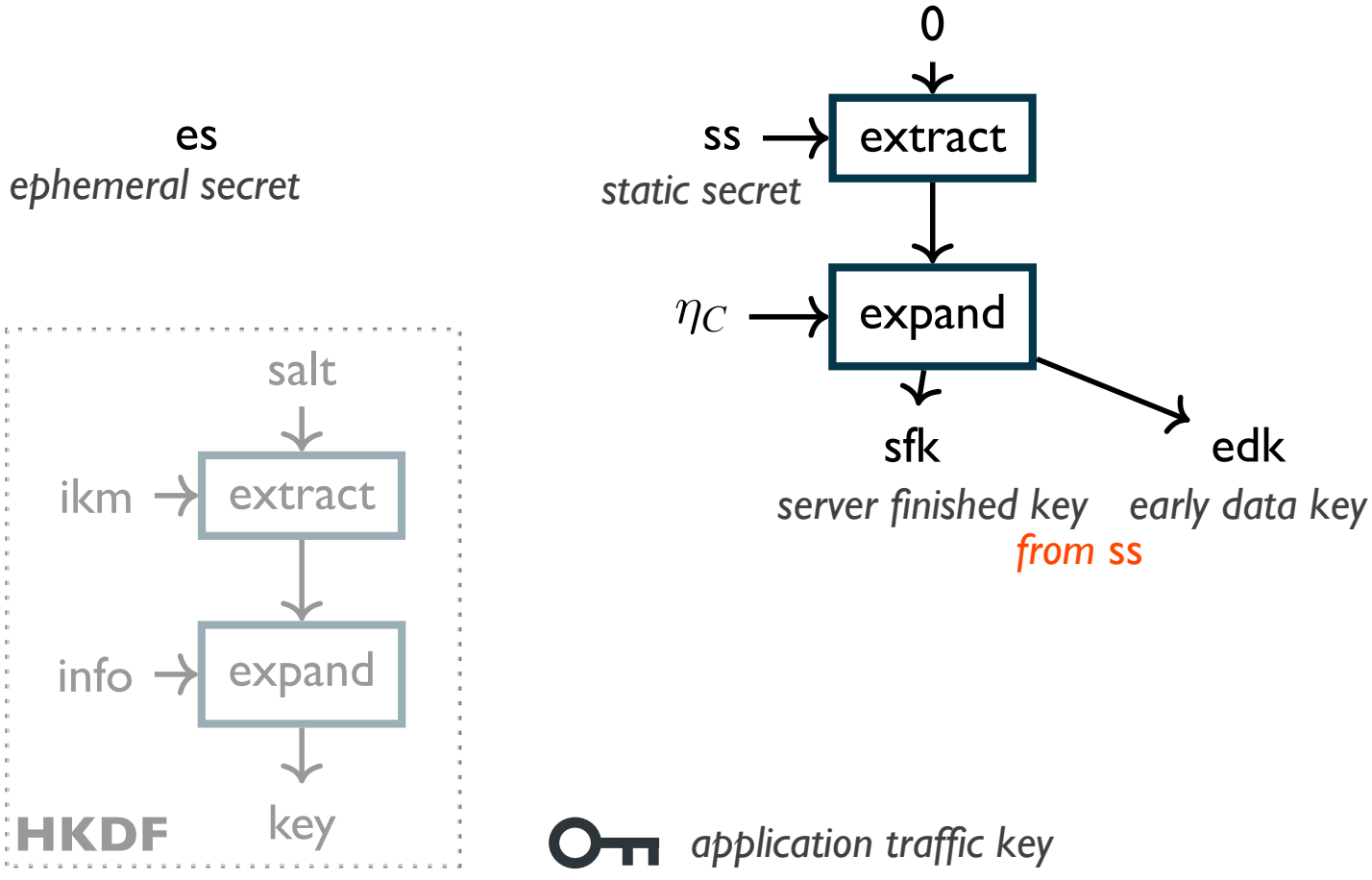
application traffic key

from ss, es

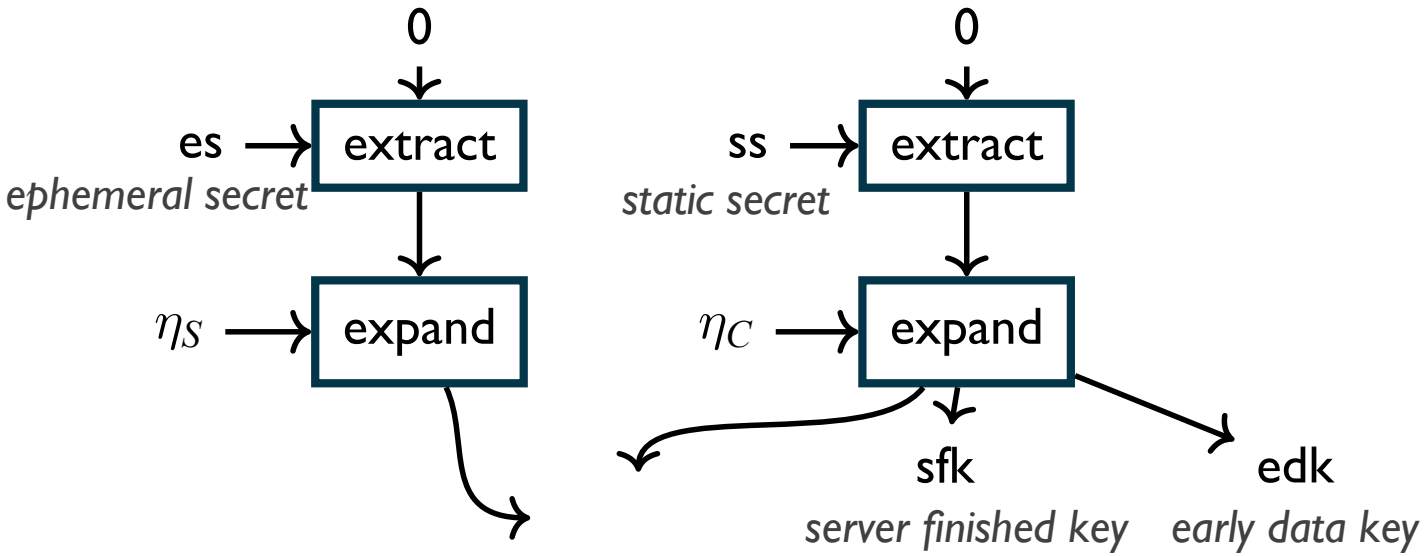
OPTLS: key derivation



OPTLS: key derivation

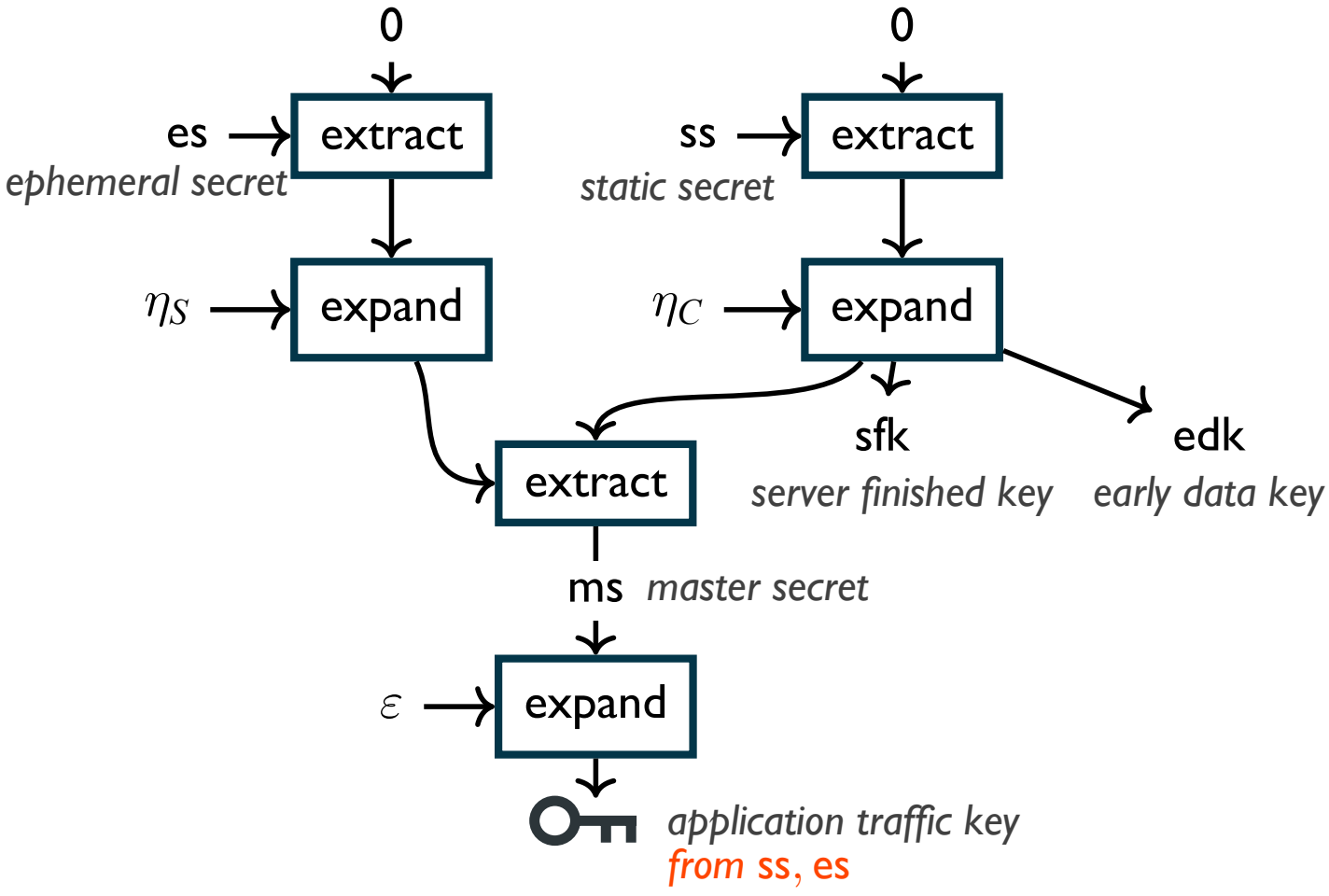


OPTLS: key derivation

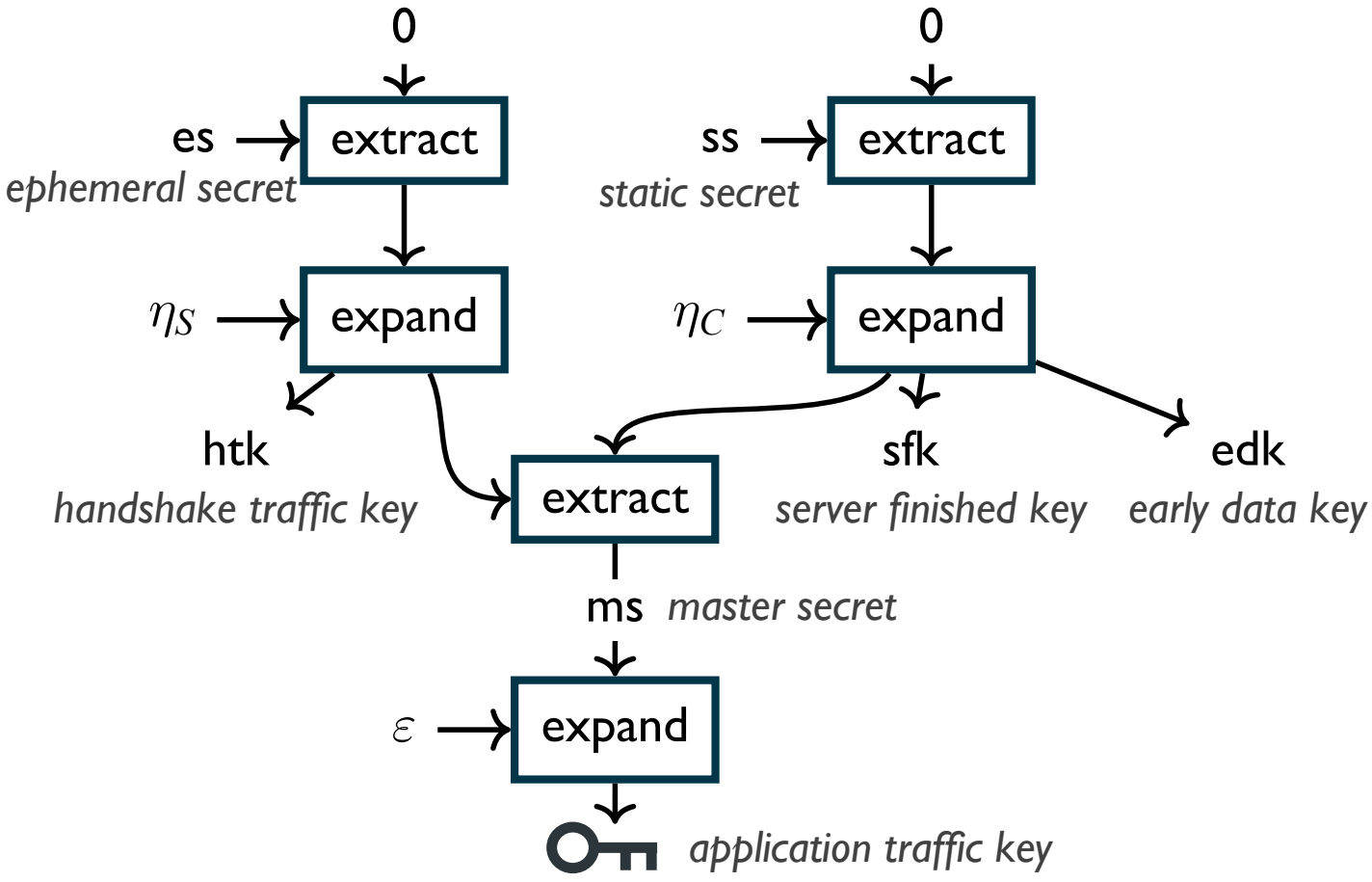


application traffic key
from ss, es

OPTLS: key derivation



OPTLS: key derivation



OPTLS **TLS** 1.3

OPTLS $\sim$ crypto core TLS 1.3
handshake

- adopts the same modes + uniform key derivation via HKDF
- default full handshake = 1-rtt non-static

OPTLS $\sim$ **crypto core** handshake **TLS** 1.3

- adopts the same modes + uniform key derivation via HKDF
- default full handshake = 1-rtt non-static

additions in **TLS** 1.3

i. session hash in HKDF *binding to unique session parameters*

ii. "always signs" in 1-rtt semi-static *continuous possession of signing key*

iii. client finished message *client key confirmation*

OPTLS

1. simple + modular + uniform crypto core upon which we could build more functionality/properties
2. served as the basis for the current **TLS** 1.3 crypto design

OPTLS

1. simple + modular + uniform crypto core upon which we could build more functionality/properties
2. served as the basis for the current **TLS** 1.3 crypto design
3. future support for DH certs and offline signatures

(*design and analysis*)

OPTLS

1. simple + modular + uniform crypto core upon which we could build more functionality/properties
2. served as the basis for the current **TLS** 1.3 crypto design
3. future support for DH certs and offline signatures

future/on-going work.

- resumption, client authentication, ...
- formal verification *c.f. miTLS & next talk*

OPTLS

1. simple + modular + uniform crypto core upon which we could build more functionality/properties
2. served as the basis for the current **TLS** 1.3 crypto design
3. future support for DH certs and offline signatures

future/on-going work.

- resumption, client authentication, ...
- formal verification

acks. Eric Rescorla, TLS WG, QUIC, ...