When One Key is Not Enough

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

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Felix Günther

Technische Universität Darmstadt, Germany

joint work with Marc Fischlin

published in ACM CCS 2014

00101110001011 Cryptoplexity
qed
Cryptography & Complexity Theory
Technische Universität Darmstadt
www.cryptoplexity.de

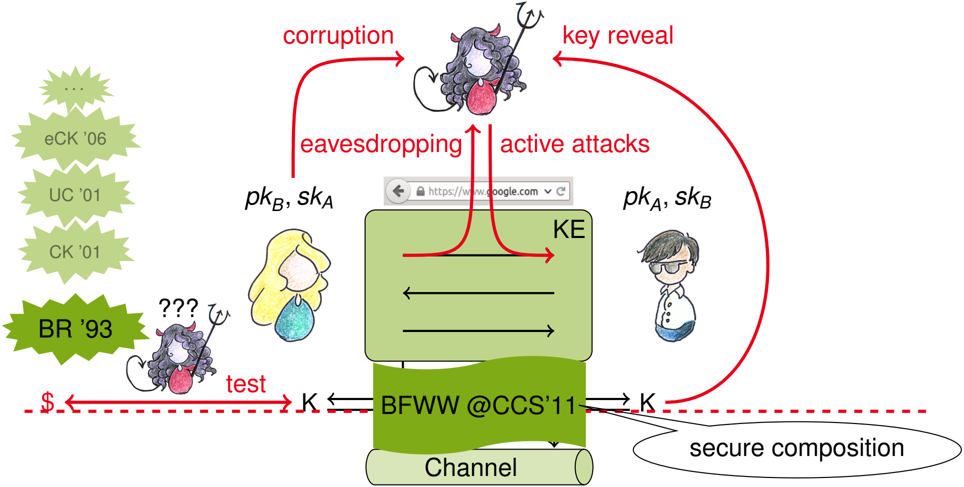
EC SPRIDE
EUROPEAN CENTER FOR
SECURITY AND PRIVACY BY DESIGN

CROSSING

Heisenberg-
Programm
Deutsche
Forschungsgemeinschaft

Key Exchange

so far…



corruption

key reveal

eavesdropping

active attacks

pk_B, sk_A

pk_A, sk_B

https://www.google.com

KE

…

eCK ’06

UC ’01

CK ’01

BR ’93

???

test

\$

K

BFWW @CCS’11

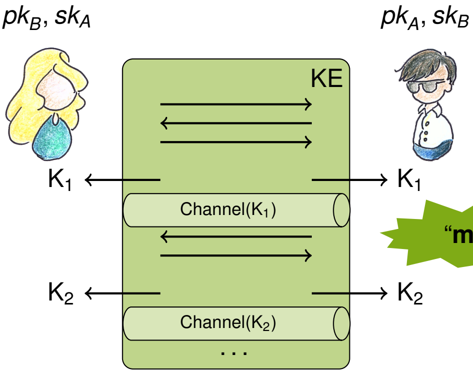
K

secure composition

Channel

drawings by *Giorgia Azzurra Marson*

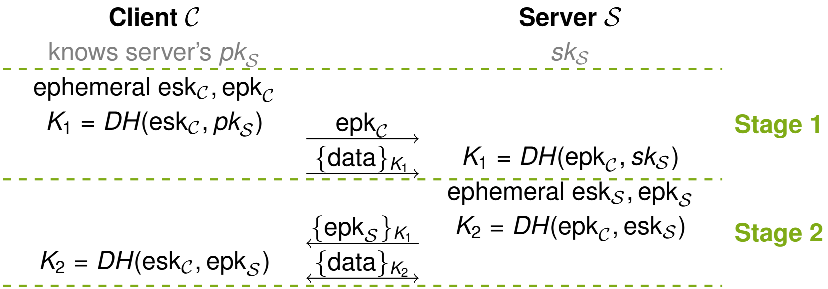
But what if. . . ?



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

Should we care?

- **QUIC** (“Quick UDP Internet Connections”, Google 2013)
  - “low-latency transport protocol with security equivalent to TLS”
  - Diffie–Hellman-based key exchange
  - aims at 0-RTT, i.e., immediately encrypts under intermediate key K_1
  - later rekeys to forward-secret K_2
  - intermediate key K_1 used to establish K_2 (i.e., in KE part)



Client $\mathcal{C}$ | **Server $\mathcal{S}$**

knows server's $pk_{\mathcal{S}}$ | $sk_{\mathcal{S}}$

ephemeral $\text{esk}_{\mathcal{C}}, \text{epk}_{\mathcal{C}}$

$K_1 = DH(\text{esk}_{\mathcal{C}}, pk_{\mathcal{S}})$ $\xrightarrow{\text{epk}_{\mathcal{C}}}$

$\xrightarrow{\{\text{data}\}_{K_1}}$ $K_1 = DH(\text{epk}_{\mathcal{C}}, sk_{\mathcal{S}})$

Stage 1

ephemeral $\text{esk}_{\mathcal{S}}, \text{epk}_{\mathcal{S}}$

$\xleftarrow{\{\text{epk}_{\mathcal{S}}\}_{K_1}}$ $K_2 = DH(\text{epk}_{\mathcal{C}}, \text{esk}_{\mathcal{S}})$

$K_2 = DH(\text{esk}_{\mathcal{C}}, \text{epk}_{\mathcal{S}})$ $\xleftrightarrow{\{\text{data}\}_{K_2}}$

Stage 2

Should we care?

- **TLS with session resumption**
  - client and server already established session and hold master key
  - client resumes session later
  - new session key is derived using (old) master key and fresh nonces

  - can also be though of as a *multi-stage* key exchange (keeps state)

  - related: TLS renegotiation considered as phases (GKS @ CCS'13)
    but renegotiation is new key exchange, not reusing the master key

Should we care?

- **TLS version 1.3**

```
ClientHello
ClientKeyShare            -------->          ServerHello
                                          ServerKeyShare
                                          {Certificate*}
                                                  {...*}
                          <--------           {Finished}
{...*}
{Finished}                -------->
[Application Data]        <------->   [Application Data]

              Figure 1.  Message flow for a full handshake
```

—IETF draft-ietf-tls-tls13-04 (work in progress)

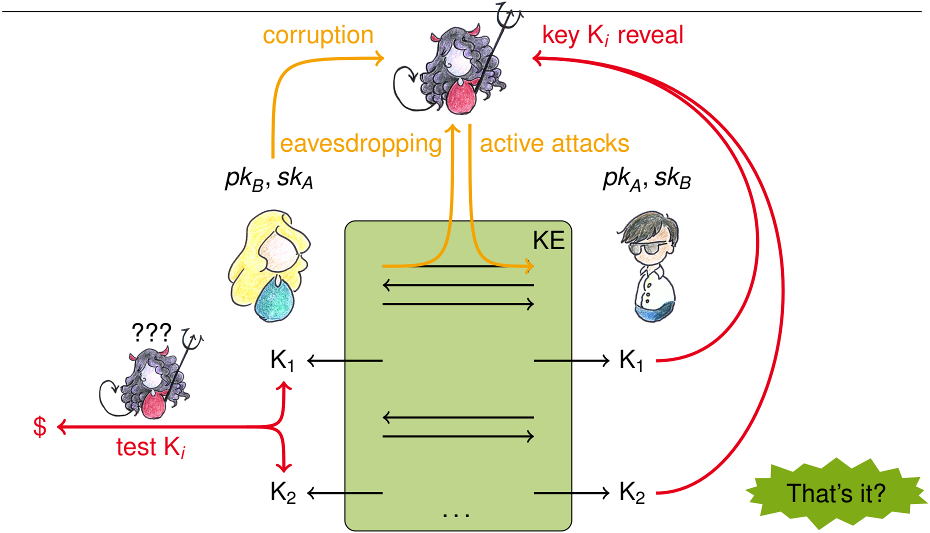
- handshake messages are protected with intermediate keys
- application data is protected with final keys

Outline

- A **Model** for Multi-Stage Key Exchange
- What about **Composition**?
- Google's **QUIC** Protocol

Model for Multi-Stage Key Exchange

TECHNISCHE UNIVERSITÄT DARMSTADT

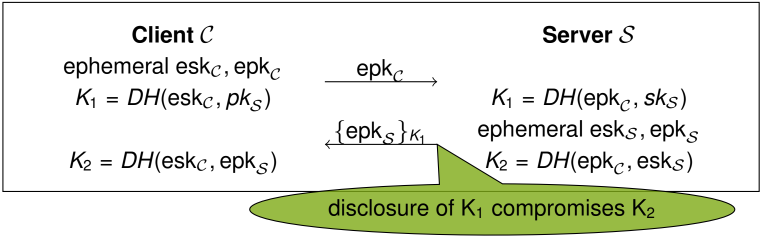


Model for Multi-Stage Key Exchange

Security aspects to consider

- **(Session-)Key Dependence**
  - multi-stage $\Rightarrow$ derived keys might build upon each other
  - we have to disallow trivial reveal queries

ex: QUIC



Client $\mathcal{C}$

ephemeral $\mathsf{esk}_{\mathcal{C}}, \mathsf{epk}_{\mathcal{C}}$

$K_1 = DH(\mathsf{esk}_{\mathcal{C}}, pk_{\mathcal{S}})$

$K_2 = DH(\mathsf{esk}_{\mathcal{C}}, \mathsf{epk}_{\mathcal{S}})$

$\xrightarrow{\mathsf{epk}_{\mathcal{C}}}$

$\xleftarrow{\{\mathsf{epk}_{\mathcal{S}}\}_{K_1}}$

Server $\mathcal{S}$

$K_1 = DH(\mathsf{epk}_{\mathcal{C}}, sk_{\mathcal{S}})$

ephemeral $\mathsf{esk}_{\mathcal{S}}, \mathsf{epk}_{\mathcal{S}}$

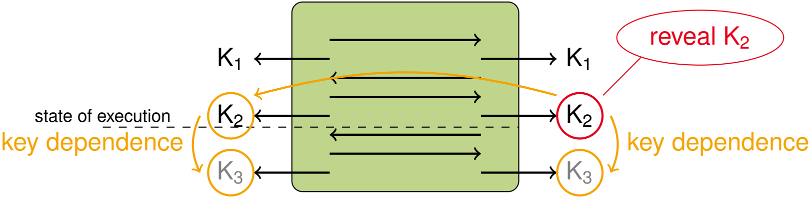
$K_2 = DH(\mathsf{epk}_{\mathcal{C}}, \mathsf{esk}_{\mathcal{S}})$

disclosure of K_1 compromises K_2

Model for Multi-Stage Key Exchange

Security aspects to consider

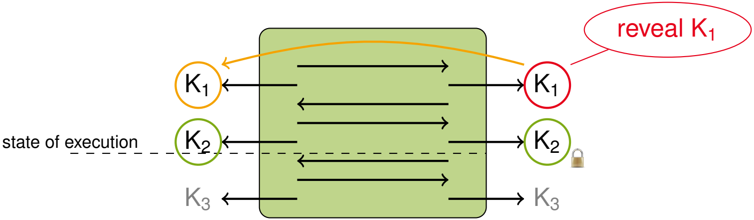
- **(Session-)Key Dependence**
  - multi-stage ⇒ derived keys might build upon each other
  - we have to disallow trivial reveal queries
  - **key-dependent**: disclosure of K_i before acceptance of K_{i+1} *may compromise* K_{i+1}
  - **key-independent**: disclosure of K_i before acceptance of K_{i+1} *without harm*



Model for Multi-Stage Key Exchange

Security aspects to consider

- **(Session-)Key Dependence**
  - multi-stage ⇒ derived keys might build upon each other
  - we have to disallow trivial reveal queries
  - **key-dependent**: disclosure of K_i before acceptance of K_{i+1} *may compromise* K_{i+1}
  - **key-independent**: disclosure of K_i before acceptance of K_{i+1} *without harm*
  - Note: revealing K_i *after* acceptance of K_{i+1} is okay (even with testing K_{i+1})



Model for Multi-Stage Key Exchange

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Security aspects to consider

- **(Session-)Key Dependence**
  - multi-stage ⇒ derived keys might build upon each other
  - we have to disallow trivial reveal queries
  - **key-dependent**: disclosure of K_i before acceptance of K_{i+1} *may compromise* K_{i+1}
  - **key-independent**: disclosure of K_i before acceptance of K_{i+1} *without harm*
  - Note: revealing K_i *after* acceptance of K_{i+1} is okay (even with testing K_{i+1})

Wait a second. . .

- key independence says: keys can be revealed at any time
- . . . so K_i can't contribute to K_{i+1}
- doesn't this mean we run a KE from scratch for each K_i?

No. see TLS: K_{i+1} depends on master key, not K_i ⇒ (session-)key independent

Model for Multi-Stage Key Exchange

Security aspects to consider (cont'd)

- **Forward Secrecy**
  - multi-stage ⇒ forward secrecy might kick in only at some stage j
  - has to be considered in case of corruptions
  - **non-forward-secret**: all session keys compromised by corruption
  - **stage-j-forward-secret**: accepted keys at stages $i \geq j$ remain secure
    ex: QUIC aims at stage-2 forward secrecy
- **Unilateral Authentication**
  - (independent of multi-stage setting)
  - distinguish one side authenticated vs. both sides authenticated
  - **unilateral authentication**: only one side authenticated (here: responder)
  - **mutual authentication**: both sides authenticated

Model for Multi-Stage Key Exchange

Let's talk about security...
For clarity we define two notions: **Match**- and **Multi-Stage** security (as BFWW'11)

Match-Security

- ensures that session identifiers effectively match partnered sessions
  - sessions with same identifier (for some stage *i*) hold the same key (at *i*)
  - sessions are partnered with the intended (authenticated) participant
    unilateral case here: responder-only authentication
  - at most two sessions share a session identifier at any stage

Model for Multi-Stage Key Exchange

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Multi-Stage Security

- Bellare–Rogaway-like key secrecy in the multi-stage setting
- adversary has to distinguish real from random keys
- adversary must not reveal *and* test same key (in single or partnered sessions)

- **Flavors**

	key-dependent	or	key-independent
+	non-forward-secret	or	stage-j-forward-secret
+	unilateral authentication	or	mutual authentication

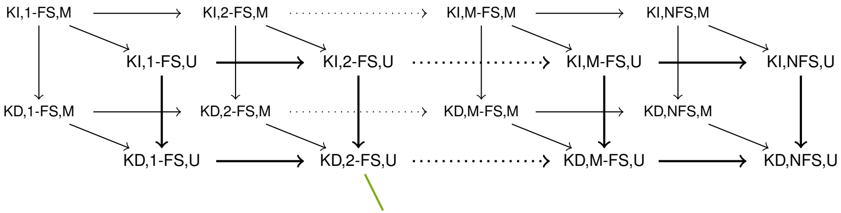
Model for Multi-Stage Key Exchange

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Multi-Stage Security Flavors

- key dependence, forward secrecy, unilateral authentication are **orthogonal**
- in principle one can think of any combination

- combinations form an ordered **hierarchy**



KI,1-FS,M KI,2-FS,M KI,M-FS,M KI,NFS,M

KI,1-FS,U KI,2-FS,U KI,M-FS,U KI,NFS,U

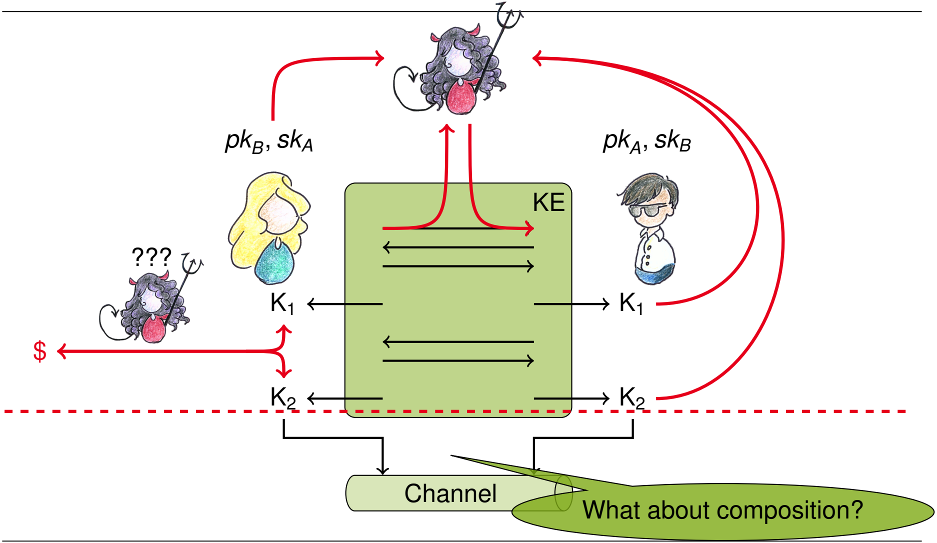
KD,1-FS,M KD,2-FS,M KD,M-FS,M KD,NFS,M

KD,1-FS,U KD,2-FS,U KD,M-FS,U KD,NFS,U

key-dependent (KD), stage-2-forward-secret (2-FS), unilateral authentication (U)

Model for Multi-Stage Key Exchange

TECHNISCHE
UNIVERSITÄT
DARMSTADT



Composition

recap: BR-secure KE + symmetric-key protocol = secure composition (BFWW'11)
can we have the same for multi-stage key exchange?

Goal

- secure multi-stage key exchange KE (with some properties. . .)
- + symmetric-key protocol Π using keys of stage i
- = secure composition $\mathsf{KE}_i; \Pi$

What's a *secure* (multi-stage) composition?

- combine games G_{KE} (for KE) and G_{Π} (for Π) to composed game $G_{\mathsf{KE}_i;\Pi}$
- $G_{\mathsf{KE}_i;\Pi}$: for every $\mathsf{K}_i \leftarrow G_{\mathsf{KE}}$ at stage i, spawn Π with K_i
- adversary $\mathcal{A}$'s task: break the protocol security in subgame G_{Π}
- allow Reveal for all stages $i' \neq i$ in $G_{\mathsf{KE}_i;\Pi}$

Composition

Our Composition Result

Take

- secure multi-stage key exchange protocol
  - key-independent
  - stage-*j*-forward-secret
  - mutual authentication (extension to unilateral case possible)
  - efficient session matching (BFWW'11)
- symmetric-key protocol
  - secure w.r.t. some security notion

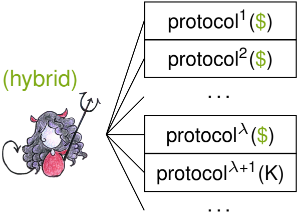
session partnering deducible from adversary communication

Then composition is secure for forward-secret stages ($i \geq j$).

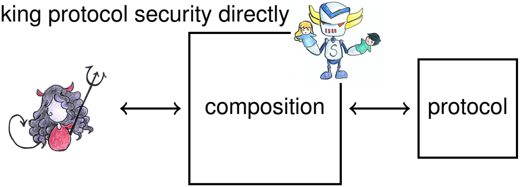
Composition

Proof idea (similar to BR-secure composition)

1. key replacement
   - gradually replace session keys K_i by random values (hybrid)
   - $\mathcal{A}$ distinguishes ⇒ we break Multi-Stage security



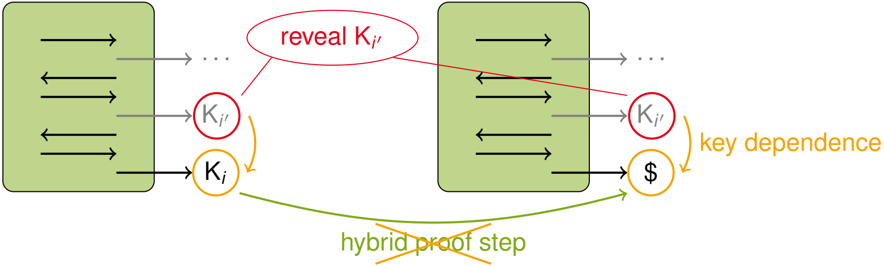
2. reduction to protocol security
   - all keys random ⇒ independent of KE
   - breaking is equivalent to breaking protocol security directly



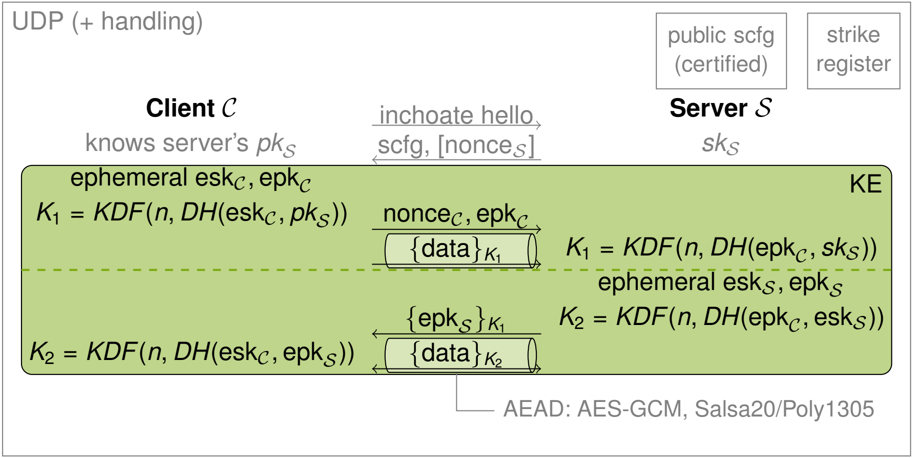
Composition

Proof ingredient example: key independence

- guarantees that compromising (reveal) $\mathsf{K}_{i'}$ ($i' < i$) doesn't affect stage-i keys
- otherwise replacing K_i with random key can be inconsistent



Google's Quick UDP Internet Connections



UDP (+ handling)

public scfg (certified)

strike register

Client $\mathcal{C}$

knows server's $pk_{\mathcal{S}}$

$\xrightarrow{\text{inchoate hello}}$

$\xleftarrow{\text{scfg, [nonce}_{\mathcal{S}}\text{]}}$

Server $\mathcal{S}$

$sk_{\mathcal{S}}$

KE

ephemeral $\text{esk}_{\mathcal{C}}, \text{epk}_{\mathcal{C}}$

$K_1 = KDF(n, DH(\text{esk}_{\mathcal{C}}, pk_{\mathcal{S}}))$

$\xrightarrow{\text{nonce}_{\mathcal{C}}, \text{epk}_{\mathcal{C}}}$

$\{\text{data}\}_{K_1}$

$K_1 = KDF(n, DH(\text{epk}_{\mathcal{C}}, sk_{\mathcal{S}}))$

ephemeral $\text{esk}_{\mathcal{S}}, \text{epk}_{\mathcal{S}}$

$K_2 = KDF(n, DH(\text{epk}_{\mathcal{C}}, \text{esk}_{\mathcal{S}}))$

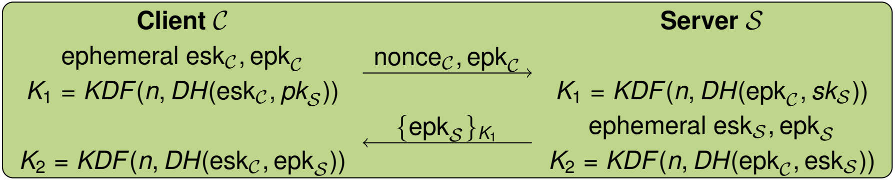
$\xleftarrow{\{\text{epk}_{\mathcal{S}}\}_{K_1}}$

$K_2 = KDF(n, DH(\text{esk}_{\mathcal{C}}, \text{epk}_{\mathcal{S}}))$

$\{\text{data}\}_{K_2}$

AEAD: AES-GCM, Salsa20/Poly1305

Google's QUIC



Client $\mathcal{C}$

ephemeral $\text{esk}_{\mathcal{C}}, \text{epk}_{\mathcal{C}}$

$\xrightarrow{\text{nonce}_{\mathcal{C}}, \text{epk}_{\mathcal{C}}}$

$K_1 = KDF(n, DH(\text{esk}_{\mathcal{C}}, pk_{\mathcal{S}}))$

$\xleftarrow{\{\text{epk}_{\mathcal{S}}\}_{K_1}}$

$K_2 = KDF(n, DH(\text{esk}_{\mathcal{C}}, \text{epk}_{\mathcal{S}}))$

Server $\mathcal{S}$

$K_1 = KDF(n, DH(\text{epk}_{\mathcal{C}}, sk_{\mathcal{S}}))$

ephemeral $\text{esk}_{\mathcal{S}}, \text{epk}_{\mathcal{S}}$

$K_2 = KDF(n, DH(\text{epk}_{\mathcal{C}}, \text{esk}_{\mathcal{S}}))$

Our (Multi-Stage) Security Result for QUIC's 0-RTT Key Exchange

- key-dependent
- stage-2-forward-secret
- (responder-authenticated) unilateral

assuming

- Gap-Diffie-Hellman is hard
- authenticated channel for 2nd message $\{\text{epk}_{\mathcal{S}}\}_{K_1}$
- (HMAC-based) key derivation function: extraction, expansion = random oracles

Google's QUIC

What about Composition?

- requirements:
  - key independence
  - stage-*j* forward secrecy
  - mutual authentication

What about Composition?

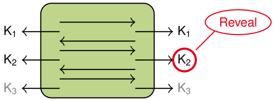
- what QUIC achieves:
  - key independence ✗
  - stage-2 forward secrecy ✓
  - unilateral authentication (✓)
- i.e., QUIC's key exchange as is doesn't allow to apply our composition result
- **but** QUIC can be easily turned into a key-independent variant **QUIC*i***:
  - TLS-like idea: keep some (master) secret not exposed in session keys
  - let an additional secret value from KDF in stage 1 enter KDF in stage 2
- QUIC*i* + composition result ⇒ (forward-)secret channels from stage 2

Summary

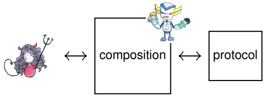
So far, KE models could not capture protocols that establish more than one key.

We

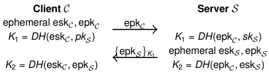
- propose a model for multi-stage key exchange



- give composition results under certain conditions (session-key independence matters!)



- show that QUIC's key exchange is multi-stage secure (key-dependent, stage-2-forward-secret, unilateral) for our composition technique: add key independence



Client $\mathcal{C}$

ephemeral $\mathsf{esk}_{\mathcal{C}}, \mathsf{epk}_{\mathcal{C}}$

$K_1 = DH(\mathsf{esk}_{\mathcal{C}}, pk_{\mathcal{S}})$

$K_2 = DH(\mathsf{esk}_{\mathcal{C}}, \mathsf{epk}_{\mathcal{S}})$

$\xrightarrow{\mathsf{epk}_{\mathcal{C}}}$

$\xleftarrow{\{\mathsf{epk}_{\mathcal{S}}\}_{K_1}}$

Server $\mathcal{S}$

$K_1 = DH(\mathsf{epk}_{\mathcal{C}}, sk_{\mathcal{S}})$

ephemeral $\mathsf{esk}_{\mathcal{S}}, \mathsf{epk}_{\mathcal{S}}$

$K_2 = DH(\mathsf{epk}_{\mathcal{C}}, \mathsf{esk}_{\mathcal{S}})$

Thank You!