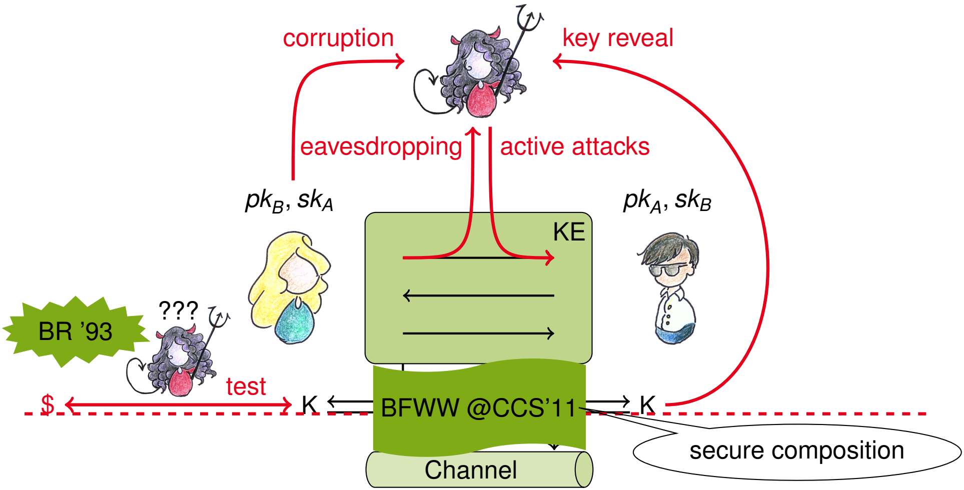
# Multi-Stage Key Exchange and the Case of Google's QUIC Protocol

Marc Fischlin and **Felix Günther**

Technische Universität Darmstadt, Germany

# Key Exchange

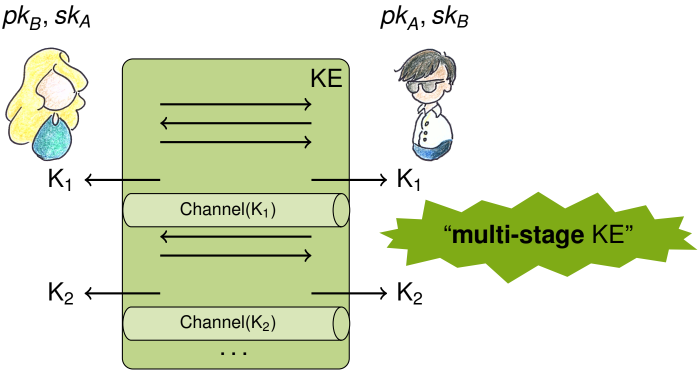
## so far…




Thanks to *Giorgia Azzurra Marson* for the drawings.

# But what if. . . ?

TECHNISCHE UNIVERSITÄT DARMSTADT




- 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?

TECHNISCHE UNIVERSITÄT DARMSTADT

- **QUIC** ("Quick UDP Internet Connections", Google 2013)
  - "low-latency transport protocol with security equivalent to TLS"
  - Diffie–Hellman-based key agreement
  - aims at 0-RTT, i.e., immediately encrypts under intermediate key $K_1$
  - later rekeys to forward-secure $K_2$
  - 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 $\mathrm{esk}_{\mathcal{C}}, \mathrm{epk}_{\mathcal{C}}$ | | | |
| $K_1 = DH(\mathrm{esk}_{\mathcal{C}}, pk_{\mathcal{S}})$ | $\xrightarrow{\mathrm{epk}_{\mathcal{C}}}$ | | **Stage 1** |
| | $\xrightarrow{\{\mathrm{data}\}_{K_1}}$ | $K_1 = DH(\mathrm{epk}_{\mathcal{C}}, sk_{\mathcal{S}})$ | |
| | | ephemeral $\mathrm{esk}_{\mathcal{S}}, \mathrm{epk}_{\mathcal{S}}$ | |
| | $\xleftarrow{\{\mathrm{epk}_{\mathcal{S}}\}_{K_1}}$ | $K_2 = DH(\mathrm{epk}_{\mathcal{C}}, \mathrm{esk}_{\mathcal{S}})$ | **Stage 2** |
| $K_2 = DH(\mathrm{esk}_{\mathcal{C}}, \mathrm{epk}_{\mathcal{S}})$ | $\xleftrightarrow{\{\mathrm{data}\}_{K_2}}$ | | |

# Should we care?

TECHNISCHE UNIVERSITÄT DARMSTADT

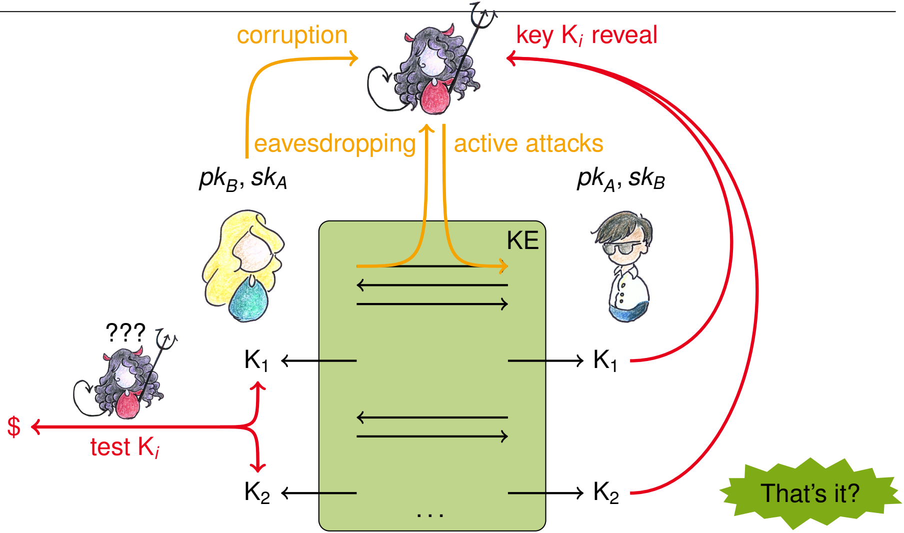
- **QUIC** (“Quick UDP Internet Connections”, Google 2013)
  - “low-latency transport protocol with security equivalent to TLS”
  - Diffie–Hellman-based key agreement
  - aims at 0-RTT, i.e., immediately encrypts under intermediate key $K_1$
  - later rekeys to forward-secure $K_2$
  - intermediate key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

- **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

# Model for Multi-Stage Key Exchange

TECHNISCHE
UNIVERSITÄT
DARMSTADT



## 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 $\mathcal{C}$** | | **Server $\mathcal{S}$** |
|---|---|---|
| ephemeral $\mathrm{esk}_{\mathcal{C}}, \mathrm{epk}_{\mathcal{C}}$ | $\xrightarrow{\mathrm{epk}_{\mathcal{C}}}$ | |
| $K_1 = DH(\mathrm{esk}_{\mathcal{C}}, pk_{\mathcal{S}})$ | | $K_1 = DH(\mathrm{epk}_{\mathcal{C}}, sk_{\mathcal{S}})$ |
| | $\xleftarrow{\{\mathrm{epk}_{\mathcal{S}}\}_{K_1}}$ | ephemeral $\mathrm{esk}_{\mathcal{S}}, \mathrm{epk}_{\mathcal{S}}$ |
| $K_2 = DH(\mathrm{esk}_{\mathcal{C}}, \mathrm{epk}_{\mathcal{S}})$ | | $K_2 = DH(\mathrm{epk}_{\mathcal{C}}, \mathrm{esk}_{\mathcal{S}})$ |

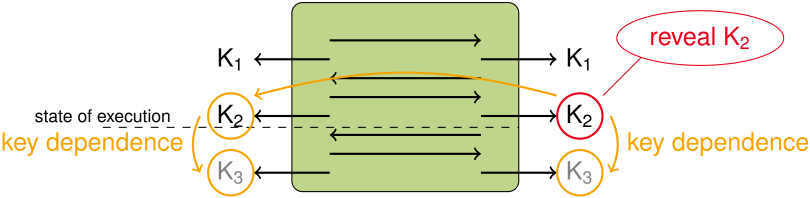
disclosure of $K_1$ compromises $K_2$

# Model for Multi-Stage Key Exchange

TECHNISCHE UNIVERSITÄT DARMSTADT

## Security Aspects to consider

- **(Session-)Key Dependence**
  - multi-stage $\Rightarrow$ derived keys might build upon each other
  - we have to disallow trivial reveal queries
  - **key-dependent** KE: disclosure of $K_i$ before acceptance of $K_{i+1}$ *compromises* $K_{i+1}$
  - **key-independent** KE: disclosure of $K_i$ before acceptance of $K_{i+1}$ *without harm*
  - <u>Note:</u> 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 (cont'd)

- **Forward Security**
  - multi-stage $\Rightarrow$ forward security might kick in only at some stage $j$
  - has to be considered in case of corruptions
  - **non-forward-secure** KE: all session keys compromised by corruption
  - **stage-$j$-forward-secure** KE: accepted keys at stages $i \geq j$ remain secure
    ex: QUIC aims at stage-2 forward security
- **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

TECHNISCHE UNIVERSITÄT DARMSTADT

Let's talk about security. . .

## 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**

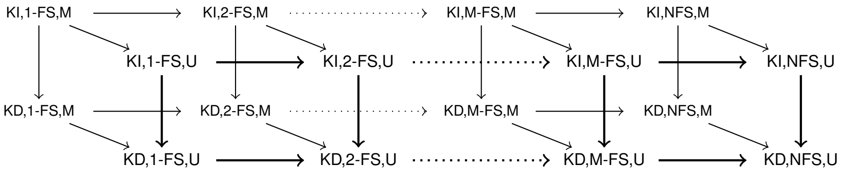
| | | | |
|---|---|---|---|
| | key-dependent | or | key-independent |
| + | non-forward-secure | or | stage-$j$-forward-secure |
| + | unilateral authentication | or | mutual authentication |

# Model for Multi-Stage Key Exchange

TECHNISCHE UNIVERSITÄT DARMSTADT

## Multi-Stage Security Flavors

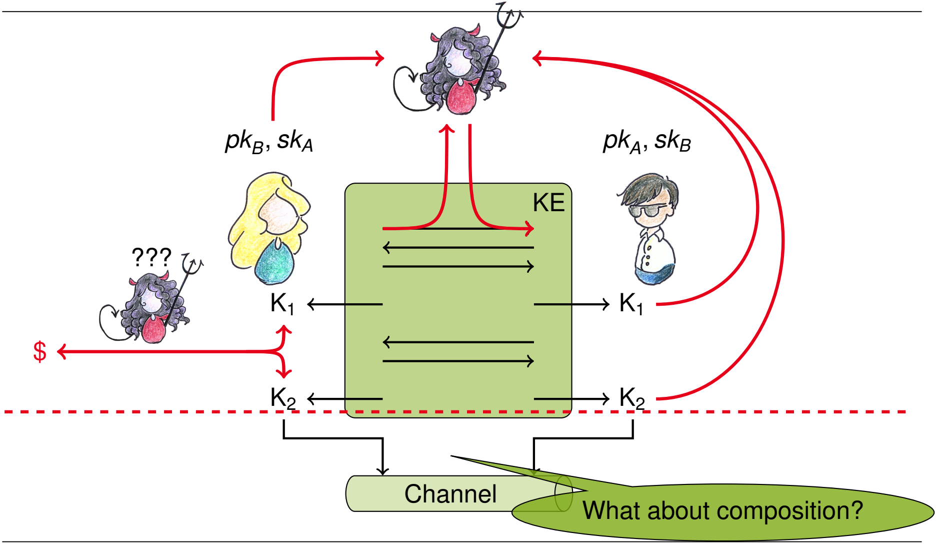
- key dependence, forward security, unilateral authentication are **orthogonal**
- in principle one can think of any combination
- combinations form an ordered **hierarchy**




key-dependent (KD), stage-2-forward-secure (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 (with some properties. . . )
- \+ symmetric-key protocol using keys of stage *i*
- = secure composition

# Composition

TECHNISCHE UNIVERSITÄT DARMSTADT

## Our Composition Result

Take

- secure multi-stage key exchange protocol
  - key-independent
  - stage-$j$-forward-secure
  - 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

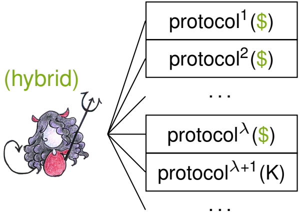
Then composition is secure for forward-secure stages ($i \geq j$).

# Composition

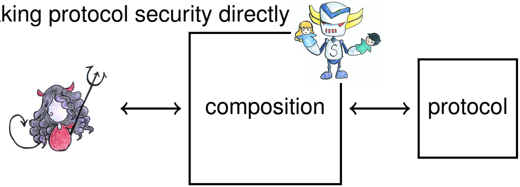
TECHNISCHE UNIVERSITÄT DARMSTADT

## Proof idea (similar to BR-secure composition)

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




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

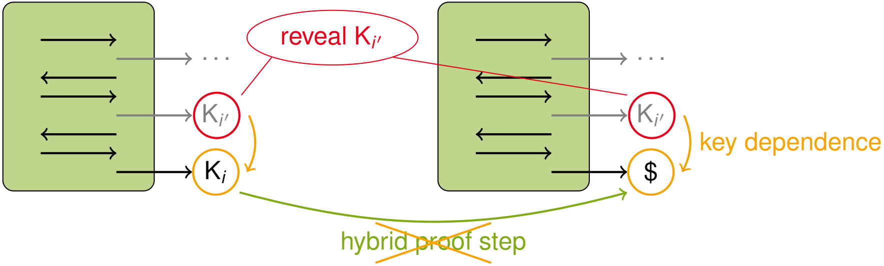


# Composition

TECHNISCHE UNIVERSITÄT DARMSTADT

## Proof ingredient example: key independence

- guarantees that compromising (reveal) $\mathsf{K}_{i'}$ ($i' < i$) doesn't affect stage-$i$ keys
- otherwise replacing $\mathsf{K}_i$ 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 $pk_{\mathcal{S}}$

inchoate hello $\rightarrow$

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

**Server $\mathcal{S}$**

$sk_{\mathcal{S}}$

KE

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

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

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

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

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

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

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

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

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

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

AEAD: AES-GCM, Salsa20/Poly1305

# Google's QUIC

| **Client $\mathcal{C}$** | | **Server $\mathcal{S}$** |
|---|---|---|
| ephemeral $\mathsf{esk}_\mathcal{C}, \mathsf{epk}_\mathcal{C}$ | $\xrightarrow{\mathsf{nonce}_\mathcal{C}, \mathsf{epk}_\mathcal{C}}$ | |
| $K_1 = KDF(n, DH(\mathsf{esk}_\mathcal{C}, pk_\mathcal{S}))$ | | $K_1 = KDF(n, DH(\mathsf{epk}_\mathcal{C}, sk_\mathcal{S}))$ |
| | $\xleftarrow{\{\mathsf{epk}_\mathcal{S}\}_{K_1}}$ | ephemeral $\mathsf{esk}_\mathcal{S}, \mathsf{epk}_\mathcal{S}$ |
| $K_2 = KDF(n, DH(\mathsf{esk}_\mathcal{C}, \mathsf{epk}_\mathcal{S}))$ | | $K_2 = KDF(n, DH(\mathsf{epk}_\mathcal{C}, \mathsf{esk}_\mathcal{S}))$ |

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

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

assuming

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

TECHNISCHE UNIVERSITÄT DARMSTADT

## What about Composition?

- requirements:
  - key independence
  - stage-$j$ forward security
  - mutual authentication

## What about Composition?

- what QUIC achieves:
  - key independence ✗
  - stage-2 forward security ✓
  - unilateral authentication (✓)
- **but** QUIC can be easily turned into a key-independent variant **QUIC*i***:
  - TLS-like idea: keep some (master) secret not exposed in Reveals
  - let an additional secret value from KDF in stage 1 enter KDF in stage 2
- QUIC*i* + composition result $\Rightarrow$ (forward-)secure channels from stage 2

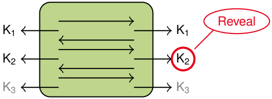
# Summary

TECHNISCHE UNIVERSITÄT DARMSTADT

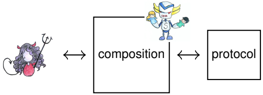
So far, KE models could not capture protocols that establish more than one key.

We

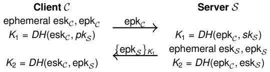
- propose a model for multi-stage key exchange




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




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




Thank You!