# Multi-Stage Key Exchange

## When one key is not enough. . .

TECHNISCHE UNIVERSITÄT DARMSTADT

Marc Fischlin and **Felix Günther**

Technische Universität Darmstadt

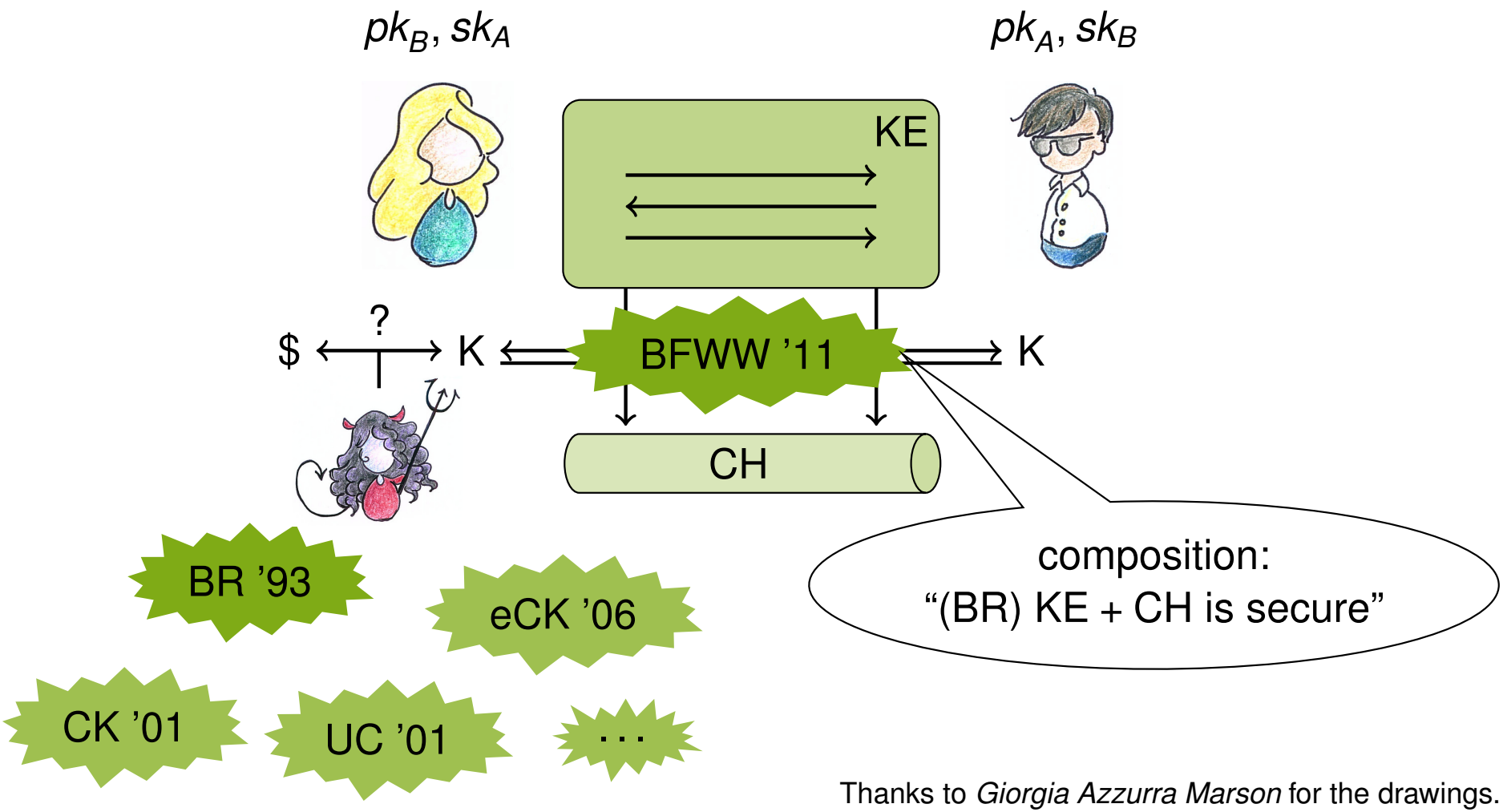

Cryptoplexity

Cryptography & Complexity Theory
Technische Universität Darmstadt
www.cryptoplexity.de

EC SPRIDE
EUROPEAN CENTER FOR SECURITY AND PRIVACY BY DESIGN

# Key Exchange
## so far…

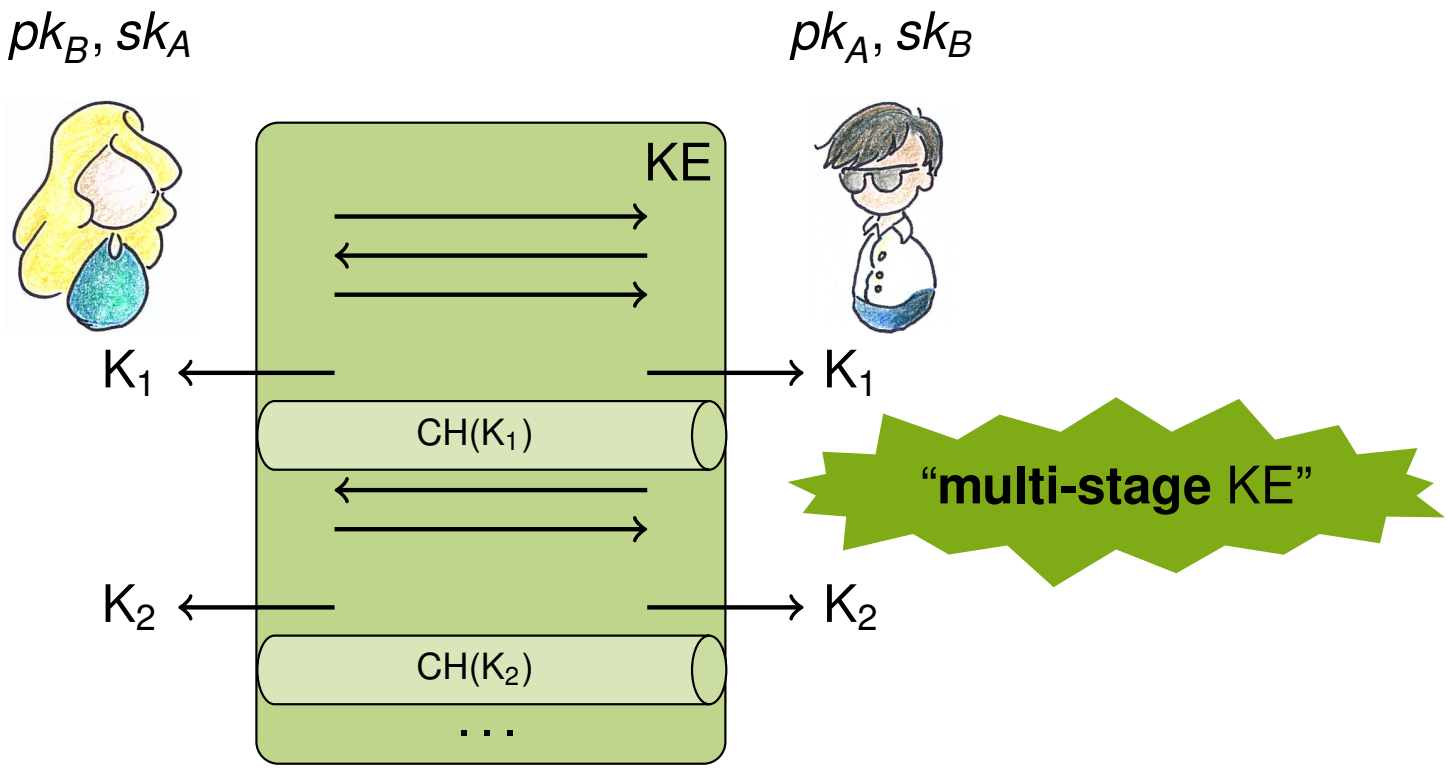
TECHNISCHE UNIVERSITÄT DARMSTADT




Thanks to *Giorgia Azzurra Marson* for the drawings.

# But what if. . . ?

TECHNISCHE UNIVERSITÄT DARMSTADT

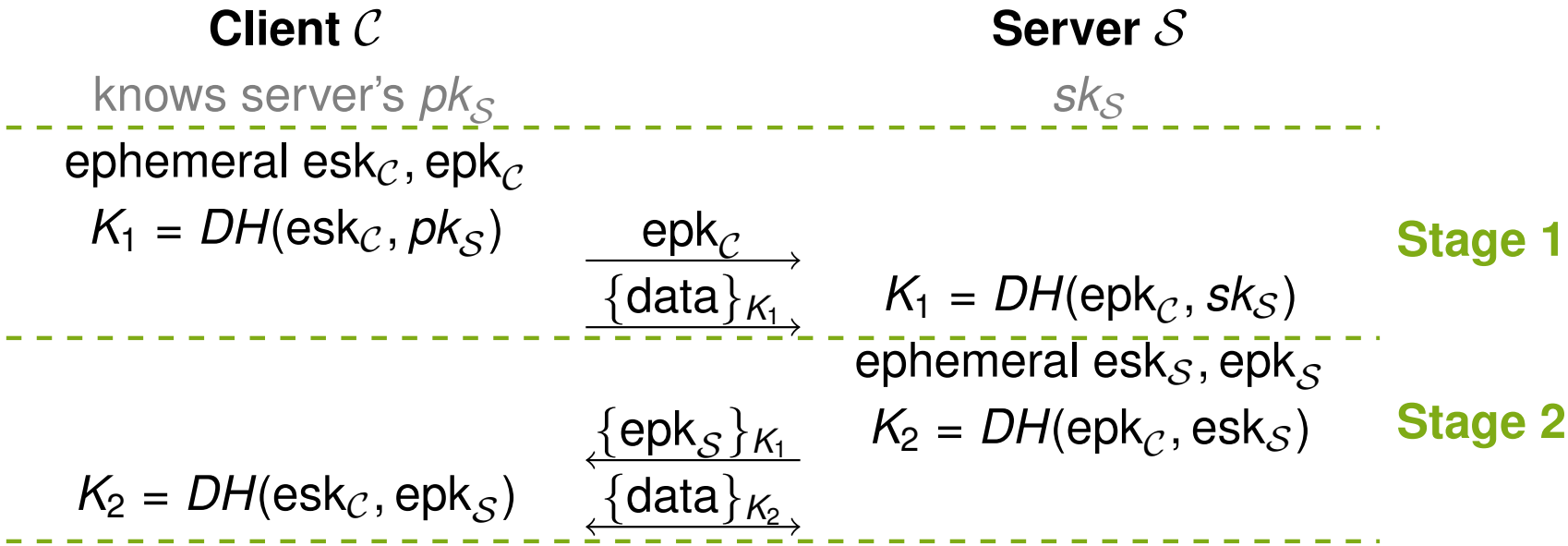



- key exchange establishes more than one key?
- . . . even uses the intermediary keys within the key exchange or channel?

# 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)



# 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

# Outline

TECHNISCHE UNIVERSITÄT DARMSTADT

- A **Model** for Multi-Stage Key Exchange
- What about **Composition**?
- A quick look at **QUIC**

# Model for Multi-Stage Key Exchange

TECHNISCHE UNIVERSITÄT DARMSTADT

inspired by BFWW @ CCS'11, BR-like, with composition in mind... (see later)

## Adversary Model / Queries

active adversary $\mathcal{A}$ interacts through queries

NewSession: Create new session for two participants.

Send: Send message to a session.

Reveal: Reveal session key (of stage $i$).

Corrupt: Corrupt participant (i.e., reveal $\mathsf{sk}_U$).

Test: Test session for real-or-random key.

NewTempKey: Create new (QUIC-motivated) "temporary keys".
(QUIC uses server ephemeral keys for ~60sec in multiple sessions)

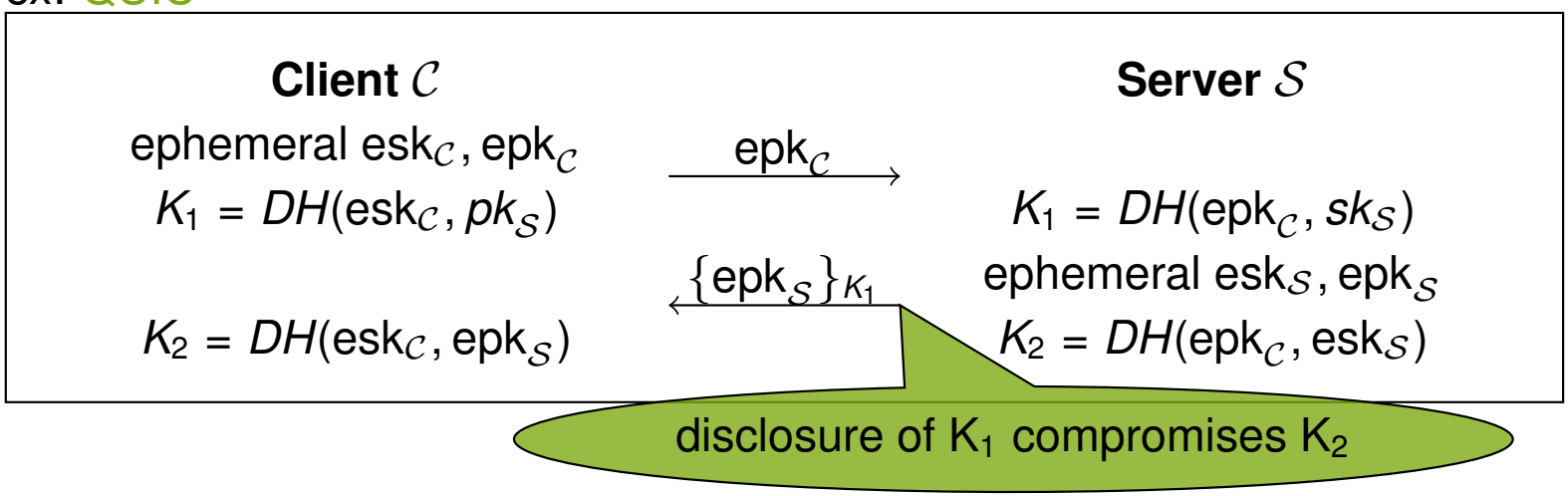
# Model for Multi-Stage Key Exchange

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



# 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*
  - Note: revealing $K_i$ *after* acceptance of $K_{i+1}$ is okay (even with Test on $K_{i+1}$)

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 $K_i$ $\Rightarrow$ (session-)key independent

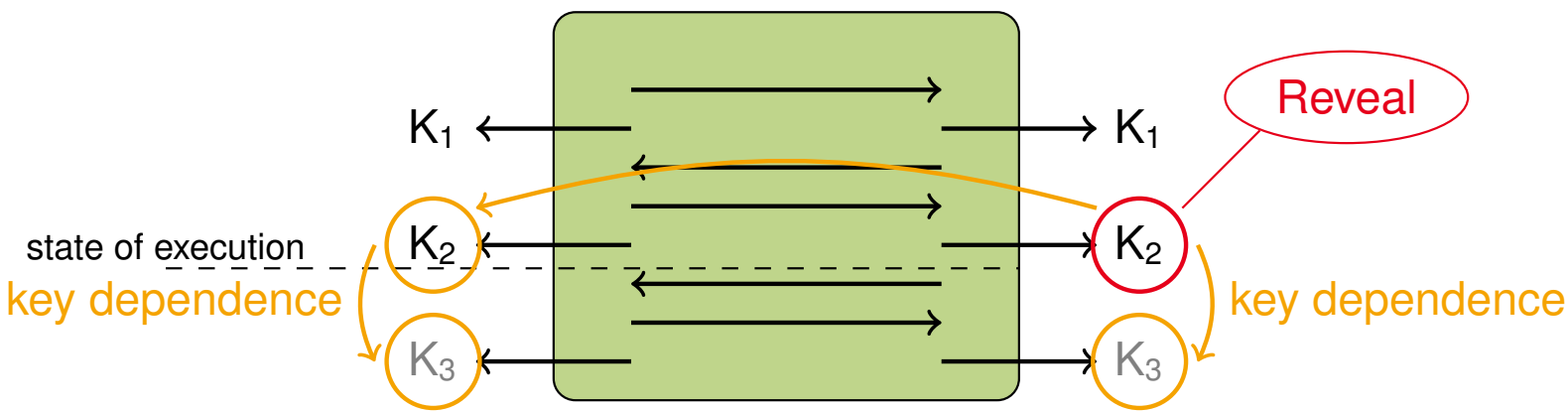
# Model for Multi-Stage Key Exchange

TECHNISCHE UNIVERSITÄT DARMSTADT

## Adversarial Queries, refined

- **Reveal**
  - so far: (accepted) partnered session key gets revealed as well
  - key dependence: future keys are compromised, too, on reveal of $K_i$ in stage = $i$
    - reveal all $K_j$ for $j > i$ in this session
    - reveal all $K_j$ for $j > i$ in partnered session with $K_i$ accepted

Example: ($K_2$s just accepted)



# Model for Multi-Stage Key Exchange

TECHNISCHE UNIVERSITÄT DARMSTADT

## Security Aspects to consider (cont'd)

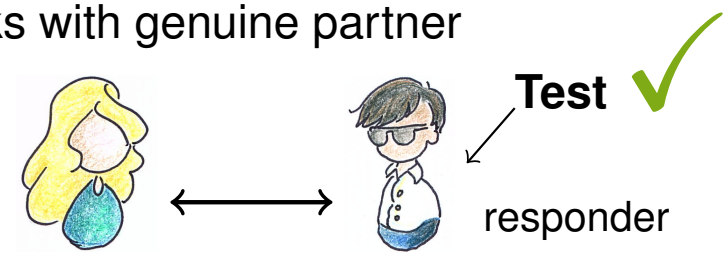
- **Forward Security**
  - multi-stage $\Rightarrow$ forward security might kick in only at some stage $j$
  - has to be considered in case of corruptions
  - **non-forward-secure** KE: all session keys compromised on Corrupt
  - **stage-$j$-forward-secure** KE: accepted keys at stages $i \geq j$ remain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

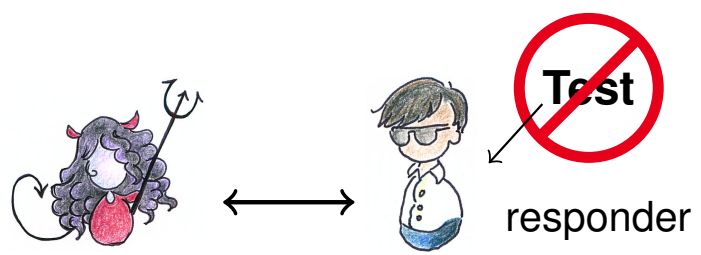
# Model for Multi-Stage Key Exchange

TECHNISCHE UNIVERSITÄT DARMSTADT

## Adversarial Queries, refined (cont'd)

- **Corrupt**
  - non–forward security: all session keys get revealed
  - stage-$j$ forward security: accepted keys $K_i$ at stages $i \geq j$ remain secure
  - key dependence: future keys get revealed as well (as for Reveal queries)
- **Test**
  - multi-stage $\Rightarrow$ keys get tested and protocol *continues*
  - use tested (genuine or random) key *in subsequent steps* to prevent trivial attacks
  - unilateral (responder-only) authentication: test on responder side only allowed if it talks with genuine partner



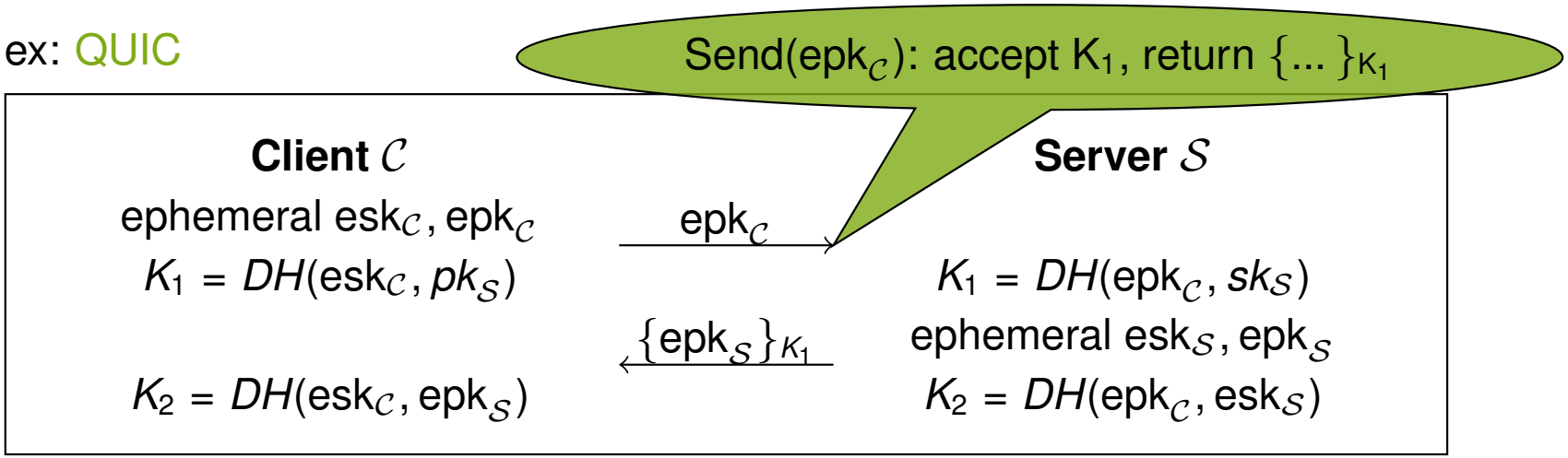



# Model for Multi-Stage Key Exchange

TECHNISCHE UNIVERSITÄT DARMSTADT

## Adversarial Queries, refined (cont'd)

- **Send**
  - multi-stage $\Rightarrow$ keys get accepted and protocol *continues*
  - reply after acceptance of $K_i$ might already use $K_i$
  - ex: QUIC




  - Problem: $\mathcal{A}$ cannot Test such keys (as state accepted$_i$ is too volatile)
  - Solution: suspend KE execution on acceptance, $\mathcal{A}$ gets special Send(continue)

# Model for Multi-Stage Key Exchange

TECHNISCHE UNIVERSITÄT DARMSTADT

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 sid effectively match the 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
- queries: NewSession, Send, Reveal, Corrupt

# Model for Multi-Stage Key Exchange

TECHNISCHE UNIVERSITÄT DARMSTADT

## Multi-Stage-security

- Bellare–Rogaway-like key secrecy in the multi-stage setting
- $\mathcal{A}$ has to guess bit $b_{\text{test}}$ ($b_{\text{test}} = 0 \iff$ Test returns random key)
- $\mathcal{A}$ must not reveal *and* test same key (in single or partnered sessions)
- queries: NewSession, Send, Reveal, Corrupt, Test
- to be Multi-Stage-secure, KE must also be Match-secure

- **Flavors**

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

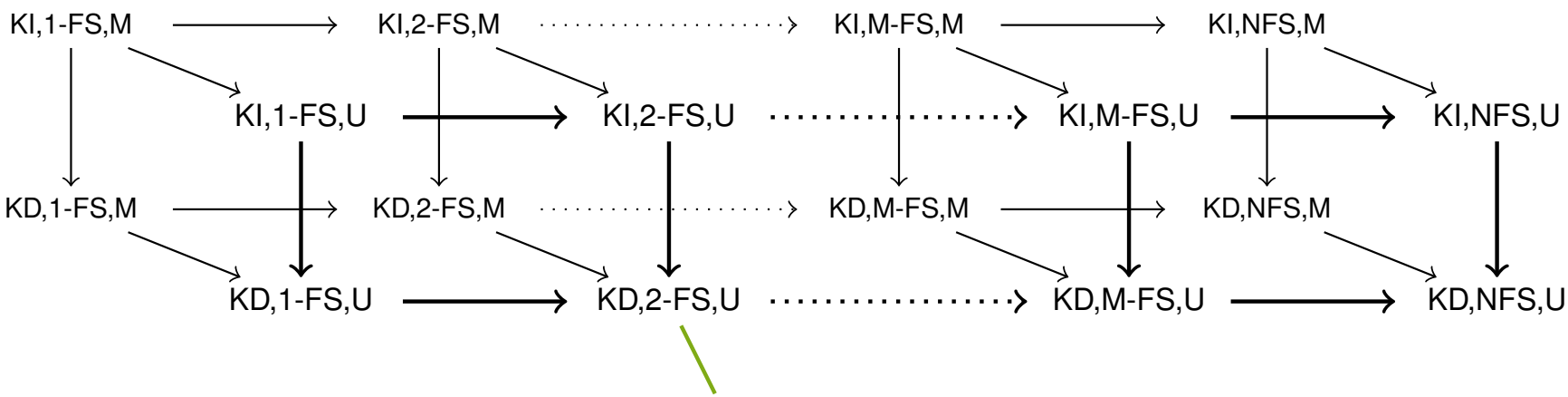
# Model for Multi-Stage Key Exchange

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)

# Composition

TECHNISCHE UNIVERSITÄT DARMSTADT

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...)
- + (arbitrary) symmetric-key protocol $\Pi$
- = secure composition KE; $\Pi$

# Composition

TECHNISCHE UNIVERSITÄT DARMSTADT

## What's a *secure* composition? (BFWW'11)

- combine games $G_{\mathsf{KE}}$ (for KE) and $G_{\Pi}$ (for Π) to composed game $G_{\mathsf{KE};\Pi}$
- $G_{\mathsf{KE};\Pi}$: for every K $\leftarrow G_{\mathsf{KE}}$, spawn Π with K
- $\mathcal{A}$'s task: break Π security in subgame $G_{\Pi}$
- queries for both subgames, except for

  Reveal: session key compromise captured (if at all) in $G_{\Pi}$

  Test: only administrative for $G_{\mathsf{KE}}$

## Multi-stage composition

- $\mathsf{KE}_i;\Pi$ spawns Π from stage-$i$ keys
- all other keys unused $\Rightarrow$ Reveal allowed for stages $i' \neq i$ in $G_{\mathsf{KE}_i;\Pi}$

# Composition

TECHNISCHE UNIVERSITÄT DARMSTADT

## Our Composition Result

Take

- multi-stage key exchange protocol KE
  - key-independent
  - stage-$j$-forward-secure
  - mutual authentication
  - efficient session matching (BFWW'11)
- symmetric-key protocol $\Pi$
  - secure w.r.t. some $G_\Pi$

session partnering deducible from $\mathcal{A} \leftrightarrow G_{\mathsf{KE};\Pi}$ communication

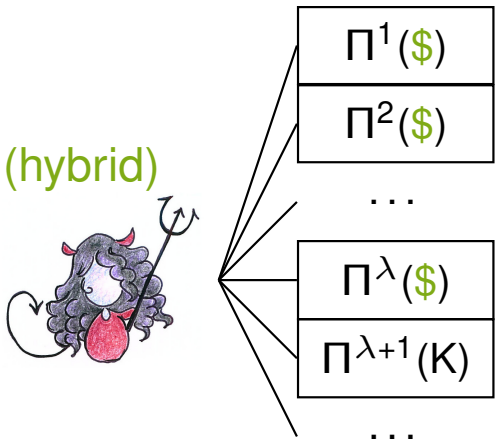
- Then composition $\mathsf{KE}_i;\Pi$ is secure for forward-secure stages ($i \geq j$)

# Composition

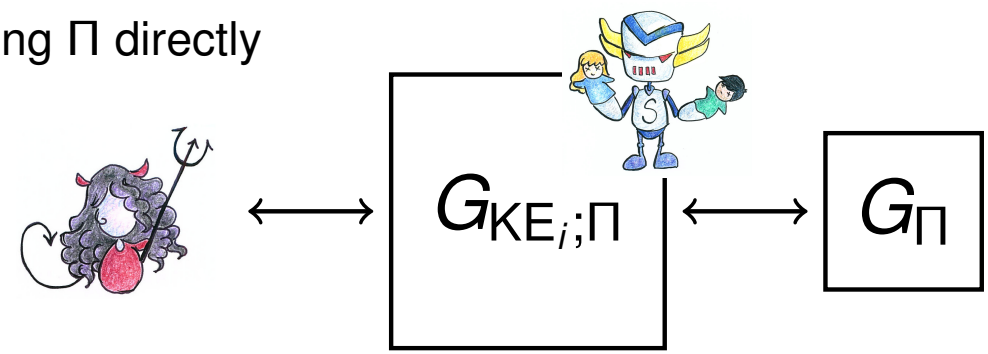
TECHNISCHE UNIVERSITÄT DARMSTADT

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

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




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



# Composition

TECHNISCHE UNIVERSITÄT DARMSTADT

## Hybrid ingredients

- **key independence**
  - guarantees that Reveal of $K_{i'}$ ($i' \neq i$) does not affect stage-$i$ keys
  - otherwise, replacement of $K_i$ with random could be detected
- **forward security of stage $i$**
  - guarantees that simulation of $\Pi$ accepted $K_i$s is sound
  - otherwise, replacement of $K_i$ with random could be detected
- **session matching**
  - allows the reduction to handle partnered sessions consistently

# Composition

TECHNISCHE UNIVERSITÄT DARMSTADT

## Hybrid ingredients (cont'd)

- **mutual authentication**
  - guarantees that Test queries are allowed for each accepted $K_i$
  - recall: unilateral authentication forbids test on responder without genuine partner


  - composition cannot provide protection in these cases (reduction can't replace keys here with random ones)

extension to the **unilateral authentication** case however is possible: restrict composition s.t. Π *not spawned* when unpartnered responder accepts

# A quick look at QUIC

TECHNISCHE UNIVERSITÄT DARMSTADT

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

## Our (Multi-Stage) Security Result for QUIC 0-RTT

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

assuming

- Gap-Diffie-Hellman
- authenticated channel for 2nd message $\{\mathsf{epk}_{\mathcal{S}}\}_{K_1}$
- key derivation function (HKDF): extraction = ROM, expansion = PRG

# A quick look at QUIC

TECHNISCHE UNIVERSITÄT DARMSTADT

## What about Composition?

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

# A quick look at QUIC

TECHNISCHE UNIVERSITÄT DARMSTADT

## 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 intermediate KDF (extraction) value of stage 1 enter KDF in stage 2

- QUIC*i* + composition result $\Rightarrow$ (forward-)secure channels from stage 2

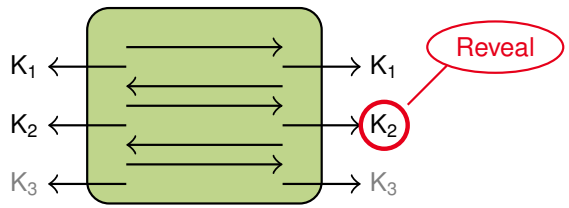
# Summary

TECHNISCHE UNIVERSITÄT DARMSTADT

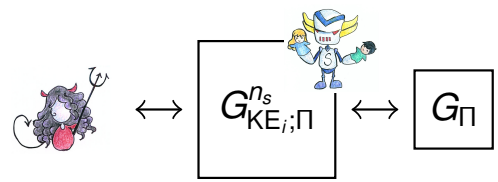
in practice, protocols apparently sometimes want to establish more than one key

We

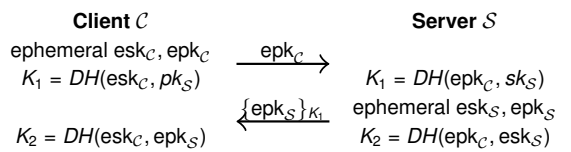
- propose a model for multi-stage key exchange




- give composition results under certain conditions (key independence, forward security, . . . )




- analyze the multi-stage security of Google's QUIC (key-dependent, stage-2-forward-secure, unilateral)




Thank You!

# Appendix: QUIC's 0-RTT Handshake

## Expanded Description

TECHNISCHE UNIVERSITÄT DARMSTADT

| **Client** $\mathcal{C}$ | | **Server** $\mathcal{S}$ |
|---|---|---|
| server's static public key $pk_{\mathcal{S}}$ | | $sk_{\mathcal{S}}$ |
| generate ephemeral keys $\text{esk}_{\mathcal{C}}, \text{epk}_{\mathcal{C}}$<br>generate $\text{nonce}_{\mathcal{C}}$ | | |
| | $\xrightarrow{\text{nonce}_{\mathcal{C}}, \text{aux}_{\mathcal{C}}, \text{epk}_{\mathcal{C}}}$ | |
| | | [generate $\text{nonce}_{\mathcal{S}}$] |
| $D_1 = \text{DH}(\text{esk}_{\mathcal{C}}, pk_{\mathcal{S}})$ | | $D_1 = \text{DH}(\text{epk}_{\mathcal{C}}, sk_{\mathcal{S}})$ |
| $\text{PRK}_1 = H(D_1, \text{nonce}_{\mathcal{C}}, [\text{nonce}_{\mathcal{S}}])$ | | $\text{PRK}_1 = H(D_1, \text{nonce}_{\mathcal{C}}, [\text{nonce}_{\mathcal{S}}])$ |
| $\text{K}_1 = \text{PRG}(\text{PRK}_1, \text{info}_1)$ | | $\text{K}_1 = \text{PRG}(\text{PRK}_1, \text{info}_1)$ |
| | | use temporary keys $\text{tsk}_{\mathcal{S}}, \text{tpk}_{\mathcal{S}}$ |
| | $\xleftarrow{\{[\text{nonce}_{\mathcal{S}}], \text{aux}_{\mathcal{S}}, \text{tpk}_{\mathcal{S}}\}_{K_1}}$ | |
| $D_2 = \text{DH}(\text{esk}_{\mathcal{C}}, \text{tpk}_{\mathcal{S}})$ | | $D_2 = \text{DH}(\text{epk}_{\mathcal{C}}, \text{tsk}_{\mathcal{S}})$ |
| $\text{PRK}_2 = H(D_2, \text{nonce}_{\mathcal{C}}, [\text{nonce}_{\mathcal{S}}])$ | | $\text{PRK}_2 = H(D_2, \text{nonce}_{\mathcal{C}}, [\text{nonce}_{\mathcal{S}}])$ |
| $\text{K}_2 = \text{PRG}(\text{PRK}_2, \text{info}_2)$ | | $\text{K}_2 = \text{PRG}(\text{PRK}_2, \text{info}_2)$ |

# Appendix: Composition
## Proof Details

TECHNISCHE UNIVERSITÄT DARMSTADT

### Hybrid argument

- reduction $\mathcal{B}$ plays against Multi-Stage game
- simulates $G_\Pi$ on its own
- forwards KE-queries NewSession, Reveal (for $i' \neq i$), Corrupt, Send
- handles Send queries resulting in $\text{accepted}_i$ as follows:
  - partnered session already accepted? use same key in $G_\Pi$
  - counter $< \lambda$? sample $K_i$ at random (counter = #accepted sessions)
  - counter $= \lambda$? set $K_i \leftarrow$ Test (if $b_{\text{test}} = 0$ random, else real)
  - counter $> \lambda$? set $K_i \leftarrow$ Reveal

$b_{\text{test}} = 0 \Rightarrow \mathcal{B}$ simulates $G^{\lambda}_{\text{KE}_i;\Pi}$

$b_{\text{test}} = 1 \Rightarrow \mathcal{B}$ simulates $G^{\lambda-1}_{\text{KE}_i;\Pi}$

$\Big\}$ $\mathcal{A}$'s distinguishing probability bounded by Multi-Stage security