Part II
Bellare-Rogaway Model (Active Adversaries)

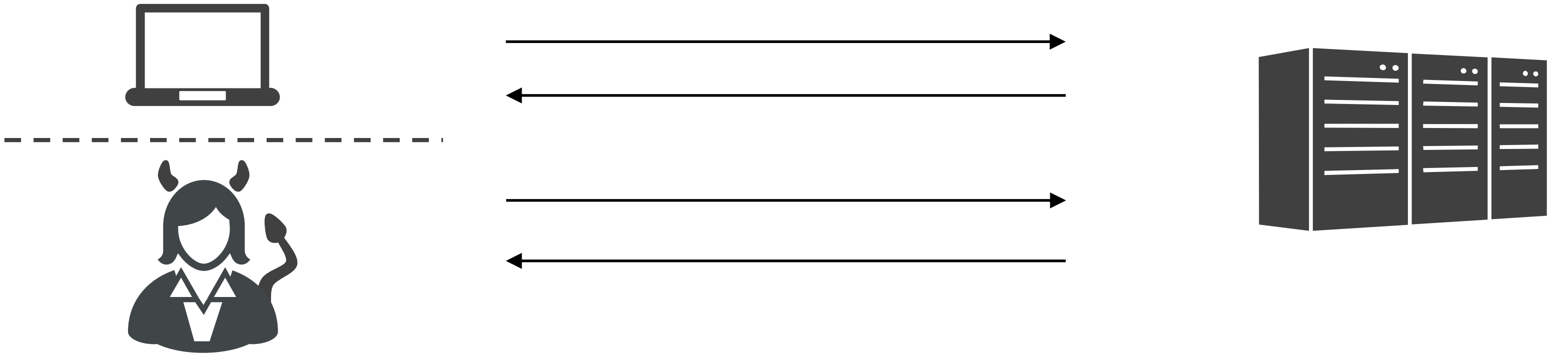
TECHNISCHE UNIVERSITÄT DARMSTADT

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

8th BIU Winter School on Key Exchange, 2018

Marc Fischlin

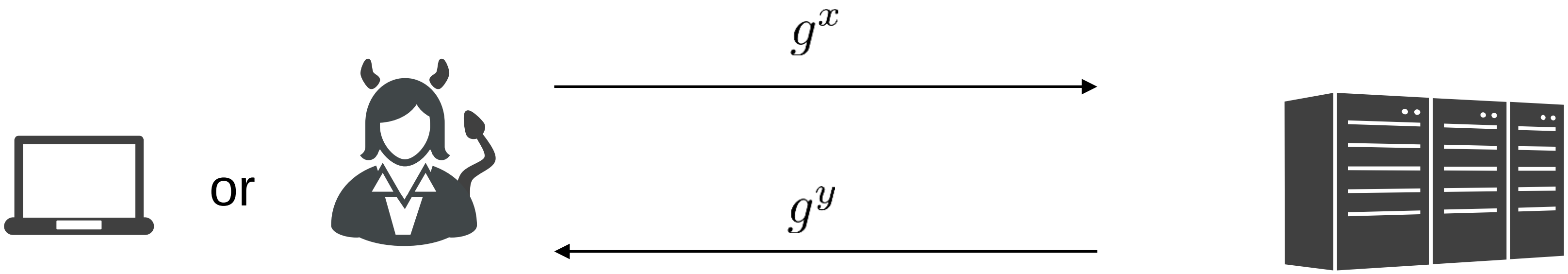
Active Attacks



Adversary may tamper, drop, or inject messages in executions

Identities

Identities?


or

g^x

g^y

In the passive security model
both scenarios are identical from server‘s view

need identities to distinguish good and bad cases in active model

Identities!

certified pk_C (via cert_C)

sk_C

certified pk_S (via cert_S)

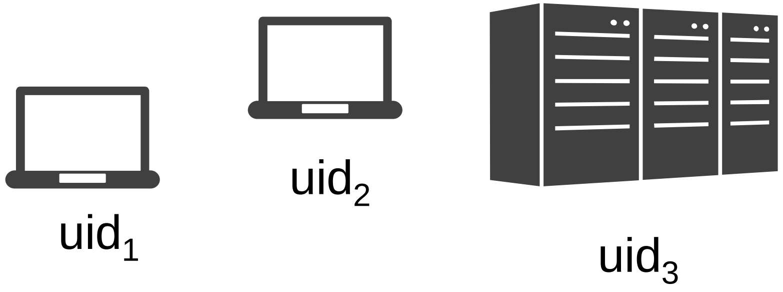
sk_S

both parties also output intended partner identity pid

Warning: We do not consider revocation nor registering adversarial keys here!

Implications for Security Model

Users are assigned user id uid



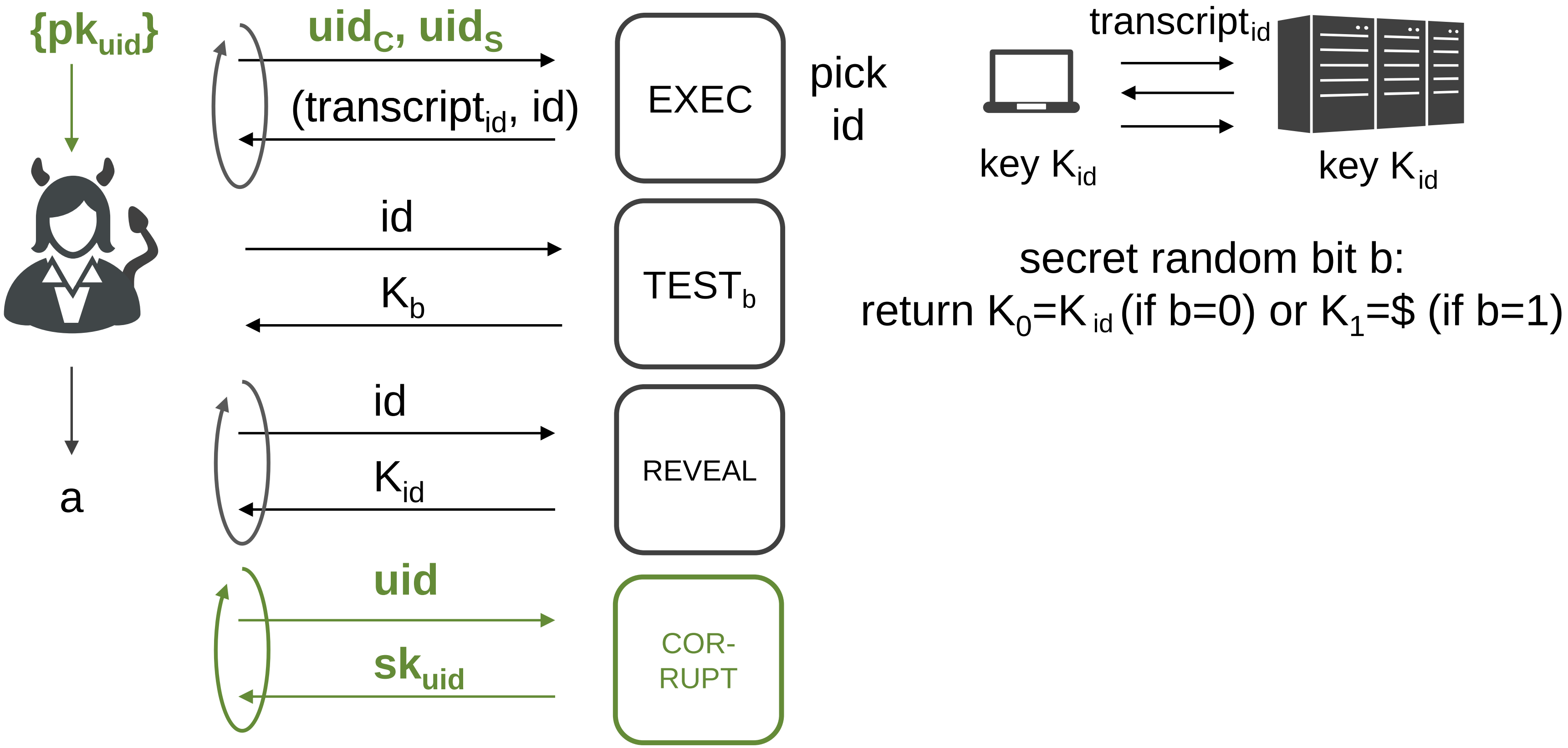
uid_1 uid_2 uid_3

Each party with identity uid receives (pk_{uid}, sk_{uid}, $cert_{uid}$)

Adversary may recover sk_{uid} from pk_{uid}

pk_{uid} → → sk_{uid}

Adding Corruption



$\{pk_{uid}\}$

uid_C, uid_S

$(transcript_{id}, id)$

EXEC

pick id

$transcript_{id}$

key K_{id}

key K_{id}

id

K_b

$TEST_b$

secret random bit b:
return $K_0=K_{id}$ (if b=0) or $K_1=\$$ (if b=1)

id

K_{id}

REVEAL

a

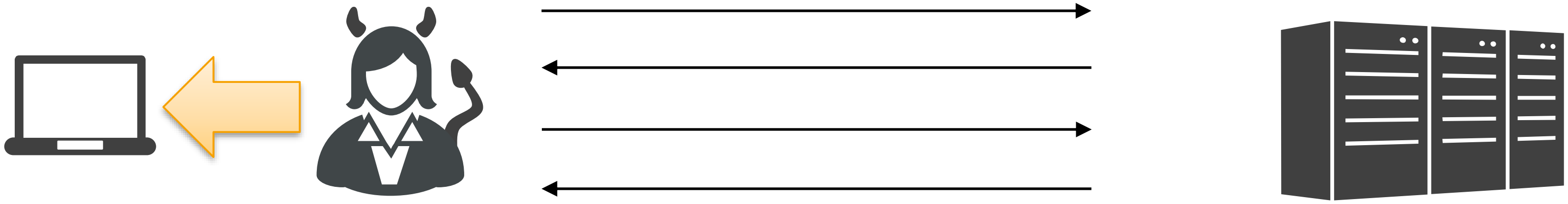
uid

sk_{uid}

COR-
RUPT

New Attack Surfaces

certified pk$_C$ (via cert$_C$)



sk$_C$ →

key K

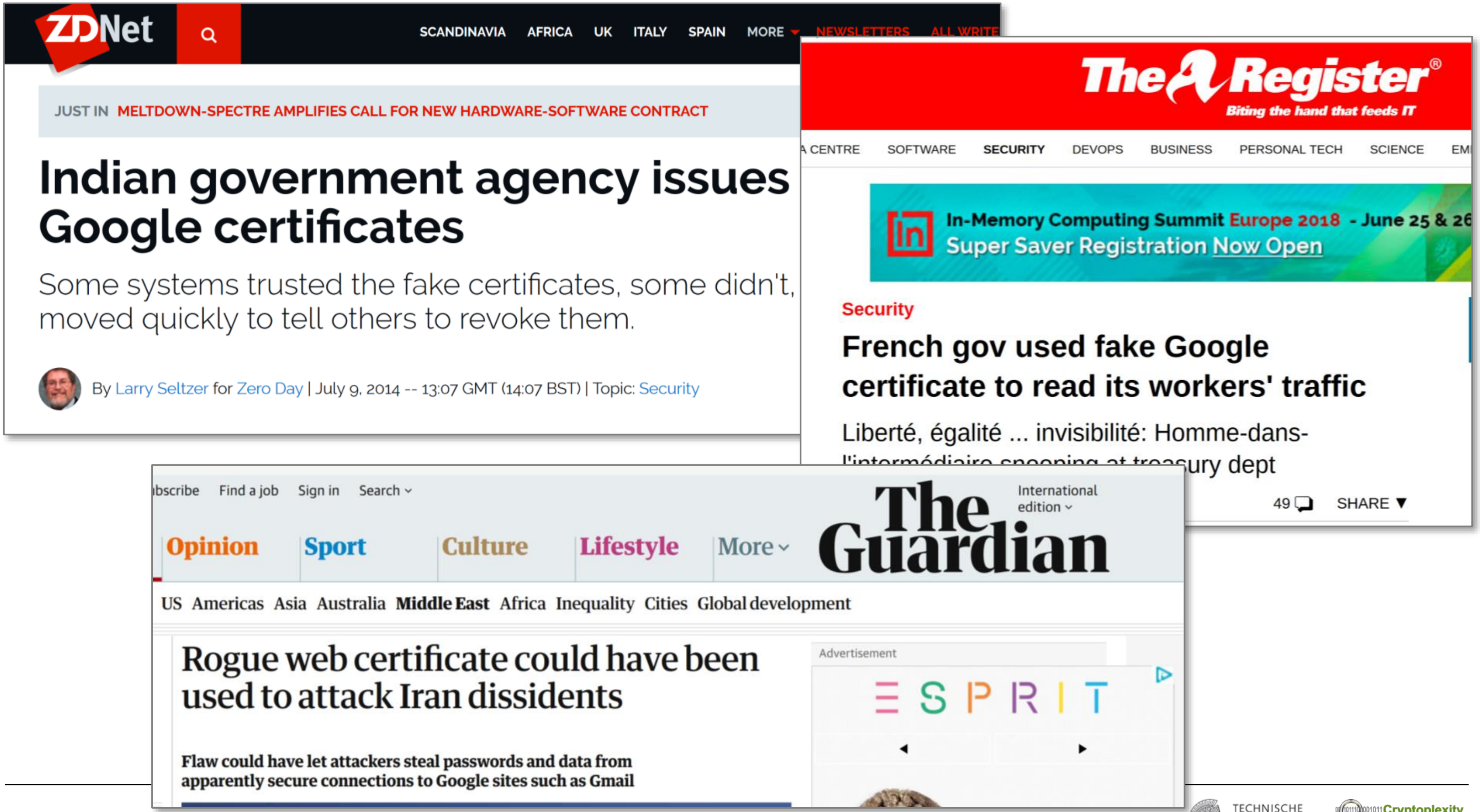
key K

(intended parter is C)

1. Corrupt client to learn sk$_C$
2. impersonate client to derive Key K
3. TEST server key

Attacks via false Identities

not via corruption,
but through
rogue certificates



JUST IN MELTDOWN-SPECTRE AMPLIFIES CALL FOR NEW HARDWARE-SOFTWARE CONTRACT

Indian government agency issues Google certificates

Some systems trusted the fake certificates, some didn't, moved quickly to tell others to revoke them.

By Larry Seltzer for Zero Day | July 9, 2014 -- 13:07 GMT (14:07 BST) | Topic: Security

Security

French gov used fake Google certificate to read its workers' traffic

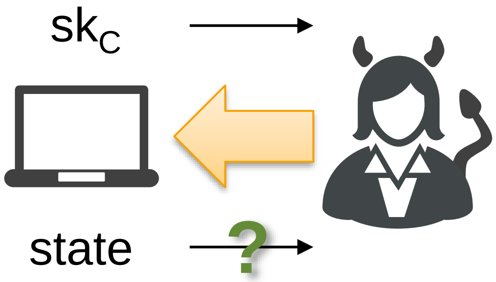
Liberté, égalité ... invisibilité: Homme-dans-l'intermédiaire snooping at treasury dept

Rogue web certificate could have been used to attack Iran dissidents

Flaw could have let attackers steal passwords and data from apparently secure connections to Google sites such as Gmail

Extensions: Corruption

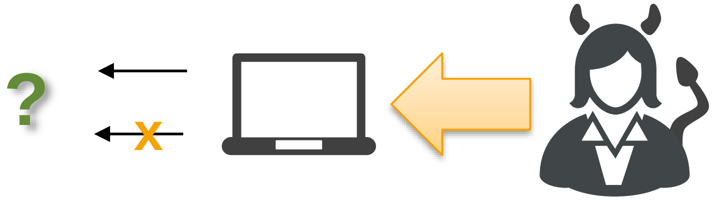
State



Adversary learns sk_C but also state (randomness,...)?

(„weak“ vs. „strong“ corruption)

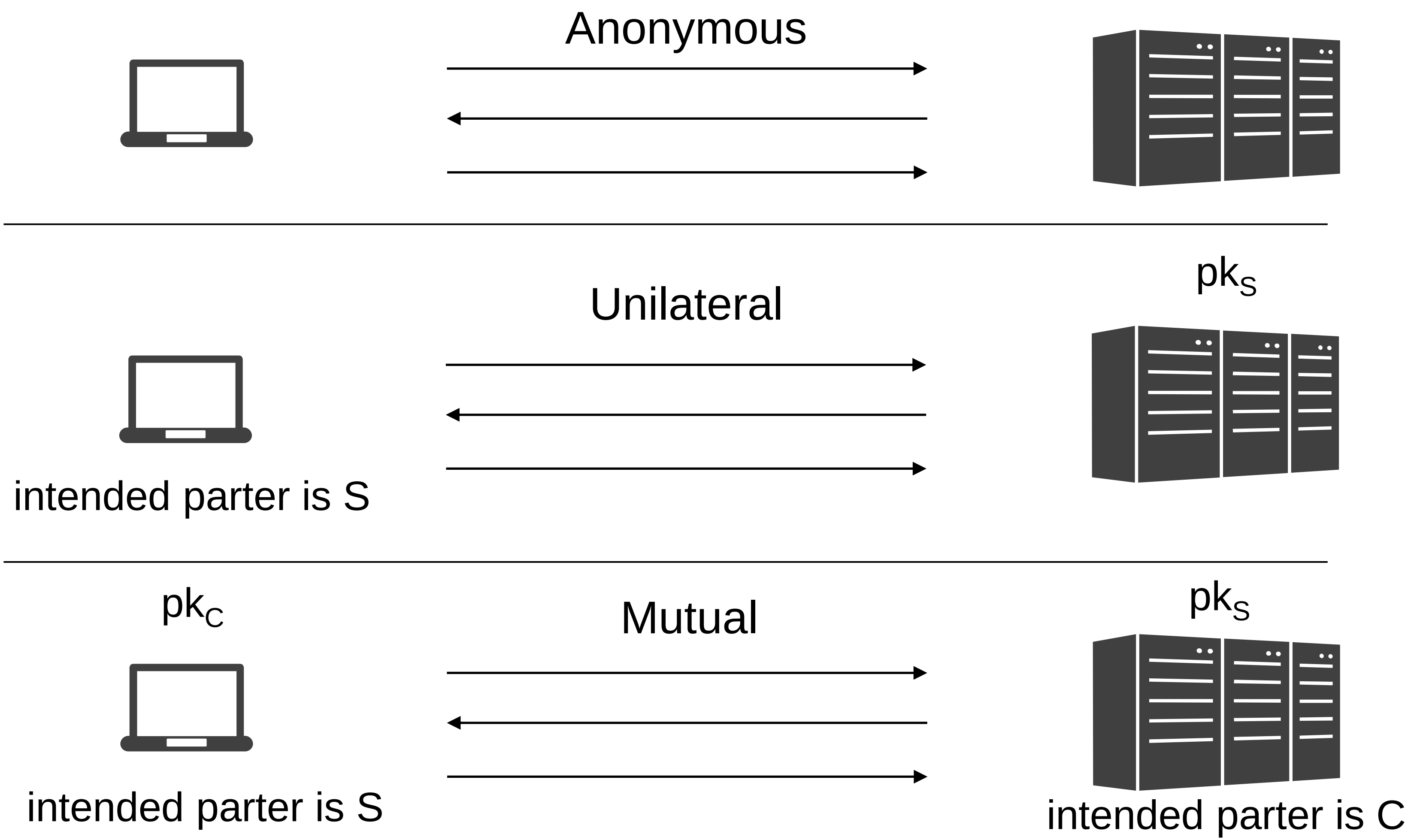
Complete take-over



Can client still run executions after corruption?

Here: Adversary only gets sk_C and corrupt party can still be active

Authenticating the Partner



Anonymous

Unilateral

pk_S

intended parter is S

Mutual

pk_C

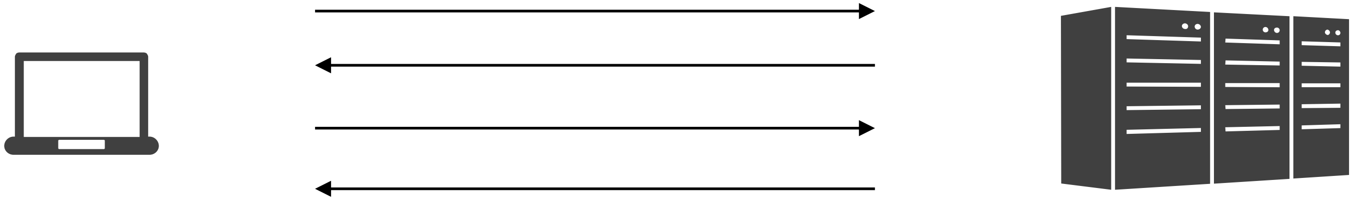
pk_S

intended parter is S

intended parter is C

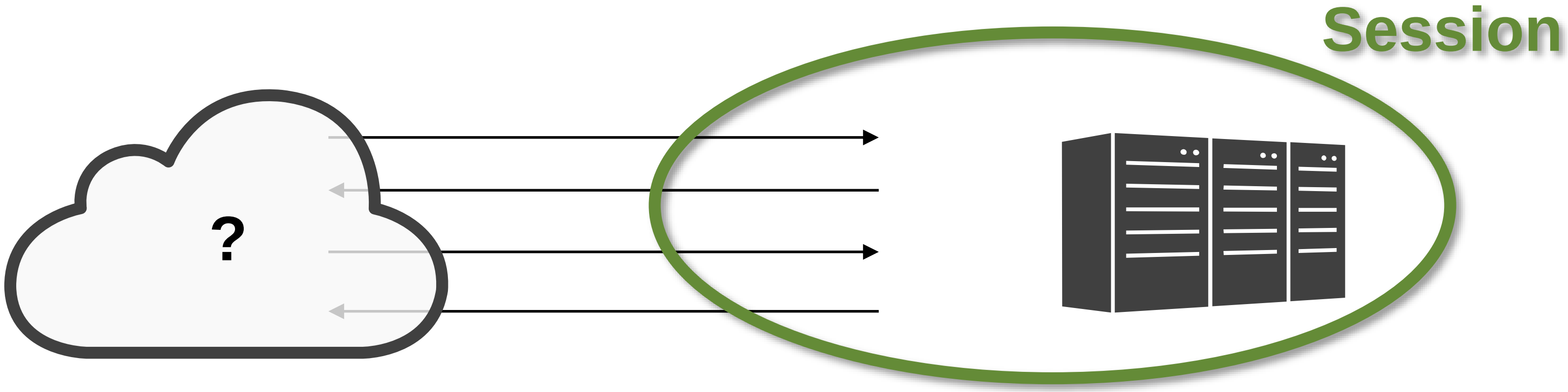
Sessions

Conceptual Change: Sessions

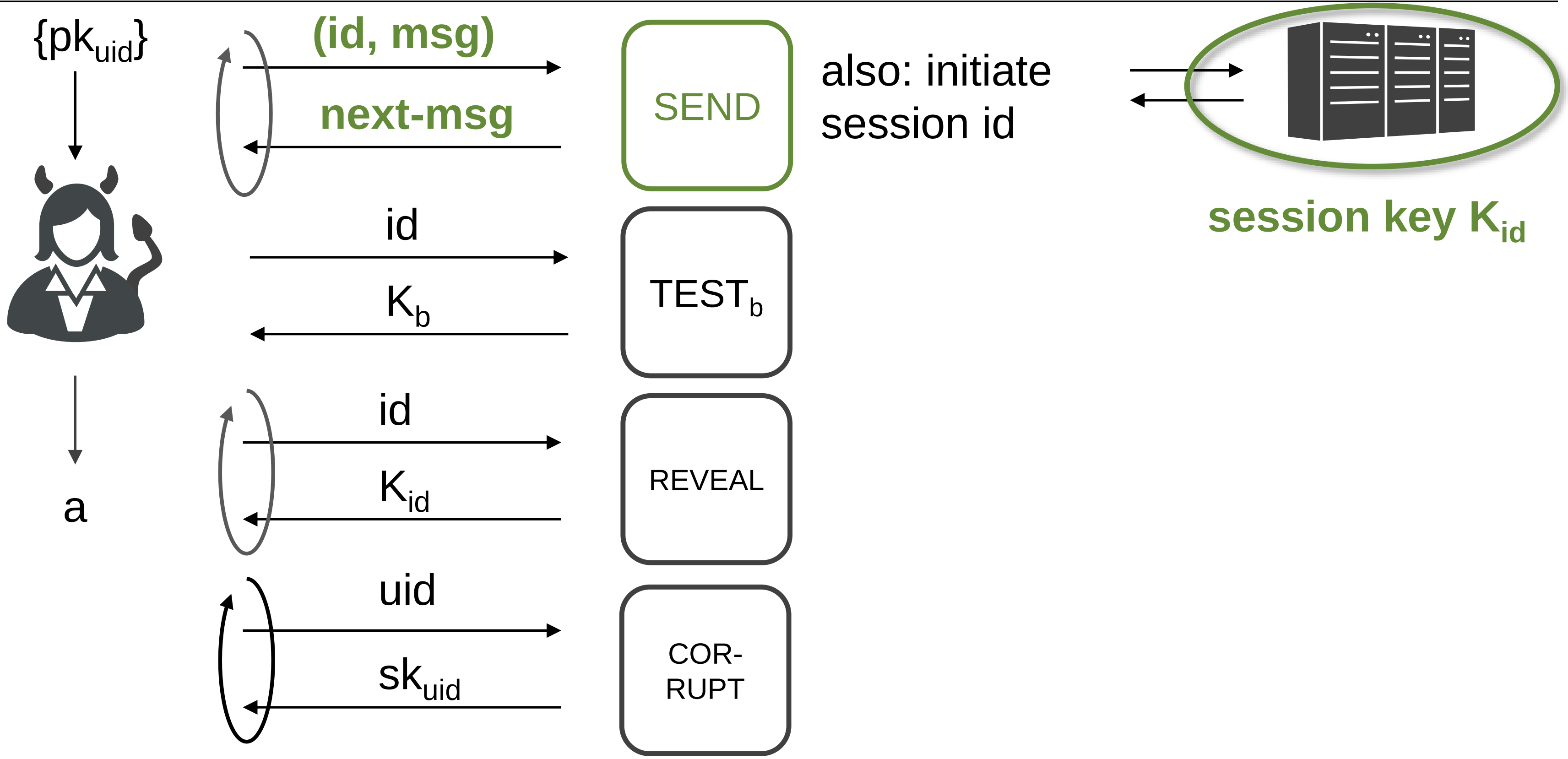


Passive adversaries: honest parties run execution

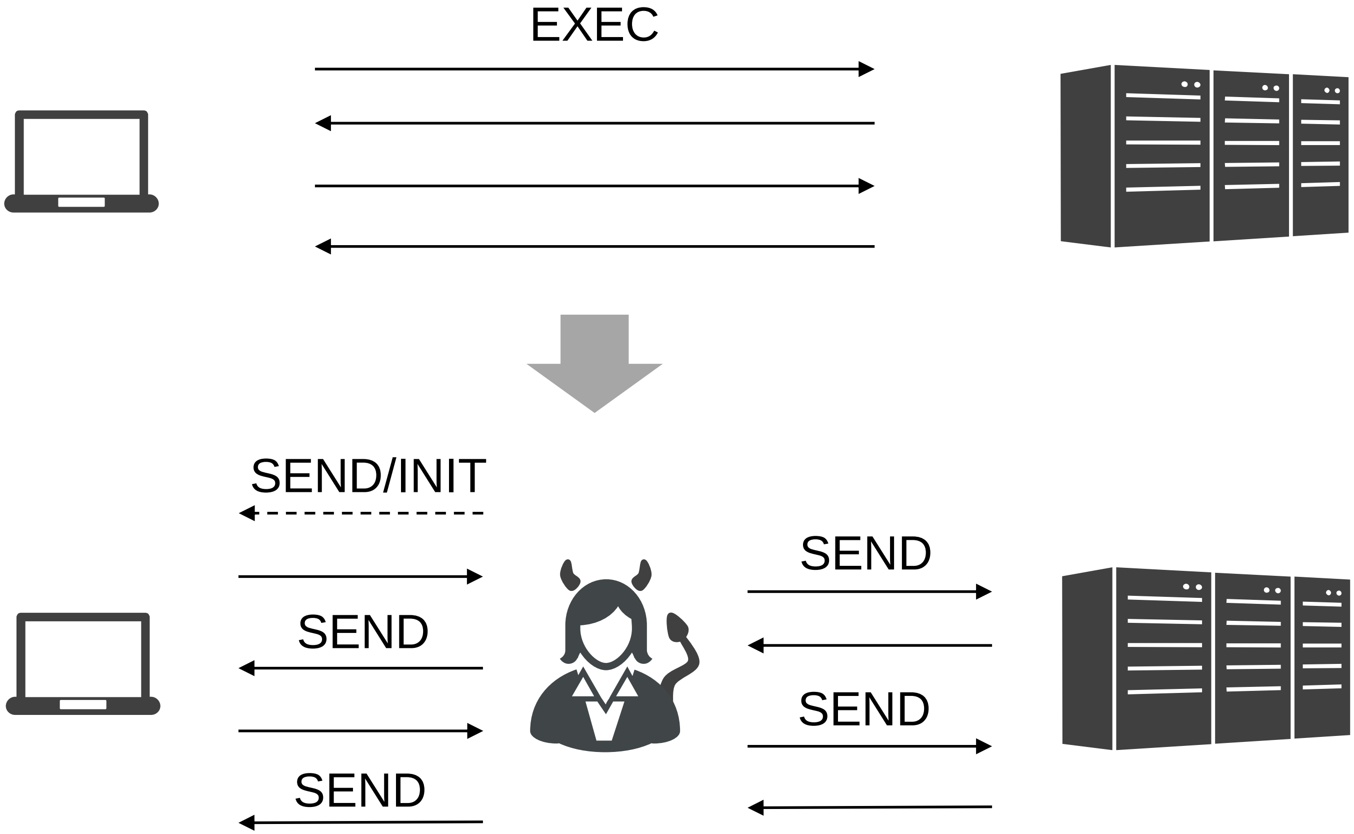
Active adversaries: unclear if there is partner at all



Adding SEND


$\{pk_{uid}\}$
(id, msg)
next-msg
SEND
also: initiate session id
session key K_{id}
id
K_b
$TEST_b$
id
K_{id}
REVEAL
a
uid
sk_{uid}
COR-
RUPT

Replacing EXEC with SEND

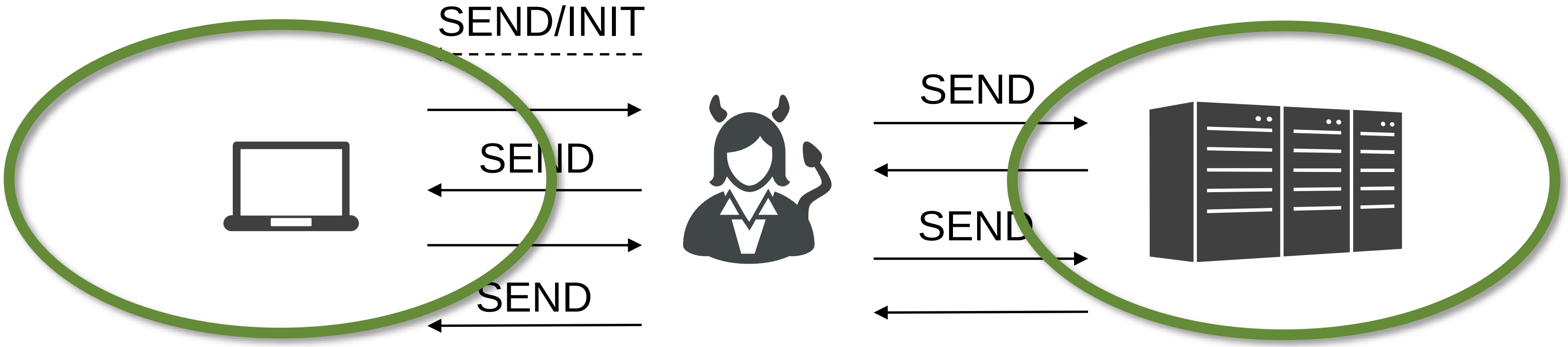


Warning: for forward secrecy later it is advantageous to also use EXEC

Freshness Condition?

Adversary should not be allowed to
TEST one party and REVEAL other party
in the following scenario:

need a notion that sessions belong together

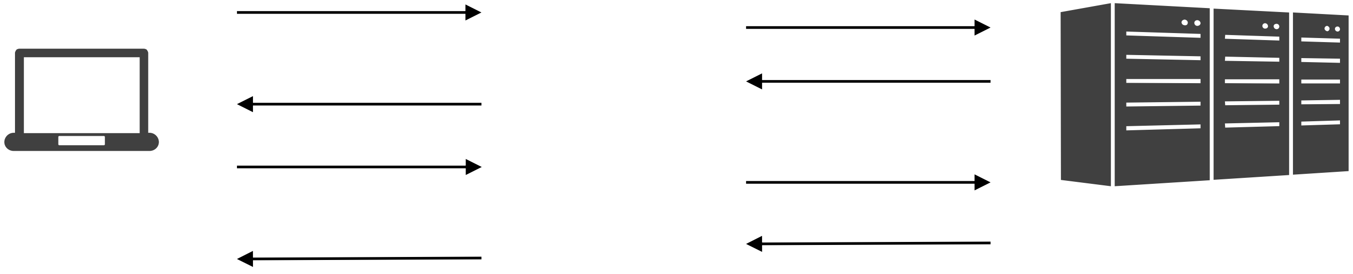


Active but somewhat passive attack: Client and Server derive same key

Session Matching or Partnering

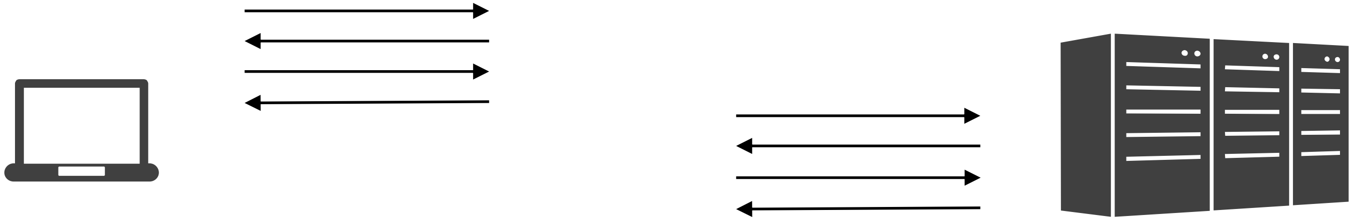
Bellare-Rogaway (BR93)	**Matching conversations**	Crypto `93
Bellare-Rogaway (BR95)	**Partnering Function**	STOC `95
Bellare-Pointcheval-Rogaway (BPR00)	**Session identifiers**	Eurocrypt 2000

Matching Conversations

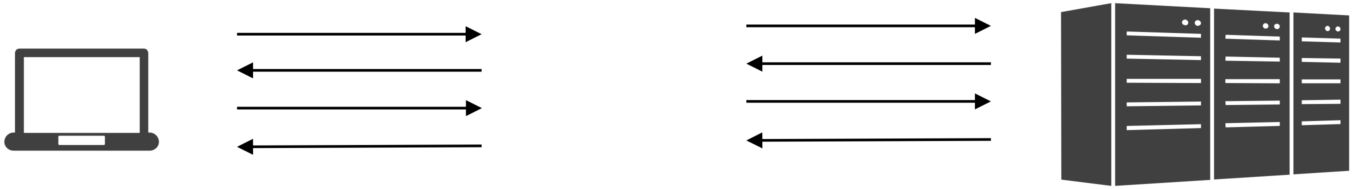


Sessions are partnered if
identical transcripts and in chronological order

Sometimes defined without chronological order:



Partnering Functions

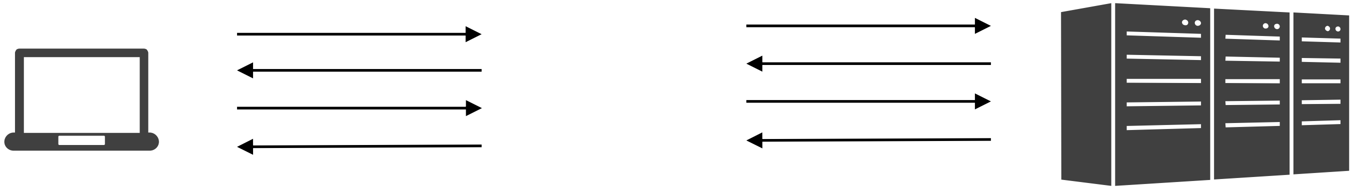


Uses notion of
(not necessarily efficiently computable)
partnering function f: {transcripts} → {id}

Sessions are partnered if
f(transcript) = f(transcript‘)

Not used anywhere anymore

Session Identifiers



specify session identifier sid

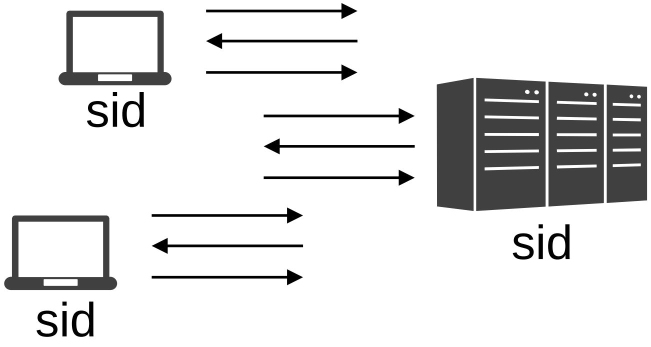
Sessions are partnered if
sid = sid‘

sid usually defined through (partial) transcript

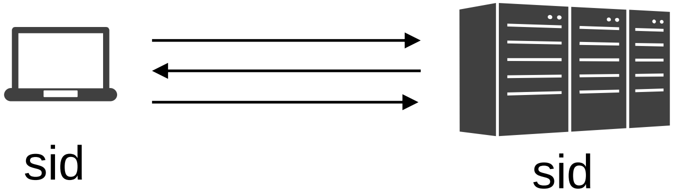
Restrictions Apply

1. Session identifiers should be unique:

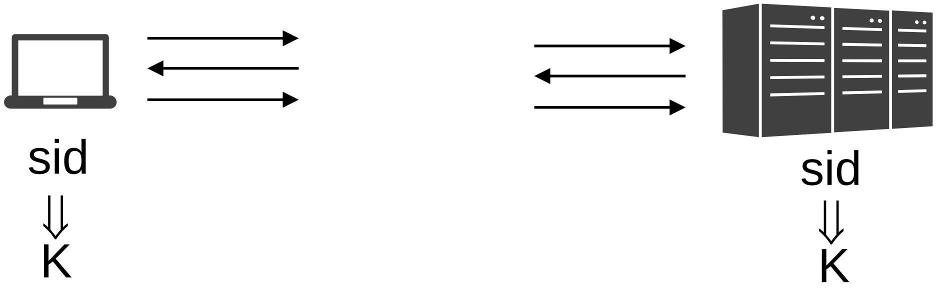
   Prob[three honest parties with same sid] ≈ 0



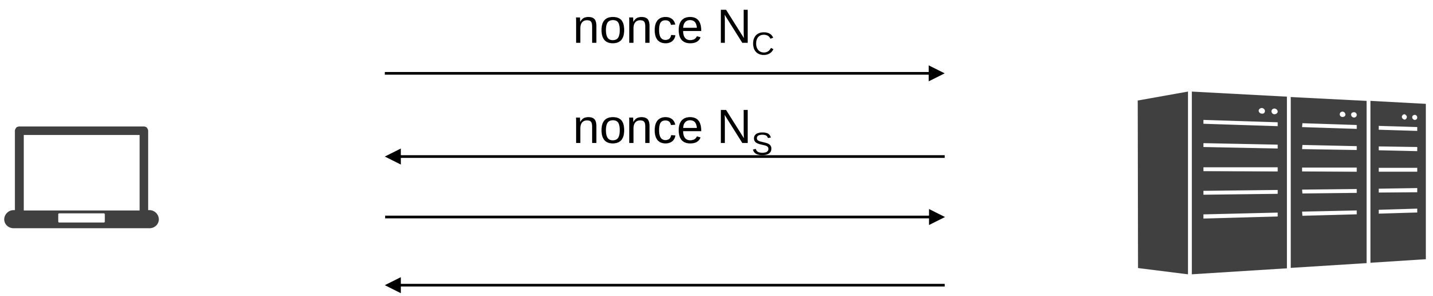
2. Same sid in genuine execution between two honest parties



3. Same sid, same key



Uniqueness is not hard



sid = (N_C, N_S , …)

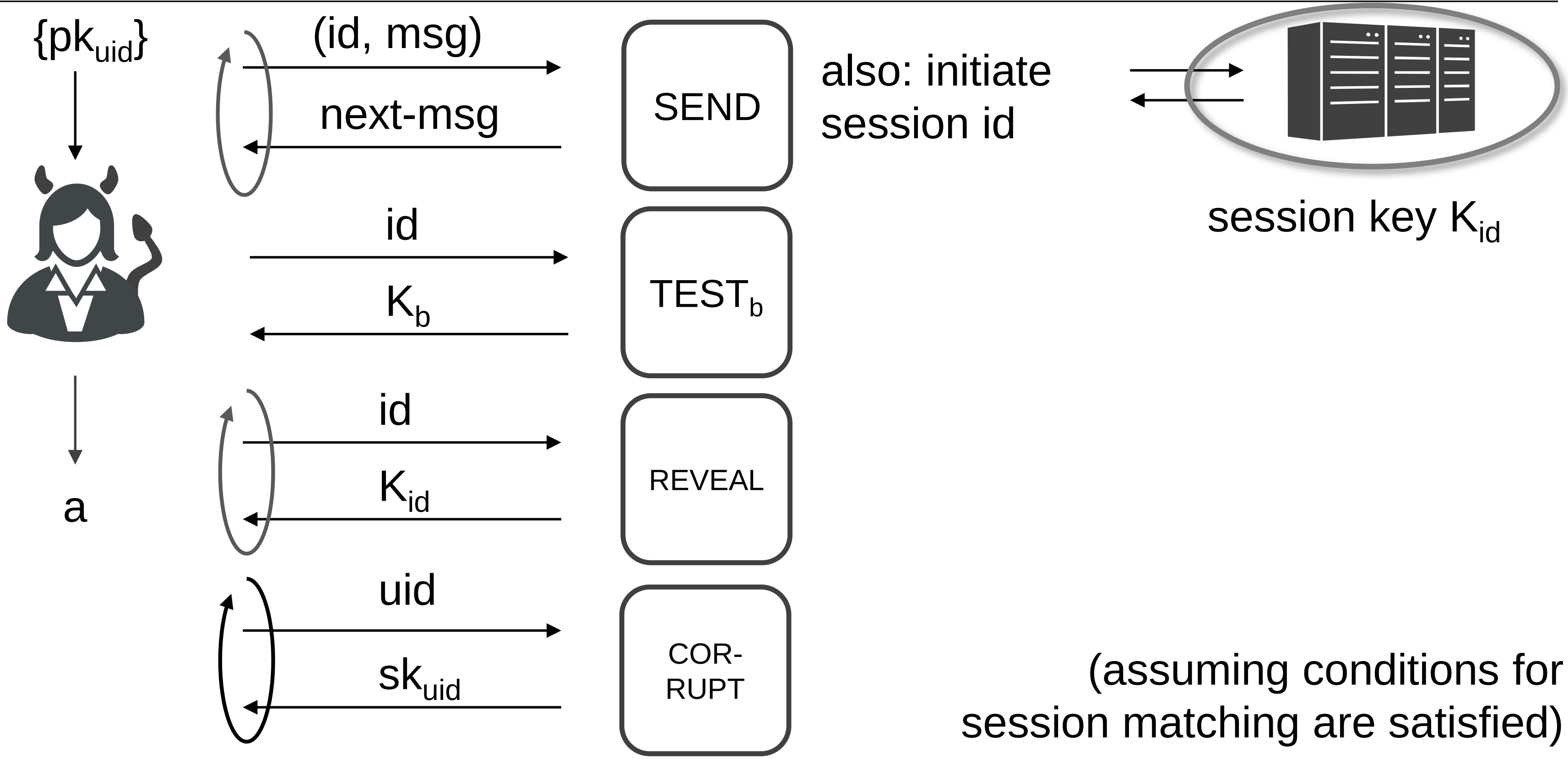
sid = (N_C, N_S , …)

Common example: TLS

Freshness

Mutual Authentication	Unilateral Authentication	Anonymous
neither TEST session nor partner session REVEALED		
neither party in TEST nor intended partner pid CORRUPT	…	…
	+	+
	if unauthenticated partner then there is honest partner session	there is honest partner session

Authenticated Key Exchange



$\{pk_{uid}\}$

a

(id, msg)

next-msg

SEND

also: initiate session id

session key K_{id}

id

K_b

$TEST_b$

id

K_{id}

REVEAL

uid

sk_{uid}

COR-
RUPT

(assuming conditions for session matching are satisfied)

Adversary wins if a=b and freshness condition satisfied

KE is BR-secure against active adversaries if for any efficient adversary: Pr[A wins] ≤ ½ +neg

„Authenticated“?

At most one other party (≤1) holds the session key (and for authenticated cases, if intended partner is honest then it is that party)

Do you see why it cannot be three parties?

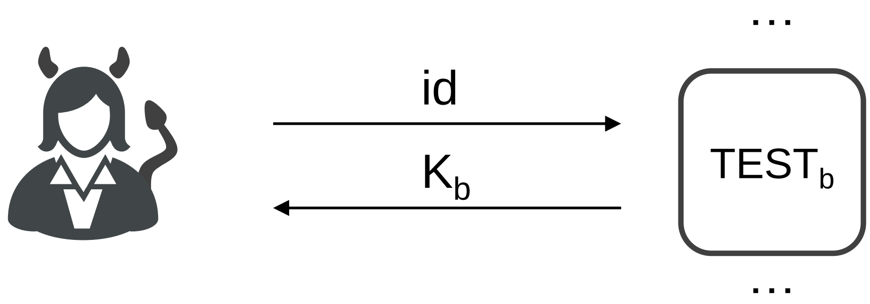
Key confirmation (≥1):

Another party holds the key

see also: Fischlin, Günther, Schmidt, Warinschi: Key Confirmation in Key Exchange…, S&P 2016

Teaser for the Break

We have defined security
for single TEST query:



…

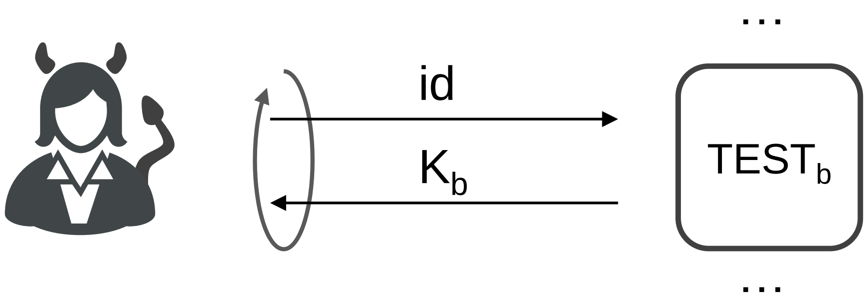
id

K_b

TEST_b

…

Is it equivalent if adversary
has multiple TEST queries?



…

id

K_b

TEST_b

…

Hint: consider first how you need to change the TEST oracle and
then how you could ensure this in a reduction to the single-query case