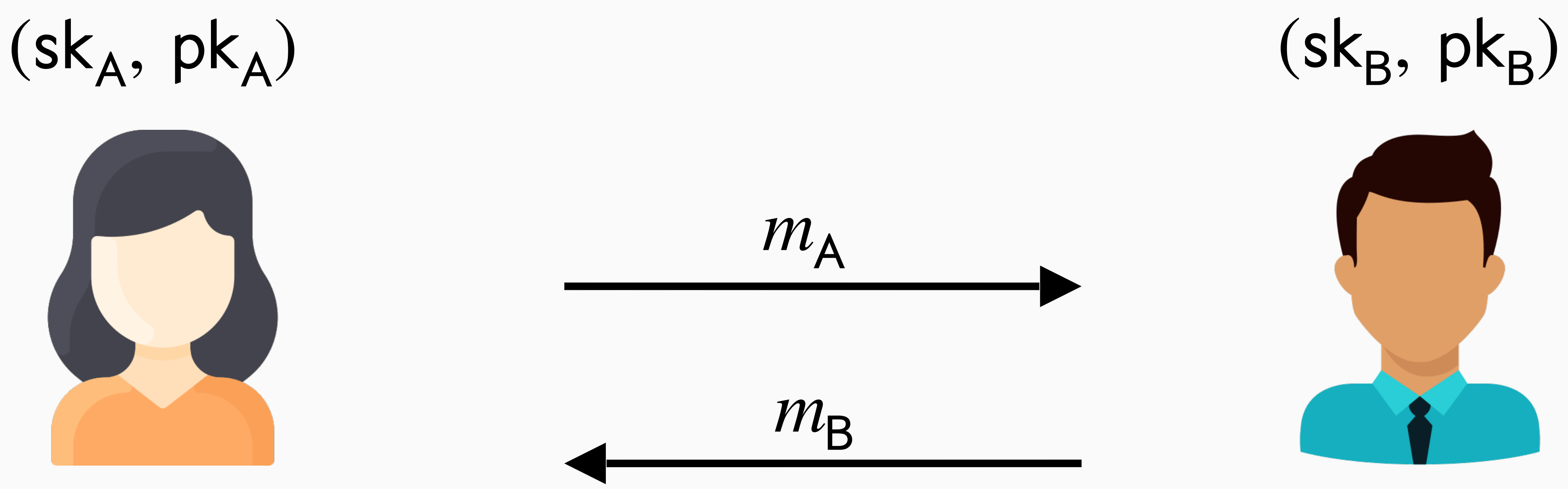
Key Exchange with Tight (Full) Forward Secrecy via Key Confirmation

Jiaxin Pan, Doreen Riepel, Runzhi Zeng

May 30, 2024

Authenticated Key Exchange (AKE)



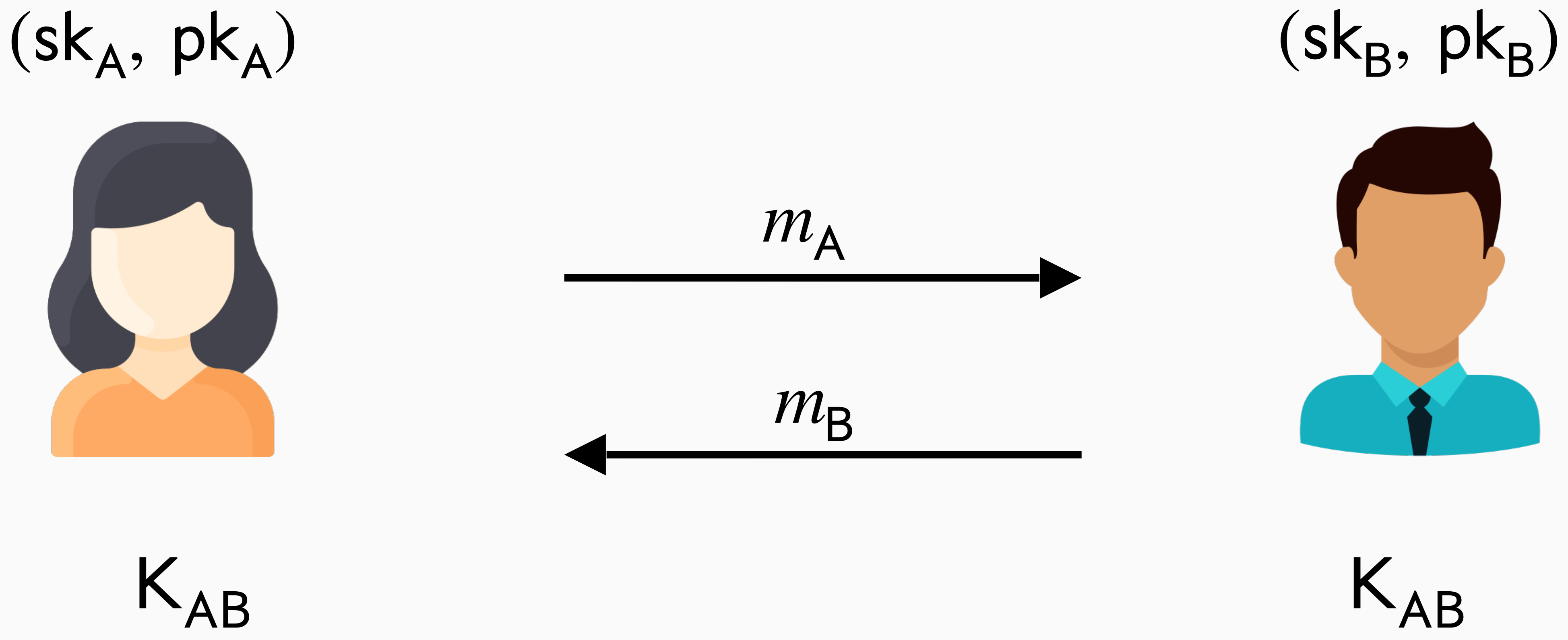
$(\mathsf{sk}_\mathsf{A}, \mathsf{pk}_\mathsf{A})$

$(\mathsf{sk}_\mathsf{B}, \mathsf{pk}_\mathsf{B})$

m_A

m_B

Authenticated Key Exchange (AKE)



$(\mathsf{sk}_\mathsf{A}, \mathsf{pk}_\mathsf{A})$

$(\mathsf{sk}_\mathsf{B}, \mathsf{pk}_\mathsf{B})$

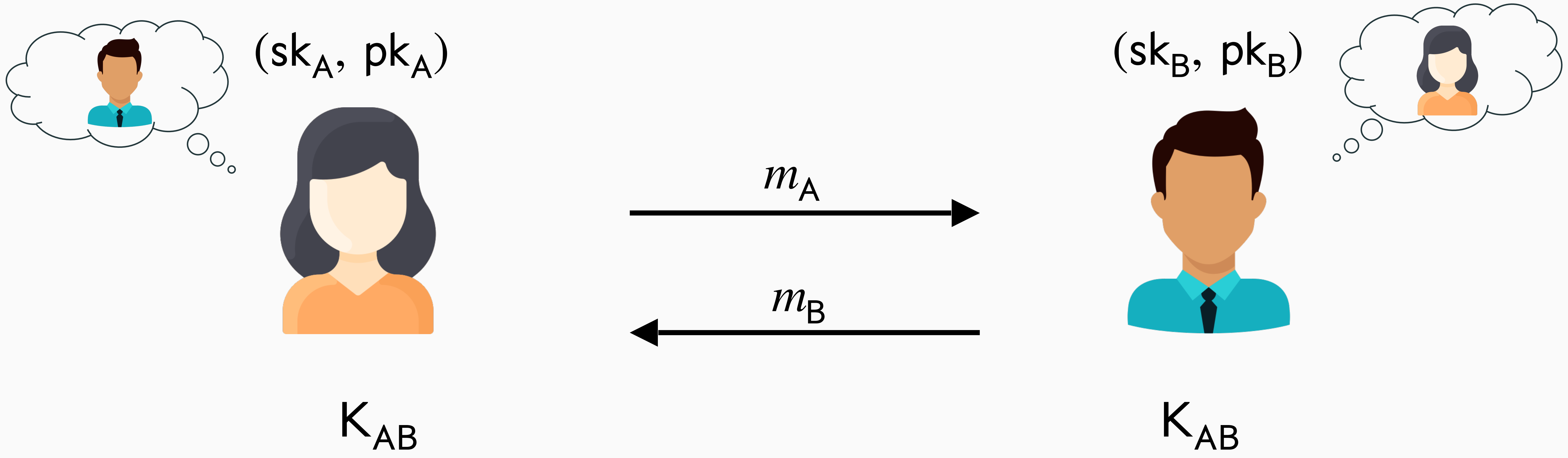
m_A

m_B

K_AB

K_AB

Authenticated Key Exchange (AKE)



$(\mathsf{sk}_A, \mathsf{pk}_A)$

$(\mathsf{sk}_B, \mathsf{pk}_B)$

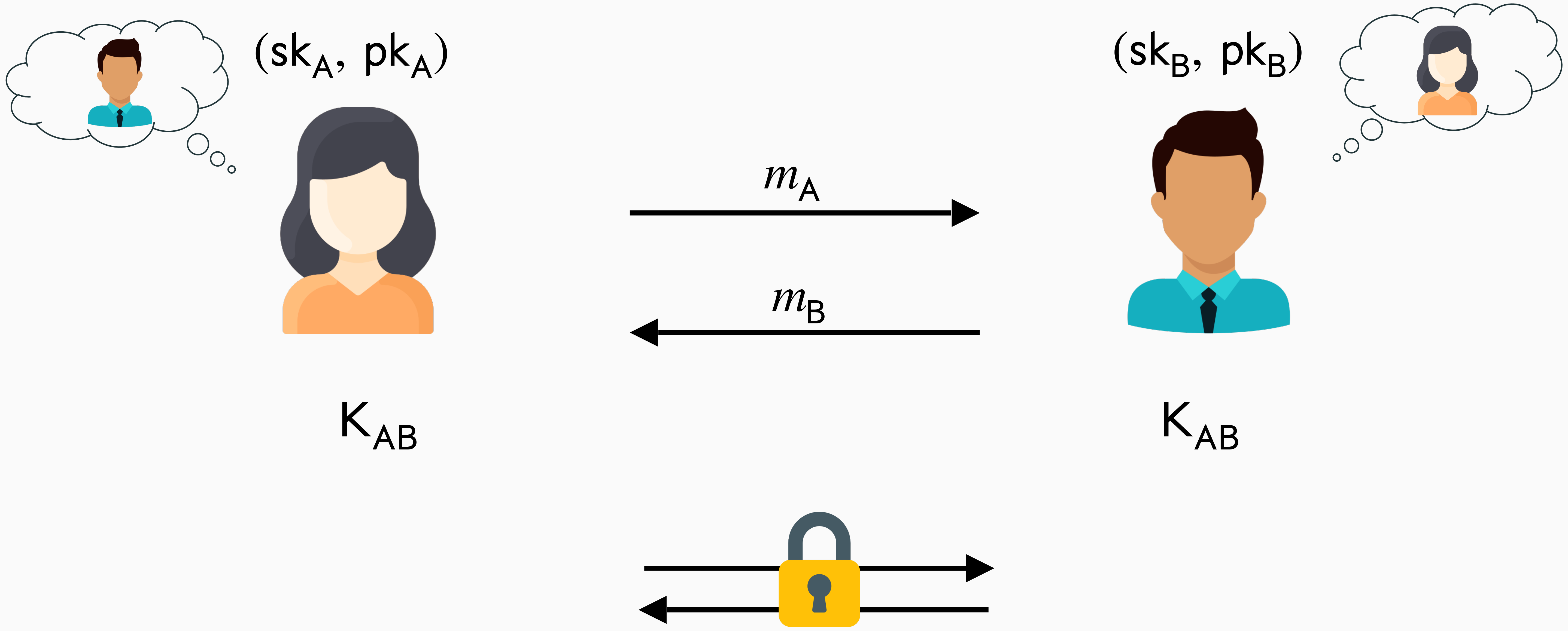
m_A

m_B

K_{AB}

K_{AB}

Authenticated Key Exchange (AKE)



$(\mathsf{sk}_\mathsf{A}, \mathsf{pk}_\mathsf{A})$

$(\mathsf{sk}_\mathsf{B}, \mathsf{pk}_\mathsf{B})$

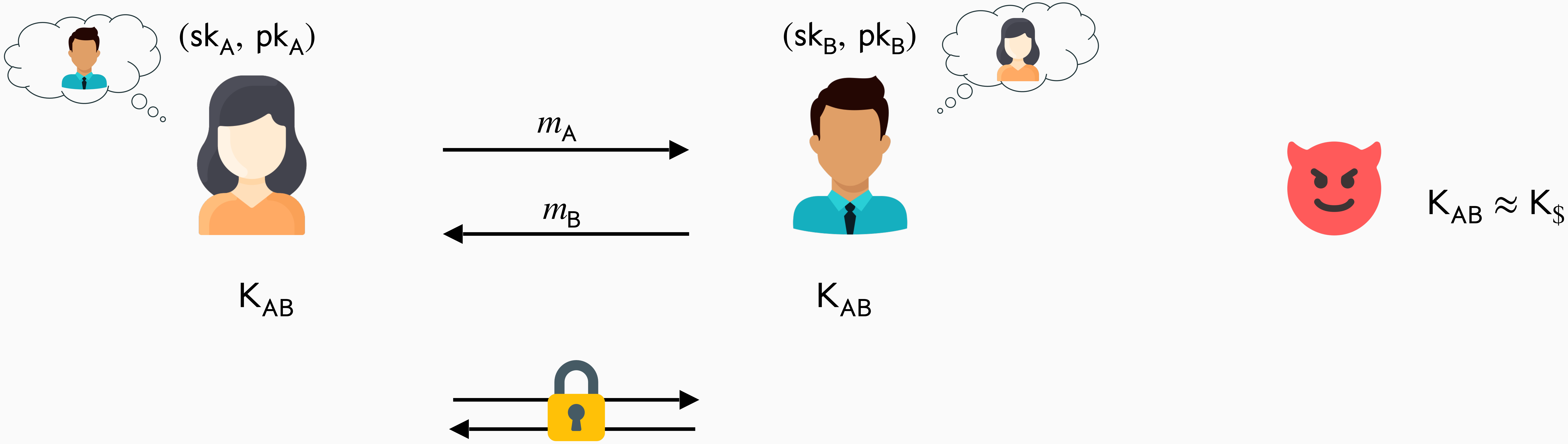
m_A

m_B

K_AB

K_AB

Authenticated Key Exchange (AKE)



$(\mathsf{sk}_\mathsf{A}, \mathsf{pk}_\mathsf{A})$

$(\mathsf{sk}_\mathsf{B}, \mathsf{pk}_\mathsf{B})$

m_A

m_B

K_AB

K_AB

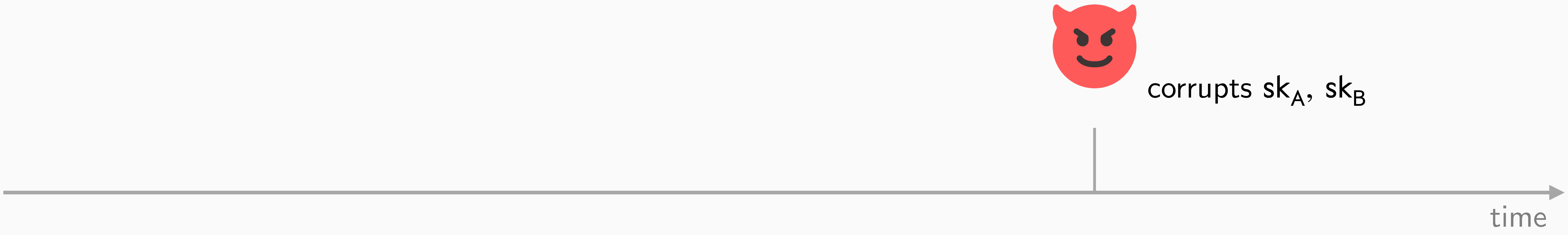
$\mathsf{K}_\mathsf{AB} \approx \mathsf{K}_\$$

Weak vs. Full Forward Secrecy

Forward secrecy: Previous session keys remain secure even when long-term keys are leaked later

Weak vs. Full Forward Secrecy

Forward secrecy: Previous session keys remain secure even when long-term keys are leaked later

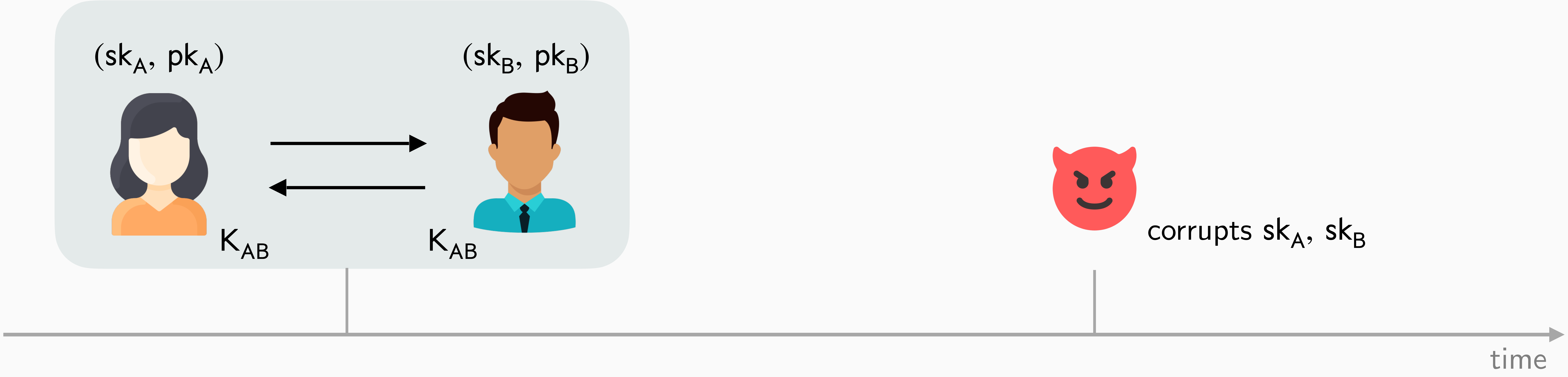


corrupts sk_A, sk_B

time

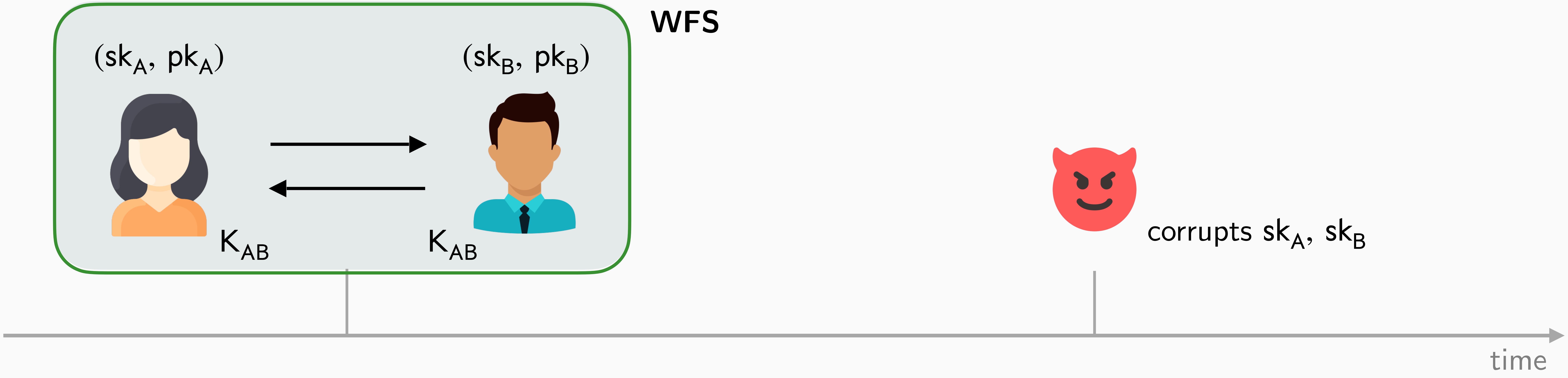
Weak vs. Full Forward Secrecy

Forward secrecy: Previous session keys remain secure even when long-term keys are leaked later



Weak vs. Full Forward Secrecy

Forward secrecy: Previous session keys remain secure even when long-term keys are leaked later



WFS

$(\mathsf{sk}_A, \mathsf{pk}_A)$

$(\mathsf{sk}_B, \mathsf{pk}_B)$

K_{AB}

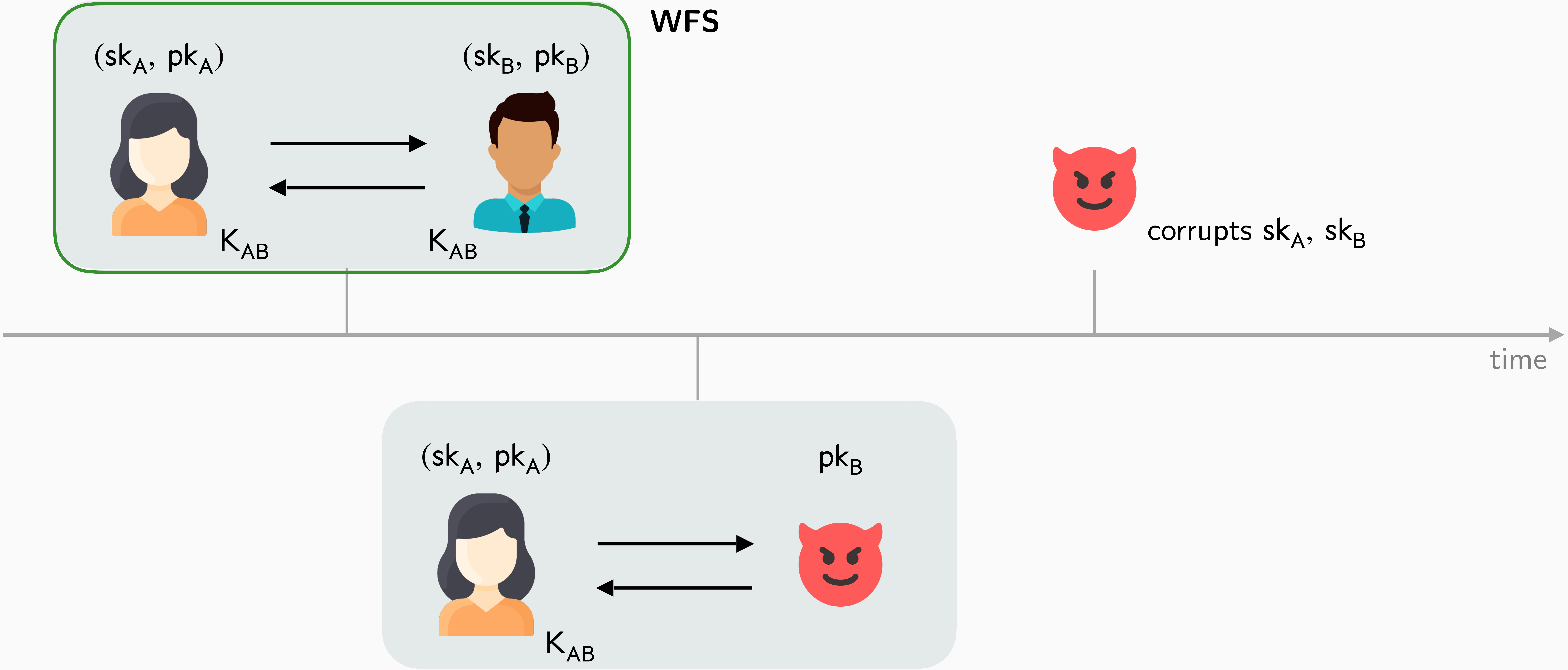
K_{AB}

corrupts sk_A, sk_B

time

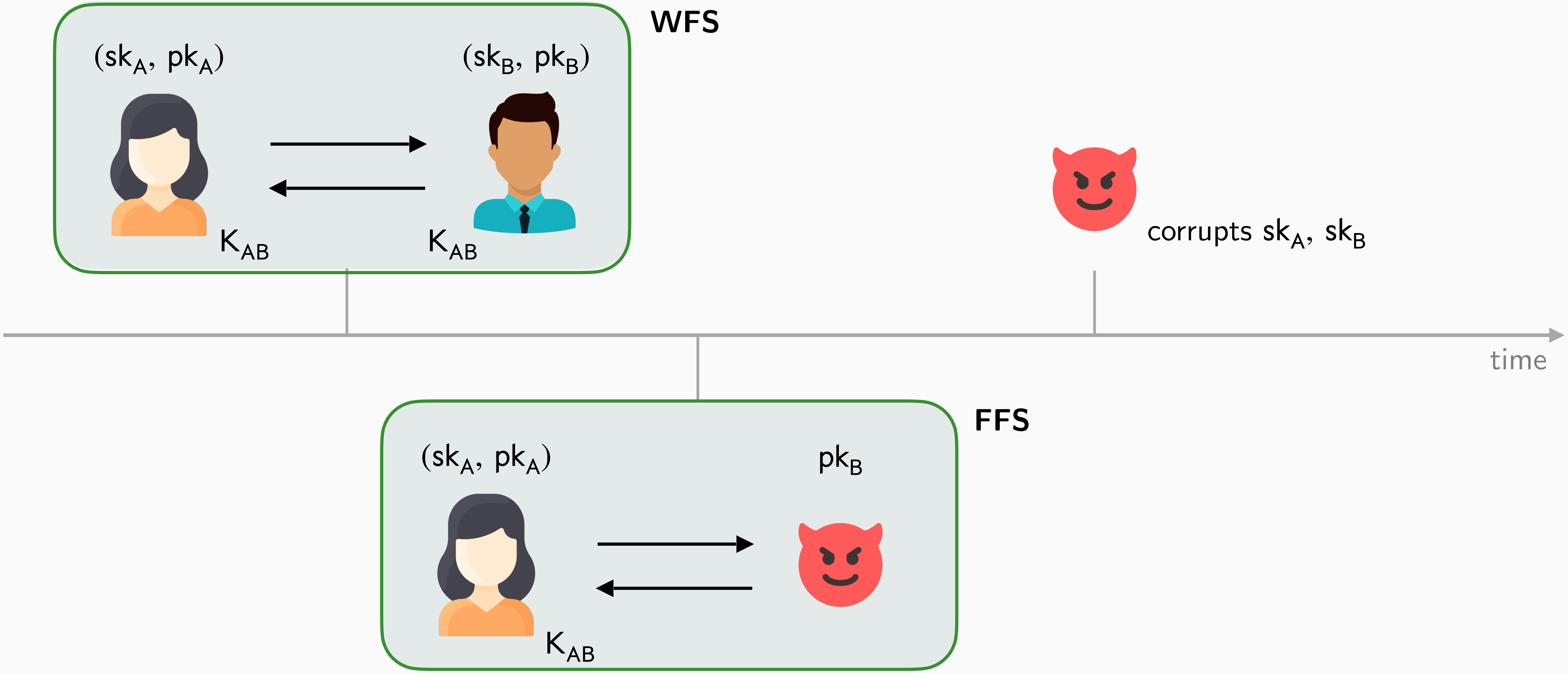
Weak vs. Full Forward Secrecy

Forward secrecy: Previous session keys remain secure even when long-term keys are leaked later



Weak vs. Full Forward Secrecy

Forward secrecy: Previous session keys remain secure even when long-term keys are leaked later

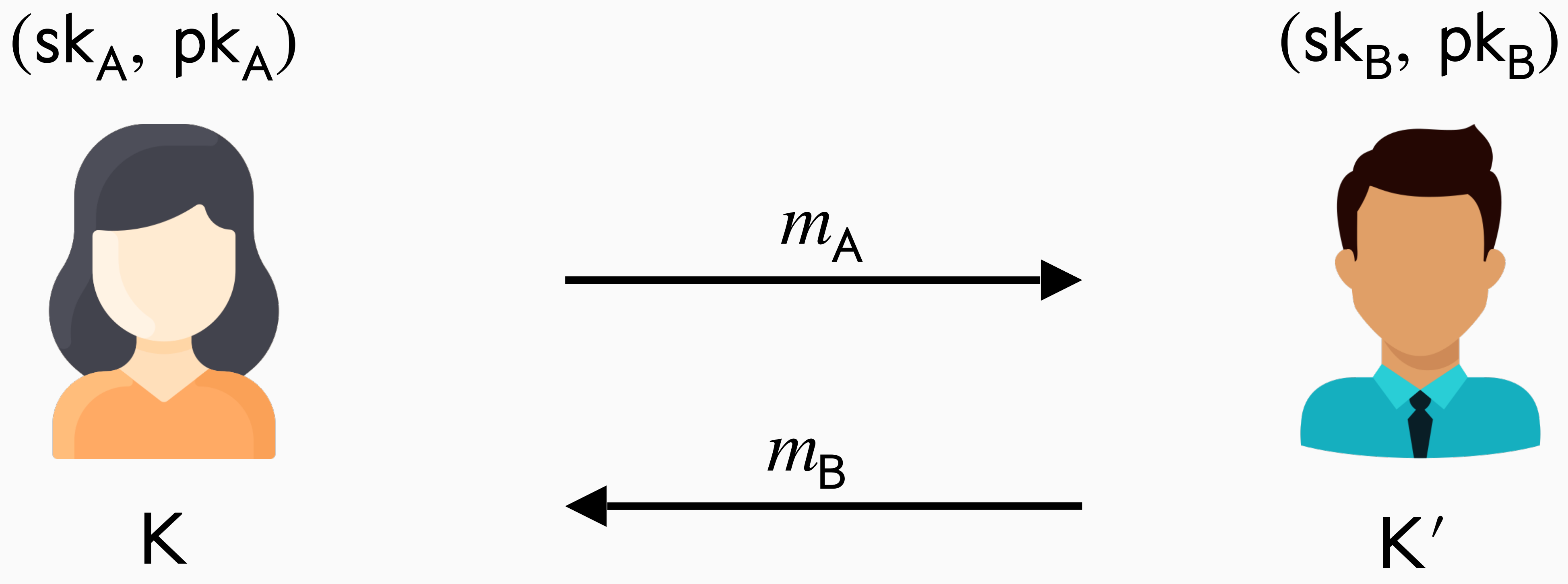


Key Confirmation

WFS-secure AKE protocol $\Pi \Rightarrow$ FFS-secure AKE protocol Π'

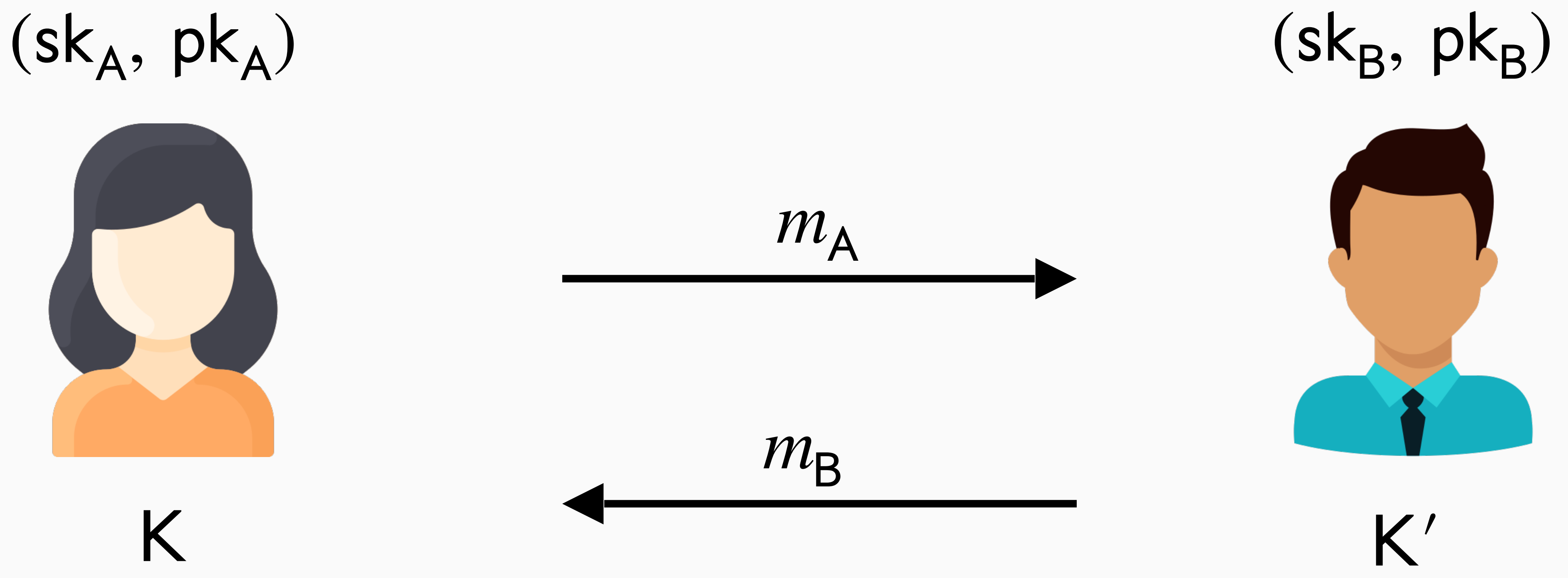
Key Confirmation

WFS-secure AKE protocol $\Pi \Rightarrow$ FFS-secure AKE protocol Π'


$(\mathsf{sk}_A, \mathsf{pk}_A)$
$(\mathsf{sk}_B, \mathsf{pk}_B)$
m_A
m_B
K
K'

Key Confirmation

WFS-secure AKE protocol Π ⇒ FFS-secure AKE protocol Π'



$(\mathsf{sk_A},\ \mathsf{pk_A})$

$(\mathsf{sk_B},\ \mathsf{pk_B})$

m_A

m_B

K

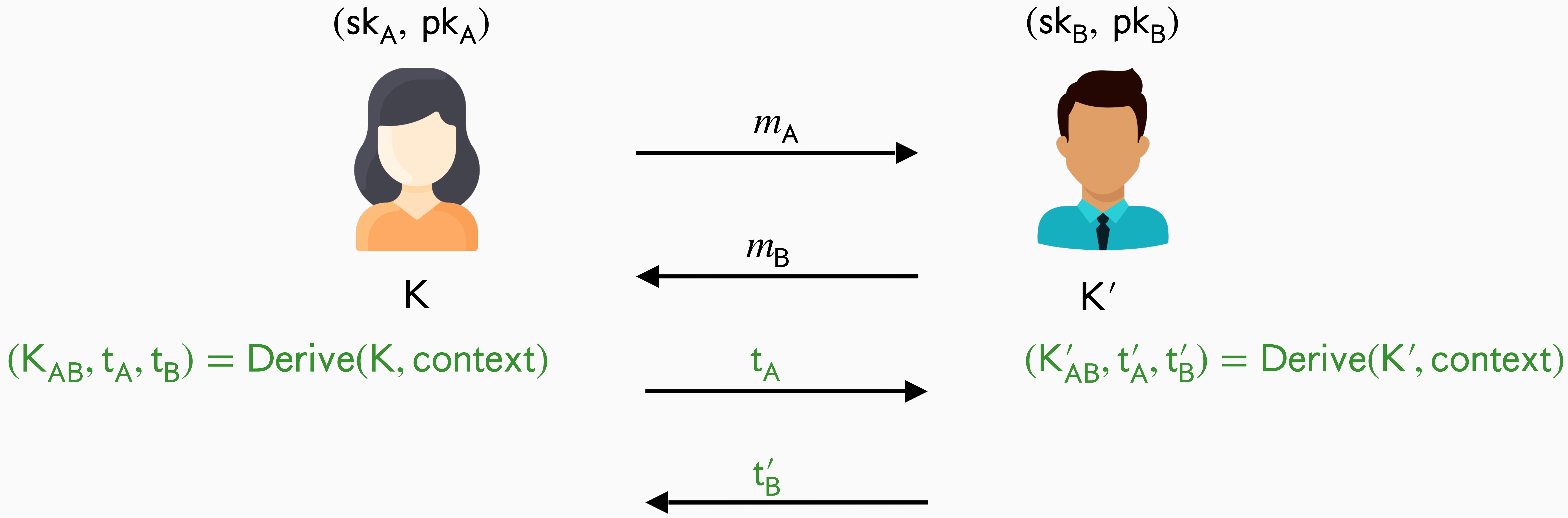
K'

$(\mathsf{K_{AB}}, \mathsf{t_A}, \mathsf{t_B}) = \mathsf{Derive}(\mathsf{K}, \mathsf{context})$

$(\mathsf{K'_{AB}}, \mathsf{t'_A}, \mathsf{t'_B}) = \mathsf{Derive}(\mathsf{K'}, \mathsf{context})$

Key Confirmation

WFS-secure AKE protocol $\Pi \Rightarrow$ FFS-secure AKE protocol Π'



$(\mathsf{sk}_A, \mathsf{pk}_A)$

$(\mathsf{sk}_B, \mathsf{pk}_B)$

m_A

m_B

K

K'

$(\mathsf{K}_{AB}, \mathsf{t}_A, \mathsf{t}_B) = \mathsf{Derive}(\mathsf{K}, \mathsf{context})$

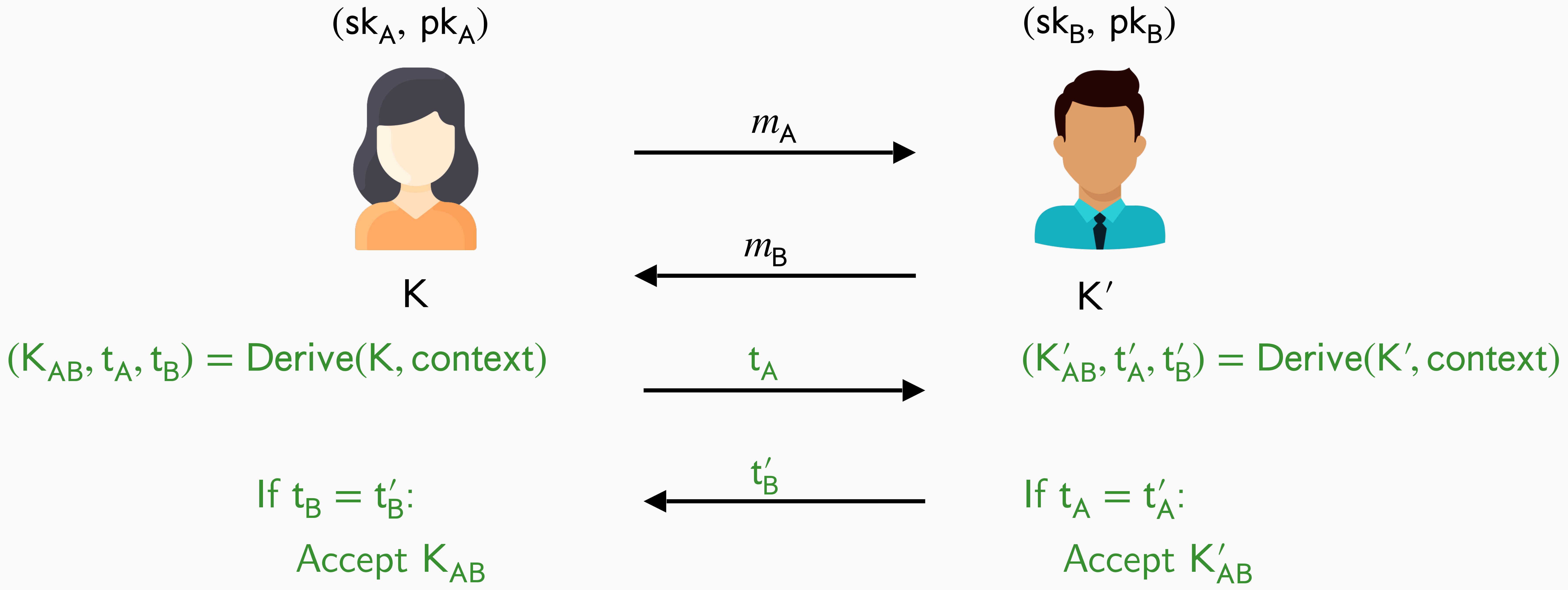
t_A

$(\mathsf{K}'_{AB}, \mathsf{t}'_A, \mathsf{t}'_B) = \mathsf{Derive}(\mathsf{K}', \mathsf{context})$

t'_B

Key Confirmation

WFS-secure AKE protocol $\Pi \Rightarrow$ FFS-secure AKE protocol Π'



$(\mathsf{sk_A}, \mathsf{pk_A})$

$(\mathsf{sk_B}, \mathsf{pk_B})$

m_{A}

m_{B}

K

K'

$(\mathsf{K_{AB}}, \mathsf{t_A}, \mathsf{t_B}) = \mathsf{Derive}(\mathsf{K}, \mathsf{context})$

$\mathsf{t_A}$

$(\mathsf{K'_{AB}}, \mathsf{t'_A}, \mathsf{t'_B}) = \mathsf{Derive}(\mathsf{K}', \mathsf{context})$

$\mathsf{t'_B}$

If $\mathsf{t_B} = \mathsf{t'_B}$:

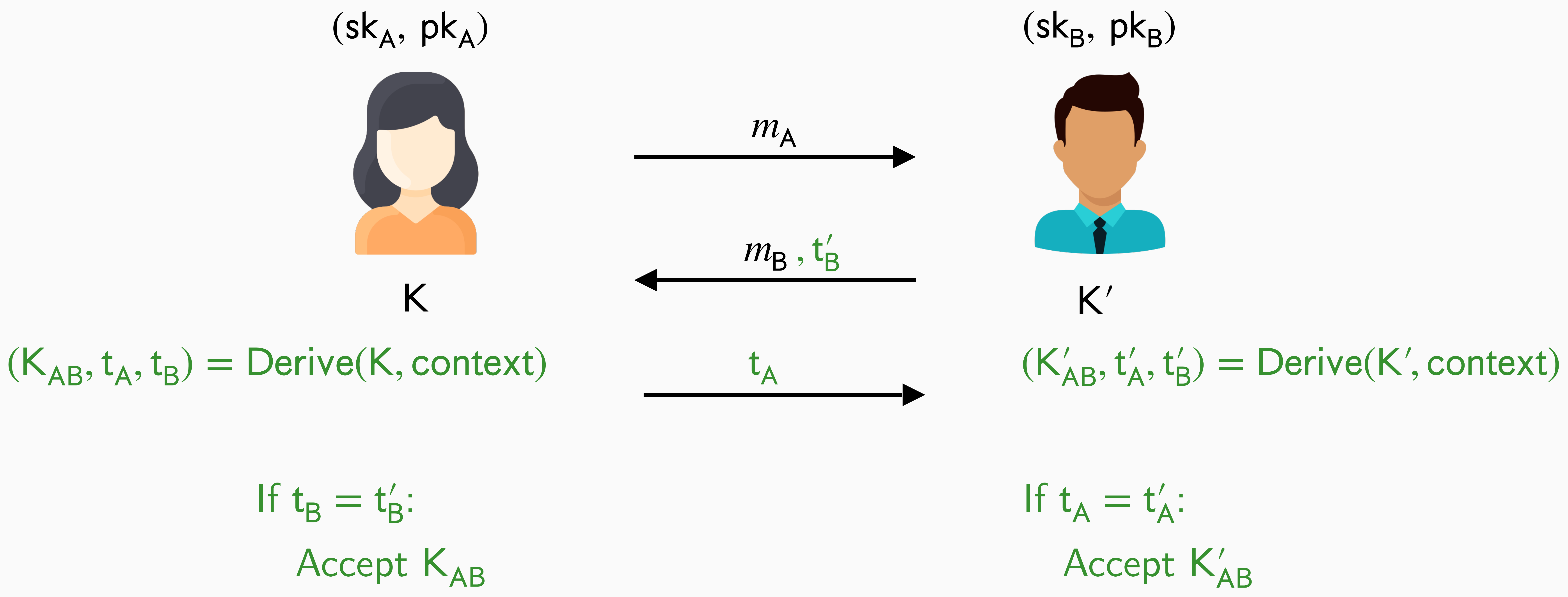
Accept $\mathsf{K_{AB}}$

If $\mathsf{t_A} = \mathsf{t'_A}$:

Accept $\mathsf{K'_{AB}}$

Key Confirmation

WFS-secure AKE protocol Π $\Rightarrow$ FFS-secure AKE protocol Π'



$(\mathsf{sk}_A, \mathsf{pk}_A)$

$(\mathsf{sk}_B, \mathsf{pk}_B)$

m_A

m_B , t'_B

K

K'

$(K_{AB}, t_A, t_B) = \mathsf{Derive}(K, \mathsf{context})$

t_A

$(K'_{AB}, t'_A, t'_B) = \mathsf{Derive}(K', \mathsf{context})$

If $t_B = t'_B$:

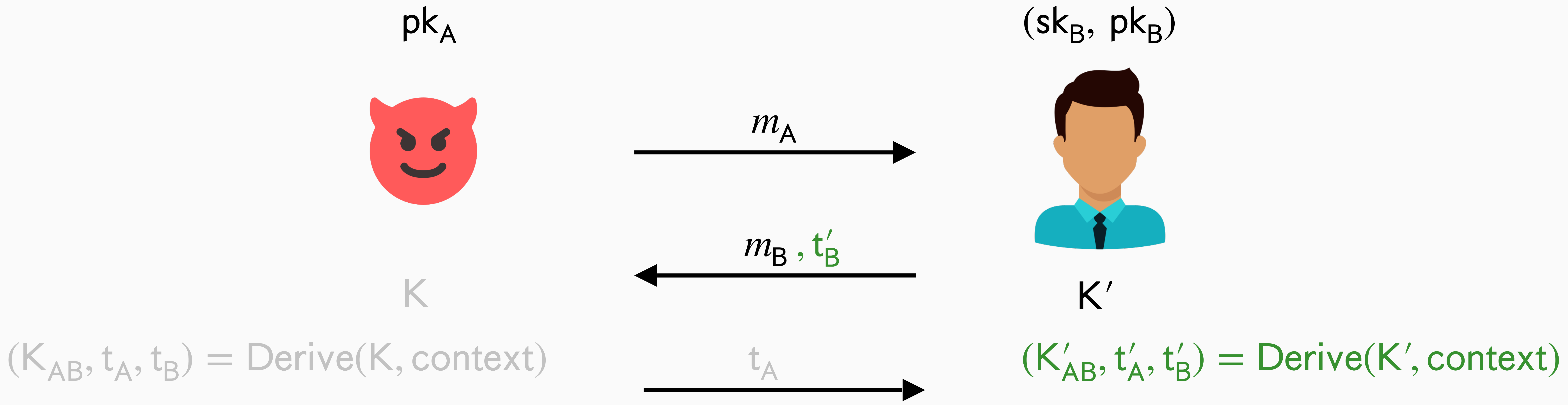
Accept K_{AB}

If $t_A = t'_A$:

Accept K'_{AB}

Key Confirmation

WFS-secure AKE protocol $\Pi \Rightarrow$ FFS-secure AKE protocol Π'



pk_A

$(\mathsf{sk}_B,\ \mathsf{pk}_B)$

m_A

m_B , t'_B

K

K'

$(K_{AB}, t_A, t_B) = \mathsf{Derive}(K, \mathsf{context})$

t_A

$(K'_{AB}, t'_A, t'_B) = \mathsf{Derive}(K', \mathsf{context})$

If $t_A = t'_A$:

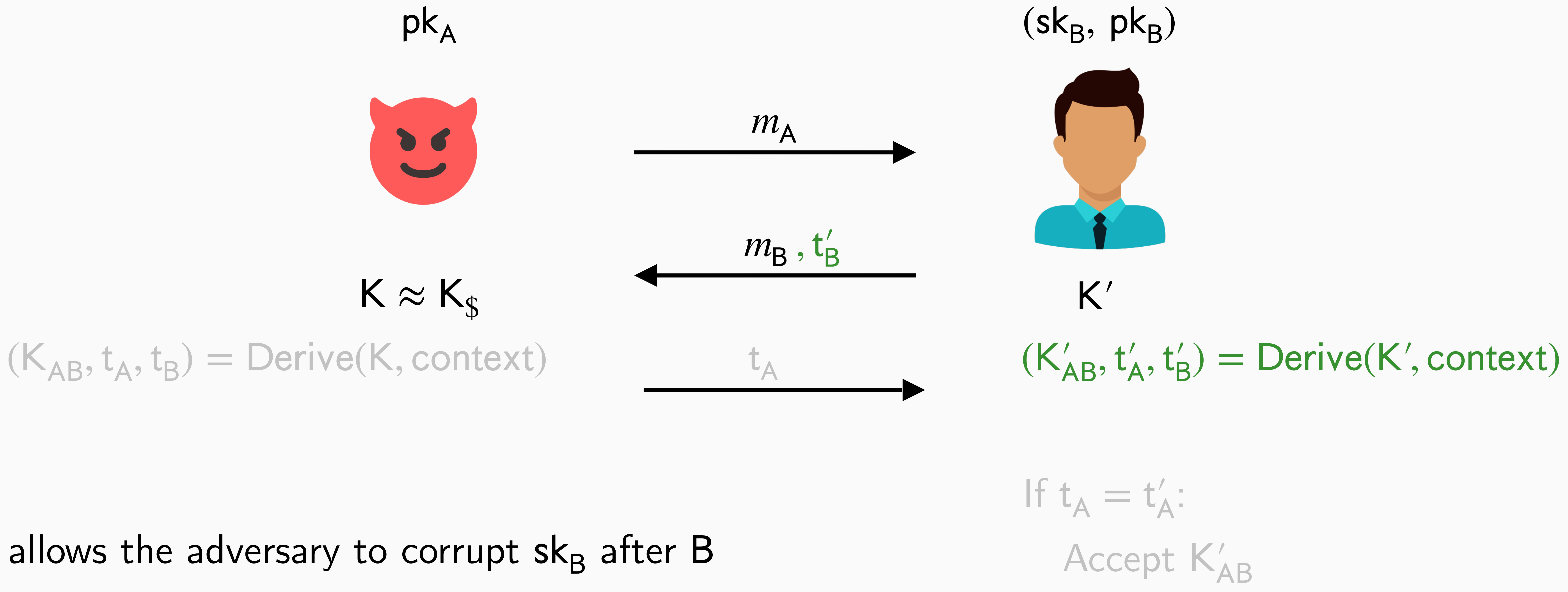
Accept K'_{AB}

Intuition

- FFS only allows the adversary to corrupt sk_B after B accepts the session key
- If the adversary cannot forge t_A, B will never accept

Key Confirmation

WFS-secure AKE protocol $\Pi \Rightarrow$ FFS-secure AKE protocol Π'



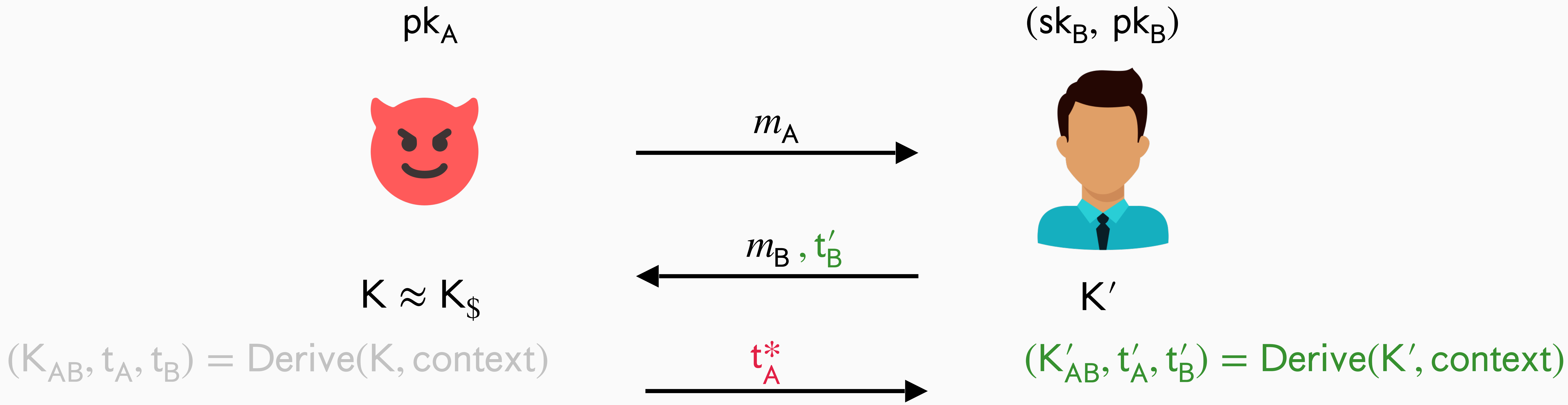
pk_A
$(\mathsf{sk}_B, \mathsf{pk}_B)$
m_A
m_B , t'_B
$K \approx K_\$$
K'
$(K_{AB}, t_A, t_B) = \mathsf{Derive}(K, \mathsf{context})$
t_A
$(K'_{AB}, t'_A, t'_B) = \mathsf{Derive}(K', \mathsf{context})$
If $t_A = t'_A$:
Accept K'_{AB}

Intuition

- FFS only allows the adversary to corrupt sk_B after B accepts the session key
- If the adversary cannot forge t_A, B will never accept

Key Confirmation

WFS-secure AKE protocol Π ⇒ FFS-secure AKE protocol Π'



pk_A

$(\mathsf{sk}_\mathsf{B}, \mathsf{pk}_\mathsf{B})$

m_A

m_B , t'_B

$\mathsf{K} \approx \mathsf{K}_\$$

K'

$(\mathsf{K}_{\mathsf{AB}}, \mathsf{t}_\mathsf{A}, \mathsf{t}_\mathsf{B}) = \mathsf{Derive}(\mathsf{K}, \mathsf{context})$

t^*_A

$(\mathsf{K}'_{\mathsf{AB}}, \mathsf{t}'_\mathsf{A}, \mathsf{t}'_\mathsf{B}) = \mathsf{Derive}(\mathsf{K}', \mathsf{context})$

If $\mathsf{t}_\mathsf{A} = \mathsf{t}'_\mathsf{A}$:

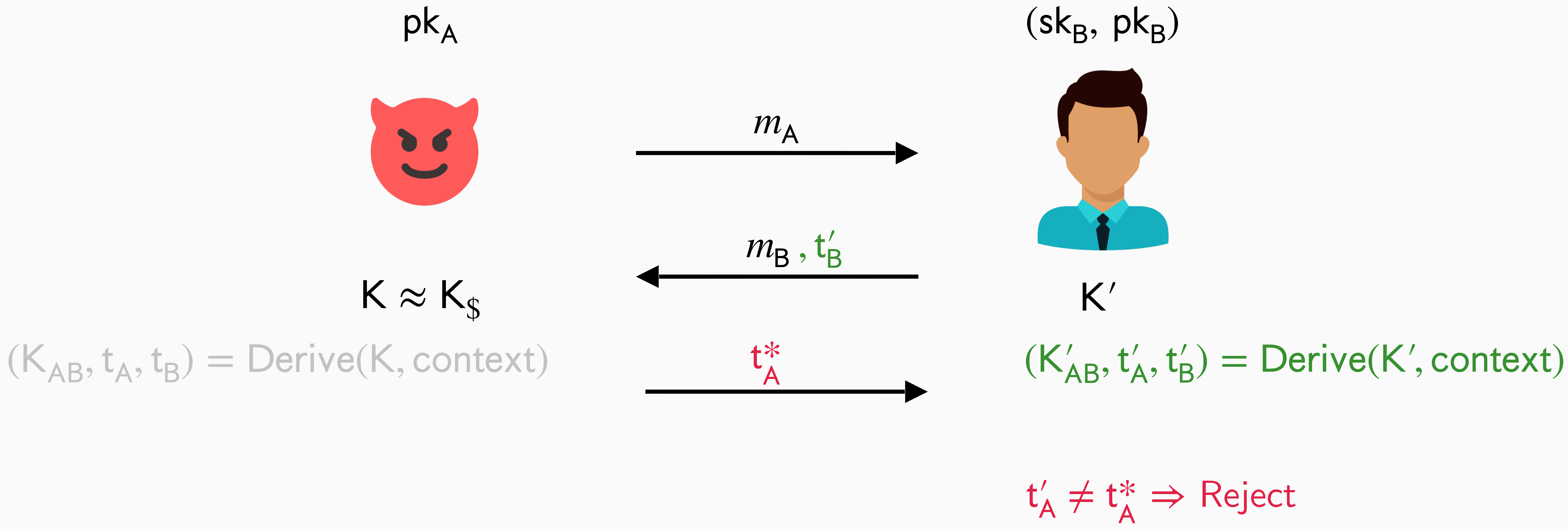
Accept $\mathsf{K}'_{\mathsf{AB}}$

Intuition

- FFS only allows the adversary to corrupt sk_B after B accepts the session key
- If the adversary cannot forge t_A, B will never accept

Key Confirmation

WFS-secure AKE protocol $\Pi \Rightarrow$ FFS-secure AKE protocol Π'



pk_A

$(\mathsf{sk}_B, \mathsf{pk}_B)$

m_A

m_B , t'_B

$\mathsf{K} \approx \mathsf{K}_\$$

K'

$(\mathsf{K}_{AB}, \mathsf{t}_A, \mathsf{t}_B) = \mathsf{Derive}(\mathsf{K}, \mathsf{context})$

t^*_A

$(\mathsf{K}'_{AB}, \mathsf{t}'_A, \mathsf{t}'_B) = \mathsf{Derive}(\mathsf{K}', \mathsf{context})$

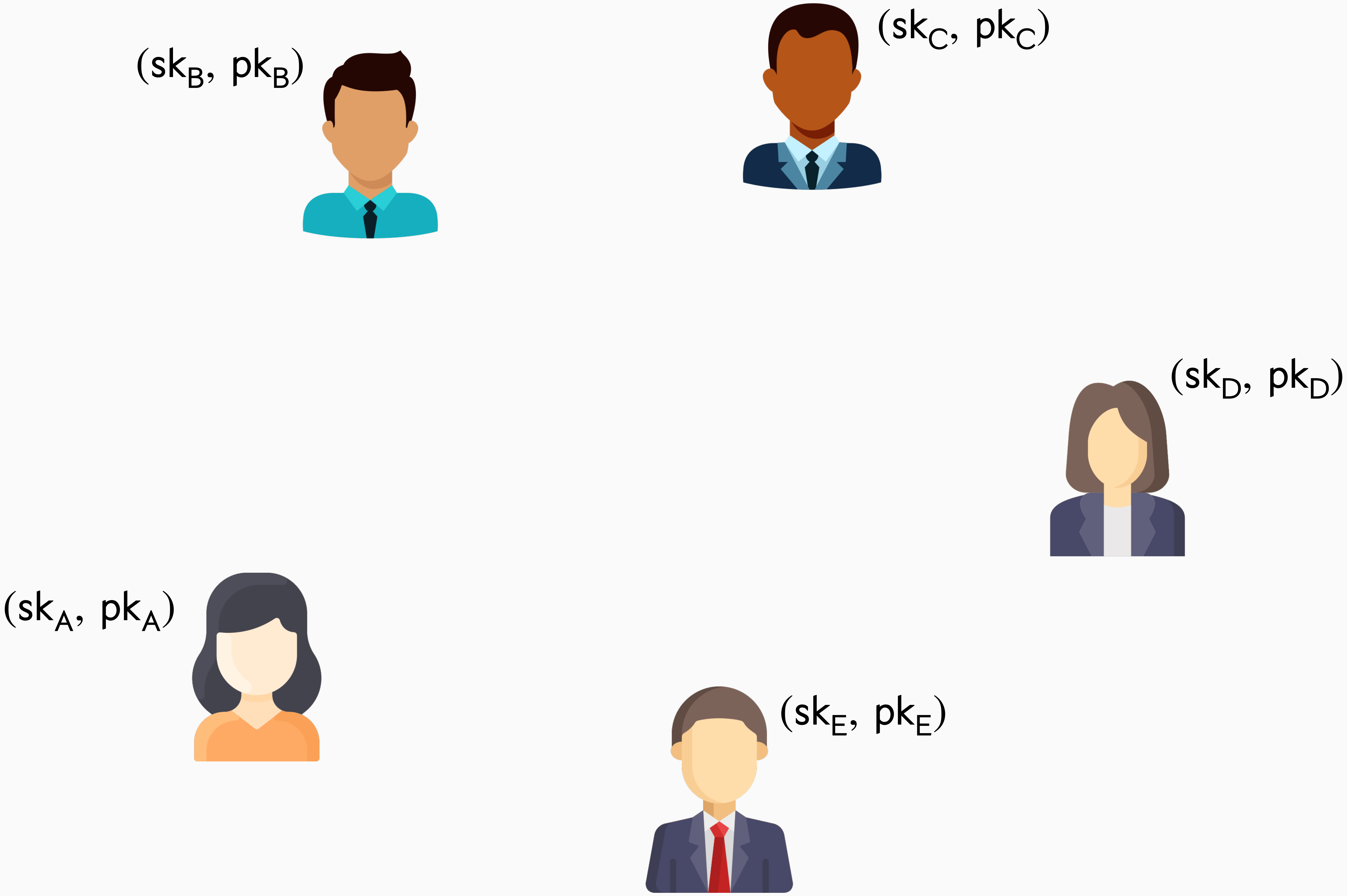
$\mathsf{t}'_A \neq \mathsf{t}^*_A \Rightarrow$ Reject

Intuition

- FFS only allows the adversary to corrupt sk_B after B accepts the session key
- If the adversary cannot forge t_A, B will never accept

Modeling Security

Execution Environment: N users, S sessions



$(\mathsf{sk}_\mathsf{B}, \mathsf{pk}_\mathsf{B})$

$(\mathsf{sk}_\mathsf{C}, \mathsf{pk}_\mathsf{C})$

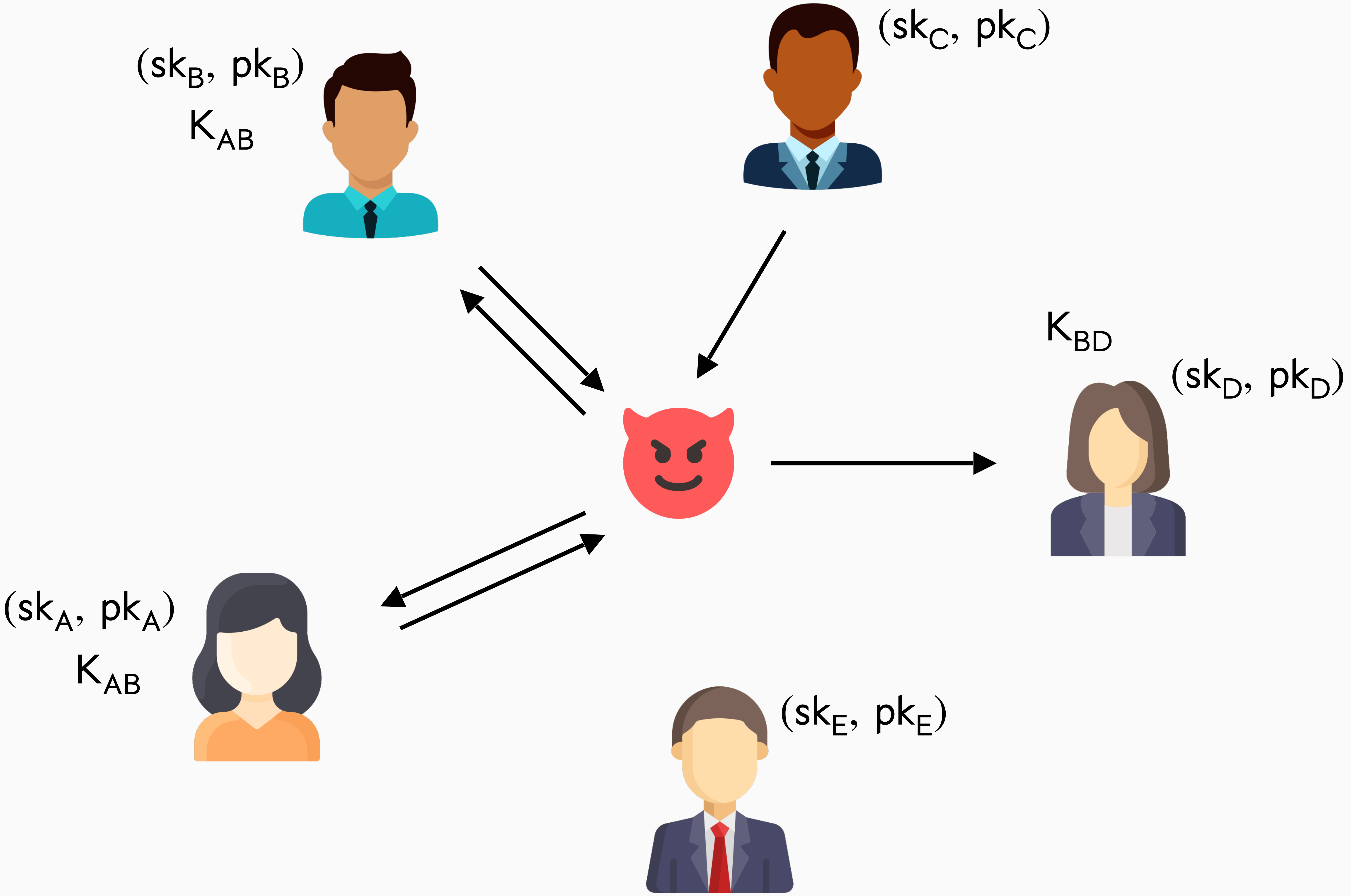
$(\mathsf{sk}_\mathsf{D}, \mathsf{pk}_\mathsf{D})$

$(\mathsf{sk}_\mathsf{A}, \mathsf{pk}_\mathsf{A})$

$(\mathsf{sk}_\mathsf{E}, \mathsf{pk}_\mathsf{E})$

Modeling Security

Execution Environment: N users, S sessions



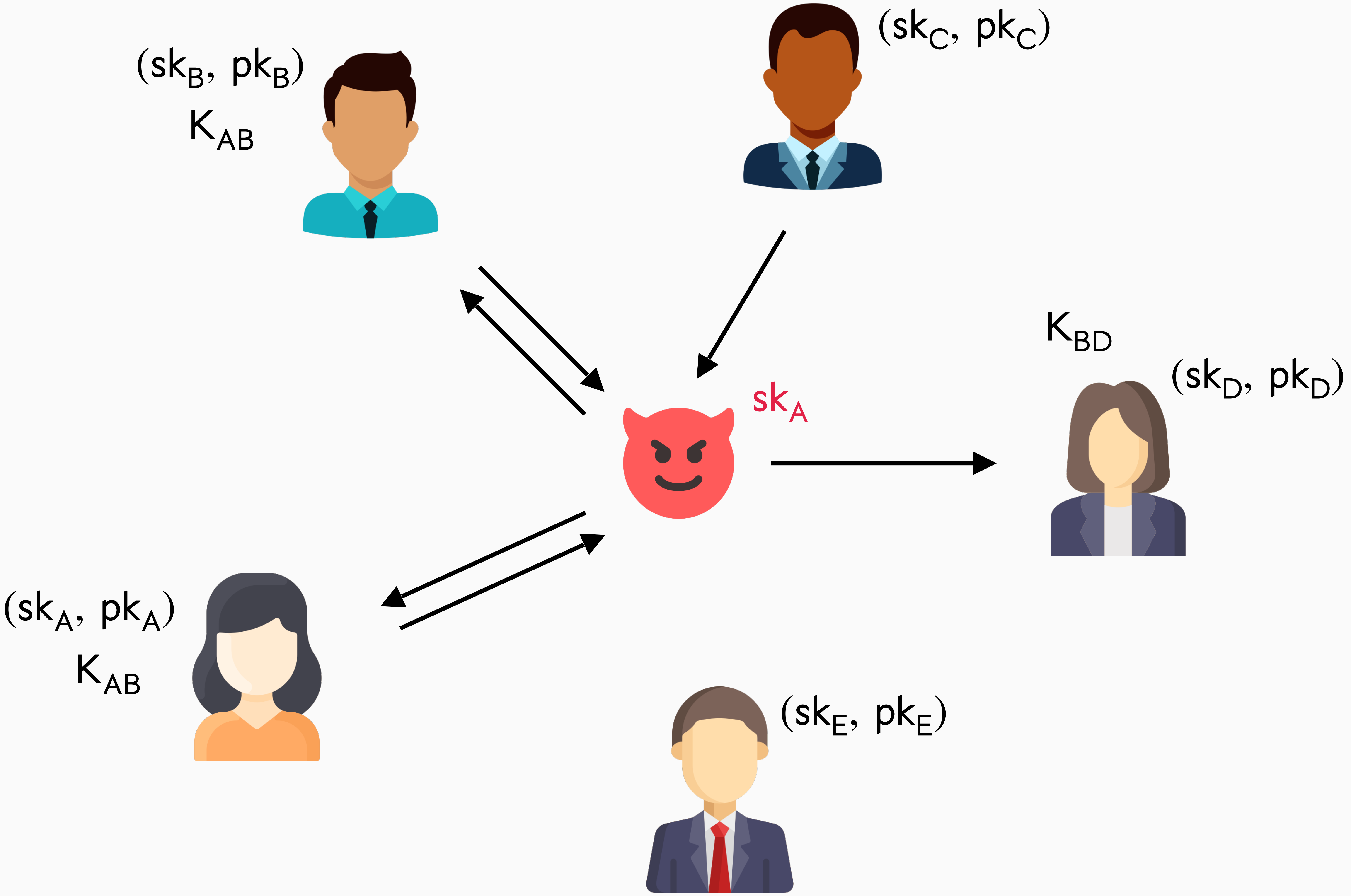
$(\mathsf{sk_B}, \mathsf{pk_B})$
$\mathsf{K_{AB}}$
$(\mathsf{sk_C}, \mathsf{pk_C})$
$\mathsf{K_{BD}}$
$(\mathsf{sk_D}, \mathsf{pk_D})$
$(\mathsf{sk_A}, \mathsf{pk_A})$
$\mathsf{K_{AB}}$
$(\mathsf{sk_E}, \mathsf{pk_E})$

Adversary

- Controls the network

Modeling Security

Execution Environment: N users, S sessions

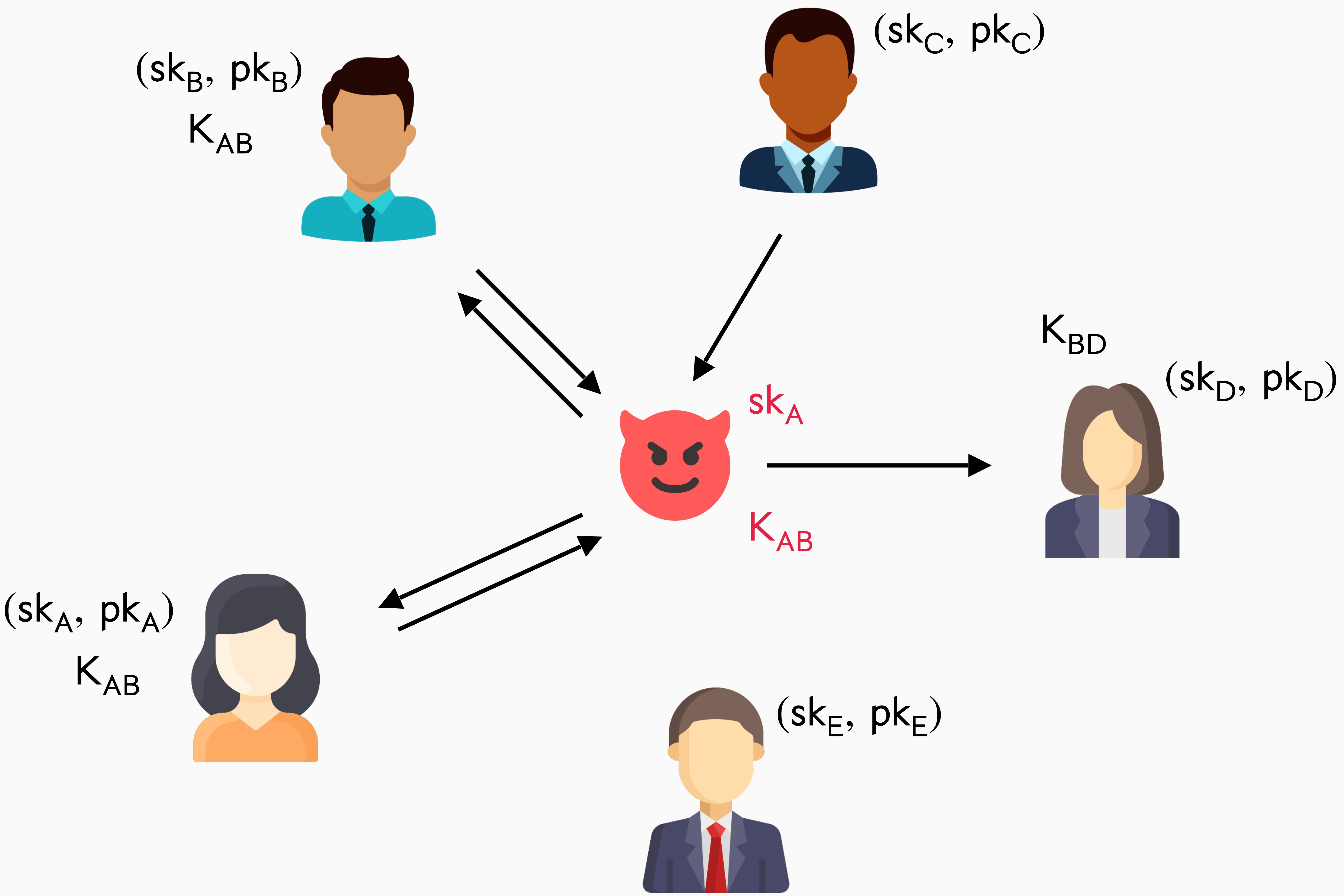


Adversary

- Controls the network
- Corrupts parties

Modeling Security

Execution Environment: N users, S sessions

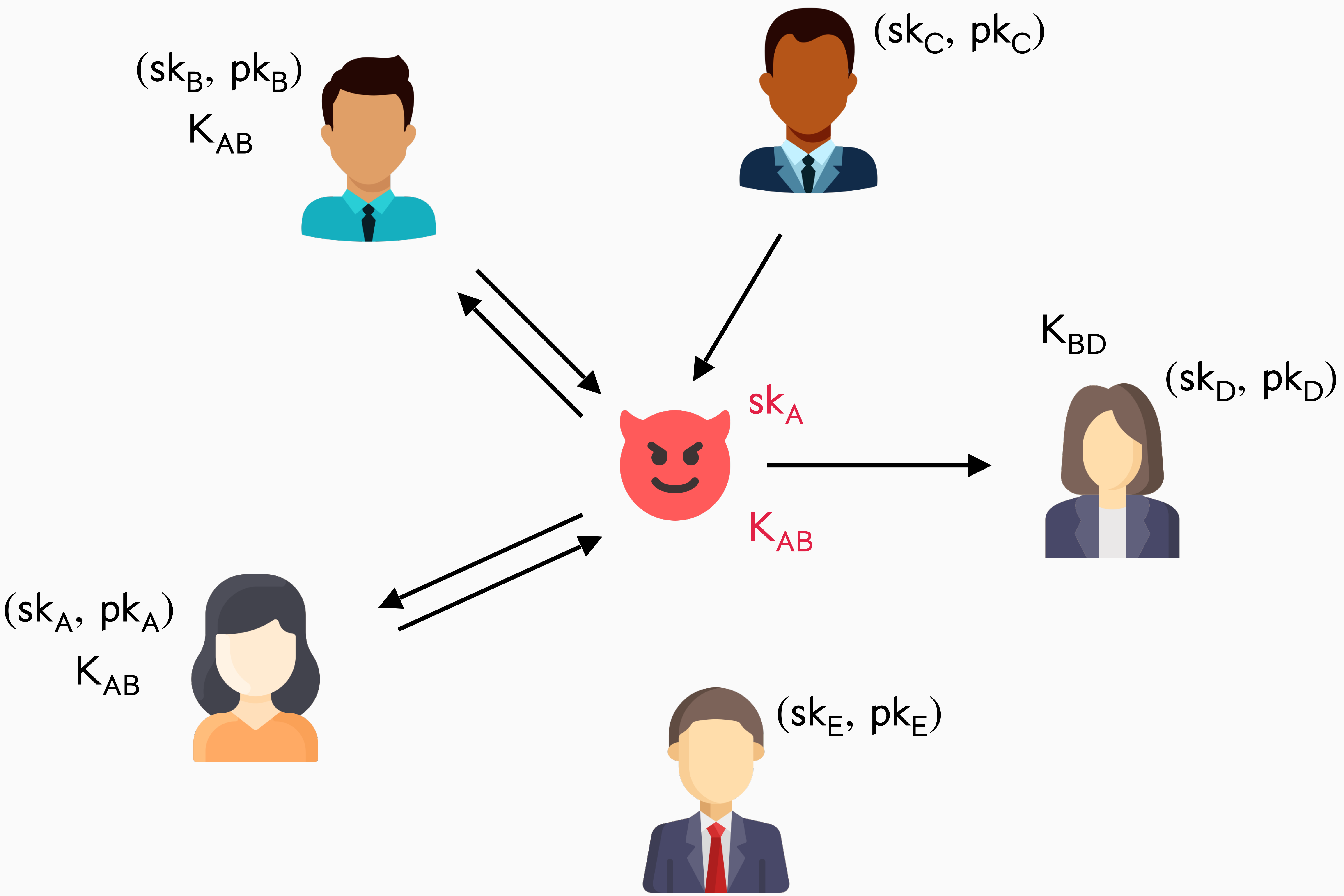


Adversary

- Controls the network
- Corrupts parties
- Reveals session keys

Modeling Security

Execution Environment: N users, S sessions



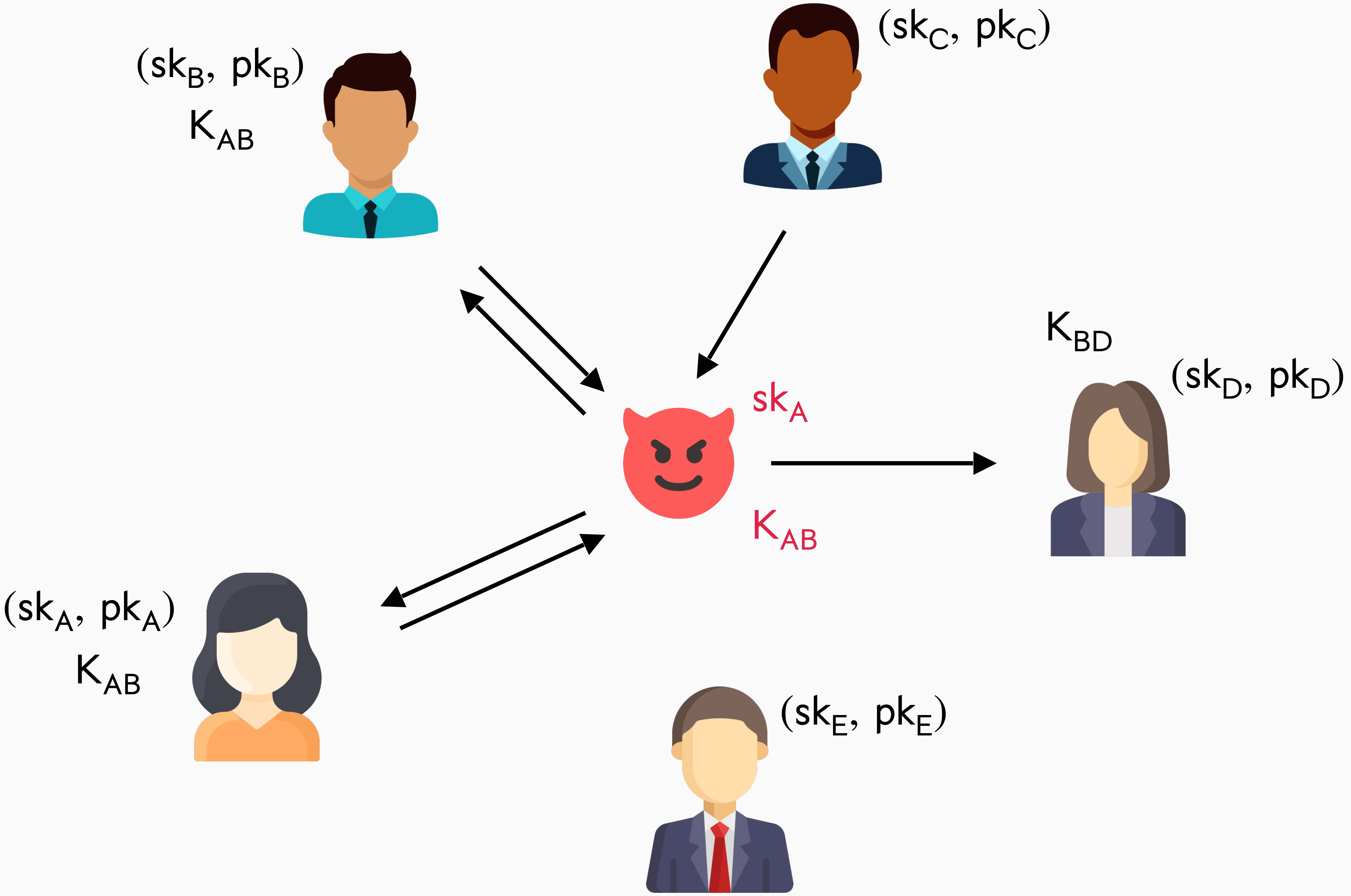
Adversary

- Controls the network
- Corrupts parties
- Reveals session keys
- Test (challenge) session keys

$$K_{BD} \approx K_{\$}$$

Modeling Security

Execution Environment: N users, S sessions



$(\mathsf{sk}_B, \mathsf{pk}_B)$
K_{AB}
$(\mathsf{sk}_C, \mathsf{pk}_C)$
sk_A
K_{AB}
K_{BD}
$(\mathsf{sk}_D, \mathsf{pk}_D)$
$(\mathsf{sk}_A, \mathsf{pk}_A)$
K_{AB}
$(\mathsf{sk}_E, \mathsf{pk}_E)$

Adversary

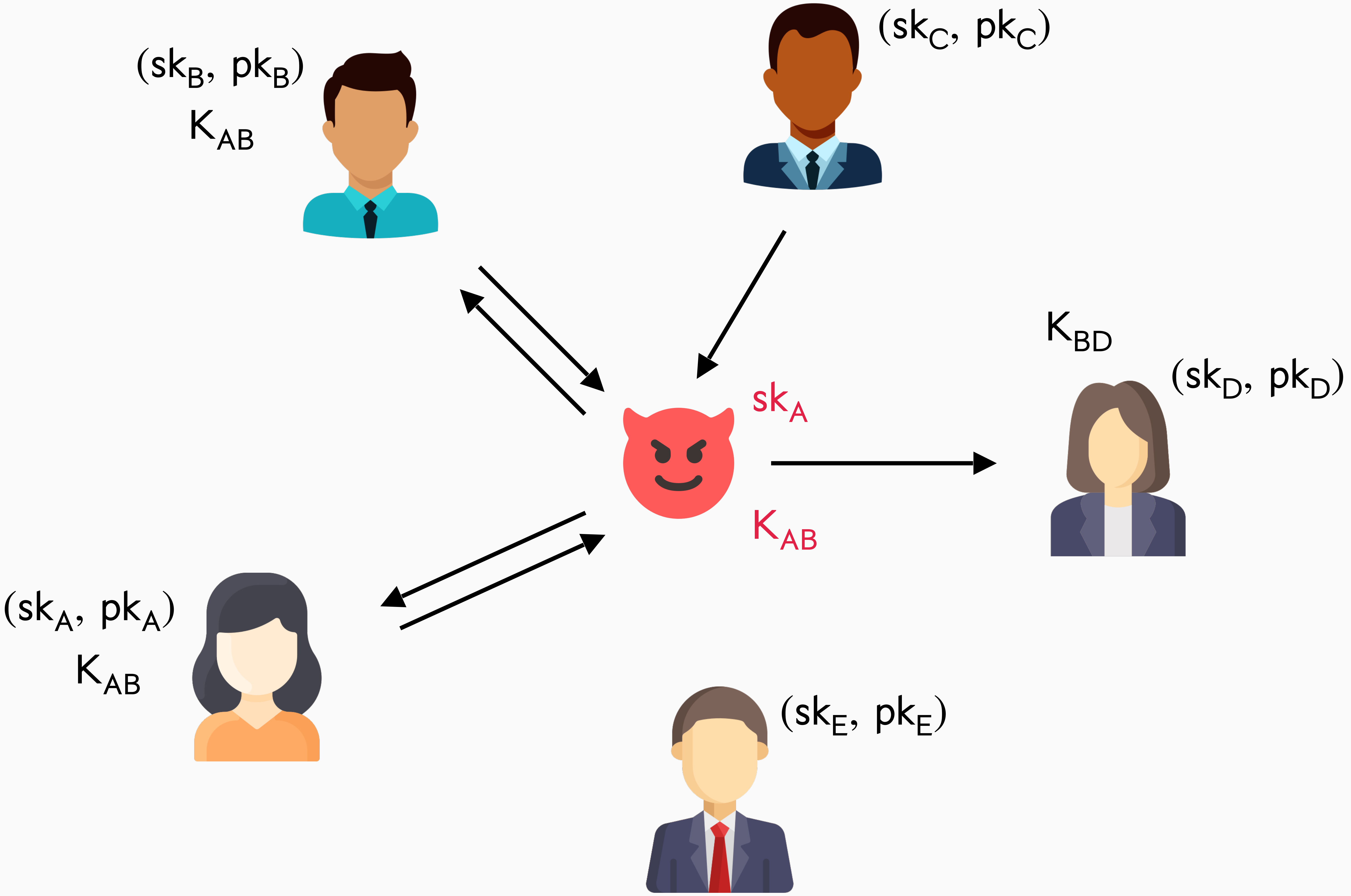
- Controls the network
- Corrupts parties
- Reveals session keys
- Test (challenge) session keys

$$\mathsf{K}_{BD} \approx \mathsf{K}_{\$}$$

Concrete Security

Modeling Security

Execution Environment: N users, S sessions



$(\mathsf{sk_B}, \mathsf{pk_B})$
$\mathsf{K_{AB}}$
$(\mathsf{sk_C}, \mathsf{pk_C})$
$\mathsf{K_{BD}}$
$(\mathsf{sk_D}, \mathsf{pk_D})$
$\mathsf{sk_A}$
$\mathsf{K_{AB}}$
$(\mathsf{sk_A}, \mathsf{pk_A})$
$\mathsf{K_{AB}}$
$(\mathsf{sk_E}, \mathsf{pk_E})$

Adversary

- Controls the network
- Corrupts parties
- Reveals session keys
- Test (challenge) session keys

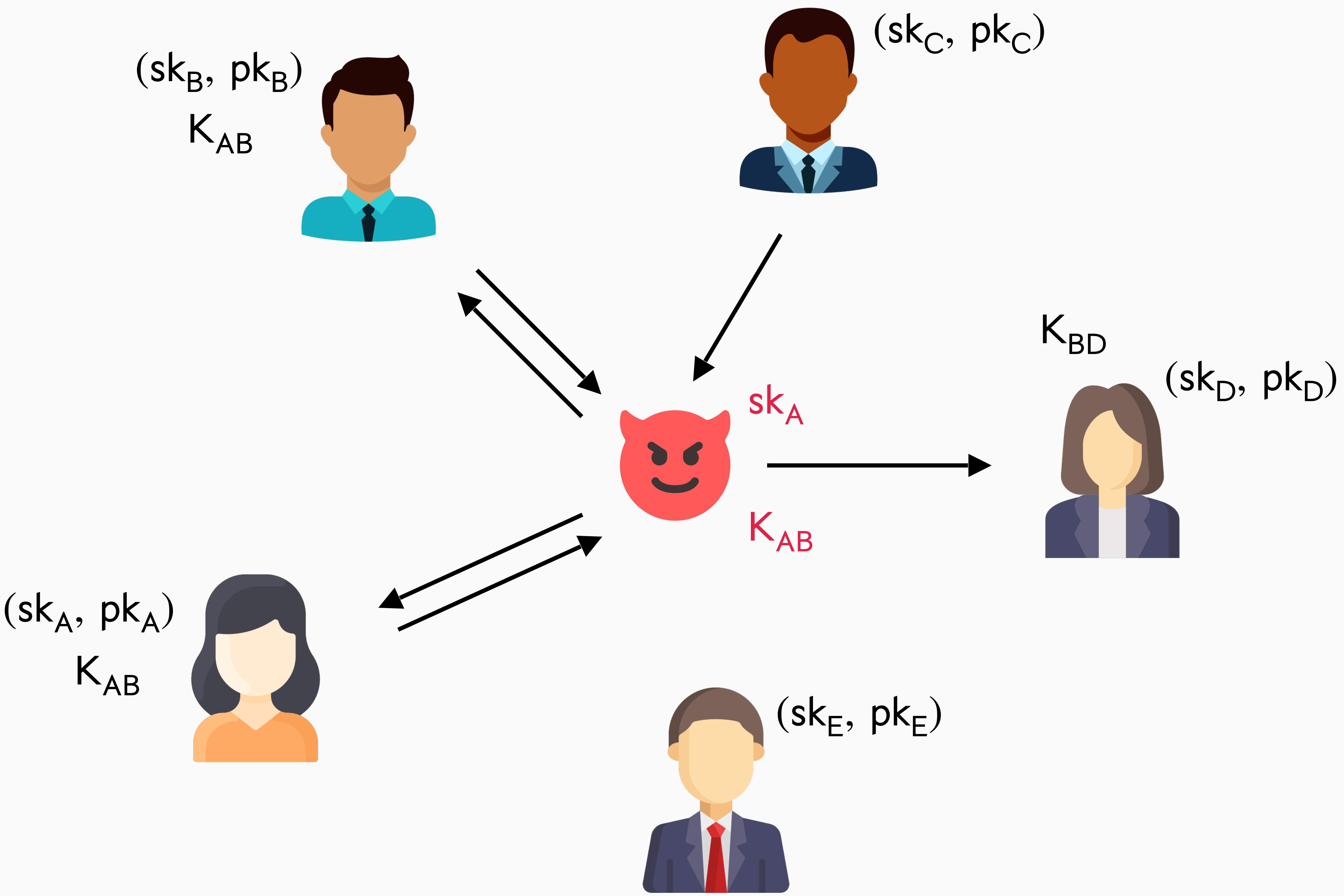
$$\mathsf{K_{BD}} \approx \mathsf{K_\$}$$

Concrete Security

$$\mathsf{Adv}^{\mathsf{AKE}}_{\Pi}(\mathcal{A}) \leq S^2 \cdot \mathsf{Adv}^{\mathsf{P}}(\mathcal{B}) + \ldots$$

Modeling Security

Execution Environment: N users, S sessions



$(\mathsf{sk}_B, \mathsf{pk}_B)$
K_{AB}
$(\mathsf{sk}_C, \mathsf{pk}_C)$
K_{BD}
$(\mathsf{sk}_D, \mathsf{pk}_D)$
sk_A
K_{AB}
$(\mathsf{sk}_A, \mathsf{pk}_A)$
K_{AB}
$(\mathsf{sk}_E, \mathsf{pk}_E)$

Adversary

- Controls the network
- Corrupts parties
- Reveals session keys
- Test (challenge) session keys

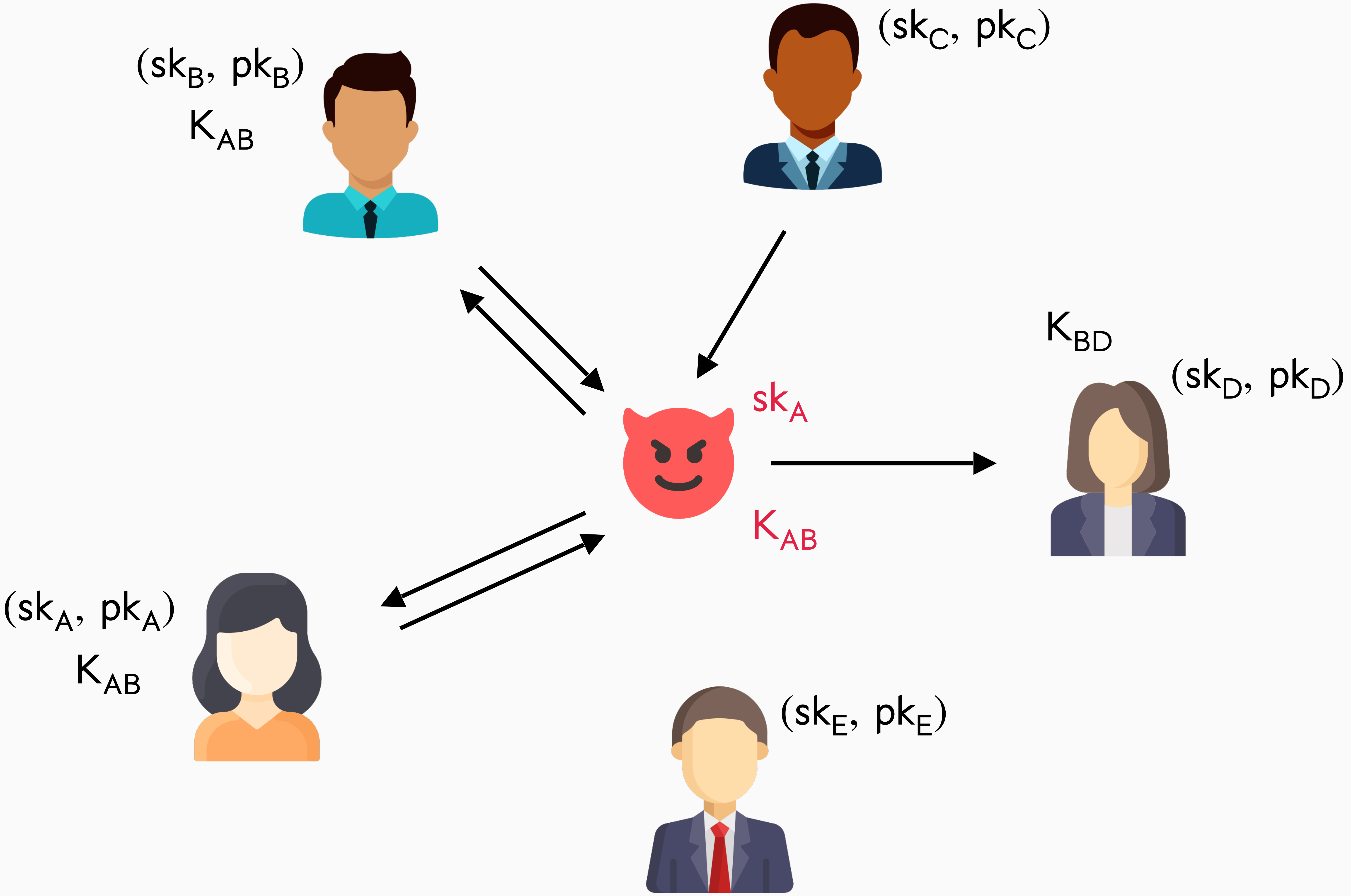
$$\mathsf{K}_{BD} \approx \mathsf{K}_{\$}$$

Concrete Security

$$\mathsf{Adv}_{\Pi}^{\mathsf{AKE}}(\mathcal{A}) \leq \underset{\text{Security Loss}}{S^2} \cdot \mathsf{Adv}^{\mathsf{P}}(\mathcal{B}) + \dots$$

Modeling Security

Execution Environment: N users, S sessions



$(\mathsf{sk_B}, \mathsf{pk_B})$
$\mathsf{K_{AB}}$
$(\mathsf{sk_C}, \mathsf{pk_C})$
$\mathsf{sk_A}$
$\mathsf{K_{AB}}$
$\mathsf{K_{BD}}$
$(\mathsf{sk_D}, \mathsf{pk_D})$
$(\mathsf{sk_A}, \mathsf{pk_A})$
$\mathsf{K_{AB}}$
$(\mathsf{sk_E}, \mathsf{pk_E})$

Adversary

- Controls the network
- Corrupts parties
- Reveals session keys
- Test (challenge) session keys

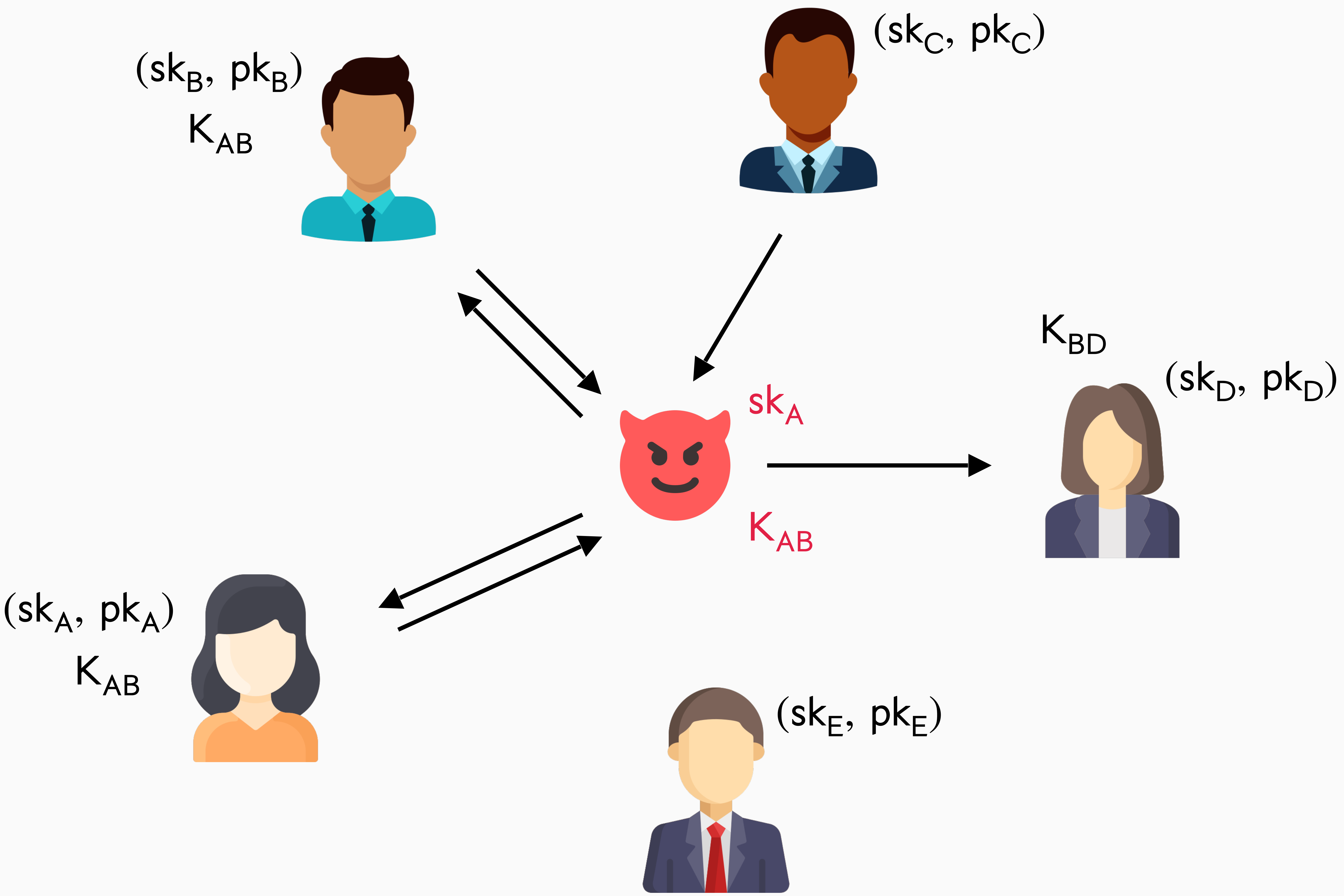
$$\mathsf{K_{BD}} \approx \mathsf{K_\$}$$

Concrete Security

$$\mathsf{Adv}^{\mathsf{AKE}}_{\Pi}(\mathcal{A}) \leq \underset{\text{Security Loss}}{N} \cdot \mathsf{Adv}^{\mathsf{P}}(\mathcal{B}) + \ldots$$

Modeling Security

Execution Environment: N users, S sessions



$(\mathsf{sk_B}, \mathsf{pk_B})$
$\mathsf{K_{AB}}$
$(\mathsf{sk_C}, \mathsf{pk_C})$
$\mathsf{sk_A}$
$\mathsf{K_{AB}}$
$\mathsf{K_{BD}}$
$(\mathsf{sk_D}, \mathsf{pk_D})$
$(\mathsf{sk_A}, \mathsf{pk_A})$
$\mathsf{K_{AB}}$
$(\mathsf{sk_E}, \mathsf{pk_E})$

Adversary

- Controls the network
- Corrupts parties
- Reveals session keys
- Test (challenge) session keys

$$\mathsf{K_{BD}} \approx \mathsf{K_\$}$$

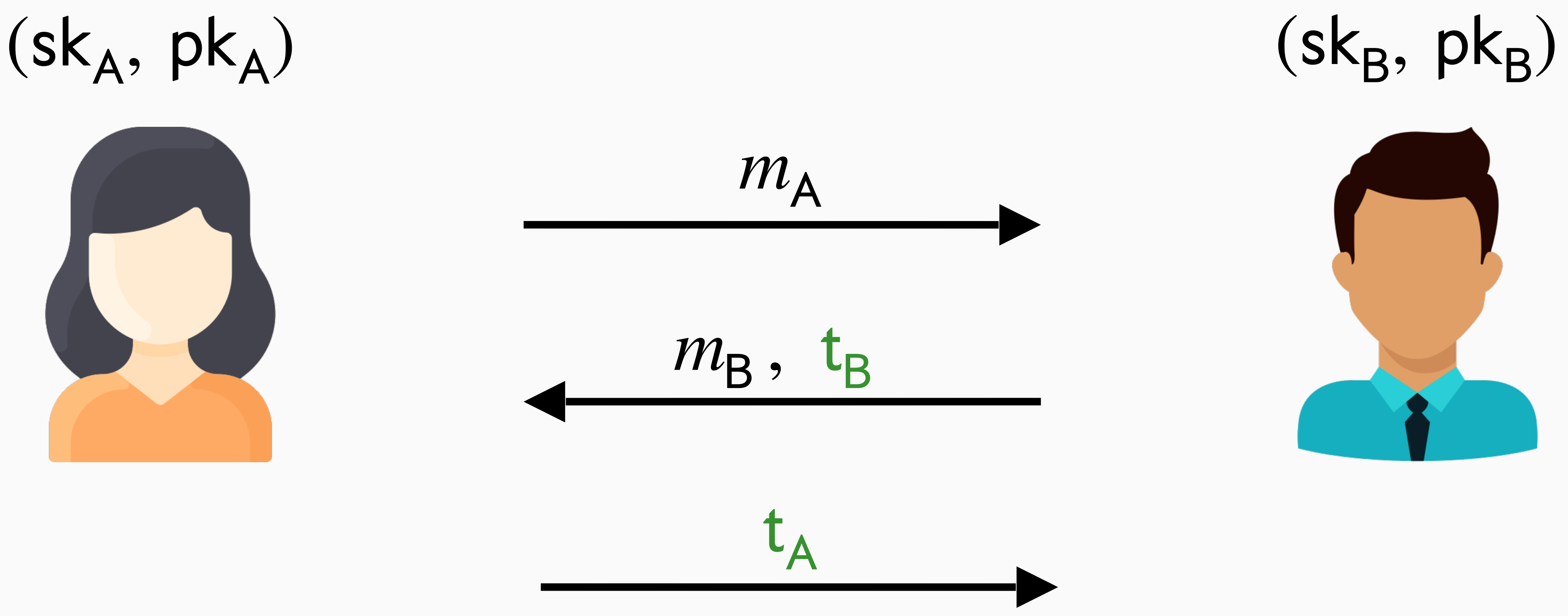
Concrete Security

$$\mathsf{Adv}^{\mathsf{AKE}}_{\Pi}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{P}}(\mathcal{B}) + \ldots$$

Tight!

Tightness Optimality

WFS-secure AKE protocol $\Pi \Rightarrow$ **FFS-secure AKE protocol** Π'



$(\mathsf{sk_A}, \mathsf{pk_A})$

$(\mathsf{sk_B}, \mathsf{pk_B})$

m_A

m_B , $\mathsf{t_B}$

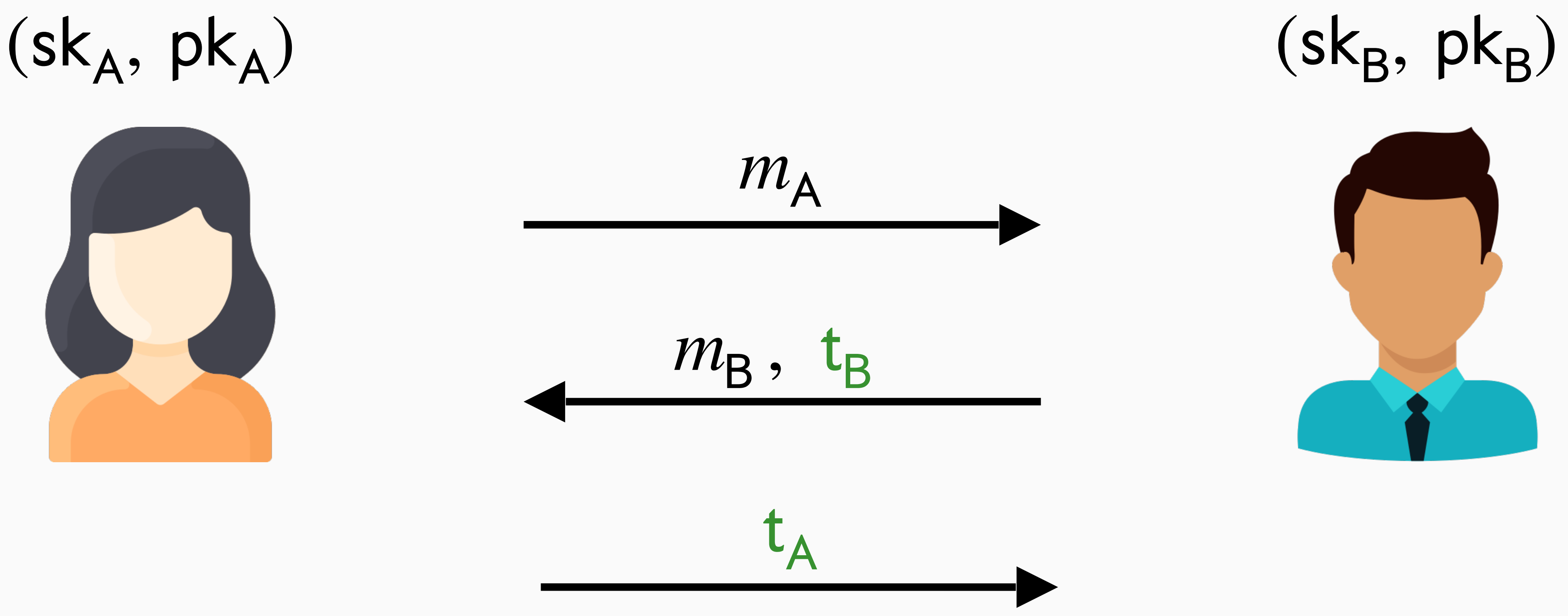
$\mathsf{t_A}$

$(\mathsf{K_{AB}}, \mathsf{t_A}, \mathsf{t_B}) = \mathsf{Derive}(\mathsf{K}, \mathsf{context})$

$(\mathsf{K_{AB}}, \mathsf{t_A}, \mathsf{t_B}) = \mathsf{Derive}(\mathsf{K}, \mathsf{context})$

Tightness Optimality

WFS-secure AKE protocol $\Pi \Rightarrow$ **FFS-secure AKE protocol** Π'



$(\mathsf{sk}_\mathsf{A}, \mathsf{pk}_\mathsf{A})$

$(\mathsf{sk}_\mathsf{B}, \mathsf{pk}_\mathsf{B})$

m_A

m_B, t_B

t_A

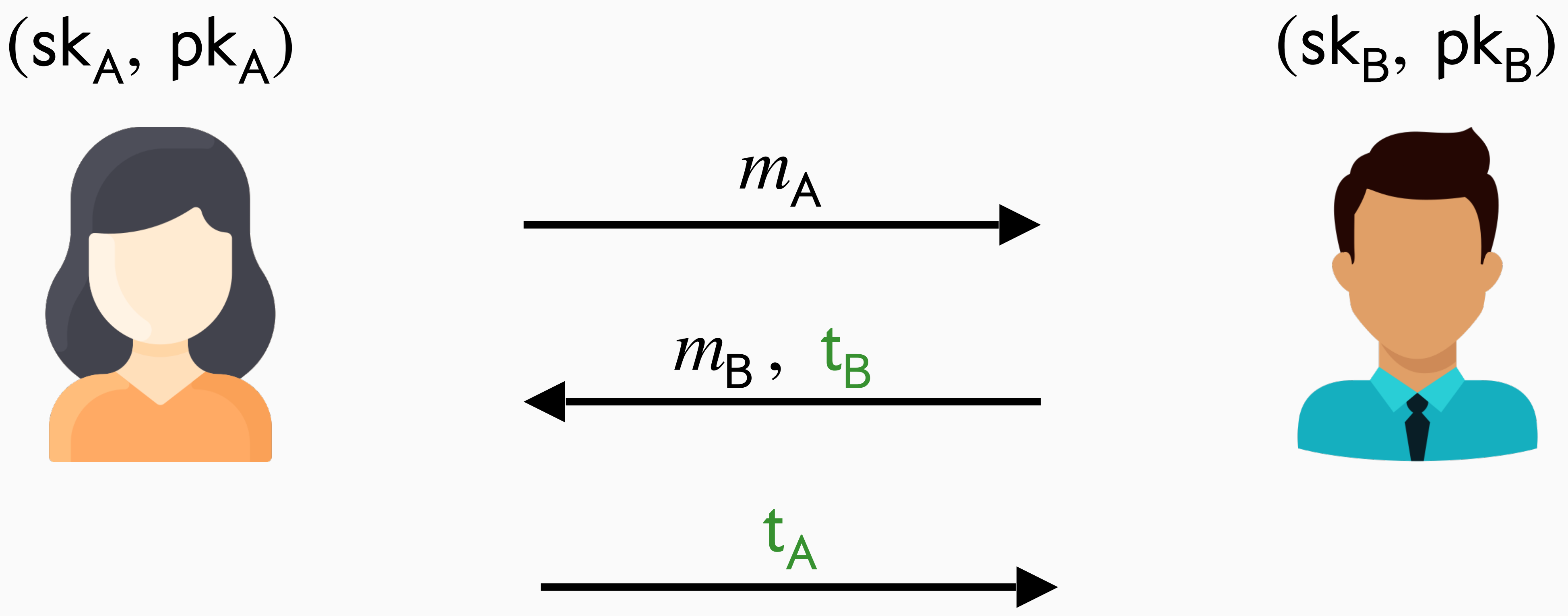
$(\mathsf{K}_\mathsf{AB}, \mathsf{t}_\mathsf{A}, \mathsf{t}_\mathsf{B}) = \mathsf{Derive}(\mathsf{K}, \mathsf{context})$

$(\mathsf{K}_\mathsf{AB}, \mathsf{t}_\mathsf{A}, \mathsf{t}_\mathsf{B}) = \mathsf{Derive}(\mathsf{K}, \mathsf{context})$

$$\mathsf{Adv}_{\Pi'}^{\mathsf{AKE-FFS}}(\mathcal{A}) \leq N \cdot \mathsf{Adv}_{\Pi}^{\mathsf{AKE-WFS}}(\mathcal{B}) + \ldots$$

Tightness Optimality

WFS-secure AKE protocol Π $\Rightarrow$ FFS-secure AKE protocol Π'



$(\mathsf{sk_A}, \mathsf{pk_A})$

$(\mathsf{sk_B}, \mathsf{pk_B})$

m_A

m_B , $\mathsf{t_B}$

$\mathsf{t_A}$

$(\mathsf{K_{AB}}, \mathsf{t_A}, \mathsf{t_B}) = \mathsf{Derive}(\mathsf{K}, \mathsf{context})$

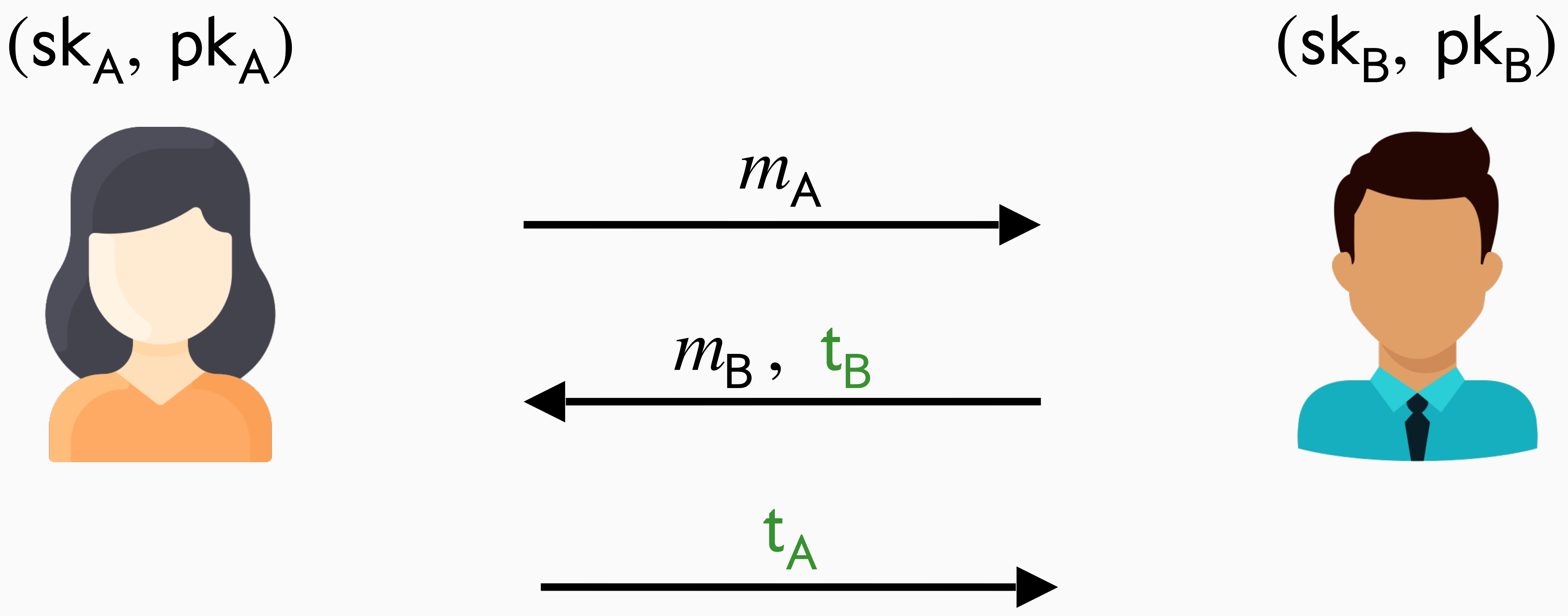
$(\mathsf{K_{AB}}, \mathsf{t_A}, \mathsf{t_B}) = \mathsf{Derive}(\mathsf{K}, \mathsf{context})$

$$\mathsf{Adv}^{\mathsf{AKE-FFS}}_{\Pi'}(\mathcal{A}) \le N \cdot \mathsf{Adv}^{\mathsf{AKE-WFS}}_{\Pi}(\mathcal{B}) + \dots$$

This is optimal for a large class of protocols [C:GGJJ23]

Tightness Optimality

WFS-secure AKE protocol Π ⇒ FFS-secure AKE protocol Π'



$(\mathsf{sk_A}, \mathsf{pk_A})$

$(\mathsf{sk_B}, \mathsf{pk_B})$

m_A

m_B, $\mathsf{t_B}$

$\mathsf{t_A}$

$(\mathsf{K_{AB}}, \mathsf{t_A}, \mathsf{t_B}) = \mathsf{Derive}(\mathsf{K}, \mathsf{context})$

$(\mathsf{K_{AB}}, \mathsf{t_A}, \mathsf{t_B}) = \mathsf{Derive}(\mathsf{K}, \mathsf{context})$

(Very brief) Intuition

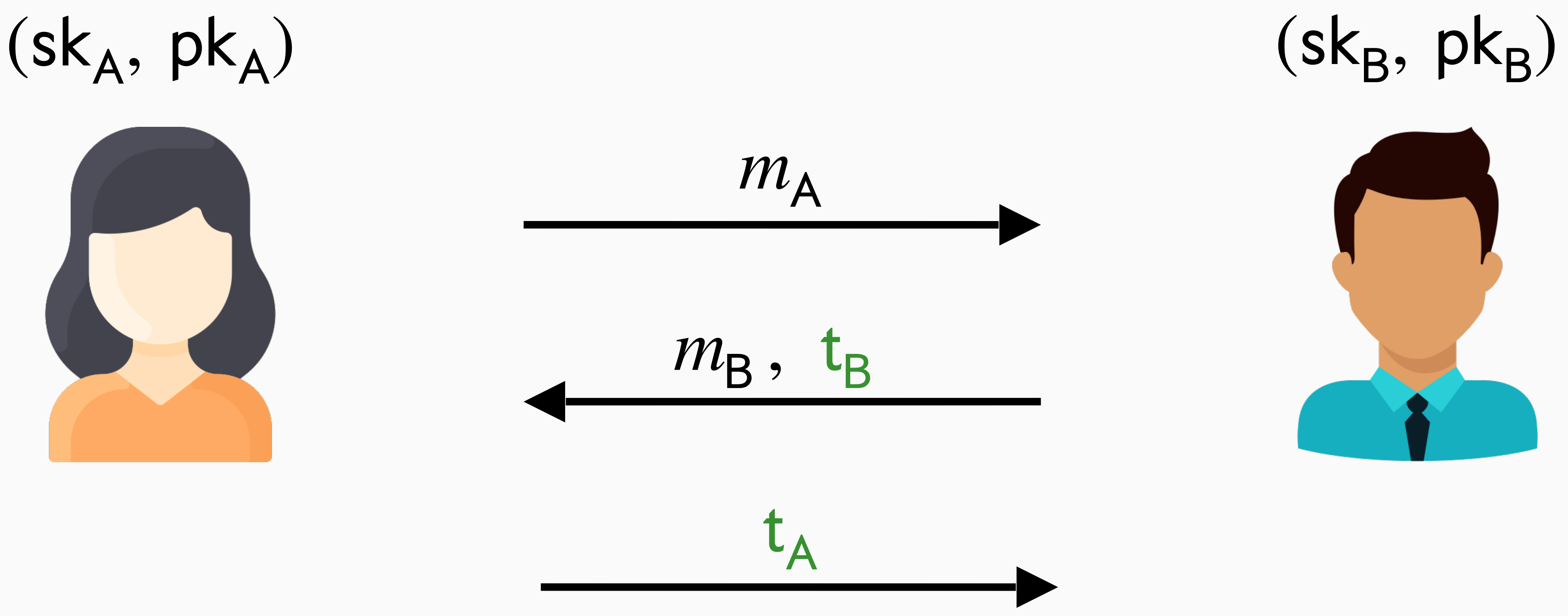
- Reduction needs to simulate valid tags without necessarily knowing K
- Must either reveal or test a session

$$\mathsf{Adv}_{\Pi'}^{\mathsf{AKE-FFS}}(\mathcal{A}) \leq N \cdot \mathsf{Adv}_{\Pi}^{\mathsf{AKE-WFS}}(\mathcal{B}) + \dots$$

This is optimal for a large class of protocols [C:GGJJ23]

Tightness Optimality

WFS-secure AKE protocol Π ⇒ FFS-secure AKE protocol Π'



$(\mathsf{sk}_\mathsf{A}, \mathsf{pk}_\mathsf{A})$

$(\mathsf{sk}_\mathsf{B}, \mathsf{pk}_\mathsf{B})$

m_A

m_B, t_B

t_A

$(\mathsf{K}_{\mathsf{AB}}, \mathsf{t}_\mathsf{A}, \mathsf{t}_\mathsf{B}) = \mathsf{Derive}(\mathsf{K}, \mathsf{context})$

$(\mathsf{K}_{\mathsf{AB}}, \mathsf{t}_\mathsf{A}, \mathsf{t}_\mathsf{B}) = \mathsf{Derive}(\mathsf{K}, \mathsf{context})$

(Very brief) Intuition

- Reduction needs to simulate valid tags without necessarily knowing K
- Must either reveal or test a session

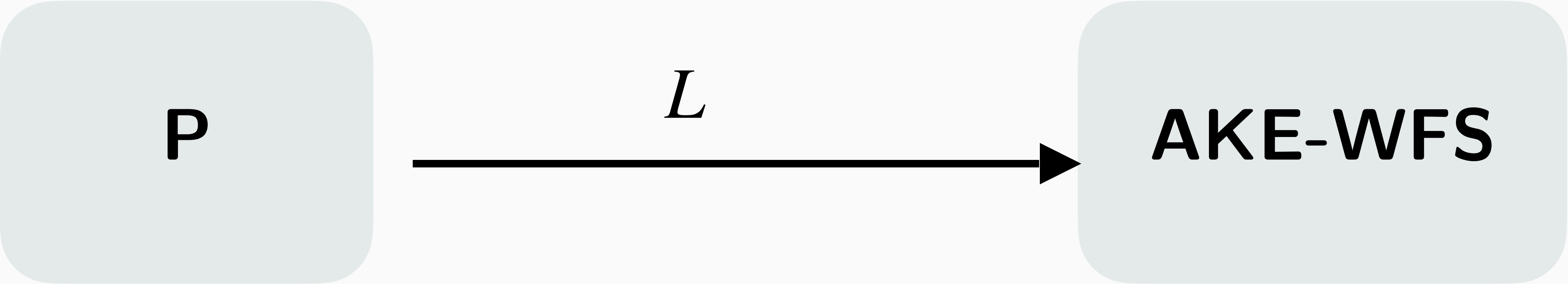
⇒ Commitment problem!

$$\mathsf{Adv}^{\mathsf{AKE-FFS}}_{\Pi'}(\mathcal{A}) \le N \cdot \mathsf{Adv}^{\mathsf{AKE-WFS}}_{\Pi}(\mathcal{B}) + \dots$$

This is optimal for a large class of protocols [C:GGJJ23]

Concrete Bounds for Key Confirmation

Naive



P

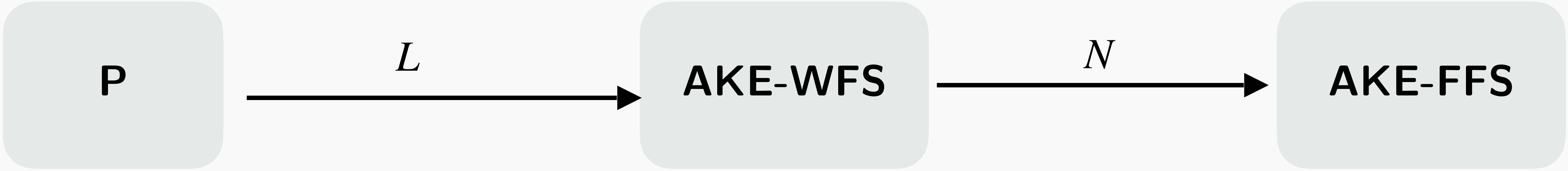
L

AKE-WFS

Security Loss →

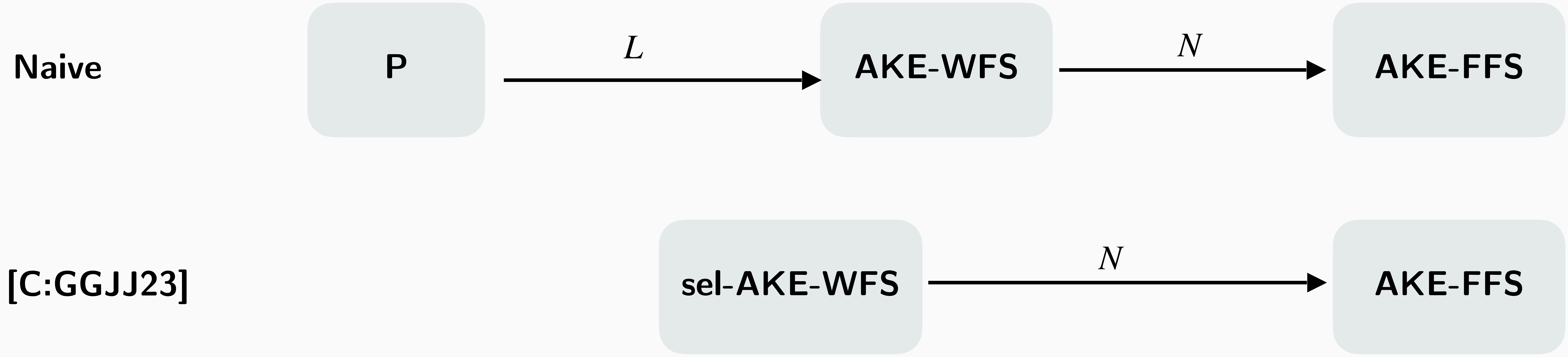
Concrete Bounds for Key Confirmation

Naive



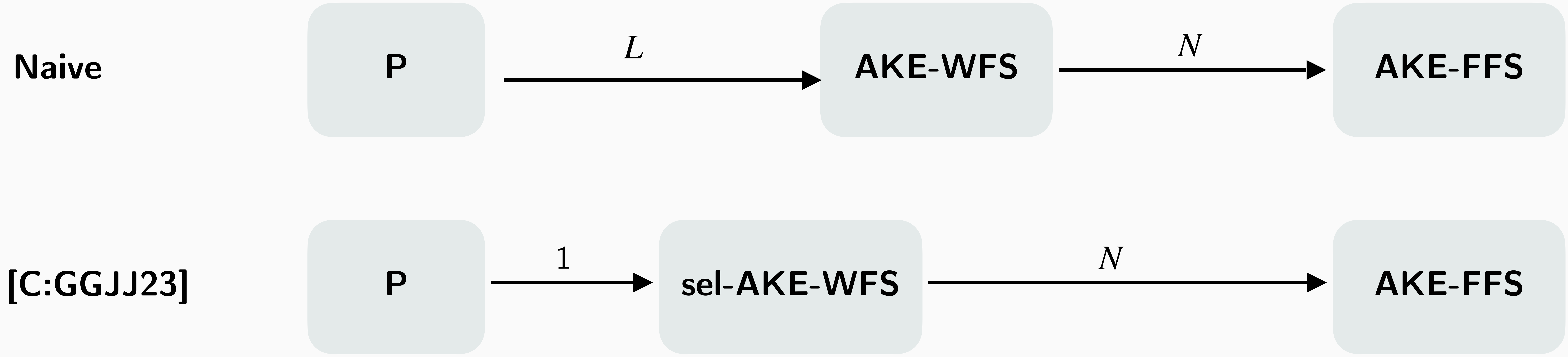
Security Loss →

Concrete Bounds for Key Confirmation



Security Loss

Concrete Bounds for Key Confirmation



Naive

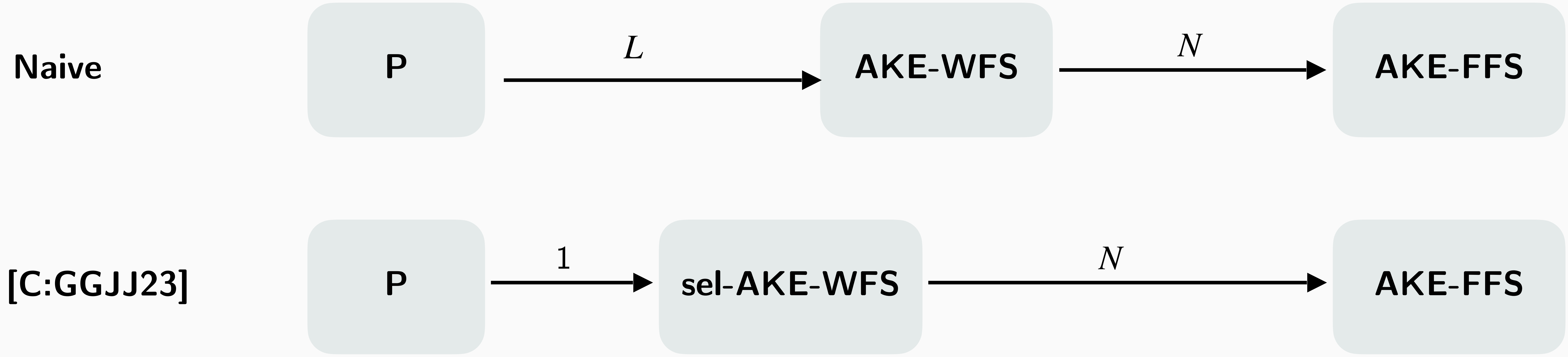
P $\xrightarrow{L}$ AKE-WFS $\xrightarrow{N}$ AKE-FFS

[C:GGJJ23]

P $\xrightarrow{1}$ sel-AKE-WFS $\xrightarrow{N}$ AKE-FFS

Security Loss →

Concrete Bounds for Key Confirmation



Naive

P $\xrightarrow{L}$ AKE-WFS $\xrightarrow{N}$ AKE-FFS

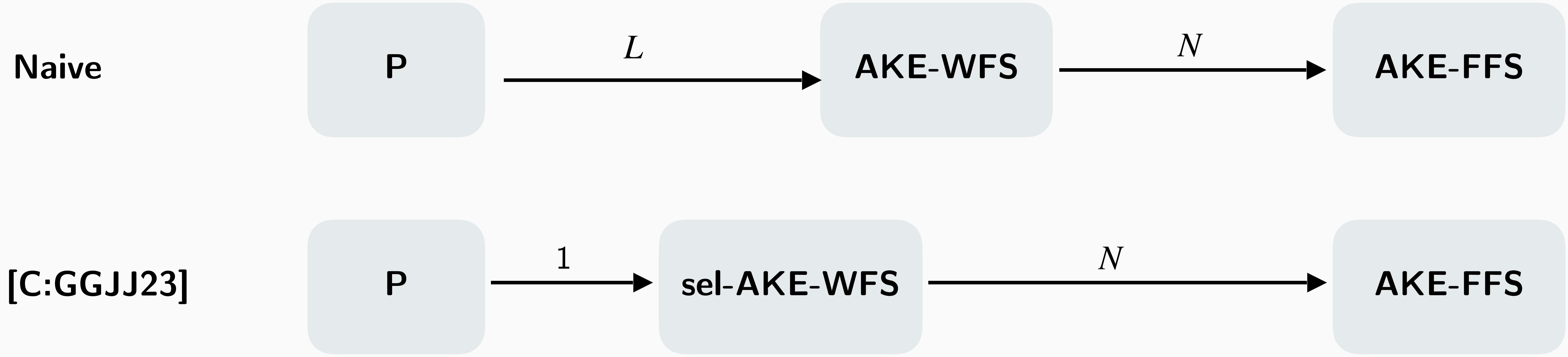
[C:GGJJ23]

P $\xrightarrow{1}$ sel-AKE-WFS $\xrightarrow{N}$ AKE-FFS

Can we get “better than optimal” tightness?

Security Loss →

Concrete Bounds for Key Confirmation



Naive: P $\xrightarrow{L}$ AKE-WFS $\xrightarrow{N}$ AKE-FFS

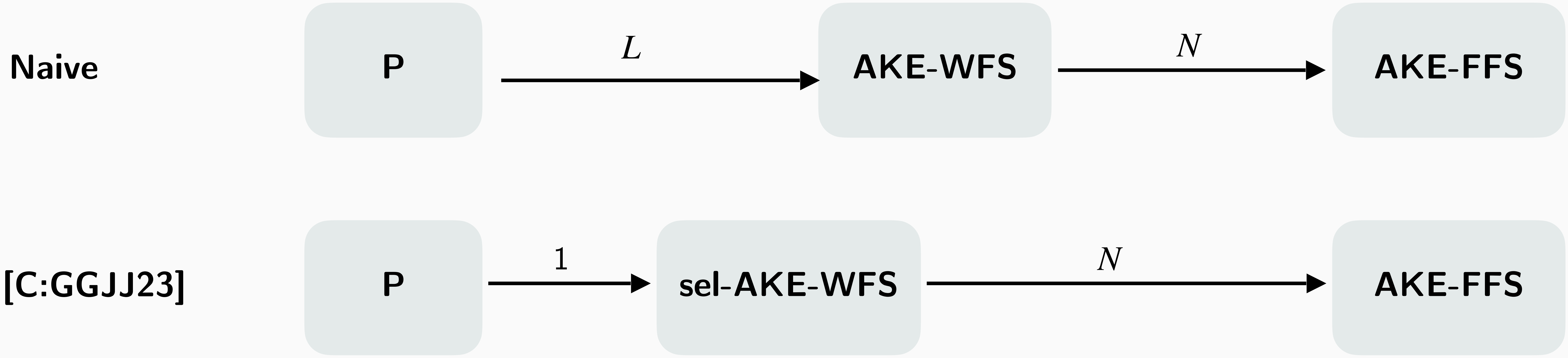
[C:GGJJ23]: P $\xrightarrow{1}$ sel-AKE-WFS $\xrightarrow{N}$ AKE-FFS

Can we get “better than optimal” tightness?

- In the above: what if $L = 1$?

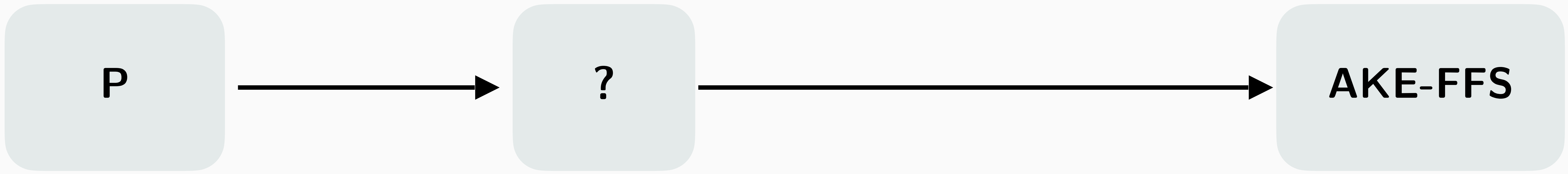
Security Loss $\longrightarrow$

Concrete Bounds for Key Confirmation



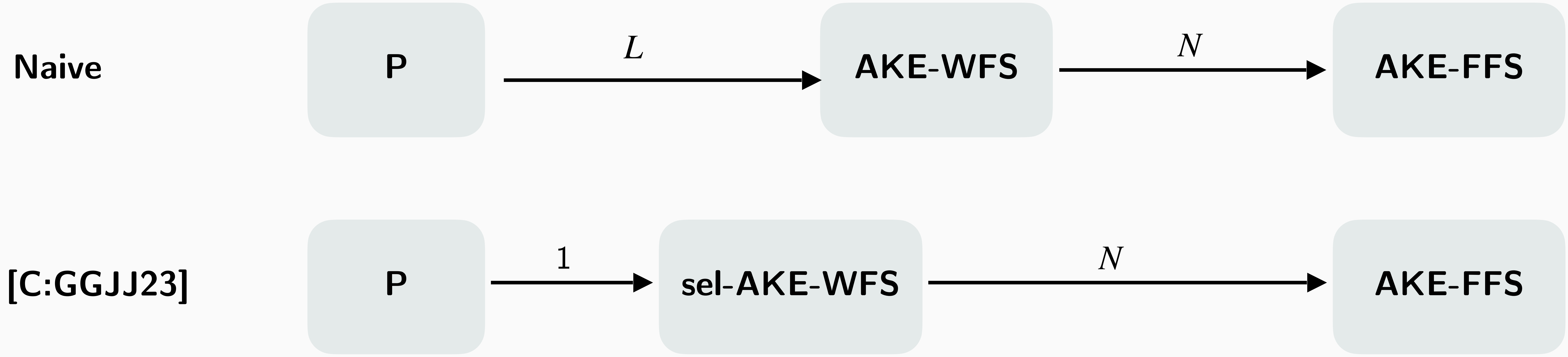
Can we get "better than optimal" tightness?

- In the above: what if $L = 1$?



Security Loss →

Concrete Bounds for Key Confirmation

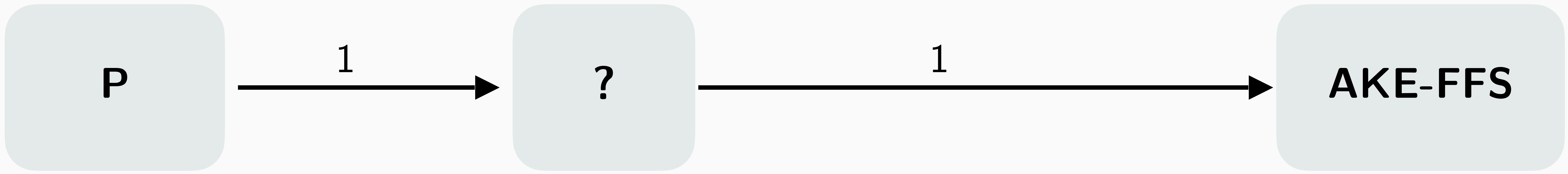


Naive: **P** $\xrightarrow{L}$ **AKE-WFS** $\xrightarrow{N}$ **AKE-FFS**

[C:GGJJ23]: **P** $\xrightarrow{1}$ **sel-AKE-WFS** $\xrightarrow{N}$ **AKE-FFS**

Can we get "better than optimal" tightness?

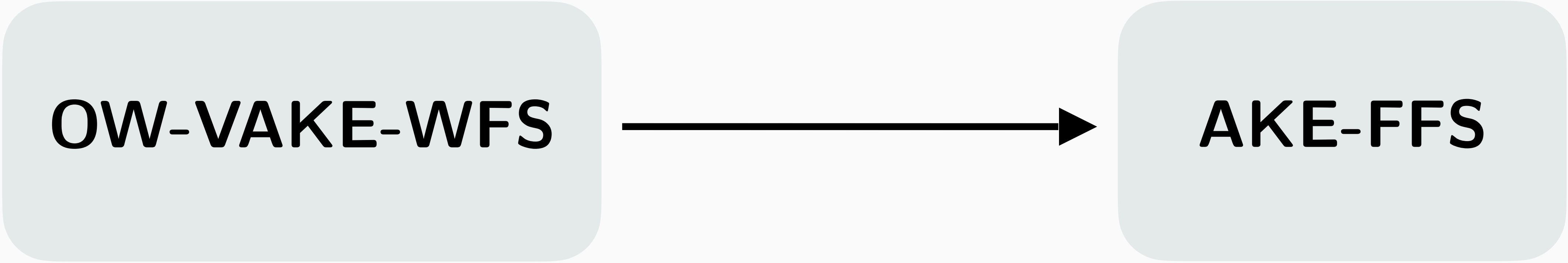
- In the above: what if $L = 1$?



P $\xrightarrow{1}$ **?** $\xrightarrow{1}$ **AKE-FFS**

Security Loss $\longrightarrow$

Our Contributions

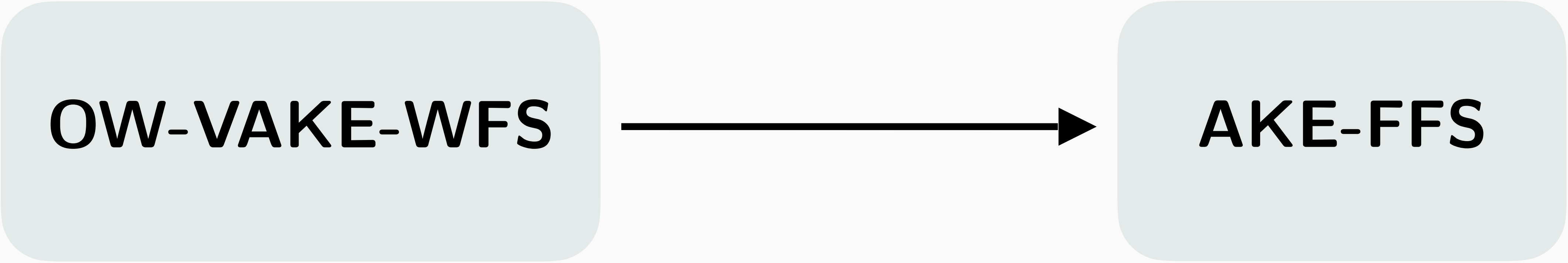


OW-VAKE-WFS

AKE-FFS

Tight

Our Contributions

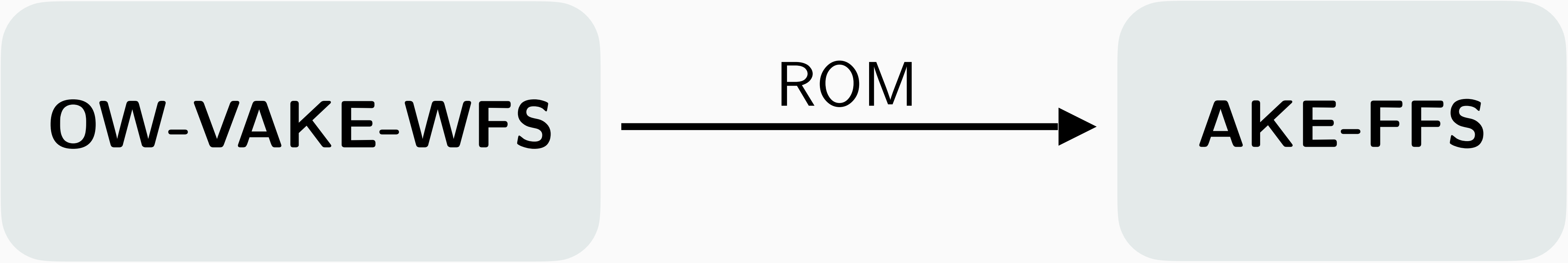


Our Results

- Definition for One-Way Verifiable AKE (OW-VAKE) with WFS

Tight →

Our Contributions

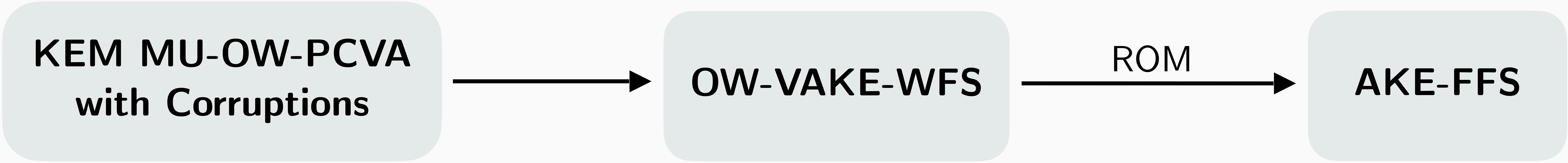


Our Results

- Definition for One-Way Verifiable AKE (OW-VAKE) with WFS
- Key Confirmation preserves tightness (in the ROM)

Tight

Our Contributions



KEM MU-OW-PCVA with Corruptions

OW-VAKE-WFS

ROM

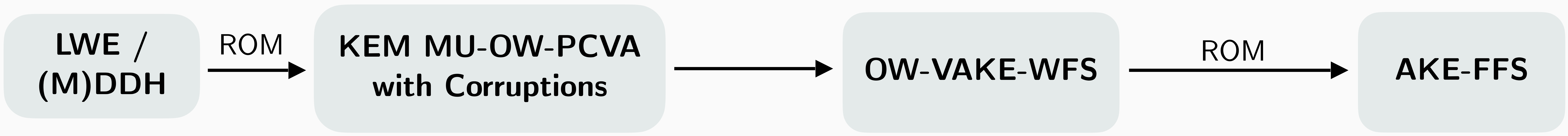
AKE-FFS

Our Results

- Definition for One-Way Verifiable AKE (OW-VAKE) with WFS
- Key Confirmation preserves tightness (in the ROM)
- OW-VAKE from OW-KEMs

Tight

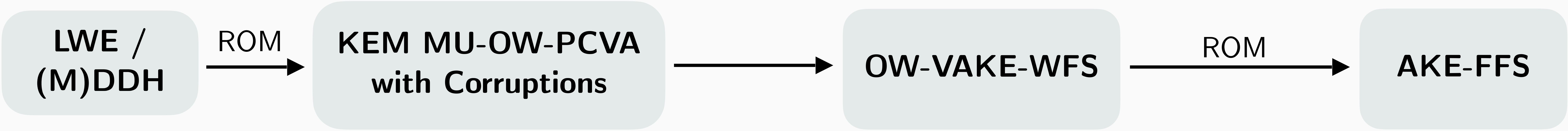
Our Contributions



Our Results

- Definition for One-Way Verifiable AKE (OW-VAKE) with WFS
- Key Confirmation preserves tightness (in the ROM)
- OW-VAKE from OW-KEMs

Our Contributions



Our Results

- Definition for One-Way Verifiable AKE (OW-VAKE) with WFS
- Key Confirmation preserves tightness (in the ROM)
- OW-VAKE from OW-KEMs
- Key Confirmation in the QROM (loss of N)

Tight

Verifiable Authenticated Key Exchange (VAKE)

AKE-WFS

OW-VAKE-WFS

Verifiable Authenticated Key Exchange (VAKE)

	AKE-WFS	**OW-VAKE-WFS**
Goal	K^* or $K_{\$}$?	Compute K^*

Verifiable Authenticated Key Exchange (VAKE)

	AKE-WFS	**OW-VAKE-WFS**
Goal	K^* or $\mathsf{K}_\$$?	Compute K^*
Session keys	Reveal K	Check $\mathsf{K}' \stackrel{?}{=} \mathsf{K}$

Verifiable Authenticated Key Exchange (VAKE)

	AKE-WFS	**OW-VAKE-WFS**
Goal	K^* or $K_\$$?	Compute K^*
Session keys	Reveal K	Check $K' \overset{?}{=} K$
Control network	Yes	Yes

Verifiable Authenticated Key Exchange (VAKE)

	AKE-WFS	OW-VAKE-WFS
Goal	K^* or $\mathsf{K}_\$$?	Compute K^*
Session keys	Reveal K	Check $\mathsf{K}' \overset{?}{=} \mathsf{K}$
Control network	Yes	Yes
Corrupt users	Yes	Yes

VAKE from KEMs

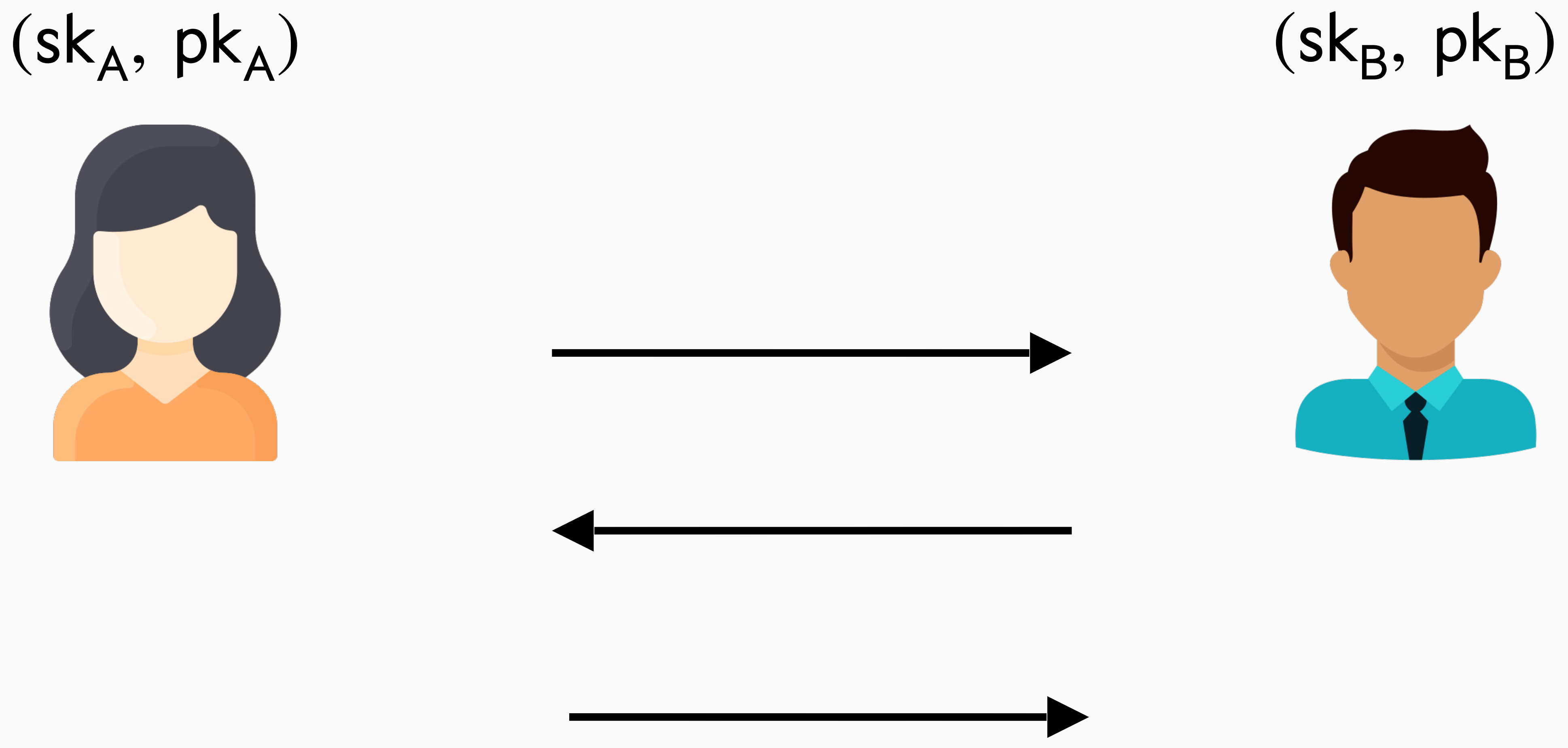
For KEM we need multi-user multi-challenge OW-PCVA security with corruptions

- Includes Plaintext Checking and Ciphertext Validity oracles
- Know tight instantiations from (M)DDH and LWE [EC:JKRS21,C:PanWagZen23]

VAKE from KEMs

For KEM we need multi-user multi-challenge OW-PCVA security with corruptions

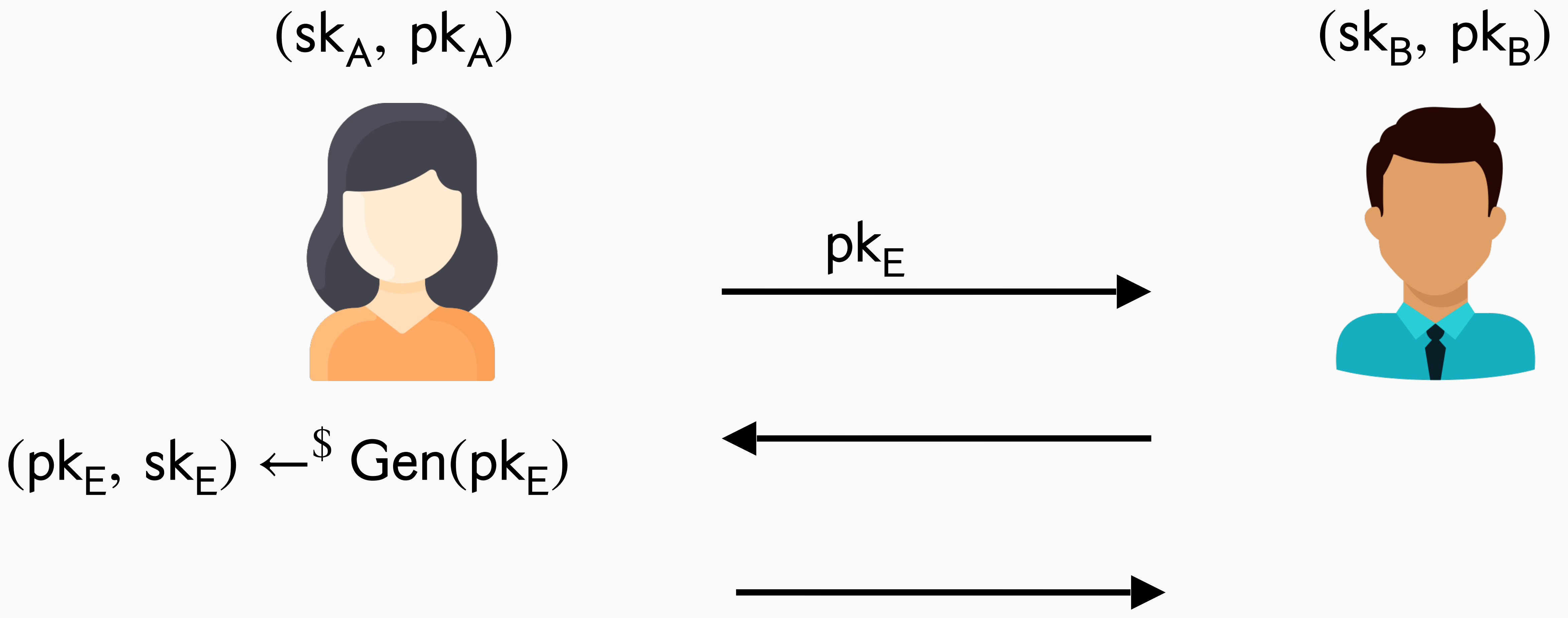
- Includes Plaintext Checking and Ciphertext Validity oracles
- Know tight instantiations from (M)DDH and LWE [EC:JKRS21,C:PanWagZen23]


$(\mathsf{sk}_A, \mathsf{pk}_A)$
$(\mathsf{sk}_B, \mathsf{pk}_B)$

VAKE from KEMs

For KEM we need multi-user multi-challenge OW-PCVA security with corruptions

- Includes Plaintext Checking and Ciphertext Validity oracles
- Know tight instantiations from (M)DDH and LWE [EC:JKRS21,C:PanWagZen23]



$(\mathsf{sk}_A, \mathsf{pk}_A)$

$(\mathsf{sk}_B, \mathsf{pk}_B)$

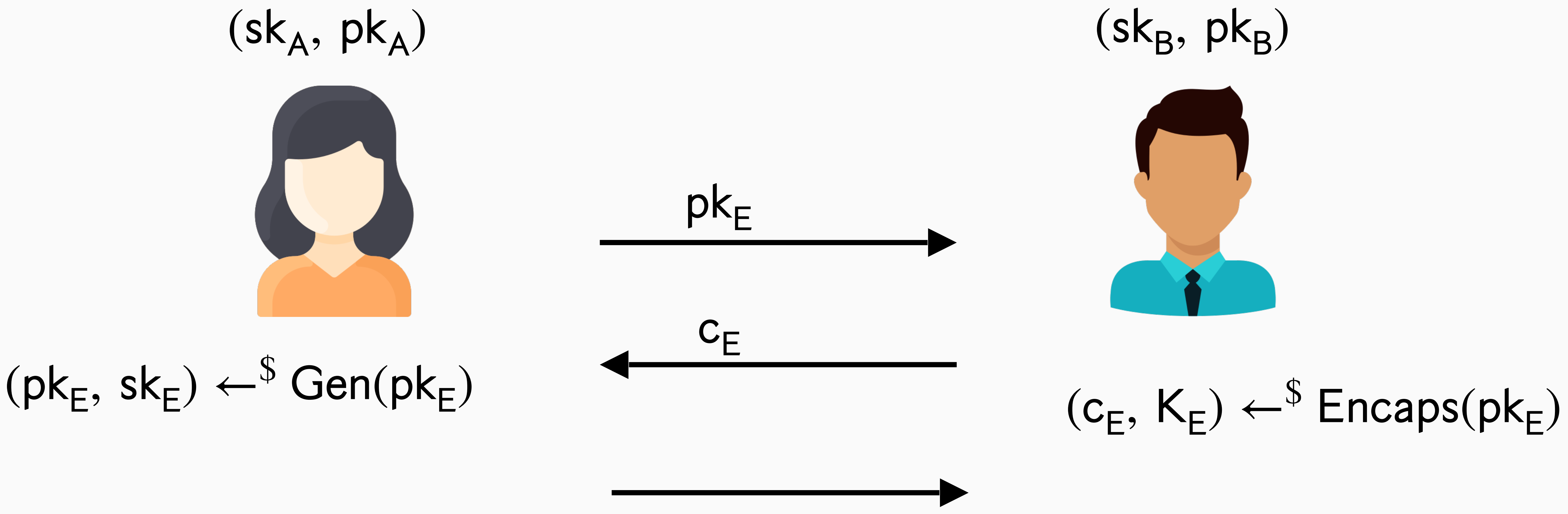
pk_E

$(\mathsf{pk}_E, \mathsf{sk}_E) \xleftarrow{\$} \mathsf{Gen}(\mathsf{pk}_E)$

VAKE from KEMs

For KEM we need multi-user multi-challenge OW-PCVA security with corruptions

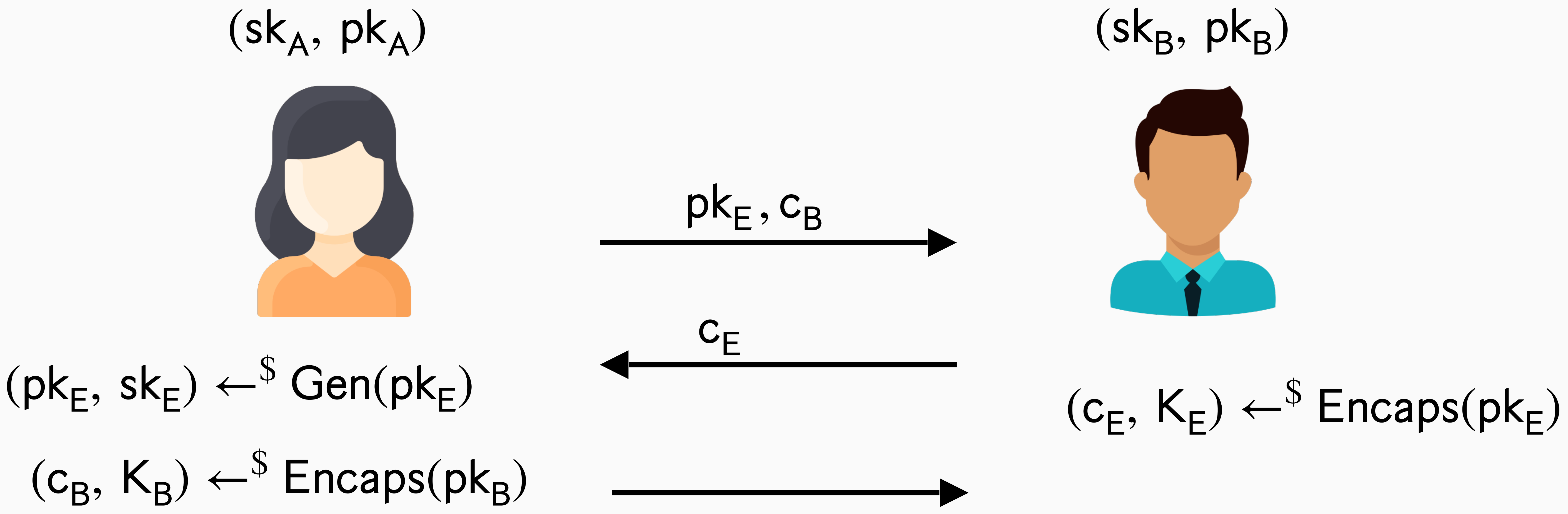
- Includes Plaintext Checking and Ciphertext Validity oracles
- Know tight instantiations from (M)DDH and LWE [EC:JKRS21,C:PanWagZen23]


$(\mathsf{sk_A}, \mathsf{pk_A})$
$(\mathsf{sk_B}, \mathsf{pk_B})$
$\mathsf{pk_E}$
$\mathsf{c_E}$
$(\mathsf{pk_E}, \mathsf{sk_E}) \xleftarrow{\$} \mathsf{Gen}(\mathsf{pk_E})$
$(\mathsf{c_E}, \mathsf{K_E}) \xleftarrow{\$} \mathsf{Encaps}(\mathsf{pk_E})$

VAKE from KEMs

For KEM we need multi-user multi-challenge OW-PCVA security with corruptions

- Includes Plaintext Checking and Ciphertext Validity oracles
- Know tight instantiations from (M)DDH and LWE [EC:JKRS21,C:PanWagZen23]



$(\mathsf{sk}_A, \mathsf{pk}_A)$

$(\mathsf{sk}_B, \mathsf{pk}_B)$

pk_E, c_B

c_E

$(\mathsf{pk}_E, \mathsf{sk}_E) \xleftarrow{\$} \mathsf{Gen}(\mathsf{pk}_E)$

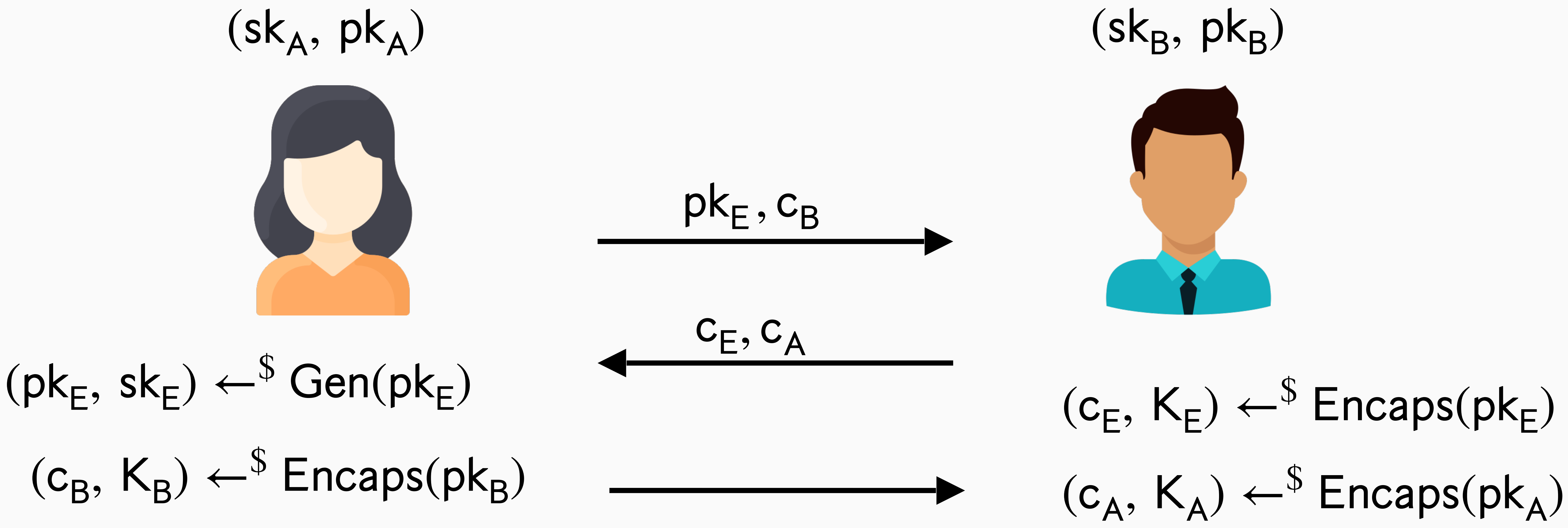
$(c_B, K_B) \xleftarrow{\$} \mathsf{Encaps}(\mathsf{pk}_B)$

$(c_E, K_E) \xleftarrow{\$} \mathsf{Encaps}(\mathsf{pk}_E)$

VAKE from KEMs

For KEM we need multi-user multi-challenge OW-PCVA security with corruptions

- Includes Plaintext Checking and Ciphertext Validity oracles
- Know tight instantiations from (M)DDH and LWE [EC:JKRS21,C:PanWagZen23]



$(\mathsf{sk}_A, \mathsf{pk}_A)$

$(\mathsf{sk}_B, \mathsf{pk}_B)$

pk_E, c_B

c_E, c_A

$(\mathsf{pk}_E, \mathsf{sk}_E) \xleftarrow{\$} \mathsf{Gen}(\mathsf{pk}_E)$

$(c_B, K_B) \xleftarrow{\$} \mathsf{Encaps}(\mathsf{pk}_B)$

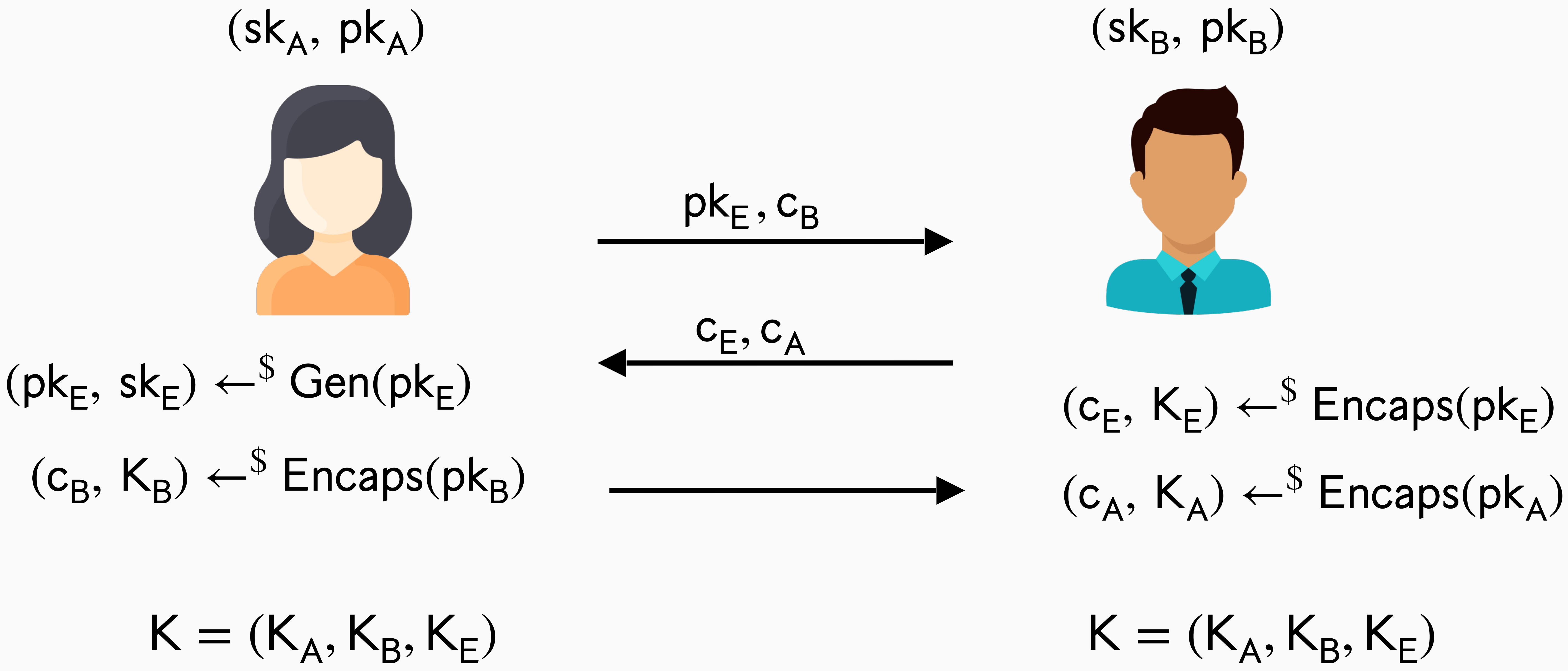
$(c_E, K_E) \xleftarrow{\$} \mathsf{Encaps}(\mathsf{pk}_E)$

$(c_A, K_A) \xleftarrow{\$} \mathsf{Encaps}(\mathsf{pk}_A)$

VAKE from KEMs

For KEM we need multi-user multi-challenge OW-PCVA security with corruptions

- Includes Plaintext Checking and Ciphertext Validity oracles
- Know tight instantiations from (M)DDH and LWE [EC:JKRS21,C:PanWagZen23]



$(\mathsf{sk}_A, \mathsf{pk}_A)$

$(\mathsf{sk}_B, \mathsf{pk}_B)$

$\mathsf{pk}_E, \mathsf{c}_B$

$\mathsf{c}_E, \mathsf{c}_A$

$(\mathsf{pk}_E, \mathsf{sk}_E) \xleftarrow{\$} \mathsf{Gen}(\mathsf{pk}_E)$

$(\mathsf{c}_B, \mathsf{K}_B) \xleftarrow{\$} \mathsf{Encaps}(\mathsf{pk}_B)$

$(\mathsf{c}_E, \mathsf{K}_E) \xleftarrow{\$} \mathsf{Encaps}(\mathsf{pk}_E)$

$(\mathsf{c}_A, \mathsf{K}_A) \xleftarrow{\$} \mathsf{Encaps}(\mathsf{pk}_A)$

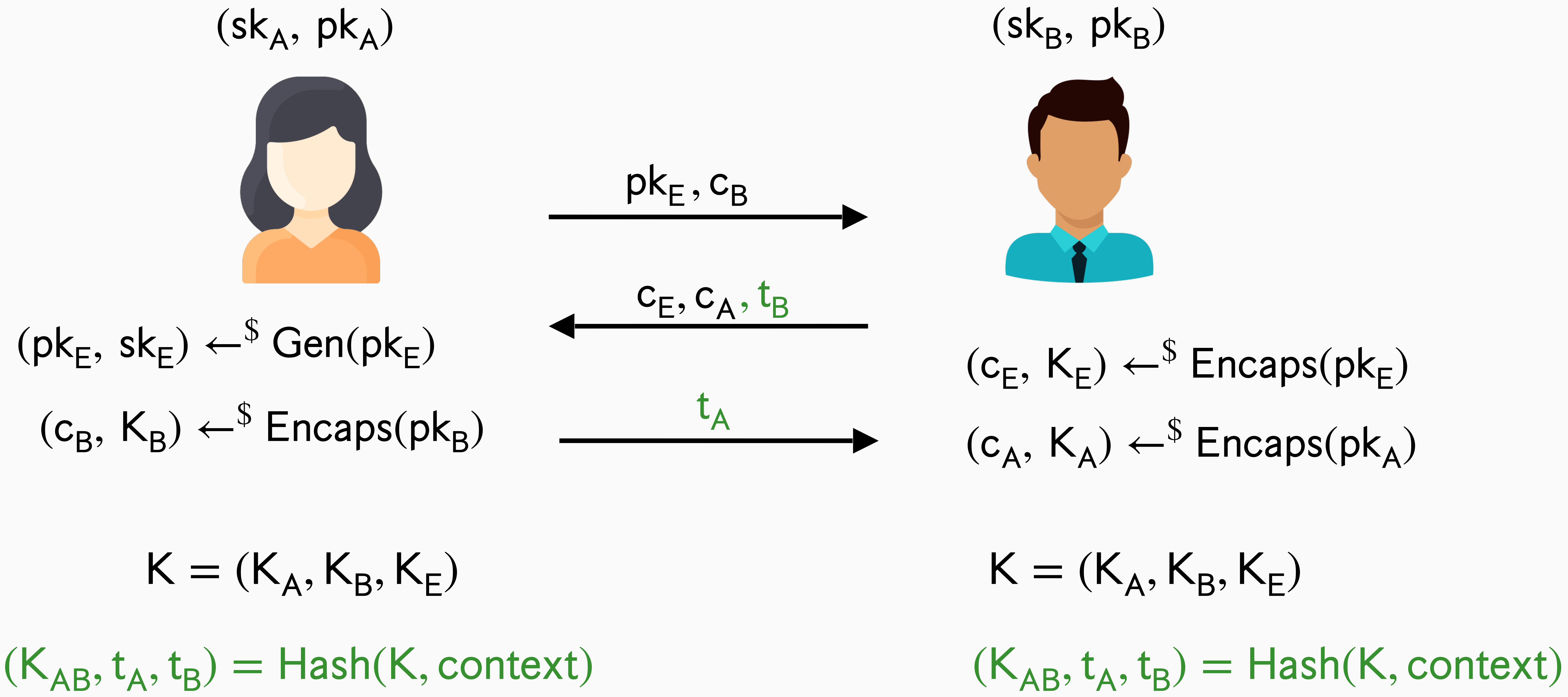
$\mathsf{K} = (\mathsf{K}_A, \mathsf{K}_B, \mathsf{K}_E)$

$\mathsf{K} = (\mathsf{K}_A, \mathsf{K}_B, \mathsf{K}_E)$

VAKE from KEMs

For KEM we need multi-user multi-challenge OW-PCVA security with corruptions

- Includes Plaintext Checking and Ciphertext Validity oracles
- Know tight instantiations from (M)DDH and LWE [EC:JKRS21,C:PanWagZen23]



$(\mathsf{sk}_A,\ \mathsf{pk}_A)$

$(\mathsf{sk}_B,\ \mathsf{pk}_B)$

$\mathsf{pk}_E, c_B \rightarrow$

$\leftarrow c_E, c_A, t_B$

$t_A \rightarrow$

$(\mathsf{pk}_E,\ \mathsf{sk}_E) \xleftarrow{\$} \mathsf{Gen}(\mathsf{pk}_E)$

$(c_B,\ K_B) \xleftarrow{\$} \mathsf{Encaps}(\mathsf{pk}_B)$

$(c_E,\ K_E) \xleftarrow{\$} \mathsf{Encaps}(\mathsf{pk}_E)$

$(c_A,\ K_A) \xleftarrow{\$} \mathsf{Encaps}(\mathsf{pk}_A)$

$K = (K_A, K_B, K_E)$

$K = (K_A, K_B, K_E)$

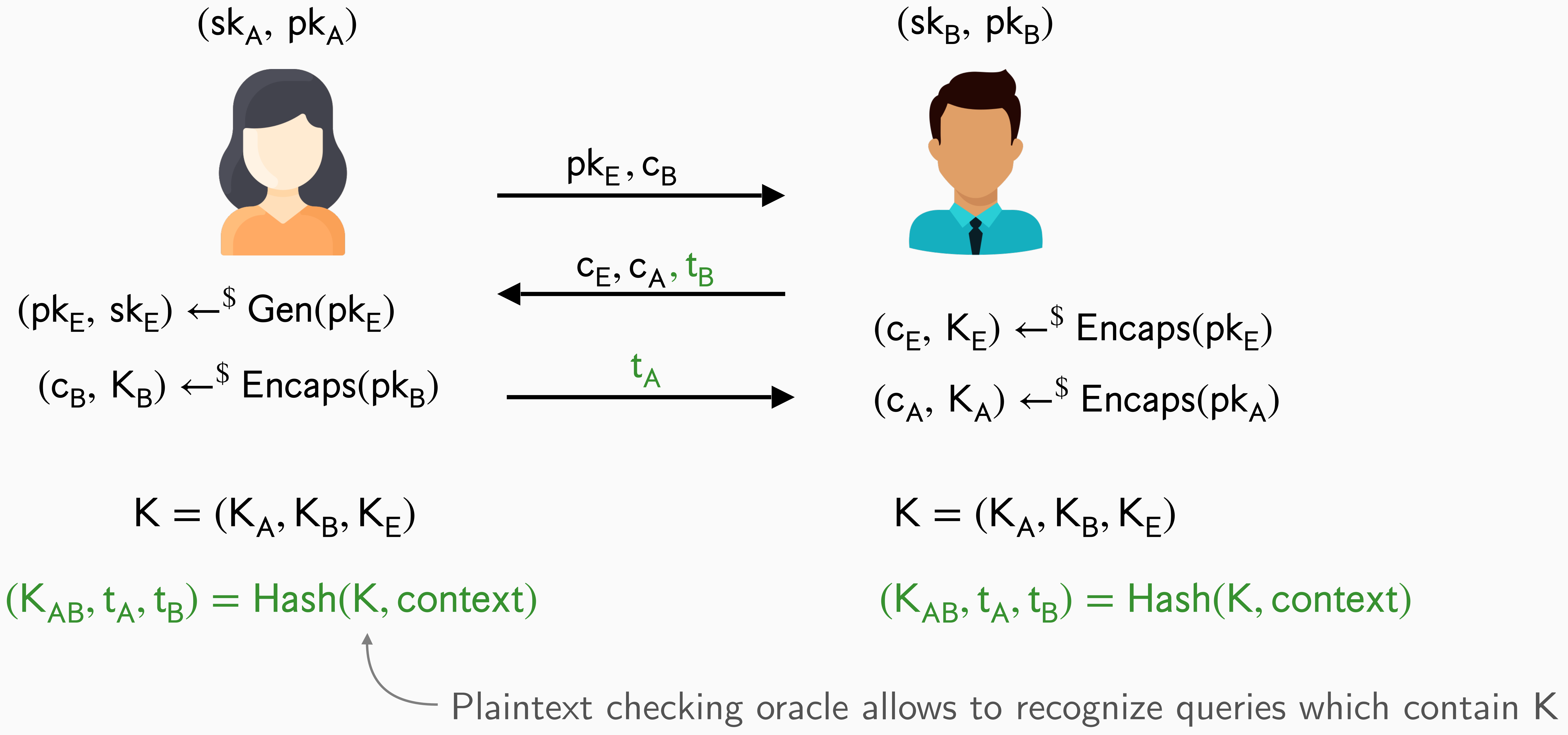
$(K_{AB}, t_A, t_B) = \mathsf{Hash}(K, \mathsf{context})$

$(K_{AB}, t_A, t_B) = \mathsf{Hash}(K, \mathsf{context})$

VAKE from KEMs

For KEM we need multi-user multi-challenge OW-PCVA security with corruptions

- Includes Plaintext Checking and Ciphertext Validity oracles
- Know tight instantiations from (M)DDH and LWE [EC:JKRS21,C:PanWagZen23]



$(\mathsf{sk}_A, \mathsf{pk}_A)$

$(\mathsf{sk}_B, \mathsf{pk}_B)$

pk_E, c_B

c_E, c_A, t_B

t_A

$(\mathsf{pk}_E, \mathsf{sk}_E) \xleftarrow{\$} \mathsf{Gen}(\mathsf{pk}_E)$

$(c_B, K_B) \xleftarrow{\$} \mathsf{Encaps}(\mathsf{pk}_B)$

$(c_E, K_E) \xleftarrow{\$} \mathsf{Encaps}(\mathsf{pk}_E)$

$(c_A, K_A) \xleftarrow{\$} \mathsf{Encaps}(\mathsf{pk}_A)$

$K = (K_A, K_B, K_E)$

$K = (K_A, K_B, K_E)$

$(K_{AB}, t_A, t_B) = \mathsf{Hash}(K, \mathsf{context})$

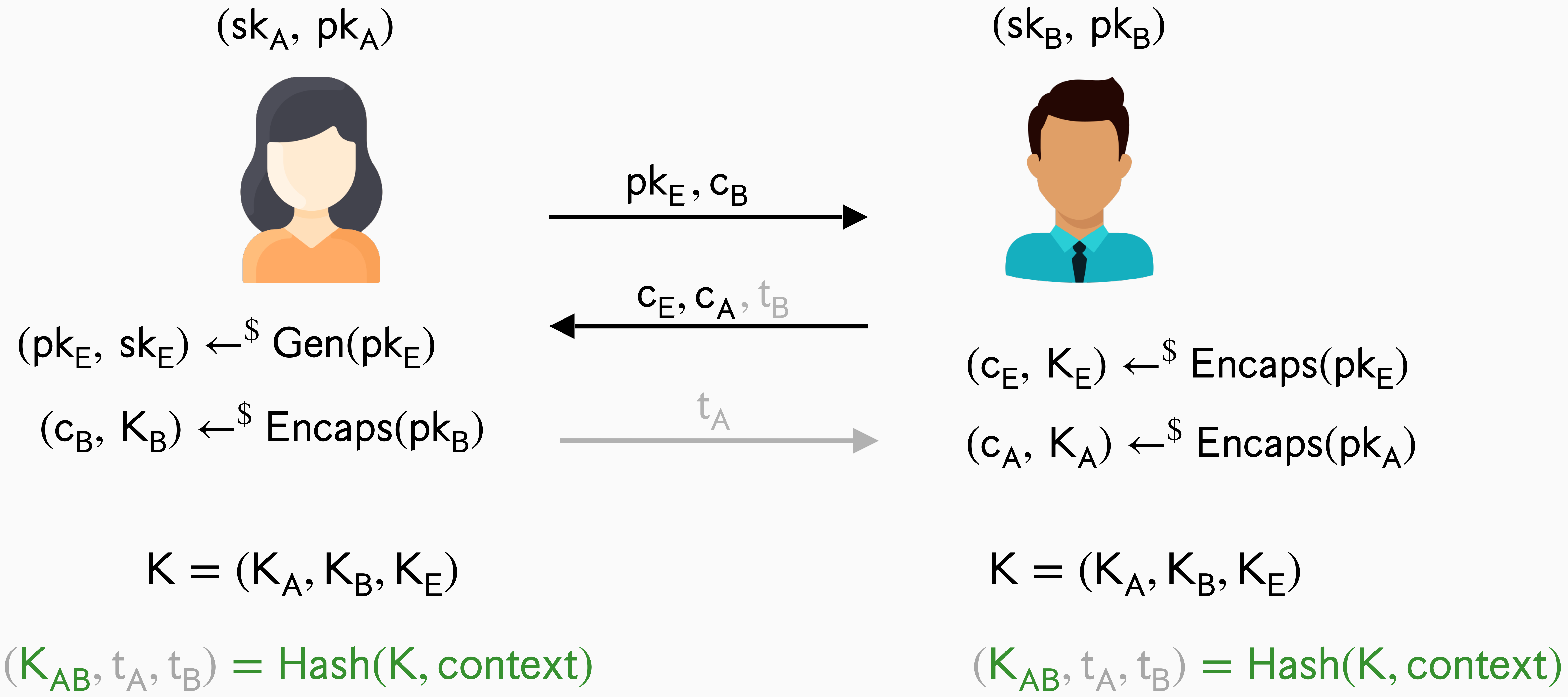
$(K_{AB}, t_A, t_B) = \mathsf{Hash}(K, \mathsf{context})$

Plaintext checking oracle allows to recognize queries which contain K

Security in the QROM

Previous Work [AC:PanWagZen23]

- (Almost) tight multi-user multi-challenge KEM from LWE in the QROM (via parameter lossy encryption)
- AKE-WFS with loss N



$(\mathsf{sk}_A, \mathsf{pk}_A)$

$(\mathsf{sk}_B, \mathsf{pk}_B)$

$\mathsf{pk}_E, \mathsf{c}_B$

$\mathsf{c}_E, \mathsf{c}_A, \mathsf{t}_B$

$(\mathsf{pk}_E, \mathsf{sk}_E) \xleftarrow{\$} \mathsf{Gen}(\mathsf{pk}_E)$

$(\mathsf{c}_B, \mathsf{K}_B) \xleftarrow{\$} \mathsf{Encaps}(\mathsf{pk}_B)$

t_A

$(\mathsf{c}_E, \mathsf{K}_E) \xleftarrow{\$} \mathsf{Encaps}(\mathsf{pk}_E)$

$(\mathsf{c}_A, \mathsf{K}_A) \xleftarrow{\$} \mathsf{Encaps}(\mathsf{pk}_A)$

$\mathsf{K} = (\mathsf{K}_A, \mathsf{K}_B, \mathsf{K}_E)$

$\mathsf{K} = (\mathsf{K}_A, \mathsf{K}_B, \mathsf{K}_E)$

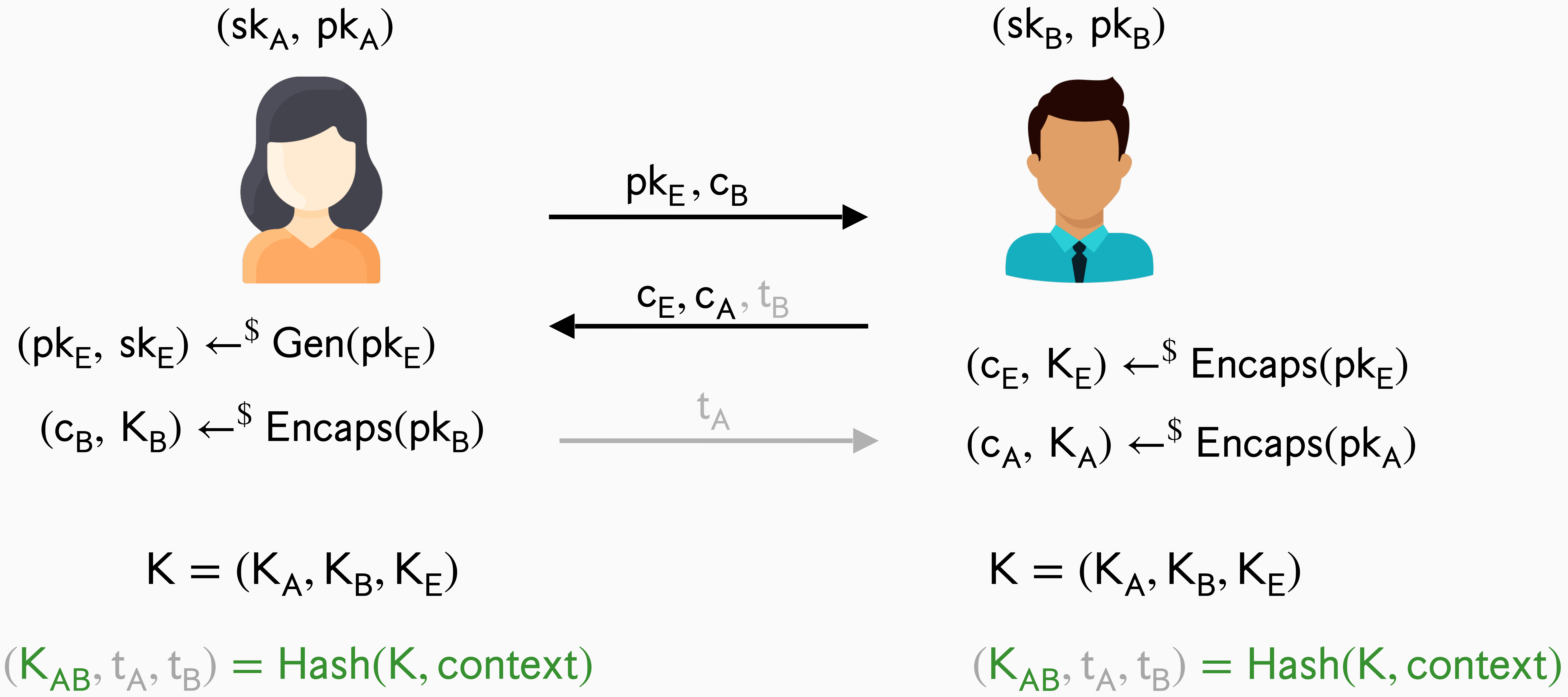
$(\mathsf{K}_{AB}, \mathsf{t}_A, \mathsf{t}_B) = \mathsf{Hash}(\mathsf{K}, \mathsf{context})$

$(\mathsf{K}_{AB}, \mathsf{t}_A, \mathsf{t}_B) = \mathsf{Hash}(\mathsf{K}, \mathsf{context})$

Security in the QROM

Previous Work [AC:PanWagZen23]

- (Almost) tight multi-user multi-challenge KEM from LWE in the QROM (via parameter lossy encryption)
- AKE-WFS with loss N



$(\mathsf{sk}_A, \mathsf{pk}_A)$

$(\mathsf{sk}_B, \mathsf{pk}_B)$

$\mathsf{pk}_E, \mathsf{c}_B$

$\mathsf{c}_E, \mathsf{c}_A, \mathsf{t}_B$

t_A

$(\mathsf{pk}_E, \mathsf{sk}_E) \xleftarrow{\$} \mathsf{Gen}(\mathsf{pk}_E)$

$(\mathsf{c}_B, \mathsf{K}_B) \xleftarrow{\$} \mathsf{Encaps}(\mathsf{pk}_B)$

$(\mathsf{c}_E, \mathsf{K}_E) \xleftarrow{\$} \mathsf{Encaps}(\mathsf{pk}_E)$

$(\mathsf{c}_A, \mathsf{K}_A) \xleftarrow{\$} \mathsf{Encaps}(\mathsf{pk}_A)$

$\mathsf{K} = (\mathsf{K}_A, \mathsf{K}_B, \mathsf{K}_E)$

$\mathsf{K} = (\mathsf{K}_A, \mathsf{K}_B, \mathsf{K}_E)$

$(\mathsf{K}_{AB}, \mathsf{t}_A, \mathsf{t}_B) = \mathsf{Hash}(\mathsf{K}, \mathsf{context})$

$(\mathsf{K}_{AB}, \mathsf{t}_A, \mathsf{t}_B) = \mathsf{Hash}(\mathsf{K}, \mathsf{context})$

We show: This bound can be preserved when adding key confirmation (achieving FFS)!

Conclusion

Our Results

- Tightly-secure AKE protocols with full forward secrecy via key confirmation (in the ROM)
- Modular approach using Verifiable AKE and KEMs which avoids the impossibility result of GGJJ
- Match QROM bounds of previous weak forward secure AKE

Conclusion

Our Results

- Tightly-secure AKE protocols with full forward secrecy via key confirmation (in the ROM)
- Modular approach using Verifiable AKE and KEMs which avoids the impossibility result of GGJJ
- Match QROM bounds of previous weak forward secure AKE

Open Questions

- Does the GGJJ impossibility also hold in the ROM?
- Tightly-secure protocols from search assumptions
- Better QROM bounds
  - via “quantum verifiability”?

Conclusion

Our Results

- Tightly-secure AKE protocols with full forward secrecy via key confirmation (in the ROM)
- Modular approach using Verifiable AKE and KEMs which avoids the impossibility result of GGJJ
- Match QROM bounds of previous weak forward secure AKE

Open Questions

- Does the GGJJ impossibility also hold in the ROM?
- Tightly-secure protocols from search assumptions
- Better QROM bounds
  - via "quantum verifiability"?

ia.cr/2024/361

Thank you!