Tightly Secure Two-Pass Authenticated Key Exchange Protocol in the CK Model

Yuting Xiao[1,2] Rui Zhang[1,2] Hui Ma[1]

[1]SKLOIS, IIE, CAS, China
[2]School of Cyber Security, UCAS, China

RSA Conference, Cryptographers' Track, 2020

Outline

• Motivations and the Problem •

Motivations

Authenticated Key Exchange (AKE)

1. An important primitive in the public key cryptography

   The basic building block of many secure communication protocols (e.g., TLS)

2. Are being widely used in daily life

3. Two evaluation factors: security & efficiency

Motivations

AKE in a large-scale setting:

There exist many (i.e., μ) users and each user may participate in multiple (i.e., ℓ) protocol execution instances.

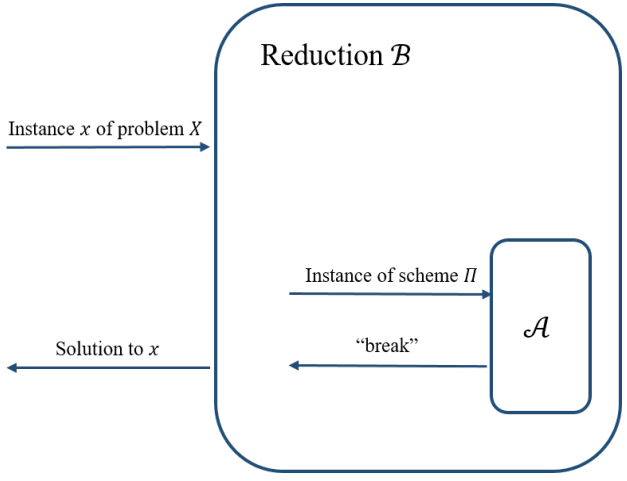
E.g., $\mu = \ell = 2^{30} \simeq 1,000,000,000$. (social networking sites.)

The Problem

1. Reduction-Based Security Proof:

- $\epsilon_{\mathcal{A}} \leq L \cdot \epsilon_{\mathcal{B}}$
- Tight Security

L is a small constant

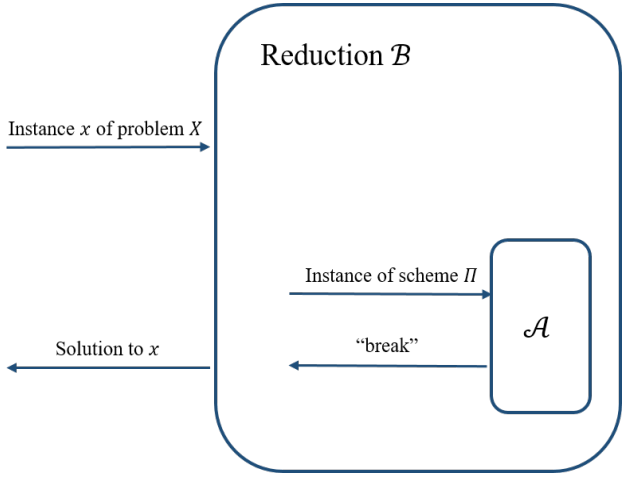


The Problem

1. Reduction-Based Security Proof:

   - $\epsilon_{\mathcal{A}} \leq L \cdot \epsilon_{\mathcal{B}}$
   - Tight Security

   L is a small constant



2. Most of AKEs come with a $1/\mu^2\ell^2$ or $1/\mu\ell$ loss factor

   The security degrades with the number of users or protocol execution instances.

The Problem

- Bader el al. [TCC 2015], Gjøsteen-Jager [CRYPTO 2018] achieved tight reduction

  They merely secure in the BR model

The Problem

- Bader el al. [TCC 2015], Gjøsteen-Jager [CRYPTO 2018] achieved tight reduction

  They merely secure in the BR model

- There exist more stronger security models

  CK, eCK, CK+, eCK-PFS .....

The Problem

- Bader el al. [TCC 2015], Gjøsteen-Jager [CRYPTO 2018] achieved tight reduction

  They merely secure in the BR model

- There exist more stronger security models

  CK, eCK, CK+, eCK-PFS .....

  How to achieve tight reduction in these models are still left as an open problem.

The Problem

- Bader el al. [TCC 2015], Gjøsteen-Jager [CRYPTO 2018] achieved tight reduction

  They merely secure in the BR model

- There exist more stronger security models

  CK, eCK, CK+, eCK-PFS .....

  How to achieve tight reduction in these models are still left as an open problem.

  We consider (almost) tight security in the CK model.

• The Main Challenges •

Basic notions for the CK model

- User identifiers: $P_1, \cdots, P_\mu$

Basic notions for the CK model

- User identifiers: $P_1, \cdots, P_\mu$
- Sessions: denote the view of a particular user in a protocol execution instance

Basic notions for the CK model

- User identifiers: $P_1, \cdots, P_\mu$
- Sessions: denote the view of a particular user in a protocol execution instance

  each session s is denoted as $(s_{actor}, s_{peer}, s_{role}, s_{sent}, s_{recv})$

Basic notions for the CK model

- User identifiers: $P_1, \cdots, P_\mu$
- Sessions: denote the view of a particular user in a protocol execution instance

  each session s is denoted as $(s_{actor}, s_{peer}, s_{role}, s_{sent}, s_{recv})$
- Matching-sessions: denote the two sessions involved in a single protocol execution instance

Basic notions for the CK model

- User identifiers: $P_1, \cdots, P_\mu$
- Sessions: denote the view of a particular user in a protocol execution instance

  each session s is denoted as $(s_{actor}, s_{peer}, s_{role}, s_{sent}, s_{recv})$
- Matching-sessions: denote the two sessions involved in a single protocol execution instance

Example (Two sessions s and s')

$s_{actor}=s'_{peer}$, $s_{peer}=s'_{actor}$, $s_{role}\neq s'_{role}$, $s_{sent}= s'_{recv}$ and $s_{recv}=s'_{sent}$

Allowed Queries

The CK model in the multi-bit challenge setting

- *establish-session* (P_i, P_j): passive attacks
- *incoming-message* (s, P_i, m): active attacks the adversary sends a message m to the session s in the name of P_i
- *corrupt* (P_i):
- *session-state reveal* (s): the intermediates stored for computing the session key
- *session-key reveal* (s):
- *test-session* (s): allowed for multiple times, but only on *completed*, *unexpired* and *unexposed* sessions
  $b_s \leftarrow_\$ \{0, 1\}$, the real key or a random key is returned

The Formal Security Definition

A session s is called *exposed* if $\mathcal{A}$ has performed:

- *session-state reveal* $(\cdot)$ or *session-key reveal* $(\cdot)$ on s
- (if the matching session s' exist)
  session-state reveal $(\cdot)$ or *session-key reveal* $(\cdot)$ on s'
- (if s' doesn't exist)
  corrupt $(\cdot)$ on the claimed owner of s' (KCI attacks)

The Formal Security Definition

A session s is called *exposed* if $\mathcal{A}$ has performed:

- *session-state reveal* $(\cdot)$ or *session-key reveal* $(\cdot)$ on s
- (if the matching session s' exist)
  session-state reveal $(\cdot)$ or *session-key reveal* $(\cdot)$ on s'
- (if s' doesn't exist)
  corrupt $(\cdot)$ on the claimed owner of s' (KCI attacks)

The Experiment (The challenger $\mathcal{C}$ and an adversary $\mathcal{A}$)

- $\mathcal{C}$ sends all public information to $\mathcal{A}$;
- $\mathcal{A}$ adaptively performs all allowed queries;
- $\mathcal{A}$ outputs a guess (s^*, b')

$\mathcal{A}$ wins the experiment if $b_{s^*} = b'$. Throughout the experiment, $\mathcal{A}$ is not allowed to expose s^*.

Abstraction of two-pass AKE protocols

Take two parties A and B as an example:

lsk_A

$msg_A(r_A) \longrightarrow$

$\longleftarrow msg_B(r_B)$

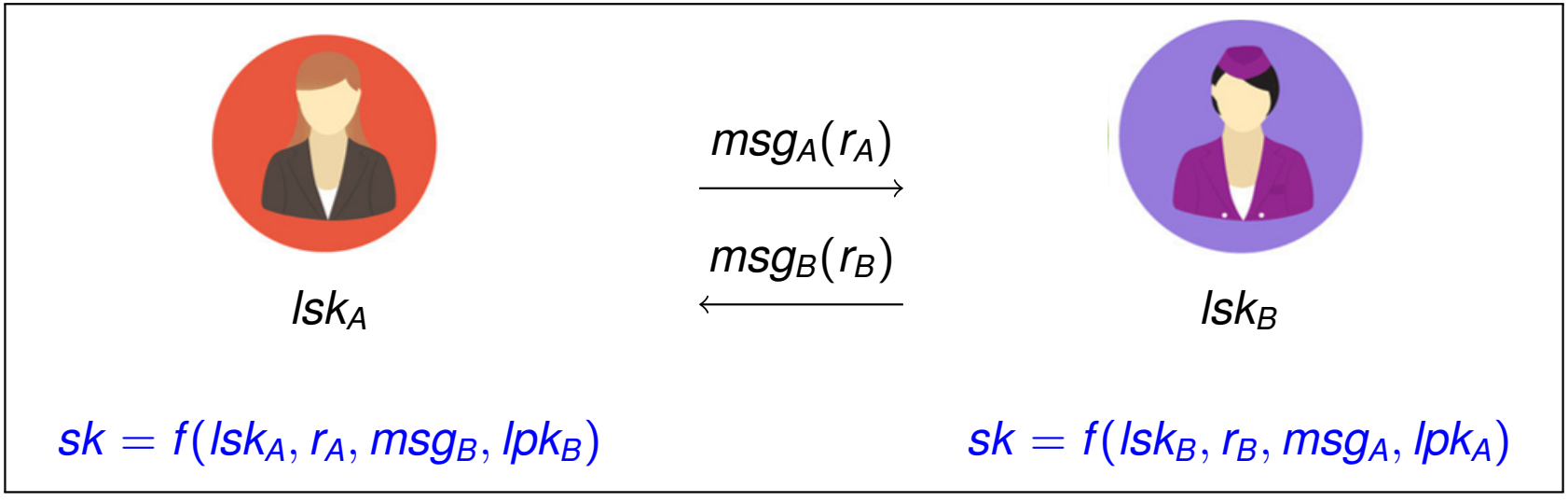
lsk_B

$sk = f(lsk_A, r_A, msg_B, lpk_B)$

$sk = f(lsk_B, r_B, msg_A, lpk_A)$

Abstraction of two-pass AKE protocols

Take two parties A and B as an example:



- We should embed challenges into the transmitted messages
- To get a tight reduction, we face two challenges

Guessing the target session

1. Guess the target session at the beginning of the experiment to embed the challenge

Parameters:

μ – the number of users

ℓ – the number of sessions per user $\Rightarrow$ reduction loss $L \simeq 1/\mu^2\ell^2$

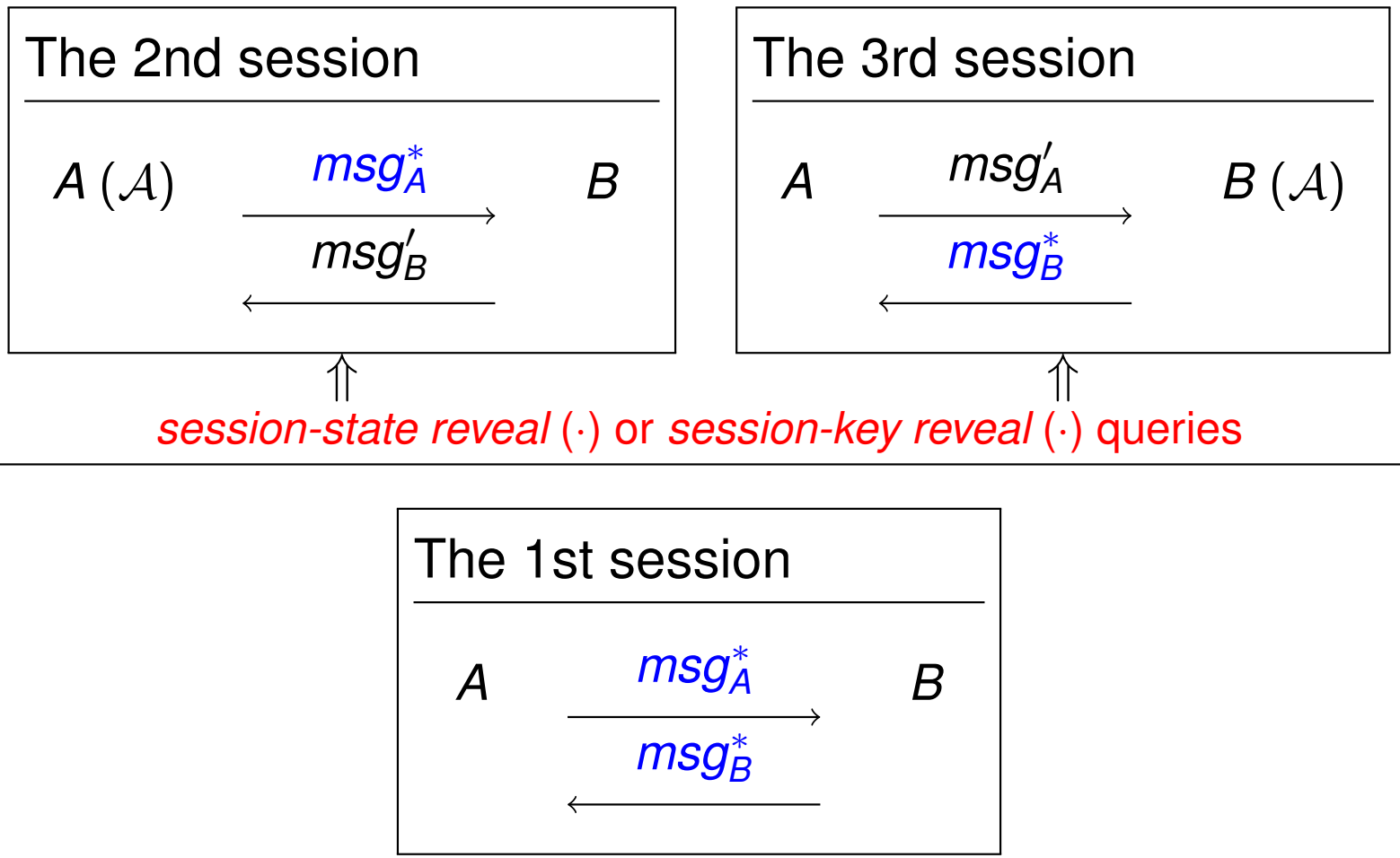
Sessions in a tangle

2. The embedded challenge in the target session may be unavoidably opened
   - $\mathcal{A}$ controls the communication channel
   - $\mathcal{A}$ is given a number of reveal queries

   !!! quite strong attack abilities

(An example in the next page)

Sessions in a tangle- example



Before $\mathcal{A}$ claiming its target, the (all) embedded challenges may have been opened.

• Our Solution •

The brief ideas

1. Primitives (KEMs) in the *multi-user* and *multi-challenge* setting with *corruption*

The brief ideas

1. Primitives (KEMs) in the *multi-user* and *multi-challenge* setting with *corruption*
    - embed challenge ciphertexts into simulated messages without guessing the target session

The brief ideas

1. Primitives (KEMs) in the *multi-user* and *multi-challenge* setting with *corruption*
   - embed challenge ciphertexts into simulated messages without guessing the target session
   - answer all corruption and reveal queries using provided *corruption* and *decryption* queries

The brief ideas

1. Primitives (KEMs) in the *multi-user* and *multi-challenge* setting with *corruption*
   - embed challenge ciphertexts into simulated messages without guessing the target session
   - answer all corruption and reveal queries using provided *corruption* and *decryption* queries

     note that embedded challenge ciphertexts are required to be opened in some cases

The brief ideas

1. Primitives (KEMs) in the *multi-user* and *multi-challenge* setting with *corruption*
   - embed challenge ciphertexts into simulated messages without guessing the target session
   - answer all corruption and reveal queries using provided *corruption* and *decryption* queries

     note that embedded challenge ciphertexts are required to be opened in some cases

2. CHK-like structure: OTS +TB-KEM

The brief ideas

1. Primitives (KEMs) in the *multi-user* and *multi-challenge* setting with *corruption*
   - embed challenge ciphertexts into simulated messages without guessing the target session
   - answer all corruption and reveal queries using provided *corruption* and *decryption* queries

   note that embedded challenge ciphertexts are required to be opened in some cases

2. CHK-like structure: OTS +TB-KEM

   the embedded challenge ciphertext of the final target session cannot be opened in any way

Building blocks -1

- IND-CPA wKEM in the *multi-user* and *multi-challenge* setting with *corruption*

Building blocks - 1

MU-IND-CPA$^{\text{Corr}}$ wKEM

$\Pi \leftarrow \text{Setup}(1^\lambda)$
$(ek_i, dk_i)_i \leftarrow \text{Gen}(\Pi)$

$ek_1, \cdots, ek_\ell \longrightarrow$

For the j-th $\mathcal{O}_E(i)$ query, a fresh coin $b_{i,j} \leftarrow\!\!\$\ \{0,1\}$ is used

$\mathcal{O}_C(i), \mathcal{O}_E(i) \longleftarrow$

$i^*, j^*, b' \longleftarrow$

Output 1 if $b' = b_{i^*,j^*}$ and $\mathcal{A}$ has not performed $\mathcal{O}_C(i^*)$ queries.

Building blocks - 2

- IND-CPA wKEM in the *multi-user* and *multi-challenge* setting with *corruption*
- IND-CCA TB-KEM in the *multi-user* and *multi-challenge* setting with *corruption* our new definition

  different coins are used to generate challenge ciphertexts and the adversary is allowed to open challenges

Building blocks - 2

MU-IND-CCA$^{\text{Corr}}$ TB-KEM

$\Pi \leftarrow \text{Setup}(1^\lambda)$

$(ek_i, dk_i)_i \leftarrow \text{Gen}(\Pi)$

$ek_1, \cdots, ek_\mu \longrightarrow$

$\mathcal{O}_C(i), \mathcal{O}_E(i, \tau_{i,j}), \mathcal{O}_D(i, c, \tau) \longleftarrow$

For each $\mathcal{O}_E(i, \tau_{i,j})$ query, a fresh coin $b_{i,j} \leftarrow\!\!\$\ \{0,1\}$ is used

$i^*, j^*, b' \longleftarrow$

Output 1 if $b' = b_{i^*,j^*}$ and $\mathcal{A}$ has never performed $\mathcal{O}_C(i^*)$ and $\mathcal{O}_D(i^*, c_{i^*,j^*}, \tau_{i^*,j^*})$ queries.

Building blocks - 3

- IND-CPA wKEM in the *multi-user* and *multi-challenge* setting with *corruption*
- IND-CCA TB-KEM in the *multi-user* and *multi-challenge* setting with *corruption* our new definition

  different coins are used to generate challenge ciphertexts and the adversary is allowed to open challenges
- EUF-CMA SIG in the *multi-user* setting with *corruption*

Building blocks - 3

MU-EUF-CMA$^{\text{Corr}}$ SIG

$\Pi \leftarrow \text{Setup}(1^\lambda)$
$(vk_i, sk_i)_i \leftarrow \text{Gen}(\Pi)$

$vk_1, \cdots, vk_\mu \longrightarrow$

$\mathcal{O}_C(i), \mathcal{O}_S(i, m) \longleftarrow$

$i^*, m^*, \sigma^* \longleftarrow$

Output 1 if (m^*, σ^*) is a valid signature under vk_{i^*}, and $\mathcal{A}$ hasn't performed $\mathcal{O}_C(i^*)$ and $\mathcal{O}_S(i^*, m^*)$ queries.

Building blocks - 4

- CPA wKEM in the *multi-user* and *multi-challenge* setting with *corruption*
- CCA TB-KEM in the *multi-user* and *multi-challenge* setting with *corruption* our new definition

  different coins are used to generate challenge ciphertexts and the adversary is allowed to open challenges
- EUF-CMA SIG in the *multi-user* setting with *corruption*
- sEUF-CMA OTS in the *multi-user* setting

Building blocks - 4

MU-sEUF-CMA OTS

$\Pi \leftarrow \text{Setup}(1^\lambda)$

$(vk_i, sk_i)_i \leftarrow \text{Gen}(\Pi)$

$vk_1, \cdots, vk_\ell$ →

$\mathcal{O}_S(i, m)$ ←

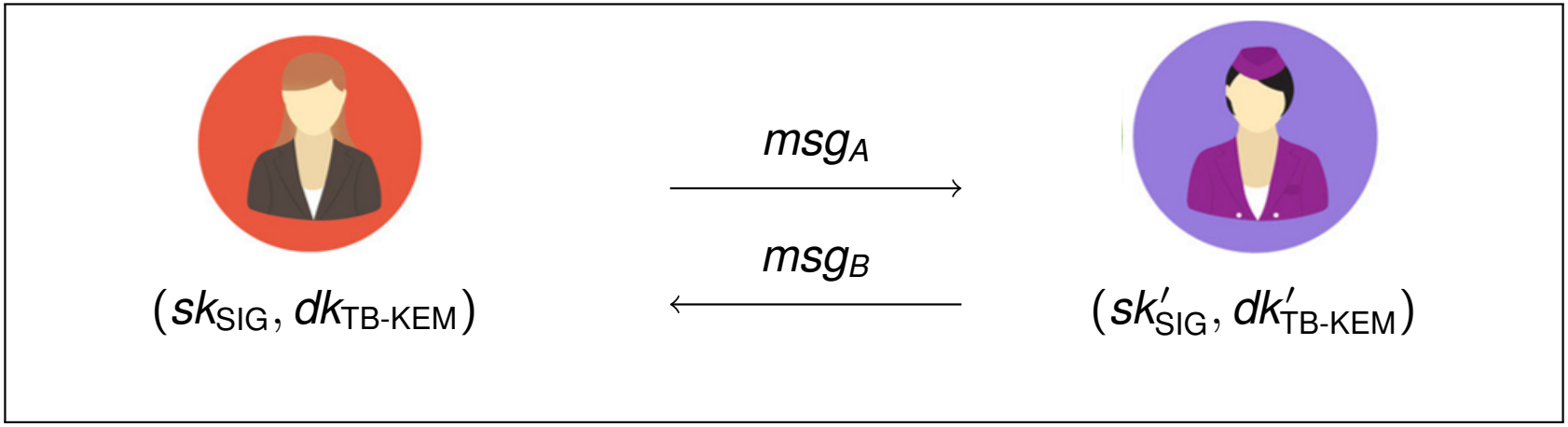
i^*, m^*, σ^* ←

For each i, $\mathcal{O}_S(i, \cdot)$ can only be asked for one time.

Output 1 if (m^*, σ^*) is a fresh valid signature under vk_{i^*}.

Our Construction

2SIG+2TB-KEM+2OTS+wKEM

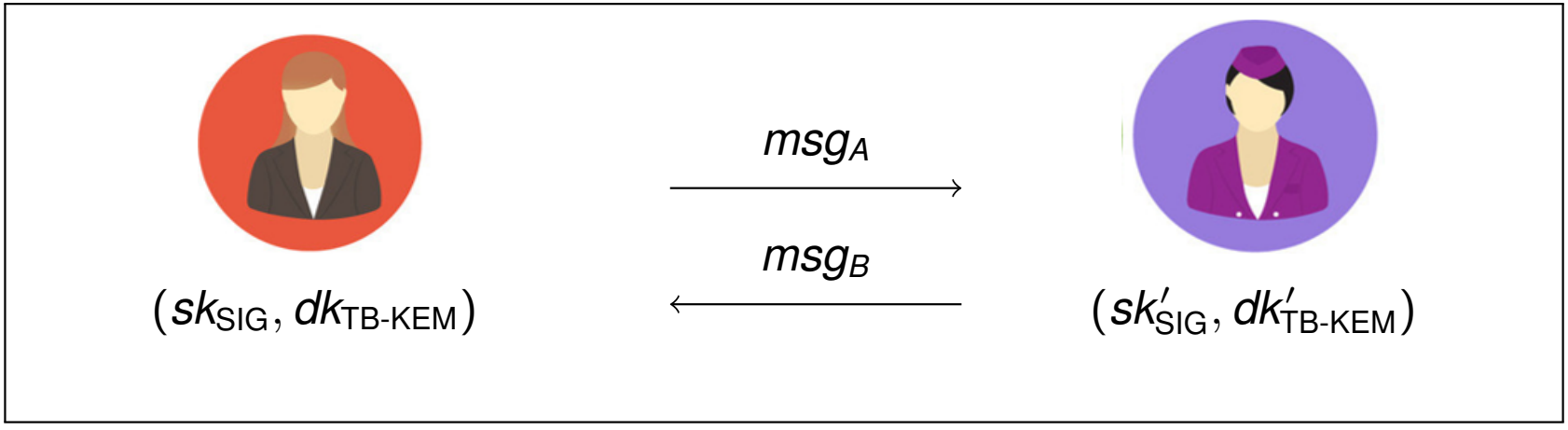


$$msg_A = (ek_{\text{wKEM}}, vk_{\text{OTS}}, c_{\text{TB-KEM}}, \sigma_{\text{OTS}}, \sigma_{\text{SIG}})$$

$$msg_B = (c_{\text{wKEM}}, vk'_{\text{OTS}}, c'_{\text{TB-KEM}}, \sigma'_{\text{OTS}}, \sigma'_{\text{SIG}})$$

Our Construction

Take two parties A and B as an example:



$$msg_A = (ek_{\text{wKEM}}, vk_{\text{OTS}}, c_{\text{TB-KEM}}, \sigma_{\text{OTS}}, \sigma_{\text{SIG}})$$

$$msg_B = (c_{\text{wKEM}}, vk'_{\text{OTS}}, c'_{\text{TB-KEM}}, \sigma'_{\text{OTS}}, \sigma'_{\text{SIG}})$$

$$sk = \text{PRF}(k_0, trans) \oplus \text{PRF}(k_1, trans) \oplus \text{PRF}(k_2, trans)$$

Proof ideas

Proof ideas

1. Passive adversaries
   - MU-IND-CPA$^{\text{Corr}}$ security of wKEM

Proof ideas

1. Passive adversaries
   - MU-IND-CPA$^{\text{Corr}}$ security of wKEM
2. Active adversaries
   - MU-EUF-CMA$^{\text{Corr}}$ security of SIG

   $\Rightarrow$ Adversaries can only launch replay attacks without knowing one's secret key.

   In particular, replaying initiation-messages.

Proof ideas

1. Passive adversaries
   - MU-IND-CPA$^{\text{Corr}}$ security of wKEM
2. Active adversaries
   - MU-EUF-CMA$^{\text{Corr}}$ security of SIG

     $\Rightarrow$ Adversaries can only launch replay attacks without knowing one's secret key.

     In particular, replaying initiation-messages.
   - MU-sEUF-CMA security of OTS

     $\Rightarrow$ each TB-KEM ciphertext is bind with a fresh tag

Proof ideas

1. Passive adversaries
   - MU-IND-CPA$^{\text{Corr}}$ security of wKEM
2. Active adversaries
   - MU-EUF-CMA$^{\text{Corr}}$ security of SIG

     $\Rightarrow$ Adversaries can only launch replay attacks without knowing one's secret key.

     In particular, replaying initiation-messages.
   - MU-sEUF-CMA security of OTS

     $\Rightarrow$ each TB-KEM ciphertext is bind with a fresh tag
   - MU-IND-CCA$^{\text{Corr}}$ security of TB-KEM

     $\Rightarrow$ the TB-KEM challenge embedded in the reply-message of the target session was unopened

How to achieve MU-IND-CCA$^{\text{Corr}}$ security of TB-KEM?

- The Naor-Yung Transform (double encryption)

TB-KEM.Setup(1^λ):	TB-KEM.Enc($\hat{ek}, \tau$):
$\Pi \leftarrow$ PKE.Setup(1^λ)	**phrase** $\hat{ek} = (ek_0, ek_1, \psi)$
$\Gamma \leftarrow$ QA-NIZK.K$_0$(1^λ)	$k \leftarrow\$ \mathcal{M}$, $c_0 \leftarrow$ PKE.Enc($ek_0, k; r_0$)
return $\hat{\Pi} = (\Pi, \Gamma)$	$c_1 \leftarrow$ PKE.Enc($ek_1, k; r_1$), lbl $= (c_0, c_1, \tau)$
TB-KEM.Gen($\hat{\Pi}$):	$\pi \leftarrow$ QA-NIZK.P($\psi, (c_0, c_1), (r_0, r_1)$, lbl)
phrase $\hat{\Pi} = (\Pi, \Gamma)$, $\delta \leftarrow\$ \{0,1\}$	**return** $(\hat{c} = (c_0, c_1, \pi), \hat{k} = k)$
$(ek_0, dk_0) \leftarrow$ PKE.Gen(Π)	TB-KEM.Dec($\hat{dk}, \hat{c}, \tau$):
$(ek_1, dk_1) \leftarrow$ PKE.Gen(Π)	**phrase** $\hat{dk} = (\delta, dk_\delta)$, $\hat{c} = (c_0, c_1, \pi)$
$\rho = (ek_0, ek_1)$	lbl $= (c_0, c_1, \tau)$
$\psi \leftarrow$ QA-NIZK.K$_1$(Γ, ρ)	$b =$ QA-NIZK.V($\psi, (c_0, c_1), \pi$, lbl)
return	$k \leftarrow$ PKE.Dec($\hat{dk}_\delta, c_\delta$)
$\hat{ek} = (ek_0, ek_1, \psi)$,$\hat{dk} = (\delta, dk_\delta)$	**if** $b = 1$ **return** $\hat{k} = k$, **else return** $\perp$

• Summary •

Summary

- Tight reduction in the CK model was an open problem

Summary

- Tight reduction in the CK model was an open problem
- Why it is difficult?
  - have to guess the target session
  - non-matching sessions of the target session may cause its embedded challenge being opened

Summary

- Tight reduction in the CK model was an open problem
- Why it is difficult?
  - have to guess the target session
  - non-matching sessions of the target session may cause its embedded challenge being opened
- Our construction – 2SIG+2TB-KEM+2OTS+wKEM
  - in the multi-user setting
  - TB-KEM - new definition

Summary

- Tight reduction in the CK model was an open problem
- Why it is difficult?
  - have to guess the target session
  - non-matching sessions of the target session may cause its embedded challenge being opened
- Our construction – 2SIG+2TB-KEM+2OTS+wKEM
  - in the multi-user setting
  - TB-KEM - new definition
- Naor-Yung transform meets our new definition for TB-KEM

THANK YOU ! ! !