Tightly-Secure Authenticated Key Exchange, Revisited

Tibor Jager[1], Eike Kiltz[2], Doreen Riepel[2], Sven Schäge[2]

October 14, 2021

[1]Bergische Universität Wuppertal
[2]Ruhr-Universität Bochum

Introduction

Authenticated Key Exchange (AKE)

- Used to establish a shared session key between two parties
- One of the most widely-used cryptographic primitives, e.g. TLS

Outline

- Security model and tightness
- Comparison to previous work
- AKE from key encapsulation mechanisms (KEMs)
- Security requirements for KEM

Two-Message Authenticated Key Exchange

Alice

(sk_A, pk_A)

Bob

(sk_B, pk_B)

Two-Message Authenticated Key Exchange

st_A

Alice

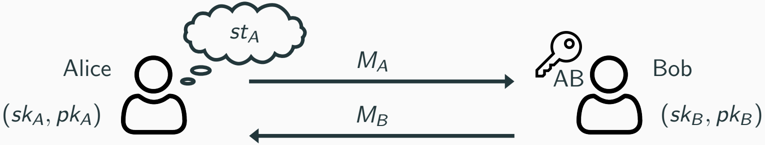
(sk_A, pk_A)

M_A

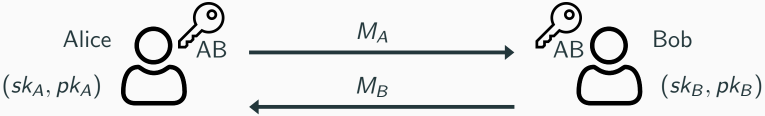
Bob

(sk_B, pk_B)

Two-Message Authenticated Key Exchange



Two-Message Authenticated Key Exchange



Alice

(sk_A, pk_A)

AB

M_A

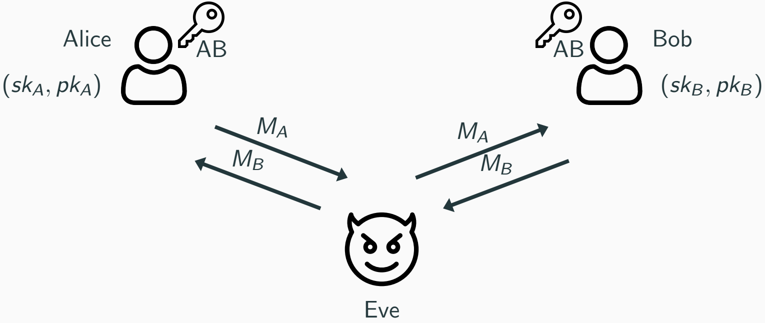
M_B

AB

Bob

(sk_B, pk_B)

Two-Message Authenticated Key Exchange



Alice

AB

(sk_A, pk_A)

Bob

AB

(sk_B, pk_B)

M_A

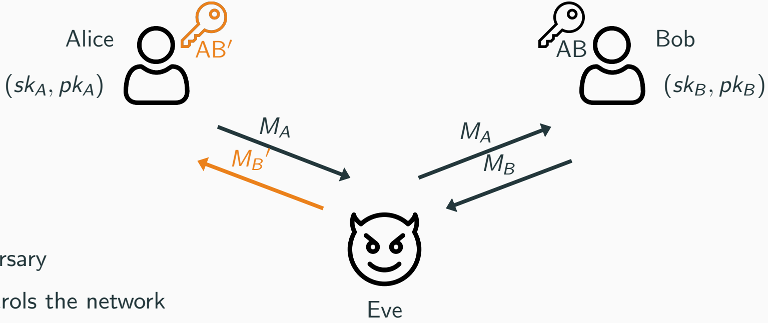
M_B

M_A

M_B

Eve

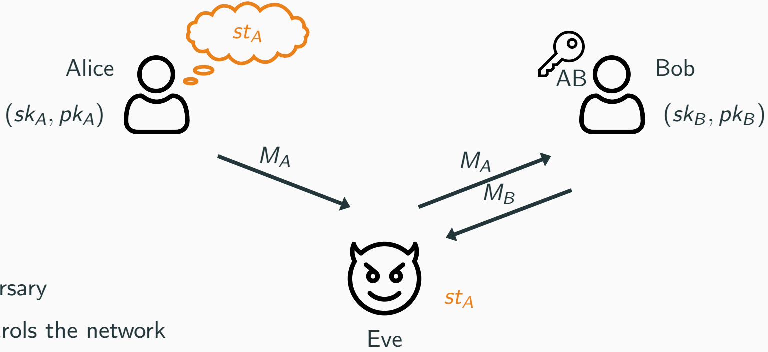
Two-Message Authenticated Key Exchange



The adversary

- controls the network

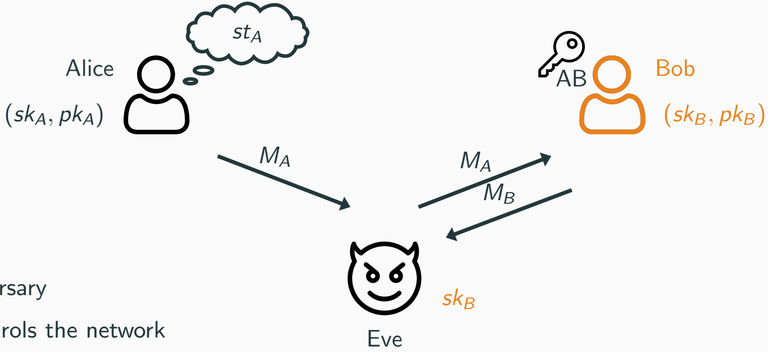
Two-Message Authenticated Key Exchange



The adversary

- controls the network
- reveals secret states

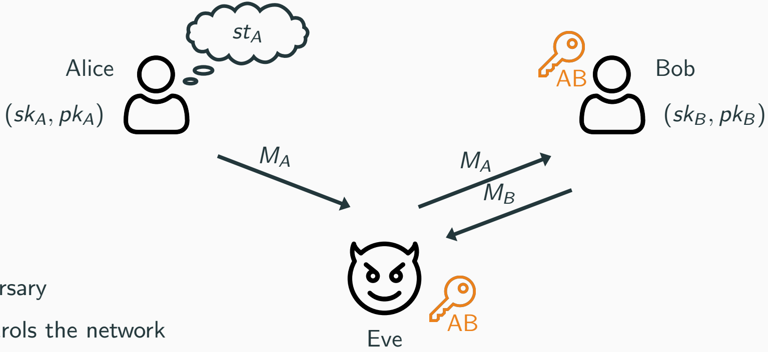
Two-Message Authenticated Key Exchange



The adversary

- controls the network
- reveals secret states
- adaptively corrupts long-term keys

Two-Message Authenticated Key Exchange

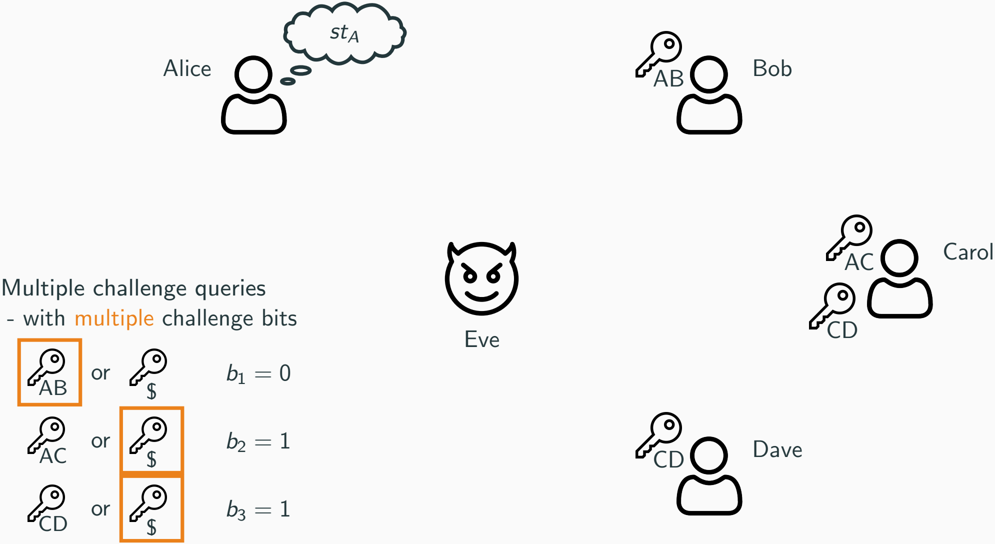


The adversary

- controls the network
- reveals secret states
- adaptively corrupts long-term keys
- reveals real session keys

Two-Message Authenticated Key Exchange

Two-Message Authenticated Key Exchange



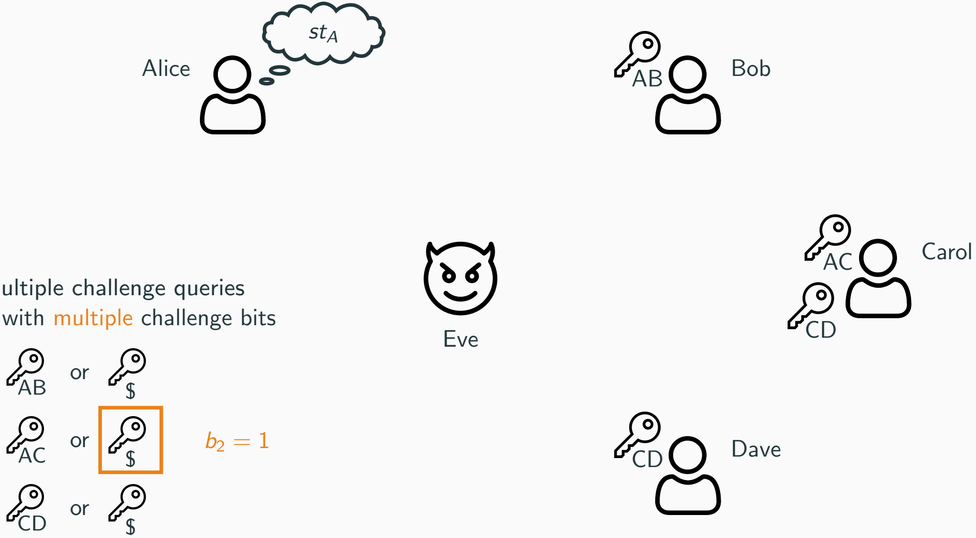
Multiple challenge queries
- with multiple challenge bits

AB or \$ $b_1 = 0$

AC or \$ $b_2 = 1$

CD or \$ $b_3 = 1$

Two-Message Authenticated Key Exchange



Multiple challenge queries

- with multiple challenge bits

$b_2 = 1$

Two-Message Authenticated Key Exchange

Multiple challenge queries
- with a single challenge bit

$b = 0$

Two-Message Authenticated Key Exchange

Multiple challenge queries
- with a single challenge bit

$b = 1$

AKE Security

Single challenge bit

- Well-established notion for multi-challenge security definitions
- Equivalent to “Real-or-Random” security
- Tightly composes with symmetric primitives

Security properties

- Forward secrecy
- Resistance against key compromise impersonation attacks
- Resistance against maximal exposure attacks

Provable Security

Security is modelled as a game between a challenger and an adversary.

Security reduction

- We turn adversary an $\mathcal{A}$ against the scheme into an adversary $\mathcal{B}$ that solves a computationally hard problem.

A reduction is called *tight* if $\mathcal{A}$ and $\mathcal{B}$

- have about the same advantage.
- run in about the same time.

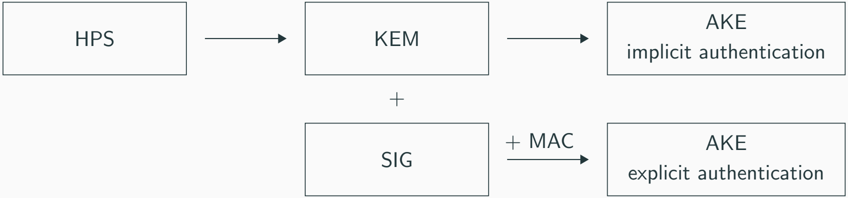
Relevance: tells us how to choose system parameters

Comparison with Previous Work

	Standard Model	Tight Proof	Ephemeral State Reveal	Single Challenge Bit
BHJKL15	✓	✓	✗	✗
GJ18	✗	✓	✗	✗
CCGJJ19	✗	✗	✗	✓
LLGW20	✓	✓	✗	✗
This work	✗	✓	✓	✓

Our AKE Protocols

Generic Construction



HPS

KEM

AKE

implicit authentication

+

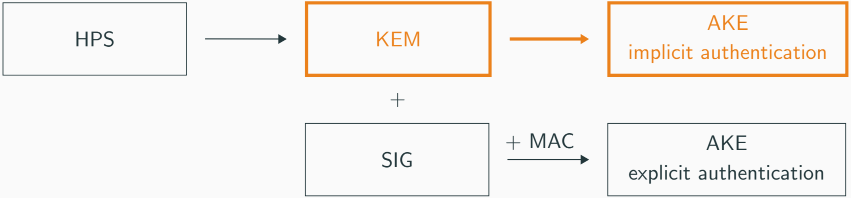
SIG

+ MAC

AKE

explicit authentication

Generic Construction



HPS

KEM

AKE
implicit authentication

+

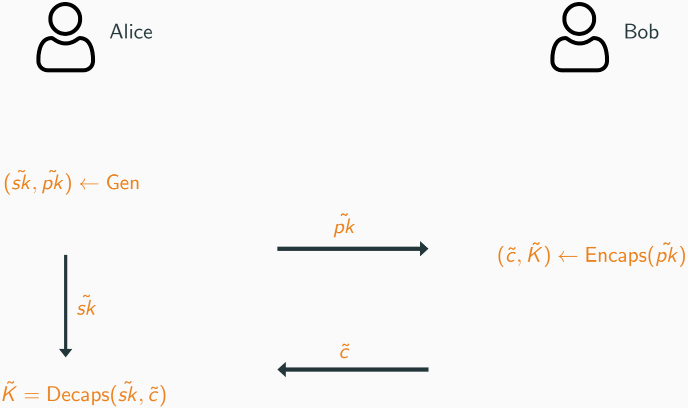
SIG

+ MAC

AKE
explicit authentication

Main question: What security properties do we need for the KEM to achieve tightness?

AKE[KEM]



Alice

Bob

$(\tilde{sk}, \tilde{pk}) \leftarrow \mathsf{Gen}$

$\tilde{pk}$

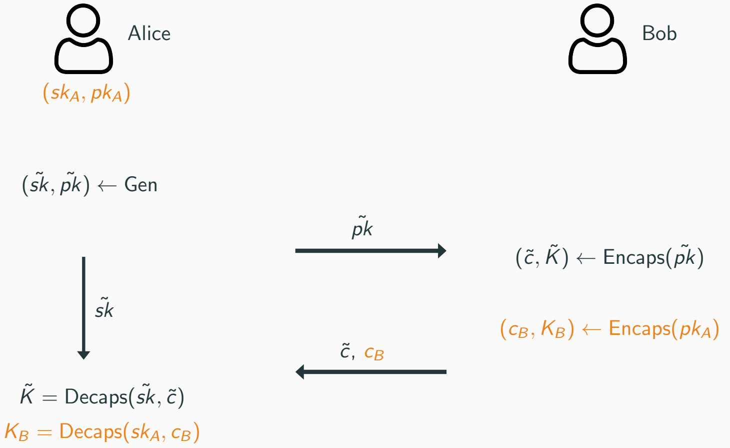
$(\tilde{c}, \tilde{K}) \leftarrow \mathsf{Encaps}(\tilde{pk})$

$\tilde{sk}$

$\tilde{c}$

$\tilde{K} = \mathsf{Decaps}(\tilde{sk}, \tilde{c})$

AKE[KEM]



Alice

(sk_A, pk_A)

Bob

$(\tilde{sk}, \tilde{pk}) \leftarrow \text{Gen}$

$\tilde{pk}$

$(\tilde{c}, \tilde{K}) \leftarrow \text{Encaps}(\tilde{pk})$

$\tilde{sk}$

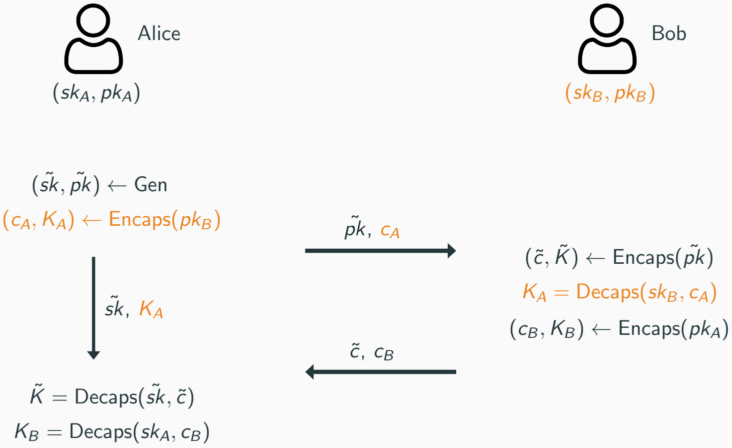
$(c_B, K_B) \leftarrow \text{Encaps}(pk_A)$

$\tilde{c}, \; c_B$

$\tilde{K} = \text{Decaps}(\tilde{sk}, \tilde{c})$

$K_B = \text{Decaps}(sk_A, c_B)$

AKE[KEM]



Alice

(sk_A, pk_A)

Bob

(sk_B, pk_B)

$(\tilde{sk}, \tilde{pk}) \leftarrow \mathsf{Gen}$

$(c_A, K_A) \leftarrow \mathsf{Encaps}(pk_B)$

$\tilde{pk},\ c_A$

$(\tilde{c}, \tilde{K}) \leftarrow \mathsf{Encaps}(\tilde{pk})$

$K_A = \mathsf{Decaps}(sk_B, c_A)$

$(c_B, K_B) \leftarrow \mathsf{Encaps}(pk_A)$

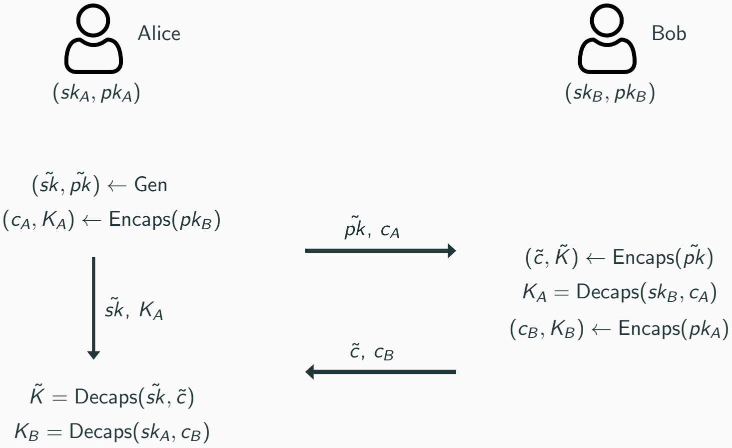
$\tilde{sk},\ K_A$

$\tilde{c},\ c_B$

$\tilde{K} = \mathsf{Decaps}(\tilde{sk}, \tilde{c})$

$K_B = \mathsf{Decaps}(sk_A, c_B)$

AKE[KEM]



Alice

(sk_A, pk_A)

Bob

(sk_B, pk_B)

$(\tilde{sk}, \tilde{pk}) \leftarrow \mathsf{Gen}$

$(c_A, K_A) \leftarrow \mathsf{Encaps}(pk_B)$

$\tilde{pk},\ c_A$

$\tilde{sk},\ K_A$

$(\tilde{c}, \tilde{K}) \leftarrow \mathsf{Encaps}(\tilde{pk})$

$K_A = \mathsf{Decaps}(sk_B, c_A)$

$(c_B, K_B) \leftarrow \mathsf{Encaps}(pk_A)$

$\tilde{c},\ c_B$

$\tilde{K} = \mathsf{Decaps}(\tilde{sk}, \tilde{c})$

$K_B = \mathsf{Decaps}(sk_A, c_B)$

$$K = \mathsf{H}(\mathsf{A}, \mathsf{B}, pk_A, pk_B, \tilde{pk}, c_A, c_B, \tilde{c}, K_A, K_B, \tilde{K})$$

Security Requirements for KEM

Corruption queries

- Need to output long-term secret keys adaptively (sk_A, sk_B)
- Previously sent ciphertexts must decrypt correctly

Non-Committing
Key Encapsulation

Reveal-key vs. challenge queries

- Ciphertexts may be revealed or used in challenges
- Need to decrypt ciphertexts coming from the adversary

State reveals

- Need to output ephemeral secret keys $(\tilde{sk})$
- Key indistinguishability even when state is compromised

Non-Committing Key Encapsulation from DDH

Public parameter: group description $(\mathbb{G}, p, g)$ and $h = g^{\omega}$ for $\omega \xleftarrow{\$} \mathbb{Z}_p$

<u>KeyGen</u>

$sk = (x_0, x_1) \xleftarrow{\$} \mathbb{Z}_p^2$

$pk = g^{x_0} h^{x_1}$

<u>Decaps(sk, c_0, c_1)</u>

$K = \mathsf{H}(pk, c, c_0^{x_0} c_1^{x_1})$

<u>Encaps(pk)</u>

$r \xleftarrow{\$} \mathbb{Z}_p$

$(c_0, c_1) = (g^r, h^r)$

$K = \mathsf{H}(pk, c_0, c_1, pk^r)$

<u>SimEncaps(sk)</u>

$(r, s) \xleftarrow{\$} \mathbb{Z}_p^2$

$(c_0, c_1) = (g^r, h^s)$

$K = \mathsf{H}(pk, c_0, c_1, c_0^{x_0} c_1^{x_1})$

Non-Committing Key Encapsulation from DDH

Public parameter: group description $(\mathbb{G}, p, g)$ and $h = g^{\omega}$ for $\omega \xleftarrow{\$} \mathbb{Z}_p$

KeyGen

$sk = (x_0, x_1) \xleftarrow{\$} \mathbb{Z}_p^2$

$pk = g^{x_0} h^{x_1}$

$\mathsf{Decaps}(sk, c_0, c_1)$

$K = \mathsf{H}(pk, c, c_0^{x_0} c_1^{x_1})$

Encaps(pk)

$r \xleftarrow{\$} \mathbb{Z}_p$

$(c_0, c_1) = (g^r, h^r)$

$K = \mathsf{H}(pk, c_0, c_1, pk^r)$

$(c_0, c_1) \in \mathcal{L}_{\mathsf{DDH}}$

SimEncaps(sk)

$(r, s) \xleftarrow{\$} \mathbb{Z}_p^2$

$(c_0, c_1) = (g^r, h^s)$

$K = \mathsf{H}(pk, c_0, c_1, c_0^{x_0} c_1^{x_1})$

$(c_0, c_1) \notin \mathcal{L}_{\mathsf{DDH}}$

Properties

- Encaps $\approx_c$ SimEncaps
- Holds even given sk

Non-Committing Key Encapsulation from DDH

Public parameter: group description $(\mathbb{G}, p, g)$ and $h = g^{\omega}$ for $\omega \xleftarrow{\$} \mathbb{Z}_p$

KeyGen

$sk = (x_0, x_1) \xleftarrow{\$} \mathbb{Z}_p^2$

$pk = g^{x_0} h^{x_1}$

Encaps(pk)

$r \xleftarrow{\$} \mathbb{Z}_p$

$(c_0, c_1) = (g^r, h^r)$

$K = \mathsf{H}(pk, c_0, c_1, pk^r)$

$(c_0, c_1) \in \mathcal{L}_{\mathsf{DDH}}$

Decaps(sk, c_0, c_1)

$K = \mathsf{H}(pk, c, c_0^{x_0} c_1^{x_1})$

SimEncaps(sk)

$(r, s) \xleftarrow{\$} \mathbb{Z}_p^2$

$(c_0, c_1) = (g^r, h^s)$

$K = \mathsf{H}(pk, c_0, c_1, c_0^{x_0} c_1^{x_1})$

$(c_0, c_1) \notin \mathcal{L}_{\mathsf{DDH}}$

Properties

- Encaps $\approx_c$ SimEncaps
- Holds even given *sk*
- Without knowledge of *sk*: $K \approx_s \$$

Summary

Contributions

- Non-committing key encapsulation from hash proof systems in the ROM
- Two tightly-secure AKE protocols
    - with state reveals
    - with a single challenge bit

ePrint: `ia.cr/2020/1279`

Follow-Up Work

- Tightly-secure AKE and signatures in the standard model (CRYPTO 2021)

Thank you!