Authenticated Key Exchange and Signatures with Tight Security in the Standard Model

Shuai Han[1], Tibor Jager[3], Eike Kiltz[2], Shengli Liu[1], Jiaxin Pan[4], Doreen Riepel[1], Sven Schäge[1]

October 8, 2021

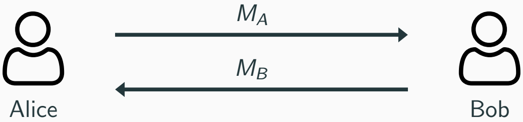
[1]Shanghai Jiao Tong University
[2]Ruhr-Universität Bochum
[3]Bergische Universität Wuppertal
[4]Norwegian University of Science and Technology

Authenticated Key Exchange

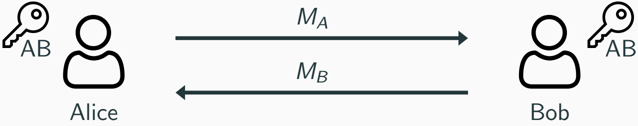
Alice

Bob

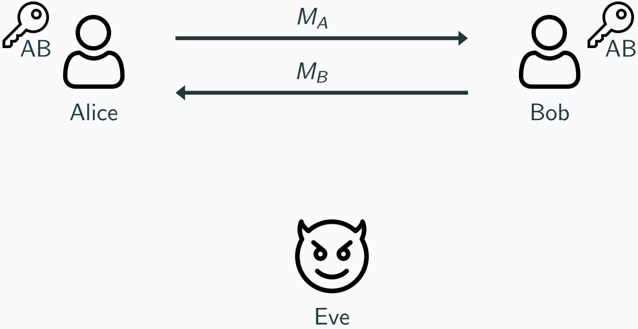
Authenticated Key Exchange



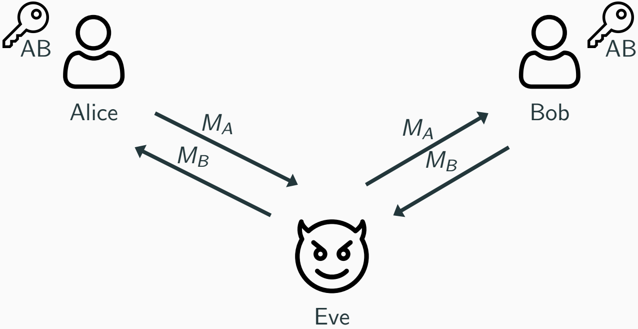
Authenticated Key Exchange



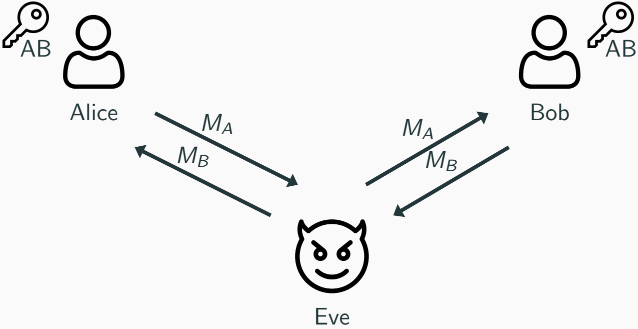
Authenticated Key Exchange



Authenticated Key Exchange



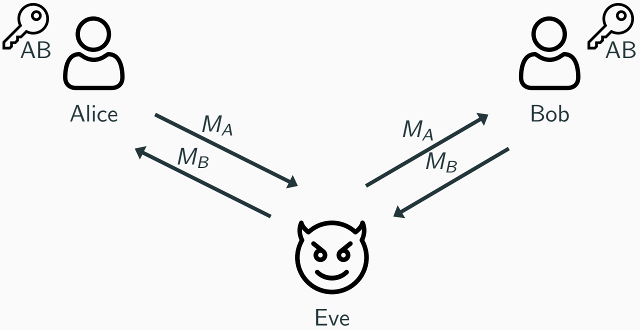
Authenticated Key Exchange



The adversary

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

Authenticated Key Exchange



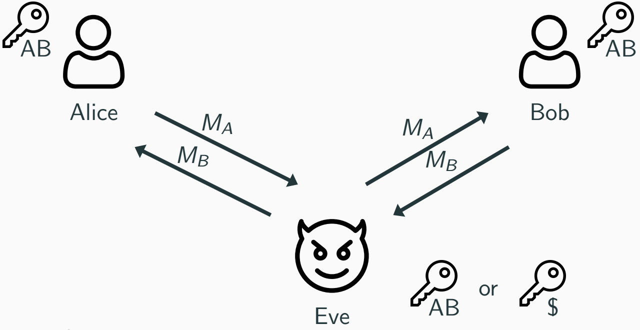
The adversary

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

Security Goals

- Authenticity
- Key Indistinguishability

Authenticated Key Exchange



The adversary

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

Security Goals

- Authenticity
- Key Indistinguishability

Provable Security

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

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.

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.

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

Difficulties in Proving Tight AKE

The commitment problem

- Need to be able to answer *key-reveal* and *test* queries for all sessions
- Need to avoid guessing the test session(s)

Difficulties in Proving Tight AKE

The commitment problem

- Need to be able to answer *key-reveal* and *test* queries for all sessions
- Need to avoid guessing the test session(s)

Long-term key reveals and tightly-secure signatures

- Signatures to achieve explicit authentication
- Need to answer adaptive *corrupt* queries and output secret signing keys
- At the same time: extract the solution to a hard problem from a signature forgery

Comparison with Previous Work

	Efficient	Standard Model	Tight Proof	Ephemeral State Reveal
BHJKL15				
GJ18				
CCGJJ19				
LLGW20				
JKRS21				
This work				

Comparison with Previous Work

	Efficient	Standard Model	Tight Proof	Ephemeral State Reveal
BHJKL15	✗	✓	✓	✗
GJ18				
CCGJJ19				
LLGW20				
JKRS21				
This work				

Comparison with Previous Work

	Efficient	Standard	Tight	Ephemeral
		Model	Proof	State Reveal
BHJKL15	✗	✓	✓	✗
GJ18	(✓)	✗	✓	✗
CCGJJ19				
LLGW20				
JKRS21				
This work				

Comparison with Previous Work

	Efficient	Standard	Tight	Ephemeral
		Model	Proof	State Reveal
BHJKL15	✗	✓	✓	✗
GJ18	(✓)	✗	✓	✗
CCGJJ19	✓	✗	✗	✗
LLGW20				
JKRS21				
This work				

Comparison with Previous Work

	Efficient	Standard	Tight	Ephemeral
		Model	Proof	State Reveal
BHJKL15	✗	✓	✓	✗
GJ18	(✓)	✗	✓	✗
CCGJJ19	✓	✗	✗	✗
LLGW20	(✗)	✓	✓	✗
JKRS21				
This work				

Comparison with Previous Work

	Efficient	Standard Model	Tight Proof	Ephemeral State Reveal
BHJKL15	✗	✓	✓	✗
GJ18	(✓)	✗	✓	✗
CCGJJ19	✓	✗	✗	✗
LLGW20	(✗)	✓	✓	✗
JKRS21	(✓)	✗	✓	✓
This work				

Comparison with Previous Work

	Efficient	Standard Model	Tight Proof	Ephemeral State Reveal
BHJKL15	✗	✓	✓	✗
GJ18	(✓)	✗	✓	✗
CCGJJ19	✓	✗	✗	✗
LLGW20	(✗)	✓	✓	✗
JKRS21	(✓)	✗	✓	✓
This work	(✓)	✓	(✓)*	✓

*Non-tight only with respect to a symmetric primitive when allowing state reveals

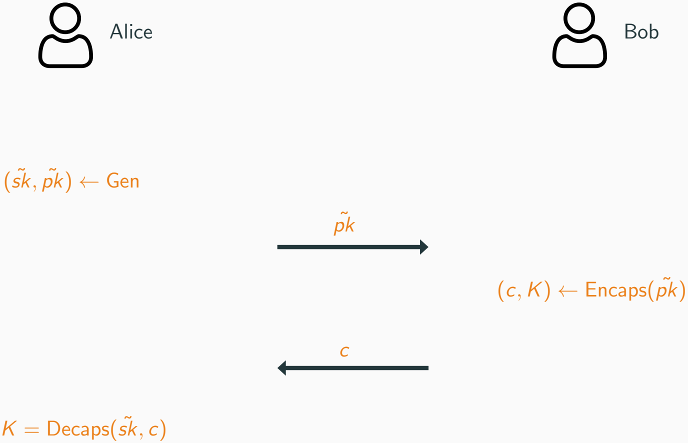
Our AKE Protocol

AKE

Alice

Bob

AKE[KEM]



Alice

Bob

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

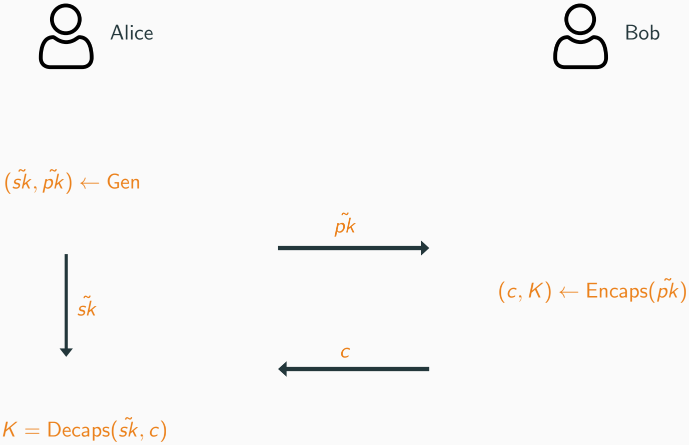
$\tilde{pk}$

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

c

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

AKE[KEM]



Alice

Bob

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

$\tilde{pk}$

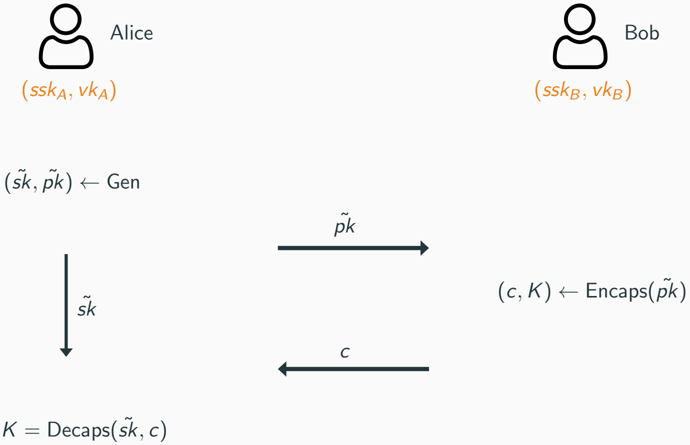
$\tilde{sk}$

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

c

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

AKE[KEM, SIG]



Alice

(ssk_A, vk_A)

Bob

(ssk_B, vk_B)

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

$\tilde{pk}$

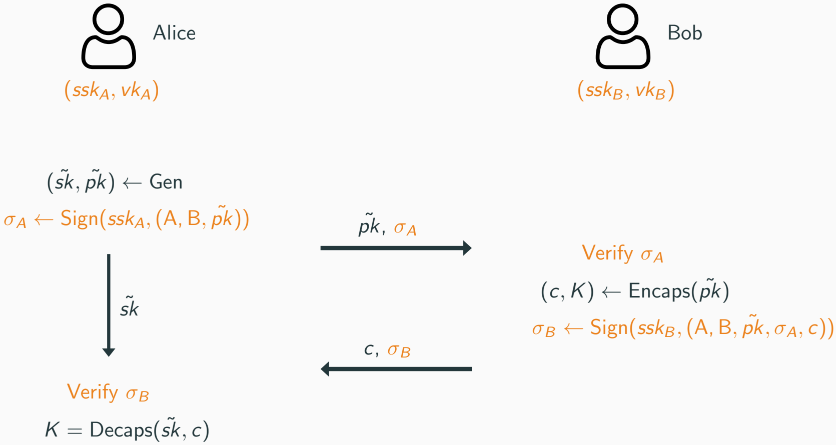
$\tilde{sk}$

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

c

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

AKE[KEM, SIG]



Alice

(ssk_A, vk_A)

Bob

(ssk_B, vk_B)

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

$\sigma_A \leftarrow \mathsf{Sign}(ssk_A, (\mathsf{A}, \mathsf{B}, \tilde{pk}))$

$\tilde{pk}, \sigma_A$

$\tilde{sk}$

Verify σ_A

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

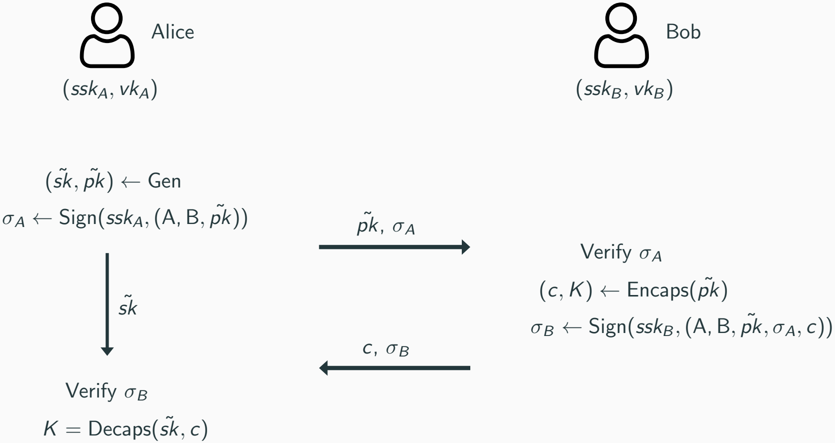
$\sigma_B \leftarrow \mathsf{Sign}(ssk_B, (\mathsf{A}, \mathsf{B}, \tilde{pk}, \sigma_A, c))$

c, σ_B

Verify σ_B

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

AKE[KEM, SIG]



Alice

(ssk_A, vk_A)

Bob

(ssk_B, vk_B)

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

$\sigma_A \leftarrow \mathsf{Sign}(ssk_A, (\mathsf{A}, \mathsf{B}, \tilde{pk}))$

$\tilde{pk},\ \sigma_A$

$\tilde{sk}$

Verify σ_A

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

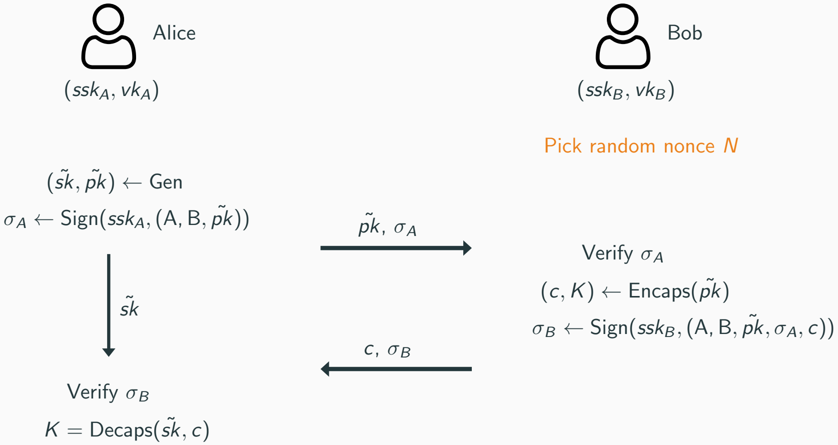
$\sigma_B \leftarrow \mathsf{Sign}(ssk_B, (\mathsf{A}, \mathsf{B}, \tilde{pk}, \sigma_A, c))$

$c,\ \sigma_B$

Verify σ_B

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

AKE[KEM, SIG, Nonce]



Alice

(ssk_A, vk_A)

Bob

(ssk_B, vk_B)

Pick random nonce N

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

$\sigma_A \leftarrow \mathrm{Sign}(ssk_A, (\mathrm{A}, \mathrm{B}, \tilde{pk}))$

$\tilde{pk},\ \sigma_A$

$\tilde{sk}$

Verify σ_A

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

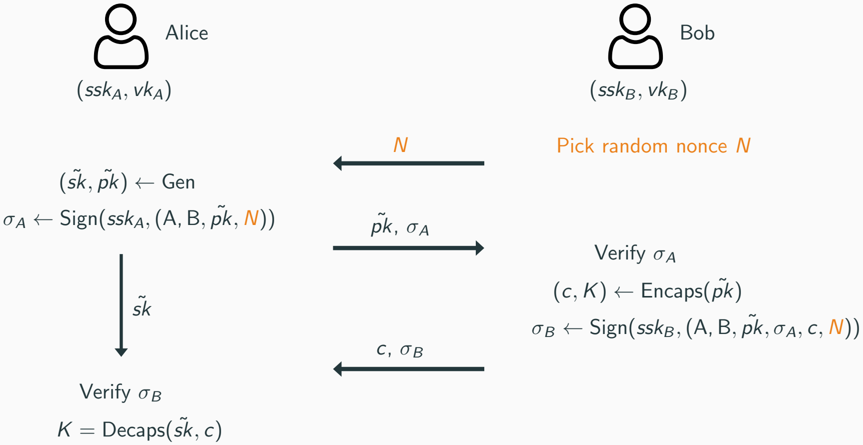
$\sigma_B \leftarrow \mathrm{Sign}(ssk_B, (\mathrm{A}, \mathrm{B}, \tilde{pk}, \sigma_A, c))$

$c,\ \sigma_B$

Verify σ_B

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

AKE[KEM, SIG, Nonce]



Alice

(ssk_A, vk_A)

Bob

(ssk_B, vk_B)

Pick random nonce N

N

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

$\sigma_A \leftarrow \text{Sign}(ssk_A, (\text{A}, \text{B}, \tilde{pk}, N))$

$\tilde{pk}, \sigma_A$

$\tilde{sk}$

Verify σ_A

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

$\sigma_B \leftarrow \text{Sign}(ssk_B, (\text{A}, \text{B}, \tilde{pk}, \sigma_A, c, N))$

c, σ_B

Verify σ_B

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

Security Requirements

KEM:	MU-MC-CCA	AKE[KEM, SIG]
	MU-SC-CCA	AKE[KEM, SIG, Nonce]

Security Requirements

KEM:	MU-MC-CCA	AKE[KEM, SIG]
	MU-SC-CCA	AKE[KEM, SIG, Nonce]

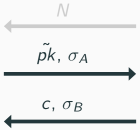
- Many AKE sessions
  → requires security for many key pairs

Security Requirements

KEM: MU-MC-CCA AKE[KEM, SIG]

MU-SC-CCA AKE[KEM, SIG, Nonce]

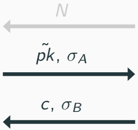
- Many AKE sessions
  → requires security for many key pairs



Security Requirements

KEM:	MU-MC-CCA	AKE[KEM, SIG]
	MU-SC-CCA	AKE[KEM, SIG, Nonce]

- Many AKE sessions
  → requires security for many key pairs
- AKE[KEM, SIG]: The adversary can replay ephemeral public key
  → many-ciphertext security

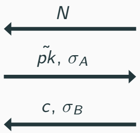


Security Requirements

KEM: MU-MC-CCA   AKE[KEM, SIG]

MU-SC-CCA   AKE[KEM, SIG, Nonce]

- Many AKE sessions
  → requires security for many key pairs
- AKE[KEM, SIG]: The adversary can replay ephemeral public key
  → many-ciphertext security
- AKE[KEM, SIG, Nonce]: A fresh nonce prevents replays
  → single-ciphertext security

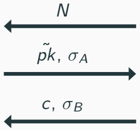


Security Requirements

KEM: MU-MC-CCA AKE[KEM, SIG]

MU-SC-CCA AKE[KEM, SIG, Nonce]

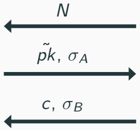
- Many AKE sessions
  → requires security for many key pairs
- AKE[KEM, SIG]: The adversary can replay ephemeral public key
  → many-ciphertext security
- AKE[KEM, SIG, Nonce]: A fresh nonce prevents replays
  → single-ciphertext security
- The adversary can choose ciphertexts when impersonating a user
  → requires to decrypt them to simulate keys correctly



Security Requirements

KEM:	MU-MC-CCA	AKE[KEM, SIG]
	MU-SC-CCA	AKE[KEM, SIG, Nonce]

- Many AKE sessions
  → requires security for many key pairs
- AKE[KEM, SIG]: The adversary can replay ephemeral public key
  → many-ciphertext security
- AKE[KEM, SIG, Nonce]: A fresh nonce prevents replays
  → single-ciphertext security
- The adversary can choose ciphertexts when impersonating a user
  → requires to decrypt them to simulate keys correctly



Security Requirements

Signature Scheme: MU-EUF-CMA$^{\text{corr}}$

- Each user has a long-term key pair
  → multi-user security

Security Requirements

Signature Scheme: MU-EUF-CMA$^{\text{corr}}$

- Each user has a long-term key pair
  → multi-user security
- Users authenticate each other explicitly
  → requires unforgeability of messages

Security Requirements

Signature Scheme: MU-EUF-CMA$^{\text{corr}}$

- Each user has a long-term key pair
  → multi-user security
- Users authenticate each other explicitly
  → requires unforgeability of messages
- The adversary can adaptively corrupt users
  → need to provide the secret signing key

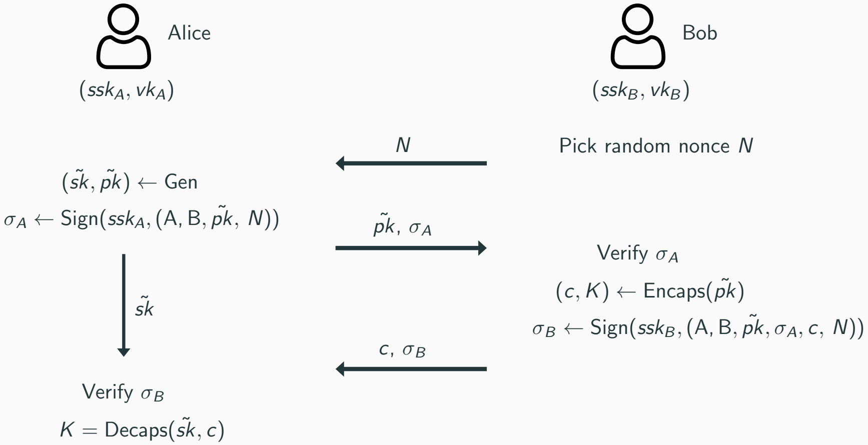
Security Requirements

Signature Scheme: MU-EUF-CMA$^{\text{corr}}$

- Each user has a long-term key pair
  → multi-user security
- Users authenticate each other explicitly
  → requires unforgeability of messages
- The adversary can adaptively corrupt users
  → need to provide the secret signing key

Security against State Reveal

AKE[KEM, SIG, Nonce]



Alice

(ssk_A, vk_A)

Bob

(ssk_B, vk_B)

Pick random nonce N

N

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

$\sigma_A \leftarrow \text{Sign}(ssk_A, (\text{A}, \text{B}, \tilde{pk}, \ N))$

$\tilde{pk}, \ \sigma_A$

Verify σ_A

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

$\sigma_B \leftarrow \text{Sign}(ssk_B, (\text{A}, \text{B}, \tilde{pk}, \sigma_A, c, \ N))$

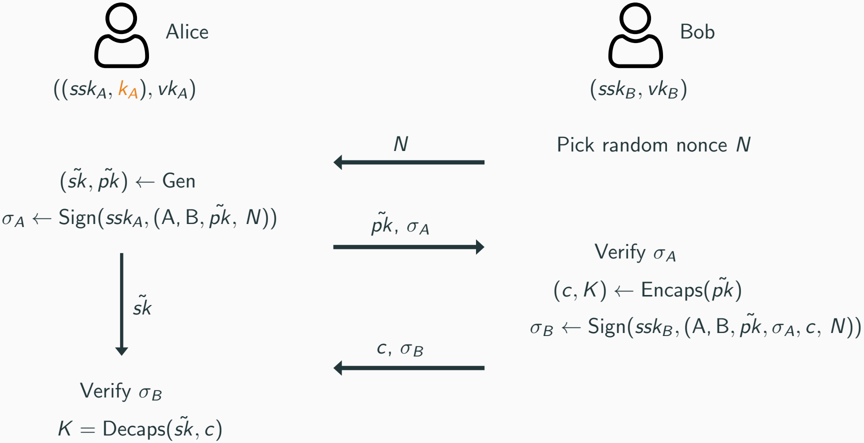
$\tilde{sk}$

$c, \ \sigma_B$

Verify σ_B

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

AKE[KEM, SIG, Nonce, SE]



Alice

$((ssk_A, k_A), vk_A)$

Bob

(ssk_B, vk_B)

Pick random nonce N

N

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

$\sigma_A \leftarrow \mathsf{Sign}(ssk_A, (\mathsf{A}, \mathsf{B}, \tilde{pk}, N))$

$\tilde{pk}, \sigma_A$

Verify σ_A

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

$\sigma_B \leftarrow \mathsf{Sign}(ssk_B, (\mathsf{A}, \mathsf{B}, \tilde{pk}, \sigma_A, c, N))$

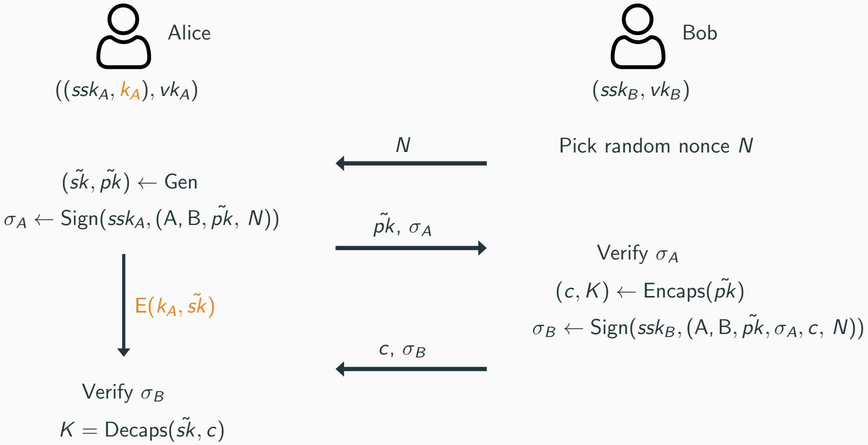
$\tilde{sk}$

c, σ_B

Verify σ_B

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

AKE[KEM, SIG, Nonce, SE]



Alice

$((ssk_A, k_A), vk_A)$

Bob

(ssk_B, vk_B)

Pick random nonce N

N

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

$\sigma_A \leftarrow \mathsf{Sign}(ssk_A, (\mathsf{A}, \mathsf{B}, \tilde{pk}, N))$

$\tilde{pk}, \sigma_A$

Verify σ_A

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

$\sigma_B \leftarrow \mathsf{Sign}(ssk_B, (\mathsf{A}, \mathsf{B}, \tilde{pk}, \sigma_A, c, N))$

$\mathsf{E}(k_A, \tilde{sk})$

c, σ_B

Verify σ_B

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

Enhanced Security Requirements

Observation: The adversary must learn k_A and $\mathsf{E}(k_A, \tilde{sk})$ to obtain $\tilde{sk}$.
$\rightarrow$ Trivial attack, no test session.

Enhanced Security Requirements

Observation: The adversary must learn k_A and $\mathrm{E}(k_A, \tilde{sk})$ to obtain $\tilde{sk}$.
→ Trivial attack, no test session.

But: we need to simulate correctly!

Enhanced Security Requirements

Observation: The adversary must learn k_A and $\mathsf{E}(k_A, \tilde{sk})$ to obtain $\tilde{sk}$.
→ Trivial attack, no test session.

But: we need to simulate correctly!

Yet another commitment problem

- After a state reveal, we don't know whether the adversary will later *corrupt* the user or *test* the session.
- Need to know all ephemeral secret key hidden inside the state.

Enhanced Security Requirements

KEM: additional algorithm $\text{Encaps}^*(sk) \to (c^*, K^*)$

- $\text{Encaps} \approx_c \text{Encaps}^*$ for many key pairs, even given secret keys
- $(pk, \text{Decaps}(sk, c), c^*, K^*) \approx_s (pk, \text{Decaps}(sk, c), c^*, \$)$

Enhanced Security Requirements

KEM: additional algorithm $\mathsf{Encaps}^*(sk) \to (c^*, K^*)$

- $\mathsf{Encaps} \approx_c \mathsf{Encaps}^*$ for many key pairs, even given secret keys
- $(pk, \mathsf{Decaps}(sk, c), c^*, K^*) \approx_s (pk, \mathsf{Decaps}(sk, c), c^*, \$)$

We show how to build such a KEM using universal$_2$ Hash Proof Systems based on (M)DDH.

Enhanced Security Requirements

KEM: additional algorithm $\text{Encaps}^*(sk) \rightarrow (c^*, K^*)$

- $\text{Encaps} \approx_c \text{Encaps}^*$ for many key pairs, even given secret keys
- $(pk, \text{Decaps}(sk, c), c^*, K^*) \approx_s (pk, \text{Decaps}(sk, c), c^*, \$)$

We show how to build such a KEM using universal$_2$ Hash Proof Systems based on (M)DDH.

Symmetric Encryption: standard CPA security

Tightly-Secure Signatures

Tightly-Secure Signature Schemes

Goal: MU-EUF-CMA$^{\text{corr}}$ security

- Previous schemes: in the ROM or tree-based
- Efficient scheme by BHJKL15, but the proof is flawed

Tightly-Secure Signature Schemes

Goal: MU-EUF-CMA$^{\text{corr}}$ security

- Previous schemes: in the ROM or tree-based
- Efficient scheme by BHJKL15, but the proof is flawed

	tightly-secure affine MAC ⇒ tightly-secure SIG
BKP14	
BHJKL15	
Our Work	

Tightly-Secure Signature Schemes

Goal: MU-EUF-CMA$^{\mathsf{corr}}$ security

- Previous schemes: in the ROM or tree-based
- Efficient scheme by BHJKL15, but the proof is flawed

	tightly-secure affine MAC ⇒ tightly-secure SIG
BKP14	single-user setting
BHJKL15	
Our Work	

Tightly-Secure Signature Schemes

Goal: MU-EUF-CMA$^{\mathsf{corr}}$ security

- Previous schemes: in the ROM or tree-based
- Efficient scheme by BHJKL15, but the proof is flawed

	tightly-secure affine MAC $\Rightarrow$ tightly-secure SIG
BKP14	single-user setting
BHJKL15	multi-user setting
Our Work	

Tightly-Secure Signature Schemes

Goal: MU-EUF-CMA$^{\text{corr}}$ security

- Previous schemes: in the ROM or tree-based
- Efficient scheme by BHJKL15, but the proof is flawed

	tightly-secure affine MAC ⇒ tightly-secure SIG
BKP14	single-user setting
BHJKL15	multi-user setting
Our Work	multi-user setting with corruptions

Tightly-Secure Signature Schemes

Goal: MU-EUF-CMA$^{\text{corr}}$ security

- Previous schemes: in the ROM or tree-based
- Efficient scheme by BHJKL15, but the proof is flawed

	tightly-secure affine MAC ⇒ tightly-secure SIG
BKP14	single-user setting
BHJKL15	multi-user setting
Our Work	multi-user setting with corruptions

We extend techniques of the LP19-HIBE to fix the scheme.

Tightly-Secure Signature Schemes

Goal: MU-EUF-CMA$^{\text{corr}}$ security

- Previous schemes: in the ROM or tree-based
- Efficient scheme by BHJKL15, but the proof is flawed

	tightly-secure affine MAC ⇒ tightly-secure SIG
BKP14	single-user setting
BHJKL15	multi-user setting
Our Work	multi-user setting with corruptions

We extend techniques of the LP19-HIBE to fix the scheme.

Still efficient: $|vk| = 1|\mathbb{G}|$, $|\sigma| = 5|\mathbb{G}|$ (instantiated under SXDH)

Summary

Contributions

- A new efficient and tight AKE protocol in the standard model.
- Security in a stronger security model, when allowing a non-tight reduction to the symmetric primitive.
- The first efficient and tightly-secure signature scheme supporting corruptions.

ePrint: `ia.cr/2021/863`

References

BKP14 Blazy, O., Kiltz, E., Pan, J.: (Hierarchical) identity-based encryption from affine message authentication. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 408-425. Springer, Heidelberg (Aug 2014)

BHJKL15 Bader, C., Hofheinz, D., Jager, T., Kiltz, E., Li, Y.: Tightly-secure authenticated key exchange. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part I. LNCS, vol. 9014, pp. 629-658. Springer, Heidelberg (Mar 2015)

GJ18 Gjøsteen, K., Jager, T.: Practical and tightly-secure digital signatures and authenticated key exchange. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part II. LNCS, vol. 10992, pp. 95-125. Springer, Heidelberg (Aug 2018)

LP19 Langrehr, R., Pan, J.: Tightly secure hierarchical identity-based encryption. In: Lin, D., Sako, K. (eds.) PKC 2019, Part I. LNCS, vol. 11442, pp. 436-465. Springer, Heidelberg (Apr 2019)

LLGW20 Liu, X., Liu, S., Gu, D., Weng, J.: Two-pass authenticated key exchange with explicit authentication and tight security. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part II. LNCS, vol. 12492, pp. 785-814. Springer, Heidelberg (Dec 2020)

JKRS21 Jager, T., Kiltz, E., Riepel, D., Schäge, S.: Tightly-secure authenticated key exchange, revisited. 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, EUROCRYPT 2021 (2021)