Shadowfax: Combiners for Deniability

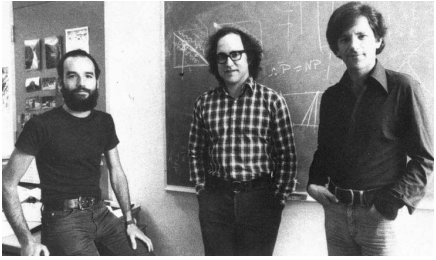
Phillip Gajland[1,2], **Vincent Hwang**[1] **and Jonas Janneck**[2]

[1] Max Planck Institute for Security and Privacy

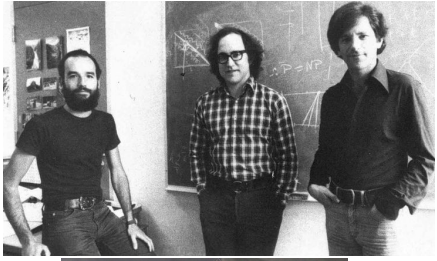
[2] Ruhr University Bochum

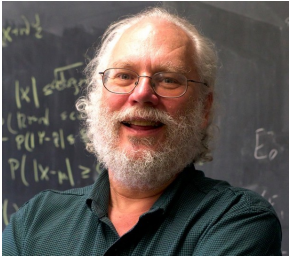
Cryptographic Applications Workshop at EUROCRYPT 2025, Madrid, Spain.

MOTIVATION: RSA, DH, SHOR AND QUANTUM

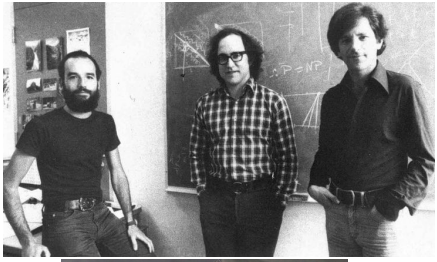


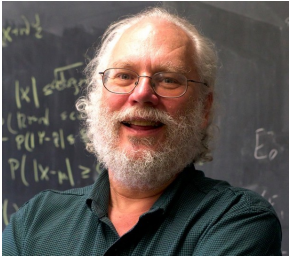
MOTIVATION: RSA, DH, SHOR AND QUANTUM

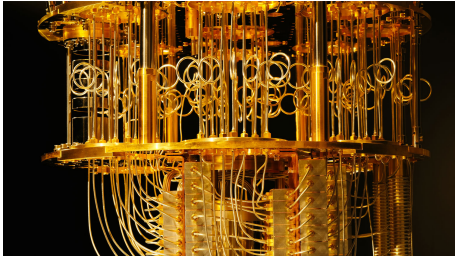




MOTIVATION: RSA, DH, SHOR AND QUANTUM







- NIST PQC standards for KEMs and Sig schemes.
- Protocols are being adapted to be post-quantum secure.
  - **X3DH → PQXDH** (deployed in **Signal**) [MP16, KS24].
  - **Apple's iMessage → PQ3** [App24].
  - **TLS →** [BCNS15], [PST20], [BBCT22], [Lan16], [Lan18], [KV19], [WR19], ...
- **Almost all proposals are hybrid!**

- NIST PQC standards for KEMs and Sig schemes.
- Protocols are being adapted to be post-quantum secure.
  - **X3DH** $\rightarrow$ **PQXDH** (deployed in **Signal**) [MP16, KS24].
  - **Apple's iMessage** $\rightarrow$ **PQ3** [App24].
  - **TLS** $\rightarrow$ [BCNS15], [PST20], [BBCT22], [Lan16], [Lan18], [KV19], [WR19], ...
- **Almost all proposals are hybrid!**

- NIST PQC standards for KEMs and Sig schemes.
- Protocols are being adapted to be post-quantum secure.
  - **X3DH → PQXDH** (deployed in **Signal**) [MP16, KS24].
  - **Apple's iMessage → PQ3** [App24].
  - **TLS** → [BCNS15], [PST20], [BBCT22], [Lan16], [Lan18], [KV19], [WR19], ...
- **Almost all proposals are hybrid!**

- NIST PQC standards for KEMs and Sig schemes.
- Protocols are being adapted to be post-quantum secure.
  - **X3DH** $\rightarrow$ **PQXDH** (deployed in **Signal**) [MP16, KS24].
  - **Apple's iMessage** $\rightarrow$ **PQ3** [App24].
  - **TLS** $\rightarrow$ [BCNS15], [PST20], [BBCT22], [Lan16], [Lan18], [KV19], [WR19], ...
- **Almost all proposals are hybrid!**

- NIST PQC standards for KEMs and Sig schemes.
- Protocols are being adapted to be post-quantum secure.
  - **X3DH** → **PQXDH** (deployed in **Signal**) [MP16, KS24].
  - **Apple's iMessage** → **PQ3** [App24].
  - **TLS** → [BCNS15], [PST20], [BBCT22], [Lan16], [Lan18], [KV19], [WR19], ...
- **Almost all proposals are hybrid!**

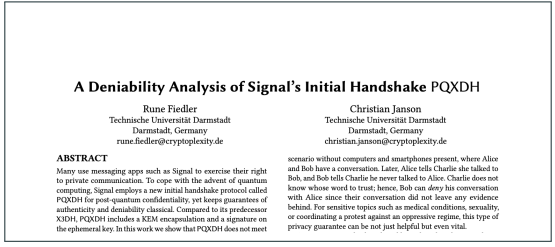
- NIST PQC standards for KEMs and Sig schemes.
- Protocols are being adapted to be post-quantum secure.
  - **X3DH** → **PQXDH** (deployed in **Signal**) [MP16, KS24].
  - **Apple's iMessage** → **PQ3** [App24].
  - **TLS** → [BCNS15], [PST20], [BBCT22], [Lan16], [Lan18], [KV19], [WR19], ...
- **Almost all proposals are hybrid!**

- **Deniability:** *"Bob is sure a message came from Alice, but Bob cannot prove to others that Alice wrote it."*
- **Classical setting:** X3DH provides deniability via implicit authentication.
- **PQXDH:** KEM and signature on ephemeral key → deniability is lost [FJ24].
- **Apple's iMessage PQ3:** lacks deniability.

- **Deniability:** *"Bob is sure a message came from Alice, but Bob cannot prove to others that Alice wrote it."*
- **Classical setting:** X3DH provides deniability via implicit authentication.
- **PQXDH:** KEM and signature on ephemeral key → deniability is lost [FJ24].
- **Apple's iMessage PQ3:** lacks deniability.

SOME SECURITY NOTIONS ARE LOST DURING PQC MIGRATION

- **Deniability:** *"Bob is sure a message came from Alice, but Bob cannot prove to others that Alice wrote it."*
- **Classical setting:** X3DH provides deniability via implicit authentication.
- **PQXDH:** KEM and signature on ephemeral key → deniability is lost [FJ24].
- **Apple's iMessage PQ3:** lacks deniability.



A Deniability Analysis of Signal's Initial Handshake PQXDH

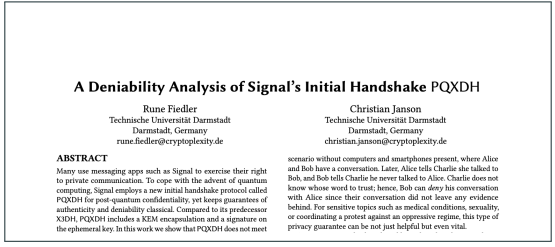
Rune Fiedler
Technische Universität Darmstadt
Darmstadt, Germany
rune.fiedler@cryptoplexity.de

Christian Janson
Technische Universität Darmstadt
Darmstadt, Germany
christian.janson@cryptoplexity.de

ABSTRACT

Many use messaging apps such as Signal to exercise their right to private communication. To cope with the advent of quantum computing, Signal employs a new initial handshake protocol called PQXDH for post-quantum confidentiality, yet keeps guarantees of authenticity and deniability classical. Compared to its predecessor X3DH, PQXDH includes a KEM encapsulation and a signature on the ephemeral key. In this work we show that PQXDH does not meet scenario without computers and smartphones present, where Alice and Bob have a conversation. Later, Alice tells Charlie she talked to Bob, and Bob tells Charlie he never talked to Alice. Charlie does not know whose word to trust; hence, Bob can *deny* his conversation with Alice since their conversation did not leave any evidence behind. For sensitive topics such as medical conditions, sexuality, or coordinating a protest against an oppressive regime, this type of privacy guarantee can be not just helpful but even vital.

- **Deniability:** *"Bob is sure a message came from Alice, but Bob cannot prove to others that Alice wrote it."*
- **Classical setting:** X3DH provides deniability via implicit authentication.
- **PQXDH:** KEM and signature on ephemeral key → deniability is lost [FJ24].
- **Apple's iMessage PQ3:** lacks deniability.



A Deniability Analysis of Signal's Initial Handshake PQXDH

Rune Fiedler
Technische Universität Darmstadt
Darmstadt, Germany
rune.fiedler@cryptoplexity.de

Christian Janson
Technische Universität Darmstadt
Darmstadt, Germany
christian.janson@cryptoplexity.de

ABSTRACT

Many use messaging apps such as Signal to exercise their right to private communication. To cope with the advent of quantum computing, Signal employs a new initial handshake protocol called PQXDH for post-quantum confidentiality, yet keeps guarantees of authenticity and deniability classical. Compared to its predecessor X3DH, PQXDH includes a KEM encapsulation and a signature on the ephemeral key. In this work we show that PQXDH does not meet scenario without computers and smartphones present, where Alice and Bob have a conversation. Later, Alice tells Charlie she talked to Bob, and Bob tells Charlie he never talked to Alice. Charlie does not know whose word to trust; hence, Bob can *deny* his conversation with Alice since their conversation did not leave any evidence behind. For sensitive topics such as medical conditions, sexuality, or coordinating a protest against an oppressive regime, this type of privacy guarantee can be not just helpful but even vital.

Q: "Can combiners preserve deniability in a post-quantum setting at minimal additional cost?"

OUTLINE

- Combiners: Two Classes
- Case Study: Authenticated KEM
- Black-box Construction: Problems
- Shadowfax
  - Construction
  - Instantiation
  - Comparison

- Combiners: Two Classes
- Case Study: Authenticated KEM
- Black-box Construction: Problems
- Shadowfax
  - Construction
  - Instantiation
  - Comparison

COMBINERS

- Construct a secure scheme Π from two schemes Π_1 and Π_2.
- Security is *binary*: either secure or insecure.
- Goal: $\Pi_1 \vee \Pi_2 \implies \Pi$.
- Examples of Π:
  - $\mathsf{PRG} := \mathsf{PRG}_1(s_1) \oplus \mathsf{PRG}_2(s_2)$
  - KEM with $k = \mathsf{H}(k_1, k_2, c_1, c_2)$ (RO model)
  - KEM with $k = \mathsf{PRF}(k_1, c_1 \| c_2) \oplus \mathsf{PRF}(k_2, c_1 \| c_2)$ (standard model [GHP18])

- Construct a secure scheme Π from two schemes Π_1 and Π_2.
- Security is *binary*: either secure or insecure.
- Goal: $\Pi_1 \vee \Pi_2 \implies \Pi$.
- Examples of Π:
  - $\mathsf{PRG} := \mathsf{PRG}_1(s_1) \oplus \mathsf{PRG}_2(s_2)$
  - KEM with $k = \mathsf{H}(k_1, k_2, c_1, c_2)$ (RO model)
  - KEM with $k = \mathsf{PRF}(k_1, c_1 \| c_2) \oplus \mathsf{PRF}(k_2, c_1 \| c_2)$ (standard model [GHP18])

- Construct a secure scheme Π from two schemes Π_1 and Π_2.
- Security is *binary*: either secure or insecure.
- Goal: $\Pi_1 \vee \Pi_2 \implies \Pi$.
- Examples of Π:
  - $\mathsf{PRG} := \mathsf{PRG}_1(s_1) \oplus \mathsf{PRG}_2(s_2)$
  - KEM with $k = \mathsf{H}(k_1, k_2, c_1, c_2)$ (RO model)
  - KEM with $k = \mathsf{PRF}(k_1, c_1 \| c_2) \oplus \mathsf{PRF}(k_2, c_1 \| c_2)$ (standard model [GHP18])

- Construct a secure scheme Π from two schemes Π_1 and Π_2.
- Security is *binary*: either secure or insecure.
- Goal: $\Pi_1 \vee \Pi_2 \implies \Pi$.
- Examples of Π:
  - $\mathsf{PRG} := \mathsf{PRG}_1(s_1) \oplus \mathsf{PRG}_2(s_2)$
  - KEM with $k = \mathsf{H}(k_1, k_2, c_1, c_2)$ (RO model)
  - KEM with $k = \mathsf{PRF}(k_1, c_1||c_2) \oplus \mathsf{PRF}(k_2, c_1||c_2)$ (standard model [GHP18])

- Construct a secure scheme Π from two schemes Π_1 and Π_2.
- Security is *binary*: either secure or insecure.
- Goal: $\Pi_1 \vee \Pi_2 \implies \Pi$.
- Examples of Π:
  - $\mathsf{PRG} \coloneqq \mathsf{PRG}_1(s_1) \oplus \mathsf{PRG}_2(s_2)$
  - KEM with $k = \mathsf{H}(k_1, k_2, c_1, c_2)$ (RO model)
  - KEM with $k = \mathsf{PRF}(k_1, c_1||c_2) \oplus \mathsf{PRF}(k_2, c_1||c_2)$ (standard model [GHP18])

- Construct a secure scheme Π from two schemes Π_1 and Π_2.
- Security is *binary*: either secure or insecure.
- Goal: $\Pi_1 \vee \Pi_2 \implies \Pi$.
- Examples of Π:
  - $\mathsf{PRG} \coloneqq \mathsf{PRG}_1(s_1) \oplus \mathsf{PRG}_2(s_2)$
  - KEM with $k = \mathsf{H}(k_1, k_2, c_1, c_2)$ (RO model)
  - KEM with $k = \mathsf{PRF}(k_1, c_1||c_2) \oplus \mathsf{PRF}(k_2, c_1||c_2)$ (standard model [GHP18])

- Construct a secure scheme Π from two schemes Π_1 and Π_2.
- Security is *binary*: either secure or insecure.
- Goal: $\Pi_1 \vee \Pi_2 \implies \Pi$.
- Examples of Π:
  - $\mathsf{PRG} \coloneqq \mathsf{PRG}_1(s_1) \oplus \mathsf{PRG}_2(s_2)$
  - KEM with $k = \mathsf{H}(k_1, k_2, c_1, c_2)$ (RO model)
  - KEM with $k = \mathsf{PRF}(k_1, c_1||c_2) \oplus \mathsf{PRF}(k_2, c_1||c_2)$ (standard model [GHP18])

- Build Π from classical ($\Pi_{\text{pre-Q}}$) and post-quantum ($\Pi_{\text{post-Q}}$) components.

Q: *Why use combiners for PQC migration?*

A: Classical implementations are mature; PQC is harder to implement. (code audits)

A: PQC schemes may lack formal proofs. (formal verification)

A: Most importantly, risk mitigation against assumption failures:

- **Quantum computers become practical: Break classical assumptions (factoring, discrete log).**
- **Post-quantum assumptions are flawed: LWE, SIDH, . . . may turn out easy.**

- Build Π from classical ($\Pi_{\text{pre-Q}}$) and post-quantum ($\Pi_{\text{post-Q}}$) components.

Q: *Why use combiners for PQC migration?*

A: Classical implementations are mature; PQC is harder to implement. (code audits)

A: PQC schemes may lack formal proofs. (formal verification)

A: **Most importantly, risk mitigation against assumption failures:**

- **Quantum computers become practical:** Break classical assumptions (factoring, discrete log).
- **Post-quantum assumptions are flawed:** LWE, SIDH, ... may turn out easy.

- Build Π from classical ($\Pi_{\text{pre-Q}}$) and post-quantum ($\Pi_{\text{post-Q}}$) components.

Q: *Why use combiners for PQC migration?*

A: Classical implementations are mature; PQC is harder to implement. (code audits)

A: PQC schemes may lack formal proofs. (formal verification)

A: **Most importantly, risk mitigation against assumption failures:**

- **Quantum computers become practical:** Break classical assumptions (factoring, discrete log).
- **Post-quantum assumptions are flawed:** LWE, SIDH, ... may turn out easy.

- Build Π from classical ($\Pi_{\text{pre-Q}}$) and post-quantum ($\Pi_{\text{post-Q}}$) components.

Q: *Why use combiners for PQC migration?*

A: Classical implementations are mature; PQC is harder to implement. (code audits)

A: PQC schemes may lack formal proofs. (formal verification)

A: Most importantly, risk mitigation against assumption failures:

- **Quantum computers become practical:** Break classical assumptions (factoring, discrete log).
- **Post-quantum assumptions are flawed:** LWE, SIDH, ... may turn out easy.

- Build Π from classical ($\Pi_{\text{pre-Q}}$) and post-quantum ($\Pi_{\text{post-Q}}$) components.

Q: *Why use combiners for PQC migration?*

A: Classical implementations are mature; PQC is harder to implement. (code audits)

A: PQC schemes may lack formal proofs. (formal verification)

A: Most importantly, risk mitigation against assumption failures:

- **Quantum computers become practical**: Break classical assumptions (factoring, discrete log).
- **Post-quantum assumptions are flawed**: LWE, SIDH, ... may turn out easy.

- Build Π from classical ($\Pi_{\text{pre-Q}}$) and post-quantum ($\Pi_{\text{post-Q}}$) components.

Q: *Why use combiners for PQC migration?*

A: Classical implementations are mature; PQC is harder to implement. (code audits)

A: PQC schemes may lack formal proofs. (formal verification)

A: Most importantly, risk mitigation against assumption failures:

- **Quantum computers become practical**: Break classical assumptions (factoring, discrete log).
- **Post-quantum assumptions are flawed**: LWE, SIDH, ... may turn out easy.

- Build Π from classical ($\Pi_{\text{pre-Q}}$) and post-quantum ($\Pi_{\text{post-Q}}$) components.

Q: *Why use combiners for PQC migration?*

A: Classical implementations are mature; PQC is harder to implement. (code audits)

A: PQC schemes may lack formal proofs. (formal verification)

A: Most importantly, risk mitigation against assumption failures:

- **Quantum computers become practical**: Break classical assumptions (factoring, discrete log).
- **Post-quantum assumptions are flawed**: LWE, SIDH, ... may turn out easy.

- Build Π from classical ($\Pi_{\text{pre-Q}}$) and post-quantum ($\Pi_{\text{post-Q}}$) components.

Q: *Why use combiners for PQC migration?*

A: Classical implementations are mature; PQC is harder to implement. (code audits)

A: PQC schemes may lack formal proofs. (formal verification)

A: Most importantly, risk mitigation against assumption failures:

- **Quantum computers become practical**: Break classical assumptions (factoring, discrete log).
- **Post-quantum assumptions are flawed**: LWE, SIDH, ... may turn out easy.

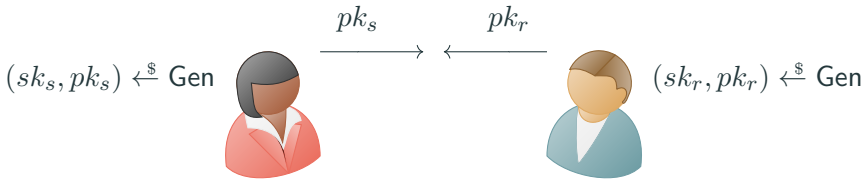
$$\textbf{Can relax to: } \textbf{Assumption}_{\textbf{pre-Q}} \vee \textbf{Assumption}_{\textbf{post-Q}} \implies \Pi$$

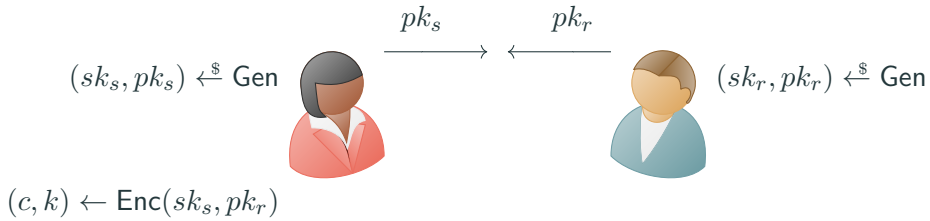
CASE STUDY: AUTHENTICATED KEM

AUTHENTICATED KEY ENCAPSULATION MECHANISM (AKEM) [ABH+21, GJK24]

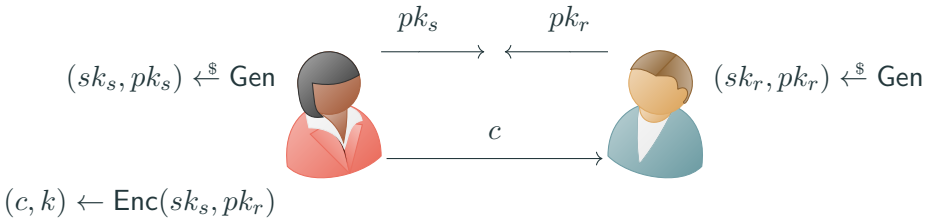
$(sk_s, pk_s) \xleftarrow{\$} \mathsf{Gen}$

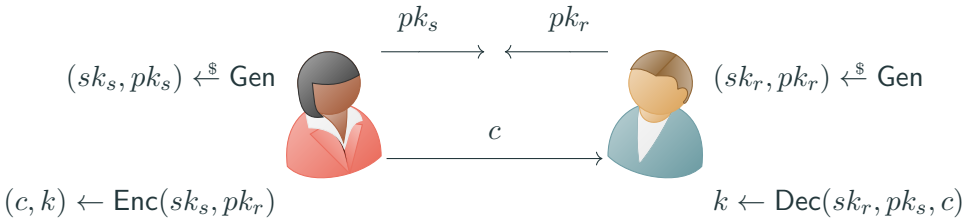
$(sk_r, pk_r) \xleftarrow{\$} \mathsf{Gen}$


pk_s
pk_r
$(sk_s, pk_s) \xleftarrow{\$} \mathsf{Gen}$
$(sk_r, pk_r) \xleftarrow{\$} \mathsf{Gen}$

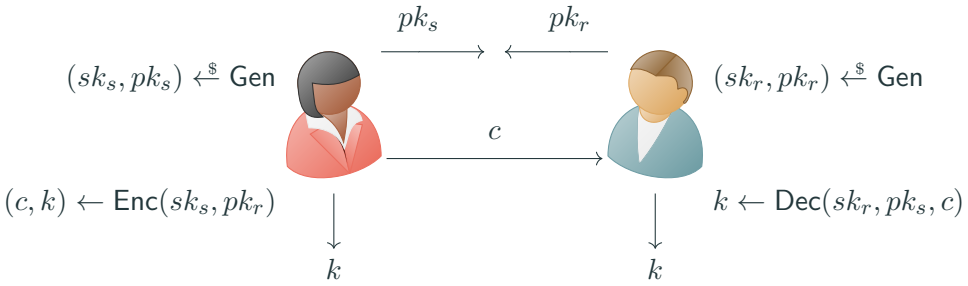

pk_s
pk_r
$(sk_s, pk_s) \xleftarrow{\$} \mathsf{Gen}$
$(sk_r, pk_r) \xleftarrow{\$} \mathsf{Gen}$
$(c, k) \leftarrow \mathsf{Enc}(sk_s, pk_r)$

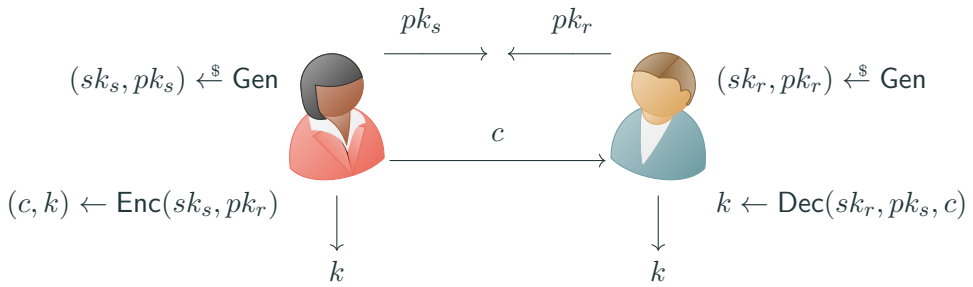
AUTHENTICATED KEY ENCAPSULATION MECHANISM (AKEM) [ABH+21, GJK24]



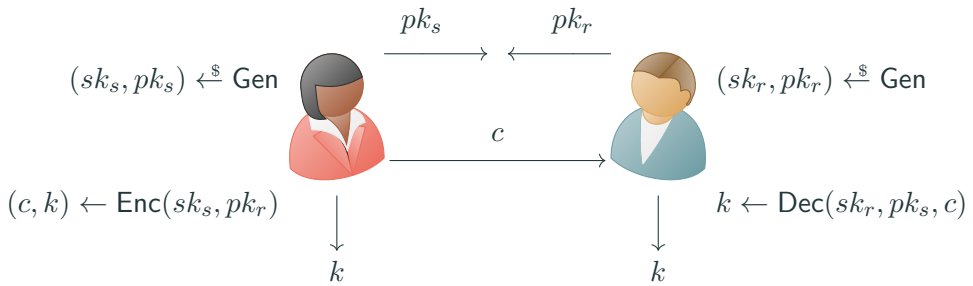

pk_s
pk_r
$(sk_s, pk_s) \xleftarrow{\$} \mathsf{Gen}$
$(sk_r, pk_r) \xleftarrow{\$} \mathsf{Gen}$
c
$(c, k) \leftarrow \mathsf{Enc}(sk_s, pk_r)$
$k \leftarrow \mathsf{Dec}(sk_r, pk_s, c)$

AUTHENTICATED KEY ENCAPSULATION MECHANISM (AKEM) [ABH+21, GJK24]

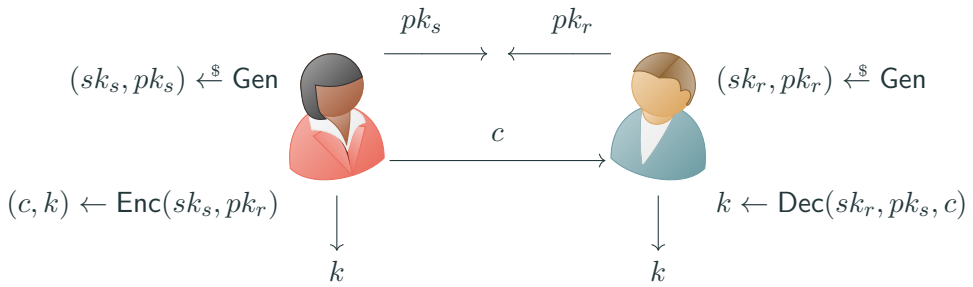




- **Confidentiality:** k should look random.



- **Confidentiality:** k should look random.
- **Authenticity:** r knows s sent the ciphertext c.



- **Confidentiality:** k should look random.
- **Authenticity:** r knows s sent the ciphertext c.
- **Deniability:** s can deny having sent c to r.
  - $\exists\, \mathsf{Sim} : \mathsf{Enc}(sk_s, pk_r) \approx \mathsf{Sim}(\dots)$ (honest/dishonest receivers)

- AKEM formalises the `auth` mode of HPKE [BBLW22].
- Messaging
  - Messaging Layer Security (MLS) [BBR+23].
  - K-waay [CHN+24].
- Designing a protocol that needs one-shot authentication? Let us know!

- AKEM formalises the `auth` mode of HPKE [BBLW22].
- Messaging
  - Messaging Layer Security (MLS) [BBR+23].
  - K-waay [CHN+24].
- Designing a protocol that needs one-shot authentication? Let us know!

- AKEM formalises the `auth` mode of HPKE [BBLW22].
- Messaging
  - Messaging Layer Security (MLS) [BBR+23].
  - K-waay [CHN+24].
- Designing a protocol that needs one-shot authentication? Let us know!

- AKEM formalises the `auth` mode of HPKE [BBLW22].
- Messaging
  - Messaging Layer Security (MLS) [BBR+23].
  - K-waay [CHN+24].
- Designing a protocol that needs one-shot authentication? Let us know!

- AKEM formalises the `auth` mode of HPKE [BBLW22].
- Messaging
  - Messaging Layer Security (MLS) [BBR+23].
  - K-waay [CHN+24].
- Designing a protocol that needs one-shot authentication? Let us know!

BLACK-BOX CONSTRUCTION OF A DENIABLE $\mathsf{AKEM} := [\mathsf{AKEM}_1, \mathsf{AKEM}_2, \mathsf{H}]$

BLACK-BOX CONSTRUCTION OF A DENIABLE $\mathsf{AKEM} := [\mathsf{AKEM}_1, \mathsf{AKEM}_2, \mathsf{H}]$

$\underline{\mathsf{Gen}}$	$\underline{\mathsf{Enc}(sk_s =: (sk_1, sk_2), pk_r =: (pk_1, pk_2))}$	$\underline{\mathsf{Dec}(pk_s =: (pk_1, pk_2), sk_r =: (sk_1, sk_2), c)}$
$(sk_1, pk_1) \xleftarrow{\$} \mathsf{AKEM}_1.\mathsf{Gen}$	$(c_1, k_1) \xleftarrow{\$} \mathsf{AKEM}_1.\mathsf{Enc}(sk_1, pk_1)$	$\textbf{parse } c \to (c_1, c_2)$
$(sk_2, pk_2) \xleftarrow{\$} \mathsf{AKEM}_2.\mathsf{Gen}$	$(c_2, k_2) \xleftarrow{\$} \mathsf{AKEM}_2.\mathsf{Enc}(sk_2, pk_2)$	$k_1 \leftarrow \mathsf{AKEM}_1.\mathsf{Dec}(pk_1, sk_1, c_1)$
$\textbf{return } ((sk_1, sk_2), (pk_1, pk_2))$	$c \coloneqq (c_1, c_2)$	$k_2 \leftarrow \mathsf{AKEM}_2.\mathsf{Dec}(pk_2, sk_2, c_2)$
	$k \coloneqq \mathsf{H}(k_1, k_2, pk_s, pk_r, c)$	$k \coloneqq \mathsf{H}(k_1, k_2, pk_s, pk_r, c)$
	$\textbf{return } (c, k)$	$\textbf{return } k$

BLACK-BOX CONSTRUCTION OF A DENIABLE AKEM $:= [\mathsf{AKEM}_1, \mathsf{AKEM}_2, \mathsf{H}]$

$\underline{\mathsf{Gen}}$	$\underline{\mathsf{Enc}(sk_s =: (sk_1, sk_2), pk_r =: (pk_1, pk_2))}$	$\underline{\mathsf{Dec}(pk_s =: (pk_1, pk_2), sk_r =: (sk_1, sk_2), c)}$
$(sk_1, pk_1) \xleftarrow{\$} \mathsf{AKEM}_1.\mathsf{Gen}$	$(c_1, k_1) \xleftarrow{\$} \mathsf{AKEM}_1.\mathsf{Enc}(sk_1, pk_1)$	**parse** $c \to (c_1, c_2)$
$(sk_2, pk_2) \xleftarrow{\$} \mathsf{AKEM}_2.\mathsf{Gen}$	$(c_2, k_2) \xleftarrow{\$} \mathsf{AKEM}_2.\mathsf{Enc}(sk_2, pk_2)$	$k_1 \leftarrow \mathsf{AKEM}_1.\mathsf{Dec}(pk_1, sk_1, c_1)$
return $((sk_1, sk_2), (pk_1, pk_2))$	$c \coloneqq (c_1, c_2)$	$k_2 \leftarrow \mathsf{AKEM}_2.\mathsf{Dec}(pk_2, sk_2, c_2)$
	$k \coloneqq \mathsf{H}(k_1, k_2, pk_s, pk_r, c)$	$k \coloneqq \mathsf{H}(k_1, k_2, pk_s, pk_r, c)$
	return (c, k)	**return** k

BLACK-BOX CONSTRUCTION OF A DENIABLE AKEM := $[\mathsf{AKEM}_1, \mathsf{AKEM}_2, \mathsf{H}]$

$\underline{\mathsf{Gen}}$	$\underline{\mathsf{Enc}(sk_s =: (sk_1, sk_2), pk_r =: (pk_1, pk_2))}$	$\underline{\mathsf{Dec}(pk_s =: (pk_1, pk_2), sk_r =: (sk_1, sk_2), c)}$
$(sk_1, pk_1) \xleftarrow{\$} \mathsf{AKEM}_1.\mathsf{Gen}$	$(c_1, k_1) \xleftarrow{\$} \mathsf{AKEM}_1.\mathsf{Enc}(sk_1, pk_1)$	**parse** $c \to (c_1, c_2)$
$(sk_2, pk_2) \xleftarrow{\$} \mathsf{AKEM}_2.\mathsf{Gen}$	$(c_2, k_2) \xleftarrow{\$} \mathsf{AKEM}_2.\mathsf{Enc}(sk_2, pk_2)$	$k_1 \leftarrow \mathsf{AKEM}_1.\mathsf{Dec}(pk_1, sk_1, c_1)$
return $((sk_1, sk_2), (pk_1, pk_2))$	$c \coloneqq (c_1, c_2)$	$k_2 \leftarrow \mathsf{AKEM}_2.\mathsf{Dec}(pk_2, sk_2, c_2)$
	$k \coloneqq \mathsf{H}(k_1, k_2, pk_s, pk_r, c)$	$k \coloneqq \mathsf{H}(k_1, k_2, pk_s, pk_r, c)$
	return (c, k)	**return** k

BLACK-BOX CONSTRUCTION OF A DENIABLE $\mathsf{AKEM} := [\mathsf{AKEM}_1, \mathsf{AKEM}_2, \mathsf{H}]$

$\underline{\mathsf{Gen}}$	$\underline{\mathsf{Enc}(sk_s =: (sk_1, sk_2), pk_r =: (pk_1, pk_2))}$	$\underline{\mathsf{Dec}(pk_s =: (pk_1, pk_2), sk_r =: (sk_1, sk_2), c)}$
$(sk_1, pk_1) \xleftarrow{\$} \mathsf{AKEM}_1.\mathsf{Gen}$	$(c_1, k_1) \xleftarrow{\$} \mathsf{AKEM}_1.\mathsf{Enc}(sk_1, pk_1)$	**parse** $c \to (c_1, c_2)$
$(sk_2, pk_2) \xleftarrow{\$} \mathsf{AKEM}_2.\mathsf{Gen}$	$(c_2, k_2) \xleftarrow{\$} \mathsf{AKEM}_2.\mathsf{Enc}(sk_2, pk_2)$	$k_1 \leftarrow \mathsf{AKEM}_1.\mathsf{Dec}(pk_1, sk_1, c_1)$
return $((sk_1, sk_2), (pk_1, pk_2))$	$c \coloneqq (c_1, c_2)$	$k_2 \leftarrow \mathsf{AKEM}_2.\mathsf{Dec}(pk_2, sk_2, c_2)$
	$k \coloneqq \mathsf{H}(k_1, k_2, pk_s, pk_r, c)$	$k \coloneqq \mathsf{H}(k_1, k_2, pk_s, pk_r, c)$
	return (c, k)	**return** k

BLACK-BOX CONSTRUCTION OF A DENIABLE $\mathsf{AKEM} := [\mathsf{AKEM}_1, \mathsf{AKEM}_2, \mathsf{H}]$

$\underline{\mathsf{Gen}}$	$\underline{\mathsf{Enc}(sk_s =: (sk_1, sk_2), pk_r =: (pk_1, pk_2))}$	$\underline{\mathsf{Dec}(pk_s =: (pk_1, pk_2), sk_r =: (sk_1, sk_2), c)}$
$(sk_1, pk_1) \xleftarrow{\$} \mathsf{AKEM}_1.\mathsf{Gen}$	$(c_1, k_1) \xleftarrow{\$} \mathsf{AKEM}_1.\mathsf{Enc}(sk_1, pk_1)$	**parse** $c \to (c_1, c_2)$
$(sk_2, pk_2) \xleftarrow{\$} \mathsf{AKEM}_2.\mathsf{Gen}$	$(c_2, k_2) \xleftarrow{\$} \mathsf{AKEM}_2.\mathsf{Enc}(sk_2, pk_2)$	$k_1 \leftarrow \mathsf{AKEM}_1.\mathsf{Dec}(pk_1, sk_1, c_1)$
return $((sk_1, sk_2), (pk_1, pk_2))$	$c \coloneqq (c_1, c_2)$	$k_2 \leftarrow \mathsf{AKEM}_2.\mathsf{Dec}(pk_2, sk_2, c_2)$
	$k \coloneqq \mathsf{H}(k_1, k_2, pk_s, pk_r, c)$	$k \coloneqq \mathsf{H}(k_1, k_2, pk_s, pk_r, c)$
	return (c, k)	**return** k

- **Confidentiality:** follows from either AKEM_1 **or** AKEM_2 ($\approx$ KEM combiner)
  - However, k also hashes sender and receiver public keys to satisfy **Ins-CCA**.
- **Authenticity:** follows from either AKEM_1 **or** AKEM_2.
- **Deniability:** follows from both AKEM_1 **and** AKEM_2.

BLACK-BOX CONSTRUCTION OF A DENIABLE $\mathsf{AKEM} := [\mathsf{AKEM}_1, \mathsf{AKEM}_2, \mathsf{H}]$

$\underline{\mathsf{Gen}}$	$\underline{\mathsf{Enc}(sk_s =: (sk_1, sk_2), pk_r =: (pk_1, pk_2))}$	$\underline{\mathsf{Dec}(pk_s =: (pk_1, pk_2), sk_r =: (sk_1, sk_2), c)}$
$(sk_1, pk_1) \xleftarrow{\$} \mathsf{AKEM}_1.\mathsf{Gen}$	$(c_1, k_1) \xleftarrow{\$} \mathsf{AKEM}_1.\mathsf{Enc}(sk_1, pk_1)$	**parse** $c \to (c_1, c_2)$
$(sk_2, pk_2) \xleftarrow{\$} \mathsf{AKEM}_2.\mathsf{Gen}$	$(c_2, k_2) \xleftarrow{\$} \mathsf{AKEM}_2.\mathsf{Enc}(sk_2, pk_2)$	$k_1 \leftarrow \mathsf{AKEM}_1.\mathsf{Dec}(pk_1, sk_1, c_1)$
return $((sk_1, sk_2), (pk_1, pk_2))$	$c \coloneqq (c_1, c_2)$	$k_2 \leftarrow \mathsf{AKEM}_2.\mathsf{Dec}(pk_2, sk_2, c_2)$
	$k \coloneqq \mathsf{H}(k_1, k_2, pk_s, pk_r, c)$	$k \coloneqq \mathsf{H}(k_1, k_2, pk_s, pk_r, c)$
	return (c, k)	**return** k

- **Confidentiality:** follows from either AKEM_1 **or** AKEM_2 ($\approx$ KEM combiner)
  - However, k also hashes sender and receiver public keys to satisfy **Ins-CCA**.
- **Authenticity:** follows from either AKEM_1 **or** AKEM_2.
- **Deniability:** follows from both AKEM_1 **and** AKEM_2.

Black-box construction of a deniable $\mathsf{AKEM} := [\mathsf{AKEM}_1, \mathsf{AKEM}_2, \mathsf{H}]$

$\underline{\mathsf{Gen}}$	$\underline{\mathsf{Enc}(sk_s =: (sk_1, sk_2), pk_r =: (pk_1, pk_2))}$	$\underline{\mathsf{Dec}(pk_s =: (pk_1, pk_2), sk_r =: (sk_1, sk_2), c)}$
$(sk_1, pk_1) \xleftarrow{\$} \mathsf{AKEM}_1.\mathsf{Gen}$	$(c_1, k_1) \xleftarrow{\$} \mathsf{AKEM}_1.\mathsf{Enc}(sk_1, pk_1)$	**parse** $c \to (c_1, c_2)$
$(sk_2, pk_2) \xleftarrow{\$} \mathsf{AKEM}_2.\mathsf{Gen}$	$(c_2, k_2) \xleftarrow{\$} \mathsf{AKEM}_2.\mathsf{Enc}(sk_2, pk_2)$	$k_1 \leftarrow \mathsf{AKEM}_1.\mathsf{Dec}(pk_1, sk_1, c_1)$
return $((sk_1, sk_2), (pk_1, pk_2))$	$c \coloneqq (c_1, c_2)$	$k_2 \leftarrow \mathsf{AKEM}_2.\mathsf{Dec}(pk_2, sk_2, c_2)$
	$k \coloneqq \mathsf{H}(k_1, k_2, pk_s, pk_r, c)$	$k \coloneqq \mathsf{H}(k_1, k_2, pk_s, pk_r, c)$
	return (c, k)	**return** k

- **Confidentiality:** follows from either AKEM_1 **or** AKEM_2 ($\approx$ KEM combiner)
  - However, k also hashes sender and receiver public keys to satisfy **Ins-CCA**.
- **Authenticity:** follows from either AKEM_1 **or** AKEM_2.
- **Deniability:** follows from both AKEM_1 **and** AKEM_2.

BLACK-BOX CONSTRUCTION OF A DENIABLE $\mathsf{AKEM} := [\mathsf{AKEM}_1, \mathsf{AKEM}_2, \mathsf{H}]$

$\underline{\mathsf{Gen}}$	$\underline{\mathsf{Enc}(sk_s =: (sk_1, sk_2), pk_r =: (pk_1, pk_2))}$	$\underline{\mathsf{Dec}(pk_s =: (pk_1, pk_2), sk_r =: (sk_1, sk_2), c)}$
$(sk_1, pk_1) \xleftarrow{\$} \mathsf{AKEM}_1.\mathsf{Gen}$	$(c_1, k_1) \xleftarrow{\$} \mathsf{AKEM}_1.\mathsf{Enc}(sk_1, pk_1)$	**parse** $c \to (c_1, c_2)$
$(sk_2, pk_2) \xleftarrow{\$} \mathsf{AKEM}_2.\mathsf{Gen}$	$(c_2, k_2) \xleftarrow{\$} \mathsf{AKEM}_2.\mathsf{Enc}(sk_2, pk_2)$	$k_1 \leftarrow \mathsf{AKEM}_1.\mathsf{Dec}(pk_1, sk_1, c_1)$
return $((sk_1, sk_2), (pk_1, pk_2))$	$c \coloneqq (c_1, c_2)$	$k_2 \leftarrow \mathsf{AKEM}_2.\mathsf{Dec}(pk_2, sk_2, c_2)$
	$k \coloneqq \mathsf{H}(k_1, k_2, pk_s, pk_r, c)$	$k \coloneqq \mathsf{H}(k_1, k_2, pk_s, pk_r, c)$
	return (c, k)	**return** k

- **Confidentiality:** follows from either AKEM_1 **or** AKEM_2 ($\approx$ KEM combiner)
  - However, k also hashes sender and receiver public keys to satisfy **Ins-CCA**.
- **Authenticity:** follows from either AKEM_1 **or** AKEM_2.
- **Deniability:** follows from both AKEM_1 **and** AKEM_2.

Q: *What does it mean to lose deniability?*

Cannot construct $\mathsf{Sim} : \mathsf{Enc}(sk_s, pk_r) \approx \mathsf{Sim}(\dots)$

Q: *Computationally or Statistically Indistinguishable?*

Comp. deniability: $\neg\text{Assumption}_{\text{pre-Q}} \vee \neg\text{Assumption}_{\text{post-Q}} \implies \neg\text{Deniability}$

Stat. deniability: $\neg\text{Assumption}_{\text{pre-Q}} \vee \neg\text{Assumption}_{\text{post-Q}} \not\Longrightarrow \neg\text{Deniability}$

A: Use statistical deniability. **Not possible** for confidentiality/authenticity!

- PAKE combiners rely on statical password hiding [HR25].
- OKEM combiners rely on statistical ciphertext uniformity [GRSV25].

Q: *What does it mean to lose deniability?*

Cannot construct $\mathsf{Sim} : \mathsf{Enc}(sk_s, pk_r) \approx \mathsf{Sim}(\dots)$

Q: *Computationally or Statistically Indistinguishable?*

Comp. deniability: $\neg\text{Assumption}_{\text{pre-Q}} \vee \neg\text{Assumption}_{\text{post-Q}} \implies \neg\text{Deniability}$

Stat. deniability: $\neg\text{Assumption}_{\text{pre-Q}} \vee \neg\text{Assumption}_{\text{post-Q}} \not\Longrightarrow \neg\text{Deniability}$

A: Use statistical deniability. **Not possible** for confidentiality/authenticity!

- PAKE combiners rely on statical password hiding [HR25].
- OKEM combiners rely on statistical ciphertext uniformity [GRSV25].

Q: *What does it mean to lose deniability?*

Cannot construct $\mathsf{Sim} : \mathsf{Enc}(sk_s, pk_r) \approx \mathsf{Sim}(\dots)$

Q: *Computationally or Statistically Indistinguishable?*

Comp. deniability: $\neg\text{Assumption}_{\text{pre-Q}} \vee \neg\text{Assumption}_{\text{post-Q}} \implies \neg\text{Deniability}$

Stat. deniability: $\neg\text{Assumption}_{\text{pre-Q}} \vee \neg\text{Assumption}_{\text{post-Q}} \not\Longrightarrow \neg\text{Deniability}$

A: Use statistical deniability. **Not possible** for confidentiality/authenticity!

- PAKE combiners rely on statical password hiding [HR25].
- OKEM combiners rely on statistical ciphertext uniformity [GRSV25].

Q: *What does it mean to lose deniability?*

Cannot construct $\mathsf{Sim} : \mathsf{Enc}(sk_s, pk_r) \approx \mathsf{Sim}(\dots)$

Q: *Computationally or Statistically Indistinguishable?*

Comp. deniability: $\neg\text{Assumption}_{\text{pre-Q}} \vee \neg\text{Assumption}_{\text{post-Q}} \implies \neg\text{Deniability}$

Stat. deniability: $\neg\text{Assumption}_{\text{pre-Q}} \vee \neg\text{Assumption}_{\text{post-Q}} \not\Longrightarrow \neg\text{Deniability}$

A: Use statistical deniability. **Not possible** for confidentiality/authenticity!

- PAKE combiners rely on statical password hiding [HR25].
- OKEM combiners rely on statistical ciphertext uniformity [GRSV25].

Q: *What does it mean to lose deniability?*

Cannot construct $\mathsf{Sim} : \mathsf{Enc}(sk_s, pk_r) \approx \mathsf{Sim}(\dots)$

Q: *Computationally or Statistically Indistinguishable?*

Comp. deniability: $\neg\text{Assumption}_{\text{pre-Q}} \vee \neg\text{Assumption}_{\text{post-Q}} \implies \neg\text{Deniability}$

Stat. deniability: $\neg\text{Assumption}_{\text{pre-Q}} \vee \neg\text{Assumption}_{\text{post-Q}} \not\Longrightarrow \neg\text{Deniability}$

A: Use statistical deniability. **Not possible** for confidentiality/authenticity!

- PAKE combiners rely on statical password hiding [HR25].
- OKEM combiners rely on statistical ciphertext uniformity [GRSV25].

Q: *What does it mean to lose deniability?*

Cannot construct $\mathsf{Sim} : \mathsf{Enc}(sk_s, pk_r) \approx \mathsf{Sim}(\dots)$

Q: *Computationally or Statistically Indistinguishable?*

Comp. deniability: $\neg\text{Assumption}_{\text{pre-Q}} \vee \neg\text{Assumption}_{\text{post-Q}} \implies \neg\text{Deniability}$

Stat. deniability: $\neg\text{Assumption}_{\text{pre-Q}} \vee \neg\text{Assumption}_{\text{post-Q}} \not\implies \neg\text{Deniability}$

A: Use statistical deniability. **Not possible** for confidentiality/authenticity!

- PAKE combiners rely on statical password hiding [HR25].
- OKEM combiners rely on statistical ciphertext uniformity [GRSV25].

Q: *What does it mean to lose deniability?*

Cannot construct $\mathsf{Sim} : \mathsf{Enc}(sk_s, pk_r) \approx \mathsf{Sim}(\dots)$

Q: *Computationally or Statistically Indistinguishable?*

Comp. deniability: $\neg\text{Assumption}_{\text{pre-Q}} \vee \neg\text{Assumption}_{\text{post-Q}} \implies \neg\text{Deniability}$

Stat. deniability: $\neg\text{Assumption}_{\text{pre-Q}} \vee \neg\text{Assumption}_{\text{post-Q}} \not\implies \neg\text{Deniability}$

A: Use statistical deniability. **Not possible** for confidentiality/authenticity!

- PAKE combiners rely on statical password hiding [HR25].
- OKEM combiners rely on statistical ciphertext uniformity [GRSV25].

Q: *What does it mean to lose deniability?*

Cannot construct $\mathsf{Sim} : \mathsf{Enc}(sk_s, pk_r) \approx \mathsf{Sim}(\dots)$

Q: *Computationally or Statistically Indistinguishable?*

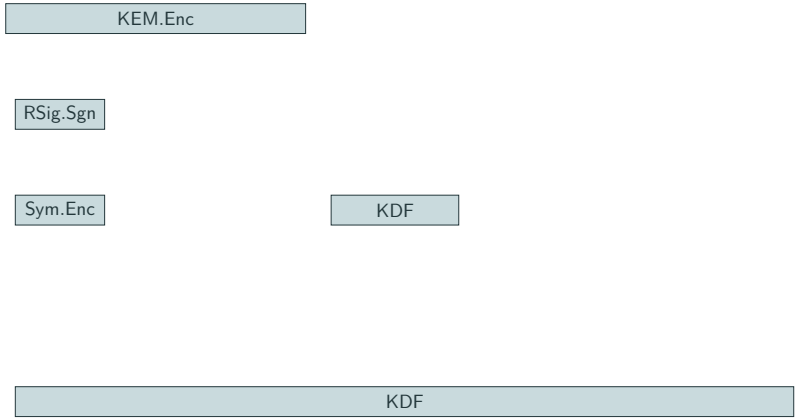
Comp. deniability: $\neg\text{Assumption}_{\text{pre-Q}} \vee \neg\text{Assumption}_{\text{post-Q}} \implies \neg\text{Deniability}$

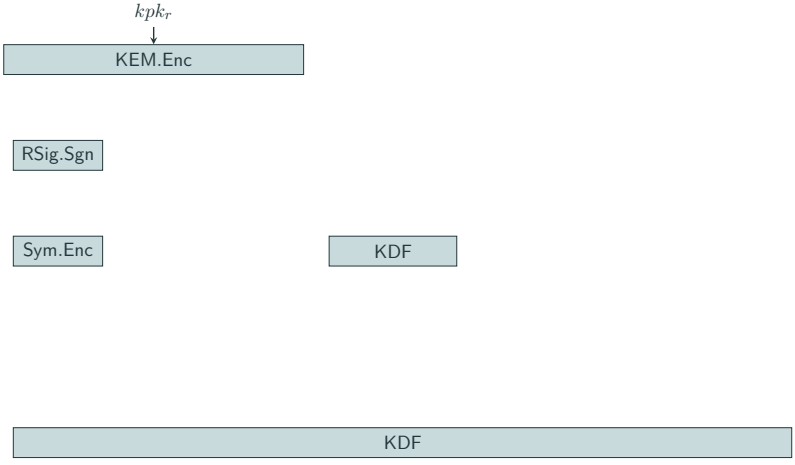
Stat. deniability: $\neg\text{Assumption}_{\text{pre-Q}} \vee \neg\text{Assumption}_{\text{post-Q}} \not\implies \neg\text{Deniability}$

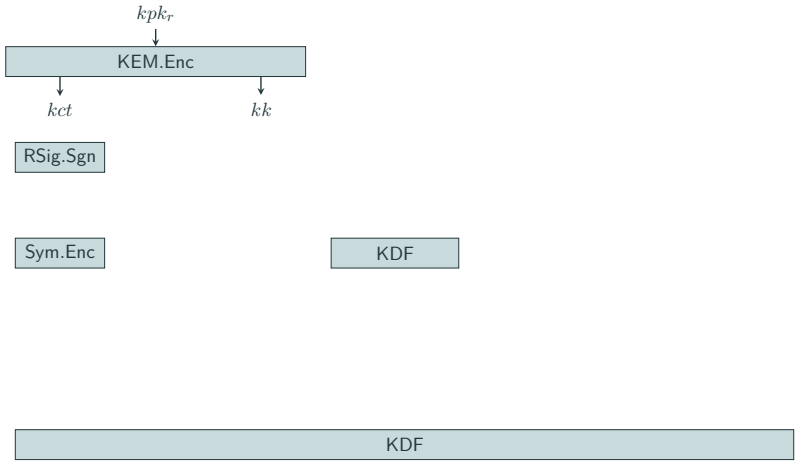
A: Use statistical deniability. **Not possible** for confidentiality/authenticity!

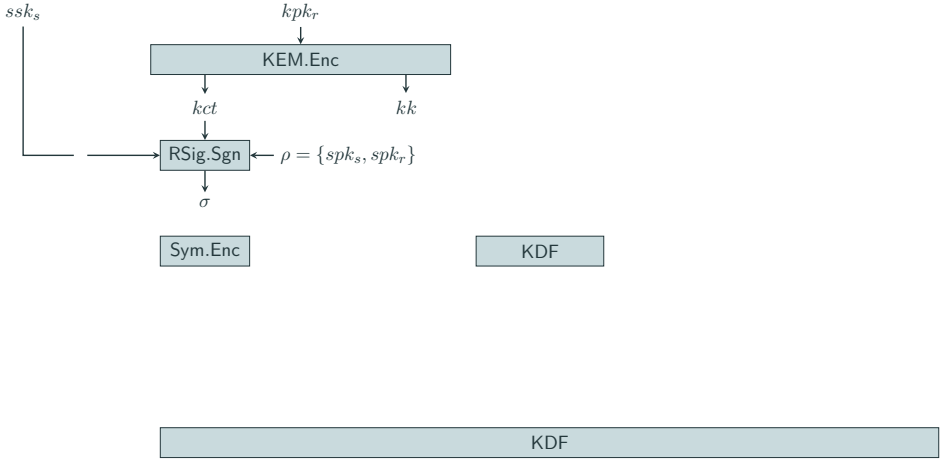
- PAKE combiners rely on statical password hiding [HR25].
- OKEM combiners rely on statistical ciphertext uniformity [GRSV25].

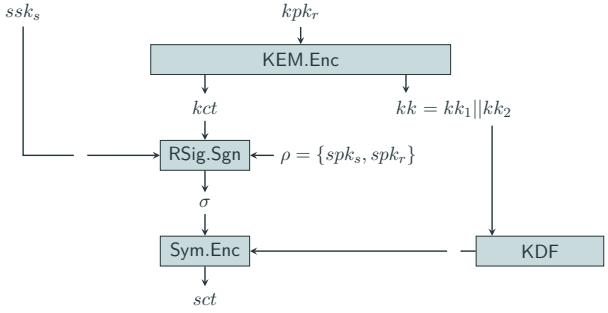
SHADOWFAX


KEM.Enc
RSig.Sgn
Sym.Enc
KDF
KDF

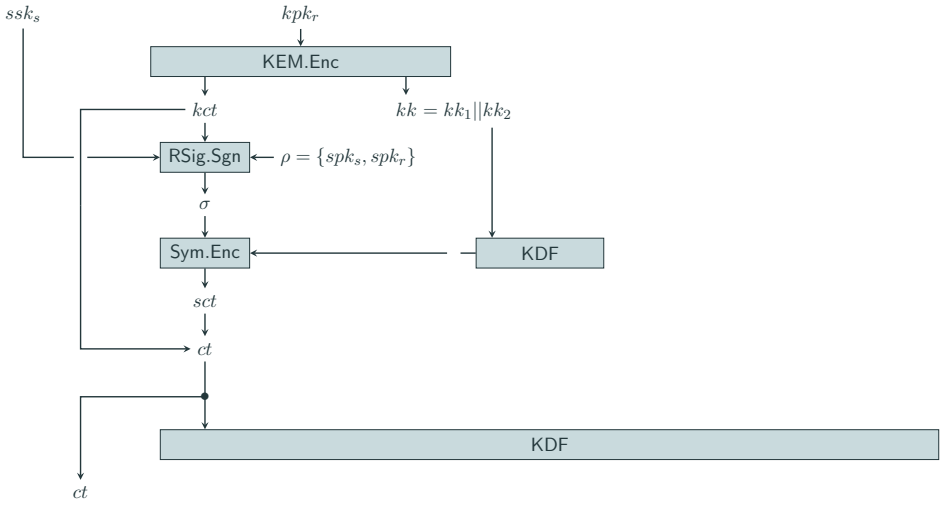

kpk_r
KEM.Enc
RSig.Sgn
Sym.Enc
KDF
KDF

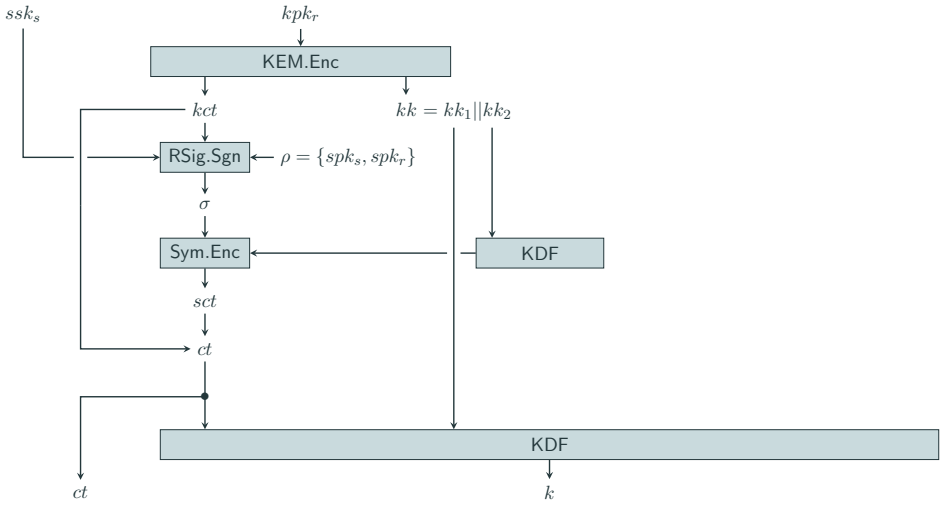

kpk_r
KEM.Enc
kct
kk
RSig.Sgn
Sym.Enc
KDF
KDF

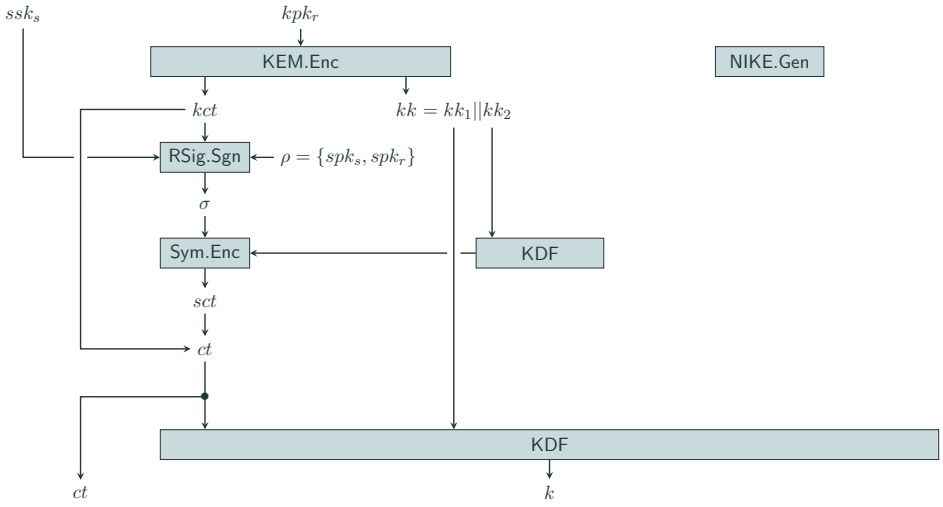

ssk_s
kpk_r
KEM.Enc
kct
kk
RSig.Sgn
$\rho = \{spk_s, spk_r\}$
σ
Sym.Enc
KDF
KDF

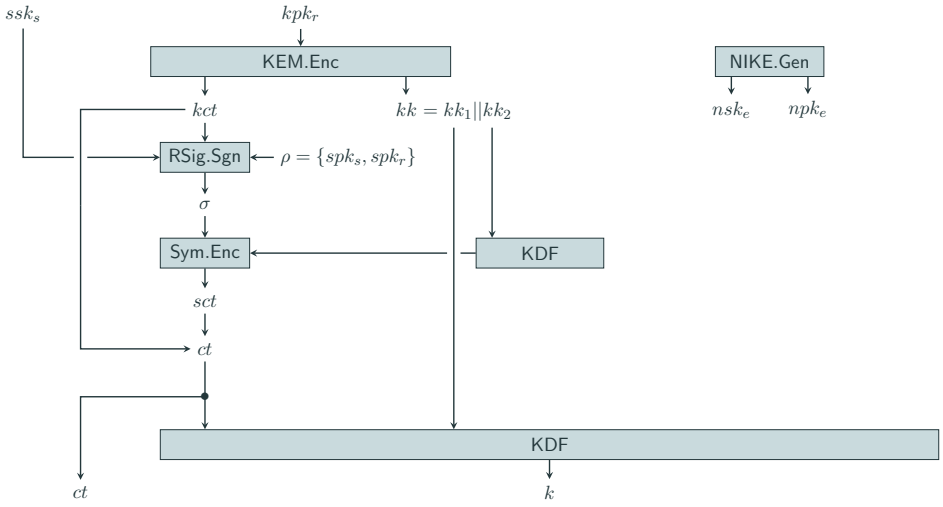

ssk_s
kpk_r
KEM.Enc
kct
$kk = kk_1||kk_2$
RSig.Sgn
$\rho = \{spk_s, spk_r\}$
σ
Sym.Enc
KDF
sct

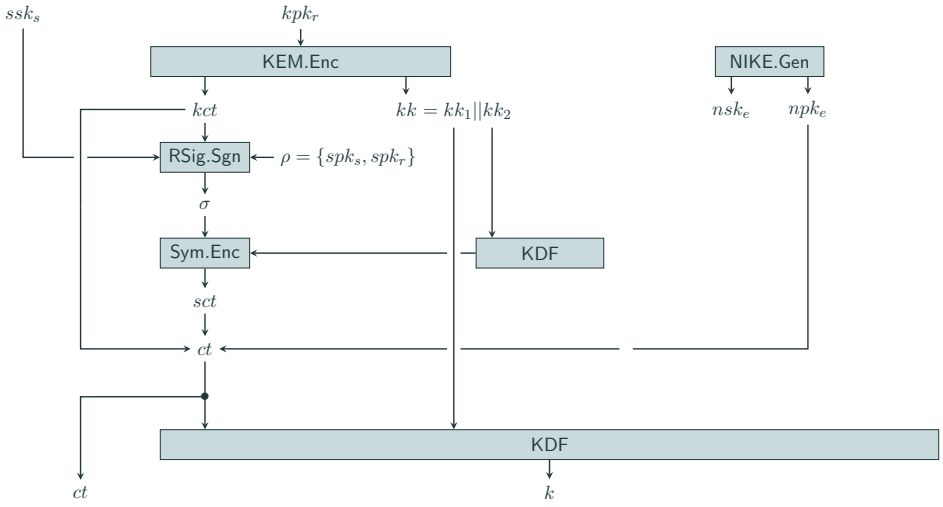

KDF

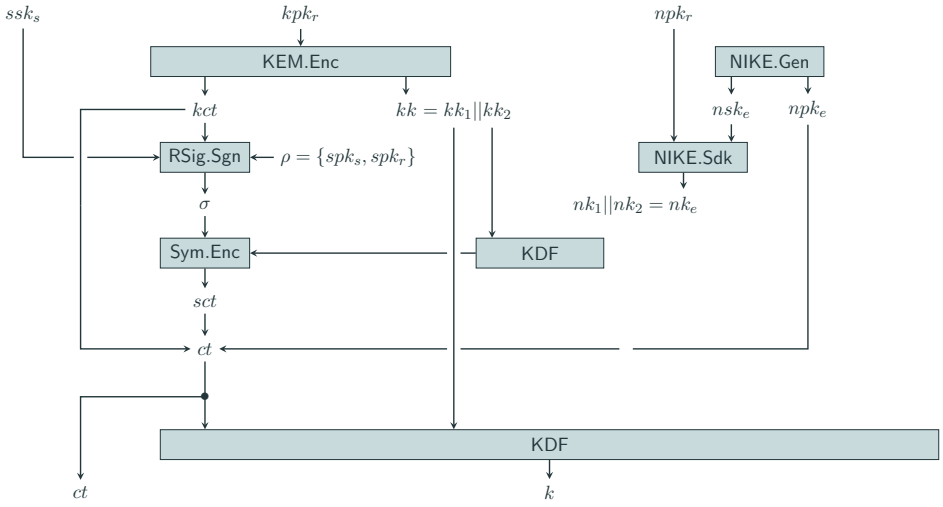

ssk_s
kpk_r
KEM.Enc
kct
$kk = kk_1||kk_2$
RSig.Sgn
$\rho = \{spk_s, spk_r\}$
σ
Sym.Enc
KDF
sct
ct
KDF
ct

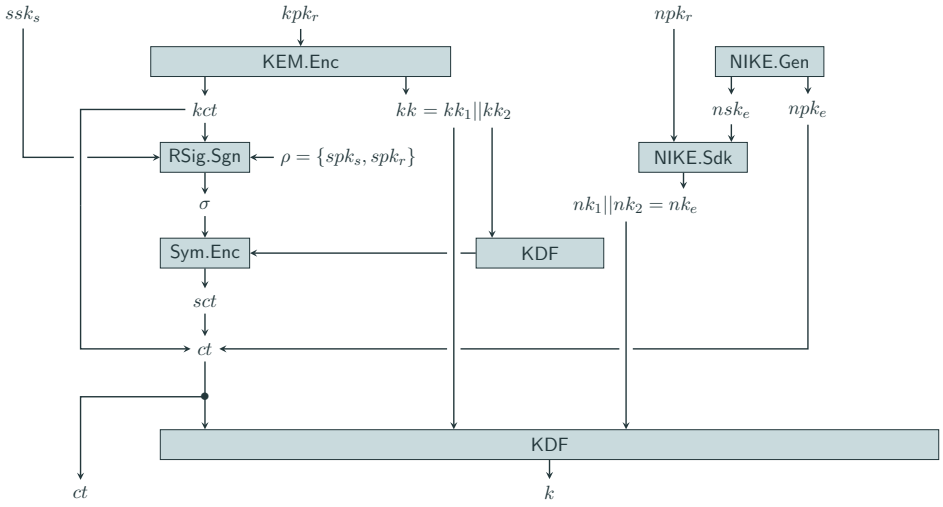

ssk_s
kpk_r
KEM.Enc
kct
$kk = kk_1||kk_2$
RSig.Sgn
$\rho = \{spk_s, spk_r\}$
σ
Sym.Enc
KDF
sct
ct
KDF
ct
k

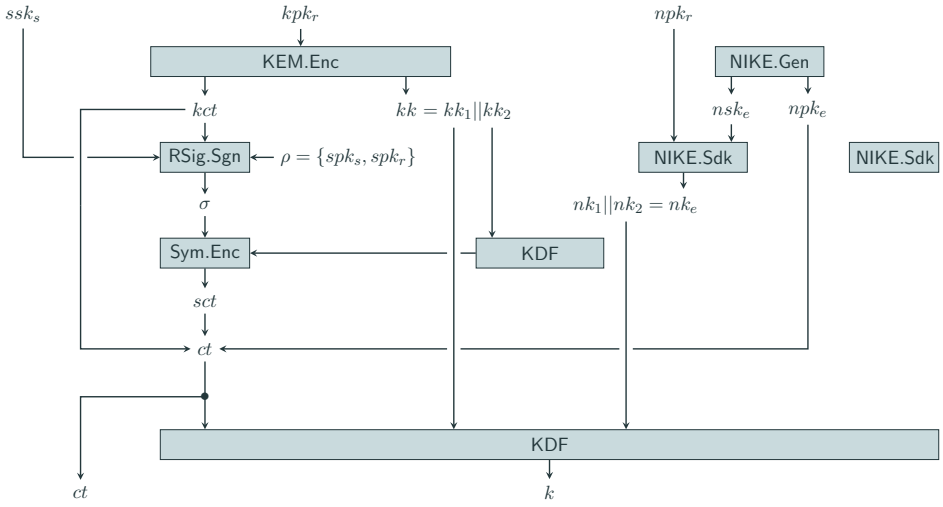

ssk_s
kpk_r
KEM.Enc
NIKE.Gen
kct
$kk = kk_1||kk_2$
RSig.Sign
$\rho = \{spk_s, spk_r\}$
σ
Sym.Enc
KDF
sct
ct
KDF
ct
k

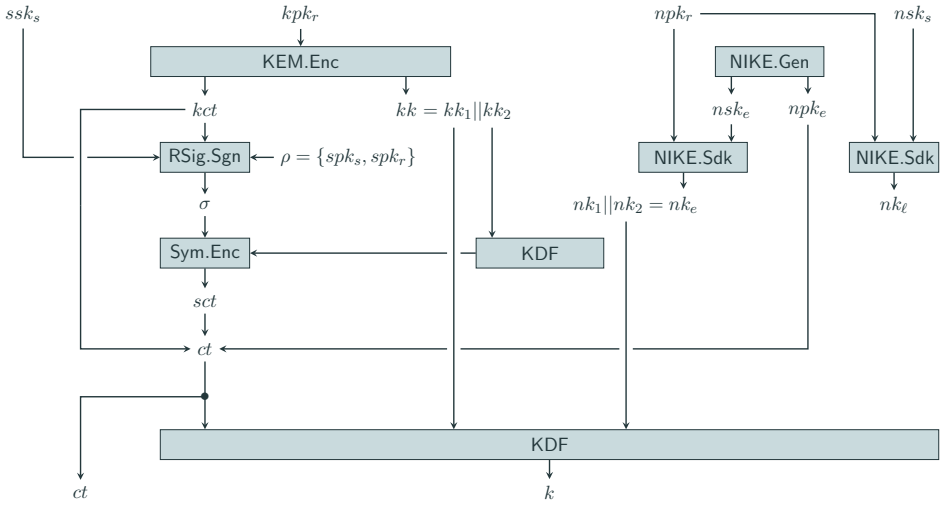

ssk_s
kpk_r
KEM.Enc
NIKE.Gen
kct
$kk = kk_1||kk_2$
nsk_e
npk_e
RSig.Sgn
$\rho = \{spk_s, spk_r\}$
σ
Sym.Enc
KDF
sct
ct
KDF
ct
k


ssk_s
kpk_r
KEM.Enc
NIKE.Gen
kct
$kk = kk_1||kk_2$
nsk_e
npk_e
RSig.Sgn
$\rho = \{spk_s, spk_r\}$
σ
Sym.Enc
KDF
sct
ct
KDF
ct
k

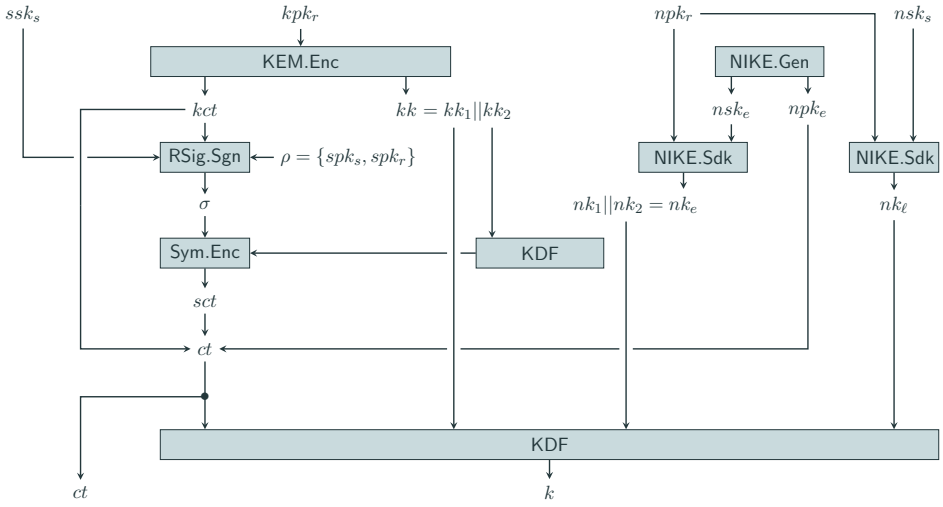

ssk_s
kpk_r
npk_r
KEM.Enc
NIKE.Gen
kct
$kk = kk_1||kk_2$
nsk_e
npk_e
RSig.Sgn
$\rho = \{spk_s, spk_r\}$
NIKE.Sdk
σ
$nk_1||nk_2 = nk_e$
Sym.Enc
KDF
sct
ct
KDF
ct
k

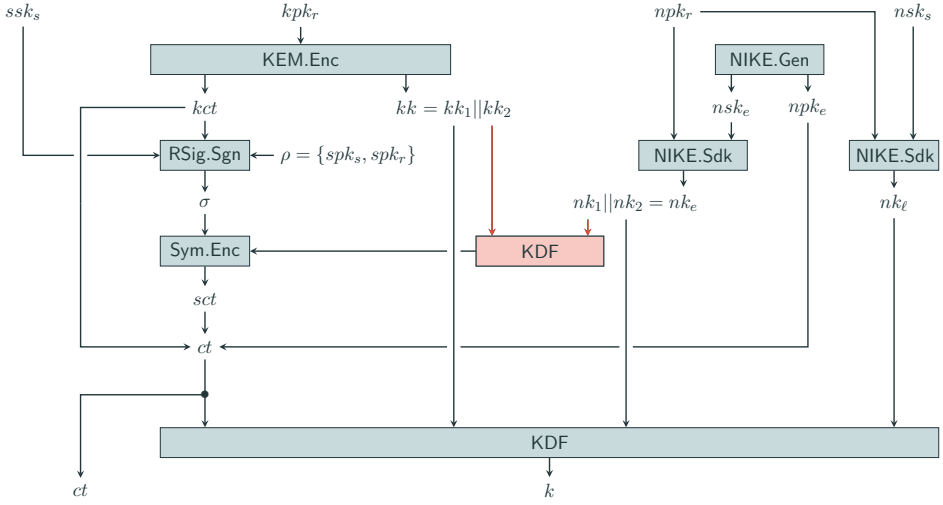

ssk_s
kpk_r
npk_r
KEM.Enc
NIKE.Gen
kct
$kk = kk_1||kk_2$
nsk_e
npk_e
RSig.Sgn
$\rho = \{spk_s, spk_r\}$
NIKE.Sdk
σ
$nk_1||nk_2 = nk_e$
Sym.Enc
KDF
sct
ct
KDF
ct
k


ssk_s
kpk_r
npk_r
KEM.Enc
NIKE.Gen
kct
$kk = kk_1||kk_2$
nsk_e
npk_e
RSig.Sgn
$\rho = \{spk_s, spk_r\}$
NIKE.Sdk
NIKE.Sdk
σ
$nk_1||nk_2 = nk_e$
Sym.Enc
KDF
sct
ct
KDF
ct
k


ssk_s
kpk_r
KEM.Enc
kct
$kk = kk_1||kk_2$
RSig.Sgn
$\rho = \{spk_s, spk_r\}$
σ
Sym.Enc
KDF
sct
ct
ct
KDF
k
npk_r
NIKE.Gen
nsk_e
npk_e
NIKE.Sdk
$nk_1||nk_2 = nk_e$
nsk_s
NIKE.Sdk
nk_ℓ

DENIABLE AKEM := (Gen, Enc, Dec) — SHADOWFAX CONSTRUCTION




ssk_s
kpk_r
KEM.Enc
kct
$kk = kk_1||kk_2$
RSig.Sgn
$\rho = \{spk_s, spk_r\}$
σ
Sym.Enc
KDF
sct
ct
ct
npk_r
NIKE.Gen
nsk_e
npk_e
NIKE.Sdk
$nk_1||nk_2 = nk_e$
nsk_s
NIKE.Sdk
nk_ℓ
KDF
k

INSTANTIATIONS OF PRIMITIVES USED IN SHADOWFAX

Primitive	Scheme (variant)	Assumption	Size (in bytes)		
			σ	c	pk
NIKE	X25519 [Ber06]	CDH	—	32	32

INSTANTIATIONS OF PRIMITIVES USED IN SHADOWFAX

Primitive	**Scheme (variant)**	**Assumption**	**Size (in bytes)**		
			σ	c	pk
NIKE	`X25519` [Ber06]	CDH	—	32	32
KEM	`bat_257_512` [FKPY22]	NTRU, Ring-LWR	—	473	521

INSTANTIATIONS OF PRIMITIVES USED IN SHADOWFAX

Primitive	Scheme (variant)	Assumption	Size (in bytes)		
			σ	c	pk
NIKE	`X25519` [Ber06]	CDH	—	32	32
KEM	`bat_257_512` [FKPY22]	NTRU, Ring-LWR	—	473	521
	`ML-KEM-512` [ML-24]	M-LWE	—	768	800

INSTANTIATIONS OF PRIMITIVES USED IN SHADOWFAX

Primitive	Scheme (variant)		Assumption	Size (in bytes)		
				σ	c	pk
NIKE	`X25519` [Ber06]		CDH	—	32	32
KEM	`bat_257_512` [FKPY22]		NTRU, Ring-LWR	—	473	521
	`ML-KEM-512` [ML-24]		M-LWE	—	768	800
RSig	Gandalf[TpdGen, PreSmp] [GJK24]	Antrag [ENS+23], Mitaka [EFG+22]	NTRU, Ring-ISIS	1 236	—	896

INSTANTIATIONS OF PRIMITIVES USED IN SHADOWFAX

Primitive	Scheme (variant)		Assumption	Size (in bytes)		
				σ	c	pk
NIKE	`X25519` [Ber06]		CDH	—	32	32
KEM	`bat_257_512` [FKPY22]		NTRU, Ring-LWR	—	473	521
	`ML-KEM-512` [ML-24]		M-LWE	—	768	800
RSig	Gandalf[TpdGen, PreSmp] [GJK24]	Antrag [ENS+23], Mitaka [EFG+22]	NTRU, Ring-ISIS	1 236	—	896
		`Falcon-512` [PFH+22]		1 276	—	897

INSTANTIATIONS OF PRIMITIVES USED IN SHADOWFAX

Primitive	Scheme (variant)		Assumption	Size (in bytes)		
				σ	c	pk
NIKE	`X25519` [Ber06]		CDH	—	32	32
KEM	`bat_257_512` [FKPY22]		NTRU, Ring-LWR	—	473	521
	`ML-KEM-512` [ML-24]		M-LWE	—	768	800
RSig	Gandalf[TpdGen, PreSmp] [GJK24]	Antrag [ENS+23], Mitaka [EFG+22]	NTRU, Ring-ISIS	1 236	—	896
		`Falcon-512` [PFH+22]		1 276	—	897

- We opt for the most compact instantiations.
- However, also possible to use NIST PQC standards.

AKEM INSTANTIATIONS

Scheme	Instantiation	Deniability	Assumption		Size (in bytes)	
			pre-Q	post-Q	c	pk
DH-AKEM [ABH+21]	`X25519`	**DR-Den***	✓	✗	32	32

Deniability properties marked with an "$*$" have not been formally proven in the respective work.

AKEM instantiations

Scheme	Instantiation	Deniability	Assumption		Size (in bytes)	
			pre-Q	post-Q	c	pk
DH-AKEM [ABH+21]	`X25519`	**DR-Den***	✓	✗	32	32
EtStH-AKEM [AJKL23]	`bat_257_542` + Antrag	✗	✗	✓	1 119	1 417
	`ML-KEM-512` + `Falcon-512`				1 434	1 697

Deniability properties marked with an "$*$" have not been formally proven in the respective work.

AKEM instantiations

Scheme	Instantiation	Deniability	Assumption		Size (in bytes)	
			pre-Q	post-Q	c	pk
DH-AKEM [ABH+21]	`X25519`	**DR-Den***	✓	✗	32	32
EtStH-AKEM [AJKL23]	`bat_257_542` + Antrag	✗	✗	✓	1 119	1 417
	`ML-KEM-512` + `Falcon-512`				1 434	1 697
NIKE-AKEM [AJKL23]	Swoosh [GdKQ+24]	**DR-Den***	✗	✓	$> 221\,184$	$> 221\,184$

Deniability properties marked with an "$*$" have not been formally proven in the respective work.

AKEM instantiations

Scheme	Instantiation	Deniability	Assumption		Size (in bytes)	
			pre-Q	post-Q	c	pk
DH-AKEM [ABH+21]	`X25519`	**DR-Den***	✓	✗	32	32
EtSt-AKEM [AJKL23]	`bat_257_542` + Antrag	✗	✗	✓	1 119	1 417
	`ML-KEM-512` + `Falcon-512`				1 434	1 697
NIKE-AKEM [AJKL23]	Swoosh [GdKQ+24]	**DR-Den***	✗	✓	$> 221\,184$	$> 221\,184$
FrodoKEX+ [CHN+24]	n/a	**DR-Den**	✗	✓	72	21 300

Deniability properties marked with an "$*$" have not been formally proven in the respective work.

AKEM instantiations

Scheme	Instantiation	Deniability	Assumption		Size (in bytes)	
			pre-Q	post-Q	c	pk
DH-AKEM [ABH+21]	`X25519`	**DR-Den***	✓	✗	32	32
EtSt H-AKEM [AJKL23]	`bat_257_542` + Antrag	✗	✗	✓	1 119	1 417
	`ML-KEM-512` + `Falcon-512`				1 434	1 697
NIKE-AKEM [AJKL23]	Swoosh [GdKQ+24]	**DR-Den***	✗	✓	$> 221\,184$	$> 221\,184$
FrodoKEX+ [CHN+24]	n/a	**DR-Den**	✗	✓	72	21 300
PQ-AKEM [GJK24]	NTRU-A + Gandalf [Antrag, Mitaka]	**HR-Den** & **DR-Den**	✗	✓	2 044	1 664
	`bat_257_542` + Gandalf [Antrag, Mitaka]				1 749	1 417

Deniability properties marked with an "$*$" have not been formally proven in the respective work.

AKEM INSTANTIATIONS

Scheme	Instantiation	Deniability	Assumption		Size (in bytes)	
			pre-Q	post-Q	c	pk
DH-AKEM [ABH+21]	X25519	**DR-Den***	✓	✗	32	32
EtSt H-AKEM [AJKL23]	bat_257_542 + Antrag	✗	✗	✓	1 119	1 417
	ML-KEM-512 + Falcon-512				1 434	1 697
NIKE-AKEM [AJKL23]	Swoosh [GdKQ+24]	**DR-Den***	✗	✓	> 221 184	> 221 184
FrodoKEX+ [CHN+24]	n/a	**DR-Den**	✗	✓	72	21 300
PQ-AKEM [GJK24]	NTRU-A + Gandalf [Antrag, Mitaka]	**HR-Den** & **DR-Den**	✗	✓	2 044	1 664
	bat_257_542 + Gandalf [Antrag, Mitaka]				1 749	1 417
Shadowfax [GHJ25]	X25519 + bat_257_542 + Gandalf [Antrag, Mitaka]	**HR-Den** & **DR-Den**	✓	✓	1 781	1 449
	X25519 + ML-KEM-512 + Gandalf [Falcon-512]				2 076	1 729

Deniability properties marked with an "∗" have not been formally proven in the respective work.

AKEM INSTANTIATIONS

Scheme	Instantiation	Deniability	Assumption		Size (in bytes)	
			pre-Q	post-Q	c	pk
DH-AKEM [ABH+21]	X25519	**DR-Den***	✓	✗	32	32
EtSt H-AKEM [AJKL23]	bat_257_542 + Antrag	✗	✗	✓	1 119	1 417
	ML-KEM-512 + Falcon-512				1 434	1 697
NIKE-AKEM [AJKL23]	Swoosh [GdKQ+24]	**DR-Den***	✗	✓	$> 221\,184$	$> 221\,184$
FrodoKEX+ [CHN+24]	n/a	**DR-Den**	✗	✓	72	21 300
PQ-AKEM [GJK24]	NTRU-A + Gandalf [Antrag, Mitaka]	**HR-Den** & **DR-Den**	✗	✓	2 044	1 664
	bat_257_542 + Gandalf [Antrag, Mitaka]				1 749	1 417
Shadowfax [GHJ25]	X25519 + bat_257_542 + Gandalf [Antrag, Mitaka]	**HR-Den** & **DR-Den**	✓	✓	1 781	1 449
	X25519 + ML-KEM-512 + Gandalf [Falcon-512]				2 076	1 729

Deniability properties marked with an "$*$" have not been formally proven in the respective work.

AKEM INSTANTIATIONS

Scheme	Instantiation	Deniability	Assumption		Size (in bytes)	
			pre-Q	post-Q	c	pk
DH-AKEM [ABH+21]	X25519	**DR-Den***	✓	✗	32	32
EtSt H-AKEM [AJKL23]	bat_257_542 + Antrag	✗	✗	✓	1 119	1 417
	ML-KEM-512 + Falcon-512				1 434	1 697
NIKE-AKEM [AJKL23]	Swoosh [GdKQ+24]	**DR-Den***	✗	✓	$> 221\,184$	$> 221\,184$
FrodoKEX+ [CHN+24]	n/a	**DR-Den**	✗	✓	72	21 300
PQ-AKEM [GJK24]	NTRU-A + Gandalf [Antrag, Mitaka]	**HR-Den** & **DR-Den**	✗	✓	2 044	1 664
	bat_257_542 + Gandalf [Antrag, Mitaka]				1 749	1 417
Shadowfax [GHJ25]	X25519 + bat_257_542 + Gandalf [Antrag, Mitaka]	**HR-Den** & **DR-Den**	✓	✓	1 781	1 449
	X25519 + ML-KEM-512 + Gandalf [Falcon-512]				2 076	1 729

Deniability properties marked with an "$*$" have not been formally proven in the respective work.

AKEM	Gen		Enc		Dec	
	kcc	**ms**	**kcc**	**ms**	**kcc**	**ms**

RSig	Gen		Sgn		Ver	
	kcc	**ms**	**kcc**	**ms**	**kcc**	**ms**

Cycle counts (in kilocycles) and time (in milliseconds) on a 3GHz Firestorm core of an Apple M1 Pro.

AKEM	Gen		Enc		Dec	
	kcc	**ms**	**kcc**	**ms**	**kcc**	**ms**
DH-AKEM [ABH+21]	208	0.07	628	0.21	419	0.14

RSig	Gen		Sgn		Ver	
	kcc	**ms**	**kcc**	**ms**	**kcc**	**ms**

Cycle counts (in kilocycles) and time (in milliseconds) on a 3GHz Firestorm core of an Apple M1 Pro.

AKEM	Gen		Enc		Dec	
	kcc	**ms**	**kcc**	**ms**	**kcc**	**ms**
DH-AKEM [ABH+21]	208	0.07	628	0.21	419	0.14
PQ-AKEM [GJK24]	24 748	8.25	1 215	0.41	332	0.11

RSig	Gen		Sgn		Ver	
	kcc	**ms**	**kcc**	**ms**	**kcc**	**ms**

Cycle counts (in kilocycles) and time (in milliseconds) on a 3GHz Firestorm core of an Apple M1 Pro.

AKEM	Gen		Enc		Dec	
	kcc	**ms**	**kcc**	**ms**	**kcc**	**ms**
DH-AKEM [ABH+21]	208	0.07	628	0.21	419	0.14
PQ-AKEM [GJK24]	24 748	8.25	1 215	0.41	332	0.11
Shadowfax [GHJ25]	24 770	8.27	1 846	0.62	746	0.25

RSig	Gen		Sgn		Ver	
	kcc	**ms**	**kcc**	**ms**	**kcc**	**ms**

Cycle counts (in kilocycles) and time (in milliseconds) on a 3GHz Firestorm core of an Apple M1 Pro.

AKEM	Gen		Enc		Dec	
	kcc	**ms**	**kcc**	**ms**	**kcc**	**ms**
DH-AKEM [ABH+21]	208	0.07	628	0.21	419	0.14
PQ-AKEM [GJK24]	24 748	8.25	1 215	0.41	332	0.11
Shadowfax [GHJ25]	24 770	8.27	1 846	0.62	746	0.25
RSig	**Gen**		**Sgn**		**Ver**	
	kcc	**ms**	**kcc**	**ms**	**kcc**	**ms**
Raptor [LAZ19]	71 420	23.81	7 980	2.66	505	0.17

Cycle counts (in kilocycles) and time (in milliseconds) on a 3GHz Firestorm core of an Apple M1 Pro.

AKEM	Gen		Enc		Dec	
	kcc	**ms**	**kcc**	**ms**	**kcc**	**ms**
DH-AKEM [ABH+21]	208	0.07	628	0.21	419	0.14
PQ-AKEM [GJK24]	24 748	8.25	1 215	0.41	332	0.11
Shadowfax [GHJ25]	24 770	8.27	1 846	0.62	746	0.25
RSig	**Gen**		**Sgn**		**Ver**	
	kcc	**ms**	**kcc**	**ms**	**kcc**	**ms**
Raptor [LAZ19]	71 420	23.81	7 980	2.66	505	0.17
Gandalf [GJK24]	13 145	4.38	1 110	0.37	95	0.03

Cycle counts (in kilocycles) and time (in milliseconds) on a 3GHz Firestorm core of an Apple M1 Pro.

SUMMARY

- Generic combiners may be too restrictive for PQC migration.
- SHADOWFAX: a deniable AKEM combiner from NIKE, KEM and RSig:
  - Compact instantiation with $|c| = 1.8$ KB and $|pk| = 1.5$ KB.
- Implementation of SHADOWFAX and GANDALF ring signature scheme [GJK24].

ia.cr/2025/154 — github.com/vincentvbh/shadowfax

phillip.gajland@{mpi-sp.org, rub.de}
p4i11ip

- Generic combiners may be too restrictive for PQC migration.
- Shadowfax: a deniable AKEM combiner from NIKE, KEM and RSig:
  - Compact instantiation with $|c| = 1.8$ KB and $|pk| = 1.5$ KB.
- Implementation of Shadowfax and Gandalf ring signature scheme [GJK24].

ia.cr/2025/154 — github.com/vincentvbh/shadowfax

phillip.gajland@{mpi-sp.org, rub.de}
p4i11ip

- Generic combiners may be too restrictive for PQC migration.
- Shadowfax: a deniable AKEM combiner from NIKE, KEM and RSig:
  - Compact instantiation with $|c| = 1.8$ KB and $|pk| = 1.5$ KB.
- Implementation of Shadowfax and Gandalf ring signature scheme [GJK24].

ia.cr/2025/154 — github.com/vincentvbh/shadowfax

phillip.gajland@{mpi-sp.org, rub.de}
p4i11ip

- Generic combiners may be too restrictive for PQC migration.
- Shadowfax: a deniable AKEM combiner from NIKE, KEM and RSig:
  - Compact instantiation with $|c| = 1.8$ KB and $|pk| = 1.5$ KB.
- Implementation of Shadowfax and Gandalf ring signature scheme [GJK24].

ia.cr/2025/154 — github.com/vincentvbh/shadowfax

phillip.gajland@{mpi-sp.org, rub.de}
p4i11ip

- Generic combiners may be too restrictive for PQC migration.
- SHADOWFAX: a deniable AKEM combiner from NIKE, KEM and RSig:
  - Compact instantiation with $|c| = 1.8$ KB and $|pk| = 1.5$ KB.
- Implementation of SHADOWFAX and GANDALF ring signature scheme [GJK24].

ia.cr/2025/154 — github.com/vincentvbh/shadowfax

phillip.gajland@{mpi-sp.org, rub.de}

p4i11ip

REFERENCES I

[ABH+21] Joël Alwen, Bruno Blanchet, Eduard Hauck, Eike Kiltz, Benjamin Lipp, and Doreen Riepel. Analysing the HPKE standard. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021, Part I*, volume 12696 of *Lecture Notes in Computer Science*, pages 87–116, Zagreb, Croatia, October 17–21, 2021. Springer, Cham, Switzerland.

[AJKL23] Joël Alwen, Jonas Janneck, Eike Kiltz, and Benjamin Lipp. The pre-shared key modes of HPKE. In Jian Guo and Ron Steinfeld, editors, *Advances in Cryptology – ASIACRYPT 2023, Part VI*, volume 14443 of *Lecture Notes in Computer Science*, pages 329–360, Guangzhou, China, December 4–8, 2023. Springer, Singapore, Singapore.

[App24] Apple. imessage with pq3: The new state of the art in quantum-secure messaging at scale. February 2024.

[BBCT22] Daniel J. Bernstein, Billy Bob Brumley, Ming-Shing Chen, and Nicola Tuveri. OpenSSLNTRU: Faster post-quantum TLS key exchange. In Kevin R. B. Butler and Kurt Thomas, editors, *USENIX Security 2022: 31st USENIX Security Symposium*, pages 845–862, Boston, MA, USA, August 10–12, 2022. USENIX Association.

[BBLW22] Richard Barnes, Karthikeyan Bhargavan, Benjamin Lipp, and Christopher A. Wood. Hybrid public key encryption. RFC 9180, feb 2022.

[BBR+23] Richard Barnes, Benjamin Beurdouche, Raphael Robert, Jon Millican, Emad Omara, and Katriel Cohn-Gordon. The messaging layer security (mls) protocol. RFC 9420, jul 2023.

[BCNS15] Joppe W. Bos, Craig Costello, Michael Naehrig, and Douglas Stebila. Post-quantum key exchange for the TLS protocol from the ring learning with errors problem. In *2015 IEEE Symposium on Security and Privacy*, pages 553–570, San Jose, CA, USA, May 17–21, 2015. IEEE Computer Society Press.

[Ber06] Daniel J. Bernstein. Curve25519: New Diffie-Hellman speed records. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *PKC 2006: 9th International Conference on Theory and Practice of Public Key Cryptography*, volume 3958 of *Lecture Notes in Computer Science*, pages 207–228, New York, NY, USA, April 24–26, 2006. Springer Berlin Heidelberg, Germany.

REFERENCES II

[CHN+24] Daniel Collins, Loïs Huguenin-Dumittan, Ngoc Khanh Nguyen, Nicolas Rolin, and Serge Vaudenay. K-waay: Fast and deniable post-quantum X3DH without ring signatures. In Davide Balzarotti and Wenyuan Xu, editors, *USENIX Security 2024: 33rd USENIX Security Symposium*, Philadelphia, PA, USA, August 14–16, 2024. USENIX Association.

[EFG+22] Thomas Espitau, Pierre-Alain Fouque, François Gérard, Mélissa Rossi, Akira Takahashi, Mehdi Tibouchi, Alexandre Wallet, and Yang Yu. Mitaka: A simpler, parallelizable, maskable variant of falcon. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology – EUROCRYPT 2022, Part III*, volume 13277 of *Lecture Notes in Computer Science*, pages 222–253, Trondheim, Norway, May 30 – June 3, 2022. Springer, Cham, Switzerland.

[ENS+23] Thomas Espitau, Thi Thu Quyen Nguyen, Chao Sun, Mehdi Tibouchi, and Alexandre Wallet. Antrag: Annular NTRU trapdoor generation - making mitaka as secure as falcon. In Jian Guo and Ron Steinfeld, editors, *Advances in Cryptology – ASIACRYPT 2023, Part VII*, volume 14444 of *Lecture Notes in Computer Science*, pages 3–36, Guangzhou, China, December 4–8, 2023. Springer, Singapore, Singapore.

[FJ24] Rune Fiedler and Christian Janson. A deniability analysis of signal's initial handshake pqxdh. *Proceedings on Privacy Enhancing Technologies*, 2024(4):907–928, October 2024.

[FKPY22] Pierre-Alain Fouque, Paul Kirchner, Thomas Pornin, and Yang Yu. BAT: Small and fast KEM over NTRU lattices. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2022(2):240–265, 2022.

[GdKQ+24] Phillip Gajland, Bor de Kock, Miguel Quaresma, Giulio Malavolta, and Peter Schwabe. SWOOSH: Efficient lattice-based non-interactive key exchange. In Davide Balzarotti and Wenyuan Xu, editors, *USENIX Security 2024: 33rd USENIX Security Symposium*, Philadelphia, PA, USA, August 14–16, 2024. USENIX Association.

[GHJ25] Phillip Gajland, Vincent Hwang, and Jonas Janneck. Shadowfax: Combiners for deniability. Cryptology ePrint Archive, Report 2025/154, 2025.

REFERENCES III

[GHP18] Federico Giacon, Felix Heuer, and Bertram Poettering. KEM combiners. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018: 21st International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 10769 of *Lecture Notes in Computer Science*, pages 190–218, Rio de Janeiro, Brazil, March 25–29, 2018. Springer, Cham, Switzerland.

[GJK24] Phillip Gajland, Jonas Janneck, and Eike Kiltz. Ring signatures for deniable AKEM: Gandalf's fellowship. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology – CRYPTO 2024, Part I*, volume 14920 of *Lecture Notes in Computer Science*, pages 305–338, Santa Barbara, CA, USA, August 18–22, 2024. Springer, Cham, Switzerland.

[GRSV25] Felix Günther, Michael Rosenberg, Douglas Stebila, and Shannon Veitch. Hybrid obfuscated key exchange and KEMs. Cryptology ePrint Archive, Report 2025/408, 2025.

[HR25] Julia Hesse and Michael Rosenberg. Pake combiners and efficient post-quantum instantiations. In Serge Fehr and Pierre-Alain Fouque, editors, *Advances in Cryptology – EUROCRYPT 2025*, pages 395–420, Cham, 2025. Springer Nature Switzerland.

[KS24] Ehren Kret and Rolfe Schmidt. The pqxdh key agreement protocol. 2024.

[KV19] Kris Kwiatkowski and Luke Valenta. The tls post-quantum experiment. 2019. https://blog.cloudflare.com/the-tls-post-quantum-experiment/.

[Lan16] Adam Langley. Cecpq1 results. 2016. https://www.imperialviolet.org/2016/11/28/cecpq1.html.

[Lan18] Adam Langley. Cecpq2. 2018. https://www.imperialviolet.org/2018/12/12/cecpq2.html.

[LAZ19] Xingye Lu, Man Ho Au, and Zhenfei Zhang. Raptor: A practical lattice-based (linkable) ring signature. In Robert H. Deng, Valérie Gauthier-Umaña, Martín Ochoa, and Moti Yung, editors, *ACNS 19: 17th International Conference on Applied Cryptography and Network Security*, volume 11464 of *Lecture Notes in Computer Science*, pages 110–130, Bogota, Colombia, June 5–7, 2019. Springer, Cham, Switzerland.

REFERENCES IV

[ML-24] Module-lattice-based key-encapsulation mechanism standard. National Institute of Standards and Technology NIST FIPS PUB 203, U.S. Department of Commerce, August 2024.

[MP16] Moxie Marlinspike and Trevor Perrin. The X3DH key agreement protocol (revision 1). Part of the Signal Protocol Documentation, 2016. https://signal.org/docs/specifications/x3dh/x3dh.pdf.

[PFH+22] Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. FALCON. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

[PST20] Christian Paquin, Douglas Stebila, and Goutam Tamvada. Benchmarking post-quantum cryptography in TLS. In Jintai Ding and Jean-Pierre Tillich, editors, *Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020*, pages 72–91, Paris, France, April 15–17, 2020. Springer, Cham, Switzerland.

[WR19] Bas Westerbaan and Cefan Daniel Rubin. Defending against future threats: Cloudflare goes post-quantum. 2019. https://blog.cloudflare.com/post-quantum-for-all/.

$b \xleftarrow{\$} \{0,1\}$

$pk_1, \ldots, pk_n$

$\ldots$

if $b = 0$

$(c, k) \xleftarrow{\$} \mathsf{Enc}(sk_s, pk_r)$

if $b = 1$

$(c, k) \xleftarrow{\$} \mathsf{Sim}(\ldots)$

$\texttt{Chall}(s \in [n], r \in [n])$

(c, k)

if $b = b'$

return win

b'

Honest receiver: $\mathsf{Sim}(\emptyset), \mathsf{A}(sk_s)$

Dishonest receiver: $\mathsf{Sim}(sk_r), \mathsf{A}(sk_s, sk_r)$

$b \xleftarrow{\$} \{0,1\}$

$\texttt{Encps}(s \in [n], pk)$

$(c,k) \xleftarrow{\$} \mathsf{Enc}(sk_s, pk)$

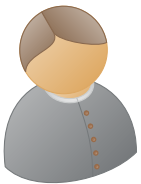
$(c,k) \xleftarrow{\$} \mathsf{Enc}(sk_s, pk_r)$

if $b = 0$

  continue

if $b = 1$

  $k \xleftarrow{\$} \mathcal{K}$



$\texttt{Decps}(pk, r \in [n], c)$

$k \leftarrow \mathsf{Dec}(pk, sk_r, c)$

if $b = b'$

  return win

$\texttt{Chall}(sk, r \in [n])/\texttt{Chall}(s \in [n], r \in [n])$

b'

(c,k)

ADDITIONAL SLIDES

Games $(\mathrm{n}, Q_{\mathtt{Enc}}, Q_{\mathtt{Chl}})$-**Out-Aut**$_{\mathsf{AKEM}}(\mathsf{A})$
$(\mathrm{n}, Q_{\mathtt{Enc}}, Q_{\mathtt{Dec}}, Q_{\mathtt{Chl}})$-**Ins-Aut**$_{\mathsf{AKEM}}(\mathsf{A})$

for $i \in [n]$
  $(sk_i, pk_i) \xleftarrow{\$} \mathsf{Gen}$
$\mathcal{D} \leftarrow \emptyset$
$b \xleftarrow{\$} \{0,1\}$
$b' \xleftarrow{\$} \mathsf{A}^{\mathtt{Encps},\mathtt{Chall}}(pk_1, \ldots, pk_n)$ // **Out-Aut**
$b' \xleftarrow{\$} \mathsf{A}^{\mathtt{Encps},\mathtt{Decps},\mathtt{Chall}}(pk_1, \ldots, pk_n)$ // **Ins-Aut**
return $[\![b = b']\!]$

Oracle $\mathtt{Chall}(pk, r \in [n], c)$ // **Out-Aut**
if $\exists\, k : (pk, pk_r, c, k) \in \mathcal{D}$
  return k
$k \leftarrow \mathsf{Dec}(pk, sk_r, c)$
if $b = 0$
  continue
if $b = 1 \wedge pk \in \{pk_1, \ldots, pk_n\} \wedge k \neq \bot$
  $k \xleftarrow{\$} \mathcal{K}$
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(pk, pk_r, c, k)\}$
return k

Oracle $\mathtt{Encps}(s \in [n], pk)$
$(c, k) \xleftarrow{\$} \mathsf{Enc}(sk_s, pk)$
$\mathcal{D} \leftarrow \mathcal{D} \cup \{(pk_s, pk, c, k)\}$
return (c, k)

Oracle $\mathtt{Decps}(pk, r \in [n], c)$ // **Ins-Aut**
$k \leftarrow \mathsf{Dec}(pk, sk_r, c)$
return k

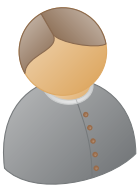
Oracle $\mathtt{Chall}(s \in [n], sk, c)$ // **Ins-Aut**
if $\exists\, k : (pk_s, \mu(sk), c, k) \in \mathcal{D}$
  return k
$k \leftarrow \mathsf{Dec}(pk_s, sk, c)$
if $b = 0$
  continue
if $b = 1 \wedge k \neq \bot$
  $k \xleftarrow{\$} \mathcal{K}$
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(pk_s, \mu(sk), c, k)\}$
return k

1.

AKEM security notion: outsider-aut [ABH+21]

$b \xleftarrow{\$} \{0,1\}$

if $\exists$ previous query
  return previous k
$k \leftarrow \mathsf{Dec}(pk, sk_r, c)$
if $b = 1 \wedge pk$ honest $\wedge\ k \neq \perp$
  $k \xleftarrow{\$} \mathcal{K}$
return k



$pk_1, \ldots, pk_n$

$\texttt{Encps}(s \in [n], pk)$

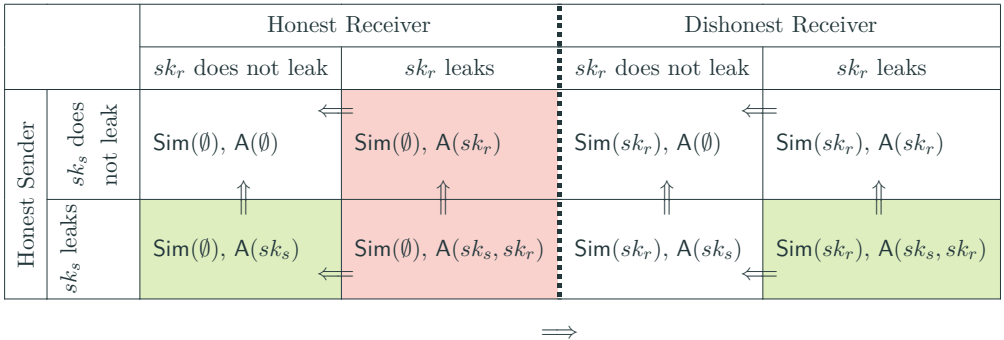
(c, k)

$\texttt{Chall}(pk, r \in [n], c)$

k

b'

if $b = b'$
  return win



- E.g. A non-interactive key exchange (NIKE) used for implicit authentication is dishonest receiver deniable, but not honest receiver deniable. [1]

[1] A single NIKE does not suffice as an AKEM.

Games $(\mathrm{n}, Q_{\mathtt{Chl}})$-$\mathbf{DR}$-$\mathbf{Den}_{\mathsf{AKEM},\mathsf{Sim}}(\mathsf{A})$

$(\mathrm{n}, Q_{\mathtt{Chl}})$-$\mathbf{HR}$-$\mathbf{Den}_{\mathsf{AKEM},\mathsf{Sim}}(\mathsf{A})$

$\mathcal{R}, \mathcal{C} \leftarrow \emptyset$
for $i \in [n]$
  $(sk_i, pk_i) \xleftarrow{\$} \mathsf{Gen}$
$b \xleftarrow{\$} \{0,1\}$
$b' \leftarrow \mathsf{A}^{\mathtt{Rev},\mathtt{Chall}}(pk_1, \ldots, pk_n)$
if $\mathcal{R} \cap \mathcal{C} \neq \emptyset$ // **HR-Den**
  return $b \xleftarrow{\$} \{0,1\}$ // **HR-Den**
return $[\![b = b']\!]$

$\mathtt{Rev}(i \in [n])$
$\mathcal{R} \leftarrow \mathcal{R} \cup \{i\}$
return sk_i

Oracle $\mathtt{Chall}(s \in [n], r \in [n])$
$\mathcal{C} \leftarrow \mathcal{C} \cup \{r\}$
$(c, k) \xleftarrow{\$} \mathsf{Enc}(sk_s, pk_r)$
if $b = 0$
  continue
if $b = 1$
  $(c, k) \xleftarrow{\$} \mathsf{Sim}(pk_s, pk_r, sk_r)$ // **DR-Den**
  $(c, k) \xleftarrow{\$} \mathsf{Sim}(pk_s, pk_r)$ // **HR-Den**
return (c, k)