On IND-qCCA security in the ROM and its applications

Loïs Huguenin-Dumittan and Serge Vaudenay

Ecole Polytechnique Fédérale de Lausanne, Switzerland

Eurocrypt 2022

Outline

IND-qCCA KEM

- KEM: $(\mathsf{Gen}, \mathsf{Encaps}, \mathsf{Decaps})$

IND-qCCA KEM

- KEM: (Gen, Encaps, Decaps)
  1. $(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{Gen}$
  2. $(K, \mathsf{ct}) \leftarrow\$ \mathsf{Encaps}(\mathsf{pk})$
  3. $K' \leftarrow \mathsf{Decaps}(\mathsf{sk}, \mathsf{ct})$

IND-qCCA KEM

- KEM:
  (Gen, Encaps, Decaps).

IND-qCCA KEM

- KEM:
  (Gen, Encaps, Decaps).

- Distinguish real key from random key

(sk, pk)

$K_0 \in_U \mathcal{K},\ b \in_U \{0,1\}$
$K_1, \mathsf{ct}^* \leftarrow \mathsf{Encaps}(\mathsf{pk})$

ct^*, K_b

b'

win iff $b = b'$

IND-qCCA KEM

(sk, pk)

- KEM:
  (Gen, Encaps, Decaps).
- Distinguish real key from random key
- with *q* (*constant!*) decapsulation queries.

ct^*, K_b

$K_0 \in_U \mathcal{K},\ b \in_U \{0,1\}$
$K_1, \mathsf{ct}^* \leftarrow \mathsf{Encaps}(\mathsf{pk})$

$\mathsf{ct}_1 \neq \mathsf{ct}^*$

$\mathsf{Decaps}(\mathsf{sk}, \mathsf{ct}_1)$

$\vdots$

$\mathsf{ct}_q \neq \mathsf{ct}^*$

$\mathsf{Decaps}(\mathsf{sk}, \mathsf{ct}_q)$

b'

win iff $b = b'$

IND-qCCA history

- Defined by Cramer et al. in 2007.

IND-qCCA history

- Defined by Cramer et al. in 2007.
- CPA $\rightarrow$ qCCA transforms in the standard model exists but are inefficient.

IND-qCCA history

- Defined by Cramer et al. in 2007.
- CPA $\rightarrow$ qCCA transforms in the standard model exists but are inefficient.
- Hasn't been very popular (between IND-CPA and IND-CCA, Diffie-Hellman was sufficient).

IND-qCCA history

- Defined by Cramer et al. in 2007.
- CPA $\rightarrow$ qCCA transforms in the standard model exists but are inefficient.
- Hasn't been very popular (between IND-CPA and IND-CCA, Diffie-Hellman was sufficient).
- PQ and Forward secrecy have changed the game:
  1. KEMs instead of Diffie-Hellman.
  2. Ephemeral keys instead of static keys.

Motivation: New protocols use IND-1CCA KEMs

EPFL

- PQ TLS 1.3 is secure with IND-1CCA KEM.

Client | **Server**

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

$\xleftrightarrow{\text{SYN, SYN ACK}}$

$\mathsf{pk}_e \leftarrow$ IND-1CCA KeyGen()

$\xrightarrow{\mathsf{pk}_e}$

$K_e, \mathsf{ct}_e \leftarrow$ IND-1CCA Encaps(pk_e)

$\mathit{keys} \leftarrow$ KeySched(K_e)

$\sigma \leftarrow$ Sign(sk_s, transcript)

$\xleftarrow{\mathsf{ct}_e, \text{AEAD}(\text{cert}, \mathsf{pk}_s, \sigma)}$

$K_e \leftarrow$ IND-1CCA Decaps($\mathsf{sk}_e, \mathsf{ct}_e$)

$\mathit{keys} \leftarrow$ KeySched(K_e)

$\xleftrightarrow{\text{AEAD}_{\mathit{keys}}(\text{key conf., data})}$

Motivation: New protocols use IND-1CCA KEMs

- PQ TLS 1.3 is secure with IND-1CCA KEM.
- KEMTLS (*Schwabe et al., 2020*) uses IND-1CCA KEM_e .

Client **Server**

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

$\xleftrightarrow{\text{SYN, SYN ACK}}$

$\mathsf{pk}_e \leftarrow$ IND-1CCA KeyGen()

$\xrightarrow{\mathsf{pk}_e}$

$K_e, \mathsf{ct}_e \leftarrow$ IND-1CCA Encaps(pk_e)

$\mathsf{tk}_c, \mathsf{tk}_s \leftarrow \mathsf{KeySched}(K_e)$

$\xleftarrow{\mathsf{ct}_e,\ \mathsf{AEAD}_{\mathsf{tk}_s}(\mathsf{cert}, \mathsf{pk}_s)}$

$K_e \leftarrow$ IND-1CCA Decaps$(\mathsf{sk}_e, \mathsf{ct}_e)$

$K_s, \mathsf{ct}_s \leftarrow \mathsf{Encaps}(\mathsf{pk}_s)$

$\mathsf{tk}_c, \mathsf{tk}_s, \mathit{keys} \leftarrow \mathsf{KeySched}(K_e, K_s)$

$\xrightarrow{\mathsf{AEAD}_{\mathsf{tk}_c}(\mathsf{ct}_s)}$

$K_s \leftarrow \mathsf{Decaps}(\mathsf{sk}_s, \mathsf{ct}_s)$

$\mathit{keys} \leftarrow \mathsf{KeySched}(K_e, K_s)$

$\xleftrightarrow{\mathsf{AEAD}_{\mathit{keys}}(\text{key conf., data})}$

Motivation: New protocols use IND-1CCA KEMs

- PQ TLS 1.3 is secure with IND-1CCA KEM.
- KEMTLS (*Schwabe et al., 2020*) uses IND-1CCA KEM$_e$.
- PQ variant of X3DH uses IND-1CCA KEMs (e.g. *Brendel et al., 2022*).

Alice		**Bob**
$(\mathsf{pk}_A^{\mathsf{dvs}}, \mathsf{sk}_A^{\mathsf{dvs}})$		$(\mathsf{pk}_B^{\mathsf{1cca,cca,dvs}}, \mathsf{sk}_B^{\mathsf{1cca,cca,dvs}})$
	$\xleftarrow{\mathsf{pk}_B^{\mathsf{1cca}},\, \mathsf{pk}_B^{\mathsf{cca}},\, \mathsf{pk}_B^{\mathsf{dvs}}}$	
	$\xrightarrow{\mathsf{pk}_A^{\mathsf{dvs}}}$	
$K_e, \mathsf{ct}_e \leftarrow \text{IND-1CCA Encaps}(\mathsf{pk}_B^{1cca})$ $K_s, \mathsf{ct}_s \leftarrow \text{IND-CCA Encaps}(\mathsf{pk}_B^{cca})$ $\sigma_{\mathsf{dvs}} \leftarrow \mathsf{Sign}(\mathsf{sk}_A^{\mathsf{DVS}}, \mathsf{pk}_B^{\mathsf{dvs}}, \mathsf{transcript})$		
	$\xrightarrow{\mathsf{ct}_e, \mathsf{ct}_s, \sigma}$	
		$\mathsf{Vrfy}(\mathsf{pk}_A^{\mathsf{dvs}}, \mathsf{sk}_B^{\mathsf{dvs}}, \mathsf{transcript})$ $K_e \leftarrow \text{IND-1CCA Decaps}(\mathsf{sk}_B^{1cca}, \mathsf{ct}_e)$ $K_s \leftarrow \text{IND-1CCA Decaps}(\mathsf{sk}_B^{cca}, \mathsf{ct}_s)$
$K \leftarrow \mathsf{KDF}(K_e, K_s, \mathsf{transcript})$		$K \leftarrow \mathsf{KDF}(K_e, K_s, \mathsf{transcript})$

Motivation

- In new protocols: IND-CPA might not be enough but IND-CCA is not necessary for ephemeral KEMs $\Rightarrow$ IND-1CCA.

Motivation

- In new protocols: IND-CPA might not be enough but IND-CCA is not necessary for ephemeral KEMs $\Rightarrow$ IND-1CCA.

Can we build more efficient IND-1CCA KEMs than IND-CCA ones? I.e. without Fujisaki-Okamoto and re-encryption.

Contributions

- We give two very simple/efficient OW-CPA PKE $\rightarrow$ IND-qCCA KEM transforms secure in the (Q)ROM.

Contributions

- We give two very simple/efficient OW-CPA PKE $\rightarrow$ IND-qCCA KEM transforms secure in the (Q)ROM.
- Compared to Fujisaki-Okamoto-like transforms:
  1. No de-randomization.
  2. No re-encryption.
  3. Decapsulation much faster than with FO-derived KEMs.

Contributions

- We give two very simple/efficient OW-CPA PKE $\rightarrow$ IND-qCCA KEM transforms secure in the (Q)ROM.
- Compared to Fujisaki-Okamoto-like transforms:
  1. No de-randomization.
  2. No re-encryption.
  3. Decapsulation much faster than with FO-derived KEMs.
- We show that PQ TLS 1.3 is secure in the ROM if KEM is only CPA secure (but bound is very loose).

Contributions

- We give two very simple/efficient OW-CPA PKE $\rightarrow$ IND-qCCA KEM transforms secure in the (Q)ROM.
- Compared to Fujisaki-Okamoto-like transforms:
  1. No de-randomization.
  2. No re-encryption.
  3. Decapsulation much faster than with FO-derived KEMs.
- We show that PQ TLS 1.3 is secure in the ROM if KEM is only CPA secure (but bound is very loose).
- $\Rightarrow$ (classical) TLS 1.3 is secure if CDH holds (no need for PRF-ODH).

Outline

Does the trivial PKE $\rightarrow$ KEM transform work?

$\mathsf{Gen}()$	$\mathsf{Encaps}(\mathsf{pk})$	$\mathsf{Decaps}(\mathsf{sk}, \mathsf{ct})$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{gen}^{\mathsf{pke}}()$	$\sigma \leftarrow\!\!\$\ \mathcal{M}$	$\sigma' \leftarrow \mathsf{dec}^{\mathsf{pke}}(\mathsf{sk}, \mathsf{ct})$
return $(\mathsf{pk}, \mathsf{sk})$	$\mathsf{ct} \leftarrow\!\!\$\ \mathsf{enc}^{\mathsf{pke}}(\mathsf{pk}, \sigma)$	**return** $H(\sigma')$
	$K \leftarrow H(\sigma)$	
	return K, ct	

Figure: Trivial transform.

Does it output a IND-qCCA KEM if PKE is OW-CPA?

~~Does the trivial PKE $\rightarrow$ KEM work?~~

EPFL

$\mathsf{Gen}()$	$\mathsf{Encaps}(\mathsf{pk})$	$\mathsf{Decaps}(\mathsf{sk}, \mathsf{ct})$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{gen}^{\mathsf{pke}}()$	$\sigma \leftarrow\$ \mathcal{M}$	$\sigma' \leftarrow \mathsf{dec}^{\mathsf{pke}}(\mathsf{sk}, \mathsf{ct})$
return $(\mathsf{pk}, \mathsf{sk})$	$\mathsf{ct} \leftarrow\$ \mathsf{enc}^{\mathsf{pke}}(\mathsf{pk}, \sigma)$	**return** $H(\sigma')$
	$K \leftarrow H(\sigma)$	
	return K, ct	

Figure: Trivial transform.

Does it output a IND-qCCA KEM if PKE is OW-CPA?

No. E.g. in most PQ schemes, $\mathcal{O}^{\mathsf{Decaps}}(\mathsf{ct}^* + \delta) \rightarrow H(\sigma^*)$ (i.e. the real key) for small δ.

Transform 1: T_{CH}

$\mathsf{Gen}()$	$\mathsf{Encaps}(\mathsf{pk})$	$\mathsf{Decaps}(\mathsf{sk}, (\mathsf{ct}, \mathsf{tag}))$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\$\, \mathsf{gen}^{\mathsf{pke}}()$	$\sigma \leftarrow\!\$\, \mathcal{M}$	$\sigma' \leftarrow \mathsf{dec}^{\mathsf{pke}}(\mathsf{sk}, \mathsf{ct})$
return $(\mathsf{pk}, \mathsf{sk})$	$\mathsf{ct} \leftarrow\!\$\, \mathsf{enc}^{\mathsf{pke}}(\mathsf{pk}, \sigma)$	**if** $H'(\sigma', \mathsf{ct}) \neq \mathsf{tag}$:
	$\mathsf{tag} \leftarrow H'(\sigma, \mathsf{ct})$	**return** $\perp$
	$K \leftarrow H(\sigma)$	**return** $H(\sigma')$
	return $K, (\mathsf{ct}, \mathsf{tag})$	

Figure: T_{CH}.

- Fix: Add confirmation hash to ciphertext.

Transform 1: T_{CH}

EPFL

$\mathsf{Gen}()$	$\mathsf{Encaps}(\mathsf{pk})$	$\mathsf{Decaps}(\mathsf{sk}, (\mathsf{ct}, \mathsf{tag}))$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{gen}^{\mathsf{pke}}()$	$\sigma \leftarrow\$ \mathcal{M}$	$\sigma' \leftarrow \mathsf{dec}^{\mathsf{pke}}(\mathsf{sk}, \mathsf{ct})$
return $(\mathsf{pk}, \mathsf{sk})$	$\mathsf{ct} \leftarrow\$ \mathsf{enc}^{\mathsf{pke}}(\mathsf{pk}, \sigma)$	**if** $H'(\sigma', \mathsf{ct}) \neq \mathsf{tag}$:
	$\mathsf{tag} \leftarrow H'(\sigma, \mathsf{ct})$	**return** $\perp$
	$K \leftarrow H(\sigma)$	**return** $H(\sigma')$
	return $K, (\mathsf{ct}, \mathsf{tag})$	

Figure: T_{CH}.

- Fix: Add confirmation hash to ciphertext.
- Attack thwarted as $\mathcal{A}$ would need $(\mathsf{ct}^* + \delta, H'(\sigma^*, \mathsf{ct}^* + \delta))$.

Security proof idea and OW-PCA

Proof idea:

- Similar to REACT[1], T_{CH} does: OW-PCA PKE $\rightarrow$ IND-(q)CCA KEM.

[1] Okamoto and Pointcheval, 2001

OW-PCA

EPFL

(sk, pk)

$\mathsf{pt}^* \in_U \mathcal{M}$

$\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{pt}^*)$

$\xleftarrow{\mathsf{ct}^*}$

$\xrightarrow{\mathsf{ct}_1,\, \mathsf{pt}_1}$

$\xleftarrow{1_{\mathsf{Dec}(\mathsf{sk},\mathsf{ct}_1)=\mathsf{pt}_1}}$

$\vdots$

$\xrightarrow{\mathsf{ct}_q,\, \mathsf{pt}_q}$

$\xleftarrow{1_{\mathsf{Dec}(\mathsf{sk},\mathsf{ct}_q)=\mathsf{pt}_q}}$

$\xrightarrow{\mathsf{pt}'}$

win iff $\mathsf{pt}' = \mathsf{pt}^*$

Security proof idea and OW-PCA

Proof idea:

- Similar to REACT[1], T_{CH} does: OW-PCA PKE $\rightarrow$ IND-(q)CCA KEM.
- OW-PCA with q queries $=$ OW-CPA with a loss of q security bits.

[1] Okamoto and Pointcheval, 2001

Security proof idea and OW-PCA

EPFL

Proof idea:

- Similar to REACT[1], T_{CH} does: OW-PCA PKE $\rightarrow$ IND-(q)CCA KEM.
- OW-PCA with q queries $=$ OW-CPA with a loss of q security bits.

Bound

$$\mathsf{Adv}_{\mathsf{KEM}}^{\mathrm{ind-qcca}}(\mathcal{A}) \leq \mathsf{negl} + (q_H + q_{H'} + q) \cdot 2^q \cdot \mathsf{Adv}_{\mathsf{PKE}}^{\mathrm{ow-cpa}}(\mathcal{B}) \ .$$

- In practice: Only suitable for small q (e.g. IND-1CCA KEM).

[1] Okamoto and Pointcheval, 2001

Transform 2: $\mathsf{T_H}$

Gen()	Encaps(pk)	Decaps(sk, ct)
$(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{gen}^{\mathsf{pke}}()$	$\sigma \leftarrow\$ \mathcal{M}$	$\sigma' \leftarrow \mathsf{dec}^{\mathsf{pke}}(\mathsf{sk}, \mathsf{ct})$
return $(\mathsf{pk}, \mathsf{sk})$	$\mathsf{ct} \leftarrow\$ \mathsf{enc}^{\mathsf{pke}}(\mathsf{pk}, \sigma)$	**return** $H(\sigma', \mathsf{ct})$
	$K \leftarrow H(\sigma, \mathsf{ct})$	
	return K, ct	

Figure: $\mathsf{T_H}$ transform.

- Hash (σ, ct) in the key and not in the tag.

Transform 2: $\mathsf{T_H}$

Gen()	Encaps(pk)	Decaps(sk, ct)
$(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{gen}^{\mathsf{pke}}()$	$\sigma \leftarrow\$ \mathcal{M}$	$\sigma' \leftarrow \mathsf{dec}^{\mathsf{pke}}(\mathsf{sk}, \mathsf{ct})$
return $(\mathsf{pk}, \mathsf{sk})$	$\mathsf{ct} \leftarrow\$ \mathsf{enc}^{\mathsf{pke}}(\mathsf{pk}, \sigma)$	**return** $H(\sigma', \mathsf{ct})$
	$K \leftarrow H(\sigma, \mathsf{ct})$	
	return K, ct	

Figure: $\mathsf{T_H}$ transform.

- Hash (σ, ct) in the key and not in the tag.
- Previous attack doesn't work: $\mathcal{O}^{\mathsf{Decaps}}(\mathsf{ct}^* + \delta) \neq H(\sigma^*, \mathsf{ct}^*)$.

Transform 2: $\mathsf{T_H}$

EPFL

$\mathsf{Gen}()$	$\mathsf{Encaps}(\mathsf{pk})$	$\mathsf{Decaps}(\mathsf{sk}, \mathsf{ct})$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\$\, \mathsf{gen}^{\mathsf{pke}}()$	$\sigma \leftarrow\!\$\, \mathcal{M}$	$\sigma' \leftarrow \mathsf{dec}^{\mathsf{pke}}(\mathsf{sk}, \mathsf{ct})$
return $(\mathsf{pk}, \mathsf{sk})$	$\mathsf{ct} \leftarrow\!\$\, \mathsf{enc}^{\mathsf{pke}}(\mathsf{pk}, \sigma)$	**return** $H(\sigma', \mathsf{ct})$
	$K \leftarrow H(\sigma, \mathsf{ct})$	
	return K, ct	

Figure: $\mathsf{T_H}$ transform.

- Hash (σ, ct) in the key and not in the tag.
- Previous attack doesn't work: $\mathcal{O}^{\mathsf{Decaps}}(\mathsf{ct}^* + \delta) \neq H(\sigma^*, \mathsf{ct}^*)$.
- KEM variant of $\mathsf{T_H}$ preserves the "symmetric" structure of underlying KEM. I.e. ct is independent of pk (e.g. DH/SIDH).

Transform 2: $\mathsf{T_H}$

Gen()	Encaps(pk)	Decaps(sk, ct)
$(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{Gen}^{\mathsf{kem}}()$	$\mathsf{ct}, \sigma \leftarrow\!\!\$\ \mathsf{Encaps}^{\mathsf{kem}}(\mathsf{pk})$	$\sigma' \leftarrow \mathsf{Decaps}^{\mathsf{kem}}(\mathsf{sk}, \mathsf{ct})$
return $(\mathsf{pk}, \mathsf{sk})$	$K \leftarrow H(\sigma, \mathsf{ct})$	**return** $H(\sigma', \mathsf{ct})$
	return K, ct	

Figure: $\mathsf{T_H}$ transform (KEM variant).

- Hash (σ, ct) in the key and not in the tag.
- Previous attack doesn't work: $\mathcal{O}^{\mathsf{Decaps}}(\mathsf{ct}^* + \delta) \neq H(\sigma^*, \mathsf{ct}^*)$.
- KEM variant of $\mathsf{T_H}$ preserves the "symmetric" structure of underlying KEM. I.e. ct is independent of pk (e.g. DH/SIDH).

$\mathsf{T_H}$ security

Bound

$$\mathsf{Adv}_{\mathsf{KEM}}^{\mathrm{ind-qcca}}(\mathcal{A}) \leq \delta + ((q_H+1)(q_H+2))^q \cdot \mathsf{Adv}_{\mathsf{PKE}}^{\mathrm{ow-cpa}}(\mathcal{B}) \ .$$

- In practice: secure for $q = 1$.

T_H security

Bound

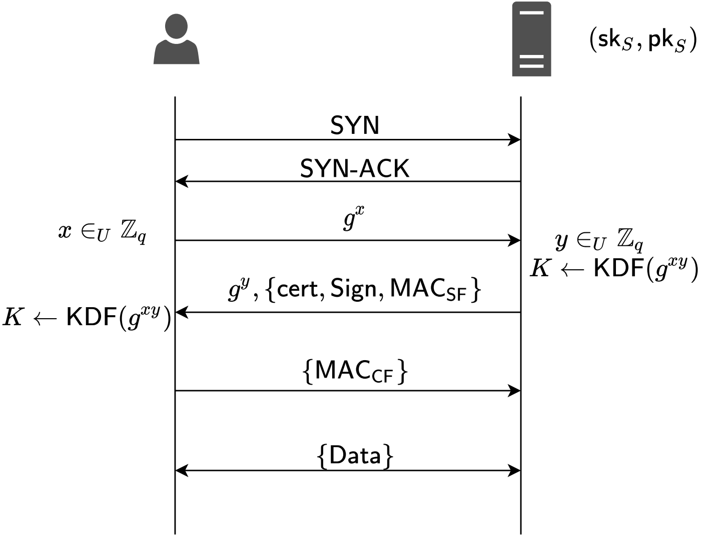
$$\mathsf{Adv}^{\mathrm{ind-qcca}}_{\mathsf{KEM}}(\mathcal{A}) \leq \delta + ((q_H+1)(q_H+2))^q \cdot \mathsf{Adv}^{\mathrm{ow-cpa}}_{\mathsf{PKE}}(\mathcal{B}) \ .$$

- In practice: secure for $q = 1$.
- Proof requires RO programming and careful guessing in the reduction (factor needs to be exponential in q not q_H!).

Outline

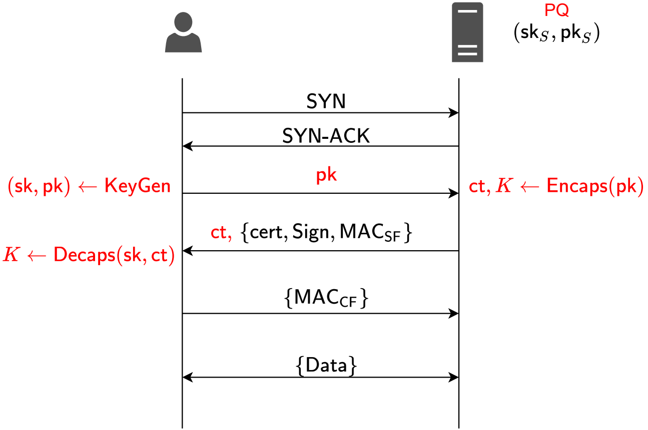
(Classical) TLS 1.3

EPFL



PQ TLS 1.3

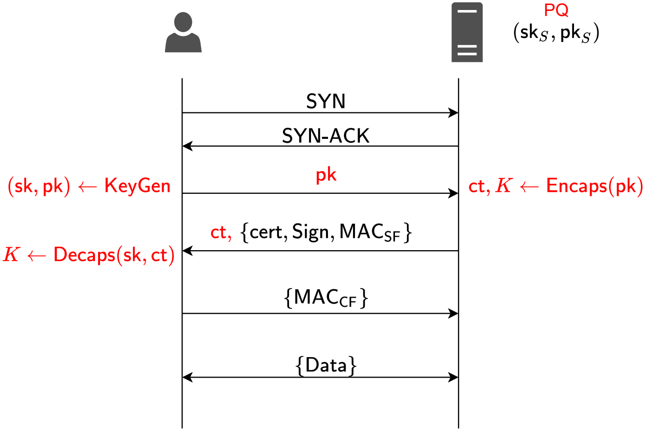
- Write DH as a KEM.



[2] *Dowling et al., 2020*

PQ TLS 1.3

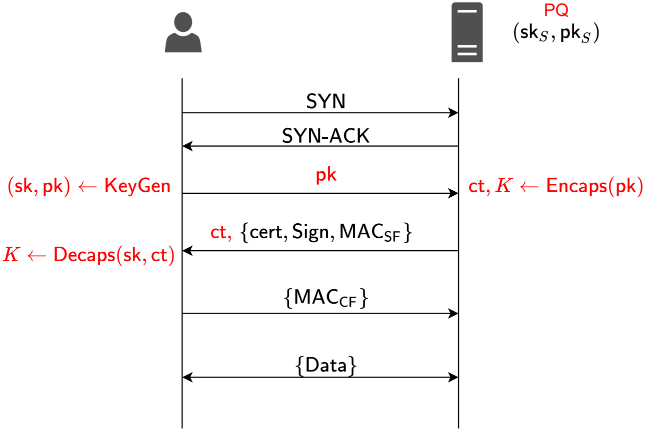
- Write DH as a KEM.
- IND-1CCA KEM can be used (trivial from the original proof[2]).



[2] *Dowling et al., 2020*

PQ TLS 1.3

- Write DH as a KEM.
- IND-1CCA KEM can be used (trivial from the original proof[2]).
- We show a OW/IND-CPA KEM can be used (in the ROM).



[2] *Dowling et al., 2020*

A note on the security model

- We use the same security model as *Dowling et al., 2020* (i.e. *MultiStage* security).

A note on the security model

- We use the same security model as *Dowling et al., 2020* (i.e. *MultiStage* security).
- In the model, $\mathcal{A}$ can send, receive, expose, etc.

A note on the security model

- We use the same security model as *Dowling et al., 2020* (i.e. *MultiStage* security).
- In the model, $\mathcal{A}$ can send, receive, expose, etc.
- When a key is derived and ready for use, it is *accepted*.

A note on the security model

- We use the same security model as *Dowling et al., 2020* (i.e. *MultiStage* security).
- In the model, $\mathcal{A}$ can send, receive, expose, etc.
- When a key is derived and ready for use, it is *accepted*.
- On acceptance of a key, the protocol pauses and $\mathcal{A}$ can call oracles before continuing.

OW-CPA KEM $\Rightarrow$ MultiStage TLS 1.3

TLS 1.3 with KEM Key Schedule

Client **Server**

$\xrightarrow{\mathsf{pk}}$

$\xleftarrow{\mathsf{ct}}$

$K \leftarrow \mathsf{Decaps}(\mathsf{sk}, \mathsf{ct})$

$\mathsf{HS} \leftarrow \mathsf{HKDF.Ext}(\ldots, K)$
$\mathsf{CHTS} \leftarrow \mathsf{HKDF.Exp}_4(\mathsf{HS}, G(\mathsf{ct}, \ldots))$
$\mathsf{SHTS} \leftarrow \mathsf{HKDF.Exp}_5(\mathsf{HS}, G(\mathsf{ct}, \ldots))$
$\mathsf{dHS} \leftarrow \mathsf{HKDF.Exp}_0(\mathsf{HS}, cnst)$

.... (Stage 1) accept $\mathsf{tk_c} \leftarrow \mathsf{HKDF.TK}(\mathsf{CHTS})$ ....

.... (Stage 2) accept $\mathsf{tk_s} \leftarrow \mathsf{HKDF.TK}(\mathsf{SHTS})$ ....

...

$\mathsf{fk}_S \leftarrow \mathsf{HKDF.Exp}_6(\mathsf{SHTS})$

$\{\mathsf{SF}\} : \mathsf{MAC}(\mathsf{fk}_S, T_7)$

$\xleftarrow{\{\mathsf{SF}\}}$

if $\mathsf{MAC}(\mathsf{fk}_S, T_7) \neq \mathsf{SF}$: **abort**

...

- Assume $\mathsf{HKDF.Ext}, \mathsf{HKDF.Exp}_i$ and G are ROs.

OW-CPA KEM $\Rightarrow$ MultiStage TLS 1.3

TLS 1.3 with KEM Key Schedule

Client **Server**

$\xrightarrow{\mathsf{pk}}$

$\xleftarrow{\mathsf{ct}}$

$K \leftarrow \mathsf{Decaps}(\mathsf{sk}, \mathsf{ct})$

$\mathsf{HS} \leftarrow \mathsf{HKDF.Ext}(\ldots, K)$
$\mathsf{CHTS} \leftarrow \mathsf{HKDF.Exp}_4(\mathsf{HS}, G(\mathsf{ct}, \ldots))$
$\mathsf{SHTS} \leftarrow \mathsf{HKDF.Exp}_5(\mathsf{HS}, G(\mathsf{ct}, \ldots))$
$\mathsf{dHS} \leftarrow \mathsf{HKDF.Exp}_0(\mathsf{HS}, \mathit{cnst})$

.... (Stage 1) accept $\mathsf{tk}_c \leftarrow \mathsf{HKDF.TK}(\mathsf{CHTS})$ ....

.... (Stage 2) accept $\mathsf{tk}_s \leftarrow \mathsf{HKDF.TK}(\mathsf{SHTS})$ ....

...

$\mathsf{fk}_S \leftarrow \mathsf{HKDF.Exp}_6(\mathsf{SHTS})$

$\{\mathsf{SF}\} : \mathsf{MAC}(\mathsf{fk}_S, T_7)$

$\xleftarrow{\{\mathsf{SF}\}}$

if $\mathsf{MAC}(\mathsf{fk}_S, T_7) \neq \mathsf{SF}$: **abort**

...

- Assume HKDF.Ext, HKDF.Exp$_i$ and G are ROs.
- In the OW-CPA reduction, we need to simulate the client receiving *one* ct.

OW-CPA KEM $\Rightarrow$ MultiStage TLS 1.3

TLS 1.3 with KEM Key Schedule

Client **Server**

$\xrightarrow{\mathsf{pk}}$

$\xleftarrow{\mathsf{ct}}$

$K \leftarrow \mathsf{Decaps}(\mathsf{sk}, \mathsf{ct})$

$\mathsf{HS} \leftarrow \mathsf{HKDF.Ext}(\ldots, K)$
$\mathsf{CHTS} \leftarrow \mathsf{HKDF.Exp}_4(\mathsf{HS}, G(\mathsf{ct}, \ldots))$
$\mathsf{SHTS} \leftarrow \mathsf{HKDF.Exp}_5(\mathsf{HS}, G(\mathsf{ct}, \ldots))$
$\mathsf{dHS} \leftarrow \mathsf{HKDF.Exp}_0(\mathsf{HS}, \mathit{cnst})$

......................(Stage 1) accept $\mathsf{tk_c} \leftarrow \mathsf{HKDF.TK(CHTS)}$......................

......................(Stage 2) accept $\mathsf{tk_s} \leftarrow \mathsf{HKDF.TK(SHTS)}$......................

...

$\mathsf{fk}_S \leftarrow \mathsf{HKDF.Exp}_6(\mathsf{SHTS})$

$\{\mathsf{SF}\} : \mathsf{MAC}(\mathsf{fk}_S, T_7)$

$\xleftarrow{\{\mathsf{SF}\}}$

if $\mathsf{MAC}(\mathsf{fk}_S, T_7) \neq \mathsf{SF}$: **abort**

...

- Assume $\mathsf{HKDF.Ext}, \mathsf{HKDF.Exp}_i$ and G are ROs.
- In the OW-CPA reduction, we need to simulate the client receiving *one* ct.
- CHTS/SHTS similar to $H(K, \mathsf{ct})$
  $\Rightarrow$ can simulate 1 decaps query.

OW-CPA KEM $\Rightarrow$ MultiStage TLS 1.3

TLS 1.3 with KEM Key Schedule

Client		**Server**
	$\xrightarrow{\mathsf{pk}}$	
	$\xleftarrow{\mathsf{ct}}$	
$K \leftarrow \mathsf{Decaps}(\mathsf{sk}, \mathsf{ct})$		

$\mathsf{HS} \leftarrow \mathsf{HKDF.Ext}(\ldots, K)$
$\mathsf{CHTS} \leftarrow \mathsf{HKDF.Exp}_4(\mathsf{HS}, G(\mathsf{ct}, \ldots))$
$\mathsf{SHTS} \leftarrow \mathsf{HKDF.Exp}_5(\mathsf{HS}, G(\mathsf{ct}, \ldots))$
$\mathsf{dHS} \leftarrow \mathsf{HKDF.Exp}_0(\mathsf{HS}, \mathit{cnst})$

.... (Stage 1) accept $\mathsf{tk_c} \leftarrow \mathsf{HKDF.TK}(\mathsf{CHTS})$ ....

.... (Stage 2) accept $\mathsf{tk_s} \leftarrow \mathsf{HKDF.TK}(\mathsf{SHTS})$ ....

...

$\mathsf{fk}_S \leftarrow \mathsf{HKDF.Exp}_6(\mathsf{SHTS})$

$\{\mathsf{SF}\} : \mathsf{MAC}(\mathsf{fk}_S, T_7)$

$\xleftarrow{\{\mathsf{SF}\}}$

if $\mathsf{MAC}(\mathsf{fk}_S, T_7) \neq \mathsf{SF}$: **abort**

...

- Assume HKDF.Ext, $\mathsf{HKDF.Exp}_i$ and G are ROs.
- In the OW-CPA reduction, we need to simulate the client receiving *one* ct.
- CHTS/SHTS similar to $H(K, \mathsf{ct})$ $\Rightarrow$ can simulate 1 decaps query.
- dHS depends only on K... but client does not abort only if $\mathcal{A}$ knows fk_S (i.e. queried K to RO).

Security bound + corollary

Theorem

For any Multi-Stage ppt adversary $\mathcal{A}$ there exists a ppt adversary $\mathcal{B}$ s.t.

$$\mathsf{Adv}^{\text{multi-stage}}_{\mathsf{TLS1.3-1RTT}}(\mathcal{A}) \leq \textit{terms involving other primitives}$$
$$+ 6n_s^2\left(q_{RO_1}(q_{RO_2}+2)^2(q_{RO_3}+2)^3 \cdot \mathsf{Adv}^{\text{ow}-\text{cpa}}_{\mathsf{KEM}}(\mathcal{B})\right),$$

where n_s is the maximal number of sessions.

- OW-CPA KEMs are sufficient for TLS 1.3 (if other primitives secure).
- Result is theoretical (bound very loose!).

Security bound + corollary

EPFL

Theorem

For any Multi-Stage ppt adversary $\mathcal{A}$ there exists a ppt adversary $\mathcal{B}$ s.t.

$$\mathsf{Adv}^{\text{multi-stage}}_{\mathsf{TLS1.3-1RTT}}(\mathcal{A}) \leq \textit{terms involving other primitives}$$
$$+ 6n_s^2\left(q_{RO_1}(q_{RO_2}+2)^2(q_{RO_3}+2)^3 \cdot \mathsf{Adv}^{\text{ow-cpa}}_{\mathsf{KEM}}(\mathcal{B})\right),$$

where n_s is the maximal number of sessions.

- OW-CPA KEMs are sufficient for TLS 1.3 (if other primitives secure).
- Result is theoretical (bound very loose!).
- Corollary: CDH assumption is sufficient in TLS 1.3.

Outline

Impact

- IND-1CCA KEMs can be used in several PQ protocols (PQ TLS 1.3, KEMTLS, etc.).

Impact

EPFL

- IND-1CCA KEMs can be used in several PQ protocols (PQ TLS 1.3, KEMTLS, etc.).
- Compared to current solutions based on IND-CCA KEMs:
  1. Halves decapsulation time (on client-side in TLS 1.3 and KEMTLS).

Impact

Scheme	Decaps re-enc. (μs)	Decaps no re-enc. (μs)	Speedup
SIKE	2316	1020	2.27
Kyber	6.1	2.8	2.17
Lightsaber	11.1	4.0	2.78
Frodo-AES	295.0	48.3	6.11

Table: Benchmark of Decaps with/without re-encryption with liboqs (avx2 enabled, security level I). Setup: Ubuntu 21.04, Intel Core i7-1165G7 @ 2.8Ghz.

Impact

- IND-1CCA KEMs can be used in several PQ protocols (PQ TLS 1.3, KEMTLS, etc.).
- Compared to current solutions based on IND-CCA KEMs:
    1. Halves decapsulation time (on client-side in TLS 1.3 and KEMTLS).
    2. Can preserve symmetry of underlying CPA-secure scheme (e.g. SIDH).

Impact

- IND-1CCA KEMs can be used in several PQ protocols (PQ TLS 1.3, KEMTLS, etc.).
- Compared to current solutions based on IND-CCA KEMs:
  1. Halves decapsulation time (on client-side in TLS 1.3 and KEMTLS).
  2. Can preserve symmetry of underlying CPA-secure scheme (e.g. SIDH).
  3. Generic and drop-in replacement in protocols (works with nearly all NIST PQ KEM proposals).

Impact

- IND-1CCA KEMs can be used in several PQ protocols (PQ TLS 1.3, KEMTLS, etc.).
- Compared to current solutions based on IND-CCA KEMs:
    1. Halves decapsulation time (on client-side in TLS 1.3 and KEMTLS).
    2. Can preserve symmetry of underlying CPA-secure scheme (e.g. SIDH).
    3. Generic and drop-in replacement in protocols (works with nearly all NIST PQ KEM proposals).
    4. KEMs easy to implement (actually simplifies the FO transform).

Impact

- IND-1CCA KEMs can be used in several PQ protocols (PQ TLS 1.3, KEMTLS, etc.).
- Compared to current solutions based on IND-CCA KEMs:
  1. Halves decapsulation time (on client-side in TLS 1.3 and KEMTLS).
  2. Can preserve symmetry of underlying CPA-secure scheme (e.g. SIDH).
  3. Generic and drop-in replacement in protocols (works with nearly all NIST PQ KEM proposals).
  4. KEMs easy to implement (actually simplifies the FO transform).
- Downside: vulnerable to misuse/reuse attack (i.e. if the ephemeral key is reused for several encapsulation/decapsulation).

Future work

- QROM security of 2nd transform (T_H) and TLS result.
  Challenge: a lot of RO programming in the classical proof.

Future work

- QROM security of 2nd transform ($\mathsf{T_H}$) and TLS result.
  Challenge: a lot of RO programming in the classical proof.
- Better bound for TLS result.

Thank you!

EPFL

Questions?