Don’t Roll Your Own FO:
Challenges in Proving Post-Quantum CCA Security of Real-World KEMs

Varun Maram
Cybersecurity Group
SandboxAQ

: https://varun-maram.github.io/

: varun-maram-pqc

Based on joint works with Paul Grubbs, Kenny Paterson, and Keita Xagawa.

SANDBOXAQ

Fujisaki-**O**kamoto transform.

Don’t Roll Your Own FO:

Challenges in Proving Post-Quantum CCA Security of Real-World KEMs

- CRYSTALS-Kyber
- Streamlined NTRU Prime

Varun Maram

Cybersecurity Group

SandboxAQ

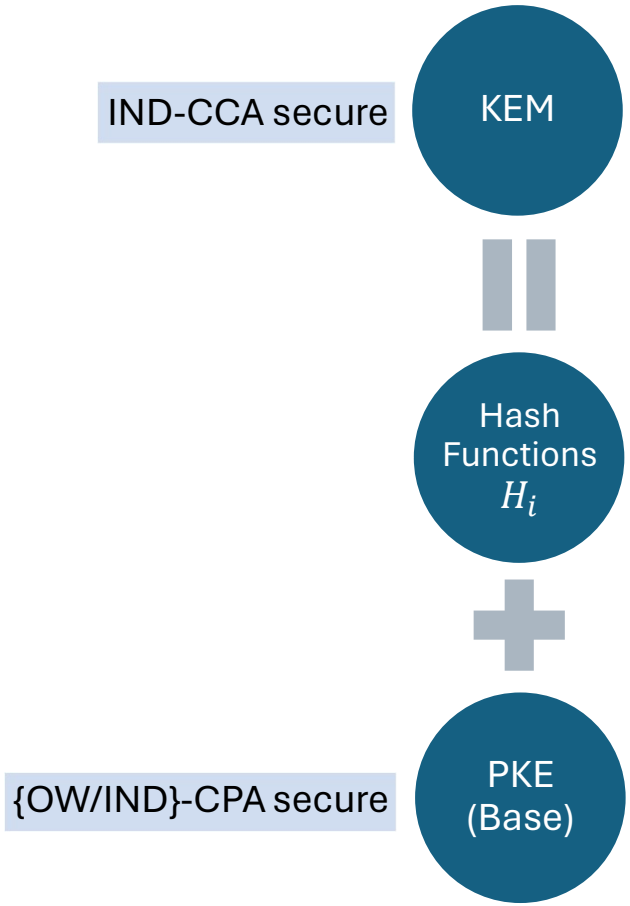
In the **Q**uantum **R**andom **O**racle **M**odel (QROM).

Based on joint works with Paul Grubbs, Kenny Paterson, and Keita Xagawa.

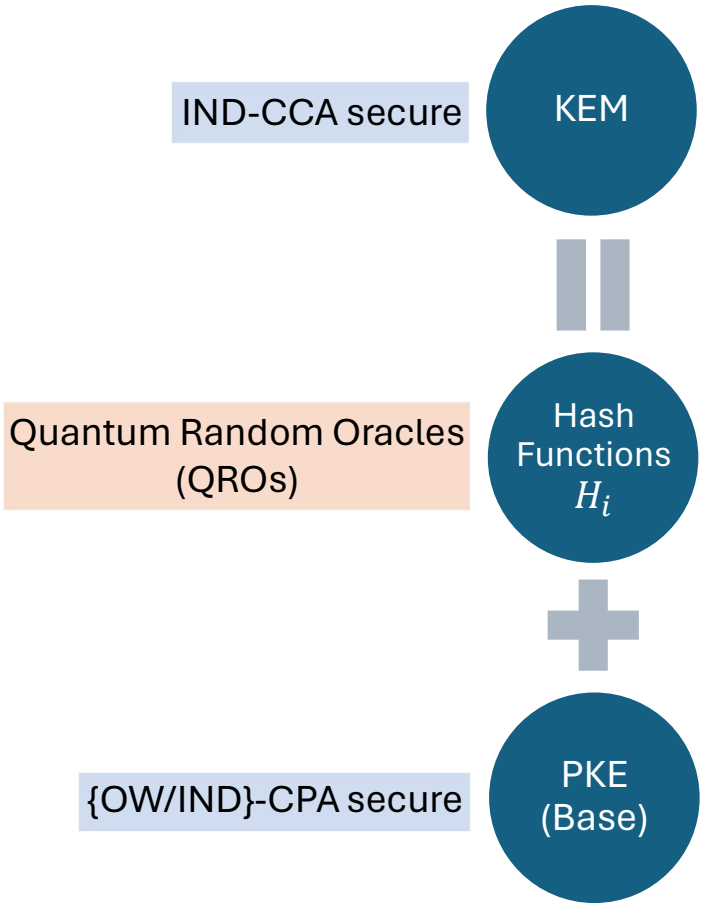
Fujisaki-Okamoto Transformation

Fujisaki-Okamoto Transformation

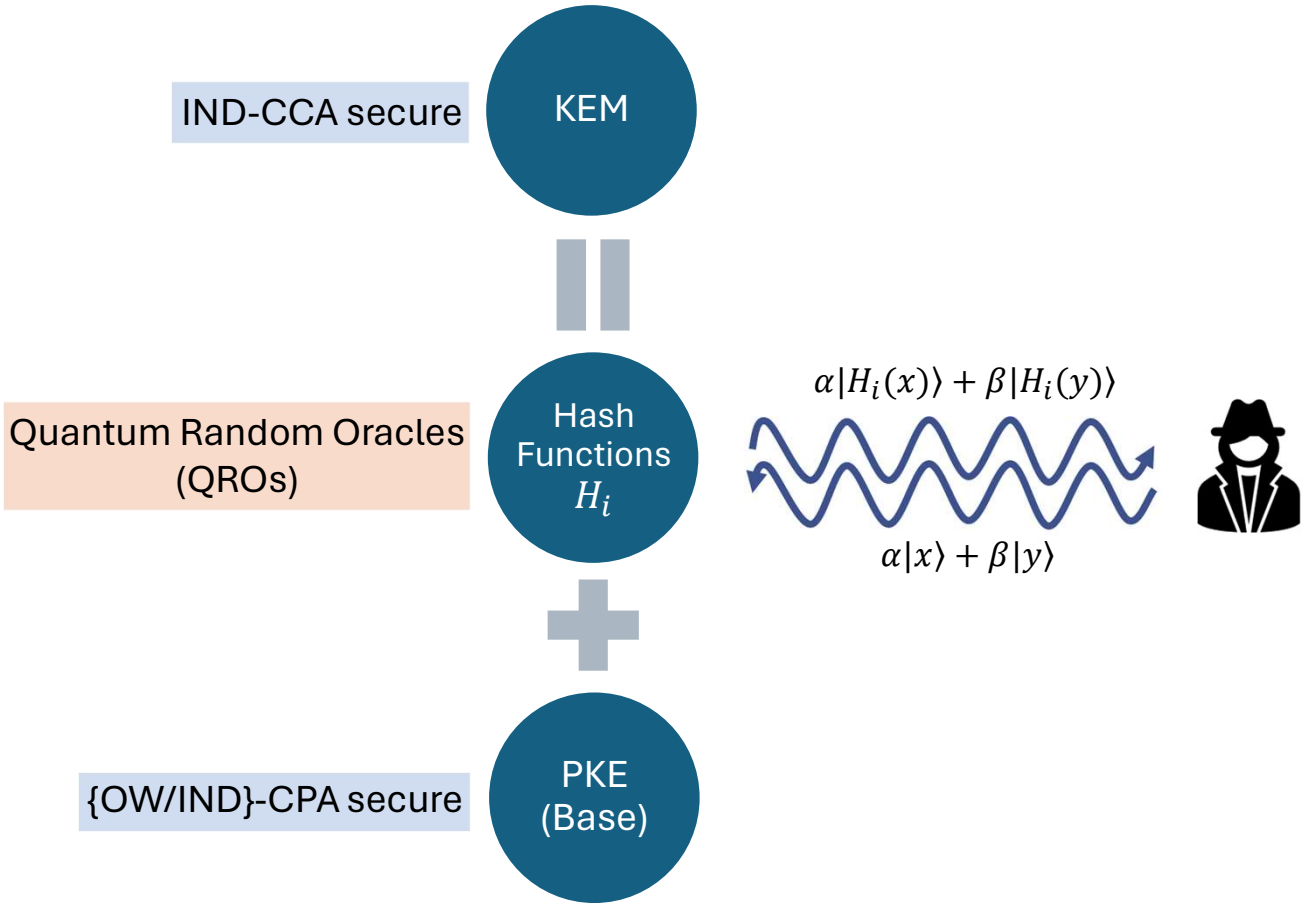
Fujisaki-Okamoto Transformation



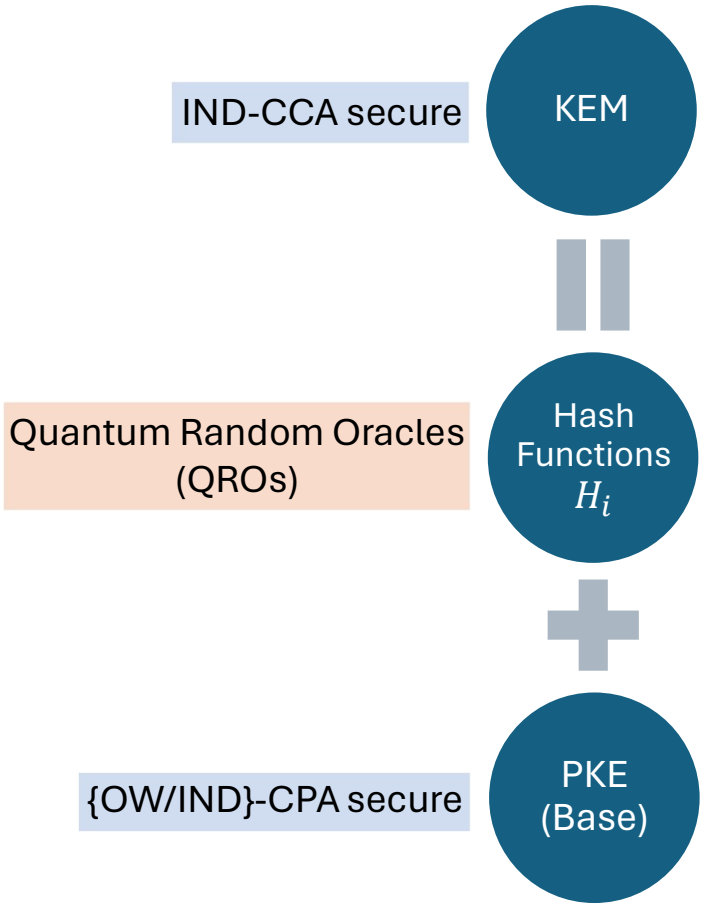
Fujisaki-Okamoto Transformation



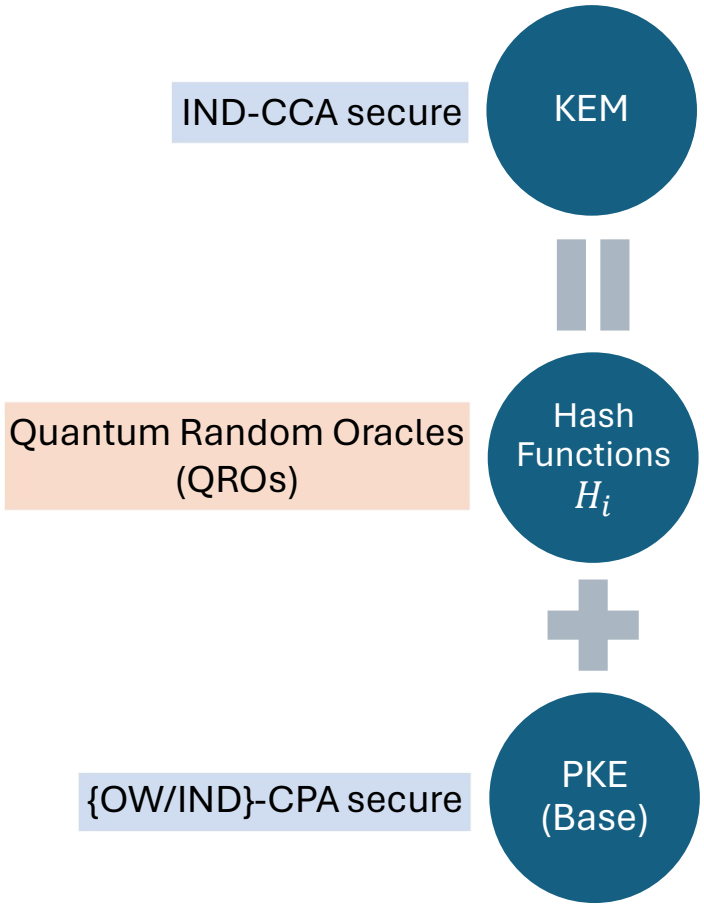
Fujisaki-Okamoto Transformation



Fujisaki-Okamoto Transformation



Fujisaki-Okamoto Transformation



FO Variants

IND-CCA secure

KEM

“key ← hash(m, c)”

Quantum Random Oracles (QROs)

Hash Functions H_i

{OW/IND}-CPA secure

PKE (Base)

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_s \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_s \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow$ Dec(sk, c)
3 : sk′ ← (sk, s)	3 : $c \leftarrow$ Enc(pk, $m; r$)	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m, c)$	4 : $c' \leftarrow$ Enc(pk, $m'; r'$)
	5 : **return** $(c, \overline{k})$	5 : **if** $c' = c$ **then**
		6 : **return** $G_k(m', c)$
		7 : **else return** $G_k(s, c)$

FO$^{\not\perp}$ [Hofheinz-Hövelmanns-Kiltz@TCC'17]

FO Variants

IND-CCA secure

KEM

"key ← hash(m, c)"

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_s \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_s \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow$ Dec(sk, c)
3 : sk′ ← (sk, s)	3 : $c \leftarrow$ Enc(pk, m; r)	3 : $r' \leftarrow G_r(m')$
4 : return (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m, c)$	4 : $c' \leftarrow$ Enc(pk, m'; r')
	5 : return $(c, \overline{k})$	5 : if $c' = c$ then
		6 : return $G_k(m', c)$
		7 : else return $G_k(s, c)$

FO$^{\not\perp}$ [Hofheinz-Hövelmanns-Kiltz@TCC'17]

Quantum Random Oracles (QROs)

Hash Functions H_i

"key ← hash(hash(m), hash(c))"

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_s \mathcal{M}$	1 : Parse sk′ = (sk, pk, h, s)
2 : $s \leftarrow_s \mathcal{M}$	2 : $m \leftarrow H(m)$	2 : $m' \leftarrow$ Dec(sk, c)
3 : pk′ ← (pk, H(pk))	3 : $h \leftarrow H$(pk)	3 : $(\overline{k'}, r') \leftarrow G_{kr}(m', h)$
4 : sk′ ← (sk, pk′, s)	4 : $(\overline{k}, r) \leftarrow G_{kr}(m, h)$	4 : $c' \leftarrow$ Enc(pk, m'; r')
5 : return (pk, sk′)	5 : $c \leftarrow$ Enc(pk, m; r)	5 : if $c' = c$ then
	6 : $k \leftarrow H'(\overline{k}, H(c))$	6 : return $H'(\overline{k'}, H(c))$
	7 : return (c, k)	7 : else return $H'(s, H(c))$

Kyber's FO variant

{OW/IND}-CPA secure

PKE (Base)

FO Variants

IND-CCA secure

KEM

"key ← hash(m, c)"

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_s \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_s \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow$ Dec(sk, c)
3 : sk′ ← (sk, s)	3 : $c \leftarrow$ Enc(pk, m; r)	3 : $r' \leftarrow G_r(m')$
4 : return (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m, c)$	4 : $c' \leftarrow$ Enc(pk, m′; r′)
	5 : return $(c, \overline{k})$	5 : if $c' = c$ then
		6 : return $G_k(m', c)$
		7 : else return $G_k(s, c)$

$FO^{\not\perp}$ [Hofheinz-Hövelmanns-Kiltz@TCC'17]

Quantum Random Oracles (QROs)

Hash Functions H_i

"key ← hash(hash(m), hash(c))"

[Grubbs-Maram-Paterson@Eurocrypt'22]

{OW/IND}-CPA secure

PKE (Base)

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_s \mathcal{M}$	1 : Parse sk′ = (sk, pk, h, s)
2 : $s \leftarrow_s \mathcal{M}$	2 : $m \leftarrow H(m)$	2 : $m' \leftarrow$ Dec(sk, c)
3 : pk′ ← (pk, H(pk))	3 : $h \leftarrow H$(pk)	3 : $(\overline{k'}, r') \leftarrow G_{kr}(m', h)$
4 : sk′ ← (sk, pk′, s)	4 : $(\overline{k}, r) \leftarrow G_{kr}(m, h)$	4 : $c' \leftarrow$ Enc(pk, m′; r′)
5 : return (pk, sk′)	5 : $c \leftarrow$ Enc(pk, m; r)	5 : if $c' = c$ then
	6 : $k \leftarrow H'(\overline{k}, H(c))$	6 : return $H'(\overline{k'}, H(c))$
	7 : return (c, k)	7 : else return $H'(s, H(c))$

These variations make prior IND-CCA security analysis of $FO^{\not\perp}$ in the QROM inapplicable.

Kyber's FO variant

FO Variants

IND-CCA secure

KEM

"key ← hash(m, c)"

Quantum Random Oracles (QROs)

Hash Functions H_i

{OW/IND}-CPA secure

PKE (Base)

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_s \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_s \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow$ Dec(sk, c)
3 : sk′ ← (sk, s)	3 : $c \leftarrow$ Enc(pk, m; r)	3 : $r' \leftarrow G_r(m')$
4 : return (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m, c)$	4 : $c' \leftarrow$ Enc(pk, m′; r′)
	5 : return $(c, \overline{k})$	5 : if $c' = c$ then
		6 : return $G_k(m', c)$
		7 : else return $G_k(s, c)$

FO$^{\not\perp}$ [Hofheinz-Hövelmanns-Kiltz@TCC'17]

Analysis by [Jiang-Zhang-Chen-Wang-Ma@Crypto'18]

FO Variants

IND-CCA secure

KEM

“key ← hash(m, c)”

Quantum Random Oracles (QROs)

Hash Functions H_i

{OW/IND}-CPA secure

PKE (Base)

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_s \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_s \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow$ Dec(sk, c)
3 : sk′ ← (sk, s)	3 : $c \leftarrow$ Enc(pk, m; r)	3 : $r' \leftarrow G_r(m')$
4 : return (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m, c)$	4 : $c' \leftarrow$ Enc(pk, m'; r')
	5 : return $(c, \overline{k})$	5 : if $c' = c$ then
		6 : return $G_k(m', c)$
		7 : else return $G_k(s, c)$

CPA-reduction needs **sk′** to answer CCA-adversary’s **Decap** queries.

FO$^{\not\perp}$ [Hofheinz-Hövelmanns-Kiltz@TCC'17]

Analysis by [Jiang-Zhang-Chen-Wang-Ma@Crypto’18]

FO Variants

IND-CCA secure

KEM

=

Quantum Random Oracles (QROs)

Hash Functions H_i

+

{OW/IND}-CPA secure

PKE (Base)

"key ← hash(m, c)"

CPA-reduction needs **sk′** to answer CCA-adversary's **Decap** queries.

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_s \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_s \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : sk′ ← (sk, s)	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : return (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m, c)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : return $(c, \overline{k})$	5 : if $c' = c$ then
		6 : return $G_k(m', c)$
		7 : else return $G_k(s, c)$

FO$^{\not\perp}$ [Hofheinz-Hövelmanns-Kiltz@TCC'17]

Analysis by [Jiang-Zhang-Chen-Wang-Ma@Crypto'18]

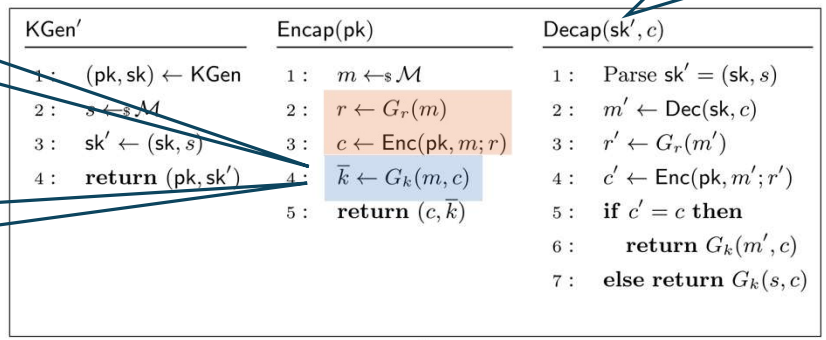
FO Variants

IND-CCA secure

KEM

“key ← hash(m, c)”

“key ← hash′(c)”



CPA-reduction needs **sk′** to answer CCA-adversary’s **Decap** queries.

FO$^{\not\perp}$ [Hofheinz-Hövelmanns-Kiltz@TCC'17]

Analysis by [Jiang-Zhang-Chen-Wang-Ma@Crypto’18]

Quantum Random Oracles (QROs)

Hash Functions H_i

{OW/IND}-CPA secure

PKE (Base)

FO Variants

IND-CCA secure

KEM

"key ← hash(m, c)"

≈

"key ← hash′(c)"

Quantum Random Oracles (QROs)

Hash Functions H_i

+

{OW/IND}-CPA secure

PKE (Base)

PKE correctness ensures there do not exist pairs (m_1, c) and (m_2, c).

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_s \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_s \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow$ Dec(sk, c)
3 : sk′ ← (sk, s)	3 : $c \leftarrow$ Enc(pk, $m; r$)	3 : $r' \leftarrow G_r(m')$
4 : return (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m, c)$	4 : $c' \leftarrow$ Enc(pk, $m'; r'$)
	5 : return $(c, \overline{k})$	5 : if $c' = c$ then
		6 : return $G_k(m', c)$
		7 : else return $G_k(s, c)$

CPA-reduction needs **sk′** to answer CCA-adversary's **Decap** queries.

FO✗ [Hofheinz-Hövelmanns-Kiltz@TCC'17]

Analysis by [Jiang-Zhang-Chen-Wang-Ma@Crypto'18]

FO Variants

IND-CCA secure

KEM

"key ← hash(m, c)"

≈

"key ← hash′(c)"

Quantum Random Oracles (QROs)

Hash Functions H_i

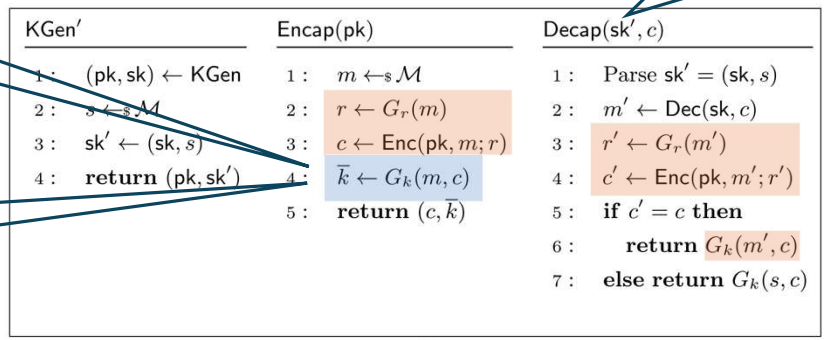
PKE correctness ensures there do not exist pairs (m_1, c) and (m_2, c).

+

{OW/IND}-CPA secure

PKE (Base)

CPA-reduction needs $\mathbf{sk}'$ to answer CCA-adversary's **Decap** queries.



FO$^{\not\perp}$ [Hofheinz-Hövelmanns-Kiltz@TCC'17]

Analysis by [Jiang-Zhang-Chen-Wang-Ma@Crypto'18]

FO Variants

IND-CCA secure

KEM

“key ← hash(m, c)”

≈

“key ← hash′(c)”

Quantum Random Oracles (QROs)

Hash Functions H_i

+

{OW/IND}-CPA secure

PKE (Base)

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_s \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_s \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow$ Dec(sk, c)
3 : sk′ ← (sk, s)	3 : $c \leftarrow$ Enc(pk, m; r)	3 : $r' \leftarrow G_r(m')$
4 : return (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m, c)$	4 : $c' \leftarrow$ Enc(pk, m'; r')
	5 : return $(c, \overline{k})$	5 : if $c' = c$ then
		6 : return $G_k(m', c)$
		7 : else return $G_k(s, c)$

FO$^{\not\perp}$ [Hofheinz-Hövelmanns-Kiltz@TCC'17]

CPA-reduction needs **sk′** to answer CCA-adversary’s **Decap** queries.

PKE correctness ensures there do not exist pairs (m_1, c) and (m_2, c).

Analysis by [Jiang-Zhang-Chen-Wang-Ma@Crypto’18]

“key ← hash′(c)”

FO Variants

IND-CCA secure

KEM

Quantum Random Oracles (QROs)

Hash Functions H_i

{OW/IND}-CPA secure

PKE (Base)

"key ← hash(m, c)"

≈

"key ← hash$'(c)$"

PKE correctness ensures there do not exist pairs (m_1, c) and (m_2, c).

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_s \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_s \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow$ Dec(sk, c)
3 : sk′ ← (sk, s)	3 : $c \leftarrow$ Enc(pk, m; r)	3 : $r' \leftarrow G_r(m')$
4 : return (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m, c)$	4 : $c' \leftarrow$ Enc(pk, m′; r′)
	5 : return $(c, \overline{k})$	5 : if $c' = c$ then
		6 : return $G_k(m', c)$
		7 : else return $G_k(s, c)$

CPA-reduction needs **sk′** to answer CCA-adversary's **Decap** queries.

FO$^{\not\perp}$ [Hofheinz-Hövelmanns-Kiltz@TCC'17]

Analysis by [Jiang-Zhang-Chen-Wang-Ma@Crypto'18]

"key ← hash$'(c)$"

"key ← hash$'(c)$" (pseudorandomness of QRO $G_k(s,\cdot)$)

FO Variants

IND-CCA secure

KEM

Quantum Random Oracles (QROs)

Hash Functions H_i

{OW/IND}-CPA secure

PKE (Base)

"key ← hash(m, c)"

≈

"key ← hash$'(c)$"

PKE correctness ensures there do not exist pairs (m_1, c) and (m_2, c).

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_s \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_s \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow$ Dec(sk, c)
3 : sk′ ← (sk, s)	3 : $c \leftarrow$ Enc(pk, m; r)	3 : $r' \leftarrow G_r(m')$
4 : return (pk, sk′)	4 : $\bar{k} \leftarrow G_k(m, c)$	4 : $c' \leftarrow$ Enc(pk, m'; r')
	5 : return $(c, \bar{k})$	5 : if $c' = c$ then
		6 : return $G_k(m', c)$
		7 : else return $G_k(s, c)$

FO$^{\not\perp}$ [Hofheinz-Hövelmanns-Kiltz@TCC'17]

Analysis by [Jiang-Zhang-Chen-Wang-Ma@Crypto'18]

"key ← hash$'(c)$"

"key ← hash$'(c)$" (pseudorandomness of QRO $G_k(s,\cdot)$)

FO Variants

IND-CCA secure

KEM

Quantum Random Oracles (QROs)

Hash Functions H_i

{OW/IND}-CPA secure

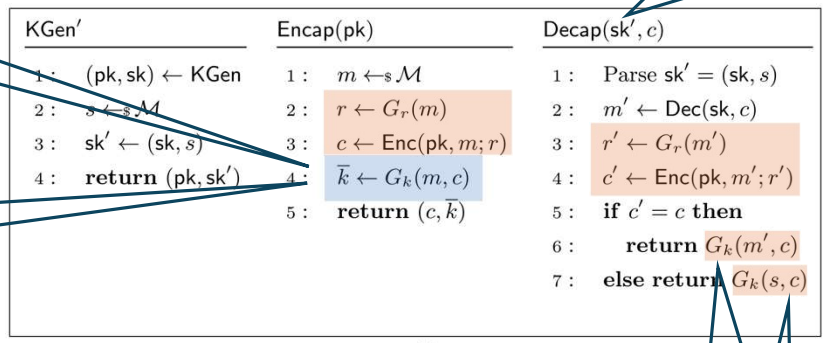
PKE (Base)

"key ← hash(m, c)"

≈

"key ← hash′(c)"

PKE correctness ensures there do not exist pairs (m_1, c) and (m_2, c).



CPA-reduction can just answer with key ≔ hash′(c)!

FO$^{\not\perp}$ [Hofheinz-Hövelmanns-Kiltz@TCC'17]

Analysis by [Jiang-Zhang-Chen-Wang-Ma@Crypto'18]

"key ← hash′(c)"

"key ← hash′(c)" (pseudorandomness of QRO $G_k(s,\cdot)$)

FO Variants

IND-CCA secure

KEM

“key ← hash(m, c)”

≈

“key ← hash′(c)”

CPA-reduction can just answer with key ≔ hash′(c)!

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_s \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_s \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow$ Dec(sk, c)
3 : sk′ ← (sk, s)	3 : $c \leftarrow$ Enc(pk, $m; r$)	3 : $r' \leftarrow G_r(m')$
4 : return (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m, c)$	4 : $c' \leftarrow$ Enc(pk, $m'; r'$)
	5 : return $(c, \overline{k})$	5 : if $c' = c$ then
		6 : return $G_k(m', c)$
		7 : else return $G_k(s, c)$

FO$^{\not\perp}$ [Hofheinz-Hövelmanns-Kiltz@TCC'17]

Quantum Random Oracles (QROs)

Hash Functions H_i

“key ← hash(hash(m), hash(c))”

+

{OW/IND}-CPA secure

PKE (Base)

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_s \mathcal{M}$	1 : Parse sk′ = (sk, pk, h, s)
2 : $s \leftarrow_s \mathcal{M}$	2 : $m \leftarrow H(m)$	2 : $m' \leftarrow$ Dec(sk, c)
3 : pk′ ← (pk, H(pk))	3 : $h \leftarrow H$(pk)	3 : $(\overline{k'}, r') \leftarrow G_{kr}(m', h)$
4 : sk′ ← (sk, pk′, s)	4 : $(\overline{k}, r) \leftarrow G_{kr}(m, h)$	4 : $c' \leftarrow$ Enc(pk, $m'; r'$)
5 : return (pk, sk′)	5 : $c \leftarrow$ Enc(pk, $m; r$)	5 : if $c' = c$ then
	6 : $k \leftarrow H'(\overline{k}, H(c))$	6 : return $H'(\overline{k'}, H(c))$
	7 : return (c, k)	7 : else return $H'(s, H(c))$

Kyber’s FO variant

FO Variants

IND-CCA secure

KEM

“key ← hash(m, c)”

≈

“key ← hash′(c)”

CPA-reduction can just answer with key ≔ hash′(c)!

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_s \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_s \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow$ Dec(sk, c)
3 : sk′ ← (sk, s)	3 : $c \leftarrow$ Enc(pk, $m; r$)	3 : $r' \leftarrow G_r(m')$
4 : return (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m, c)$	4 : $c' \leftarrow$ Enc(pk, $m'; r'$)
	5 : return $(c, \overline{k})$	5 : if $c' = c$ then
		6 : return $G_k(m', c)$
		7 : else return $G_k(s, c)$

FO$^{\not\perp}$ [Hofheinz-Hövelmanns-Kiltz@TCC'17]

Quantum Random Oracles (QROs)

Hash Functions H_i

“key ← hash(hash(m), hash(c))”

+

{OW/IND}-CPA secure

PKE (Base)

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_s \mathcal{M}$	1 : Parse sk′ = (sk, pk, h, s)
2 : $s \leftarrow_s \mathcal{M}$	2 : $m \leftarrow H(m)$	2 : $m' \leftarrow$ Dec(sk, c)
3 : pk′ ← (pk, H(pk))	3 : $h \leftarrow H$(pk)	3 : $(\overline{k'}, r') \leftarrow G_{kr}(m', h)$
4 : sk′ ← (sk, pk′, s)	4 : $(\overline{k}, r) \leftarrow G_{kr}(m, h)$	4 : $c' \leftarrow$ Enc(pk, $m'; r'$)
5 : return (pk, sk′)	5 : $c \leftarrow$ Enc(pk, $m; r$)	5 : if $c' = c$ then
	6 : $k \leftarrow H'(\overline{k}, H(c))$	6 : return $H'(\overline{k'}, H(c))$
	7 : return (c, k)	7 : else return $H'(s, H(c))$

Kyber’s FO variant

FO Variants

IND-CCA secure

KEM

“key ← hash(m, c)”

≈

“key ← hash′(c)”

CPA-reduction can just answer with key ≔ hash′(c)!

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_s \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_s \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow$ Dec(sk, c)
3 : sk′ ← (sk, s)	3 : $c \leftarrow$ Enc(pk, m; r)	3 : $r' \leftarrow G_r(m')$
4 : return (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m, c)$	4 : $c' \leftarrow$ Enc(pk, m'; r')
	5 : return $(c, \overline{k})$	5 : if $c' = c$ then
		6 : return $G_k(m', c)$
		7 : else return $G_k(s, c)$

FO$^{\not\perp}$ [Hofheinz-Hövelmanns-Kiltz@TCC'17]

Quantum Random Oracles (QROs)

Hash Functions H_i

“key ← hash(hash(m), hash(c))”

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_s \mathcal{M}$	1 : Parse sk′ = (sk, pk, h, s)
2 : $s \leftarrow_s \mathcal{M}$	2 : $m \leftarrow H(m)$	2 : $m' \leftarrow$ Dec(sk, c)
3 : pk′ ← (pk, H(pk))	3 : $h \leftarrow H$(pk)	3 : $(\overline{k'}, r') \leftarrow G_{kr}(m', h)$
4 : sk′ ← (sk, pk′, s)	4 : $(\overline{k}, r) \leftarrow G_{kr}(m, h)$	4 : $c' \leftarrow$ Enc(pk, m'; r')
5 : return (pk, sk′)	5 : $c \leftarrow$ Enc(pk, m; r)	5 : if $c' = c$ then
	6 : $k \leftarrow H'(\overline{k}, H(c))$	6 : return $H'(\overline{k'}, H(c))$
	7 : return (c, k)	7 : else return $H'(s, H(c))$

Kyber’s FO variant

{OW/IND}-CPA secure

PKE (Base)

Given “hash(m)” and “hash(c)”, how to extract (m, c) for checking this programming condition?

FO Variants

IND-CCA secure

KEM

"key ← hash(m, c)"

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_s \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_s \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow$ Dec(sk, c)
3 : sk′ ← (sk, s)	3 : $c \leftarrow$ Enc(pk, m; r)	3 : $r' \leftarrow G_r(m')$
4 : return (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m, c)$	4 : $c' \leftarrow$ Enc(pk, m′; r′)
	5 : return $(c, \overline{k})$	5 : if $c' = c$ then
		6 : return $G_k(m', c)$
		7 : else return $G_k(s, c)$

$FO^{\not\perp}$ [Hofheinz-Hövelmanns-Kiltz@TCC'17]

Quantum Random Oracles (QROs)

Hash Functions H_i

"key ← hash(hash(m), hash(c))"

[Grubbs-Maram-Paterson@Eurocrypt'22]

{OW/IND}-CPA secure

PKE (Base)

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_s \mathcal{M}$	1 : Parse sk′ = (sk, pk, h, s)
2 : $s \leftarrow_s \mathcal{M}$	2 : $m \leftarrow H(m)$	2 : $m' \leftarrow$ Dec(sk, c)
3 : pk′ ← (pk, H(pk))	3 : $h \leftarrow H$(pk)	3 : $(\overline{k'}, r') \leftarrow G_{kr}(m', h)$
4 : sk′ ← (sk, pk′, s)	4 : $(\overline{k}, r) \leftarrow G_{kr}(m, h)$	4 : $c' \leftarrow$ Enc(pk, m′; r′)
5 : return (pk, sk′)	5 : $c \leftarrow$ Enc(pk, m; r)	5 : if $c' = c$ then
	6 : $k \leftarrow H'(\overline{k}, H(c))$	6 : return $H'(\overline{k'}, H(c))$
	7 : return (c, k)	7 : else return $H'(s, H(c))$

These variations make prior IND-CCA security analysis of $FO^{\not\perp}$ in the QROM inapplicable.

Kyber's FO variant

FO Variants

IND-CCA secure

KEM

“key ← hash(m, c)”

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_{\$} \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_{\$} \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : sk′ ← (sk, s)	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : return (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m, c)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : return $(c, \overline{k})$	5 : if $c' = c$ then
		6 : return $G_k(m', c)$
		7 : else return $G_k(s, c)$

$\mathrm{FO}^{\not\perp}$ [Hofheinz-Hövelmanns-Kiltz@TCC'17]

Quantum Random Oracles (QROs)

Hash Functions H_i

“key ← hash(hash(m), hash(c))”

[Maram-Xagawa@PKC’23]

{OW/IND}-CPA secure

PKE (Base)

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_{\$} \mathcal{M}$	1 : Parse sk′ = (sk, pk, h, s)
2 : $s \leftarrow_{\$} \mathcal{M}$	2 : $m \leftarrow H(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : pk′ ← (pk, H(pk))	3 : $h \leftarrow H(\mathsf{pk})$	3 : $(\overline{k'}, r') \leftarrow G_{kr}(m', h)$
4 : sk′ ← (sk, pk′, s)	4 : $(\overline{k}, r) \leftarrow G_{kr}(m, h)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
5 : return (pk, sk′)	5 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	5 : if $c' = c$ then
	6 : $k \leftarrow H'(\overline{k}, H(c))$	6 : return $H'(\overline{k'}, H(c))$
	7 : return (c, k)	7 : else return $H'(s, H(c))$

We provide a concrete proof of IND-CCA security for Kyber in the QROM.

$\mathrm{Adv}^{\mathrm{IND-CCA}}_{\mathrm{Kyber}} \leq \mathrm{Adv}^{\mathrm{IND-CCA}}_{\mathrm{FO}^{\not\perp}} + \mathrm{Adv}^{\mathrm{CR}}_{H}$

Kyber’s FO variant

FO Variants

IND-CCA secure

KEM

"key ← hash(m, c)"

Collision-resistance of nested hash H.

Quantum Random Oracles (QROs)

Hash Functions H_i

"key ← hash(hash(m), hash(c))"

[Maram-Xagawa@PKC'23]

{OW/IND}-CPA secure

PKE (Base)

We provide a concrete proof of IND-CCA security for Kyber in the QROM.

$\mathrm{Adv}^{\mathrm{IND-CCA}}_{\mathrm{Kyber}} \leq \mathrm{Adv}^{\mathrm{IND-CCA}}_{\mathrm{FO}^{\not\perp}} + \mathrm{Adv}^{\mathrm{CR}}_{H}$

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_{\$} \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_{\$} \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : sk′ ← (sk, s)	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : return (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m, c)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : return $(c, \overline{k})$	5 : if $c' = c$ then
		6 : return $G_k(m', c)$
		7 : else return $G_k(s, c)$

$\mathrm{FO}^{\not\perp}$ [Hofheinz-Hövelmanns-Kiltz@TCC'17]

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_{\$} \mathcal{M}$	1 : Parse sk′ = (sk, pk, h, s)
2 : $s \leftarrow_{\$} \mathcal{M}$	2 : $m \leftarrow H(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : pk′ ← (pk, H(pk))	3 : $h \leftarrow H(\mathsf{pk})$	3 : $(\overline{k'}, r') \leftarrow G_{kr}(m', h)$
4 : sk′ ← (sk, pk′, s)	4 : $(\overline{k}, r) \leftarrow G_{kr}(m, h)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
5 : return (pk, sk′)	5 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	5 : if $c' = c$ then
	6 : $k \leftarrow H'(\overline{k}, H(c))$	6 : return $H'(\overline{k'}, H(c))$
	7 : return (c, k)	7 : else return $H'(s, H(c))$

Kyber's FO variant

FO Variants

IND-CCA secure

KEM

"key ← hash(m, c)"

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_s \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_s \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow$ Dec(sk, c)
3 : sk′ ← (sk, s)	3 : $c \leftarrow$ Enc(pk, m; r)	3 : $r' \leftarrow G_r(m')$
4 : return (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m, c)$	4 : $c' \leftarrow$ Enc(pk, m′; r′)
	5 : return $(c, \overline{k})$	5 : if $c' = c$ then
		6 : return $G_k(m', c)$
		7 : else return $G_k(s, c)$

$FO^{\not\perp}$ [Hofheinz-Hövelmanns-Kiltz@TCC'17]

Quantum Random Oracles (QROs)

Hash Functions H_i

"key ← hash(hash(m), hash(c))"

[Grubbs-Maram-Paterson@Eurocrypt'22]

{OW/IND}-CPA secure

PKE (Base)

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_s \mathcal{M}$	1 : Parse sk′ = (sk, pk, h, s)
2 : $s \leftarrow_s \mathcal{M}$	2 : $m \leftarrow H(m)$	2 : $m' \leftarrow$ Dec(sk, c)
3 : pk′ ← (pk, H(pk))	3 : $h \leftarrow H$(pk)	3 : $(\overline{k'}, r') \leftarrow G_{kr}(m', h)$
4 : sk′ ← (sk, pk′, s)	4 : $(\overline{k}, r) \leftarrow G_{kr}(m, h)$	4 : $c' \leftarrow$ Enc(pk, m′; r′)
5 : return (pk, sk′)	5 : $c \leftarrow$ Enc(pk, m; r)	5 : if $c' = c$ then
	6 : $k \leftarrow H'(\overline{k}, H(c))$	6 : return $H'(\overline{k'}, H(c))$
	7 : return (c, k)	7 : else return $H'(s, H(c))$

These variations make prior IND-CCA security analysis of $FO^{\not\perp}$ in the QROM inapplicable.

Kyber's FO variant

FO Variants

IND-CCA secure

KEM

“key ← hash(m, c)”

Quantum Random Oracles (QROs)

Hash Functions H_i

“key ← hash(hash(m), hash(c))

[Grubbs-Maram-Paterson@Eurocrypt’22]

{OW/IND}-CPA secure

PKE (Base)

These variations make prior IND-CCA security analysis of FO$^{\not\perp}$ in the QROM inapplicable.

KGen′

1 : (pk, sk)
2 : $s \leftarrow_s \mathcal{A}$
3 : sk′ ← (
4 : return

KGen′

1 : (pk, sk) ←
2 : $s \leftarrow_s \mathcal{M}$
3 : pk′ ← (pk,
4 : sk′ ← (sk,
5 : return (p

Kyber decisions, part 2: FO transform 692 views

Peter Schwabe

to pqc-forum, aut...@pq-

Dear all,

This is the second ma
standardization. Kybe
rounds) uses a tweak
KEM from a CPA-secu
public key into the ran
hash of the ciphertext
are the following:

* Hashing the (hash o
"contributory", i.e., the
parties;

* hashing the (hash of
protection against mu
failures; and

* hashing the (hash of
this shared key depen

Through the course of
the FO transform and
question for standard
incorporated into Kybe

2.) Hashing the (hash of the) ciphertext into the final shared key does not help at all with any formal security property or with proofs. On the contrary, hashing the hash of the ciphertext into the final key complicates QROM proofs as pointed out in https://eprint.iacr.org/2021/708.pdf. Removing this tweak simplifies proofs and speeds up encapsulation. Note that decapsulation will still need to compute a hash over the full ciphertext for implicit rejection and, to avoid timing side channels, needs to do so in every decapsulation, not just after a decryption failure. So, there won't be any performance gain in decapsulation. Should the Kyber standard drop hashing the hash of the ciphertext into the shared key?

The obvious disadvantage with both possible changes is that such changes at such a late stage require very careful evaluation. We may have missed some non-standard property that Kyber achieves with these tweaks, but does not without. Also, the modifications are not completely orthogonal to potential modifications of symmetric crypto (see the previous mail), because they require changes to the hashing inside the FO transform with possible consequences for domain separation.

Again, we're looking forward to hear what everybody thinks!

All the best,

The Kyber team

1.) https://eprint.iacr.org/2021/1351 shows that as a protection against multi-target failure attacks it is not necessary to make random coins dependent on the full public key. It is sufficient to hash in a prefix of the public key, if that prefix has sufficiently high min-entropy. We are not aware of any formal definition of a KEM being "contributory", but intuitively also for this property using such a prefix would be sufficient. Using prefix(pk) instead of H(pk) would require fewer Keccak permutations in Kyber and thus speed up encapsulation. Should the Kyber standard use prefix(pk) rather than H(pk)?

, h)
′)
$H(c)$)
, $H(c)$)

Kyber’s FO variant

FO Variants

IND-CCA secure

KEM

"key ← hash(m, c)"

KGen′

1 : (pk, sk)
2 : $s \leftarrow_s \mathcal{M}$
3 : sk′ ← (
4 : return

Kyber decisions, part 2: FO transform 692 views

Peter Schwabe
to pqc-forum, aut...@pq-

Dear all,

This is the second ma
standardization. Kybe
rounds) uses a tweak
KEM from a CPA-secu
public key into the ran
hash of the ciphertext
are the following:

* Hashing the (hash o
"contributory", i.e., the
parties;

* hashing the (hash of
protection against mu
failures; and

* hashing the (hash of
this shared key depen

Through the course of
the FO transform and
question for standard
incorporated into Kybe

2.) Hashing the (hash of the) ciphertext into the final shared key does not help at all with any formal security property or with proofs. On the contrary, hashing the hash of the ciphertext into the final key complicates QROM proofs as pointed out in https://eprint.iacr.org/2021/708.pdf. Removing this tweak simplifies proofs and speeds up encapsulation. Note that decapsulation will still need to compute a hash over the full ciphertext for implicit rejection and, to avoid timing side channels, needs to do so in every decapsulation, not just after a decryption failure. So, there won't be any performance gain in decapsulation. Should the Kyber standard drop hashing the hash of the ciphertext into the shared key?

The obvious disadvantage with both possible changes is that such changes at such a late stage require very careful evaluation. We may have missed some non-standard property that Kyber achieves with these tweaks, but does not without. Also, the modifications are not completely orthogonal to potential modifications of symmetric crypto (see the previous mail), because they require changes to the hashing inside the FO transform with possible consequences for domain separation.

Again, we're looking forward to hear what everybody thinks!

All the best,

The Kyber team

1.) https://eprint.iacr.org/2021/1351 shows that as a protection against multi-target failure attacks it is not necessary to make random coins dependent on the full public key. It is sufficient to hash in a prefix of the public key, if that prefix has sufficiently high min-entropy. We are not aware of any formal definition of a KEM being "contributory", but intuitively also for this property using such a prefix would be sufficient. Using prefix(pk) instead of H(pk) would require fewer Keccak permutations in Kyber and thus speed up encapsulation. Should the Kyber standard use prefix(pk) rather than H(pk)?

Quant

dustin...@nist.gov Apr 28, 2023, 2:50:13 PM
to pqc-forum, b...@cloudflare.com, pqc-...@list.nist.gov, Andreas Hülsing

All,

NIST is planning to incorporate the CRYSTAL-Kyber team's suggestion of tweaking the FO transform into the draft standard, as described in Peter Schwabe's earlier post. Any comments or feedback are welcome.

Dustin

[Grubbs-Maram-Paterson@Eurocrypt'22]

2 : $s \leftarrow_s \mathcal{M}$
3 : pk′ ← (pk,
4 : sk′ ← (sk,
5 : return (p

{OW/IND}-CPA secure

PKE (Base)

These variations make prior IND-CCA security analysis of FO$^{\not\perp}$ in the QROM inapplicable.

Kyber's FO variant

FO Variants

IND-CCA secure

KEM

“key ← hash(m, c)”

Quantum Random Oracles (QROs)

Hash Functions H_i

“key ← hash(hash(m), hash(c))”

[Grubbs-Maram-Paterson@Eurocrypt’22]

{OW/IND}-CPA secure

PKE (Base)

These variations make prior IND-CCA security analysis of FO$^{\not\perp}$ in the QROM inapplicable.

KGen′

1 : (pk, sk) ←

2 : $s \leftarrow_s \mathcal{M}$

3 : sk′ ← (s

4 : return

KGen′

1 : (pk, sk) ← K

2 : $s \leftarrow_s \mathcal{M}$

3 : pk′ ← (pk,

4 : sk′ ← (sk, p

5 : return (pk

Check for updates

FIPS 203 (Draft)

Federal Information Processing Standards Publication

Module-Lattice-based Key-Encapsulation Mechanism Standard

Categor

Informat

National

Gaithers

This pub

https://do

Publishe

U.S. Depa

Gina M. R

National Institute of Standards and Technology

Laurie E. Locascio, NIST Director and Under Secretary of Commerce for Standards and Technology

as it is based on the so-called Module Learning With Errors assumption.

1.3 Differences From the CRYSTALS-KYBER Submission

Below is a list of all scheme differences between CRYSTALS-KYBER (as described in [4]) and the ML-KEM scheme specified in this document. The list consists only of those differences that result in differing input-output behavior of the main algorithms (i.e., KeyGen, Encaps, Decaps) of CRYSTALS-KYBER and ML-KEM. Recall that a conforming implementation need only match the input-output behavior of these three algorithms (see “Implementations” above, and Section 3.3 below). Consequently, the list below does not include any of the numerous differences in how the main algorithms actually produce outputs from inputs (e.g., via different computational steps or different subroutines). The list below also does not include any differences in presentation between this standard and [4].

- In the third-round specification [4], the shared secret key was treated as a variable-length value whose length depends on how this key would be used in the relevant application. In this specification, the length of the shared secret key is fixed to 256 bits. In this specification, this key can be used directly in applications as a symmetric key; alternatively, symmetric keys can be derived from this key, as specified in Section 3.3.
- The ML-KEM.Encaps and ML-KEM.Decaps algorithms in this specification use a different variant of the Fujisaki-Okamoto transform (see [5, 6]) than the third-round specification [4]. Specifically, ML-KEM.Encaps no longer includes a hash of the ciphertext in the derivation of the shared secret, and ML-KEM.Decaps has been adjusted to match this change.

Kyber’s FO variant

FO Variants

IND-CCA secure

KEM

"key ← hash(m, c)"

Quantum Random Oracles (QROs)

Hash Functions H_i

"key ← hash(hash(m), hash(c))

The ML-KEM.Encaps and ML-KEM.Decaps algorithms in this specification use a different variant of the Fujisaki-Okamoto transform (see [5, 6]) than the third-round specification [4]. Specifically, ML-KEM.Encaps no longer includes a hash of the ciphertext in the derivation of the shared secret, and ML-KEM.Decaps has been adjusted to match this

{OW/IND}-CPA secure

PKE (Base)

These variations make prior IND-CCA security analysis of FO$^{\not}$ in the QROM inapplicable.

Kyber's FO variant

KGen′

1 : (pk, sk)
2 : $s \leftarrow s\mathcal{M}$
3 : sk′ ← (s
4 : return
5 : return (pk

Check for updates

FIPS 203 (Draft)

Federal Information Processing Standards Publication

Module-Lattice-based Key-Encapsulation Mechanism Standard

Categor

as it is based on the so-called Module Learning With Errors assumption.

1.3 Differences From the CRYSTALS-KYBER Submission

Below is a list of all scheme differences between CRYSTALS-KYBER (as described in [4]) and the ML-KEM scheme specified in this document. The list consists only of those differences that result in differing input-output behavior of the main algorithms (i.e., KeyGen, Encaps, Decaps) of CRYSTALS-KYBER and ML-KEM. Recall that a conforming implementation need only match the input-output behavior of these three algorithms (see "Implementations" above, and Section 3.3 below). Consequently, the list below does not include any of the numerous differences in how the main algorithms actually produce outputs from inputs (e.g., via different computational steps or different subroutines). The list below also does not include any differences in presentation between this standard and [4].

- In the third-round specification [4], the shared secret key was treated as a variable-length value whose length depends on how this key would be used in the relevant application. In this specification, the length of the shared secret key is fixed to 256 bits. In this specification, this key can be used directly in applications as a symmetric key; alternatively, symmetric keys can be derived from this key, as specified in Section 3.3.
- The ML-KEM.Encaps and ML-KEM.Decaps algorithms in this specification use a different variant of the Fujisaki-Okamoto transform (see [5, 6]) than the third-round specification [4]. Specifically, ML-KEM.Encaps no longer includes a hash of the ciphertext in the derivation of the shared secret, and ML-KEM.Decaps has been adjusted to match this change.

FO Variants

IND-CCA secure

KEM

"key ← hash(m, c)"

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_s \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_s \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow$ Dec(sk, c)
3 : sk′ ← (sk, s)	3 : $c \leftarrow$ Enc(pk, m; r)	3 : $r' \leftarrow G_r(m')$
4 : return (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m, c)$	4 : $c' \leftarrow$ Enc(pk, m′; r′)
	5 : return $(c, \overline{k})$	5 : if $c' = c$ then
		6 : return $G_k(m', c)$
		7 : else return $G_k(s, c)$

$FO^{\not\perp}$ [Hofheinz-Hövelmanns-Kiltz@TCC'17]

Quantum Random Oracles (QROs)

Hash Functions H_i

"key ← hash(hash(m), hash(c))"

[Grubbs-Maram-Paterson@Eurocrypt'22]

{OW/IND}-CPA secure

PKE (Base)

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_s \mathcal{M}$	1 : Parse sk′ = (sk, pk, h, s)
2 : $s \leftarrow_s \mathcal{M}$	2 : $m \leftarrow H(m)$	2 : $m' \leftarrow$ Dec(sk, c)
3 : pk′ ← (pk, H(pk))	3 : $h \leftarrow H$(pk)	3 : $(\overline{k'}, r') \leftarrow G_{kr}(m', h)$
4 : sk′ ← (sk, pk′, s)	4 : $(\overline{k}, r) \leftarrow G_{kr}(m, h)$	4 : $c' \leftarrow$ Enc(pk, m′; r′)
5 : return (pk, sk′)	5 : $c \leftarrow$ Enc(pk, m; r)	5 : if $c' = c$ then
	6 : $k \leftarrow H'(\overline{k}, H(c))$	6 : return $H'(\overline{k'}, H(c))$
	7 : return (c, k)	7 : else return $H'(s, H(c))$

These variations make prior IND-CCA security analysis of $FO^{\not\perp}$ in the QROM inapplicable.

Kyber's FO variant

FO Variants

IND-CCA secure

KEM

“key ← hash(m, c)”

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_{\$} \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_{\$} \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : sk′ ← (sk, s)	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : return (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m, c)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : return $(c, \overline{k})$	5 : if $c' = c$ then
		6 : return $G_k(m', c)$
		7 : else return $G_k(s, c)$

$\mathrm{FO}^{\not\perp}$ [Hofheinz-Hövelmanns-Kiltz@TCC'17]

Quantum Random Oracles (QROs)

Hash Functions H_i

“key ← hash(hash(m), hash(c))”

[Maram-Xagawa@PKC’23]

{OW/IND}-CPA secure

PKE (Base)

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_{\$} \mathcal{M}$	1 : Parse sk′ = (sk, pk, h, s)
2 : $s \leftarrow_{\$} \mathcal{M}$	2 : $m \leftarrow H(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : pk′ ← (pk, H(pk))	3 : $h \leftarrow H(\mathsf{pk})$	3 : $(\overline{k'}, r') \leftarrow G_{kr}(m', h)$
4 : sk′ ← (sk, pk′, s)	4 : $(\overline{k}, r) \leftarrow G_{kr}(m, h)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
5 : return (pk, sk′)	5 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	5 : if $c' = c$ then
	6 : $k \leftarrow H'(\overline{k}, H(c))$	6 : return $H'(\overline{k'}, H(c))$
	7 : return (c, k)	7 : else return $H'(s, H(c))$

We provide a concrete proof of IND-CCA security for Kyber in the QROM.

$\mathrm{Adv}^{\mathrm{IND-CCA}}_{\mathrm{Kyber}} \leq \mathrm{Adv}^{\mathrm{IND-CCA}}_{\mathrm{FO}^{\not\perp}} + \mathrm{Adv}^{\mathrm{CR}}_{H}$

Kyber’s FO variant

Analysis of Kyber

IND-CCA secure

KEM

"key ← hash(m, c)"

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_s \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_s \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow$ Dec(sk, c)
3 : sk′ ← (sk, s)	3 : $c \leftarrow$ Enc(pk, m; r)	3 : $r' \leftarrow G_r(m')$
4 : return (pk, sk′)	4 : $\bar{k} \leftarrow G_k(m, c)$	4 : $c' \leftarrow$ Enc(pk, m'; r')
	5 : return $(c, \bar{k})$	5 : if $c' = c$ then
		6 : return $G_k(m', c)$
		7 : else return $G_k(s, c)$

FO$^{\not\perp}$ [Hofheinz-Hövelmanns-Kiltz@TCC'17]

Quantum Random Oracles (QROs)

Hash Functions H_i

"key ← hash(hash(m), hash(c))"

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_s \mathcal{M}$	1 : Parse sk′ = (sk, pk, h, s)
2 : $s \leftarrow_s \mathcal{M}$	2 : $m \leftarrow H(m)$	2 : $m' \leftarrow$ Dec(sk, c)
3 : pk′ ← (pk, H(pk))	3 : $h \leftarrow H$(pk)	3 : $(\bar{k}', r') \leftarrow G_{kr}(m', h)$
4 : sk′ ← (sk, pk′, s)	4 : $(\bar{k}, r) \leftarrow G_{kr}(m, h)$	4 : $c' \leftarrow$ Enc(pk, m'; r')
5 : return (pk, sk′)	5 : $c \leftarrow$ Enc(pk, m; r)	5 : if $c' = c$ then
	6 : $k \leftarrow H'(\bar{k}, H(c))$	6 : return $H'(\bar{k}', H(c))$
	7 : return (c, k)	7 : else return $H'(s, H(c))$

Kyber's FO variant

{OW/IND}-CPA secure

PKE (Base)

Analysis of Kyber

IND-CCA secure

KEM

"key ← hash(m, c)"

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_\$ \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_\$ \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow$ Dec(sk, c)
3 : sk′ ← (sk, s)	3 : $c \leftarrow$ Enc(pk, $m; r$)	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\bar{k} \leftarrow G_k(m, c)$	4 : $c' \leftarrow$ Enc(pk, $m'; r'$)
	5 : **return** $(c, \bar{k})$	5 : **if** $c' = c$ **then**
		6 : **return** $G_k(m', c)$
		7 : **else return** $G_k(s, c)$

$\mathrm{FO}^{\not\perp}$ [Hofheinz-Hövelmanns-Kiltz@TCC'17]

Quantum Random Oracles (QROs)

Hash Functions H_i

"key ← hash(m)"

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_\$ \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_\$ \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow$ Dec(sk, c)
3 : sk′ ← (sk, s)	3 : $c \leftarrow$ Enc(pk, $m; r$)	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\bar{k} \leftarrow G_k(m)$	4 : $c' \leftarrow$ Enc(pk, $m'; r'$)
	5 : **return** $(c, \bar{k})$	5 : **if** $c' = c$ **then**
		6 : **return** $G_k(m')$
		7 : **else return** $G_k(s, c)$

$\mathrm{FO}_m^{\not\perp}$ [Hofheinz-Hövelmanns-Kiltz@TCC'17]

{OW/IND}-CPA secure

PKE (Base)

Analysis of Kyber

IND-CCA secure

KEM

"key ← hash(m, c)"

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_\$ \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_\$ \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow$ Dec(sk, c)
3 : sk′ ← (sk, s)	3 : $c \leftarrow$ Enc(pk, m; r)	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\bar{k} \leftarrow G_k(m, c)$	4 : $c' \leftarrow$ Enc(pk, m'; r')
	5 : **return** $(c, \bar{k})$	5 : **if** $c' = c$ **then**
		6 : **return** $G_k(m', c)$
		7 : **else return** $G_k(s, c)$

$\text{FO}^{\not\perp}$ [Hofheinz-Hövelmanns-Kiltz@TCC'17]

Quantum Random Oracles (QROs)

Hash Functions H_i

"key ← hash(m)"

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_\$ \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_\$ \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow$ Dec(sk, c)
3 : sk′ ← (sk, s)	3 : $c \leftarrow$ Enc(pk, m; r)	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\bar{k} \leftarrow G_k(m)$	4 : $c' \leftarrow$ Enc(pk, m'; r')
	5 : **return** $(c, \bar{k})$	5 : **if** $c' = c$ **then**
		6 : **return** $G_k(m')$
		7 : **else return** $G_k(s, c)$

{OW/IND}-CPA secure

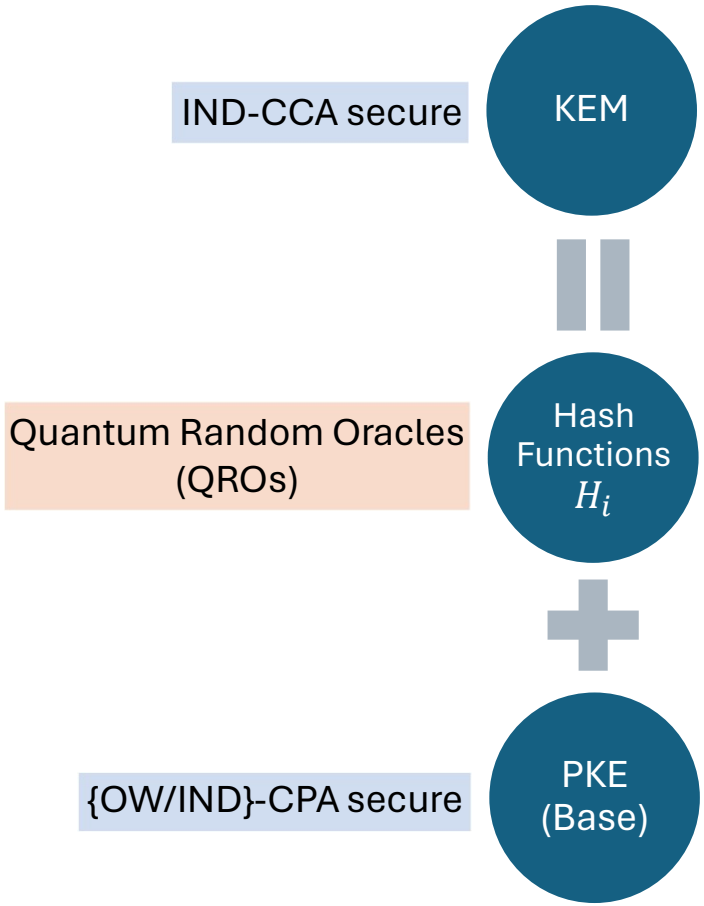
PKE (Base)

Shown by [Bindel-Hamburg-Hövelmanns-Hülsing-Persichetti@TCC'19]:

$$\text{Adv}^{\text{IND-CCA}}_{\text{FO}^{\not\perp}_m} \approx \text{Adv}^{\text{IND-CCA}}_{\text{FO}^{\not\perp}}$$

$\text{FO}^{\not\perp}_m$ [Hofheinz-Hövelmanns-Kiltz@TCC'17]

Analysis of Kyber



KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_{\$} \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_{\$} \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : sk′ ← (sk, s)	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : **return** $(c, \overline{k})$	5 : **if** $c' = c$ **then**
		6 : **return** $G_k(m')$
		7 : **else return** $G_k(s, c)$

$\mathrm{FO}^{\not\perp}_{m}$ [Hofheinz-Hövelmanns-Kiltz@TCC'17]

Analysis of Kyber

KGen′	Encap(pk)	Decap(sk, c)
1 : $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}$	1 : $m \leftarrow_{\$} \mathcal{M}$	1 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
2 : **return** $(\mathsf{pk}, \mathsf{sk})$	2 : $r \leftarrow G_r(m)$	2 : $r' \leftarrow G_r(m')$
	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	4 : $\overline{k} \leftarrow G_k(m)$	4 : $\overline{k'} \leftarrow G_k(m')$
	5 : **return** $(c, \overline{k})$	5 : **if** $c' = c$ **then**
		6 : **return** $\overline{k'}$
		7 : **else return** $\perp$

$\mathsf{FO}_m^{\perp}$ [Hofheinz-Hövelmanns-Kiltz@TCC'17]

KGen′	Encap(pk)	Decap(sk′, c)
1 : $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}$	1 : $m \leftarrow_{\$} \mathcal{M}$	1 : Parse $\mathsf{sk}' = (\mathsf{sk}, s)$
2 : $s \leftarrow_{\$} \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : $\mathsf{sk}' \leftarrow (\mathsf{sk}, s)$	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : **return** $(\mathsf{pk}, \mathsf{sk}')$	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : **return** $(c, \overline{k})$	5 : **if** $c' = c$ **then**
		6 : **return** $G_k(m')$
		7 : **else return** $G_k(s, c)$

$\mathsf{FO}_m^{\not\perp}$ [Hofheinz-Hövelmanns-Kiltz@TCC'17]

Analysis of Kyber

KGen′	Encap(pk)	Decap(sk, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_\$ \mathcal{M}$	1 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
2 : **return** (pk, sk)	2 : $r \leftarrow G_r(m)$	2 : $r' \leftarrow G_r(m')$
	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	4 : $\overline{k} \leftarrow G_k(m)$	4 : $\overline{k'} \leftarrow G_k(m')$
	5 : **return** $(c, \overline{k})$	5 : **if** $c' = c$ **then**
		6 : **return** $\overline{k'}$
		7 : **else return** $\perp$

$\mathsf{FO}_m^{\perp}$ [Hofheinz-Hövelmanns-Kiltz@TCC'17]

"Explicit rejection"

"Implicit rejection"

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_\$ \mathcal{M}$	1 : Parse $\mathsf{sk}' = (\mathsf{sk}, s)$
2 : $s \leftarrow_\$ \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : $\mathsf{sk}' \leftarrow (\mathsf{sk}, s)$	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : **return** $(c, \overline{k})$	5 : **if** $c' = c$ **then**
		6 : **return** $G_k(m')$
		7 : **else return** $G_k(s, c)$

$\mathsf{FO}_m^{\not\perp}$ [Hofheinz-Hövelmanns-Kiltz@TCC'17]

Analysis of Kyber

KGen′	Encap(pk)	Decap(sk, c)
1 : $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}$	1 : $m \leftarrow_{\$} \mathcal{M}$	1 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
2 : **return** $(\mathsf{pk}, \mathsf{sk})$	2 : $r \leftarrow G_r(m)$	2 : $r' \leftarrow G_r(m')$
	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	4 : $\overline{k} \leftarrow G_k(m)$	4 : $\overline{k'} \leftarrow G_k(m')$
	5 : **return** $(c, \overline{k})$	5 : **if** $c' = c$ **then**
		6 : **return** $\overline{k'}$
		7 : **else return** $\perp$

$\mathsf{FO}_m^{\perp}$ [Hofheinz-Hövelmanns-Kiltz@TCC'17]

Analysis of Kyber

KGen′	Encap(pk)	Decap(sk, c)
1 : $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}$	1 : $m \leftarrow_{\$} \mathcal{M}$	1 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
2 : **return** $(\mathsf{pk}, \mathsf{sk})$	2 : $r \leftarrow G_r(m)$	2 : $r' \leftarrow G_r(m')$
	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	4 : $\overline{k} \leftarrow G_k(m)$	4 : $\overline{k'} \leftarrow G_k(m')$
	5 : **return** $(c, \overline{k})$	5 : **if** $c' = c$ **then**
		6 : **return** $\overline{k'}$
		7 : **else return** $\perp$

$\mathsf{FO}_m^{\perp}$ [Hofheinz-Hövelmanns-Kiltz@TCC'17]

Analysis of Kyber

KGen′	Encap(pk)	Decap(sk, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_{\$} \mathcal{M}$	1 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
2 : **return** (pk, sk)	2 : $r \leftarrow G_r(m)$	2 : $r' \leftarrow G_r(m')$
	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	4 : $\overline{k} \leftarrow G_k(m)$	4 : $\overline{k'} \leftarrow G_k(m')$
	5 : **return** $(c, \overline{k})$	5 : **if** $c' = c$ **then**
		6 : **return** $\overline{k'}$
		7 : **else return** $\perp$

$\mathsf{FO}_m^{\perp}$ [Hofheinz-Hövelmanns-Kiltz@TCC'17]

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_{\$} \mathcal{M}$	1 : Parse $\mathsf{sk}' = (\mathsf{sk}, s)$
2 : $s \leftarrow_{\$} \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : $\mathsf{sk}' \leftarrow (\mathsf{sk}, s)$	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : $k \leftarrow H'(\overline{k}, H(c))$	5 : $\overline{k'} \leftarrow G_k(m')$
	6 : **return** (c, k)	6 : **if** $c' = c$ **then**
		7 : **return** $H'(\overline{k'}, H(c))$
		8 : **else return** $H'(s, H(c))$

Kyber (simplified)

Ignoring initial hashes "$H(m)$" and "$H(pk)$" in $Encap$.

Analysis of Kyber

KGen′	Encap(pk)	Decap(sk, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_\$ \mathcal{M}$	1 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
2 : **return** (pk, sk)	2 : $r \leftarrow G_r(m)$	2 : $r' \leftarrow G_r(m')$
	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	4 : $\overline{k} \leftarrow G_k(m)$	4 : $\overline{k'} \leftarrow G_k(m')$
	5 : **return** $(c, \overline{k})$	5 : **if** $c' = c$ **then**
		6 : **return** $\overline{k'}$
		7 : **else return** $\perp$

$\mathsf{FO}_m^\perp$ [Hofheinz-Hövelmanns-Kiltz@TCC'17]

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_\$ \mathcal{M}$	1 : Parse $\mathsf{sk}' = (\mathsf{sk}, s)$
2 : $s \leftarrow_\$ \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : $\mathsf{sk}' \leftarrow (\mathsf{sk}, s)$	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : $k \leftarrow H'(\overline{k}, H(c))$	5 : $\overline{k'} \leftarrow G_k(m')$
	6 : **return** (c, k)	6 : **if** $c' = c$ **then**
		7 : **return** $H'(\overline{k'}, H(c))$
		8 : **else return** $H'(s, H(c))$

Kyber (simplified)

Analysis of Kyber

KGen′	Encap(pk)	Decap(sk, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_\$ \mathcal{M}$	1 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
2 : **return** (pk, sk)	2 : $r \leftarrow G_r(m)$	2 : $r' \leftarrow G_r(m')$
	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	4 : $\overline{k} \leftarrow G_k(m)$	4 : $\overline{k'} \leftarrow G_k(m')$
	5 : **return** $(c, \overline{k})$	5 : **if** $c' = c$ **then**
		6 : **return** $\overline{k'}$
		7 : **else return** $\perp$

$\mathsf{FO}_m^{\perp}$ [Hofheinz-Hövelmanns-Kiltz@TCC'17]

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_\$ \mathcal{M}$	1 : Parse $\mathsf{sk}' = (\mathsf{sk}, s)$
2 : $s \leftarrow_\$ \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : $\mathsf{sk}' \leftarrow (\mathsf{sk}, s)$	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : $k \leftarrow H'(\overline{k}, H(c))$	5 : $\overline{k'} \leftarrow G_k(m')$
	6 : **return** (c, k)	6 : **if** $c' = c$ **then**
		7 : **return** $H'(\overline{k'}, H(c))$
		8 : **else return** $H'(s, H(c))$

Kyber (simplified)

Analysis of Kyber

KGen′	Encap(pk)	Decap(sk, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_\$ \mathcal{M}$	1 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
2 : **return** (pk, sk)	2 : $r \leftarrow G_r(m)$	2 : $r' \leftarrow G_r(m')$
	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	4 : $\overline{k} \leftarrow G_k(m)$	4 : $\overline{k'} \leftarrow G_k(m')$
	5 : **return** $(c, \overline{k})$	5 : **if** $c' = c$ **then**
		6 : **return** $\overline{k'}$
		7 : **else return** $\perp$

$\mathrm{FO}_m^{\perp}$ [Hofheinz-Hövelmanns-Kiltz@TCC'17]

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_\$ \mathcal{M}$	1 : Parse $\mathsf{sk}' = (\mathsf{sk}, s)$
2 : $s \leftarrow_\$ \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : $\mathsf{sk}' \leftarrow (\mathsf{sk}, s)$	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : $k \leftarrow H'(\overline{k}, H(c))$	5 : $\overline{k'} \leftarrow G_k(m')$
	6 : **return** (c, k)	6 : **if** $c' = c$ **then**
		7 : **return** $H'(\overline{k'}, H(c))$
		8 : **else return** $H'(s, H(c))$

Kyber (simplified)

$(c, \overline{k})$

(c, k)

H, H'
$k \leftarrow H'(\overline{k}, H(c))$

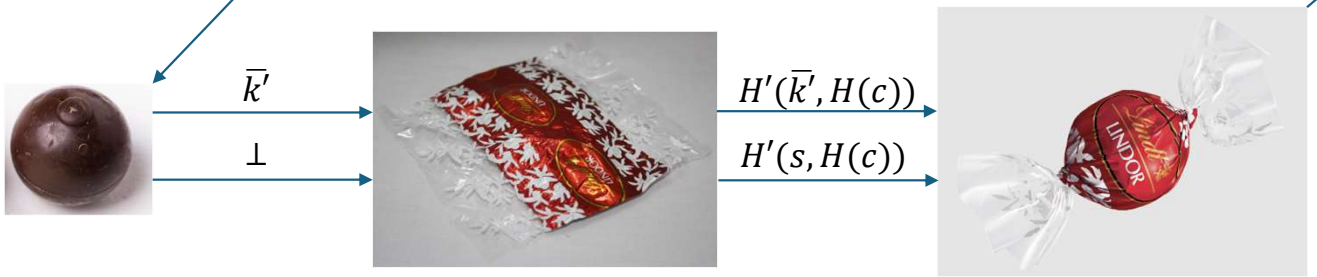
Analysis of Kyber

KGen′	Encap(pk)	Decap(sk, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_{\$} \mathcal{M}$	1 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
2 : **return** (pk, sk)	2 : $r \leftarrow G_r(m)$	2 : $r' \leftarrow G_r(m')$
	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	4 : $\overline{k} \leftarrow G_k(m)$	4 : $\overline{k'} \leftarrow G_k(m')$
	5 : **return** $(c, \overline{k})$	5 : **if** $c' = c$ **then**
		6 : **return** $\overline{k'}$
		7 : **else return** $\perp$

$\mathsf{FO}_m^{\perp}$ [Hofheinz-Hövelmanns-Kiltz@TCC'17]

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_{\$} \mathcal{M}$	1 : Parse $\mathsf{sk}' = (\mathsf{sk}, s)$
2 : $s \leftarrow_{\$} \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : $\mathsf{sk}' \leftarrow (\mathsf{sk}, s)$	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : $k \leftarrow H'(\overline{k}, H(c))$	5 : $\overline{k'} \leftarrow G_k(m')$
	6 : **return** (c, k)	6 : **if** $c' = c$ **then**
		7 : **return** $H'(\overline{k'}, H(c))$
		8 : **else return** $H'(s, H(c))$

Kyber (simplified)



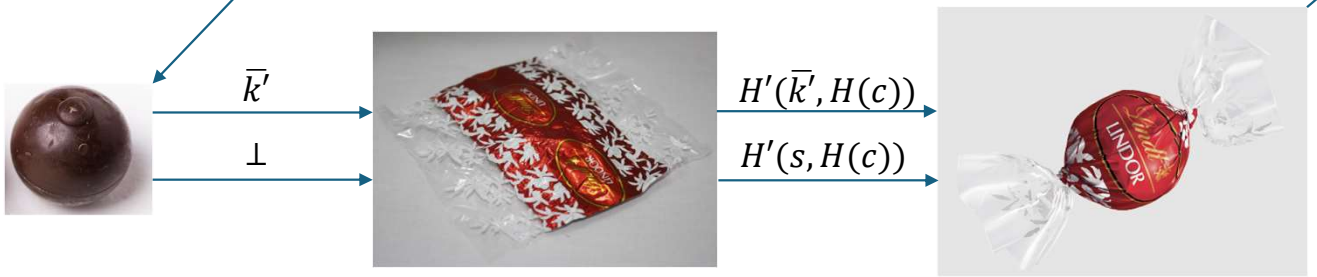
Analysis of Kyber

KGen′	Encap(pk)	Decap(sk, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_{\$} \mathcal{M}$	1 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
2 : **return** (pk, sk)	2 : $r \leftarrow G_r(m)$	2 : $r' \leftarrow G_r(m')$
	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	4 : $\overline{k} \leftarrow G_k(m)$	4 : $\overline{k'} \leftarrow G_k(m')$
	5 : **return** $(c, \overline{k})$	5 : **if** $c' = c$ **then**
		6 : **return** $\overline{k'}$
		7 : **else return** $\perp$

$\mathsf{FO}_m^{\perp}$ [Hofheinz-Hövelmanns-Kiltz@TCC'17]

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_{\$} \mathcal{M}$	1 : Parse $\mathsf{sk}' = (\mathsf{sk}, s)$
2 : $s \leftarrow_{\$} \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : $\mathsf{sk}' \leftarrow (\mathsf{sk}, s)$	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : $k \leftarrow H'(\overline{k}, H(c))$	5 : $\overline{k'} \leftarrow G_k(m')$
	6 : **return** (c, k)	6 : **if** $c' = c$ **then**
		7 : **return** $H'(\overline{k'}, H(c))$
		8 : **else return** $H'(s, H(c))$

Kyber (simplified)



IND-CCA security of $\mathsf{FO}_m^{\perp}$ KEMs in the QROM $\Rightarrow$ IND-CCA security of Kyber in the QROM

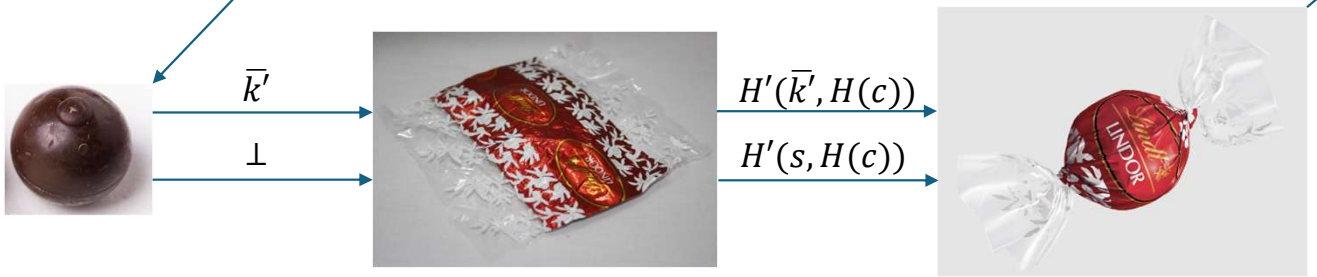
Analysis of Kyber

KGen′	Encap(pk)	Decap(sk, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_{\$} \mathcal{M}$	1 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
2 : **return** (pk, sk)	2 : $r \leftarrow G_r(m)$	2 : $r' \leftarrow G_r(m')$
	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	4 : $\overline{k} \leftarrow G_k(m)$	4 : $\overline{k'} \leftarrow G_k(m')$
	5 : **return** $(c, \overline{k})$	5 : **if** $c' = c$ **then**
		6 : **return** $\overline{k'}$
		7 : **else return** $\perp$

$\mathsf{FO}_m^{\perp}$ [Hofheinz-Hövelmanns-Kiltz@TCC'17]

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_{\$} \mathcal{M}$	1 : Parse $\mathsf{sk}' = (\mathsf{sk}, s)$
2 : $s \leftarrow_{\$} \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : $\mathsf{sk}' \leftarrow (\mathsf{sk}, s)$	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : $k \leftarrow H'(\overline{k}, H(c))$	5 : $\overline{k'} \leftarrow G_k(m')$
	6 : **return** (c, k)	6 : **if** $c' = c$ **then**
		7 : **return** $H'(\overline{k'}, H(c))$
		8 : **else return** $H'(s, H(c))$

Kyber (simplified)



IND-CCA security of $\mathsf{FO}_m^{\perp}$ KEMs in the QROM $\Rightarrow$ IND-CCA security of Kyber in the QROM

Analysis of Kyber

Tighter Proofs of CCA Security in the Quantum Random Oracle Model

Measure-Rewind-Measure: Tighter Quantum Random Oracle Model Proofs for One-Way to Hiding and CCA Security

Nina

Veronika Kuchta[1], Amin Sakzad[1(✉)], Damien Stehlé[2,3], Ron Steinfeld[1(✉)] and Shi-Feng Sun[1,4]

[4] Ein [1] Faculty of Information Technology, Monash University, Melbourne, Australia
{amin.sakzad,ron.steinfeld}@monash.edu
[2] Univ. Lyon, EnsL, UCBL, CNRS, Inria, LIP, 69342 Lyon Cedex 07, France
[3] Institut Universitaire de France, Paris, France
[4] Data61, CSIRO, Canberra, Australia [Eurocrypt'20]

Non-tight proofs, compared to $\mathsf{FO}^{\not\perp}_m$.

IND-CCA security of $\mathsf{FO}^{\perp}_m$ KEMs in the QROM $\Rightarrow$ IND-CCA security of Kyber in the QROM

Analysis of Kyber

Tighter Proofs of CCA Security in the Quantum Random Oracle Model

Measure-Rewind-Measure: Tighter Quantum Random Oracle Model Proofs for One-Way to Hiding and CCA Security

Nina

Veronika Kuchta[1], Amin Sakzad[1(✉)], Damien Stehlé[2,3], Ron Steinfeld[1(✉)] and Shi-Feng Sun[1,4]

[4] Ein

[1] Faculty of Information Technology, Monash University, Melbourne, Australia
{amin.sakzad,ron.steinfeld}@monash.edu
[2] Univ. Lyon, EnsL, UCBL, CNRS, Inria, LIP, 69342 Lyon Cedex 07, France
[3] Institut Universitaire de France, Paris, France
[4] Data61, CSIRO, Canberra, Australia [Eurocrypt'20]

Post-Quantum Verification of Fujisaki-Okamoto

Dominique Unruh(✉)

University of Tartu, Tartu, Estonia
unruh@ut.ee

Abstract. We present a computer-verified formalization of the post-quantum security proof of the Fujisaki-Okamoto transform (as analyzed by Hövelmanns, Kiltz, Schäge, and Unruh, PKC 2020). The formalization is done in quantum relational Hoare logic and checked in the `qrhl-tool` (Unruh, POPL 2019). [Asiacrypt'20]

Non-tight proofs, compared to $\mathsf{FO}^{\not\perp}_m$.

Lacking formal verification, in contrast to $\mathsf{FO}^{\not\perp}_m$.

IND-CCA security of $\mathsf{FO}^{\perp}_m$ KEMs in the QROM $\Rightarrow$ IND-CCA security of Kyber in the QROM

Analysis of Kyber

KGen′	Encap(pk)	Decap(sk, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_{\$} \mathcal{M}$	1 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
2 : **return** (pk, sk)	2 : $r \leftarrow G_r(m)$	2 : $r' \leftarrow G_r(m')$
	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	4 : $\overline{k} \leftarrow G_k(m)$	4 : **if** $c' = c$ **then**
	5 : **return** $(c, \overline{k})$	5 : **return** $G_k(m')$
		6 : **else return** $\perp$

$\mathsf{FO}_m^{\perp}$[Hofheinz-Hövelmanns-Kiltz@TCC'17]

"Explicit rejection"

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_{\$} \mathcal{M}$	1 : Parse $\mathsf{sk}' = (\mathsf{sk}, s)$
2 : $s \leftarrow_{\$} \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : $\mathsf{sk}' \leftarrow (\mathsf{sk}, s)$	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : $k \leftarrow H'(\overline{k}, H(c))$	5 : $\overline{k'} \leftarrow G_k(m')$
	6 : **return** (c, k)	6 : **if** $c' = c$ **then**
		7 : **return** $H'(\overline{k'}, H(c))$
		8 : **else return** $H'(s, H(c))$

Kyber (simplified)

IND-CCA security of $\mathsf{FO}_m^{\perp}$ KEMs in the QROM $\Rightarrow$ IND-CCA security of Kyber in the QROM

Analysis of Kyber

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_\$ \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_\$ \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : sk′ ← (sk, s)	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : return (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : return $(c, \overline{k})$	5 : if $c' = c$ then
		6 : return $G_k(m')$
		7 : else return $G_k(s, c)$

$\mathsf{FO}_m^{\not\perp}$[Hofheinz-Hövelmanns-Kiltz@TCC'17]

"Implicit rejection"

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_\$ \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_\$ \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : sk′ ← (sk, s)	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : return (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : $k \leftarrow H'(\overline{k}, H(c))$	5 : $\overline{k'} \leftarrow G_k(m')$
	6 : return (c, k)	6 : if $c' = c$ then
		7 : return $H'(\overline{k'}, H(c))$
		8 : else return $H'(s, H(c))$

Kyber (simplified)

IND-CCA security of $\mathsf{FO}_m^{\not\perp}$ KEMs in the QROM $\Rightarrow$?

Analysis of Kyber

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_{\$} \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_{\$} \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : sk′ ← (sk, s)	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : **return** $(c, \overline{k})$	5 : **if** $c' = c$ **then**
		6 : **return** $G_k(m')$
		7 : **else return** $G_k(s, c)$

$\mathsf{FO}_m^{\not\perp}$[Hofheinz-Hövelmanns-Kiltz@TCC'17]

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_{\$} \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_{\$} \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : sk′ ← (sk, s)	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : $k \leftarrow H'(\overline{k}, H(c))$	5 : $\overline{k'} \leftarrow G_k(m')$
	6 : **return** (c, k)	6 : **if** $c' = c$ **then**
		7 : **return** $H'(\overline{k'}, H(c))$
		8 : **else return** $H'(s, H(c))$

Kyber (simplified)

IND-CCA security of $\mathsf{FO}_m^{\not\perp}$ KEMs in the QROM $\Rightarrow$?

Analysis of Kyber

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_\$ \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_\$ \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow$ Dec(sk, c)
3 : sk′ ← (sk, s)	3 : $c \leftarrow$ Enc(pk, m; r)	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow$ Enc(pk, m'; r')
	5 : **return** $(c, \overline{k})$	5 : **if** $c' = c$ **then**
		6 : **return** $G_k(m')$
		7 : **else return** $G_k(s, c)$

$\text{FO}_m^{\not\perp}$[Hofheinz-Hövelmanns-Kiltz@TCC'17]

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_\$ \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_\$ \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow$ Dec(sk, c)
3 : sk′ ← (sk, s)	3 : $c \leftarrow$ Enc(pk, m; r)	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow$ Enc(pk, m'; r')
	5 : $k \leftarrow H'(\overline{k}, H(c))$	5 : $\overline{k'} \leftarrow G_k(m')$
	6 : **return** (c, k)	6 : **if** $c' = c$ **then**
		7 : **return** $H'(\overline{k'}, H(c))$
		8 : **else return** $H'(s, H(c))$

Kyber (simplified)

$\overline{k}' = G_k(m')$?
or
$\overline{k}' = G_k(s, c)$?

$\overline{k}'$

?

IND-CCA security of $\text{FO}_m^{\not\perp}$ KEMs in the QROM $\Rightarrow$?

Analysis of Kyber

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_\$ \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_\$ \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : sk′ ← (sk, s)	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : return (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : return $(c, \overline{k})$	5 : if $c' = c$ then
		6 : return $G_k(m')$
		7 : else return $G_k(s, c)$

$\mathsf{FO}_m^{\not\perp}$[Hofheinz-Hövelmanns-Kiltz@TCC'17]

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_\$ \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_\$ \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : sk′ ← (sk, s)	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : return (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : $k \leftarrow H'(\overline{k}, H(c))$	5 : $\overline{k'} \leftarrow G_k(m')$
	6 : return (c, k)	6 : if $c' = c$ then
		7 : return $H'(\overline{k'}, H(c))$
		8 : else return $H'(s, H(c))$

Kyber (simplified)

IND-CCA security of $\mathsf{FO}_m^{\not\perp}$ KEMs in the QROM $\Rightarrow$?

Analysis of Kyber

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_{\$} \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_{\$} \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : sk′ ← (sk, s)	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : **return** $(c, \overline{k})$	5 : **if** $c' = c$ **then**
		6 : **return** $G_k(m')$
		7 : **else return** $\overline{H}(H(c))$

$\mathsf{FO}_m^{\not\perp}$
modified

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_{\$} \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_{\$} \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : sk′ ← (sk, s)	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : $k \leftarrow H'(\overline{k}, H(c))$	5 : $\overline{k'} \leftarrow G_k(m')$
	6 : **return** (c, k)	6 : **if** $c' = c$ **then**
		7 : **return** $H'(\overline{k'}, H(c))$
		8 : **else return** $H'(\overline{H}(H(c)), H(c))$

Kyber (simplified)
modified

IND-CCA security of $\mathsf{FO}_m^{\not\perp}$ KEMs in the QROM $\Rightarrow$?

Analysis of Kyber

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_\$ \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_\$ \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : sk′ ← (sk, s)	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : **return** $(c, \overline{k})$	5 : **if** $c' = c$ **then**
		6 : **return** $G_k(m')$
		7 : **else return** $\overline{H}(H(c))$

$\mathsf{FO}_m^{\not\perp}$
modified

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_\$ \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_\$ \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : sk′ ← (sk, s)	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : $k \leftarrow H'(\overline{k}, H(c))$	5 : $\overline{k'} \leftarrow G_k(m')$
	6 : **return** (c, k)	6 : **if** $c' = c$ **then**
		7 : **return** $H'(\overline{k'}, H(c))$
		8 : **else return** $H'(\overline{H}(H(c)), H(c))$

Kyber (simplified)
modified

$\overline{k}'$

$H'(\overline{k}', H(c))$

IND-CCA security of $\mathsf{FO}_m^{\not\perp}$ KEMs in the QROM $\Rightarrow$?

Analysis of Kyber

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_\$ \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_\$ \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : sk′ ← (sk, s)	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : **return** $(c, \overline{k})$	5 : **if** $c' = c$ **then**
		6 : **return** $G_k(m')$
		7 : **else return** $\overline{H}(H(c))$

$\mathsf{FO}_m^{\not\perp}$
modified

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_\$ \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_\$ \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : sk′ ← (sk, s)	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : $k \leftarrow H'(\overline{k}, H(c))$	5 : $\overline{k'} \leftarrow G_k(m')$
	6 : **return** (c, k)	6 : **if** $c' = c$ **then**
		7 : **return** $H'(\overline{k'}, H(c))$
		8 : **else return** $H'(\overline{H}(H(c)), H(c))$

Kyber (simplified)
modified

$\overline{k}' = G_k(m')$

$\overline{k}'$

$H'(\overline{k}', H(c))$

IND-CCA security of $\mathsf{FO}_m^{\not\perp}$ KEMs in the QROM $\Rightarrow$?

Analysis of Kyber

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_\$ \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_\$ \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : sk′ ← (sk, s)	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : **return** $(c, \overline{k})$	5 : **if** $c' = c$ **then**
		6 : **return** $G_k(m')$
		7 : **else return** $\overline{H}(H(c))$

$\mathsf{FO}_m^{\not\perp}$
modified

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_\$ \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_\$ \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : sk′ ← (sk, s)	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : $k \leftarrow H'(\overline{k}, H(c))$	5 : $\overline{k'} \leftarrow G_k(m')$
	6 : **return** (c, k)	6 : **if** $c' = c$ **then**
		7 : **return** $H'(\overline{k'}, H(c))$
		8 : **else return** $H'(\overline{H}(H(c)), H(c))$

Kyber (simplified)
modified

$\overline{k}' = \overline{H}(H(c))$

$\overline{k}'$

$H'(\overline{k}', H(c))$

IND-CCA security of $\mathsf{FO}_m^{\not\perp}$ KEMs in the QROM $\Rightarrow$?

Analysis of Kyber

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_\$ \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_\$ \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : sk′ ← (sk, s)	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : **return** $(c, \overline{k})$	5 : **if** $c' = c$ **then**
		6 : **return** $G_k(m')$
		7 : **else return** $\overline{H}(H(c))$

$\mathsf{FO}_m^{\not\perp}$
modified

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_\$ \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_\$ \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : sk′ ← (sk, s)	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : $k \leftarrow H'(\overline{k}, H(c))$	5 : $\overline{k'} \leftarrow G_k(m')$
	6 : **return** (c, k)	6 : **if** $c' = c$ **then**
		7 : **return** $H'(\overline{k'}, H(c))$
		8 : **else return** $H'(\overline{H}(H(c)), H(c))$

Kyber (simplified)
modified

$\overline{k}'$

$H'(\overline{k}', H(c))$

IND-CCA security of $\mathsf{FO}_m^{\not\perp}$ KEMs in the QROM $\Rightarrow$?

Analysis of Kyber

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_\$ \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_\$ \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : sk′ ← (sk, s)	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : **return** $(c, \overline{k})$	5 : **if** $c' = c$ **then**
		6 : **return** $G_k(m')$
		7 : **else return** $\overline{H}(H(c))$

$\mathsf{FO}_m^{\not\perp}$ modified

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_\$ \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_\$ \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : sk′ ← (sk, s)	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : $k \leftarrow H'(\overline{k}, H(c))$	5 : $\overline{k'} \leftarrow G_k(m')$
	6 : **return** (c, k)	6 : **if** $c' = c$ **then**
		7 : **return** $H'(\overline{k'}, H(c))$
		8 : **else return** $H'(\overline{H}(H(c)), H(c))$

Kyber (simplified) modified

$\mathsf{FO}_m^{\not\perp}$

$\approx$

$\overline{k}'$

$H'(\overline{k}', H(c))$

We also show that IND-CCA security of both KEMs are computationally equivalent.

IND-CCA security of $\mathsf{FO}_m^{\not\perp}$ KEMs in the QROM $\Rightarrow$?

Analysis of Kyber

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_{\$} \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_{\$} \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : sk′ ← (sk, s)	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : **return** $(c, \overline{k})$	5 : **if** $c' = c$ **then**
		6 : **return** $G_k(m')$
		7 : **else return** $G_k(s, c)$

$\mathsf{FO}_m^{\not\perp}$

Analysis of Kyber

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_\$ \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_\$ \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : sk′ ← (sk, s)	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : **return** $(c, \overline{k})$	5 : **if** $c' = c$ **then**
		6 : **return** $G_k(m')$
		7 : **else return** $G_k(s, c)$

$\mathsf{FO}_m^{\not\perp}$

IND-CCA adversary

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_\$ \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_\$ \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : sk′ ← (sk, s)	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : **return** $(c, \overline{k})$	5 : **if** $c' = c$ **then**
		6 : **return** $G_k(m')$
		7 : **else return** $\overline{H}(H(c))$

$\mathsf{FO}_m^{\not\perp}$ modified

Analysis of Kyber

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_\$ \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_\$ \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : sk′ ← (sk, s)	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : **return** $(c, \overline{k})$	5 : **if** $c' = c$ **then**
		6 : **return** $G_k(m')$
		7 : **else return** $G_k(s, c)$

$\mathsf{FO}_m^{\not\perp}$

IND-CCA adversary

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_\$ \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_\$ \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : sk′ ← (sk, s)	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : **return** $(c, \overline{k})$	5 : **if** $c' = c$ **then**
		6 : **return** $G_k(m')$
		7 : **else return** $H''(c)$

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_\$ \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_\$ \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : sk′ ← (sk, s)	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : **return** $(c, \overline{k})$	5 : **if** $c' = c$ **then**
		6 : **return** $G_k(m')$
		7 : **else return** $\overline{H}(H(c))$

$\mathsf{FO}_m^{\not\perp}$ modified

Analysis of Kyber

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_\$ \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_\$ \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : sk′ ← (sk, s)	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : **return** $(c, \overline{k})$	5 : **if** $c' = c$ **then**
		6 : **return** $G_k(m')$
		7 : **else return** $G_k(s, c)$

$\mathsf{FO}_m^{\not\perp}$

... pseudorandomness of QRO "$G_k(s,\cdot)$".

≈

IND-CCA adversary

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_\$ \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_\$ \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : sk′ ← (sk, s)	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : **return** $(c, \overline{k})$	5 : **if** $c' = c$ **then**
		6 : **return** $G_k(m')$
		7 : **else return** $H''(c)$

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_\$ \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_\$ \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : sk′ ← (sk, s)	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : **return** $(c, \overline{k})$	5 : **if** $c' = c$ **then**
		6 : **return** $G_k(m')$
		7 : **else return** $\overline{H}(H(c))$

$\mathsf{FO}_m^{\not\perp}$ modified

Analysis of Kyber

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_\$ \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_\$ \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : sk′ ← (sk, s)	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : **return** $(c, \overline{k})$	5 : **if** $c' = c$ **then**
		6 : **return** $G_k(m')$
		7 : **else return** $G_k(s, c)$

$\mathsf{FO}_m^{\not\perp}$

IND-CCA adversary

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_\$ \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_\$ \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : sk′ ← (sk, s)	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : **return** $(c, \overline{k})$	5 : **if** $c' = c$ **then**
		6 : **return** $G_k(m')$
		7 : **else return** $H''(c)$

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_\$ \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_\$ \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : sk′ ← (sk, s)	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : **return** $(c, \overline{k})$	5 : **if** $c' = c$ **then**
		6 : **return** $G_k(m')$
		7 : **else return** $\overline{H}(H(c))$

$\mathsf{FO}_m^{\not\perp}$ modified

Analysis of Kyber

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_\$ \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_\$ \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : sk′ ← (sk, s)	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : **return** $(c, \overline{k})$	5 : **if** $c' = c$ **then**
		6 : **return** $G_k(m')$
		7 : **else return** $G_k(s, c)$

$\mathsf{FO}_m^{\not\perp}$

IND-CCA adversary

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_\$ \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_\$ \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : sk′ ← (sk, s)	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : **return** $(c, \overline{k})$	5 : **if** $c' = c$ **then**
		6 : **return** $G_k(m')$
		7 : **else return** $H''(c)$

... collision-resistance of QRO "$H(\cdot)$".

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_\$ \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_\$ \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : sk′ ← (sk, s)	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : **return** $(c, \overline{k})$	5 : **if** $c' = c$ **then**
		6 : **return** $G_k(m')$
		7 : **else return** $\overline{H}(H(c))$

$\mathsf{FO}_m^{\not\perp}$ modified

Analysis of Kyber

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_\$ \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_\$ \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : sk′ ← (sk, s)	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : return (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : return $(c, \overline{k})$	5 : if $c' = c$ then
		6 : return $G_k(m')$
		7 : else return $G_k(s, c)$

$\mathsf{FO}_m^{\not\perp}$

... pseudorandomness of QRO "$G_k(s,\cdot)$".

≈

≈

IND-CCA adversary

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_\$ \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_\$ \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : sk′ ← (sk, s)	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : return (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : return $(c, \overline{k})$	5 : if $c' = c$ then
		6 : return $G_k(m')$
		7 : else return $H''(c)$

≈

... collision-resistance of QRO "$H(\cdot)$".

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_\$ \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_\$ \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : sk′ ← (sk, s)	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : return (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : return $(c, \overline{k})$	5 : if $c' = c$ then
		6 : return $G_k(m')$
		7 : else return $\overline{H}(H(c))$

$\mathsf{FO}_m^{\not\perp}$ modified

Analysis of Kyber

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_\$ \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_\$ \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : sk′ ← (sk, s)	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : **return** $(c, \overline{k})$	5 : **if** $c' = c$ **then**
		6 : **return** $G_k(m')$
		7 : **else return** $\overline{H}(H(c))$

$\mathsf{FO}_m^{\not\perp}$ modified

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_\$ \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_\$ \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : sk′ ← (sk, s)	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : $k \leftarrow H'(\overline{k}, H(c))$	5 : $\overline{k'} \leftarrow G_k(m')$
	6 : **return** (c, k)	6 : **if** $c' = c$ **then**
		7 : **return** $H'(\overline{k'}, H(c))$
		8 : **else return** $H'(\overline{H}(H(c)), H(c))$

Kyber (simplified) modified

$\mathsf{FO}_m^{\not\perp}$

$\approx$

$\overline{k}'$

$H'(\overline{k}', H(c))$

We also show that IND-CCA security of both KEMs are computationally equivalent.

IND-CCA security of $\mathsf{FO}_m^{\not\perp}$ KEMs in the QROM $\Rightarrow$?

Analysis of Kyber

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_\$ \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_\$ \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : sk′ ← (sk, s)	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : **return** $(c, \overline{k})$	5 : **if** $c' = c$ **then**
		6 : **return** $G_k(m')$
		7 : **else return** $\overline{H}(H(c))$

$\mathsf{FO}_m^{\not\perp}$ modified

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_\$ \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_\$ \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : sk′ ← (sk, s)	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : $k \leftarrow H'(\overline{k}, H(c))$	5 : $\overline{k'} \leftarrow G_k(m')$
	6 : **return** (c, k)	6 : **if** $c' = c$ **then**
		7 : **return** $H'(\overline{k'}, H(c))$
		8 : **else return** $H'(\overline{H}(H(c)), H(c))$

Kyber (simplified) modified

Kyber (simplified)

$\mathsf{FO}_m^{\not\perp}$

≈

$\overline{k}'$

$H'(\overline{k}', H(c))$

≈

We also show that IND-CCA security of both KEMs are computationally equivalent.

IND-CCA security of $\mathsf{FO}_m^{\not\perp}$ KEMs in the QROM ⇒ ?

We also show that IND-CCA security of both KEMs are computationally equivalent.

Analysis of Kyber

KGen′

1 : $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}$
2 : $s \leftarrow_{\$} \mathcal{M}$
3 : $\mathsf{sk}' \leftarrow (\mathsf{sk}, s)$
4 : **return** $(\mathsf{pk}, \mathsf{sk}')$

Encap(pk)

1 : $m \leftarrow_{\$} \mathcal{M}$
2 : $r \leftarrow G_r(m)$
3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$
4 : $\overline{k} \leftarrow G_k(m)$
5 : $k \leftarrow H'(\overline{k}, H(c))$
6 : **return** (c, k)

Decap(sk', c)

1 : Parse $\mathsf{sk}' = (\mathsf{sk}, s)$
2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : $r' \leftarrow G_r(m')$
4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
5 : $\overline{k'} \leftarrow G_k(m')$
6 : **if** $c' = c$ **then**
7 : **return** $H'(\overline{k'}, H(c))$
8 : **else return** $H'(s, H(c))$

(simplified) Kyber

IND-CCA adversary

Analysis of Kyber

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_{\$} \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_{\$} \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : sk′ ← (sk, s)	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : return (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : $k \leftarrow H'(\overline{k}, H(c))$	5 : $\overline{k'} \leftarrow G_k(m')$
	6 : return (c, k)	6 : if $c' = c$ then
		7 : return $H'(\overline{k'}, H(c))$
		8 : else return $H'(s, H(c))$

(simplified) Kyber

IND-CCA adversary

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_{\$} \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_{\$} \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : sk′ ← (sk, s)	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : return (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : $k \leftarrow H'(\overline{k}, H(c))$	5 : $\overline{k'} \leftarrow G_k(m')$
	6 : return (c, k)	6 : if $c' = c$ then
		7 : return $H'(\overline{k'}, H(c))$
		8 : else return $H'(\overline{H}(H(c)), H(c))$

(simplified) Kyber modified

Analysis of Kyber

KGen′

1 : $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}$
2 : $s \leftarrow_{\$} \mathcal{M}$
3 : $\mathsf{sk}' \leftarrow (\mathsf{sk}, s)$
4 : **return** $(\mathsf{pk}, \mathsf{sk}')$

Encap(pk)

1 : $m \leftarrow_{\$} \mathcal{M}$
2 : $r \leftarrow G_r(m)$
3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$
4 : $\overline{k} \leftarrow G_k(m)$
5 : $k \leftarrow H'(\overline{k}, H(c))$
6 : **return** (c, k)

Decap(sk′, c)

1 : Parse $\mathsf{sk}' = (\mathsf{sk}, s)$
2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : $r' \leftarrow G_r(m')$
4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
5 : $\overline{k'} \leftarrow G_k(m')$
6 : **if** $c' = c$ **then**
7 : **return** $H'(\overline{k'}, H(c))$
8 : **else return** $H'(s, H(c))$

(simplified) Kyber

IND-CCA adversary

KGen′

1 : $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}$
2 : $s \leftarrow_{\$} \mathcal{M}$
3 : $\mathsf{sk}' \leftarrow (\mathsf{sk}, s)$
4 : **return** $(\mathsf{pk}, \mathsf{sk}')$

Encap(pk)

1 : $m \leftarrow_{\$} \mathcal{M}$
2 : $r \leftarrow G_r(m)$
3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$
4 : $\overline{k} \leftarrow G_k(m)$
5 : $k \leftarrow H'(\overline{k}, H(c))$
6 : **return** (c, k)

Decap(sk′, c)

1 : Parse $\mathsf{sk}' = (\mathsf{sk}, s)$
2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : $r' \leftarrow G_r(m')$
4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
5 : $\overline{k'} \leftarrow G_k(m')$
6 : **if** $c' = c$ **then**
7 : **return** $H'(\overline{k'}, H(c))$
8 : **else return** $H''(H(c))$

KGen′

1 : $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}$
2 : $s \leftarrow_{\$} \mathcal{M}$
3 : $\mathsf{sk}' \leftarrow (\mathsf{sk}, s)$
4 : **return** $(\mathsf{pk}, \mathsf{sk}')$

Encap(pk)

1 : $m \leftarrow_{\$} \mathcal{M}$
2 : $r \leftarrow G_r(m)$
3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$
4 : $\overline{k} \leftarrow G_k(m)$
5 : $k \leftarrow H'(\overline{k}, H(c))$
6 : **return** (c, k)

Decap(sk′, c)

1 : Parse $\mathsf{sk}' = (\mathsf{sk}, s)$
2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : $r' \leftarrow G_r(m')$
4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
5 : $\overline{k'} \leftarrow G_k(m')$
6 : **if** $c' = c$ **then**
7 : **return** $H'(\overline{k'}, H(c))$
8 : **else return** $H'(\overline{H}(H(c)), H(c))$

(simplified) Kyber modified

Analysis of Kyber

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_{\$} \mathcal{M}$	1 : Parse $\mathsf{sk}' = (\mathsf{sk}, s)$
2 : $s \leftarrow_{\$} \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : $\mathsf{sk}' \leftarrow (\mathsf{sk}, s)$	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : **return** $(\mathsf{pk}, \mathsf{sk}')$	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : $k \leftarrow H'(\overline{k}, H(c))$	5 : $\overline{k'} \leftarrow G_k(m')$
	6 : **return** (c, k)	6 : **if** $c' = c$ **then**
		7 : **return** $H'(\overline{k'}, H(c))$
		8 : **else return** $H'(s, H(c))$

(simplified) Kyber

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_{\$} \mathcal{M}$	1 : Parse $\mathsf{sk}' = (\mathsf{sk}, s)$
2 : $s \leftarrow_{\$} \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : $\mathsf{sk}' \leftarrow (\mathsf{sk}, s)$	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : **return** $(\mathsf{pk}, \mathsf{sk}')$	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : $k \leftarrow H'(\overline{k}, H(c))$	5 : $\overline{k'} \leftarrow G_k(m')$
	6 : **return** (c, k)	6 : **if** $c' = c$ **then**
		7 : **return** $H'(\overline{k'}, H(c))$
		8 : **else return** $H'(\overline{H}(H(c)), H(c))$

(simplified) Kyber modified

IND-CCA adversary

≈

… pseudorandomness of QRO "$H'(s,\cdot)$".

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_{\$} \mathcal{M}$	1 : Parse $\mathsf{sk}' = (\mathsf{sk}, s)$
2 : $s \leftarrow_{\$} \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : $\mathsf{sk}' \leftarrow (\mathsf{sk}, s)$	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : **return** $(\mathsf{pk}, \mathsf{sk}')$	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : $k \leftarrow H'(\overline{k}, H(c))$	5 : $\overline{k'} \leftarrow G_k(m')$
	6 : **return** (c, k)	6 : **if** $c' = c$ **then**
		7 : **return** $H'(\overline{k'}, H(c))$
		8 : **else return** $H''(H(c))$

Analysis of Kyber

(simplified)
Kyber

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_{\$} \mathcal{M}$	1 : Parse $\mathsf{sk}' = (\mathsf{sk}, s)$
2 : $s \leftarrow_{\$} \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : $\mathsf{sk}' \leftarrow (\mathsf{sk}, s)$	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : **return** $(\mathsf{pk}, \mathsf{sk}')$	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : $k \leftarrow H'(\overline{k}, H(c))$	5 : $\overline{k'} \leftarrow G_k(m')$
	6 : **return** (c, k)	6 : **if** $c' = c$ **then**
		7 : **return** $H'(\overline{k'}, H(c))$
		8 : **else return** $H'(s, H(c))$

IND-CCA adversary

(simplified)
Kyber modified

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_{\$} \mathcal{M}$	1 : Parse $\mathsf{sk}' = (\mathsf{sk}, s)$
2 : $s \leftarrow_{\$} \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : $\mathsf{sk}' \leftarrow (\mathsf{sk}, s)$	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : **return** $(\mathsf{pk}, \mathsf{sk}')$	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : $k \leftarrow H'(\overline{k}, H(c))$	5 : $\overline{k'} \leftarrow G_k(m')$
	6 : **return** (c, k)	6 : **if** $c' = c$ **then**
		7 : **return** $H'(\overline{k'}, H(c))$
		8 : **else return** $H'(\overline{H}(H(c)), H(c))$

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_{\$} \mathcal{M}$	1 : Parse $\mathsf{sk}' = (\mathsf{sk}, s)$
2 : $s \leftarrow_{\$} \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : $\mathsf{sk}' \leftarrow (\mathsf{sk}, s)$	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : **return** $(\mathsf{pk}, \mathsf{sk}')$	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : $k \leftarrow H'(\overline{k}, H(c))$	5 : $\overline{k'} \leftarrow G_k(m')$
	6 : **return** (c, k)	6 : **if** $c' = c$ **then**
		7 : **return** $H'(\overline{k'}, H(c))$
		8 : **else return** $H''(H(c))$

Analysis of Kyber

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_{\$} \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_{\$} \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : sk′ ← (sk, s)	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : return (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : $k \leftarrow H'(\overline{k}, H(c))$	5 : $\overline{k'} \leftarrow G_k(m')$
	6 : return (c, k)	6 : if $c' = c$ then
		7 : return $H'(\overline{k'}, H(c))$
		8 : else return $H'(s, H(c))$

(simplified) Kyber

IND-CCA adversary

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_{\$} \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_{\$} \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : sk′ ← (sk, s)	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : return (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : $k \leftarrow H'(\overline{k}, H(c))$	5 : $\overline{k'} \leftarrow G_k(m')$
	6 : return (c, k)	6 : if $c' = c$ then
		7 : return $H'(\overline{k'}, H(c))$
		8 : else return $H''(H(c))$

... generalized “One-Way To Hiding (OW2H)” lemma in the QROM [Ambainis-Hamburg-Unruh@Crypto’19].

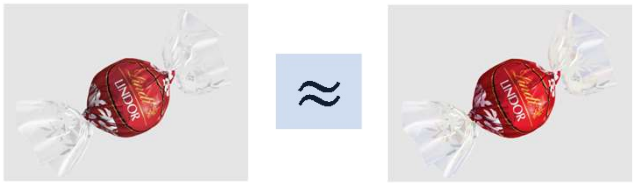
KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_{\$} \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_{\$} \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : sk′ ← (sk, s)	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : return (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : $k \leftarrow H'(\overline{k}, H(c))$	5 : $\overline{k'} \leftarrow G_k(m')$
	6 : return (c, k)	6 : if $c' = c$ then
		7 : return $H'(\overline{k'}, H(c))$
		8 : else return $H'(\overline{H}(H(c)), H(c))$

(simplified) Kyber modified

Analysis of Kyber

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_{\$} \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_{\$} \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : sk′ ← (sk, s)	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : return (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : $k \leftarrow H'(\overline{k}, H(c))$	5 : $\overline{k'} \leftarrow G_k(m')$
	6 : return (c, k)	6 : if $c' = c$ then
		7 : return $H'(\overline{k'}, H(c))$
		8 : else return $H'(s, H(c))$

(simplified) Kyber



≈

IND-CCA adversary

… pseudorandomness of QRO "$H'(s,\cdot)$".

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_{\$} \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_{\$} \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : sk′ ← (sk, s)	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : return (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : $k \leftarrow H'(\overline{k}, H(c))$	5 : $\overline{k'} \leftarrow G_k(m')$
	6 : return (c, k)	6 : if $c' = c$ then
		7 : return $H'(\overline{k'}, H(c))$
		8 : else return $H''(H(c))$

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_{\$} \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_{\$} \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : sk′ ← (sk, s)	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : return (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : $k \leftarrow H'(\overline{k}, H(c))$	5 : $\overline{k'} \leftarrow G_k(m')$
	6 : return (c, k)	6 : if $c' = c$ then
		7 : return $H'(\overline{k'}, H(c))$
		8 : else return $H'(\overline{H}(H(c)), H(c))$

(simplified) Kyber modified

… generalized "One-Way To Hiding (OW2H)" lemma in the QROM [Ambainis-Hamburg-Unruh@Crypto'19].

Analysis of Kyber

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_\$ \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_\$ \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : sk′ ← (sk, s)	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : **return** $(c, \overline{k})$	5 : **if** $c' = c$ **then**
		6 : **return** $G_k(m')$
		7 : **else return** $\overline{H}(H(c))$

$\mathsf{FO}_m^{\not\perp}$ modified

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_\$ \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_\$ \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : sk′ ← (sk, s)	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : $k \leftarrow H'(\overline{k}, H(c))$	5 : $\overline{k'} \leftarrow G_k(m')$
	6 : **return** (c, k)	6 : **if** $c' = c$ **then**
		7 : **return** $H'(\overline{k'}, H(c))$
		8 : **else return** $H'(\overline{H}(H(c)), H(c))$

Kyber (simplified) modified

Kyber (simplified)

$\mathsf{FO}_m^{\not\perp}$

≈

$\overline{k}'$

$H'(\overline{k}', H(c))$

≈

We also show that IND-CCA security of both KEMs are computationally equivalent.

IND-CCA security of $\mathsf{FO}_m^{\not\perp}$ KEMs in the QROM ⇒ ?

We also show that IND-CCA security of both KEMs are computationally equivalent.

Analysis of Kyber

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_{\$} \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_{\$} \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : sk′ ← (sk, s)	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : **return** $(c, \overline{k})$	5 : **if** $c' = c$ **then**
		6 : **return** $G_k(m')$
		7 : **else return** $\overline{H}(H(c))$

$\mathsf{FO}_m^{\not\perp}$ modified

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_{\$} \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_{\$} \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$
3 : sk′ ← (sk, s)	3 : $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m; r)$	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m)$	4 : $c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m'; r')$
	5 : $k \leftarrow H'(\overline{k}, H(c))$	5 : $\overline{k'} \leftarrow G_k(m')$
	6 : **return** (c, k)	6 : **if** $c' = c$ **then**
		7 : **return** $H'(\overline{k'}, H(c))$
		8 : **else return** $H'(\overline{H}(H(c)), H(c))$

Kyber (simplified) modified

Kyber (simplified)

$\mathsf{FO}_m^{\not\perp}$

$\approx$

$\overline{k}'$

$H'(\overline{k}', H(c))$

$\approx$

We also show that IND-CCA security of both KEMs are computationally equivalent.

IND-CCA security of $\mathsf{FO}_m^{\not\perp}$ KEMs in the QROM $\Rightarrow$ IND-CCA security of Kyber in the QROM

We also show that IND-CCA security of both KEMs are computationally equivalent.

Analysis of Kyber

IND-CCA secure

KEM

“key ← hash(m, c)”

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_{\$} \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_{\$} \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow$ Dec(sk, c)
3 : sk′ ← (sk, s)	3 : $c \leftarrow$ Enc(pk, m; r)	3 : $r' \leftarrow G_r(m')$
4 : return (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m, c)$	4 : $c' \leftarrow$ Enc(pk, m′; r′)
	5 : return $(c, \overline{k})$	5 : if $c' = c$ then
		6 : return $G_k(m', c)$
		7 : else return $G_k(s, c)$

FO$^{\not\perp}$ [Hofheinz-Hövelmanns-Kiltz@TCC'17]

Quantum Random Oracles (QROs)

Hash Functions H_i

“key ← hash(hash(m), hash(c))”

[Maram-Xagawa@PKC’23]

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_{\$} \mathcal{M}$	1 : Parse sk′ = (sk, pk, h, s)
2 : $s \leftarrow_{\$} \mathcal{M}$	2 : $m \leftarrow H(m)$	2 : $m' \leftarrow$ Dec(sk, c)
3 : pk′ ← (pk, H(pk))	3 : $h \leftarrow H$(pk)	3 : $(\overline{k'}, r') \leftarrow G_{kr}(m', h)$
4 : sk′ ← (sk, pk′, s)	4 : $(\overline{k}, r) \leftarrow G_{kr}(m, h)$	4 : $c' \leftarrow$ Enc(pk, m′; r′)
5 : return (pk, sk′)	5 : $c \leftarrow$ Enc(pk, m; r)	5 : if $c' = c$ then
	6 : $k \leftarrow H'(\overline{k}, H(c))$	6 : return $H'(\overline{k'}, H(c))$
	7 : return (c, k)	7 : else return $H'(s, H(c))$

{OW/IND}-CPA secure

PKE (Base)

We provide a concrete proof of IND-CCA security for Kyber in the QROM.

$\mathrm{Adv}^{\mathrm{IND-CCA}}_{\mathrm{Kyber}} \leq \mathrm{Adv}^{\mathrm{IND-CCA}}_{\mathrm{FO}^{\not\perp}} + \mathrm{Adv}^{\mathrm{CR}}_{H}$

Kyber’s FO variant

Analysis of Kyber

$$\mathrm{Adv}^{\mathrm{IND-C}}_{\mathrm{Kyber}} \leq \mathrm{Adv}^{\mathrm{IND-CCA}}_{\mathrm{FO}^{\not\perp}} + \mathrm{Adv}^{\mathrm{CR}}_{H}$$

Analysis of Kyber

$$\mathrm{Adv}^{\mathrm{IND-CCA}}_{\mathrm{Kyber}} \leq \mathrm{Adv}^{\mathrm{IND-CCA}}_{\mathsf{FO}^{\not\perp}_m} + \mathrm{Adv}^{\mathrm{CR}}_{H}$$

Analysis of Kyber

$$\mathrm{Adv}^{\mathrm{IND-CCA}}_{\mathrm{Kyber}} \leq \mathrm{Adv}^{\mathrm{IND-CCA}}_{\mathsf{FO}^{\not\perp}_m} + \mathrm{Adv}^{\mathrm{CR}}_{H}$$

Shown by [Bindel-Hamburg-Hövelmanns-Hülsing-Persichetti@TCC'19]:

$$\mathrm{Adv}^{\mathrm{IND-C}}_{\mathsf{FO}^{\not\perp}_m} \approx \mathrm{Adv}^{\mathrm{IND-CCA}}_{\mathsf{FO}^{\not\perp}}$$

Analysis of Kyber

$$\mathrm{Adv}^{\mathrm{IND-CCA}}_{\mathrm{Kyber}} \leq \mathrm{Adv}^{\mathrm{IND-CCA}}_{\mathrm{FO}^{\not\perp}_m} + \mathrm{Adv}^{\mathrm{CR}}_{H}$$

Analysis of Kyber

$$\mathrm{Adv}^{\mathrm{IND-CCA}}_{\mathrm{Kyber}} \leq \mathrm{Adv}^{\mathrm{IND-C}}_{\mathrm{FO}^{\not\perp}_m} + \mathrm{Adv}^{\mathrm{CR}}_{H}$$

Collision-resistance of nested hash "$H(c)$", when modelled as a QRO.

Analysis of Kyber

$$\mathrm{Adv}^{\mathrm{IND-}}_{\mathrm{Kyber}} \leq \mathrm{Adv}^{\mathrm{IND-C}}_{\mathrm{FO}^{\not\perp}_m} + \mathrm{Adv}^{\mathrm{CR}}_{H}$$

Collision-resistance of nested hash "$H(c)$", when modelled as a QRO.

$\mathrm{FO}^{\not\perp}_m$ modified

$\approx$

$\mathrm{FO}^{\not\perp}_m$

IND-CCA security of both KEMs computationally equivalent.

Don’t Roll Your Own FO:

Challenges in Proving Post-Quantum CCA Security of Real-World KEMs

Fujisaki-**O**kamoto transform.

- CRYSTALS-Kyber
- Streamlined NTRU Prime

Varun Maram

Cybersecurity Group

SandboxAQ

In the **Q**uantum **R**andom **O**racle **M**odel (QROM).

Based on joint works with Paul Grubbs, Kenny Paterson, and Keita Xagawa.

Fujisaki-Okamoto transform.

Don’t Roll Your Own FO:

Challenges in Proving Post-Quantum CCA Security of Real-World KEMs

- CRYSTALS-Kyber
- Streamlined NTRU Prime

Varun Maram

Cybersecurity Group

SandboxAQ

In the Quantum Random Oracle Model (QROM).

Based on joint works with Paul Grubbs, Kenny Paterson, and Keita Xagawa.

FO Variants

IND-CCA secure

KEM

"key ← hash(m, c)"

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_s \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_s \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow$ Dec(sk, c)
3 : sk′ ← (sk, s)	3 : $c \leftarrow$ Enc(pk, m; r)	3 : $r' \leftarrow G_r(m')$
4 : return (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m, c)$	4 : $c' \leftarrow$ Enc(pk, m′; r′)
	5 : return $(c, \overline{k})$	5 : if $c' = c$ then
		6 : return $G_k(m', c)$
		7 : else return $G_k(s, c)$

FO$^{\not\perp}$ [Hofheinz-Hövelmanns-Kiltz@TCC'17]

Quantum Random Oracles (QROs)

Hash Functions H_i

"key ← hash(hash(m), hash(c))"

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_s \mathcal{M}$	1 : Parse sk′ = (sk, pk, h, s)
2 : $s \leftarrow_s \mathcal{M}$	2 : $m \leftarrow H(m)$	2 : $m' \leftarrow$ Dec(sk, c)
3 : pk′ ← (pk, H(pk))	3 : $h \leftarrow H$(pk)	3 : $(\overline{k'}, r') \leftarrow G_{kr}(m', h)$
4 : sk′ ← (sk, pk′, s)	4 : $(\overline{k}, r) \leftarrow G_{kr}(m, h)$	4 : $c' \leftarrow$ Enc(pk, m′; r′)
5 : return (pk, sk′)	5 : $c \leftarrow$ Enc(pk, m; r)	5 : if $c' = c$ then
	6 : $k \leftarrow H'(\overline{k}, H(c))$	6 : return $H'(\overline{k'}, H(c))$
	7 : return (c, k)	7 : else return $H'(s, H(c))$

Kyber's FO variant

{OW/IND}-CPA secure

PKE (Base)

FO Variants

IND-CCA secure

KEM

"key ← hash(m, c)"

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_s \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_s \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow$ Dec(sk, c)
3 : sk′ ← (sk, s)	3 : $c \leftarrow$ Enc(pk, $m; r$)	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m, c)$	4 : $c' \leftarrow$ Enc(pk, $m'; r'$)
	5 : **return** $(c, \overline{k})$	5 : **if** $c' = c$ **then**
		6 : **return** $G_k(m', c)$
		7 : **else return** $G_k(s, c)$

FO$^{\not\perp}$ [Hofheinz-Hövelmanns-Kiltz@TCC'17]

Quantum Random Oracles (QROs)

Hash Functions H_i

"key ← hash(hash(m), c, hash(hash(m)))"

{OW/IND}-CPA secure

PKE (Base)

$\overline{\text{Gen}}(1^\kappa)$	$\overline{\text{Enc}}(ek)$	$\overline{\text{Dec}}(\overline{dk}, (c_0, c_1))$, where $\overline{dk} = (dk, ek, s)$
$(ek, dk) \leftarrow \text{Gen}(1^\kappa)$	$\mu \leftarrow \mathcal{M}$	$\mu' := \text{Dec}(dk, c_0)$
$s \leftarrow \{0,1\}^\ell$	$c_0 := \text{Enc}(ek, \mu)$	**if** $\mu' = \perp$, **then return** $K := \text{H}_0(\text{H}_3(s), c_0, c_1)$
$\overline{dk} := (dk, ek, s)$	$c_1 := \text{H}_2(\text{H}_3(\mu), \text{H}_4(ek))$	$c_0' := \text{Enc}(ek, \mu')$
return $(ek, \overline{dk})$	$K := \text{H}_1(\text{H}_3(\mu), c_0, c_1)$	$c_1' := \text{H}_2(\text{H}_3(\mu'), \text{H}_4(ek))$
	return $((c_0, c_1), K)$	**if** $(c_0, c_1) \neq (c_0', c_1')$, **then return** $K := \text{H}_0(\text{H}_3(s), c_0, c_1)$
		else return $K := \text{H}_1(\text{H}_3(\mu'), c_0, c_1)$

Streamlined NTRU Prime's FO variant

Streamlined NTRU Prime (SNTRUp)

NTRU Prime

NTRU Prime is a third-round candidate in NIST's Post-Quantum Cryptography Standardization Project. The third-round submission package specifies twelve KEMs:

- Streamlined NTRU Prime: `sntrup653`, `sntrup761`, `sntrup857`, `sntrup953`, `sntrup1013`, `sntrup1277`.
- NTRU LPRime: `ntrulpr653`, `ntrulpr761`, `ntrulpr857`, `ntrulpr953`, `ntrulpr1013`, `ntrulpr1277`.

Streamlined NTRU Prime (SNTRUp)

NTRU Prime

Intro | Warnings | Papers | Security | Softw
Speed | NISTPQC | FAQ

NTRU Prime is a third-round candidate in NIST's Post-Quantur Cryptography Standardization Project. The third-round submiss specifies twelve KEMs:

- Streamlined NTRU Prime: `sntrup653`, `sntrup761`, `sntr` `sntrup953`, `sntrup1013`, `sntrup1277`.
- NTRU LPRime: `ntrulpr653`, `ntrulpr761`, `ntrulpr857`, `ntrulpr1013`, `ntrulpr1277`.

OpenSSH 9.0/9.0p1 (2022-04-08)

OpenSSH 9.0 was released on 2022-04-08. It is available from the mirrors listed at https://www.openssh.com/.
OpenSSH is a 100% complete SSH protocol 2.0 implementation and includes sftp client and server support.

Once again, we would like continued support of the code or patches, reported project. More information https://www.openssh.com/d

Changes since OpenSSH 8.9
==========

This release is focused o

Potentially-incompatible

This release switches scp to using the SFTP protoco

Legacy scp/rcp performs w "scp host:* .") through t requiring double quoting included on scp(1) comman as shell commands on the

This creates one area of potential incompatibility: scp(1) when using the SFTP protocol no longer requires this finicky and brittle quoting, and attempts to use it may cause transfers to fail. We consider the removal of the need for double-quoting shell characters in file names to be a benefit and do not intend to introduce bug-compatibility for legacy scp/rcp in scp(1) when using the SFTP protocol.

Another area of potential incompatibility relates to the use of remote paths relative to other user's home directories, for example - "scp host:~user/file /tmp". The SFTP protocol has no native way to expand a ~user path. However, sftp-server(8) in OpenSSH 8.7 and later support a protocol extension "expand-path@openssh.com" to support this.

In case of incompatibility, the scp(1) client may be instructed to use the legacy scp/rcp using the -O flag.

New features
---

* ssh(1), sshd(8): use the hybrid Streamlined NTRU Prime + x25519 key exchange method by default ("sntrup761x25519-sha512@openssh.com"). The NTRU algorithm is believed to resist attacks enabled by future quantum computers and is paired with the X25519 ECDH key exchange (the previous default) as a backstop against any weaknesses in NTRU Prime that may be discovered in the future. The combination ensures that the hybrid exchange offers at least as good security as the status quo.

FO Variants

IND-CCA secure

KEM

"key ← hash(m, c)"

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_s \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_s \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow$ Dec(sk, c)
3 : sk′ ← (sk, s)	3 : $c \leftarrow$ Enc(pk, $m; r$)	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m, c)$	4 : $c' \leftarrow$ Enc(pk, $m'; r'$)
	5 : **return** $(c, \overline{k})$	5 : **if** $c' = c$ **then**
		6 : **return** $G_k(m', c)$
		7 : **else return** $G_k(s, c)$

FO$^{\not\perp}$ [Hofheinz-Hövelmanns-Kiltz@TCC'17]

Quantum Random Oracles (QROs)

Hash Functions H_i

"key ← hash(hash(m), c, hash(hash(m)))"

{OW/IND}-CPA secure

PKE (Base)

$\overline{\text{Gen}}(1^\kappa)$	$\overline{\text{Enc}}(ek)$	$\overline{\text{Dec}}(\overline{dk}, (c_0, c_1))$, where $\overline{dk} = (dk, ek, s)$
$(ek, dk) \leftarrow \text{Gen}(1^\kappa)$	$\mu \leftarrow \mathcal{M}$	$\mu' := \text{Dec}(dk, c_0)$
$s \leftarrow \{0,1\}^\ell$	$c_0 := \text{Enc}(ek, \mu)$	**if** $\mu' = \perp$, **then return** $K := \text{H}_0(\text{H}_3(s), c_0, c_1)$
$\overline{dk} := (dk, ek, s)$	$c_1 := \text{H}_2(\text{H}_3(\mu), \text{H}_4(ek))$	$c_0' := \text{Enc}(ek, \mu')$
return $(ek, \overline{dk})$	$K := \text{H}_1(\text{H}_3(\mu), c_0, c_1)$	$c_1' := \text{H}_2(\text{H}_3(\mu'), \text{H}_4(ek))$
	return $((c_0, c_1), K)$	**if** $(c_0, c_1) \neq (c_0', c_1')$, **then return** $K := \text{H}_0(\text{H}_3(s), c_0, c_1)$
		else return $K := \text{H}_1(\text{H}_3(\mu'), c_0, c_1)$

Streamlined NTRU Prime's FO variant

Analysis of SNTRUp

IND-CCA secure

KEM

“key ← hash(m, c)”

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_s \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_s \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow$ Dec(sk, c)
3 : sk′ ← (sk, s)	3 : $c \leftarrow$ Enc(pk, $m; r$)	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m, c)$	4 : $c' \leftarrow$ Enc(pk, $m'; r'$)
	5 : **return** $(c, \overline{k})$	5 : **if** $c' = c$ **then**
		6 : **return** $G_k(m', c)$
		7 : **else return** $G_k(s, c)$

FO$^{\not}$ [Hofheinz-Hövelmanns-Kiltz@TCC'17]

Quantum Random Oracles (QROs)

Hash Functions H_i

“key ← hash(hash(m), c, hash(hash(m)))”

$\overline{\text{Gen}}(1^\kappa)$	$\overline{\text{Enc}}(ek)$	$\overline{\text{Dec}}(\overline{dk}, (c_0, c_1))$, where $\overline{dk} = (dk, ek, s)$
$(ek, dk) \leftarrow \text{Gen}(1^\kappa)$	$\mu \leftarrow \mathcal{M}$	$\mu' := \text{Dec}(dk, c_0)$
$s \leftarrow \{0,1\}^\ell$	$c_0 := \text{Enc}(ek, \mu)$	**if** $\mu' = \perp$, **then return** $K := \text{H}_0(\text{H}_3(s), c_0, c_1)$
$\overline{dk} := (dk, ek, s)$	$c_1 := \text{H}_2(\text{H}_3(\mu), \text{H}_4(ek))$	$c_0' := \text{Enc}(ek, \mu')$
return $(ek, \overline{dk})$	$K := \text{H}_1(\text{H}_3(\mu), c_0, c_1)$	$c_1' := \text{H}_2(\text{H}_3(\mu'), \text{H}_4(ek))$
	return $((c_0, c_1), K)$	**if** $(c_0, c_1) \neq (c_0', c_1')$, **then return** $K := \text{H}_0(\text{H}_3(s), c_0, c_1)$
		else return $K := \text{H}_1(\text{H}_3(\mu'), c_0, c_1)$

{OW/IND}-CPA secure

PKE (Base)

Streamlined NTRU Prime’s FO variant

Analysis of SNTRUp

IND-CCA secure

KEM

“key ← hash(m, c)”

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_s \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_s \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow$ Dec(sk, c)
3 : sk′ ← (sk, s)	3 : $c \leftarrow$ Enc(pk, $m; r$)	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m, c)$	4 : $c' \leftarrow$ Enc(pk, $m'; r'$)
	5 : **return** $(c, \overline{k})$	5 : **if** $c' = c$ **then**
		6 : **return** $G_k(m', c)$
		7 : **else return** $G_k(s, c)$

FO$^{\not\perp}$ [Hofheinz-Hövelmanns-Kiltz@TCC'17]

Quantum Random Oracles (QROs)

Hash Functions H_i

“key ← hash(hash(m), c_0, c_1)”

$\overline{\text{Gen}}(1^\kappa)$	$\overline{\text{Enc}}(ek)$	$\overline{\text{Dec}}(\overline{dk}, (c_0, c_1))$, where $\overline{dk} = (dk, ek, s)$
$(ek, dk) \leftarrow \text{Gen}(1^\kappa)$	$\mu \leftarrow \mathcal{M}$	$\mu' := \text{Dec}(dk, c_0)$
$s \leftarrow \{0,1\}^\ell$	$c_0 := \text{Enc}(ek, \mu)$	**if** $\mu' = \perp$, **then return** $K := \text{H}_0(\text{H}_3(s), c_0, c_1)$
$\overline{dk} := (dk, ek, s)$	$c_1 := \text{H}_2(\text{H}_3(\mu), \text{H}_4(ek))$	$c_0' := \text{Enc}(ek, \mu')$
return $(ek, \overline{dk})$	$K := \text{H}_1(\text{H}_3(\mu), c_0, c_1)$	$c_1' := \text{H}_2(\text{H}_3(\mu'), \text{H}_4(ek))$
	return $((c_0, c_1), K)$	**if** $(c_0, c_1) \neq (c_0', c_1')$, **then return** $K := \text{H}_0(\text{H}_3(s), c_0, c_1)$
		else return $K := \text{H}_1(\text{H}_3(\mu'), c_0, c_1)$

Streamlined NTRU Prime’s FO variant

{OW/IND}-CPA secure

PKE (Base)

Analysis of SNTRUp

IND-CCA secure

KEM

"key ← hash(m, c)"

KGen′	Encap(pk)	Decap(sk′, c)
1 : (pk, sk) ← KGen	1 : $m \leftarrow_s \mathcal{M}$	1 : Parse sk′ = (sk, s)
2 : $s \leftarrow_s \mathcal{M}$	2 : $r \leftarrow G_r(m)$	2 : $m' \leftarrow$ Dec(sk, c)
3 : sk′ ← (sk, s)	3 : $c \leftarrow$ Enc(pk, $m; r$)	3 : $r' \leftarrow G_r(m')$
4 : **return** (pk, sk′)	4 : $\overline{k} \leftarrow G_k(m, c)$	4 : $c' \leftarrow$ Enc(pk, $m'; r'$)
	5 : **return** $(c, \overline{k})$	5 : **if** $c' = c$ **then**
		6 : **return** $G_k(m', c)$
		7 : **else return** $G_k(s, c)$

FO$^{\not\perp}$ [Hofheinz-Hövelmanns-Kiltz@TCC'17]

Quantum Random Oracles (QROs)

Hash Functions H_i

"key ← hash(hash(m), c_0, c_1)"

$\overline{\text{Gen}}(1^\kappa)$	$\overline{\text{Enc}}(ek)$	$\overline{\text{Dec}}(\overline{dk}, (c_0, c_1))$, where $\overline{dk} = (dk, ek, s)$
$(ek, dk) \leftarrow \text{Gen}(1^\kappa)$	$\mu \leftarrow \mathcal{M}$	$\mu' := \text{Dec}(dk, c_0)$
$s \leftarrow \{0,1\}^\ell$	$c_0 := \text{Enc}(ek, \mu)$	**if** $\mu' = \perp$, **then return** $K := \text{H}_0(\text{H}_3(s), c_0, c_1)$
$\overline{dk} := (dk, ek, s)$	$c_1 := \text{H}_2(\text{H}_3(\mu), \text{H}_4(ek))$	$c_0' := \text{Enc}(ek, \mu')$
return $(ek, \overline{dk})$	$K := \text{H}_1(\text{H}_3(\mu), c_0, c_1)$	$c_1' := \text{H}_2(\text{H}_3(\mu'), \text{H}_4(ek))$
	return $((c_0, c_1), K)$	**if** $(c_0, c_1) \neq (c_0', c_1')$, **then return** $K := \text{H}_0(\text{H}_3(s), c_0, c_1)$
		else return $K := \text{H}_1(\text{H}_3(\mu'), c_0, c_1)$

{OW/IND}-CPA secure

PKE (Base)

"c_1 ← hash(hash(m))"

Streamlined NTRU Prime's FO variant

Analysis of SNTRUp

"key ← hash(m, c_0, c_1)"

"c_1 ← hash(m)"

IND-CCA secure

KEM

$\overline{\text{Gen}}(1^\kappa)$
1 : $(ek, dk) \leftarrow \text{Gen}(1^\kappa)$
2 : $s \leftarrow \mathcal{M}$
3 : $\overline{dk} := (dk, ek, s)$
4 : return $(ek, \overline{dk})$

$\overline{\text{Enc}}(ek)$
1 : $\mu \leftarrow \mathcal{D}_\mathcal{M}$
2 : $c_0 := \text{Enc}(ek, \mu)$
3 : $c_1 := \text{F}(\mu[, ek])$
4 : $K := \text{H}(\mu, c_0, c_1)$
5 : return $((c_0, c_1), K)$

$\overline{\text{Dec}}(\overline{dk}, (c_0, c_1))$, where $\overline{dk} = (dk, ek, s)$
1 : $\mu' \leftarrow \text{Dec}(dk, c_0)$
2 : if $\mu' = \perp$ or $c_0 \neq \text{Enc}(ek, \mu')$ or $c_1 \neq \text{F}(\mu'[, ek])$
3 : then return $K := \text{H}(s, c_0, c_1)$
4 : else return $K := \text{H}(\mu', c_0, c_1)$

$\text{HU}^{\not\perp}$ [Xagawa@Eurocrypt'22]

Quantum Random Oracles (QROs)

Hash Functions H_i

"key ← hash(hash(m), c_0, c_1)"

$\overline{\text{Gen}}(1^\kappa)$
$(ek, dk) \leftarrow \text{Gen}(1^\kappa)$
$s \leftarrow \{0,1\}^\ell$
$\overline{dk} := (dk, ek, s)$
return $(ek, \overline{dk})$

$\overline{\text{Enc}}(ek)$
$\mu \leftarrow \mathcal{M}$
$c_0 := \text{Enc}(ek, \mu)$
$c_1 := \text{H}_2(\text{H}_3(\mu), \text{H}_4(ek))$
$K := \text{H}_1(\text{H}_3(\mu), c_0, c_1)$
return $((c_0, c_1), K)$

$\overline{\text{Dec}}(\overline{dk}, (c_0, c_1))$, where $\overline{dk} = (dk, ek, s)$
$\mu' := \text{Dec}(dk, c_0)$
if $\mu' = \perp$, then return $K := \text{H}_0(\text{H}_3(s), c_0, c_1)$
$c_0' := \text{Enc}(ek, \mu')$
$c_1' := \text{H}_2(\text{H}_3(\mu'), \text{H}_4(ek))$
if $(c_0, c_1) \neq (c_0', c_1')$, then return $K := \text{H}_0(\text{H}_3(s), c_0, c_1)$
else return $K := \text{H}_1(\text{H}_3(\mu'), c_0, c_1)$

"c_1 ← hash(hash(m))"

Streamlined NTRU Prime's FO variant

PKE (Base)

{OW/IND}-CPA secure

Analysis of SNTRUp

"$c_1 \leftarrow \mathrm{hash}(m)$"

IND-CCA secure

KEM

$\overline{\mathsf{Gen}}(1^\kappa)$
1 : $(ek, dk) \leftarrow \mathsf{Gen}(1^\kappa)$
2 : $s \leftarrow \mathcal{M}$
3 : $\overline{dk} := (dk, ek, s)$
4 : return $(ek, \overline{dk})$

$\overline{\mathsf{Enc}}(ek)$
1 : $\mu \leftarrow \mathcal{D}_\mathcal{M}$
2 : $c_0 := \mathsf{Enc}(ek, \mu)$
3 : $c_1 := \mathsf{F}(\mu[, ek])$
4 : $K := \mathsf{H}(\mu, c_0, c_1)$
5 : return $((c_0, c_1), K)$

$\overline{\mathsf{Dec}}(\overline{dk}, (c_0, c_1))$, where $\overline{dk} = (dk, ek, s)$
1 : $\mu' \leftarrow \mathsf{Dec}(dk, c_0)$
2 : if $\mu' = \perp$ or $c_0 \neq \mathsf{Enc}(ek, \mu')$ or $c_1 \neq \mathsf{F}(\mu'[, ek])$
3 : then return $K := \mathsf{H}(s, c_0, c_1)$
4 : else return $K := \mathsf{H}(\mu', c_0, c_1)$

$\mathsf{HU}^{\not\perp}$ [Xagawa@Eurocrypt'22]

Quantum Random Oracles (QROs)

Hash Functions H_i

+

{OW/IND}-CPA secure

PKE (Base)

$\overline{\mathsf{Gen}}(1^\kappa)$
$(ek, dk) \leftarrow \mathsf{Gen}(1^\kappa)$
$s \leftarrow \{0,1\}^\ell$
$\overline{dk} := (dk, ek, s)$
return $(ek, \overline{dk})$

$\overline{\mathsf{Enc}}(ek)$
$\mu \leftarrow \mathcal{M}$
$c_0 := \mathsf{Enc}(ek, \mu)$
$c_1 := \mathsf{H}_2(\mathsf{H}_3(\mu), \mathsf{H}_4(ek))$
$K := \mathsf{H}_1(\mathsf{H}_3(\mu), c_0, c_1)$
return $((c_0, c_1), K)$

$\overline{\mathsf{Dec}}(\overline{dk}, (c_0, c_1))$, where $\overline{dk} = (dk, ek, s)$
$\mu' := \mathsf{Dec}(dk, c_0)$
if $\mu' = \perp$, then return $K := \mathsf{H}_0(\mathsf{H}_3(s), c_0, c_1)$
$c_0' := \mathsf{Enc}(ek, \mu')$
$c_1' := \mathsf{H}_2(\mathsf{H}_3(\mu'), \mathsf{H}_4(ek))$
if $(c_0, c_1) \neq (c_0', c_1')$, then return $K := \mathsf{H}_0(\mathsf{H}_3(s), c_0, c_1)$
else return $K := \mathsf{H}_1(\mathsf{H}_3(\mu'), c_0, c_1)$

"$c_1 \leftarrow \mathrm{hash}(\mathrm{hash}(m))$"

Streamlined NTRU Prime's FO variant

Analysis of SNTRUp

"$c_1 \leftarrow \mathrm{hash}(m)$"

IND-CCA secure

KEM

Quantum Random Oracles (QROs)

Hash Functions H_i

{OW/IND}-CPA secure

PKE (Base)

$\overline{\mathsf{Gen}}(1^\kappa)$	$\overline{\mathsf{Enc}}(ek)$	$\overline{\mathsf{Dec}}(\overline{dk}, (c_0, c_1))$, where $\overline{dk} = (dk, ek, s)$
1 : $(ek, dk) \leftarrow \mathsf{Gen}(1^\kappa)$	1 : $\mu \leftarrow \mathcal{D}_\mathcal{M}$	1 : $\mu' \leftarrow \mathsf{Dec}(dk, c_0)$
2 : $s \leftarrow \mathcal{M}$	2 : $c_0 := \mathsf{Enc}(ek, \mu)$	2 : **if** $\mu' = \perp$ or $c_0 \neq \mathsf{Enc}(ek, \mu')$ or $c_1 \neq \mathsf{F}(\mu'[, ek])$
3 : $\overline{dk} := (dk, ek, s)$	3 : $c_1 := \mathsf{F}(\mu[, ek])$	3 : **then return** $K := \mathsf{H}(s, c_0, c_1)$
4 : **return** $(ek, \overline{dk})$	4 : $K := \mathsf{H}(\mu, c_0, c_1)$	4 : **else return** $K := \mathsf{H}(\mu', c_0, c_1)$
	5 : **return** $((c_0, c_1), K)$	

"$c_1 \leftarrow \mathrm{hash}(m)$" and "$c_1 \leftarrow \mathrm{hash}(\mathrm{hash}(m))$" are quantum **indifferentiable**. [Zhandry@Crypto'19]

$\mathrm{HU}^{\not\perp}$ [Xagawa@Eurocrypt'22]

$\overline{\mathsf{Gen}}(1^\kappa)$	$\overline{\mathsf{Enc}}(ek)$	$\overline{\mathsf{Dec}}(\overline{dk}, (c_0, c_1))$, where $\overline{dk} = (dk, ek, s)$
$(ek, dk) \leftarrow \mathsf{Gen}(1^\kappa)$	$\mu \leftarrow \mathcal{M}$	$\mu' := \mathsf{Dec}(dk, c_0)$
$s \leftarrow \{0,1\}^\ell$	$c_0 := \mathsf{Enc}(ek, \mu)$	**if** $\mu' = \perp$, **then return** $K := \mathsf{H}_0(\mathsf{H}_3(s), c_0, c_1)$
$\overline{dk} := (dk, ek, s)$	$c_1 := \mathsf{H}_2(\mathsf{H}_3(\mu), \mathsf{H}_4(ek))$	$c_0' := \mathsf{Enc}(ek, \mu')$
return $(ek, \overline{dk})$	$K := \mathsf{H}_1(\mathsf{H}_3(\mu), c_0, c_1)$	$c_1' := \mathsf{H}_2(\mathsf{H}_3(\mu'), \mathsf{H}_4(ek))$
	return $((c_0, c_1), K)$	**if** $(c_0, c_1) \neq (c_0', c_1')$, **then return** $K := \mathsf{H}_0(\mathsf{H}_3(s), c_0, c_1)$
		else return $K := \mathsf{H}_1(\mathsf{H}_3(\mu'), c_0, c_1)$

"$c_1 \leftarrow \mathrm{hash}(\mathrm{hash}(m))$"

Streamlined NTRU Prime's FO variant

Analysis of SNTRUp

"key ← hash(m, c_0, c_1)"

IND-CCA secure

KEM

Quantum Random Oracles (QROs)

Hash Functions H_i

{OW/IND}-CPA secure

PKE (Base)

$\overline{\mathsf{Gen}}(1^\kappa)$	$\overline{\mathsf{Enc}}(ek)$	$\overline{\mathsf{Dec}}(\overline{dk}, (c_0, c_1))$, where $\overline{dk} = (dk, ek, s)$
1 : $(ek, dk) \leftarrow \mathsf{Gen}(1^\kappa)$	1 : $\mu \leftarrow \mathcal{D}_\mathcal{M}$	1 : $\mu' \leftarrow \mathsf{Dec}(dk, c_0)$
2 : $s \leftarrow \mathcal{M}$	2 : $c_0 := \mathsf{Enc}(ek, \mu)$	2 : **if** $\mu' = \perp$ or $c_0 \neq \mathsf{Enc}(ek, \mu')$ or $c_1 \neq \mathsf{F}(\mu'[, ek])$
3 : $\overline{dk} := (dk, ek, s)$	3 : $c_1 := \mathsf{F}(\mu[, ek])$	3 : **then return** $K := \mathsf{H}(s, c_0, c_1)$
4 : **return** $(ek, \overline{dk})$	4 : $K := \mathsf{H}(\mu, c_0, c_1)$	4 : **else return** $K := \mathsf{H}(\mu', c_0, c_1)$
	5 : **return** $((c_0, c_1), K)$	

"key ← hash(m, c_0, c_1)" and "key ← hash(hash(m), c_0, c_1)" are quantum **indifferentiable**. [Zhandry@Crypto'19]

$\mathsf{HU}^{\not\perp}$ [Xagawa@Eurocrypt'22]

$\overline{\mathsf{Gen}}(1^\kappa)$	$\overline{\mathsf{Enc}}(ek)$	$\overline{\mathsf{Dec}}(\overline{dk}, (c_0, c_1))$, where $\overline{dk} = (dk, ek, s)$
$(ek, dk) \leftarrow \mathsf{Gen}(1^\kappa)$	$\mu \leftarrow \mathcal{M}$	$\mu' := \mathsf{Dec}(dk, c_0)$
$s \leftarrow \{0,1\}^\ell$	$c_0 := \mathsf{Enc}(ek, \mu)$	**if** $\mu' = \perp$, **then return** $K := \mathsf{H}_0(\mathsf{H}_3(s), c_0, c_1)$
$\overline{dk} := (dk, ek, s)$	$c_1 := \mathsf{H}_2(\mathsf{H}_3(\mu), \mathsf{H}_4(ek))$	$c_0' := \mathsf{Enc}(ek, \mu')$
return $(ek, \overline{dk})$	$K := \mathsf{H}_1(\mathsf{H}_3(\mu), c_0, c_1)$	$c_1' := \mathsf{H}_2(\mathsf{H}_3(\mu'), \mathsf{H}_4(ek))$
	return $((c_0, c_1), K)$	**if** $(c_0, c_1) \neq (c_0', c_1')$, **then return** $K := \mathsf{H}_0(\mathsf{H}_3(s), c_0, c_1)$
		else return $K := \mathsf{H}_1(\mathsf{H}_3(\mu'), c_0, c_1)$

"key ← hash(hash(m), c_0, c_1)"

Streamlined NTRU Prime's FO variant

D. J. Bernstein Oct 8, 2021, 4:55:56 PM

to pqc-...@list.nist.gov

To say a bit more about the indifferentiability perspective: The point of the results in, e.g.,

https://cs.nyu.edu/~dodis/ps/merkle.pdf Lemma 3.2
https://hal.inria.fr/file/index/docid/765883/filename/main.pdf

is that a ROM proof of any protocol using H(x,y) for fixed-length x,y implies a ROM proof of the same protocol using H_2(H_1(x),y); and

https://eprint.iacr.org/2018/276 Theorem 4

says something analogous for QROM. The results are tight when H_1 has reasonably long output. (If H_1 is modeled as collision-free then H_2(H_1(x),y) is a perfect simulation of H(x,y), although formalizing this is harder than doing ROM proofs.)

Quantum Random Oracles (QROs)

Hash Functions H_i

PKE (Base)

{OW/IND}-CPA secure

SNTRUp

"key ← hash(m, c_0, c_1)"

	$\overline{\text{Enc}}(ek)$	$\overline{\text{Dec}}(\overline{dk}, (c_0, c_1))$, where $\overline{dk} = (dk, ek, s)$
$^{\kappa})$	1 : $\mu \leftarrow \mathcal{D}_{\mathcal{M}}$	1 : $\mu' \leftarrow \text{Dec}(dk, c_0)$
	2 : $c_0 := \text{Enc}(ek, \mu)$	2 : **if** $\mu' = \perp$ or $c_0 \neq \text{Enc}(ek, \mu')$ or $c_1 \neq \text{F}(\mu'[, ek])$
	3 : $c_1 := \text{F}(\mu[, ek])$	3 : **then return** $K := \text{H}(s, c_0, c_1)$
	4 : $K := \text{H}(\mu, c_0, c_1)$	4 : **else return** $K := \text{H}(\mu', c_0, c_1)$
	5 : **return** $((c_0, c_1), K)$	

"key ← hash(m, c_0, c_1)" and "key ← hash(hash(m), c_0, c_1)" are quantum **indifferentiable**. [Zhandry@Crypto'19]

HU$\not{\perp}$ [Xagawa@Eurocrypt'22]

$\overline{\text{Gen}}(1^{\kappa})$	$\overline{\text{Enc}}(ek)$	$\overline{\text{Dec}}(\overline{dk}, (c_0, c_1))$, where $\overline{dk} = (dk, ek, s)$
$(ek, dk) \leftarrow \text{Gen}(1^{\kappa})$	$\mu \leftarrow \mathcal{M}$	$\mu' := \text{Dec}(dk, c_0)$
$s \leftarrow \{0,1\}^{\ell}$	$c_0 := \text{Enc}(ek, \mu)$	**if** $\mu' = \perp$, **then return** $K := \text{H}_0(\text{H}_3(s), c_0, c_1)$
$\overline{dk} := (dk, ek, s)$	$c_1 := \text{H}_2(\text{H}_3(\mu), \text{H}_4(ek))$	$c_0' := \text{Enc}(ek, \mu')$
return $(ek, \overline{dk})$	$K := \text{H}_1(\text{H}_3(\mu), c_0, c_1)$	$c_1' := \text{H}_2(\text{H}_3(\mu'), \text{H}_4(ek))$
	return $((c_0, c_1), K)$	**if** $(c_0, c_1) \neq (c_0', c_1')$, **then return** $K := \text{H}_0(\text{H}_3(s), c_0, c_1)$
		else return $K := \text{H}_1(\text{H}_3(\mu'), c_0, c_1)$

"key ← hash(hash(m), c_0, c_1)"

Streamlined NTRU Prime's FO variant

Analysis of SNTRUp

"key ← hash(m, c_0, c_1)"

IND-CCA secure

KEM

Quantum Random Oracles (QROs)

Hash Functions H_i

{OW/IND}-CPA secure

PKE (Base)

$\overline{\mathsf{Gen}}(1^\kappa)$	$\overline{\mathsf{Enc}}(ek)$	$\overline{\mathsf{Dec}}(\overline{dk}, (c_0, c_1))$, where $\overline{dk} = (dk, ek, s)$
1 : $(ek, dk) \leftarrow \mathsf{Gen}(1^\kappa)$	1 : $\mu \leftarrow \mathcal{D}_\mathcal{M}$	1 : $\mu' \leftarrow \mathsf{Dec}(dk, c_0)$
2 : $s \leftarrow \mathcal{M}$	2 : $c_0 := \mathsf{Enc}(ek, \mu)$	2 : **if** $\mu' = \perp$ or $c_0 \neq \mathsf{Enc}(ek, \mu')$ or $c_1 \neq \mathsf{F}(\mu'[, ek])$
3 : $\overline{dk} := (dk, ek, s)$	3 : $c_1 := \mathsf{F}(\mu[, ek])$	3 : **then return** $K := \mathsf{H}(s, c_0, c_1)$
4 : **return** $(ek, \overline{dk})$	4 : $K := \mathsf{H}(\mu, c_0, c_1)$	4 : **else return** $K := \mathsf{H}(\mu', c_0, c_1)$
	5 : **return** $((c_0, c_1), K)$	

"key ← hash(m, c_0, c_1)" and "key ← hash(hash(m), c_0, c_1)" are quantum **indifferentiable**. [Zhandry@Crypto'19]

$\mathsf{HU}^{\not\perp}$ [Xagawa@Eurocrypt'22]

$\overline{\mathsf{Gen}}(1^\kappa)$	$\overline{\mathsf{Enc}}(ek)$	$\overline{\mathsf{Dec}}(\overline{dk}, (c_0, c_1))$, where $\overline{dk} = (dk, ek, s)$
$(ek, dk) \leftarrow \mathsf{Gen}(1^\kappa)$	$\mu \leftarrow \mathcal{M}$	$\mu' := \mathsf{Dec}(dk, c_0)$
$s \leftarrow \{0,1\}^\ell$	$c_0 := \mathsf{Enc}(ek, \mu)$	**if** $\mu' = \perp$, **then return** $K := \mathsf{H}_0(\mathsf{H}_3(s), c_0, c_1)$
$\overline{dk} := (dk, ek, s)$	$c_1 := \mathsf{H}_2(\mathsf{H}_3(\mu), \mathsf{H}_4(ek))$	$c_0' := \mathsf{Enc}(ek, \mu')$
return $(ek, \overline{dk})$	$K := \mathsf{H}_1(\mathsf{H}_3(\mu), c_0, c_1)$	$c_1' := \mathsf{H}_2(\mathsf{H}_3(\mu'), \mathsf{H}_4(ek))$
	return $((c_0, c_1), K)$	**if** $(c_0, c_1) \neq (c_0', c_1')$, **then return** $K := \mathsf{H}_0(\mathsf{H}_3(s), c_0, c_1)$
		else return $K := \mathsf{H}_1(\mathsf{H}_3(\mu'), c_0, c_1)$

"key ← hash(hash(m), c_0, c_1)"

Streamlined NTRU Prime's FO variant

Analysis of SNTRUp

"$c_1 \leftarrow \mathrm{hash}(m)$"

"$\mathrm{key} \leftarrow \mathrm{hash}(m, c_0, c_1)$"

IND-CCA secure

KEM

"$c_1 \leftarrow \mathrm{hash}(\mathrm{hash}(m))$"

Quantum Random Oracles (QROs)

Hash Functions H_i

{OW/IND}-CPA secure

PKE (Base)

$\overline{\mathsf{Gen}}(1^\kappa)$	$\overline{\mathsf{Enc}}(ek)$	$\overline{\mathsf{Dec}}(\overline{dk}, (c_0, c_1))$, where $\overline{dk} = (dk, ek, s)$
1 : $(ek, dk) \leftarrow \mathsf{Gen}(1^\kappa)$	1 : $\mu \leftarrow \mathcal{D}_\mathcal{M}$	1 : $\mu' \leftarrow \mathsf{Dec}(dk, c_0)$
2 : $s \leftarrow \mathcal{M}$	2 : $c_0 := \mathsf{Enc}(ek, \mu)$	2 : if $\mu' = \perp$ or $c_0 \neq \mathsf{Enc}(ek, \mu')$ or $c_1 \neq \mathsf{F}(\mu'[, ek])$
3 : $\overline{dk} := (dk, ek, s)$	3 : $c_1 := \mathsf{F}(\mu[, ek])$	3 : then return $K := \mathsf{H}(s, c_0, c_1)$
4 : return $(ek, \overline{dk})$	4 : $K := \mathsf{H}(\mu, c_0, c_1)$	4 : else return $K := \mathsf{H}(\mu', c_0, c_1)$
	5 : return $((c_0, c_1), K)$	

"$\mathrm{key} \leftarrow \mathrm{hash}(m, c_0, c_1)$" and "$\mathrm{key} \leftarrow \mathrm{hash}(\mathrm{hash}(m), c_0, c_1)$" are quantum **indifferentiable**. [Zhandry@Crypto'19]

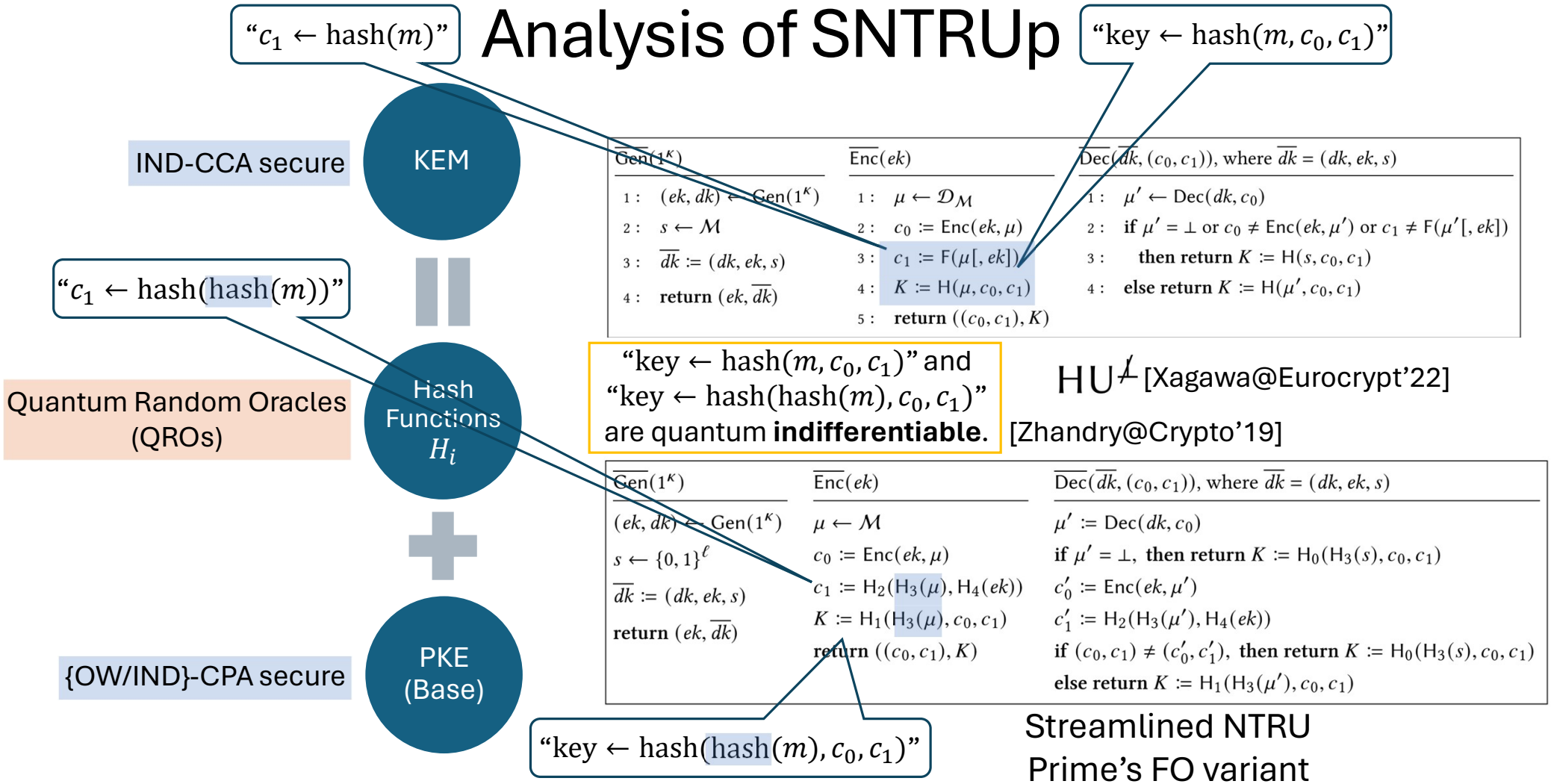
$\mathrm{HU}^{\not\perp}$ [Xagawa@Eurocrypt'22]

$\overline{\mathsf{Gen}}(1^\kappa)$	$\overline{\mathsf{Enc}}(ek)$	$\overline{\mathsf{Dec}}(\overline{dk}, (c_0, c_1))$, where $\overline{dk} = (dk, ek, s)$
$(ek, dk) \leftarrow \mathsf{Gen}(1^\kappa)$	$\mu \leftarrow \mathcal{M}$	$\mu' := \mathsf{Dec}(dk, c_0)$
$s \leftarrow \{0,1\}^\ell$	$c_0 := \mathsf{Enc}(ek, \mu)$	if $\mu' = \perp$, then return $K := \mathsf{H}_0(\mathsf{H}_3(s), c_0, c_1)$
$\overline{dk} := (dk, ek, s)$	$c_1 := \mathsf{H}_2(\mathsf{H}_3(\mu), \mathsf{H}_4(ek))$	$c_0' := \mathsf{Enc}(ek, \mu')$
return $(ek, \overline{dk})$	$K := \mathsf{H}_1(\mathsf{H}_3(\mu), c_0, c_1)$	$c_1' := \mathsf{H}_2(\mathsf{H}_3(\mu'), \mathsf{H}_4(ek))$
	return $((c_0, c_1), K)$	if $(c_0, c_1) \neq (c_0', c_1')$, then return $K := \mathsf{H}_0(\mathsf{H}_3(s), c_0, c_1)$
		else return $K := \mathsf{H}_1(\mathsf{H}_3(\mu'), c_0, c_1)$

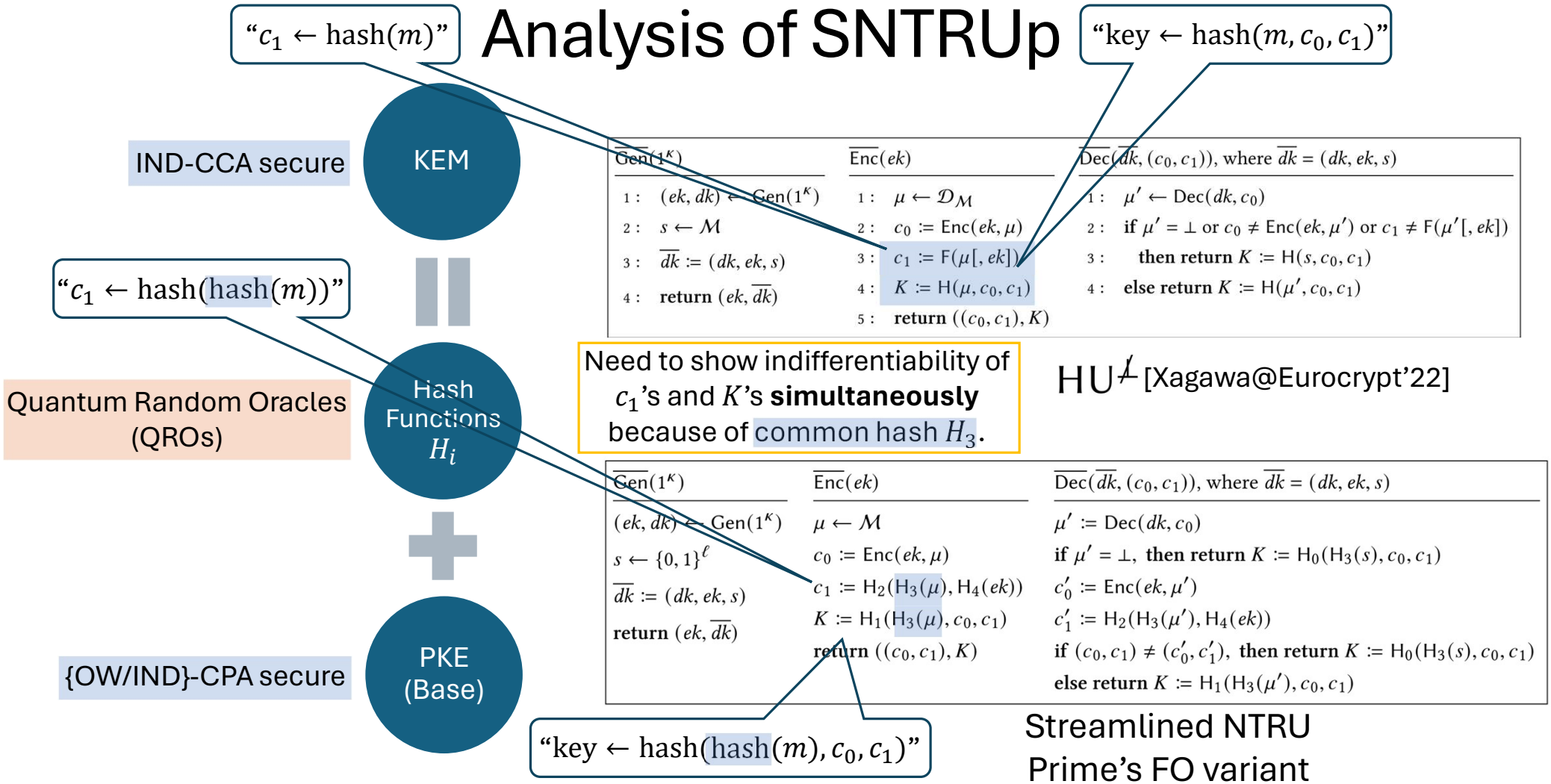
"$\mathrm{key} \leftarrow \mathrm{hash}(\mathrm{hash}(m), c_0, c_1)$"

Streamlined NTRU Prime's FO variant

Analysis of SNTRUp



Analysis of SNTRUp



$\overline{\mathsf{Gen}}(1^\kappa)$

1 : $(ek, dk) \leftarrow \mathsf{Gen}(1^\kappa)$
2 : $s \leftarrow \mathcal{M}$
3 : $\overline{dk} := (dk, ek, s)$
4 : **return** $(ek, \overline{dk})$

$\overline{\mathsf{Enc}}(ek)$

1 : $\mu \leftarrow \mathcal{D}_{\mathcal{M}}$
2 : $c_0 := \mathsf{Enc}(ek, \mu)$
3 : $c_1 := \mathsf{F}(\mu[, ek])$
4 : $K := \mathsf{H}(\mu, c_0, c_1)$
5 : **return** $((c_0, c_1), K)$

$\overline{\mathsf{Dec}}(\overline{dk}, (c_0, c_1))$, where $\overline{dk} = (dk, ek, s)$

1 : $\mu' \leftarrow \mathsf{Dec}(dk, c_0)$
2 : **if** $\mu' = \perp$ or $c_0 \neq \mathsf{Enc}(ek, \mu')$ or $c_1 \neq \mathsf{F}(\mu'[, ek])$
3 : **then return** $K := \mathsf{H}(s, c_0, c_1)$
4 : **else return** $K := \mathsf{H}(\mu', c_0, c_1)$

$\overline{\mathsf{Gen}}(1^\kappa)$

$(ek, dk) \leftarrow \mathsf{Gen}(1^\kappa)$
$s \leftarrow \{0,1\}^\ell$
$\overline{dk} := (dk, ek, s)$
return $(ek, \overline{dk})$

$\overline{\mathsf{Enc}}(ek)$

$\mu \leftarrow \mathcal{M}$
$c_0 := \mathsf{Enc}(ek, \mu)$
$c_1 := \mathsf{H}_2(\mathsf{H}_3(\mu), \mathsf{H}_4(ek))$
$K := \mathsf{H}_1(\mathsf{H}_3(\mu), c_0, c_1)$
return $((c_0, c_1), K)$

$\overline{\mathsf{Dec}}(\overline{dk}, (c_0, c_1))$, where $\overline{dk} = (dk, ek, s)$

$\mu' := \mathsf{Dec}(dk, c_0)$
if $\mu' = \perp$, **then return** $K := \mathsf{H}_0(\mathsf{H}_3(s), c_0, c_1)$
$c_0' := \mathsf{Enc}(ek, \mu')$
$c_1' := \mathsf{H}_2(\mathsf{H}_3(\mu'), \mathsf{H}_4(ek))$
if $(c_0, c_1) \neq (c_0', c_1')$, **then return** $K := \mathsf{H}_0(\mathsf{H}_3(s), c_0, c_1)$
else return $K := \mathsf{H}_1(\mathsf{H}_3(\mu'), c_0, c_1)$

Analysis of SNTRUp

"$c_1 \leftarrow \mathrm{hash}(m)$"

"key $\leftarrow \mathrm{hash}(m, c_0, c_1)$"

IND-CCA secure

KEM

$\overline{\mathsf{Gen}}(1^\kappa)$
1 : $(ek, dk) \leftarrow \mathsf{Gen}(1^\kappa)$
2 : $s \leftarrow \mathcal{M}$
3 : $\overline{dk} := (dk, ek, s)$
4 : return $(ek, \overline{dk})$

$\overline{\mathsf{Enc}}(ek)$
1 : $\mu \leftarrow \mathcal{D}_{\mathcal{M}}$
2 : $c_0 := \mathsf{Enc}(ek, \mu)$
3 : $c_1 := \mathsf{F}(\mu[, ek])$
4 : $K := \mathsf{H}(\mu, c_0, c_1)$
5 : return $((c_0, c_1), K)$

$\overline{\mathsf{Dec}}(\overline{dk}, (c_0, c_1))$, where $\overline{dk} = (dk, ek, s)$
1 : $\mu' \leftarrow \mathsf{Dec}(dk, c_0)$
2 : if $\mu' = \perp$ or $c_0 \neq \mathsf{Enc}(ek, \mu')$ or $c_1 \neq \mathsf{F}(\mu'[, ek])$
3 : then return $K := \mathsf{H}(s, c_0, c_1)$
4 : else return $K := \mathsf{H}(\mu', c_0, c_1)$

"$c_1 \leftarrow \mathrm{hash}(\mathrm{hash}(m))$"

Quantum Random Oracles (QROs)

Hash Functions H_i

Need to show indifferentiability of c_1's and K's **simultaneously** because of common hash H_3.

$\mathrm{HU}^{\not\perp}$ [Xagawa@Eurocrypt'22]

$\overline{\mathsf{Gen}}(1^\kappa)$
$(ek, dk) \leftarrow \mathsf{Gen}(1^\kappa)$
$s \leftarrow \{0,1\}^\ell$
$\overline{dk} := (dk, ek, s)$
return $(ek, \overline{dk})$

$\overline{\mathsf{Enc}}(ek)$
$\mu \leftarrow \mathcal{M}$
$c_0 := \mathsf{Enc}(ek, \mu)$
$c_1 := \mathsf{H}_2(\mathsf{H}_3(\mu), \mathsf{H}_4(ek))$
$K := \mathsf{H}_1(\mathsf{H}_3(\mu), c_0, c_1)$
return $((c_0, c_1), K)$

$\overline{\mathsf{Dec}}(\overline{dk}, (c_0, c_1))$, where $\overline{dk} = (dk, ek, s)$
$\mu' := \mathsf{Dec}(dk, c_0)$
if $\mu' = \perp$, then return $K := \mathsf{H}_0(\mathsf{H}_3(s), c_0, c_1)$
$c_0' := \mathsf{Enc}(ek, \mu')$
$c_1' := \mathsf{H}_2(\mathsf{H}_3(\mu'), \mathsf{H}_4(ek))$
if $(c_0, c_1) \neq (c_0', c_1')$, then return $K := \mathsf{H}_0(\mathsf{H}_3(s), c_0, c_1)$
else return $K := \mathsf{H}_1(\mathsf{H}_3(\mu'), c_0, c_1)$

Streamlined NTRU Prime's FO variant

PKE (Base)

{OW/IND}-CPA secure

"key $\leftarrow \mathrm{hash}(\mathrm{hash}(m), c_0, c_1)$"

Analysis of SNTRUp

"$c_1 \leftarrow \mathrm{hash}(m)$"

"$\mathrm{key} \leftarrow \mathrm{hash}(m, c_0, c_1)$"

IND-CCA secure

KEM

"$c_1 \leftarrow \mathrm{hash}(\mathrm{hash}(m))$"

Quantum Random Oracles (QROs)

Hash Functions H_i

{OW/IND}-CPA secure

PKE (Base)

"$\mathrm{key} \leftarrow \mathrm{hash}(\mathrm{hash}(m), c_0, c_1)$"

$\overline{\mathsf{Gen}}(1^\kappa)$	$\overline{\mathsf{Enc}}(ek)$	$\overline{\mathsf{Dec}}(\overline{dk}, (c_0, c_1))$, where $\overline{dk} = (dk, ek, s)$
1 : $(ek, dk) \leftarrow \mathsf{Gen}(1^\kappa)$	1 : $\mu \leftarrow \mathcal{D}_{\mathcal{M}}$	1 : $\mu' \leftarrow \mathsf{Dec}(dk, c_0)$
2 : $s \leftarrow \mathcal{M}$	2 : $c_0 := \mathsf{Enc}(ek, \mu)$	2 : **if** $\mu' = \perp$ or $c_0 \neq \mathsf{Enc}(ek, \mu')$ or $c_1 \neq \mathsf{F}(\mu'[, ek])$
3 : $\overline{dk} := (dk, ek, s)$	3 : $c_1 := \mathsf{F}(\mu[, ek])$	3 : **then return** $K := \mathsf{H}(s, c_0, c_1)$
4 : **return** $(ek, \overline{dk})$	4 : $K := \mathsf{H}(\mu, c_0, c_1)$	4 : **else return** $K := \mathsf{H}(\mu', c_0, c_1)$
	5 : **return** $((c_0, c_1), K)$	

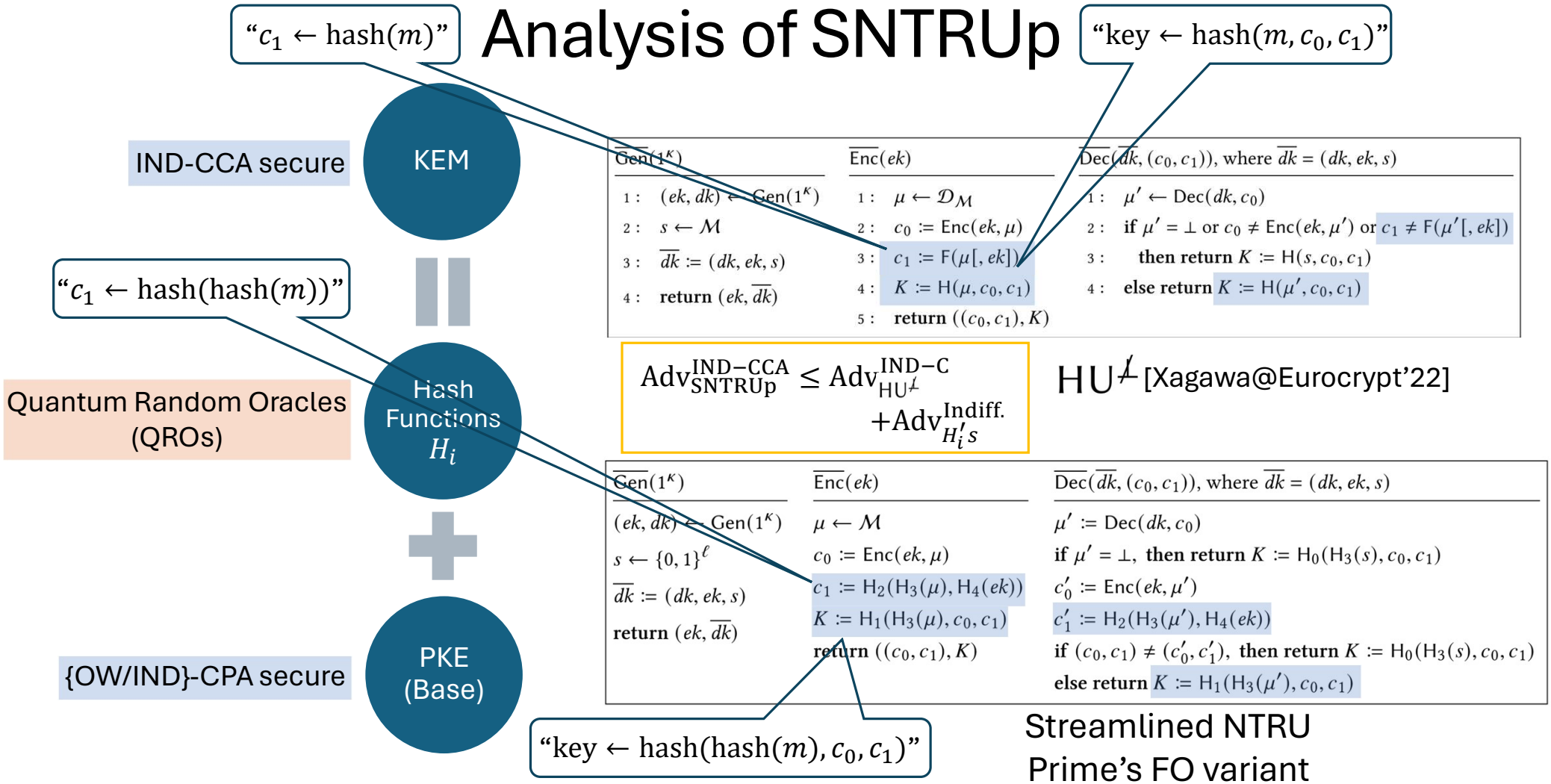
Need to show indifferentiability of c_1's and K's **simultaneously** because of common hash H_3.

$\mathrm{HU}^{\not\perp}$ [Xagawa@Eurocrypt'22]

$\overline{\mathsf{Gen}}(1^\kappa)$	$\overline{\mathsf{Enc}}(ek)$	$\overline{\mathsf{Dec}}(\overline{dk}, (c_0, c_1))$, where $\overline{dk} = (dk, ek, s)$
$(ek, dk) \leftarrow \mathsf{Gen}(1^\kappa)$	$\mu \leftarrow \mathcal{M}$	$\mu' := \mathsf{Dec}(dk, c_0)$
$s \leftarrow \{0,1\}^\ell$	$c_0 := \mathsf{Enc}(ek, \mu)$	**if** $\mu' = \perp$, **then return** $K := \mathsf{H}_0(\mathsf{H}_3(s), c_0, c_1)$
$\overline{dk} := (dk, ek, s)$	$c_1 := \mathsf{H}_2(\mathsf{H}_3(\mu), \mathsf{H}_4(ek))$	$c_0' := \mathsf{Enc}(ek, \mu')$
return $(ek, \overline{dk})$	$K := \mathsf{H}_1(\mathsf{H}_3(\mu), c_0, c_1)$	$c_1' := \mathsf{H}_2(\mathsf{H}_3(\mu'), \mathsf{H}_4(ek))$
	return $((c_0, c_1), K)$	**if** $(c_0, c_1) \neq (c_0', c_1')$, **then return** $K := \mathsf{H}_0(\mathsf{H}_3(s), c_0, c_1)$
		else return $K := \mathsf{H}_1(\mathsf{H}_3(\mu'), c_0, c_1)$

Streamlined NTRU Prime's FO variant

Analysis of SNTRUp



$\overline{\text{Gen}}(1^\kappa)$

1 : $(ek, dk) \leftarrow \text{Gen}(1^\kappa)$
2 : $s \leftarrow \mathcal{M}$
3 : $\overline{dk} := (dk, ek, s)$
4 : return $(ek, \overline{dk})$

$\overline{\text{Enc}}(ek)$

1 : $\mu \leftarrow \mathcal{D}_{\mathcal{M}}$
2 : $c_0 := \text{Enc}(ek, \mu)$
3 : $c_1 := \text{F}(\mu[, ek])$
4 : $K := \text{H}(\mu, c_0, c_1)$
5 : return $((c_0, c_1), K)$

$\overline{\text{Dec}}(\overline{dk}, (c_0, c_1))$, where $\overline{dk} = (dk, ek, s)$

1 : $\mu' \leftarrow \text{Dec}(dk, c_0)$
2 : if $\mu' = \perp$ or $c_0 \neq \text{Enc}(ek, \mu')$ or $c_1 \neq \text{F}(\mu'[, ek])$
3 : then return $K := \text{H}(s, c_0, c_1)$
4 : else return $K := \text{H}(\mu', c_0, c_1)$

$\overline{\text{Gen}}(1^\kappa)$

$(ek, dk) \leftarrow \text{Gen}(1^\kappa)$
$s \leftarrow \{0,1\}^\ell$
$\overline{dk} := (dk, ek, s)$
return $(ek, \overline{dk})$

$\overline{\text{Enc}}(ek)$

$\mu \leftarrow \mathcal{M}$
$c_0 := \text{Enc}(ek, \mu)$
$c_1 := \text{H}_2(\text{H}_3(\mu), \text{H}_4(ek))$
$K := \text{H}_1(\text{H}_3(\mu), c_0, c_1)$
return $((c_0, c_1), K)$

$\overline{\text{Dec}}(\overline{dk}, (c_0, c_1))$, where $\overline{dk} = (dk, ek, s)$

$\mu' := \text{Dec}(dk, c_0)$
if $\mu' = \perp$, then return $K := \text{H}_0(\text{H}_3(s), c_0, c_1)$
$c_0' := \text{Enc}(ek, \mu')$
$c_1' := \text{H}_2(\text{H}_3(\mu'), \text{H}_4(ek))$
if $(c_0, c_1) \neq (c_0', c_1')$, then return $K := \text{H}_0(\text{H}_3(s), c_0, c_1)$
else return $K := \text{H}_1(\text{H}_3(\mu'), c_0, c_1)$

Fujisaki-Okamoto transform.

Don’t Roll Your Own FO:

Challenges in Proving Post-Quantum CCA Security of Real-World KEMs

- CRYSTALS-Kyber
- Streamlined NTRU Prime

Varun Maram

Cybersecurity Group

SandboxAQ

In the Quantum Random Oracle Model (QROM).

Based on joint works with Paul Grubbs, Kenny Paterson, and Keita Xagawa.

Don't ~~Roll Your Own~~ Need FO(?):

Challenges in Proving Post-Quantum CCA Security of Real-World KEMs

Fujisaki-**O**kamoto transform.

- CRYSTALS-Kyber
- Streamlined NTRU Prime

Varun Maram

Cybersecurity Group

SandboxAQ

In the **Q**uantum **R**andom **O**racle **M**odel (QROM).

Based on joint works with Paul Grubbs, Kenny Paterson, and Keita Xagawa.

Post-quantum SSH

NTRU Prime

Intro | Warnings | Papers | Security | Softw
Speed | NISTPQC | FAQ

NTRU Prime is a third-round candidate in NIST's Post-Quantur Cryptography Standardization Project. The third-round submiss specifies twelve KEMs:

- Streamlined NTRU Prime: `sntrup653`, `sntrup761`, `sntr` `sntrup953`, `sntrup1013`, `sntrup1277`.
- NTRU LPRime: `ntrulpr653`, `ntrulpr761`, `ntrulpr857`, `ntrulpr1013`, `ntrulpr1277`.

OpenSSH 9.0/9.0p1 (2022-04-08)

OpenSSH 9.0 was released on 2022-04-08. It is available from the mirrors listed at https://www.openssh.com/.
OpenSSH is a 100% complete SSH protocol 2.0 implementation and includes sftp client and server support.

Once again, we would like continued support of the code or patches, reported project. More information https://www.openssh.com/d

Changes since OpenSSH 8.9
==========

This release is focused o

Potentially-incompatible

This release switches scp to using the SFTP protoco

Legacy scp/rcp performs w "scp host:* .") through t requiring double quoting included on scp(1) comman as shell commands on the

This creates one area of potential incompatibility: scp(1) when using the SFTP protocol no longer requires this finicky and brittle quoting, and attempts to use it may cause transfers to fail. We consider the removal of the need for double-quoting shell characters in file names to be a benefit and do not intend to introduce bug-compatibility for legacy scp/rcp in scp(1) when using the SFTP protocol.

Another area of potential incompatibility relates to the use of remote paths relative to other user's home directories, for example - "scp host:~user/file /tmp". The SFTP protocol has no native way to expand a ~user path. However, sftp-server(8) in OpenSSH 8.7 and later support a protocol extension "expand-path@openssh.com" to support this.

In case of incompatibility, the scp(1) client may be instructed to use the legacy scp/rcp using the -O flag.

New features
---

* ssh(1), sshd(8): use the hybrid Streamlined NTRU Prime + x25519 key exchange method by default ("sntrup761x25519-sha512@openssh.com"). The NTRU algorithm is believed to resist attacks enabled by future quantum computers and is paired with the X25519 ECDH key exchange (the previous default) as a backstop against any weaknesses in NTRU Prime that may be discovered in the future. The combination ensures that the hybrid exchange offers at least as good security as the status quo.

Post-quantum SSH

7.4 Final result: Multi-ciphersuite SSH

Combining Lemmas 5 and 6 from the previous subsection with the composition theorem (Theorem 3) immediately yields that the SSH protocol is multi-ciphersuite secure, even with key re-use across ciphersuites.

Corollary 1 (SSH is multi-ciphersuite secure)**.** *Let* $\vec{\mathtt{SSH}}$ *be the multi-ciphersuite SSH protocol with each of the* $n_{\mathtt{SP}}$ *ciphersuites* $\mathtt{SSH}_c$ *being a signed-Diffie–Hellman ciphersuite as in Section 4. The algorithms* $\mathcal{B}_1, \ldots, \mathcal{B}_5$ *inferred from the proof are such that, for all algorithms* $\mathcal{A}$*:*

$$\mathrm{Adv}^{\mathsf{mcs\text{-}acce\text{-}so\text{-}auth}}_{\vec{\mathtt{SSH}},c}(\mathcal{A}) \leq n_{\mathtt{SP}} \left(\frac{(n_P n_S)^2}{2^\mu} + \mathrm{Adv}^{\mathsf{cr}}_{\mathrm{H}_c}(\mathcal{B}_1^{\mathcal{A}}) + n_P \mathrm{Adv}^{\mathsf{suf\text{-}cma}}_{\mathtt{SIG}_c}(\mathcal{B}_2^{\mathcal{A}}) \right)$$

and

$$\begin{aligned} \mathrm{Adv}^{\mathsf{mcs\text{-}acce\text{-}so\text{-}aenc}}_{\vec{\mathtt{SSH}},c}(\mathcal{A}) \leq\ & \mathrm{Adv}^{\mathsf{mcs\text{-}acce\text{-}so\text{-}auth}}_{\vec{\mathtt{SSH}},c}(\mathcal{A}) \\ & + n_P n_S \left(\mathrm{Adv}^{\mathsf{ddh}}_{g_c,q_q}(\mathcal{B}_3^{\mathcal{A}}) + \mathrm{Adv}^{\mathsf{prf}}_{\mathrm{PRF}_c}(\mathcal{B}_4^{\mathcal{A}}) + \mathrm{Adv}^{\mathsf{bsae}}_{\mathtt{StE}_c}(\mathcal{B}_5^{\mathcal{A}}) \right) \end{aligned}$$

and $\mathcal{B}_1^{\mathcal{A}}, \ldots, \mathcal{B}_5^{\mathcal{A}}$ *have approximately the same running time as* $\mathcal{A}$*. Moreover, analogous versions of the theorem apply for mutual authentication.*

"Pre-quantum" analysis by [Bergsma-Dowling-Kohlar-Schwenk-Stebila@CCS'14]

Post-quantum SSH

7.4 Final result: Multi-ciphersuite SSH

Combining Lemmas 5 and 6 from the previous subsection with the composition theorem (Theorem 3) immediately yields that the SSH protocol is multi-ciphersuite secure, even with key re-use across ciphersuites.

Corollary 1 (SSH is multi-ciphersuite secure)**.** *Let* $\vec{\mathtt{SSH}}$ *be the multi-ciphersuite SSH protocol with each of the* $n_{\mathtt{SP}}$ *ciphersuites* $\mathtt{SSH}_c$ *being a signed-Diffie–Hellman ciphersuite as in Section 4. The algorithms* $\mathcal{B}_1, \ldots, \mathcal{B}_5$ *inferred from the proof are such that, for all algorithms* $\mathcal{A}$*:*

$$\mathrm{Adv}^{\mathsf{mcs\text{-}acce\text{-}so\text{-}auth}}_{\vec{\mathtt{SSH}},c}(\mathcal{A}) \leq n_{\mathtt{SP}} \left(\frac{(n_P n_S)^2}{2^\mu} + \mathrm{Adv}^{\mathsf{cr}}_{\mathrm{H}_c}(\mathcal{B}_1^{\mathcal{A}}) + n_P \mathrm{Adv}^{\mathsf{suf\text{-}cma}}_{\mathtt{SIG}_c}(\mathcal{B}_2^{\mathcal{A}}) \right)$$

and

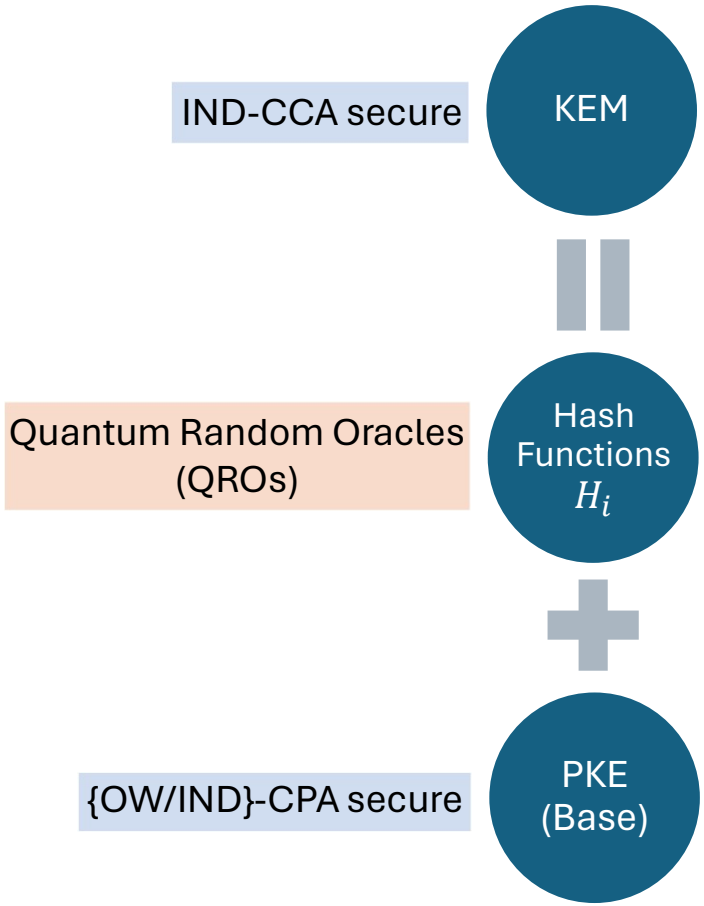
$$\mathrm{Adv}^{\mathsf{mcs\text{-}acce\text{-}so\text{-}aenc}}_{\vec{\mathtt{SSH}},c}(\mathcal{A}) \leq \mathrm{Adv}^{\mathsf{mcs\text{-}acce\text{-}so\text{-}auth}}_{\vec{\mathtt{SSH}},c}(\mathcal{A}) + n_P n_S \left(\mathrm{Adv}^{\mathsf{ddh}}_{g_c,q_q}(\mathcal{B}_3^{\mathcal{A}}) + \mathrm{Adv}^{\mathsf{prf}}_{\mathrm{PRF}_c}(\mathcal{B}_4^{\mathcal{A}}) + \mathrm{Adv}^{\mathsf{bsae}}_{\mathtt{StE}_c}(\mathcal{B}_5^{\mathcal{A}}) \right)$$

CPA security of (hybrid) KEM should suffice.

and $\mathcal{B}_1^{\mathcal{A}}, \ldots, \mathcal{B}_5^{\mathcal{A}}$ *have approximately the same running time as* $\mathcal{A}$*. Moreover, analogous versions of the theorem apply for mutual authentication.*

"Pre-quantum" analysis by [Bergsma-Dowling-Kohlar-Schwenk-Stebila@CCS'14]

FO Variants



"key ← hash(hash(m), c, hash(hash(m)))"

$\overline{\mathsf{Gen}}(1^\kappa)$	$\overline{\mathsf{Enc}}(\overline{ek})$	$\overline{\mathsf{Dec}}(\overline{dk}, (c_0, c_1))$, where $\overline{dk} = (dk, ek, s)$
$(ek, dk) \leftarrow \mathsf{Gen}(1^\kappa)$	$\mu \leftarrow \mathcal{M}$	$\mu' := \mathsf{Dec}(dk, c_0)$
$s \leftarrow \{0,1\}^\ell$	$c_0 := \mathsf{Enc}(ek, \mu)$	**if** $\mu' = \perp$, **then return** $K := \mathsf{H}_0(\mathsf{H}_3(s), c_0, c_1)$
$\overline{dk} := (dk, ek, s)$	$c_1 := \mathsf{H}_2(\mathsf{H}_3(\mu), \mathsf{H}_4(ek))$	$c_0' := \mathsf{Enc}(ek, \mu')$
return $(ek, \overline{dk})$	$K := \mathsf{H}_1(\mathsf{H}_3(\mu), c_0, c_1)$	$c_1' := \mathsf{H}_2(\mathsf{H}_3(\mu'), \mathsf{H}_4(ek))$
	return $((c_0, c_1), K)$	**if** $(c_0, c_1) \neq (c_0', c_1')$, **then return** $K := \mathsf{H}_0(\mathsf{H}_3(s), c_0, c_1)$
		else return $K := \mathsf{H}_1(\mathsf{H}_3(\mu'), c_0, c_1)$

Streamlined NTRU Prime's FO variant

FO Variants

IND-CCA secure

KEM

If target is **IND-CPA security**, then there is a much simpler and more efficient (hybrid) KEM construction – i.e., **no need for FO**!

Quantum Random Oracles (QROs)

Hash Functions H_i

"key ← hash(hash(m), c, hash(hash(m)))"

{OW/IND}-CPA secure

PKE (Base)

$\overline{\mathsf{Gen}}(1^\kappa)$	$\overline{\mathsf{Enc}}(ek)$	$\overline{\mathsf{Dec}}(\overline{dk}, (c_0, c_1))$, where $\overline{dk} = (dk, ek, s)$
$(ek, dk) \leftarrow \mathsf{Gen}(1^\kappa)$	$\mu \leftarrow \mathcal{M}$	$\mu' := \mathsf{Dec}(dk, c_0)$
$s \leftarrow \{0,1\}^\ell$	$c_0 := \mathsf{Enc}(ek, \mu)$	**if** $\mu' = \perp$, **then return** $K := \mathsf{H}_0(\mathsf{H}_3(s), c_0, c_1)$
$\overline{dk} := (dk, ek, s)$	$c_1 := \mathsf{H}_2(\mathsf{H}_3(\mu), \mathsf{H}_4(ek))$	$c_0' := \mathsf{Enc}(ek, \mu')$
return $(ek, \overline{dk})$	$K := \mathsf{H}_1(\mathsf{H}_3(\mu), c_0, c_1)$	$c_1' := \mathsf{H}_2(\mathsf{H}_3(\mu'), \mathsf{H}_4(ek))$
	return $((c_0, c_1), K)$	**if** $(c_0, c_1) \neq (c_0', c_1')$, **then return** $K := \mathsf{H}_0(\mathsf{H}_3(s), c_0, c_1)$
		else return $K := \mathsf{H}_1(\mathsf{H}_3(\mu'), c_0, c_1)$

Streamlined NTRU Prime's FO variant

FO Variants

IND-CCA secure

KEM

If target is **IND-CPA security**, then there is a much simpler and more efficient (hybrid) KEM construction – i.e., **no need for FO**!

7. Security Considerations

The shared secrets computed in the hybrid key exchange should be computed in a way that achieves the "hybrid" property: the resulting secret is secure as long as at least one of the component key exchange algorithms is unbroken. See [GIACON] and [BINDEL] for an investigation of these issues. Under the assumption that shared secrets are fixed length once the combination is fixed, the construction from Section 3.3 corresponds to the dual-PRF combiner of [BINDEL] which is shown to preserve security under the assumption that the hash function is a dual-PRF.

As noted in Section 2, KEMs used in the manner described in this document MUST explicitly be designed to be secure in the event that the public key is reused, such as achieving IND-CCA2 security or having a transform like the Fujisaki-Okamoto transform applied. Kyber has such security properties. However, some other post-quantum KEMs are designed to be IND-CPA-secure (i.e., without countermeasures such as the FO transform) are completely insecure under public key reuse; for example, some lattice-based IND-CPA-secure KEMs are vulnerable to attacks that recover the private key after just a few thousand samples [FLUHRER].

Datatracker

draft-ietf-tls-hybrid-design-09

Internet-Draft

Info | Contents | Prefs

Document type

This is an older version of an Internet-Draft whose latest revision state is "Active".

Expired & archived

Select version

00 01 02 03 04 05 06 07 08 09 10

Compare versions

Doesn't defend against **key-reuse attacks** in bad SSH implementations, however...

$\overline{\mathrm{Dec}}(\overline{dk}, (c_0, c_1))$, where $\overline{dk} = (dk, ek, s)$

$\mu' := \mathrm{Dec}(dk, c_0)$

if $\mu' = \perp$, **then return** $K := \mathrm{H}_0(\mathrm{H}_3(s), c_0, c_1)$

$c_0' := \mathrm{Enc}(ek, \mu')$

$c_1' := \mathrm{H}_2(\mathrm{H}_3(\mu'), \mathrm{H}_4(ek))$

if $(c_0, c_1) \neq (c_0', c_1')$, **then return** $K := \mathrm{H}_0(\mathrm{H}_3(s), c_0, c_1)$

else return $K := \mathrm{H}_1(\mathrm{H}_3(\mu'), c_0, c_1)$

Streamlined NTRU Prime's FO variant

FO Variants

Is there a notion **weaker than IND-CCA2** which captures key-reuse attacks while allowing for more efficient schemes?

IND-CCA secure

KEM

If target is **IND-CPA security**, then there is a much simpler and more efficient (hybrid) KEM construction – i.e., **no need for FO**!

7. Security Considerations

The shared secrets computed in the hybrid key exchange should be computed in a way that achieves the "hybrid" property: the resulting secret is secure as long as at least one of the component key exchange algorithms is unbroken. See [GIACON] and [BINDEL] for an investigation of these issues. Under the assumption that shared secrets are fixed length once the combination is fixed, the construction from Section 3.3 corresponds to the dual-PRF combiner of [BINDEL] which is shown to preserve security under the assumption that the hash function is a dual-PRF.

As noted in Section 2, KEMs used in the manner described in this document MUST explicitly be designed to be secure in the event that the public key is reused, such as achieving IND-CCA2 security or having a transform like the Fujisaki-Okamoto transform applied. Kyber has such security properties. However, some other post-quantum KEMs are designed to be IND-CPA-secure (i.e., without countermeasures such as the FO transform) are completely insecure under public key reuse; for example, some lattice-based IND-CPA-secure KEMs are vulnerable to attacks that recover the private key after just a few thousand samples [FLUHRER].

Datatracker

draft-ietf-tls-hybrid-design-09

Internet-Draft

Info | Contents | Prefs

Document type

This is an older version of an Internet-Draft whose latest revision state is "Active".

Expired & archived

Select version

00 01 02 03 04 05 06 07 08 09 10

Compare versions

Doesn't defend against **key-reuse attacks** in bad SSH implementations, however...

$\overline{\mathrm{Dec}}(\overline{dk}, (c_0, c_1))$, where $\overline{dk} = (dk, ek, s)$

$\mu' := \mathrm{Dec}(dk, c_0)$

if $\mu' = \perp$, **then return** $K := \mathrm{H}_0(\mathrm{H}_3(s), c_0, c_1)$

$c_0' := \mathrm{Enc}(ek, \mu')$

$c_1' := \mathrm{H}_2(\mathrm{H}_3(\mu'), \mathrm{H}_4(ek))$

if $(c_0, c_1) \neq (c_0', c_1')$, **then return** $K := \mathrm{H}_0(\mathrm{H}_3(s), c_0, c_1)$

else return $K := \mathrm{H}_1(\mathrm{H}_3(\mu'), c_0, c_1)$

Streamlined NTRU Prime's FO variant

Fujisaki-**O**kamoto transform.

Don’t Roll Your Own FO:

Challenges in Proving Post-Quantum CCA Security of Real-World KEMs

- CRYSTALS-Kyber
- Streamlined NTRU Prime

Varun Maram

Cybersecurity Group

SandboxAQ

In the **Q**uantum **R**andom **O**racle **M**odel (QROM).

Based on joint works with Paul Grubbs, Kenny Paterson, and Keita Xagawa.